UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

I.T. SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

IMPLEMENTACIÓN DE UN CODIFICADOR DE AUDIO SIN PÉRDIDAS

AUTORA: SONIA GARCÍA MARTÍN

TUTOR: JOSE MIGUEL LEIVA MURILLO

I. RESUMEN

Este Proyecto de Fin de Carrera se enmarca dentro del estudio de la codificación y compresión de las señales de audio.

La compresión de audio ha sido, desde hace unos años, tema de investigación prioritaria. Las razones son evidentes, la digitalización de señales de audio da lugar a unos tamaños de archivos muy grandes que provocan tasas de transmisión muy elevadas, por tanto, es necesario acudir a la compresión. Se han desarrollado diferentes métodos de compresión para reducir el tamaño de los ficheros, la compresión sin pérdidas es la que se va a estudiar durante este proyecto.

Se han realizado una serie de experimentos con diferentes configuraciones con el objeto de determinar cuáles producen una compresión mayor de la señal de audio. Se ha partido de un codificador básico sin pérdidas, AudioPak y se han implementado diversas extensiones. Estas extensiones han sido:

- Enventanado de longitud variable: pasar del enventanado de longitud fija de AudioPak, a un enventanado de longitud variable.
- Coeficientes del predictor: variar los coeficientes enteros de la codificación lineal de AudioPak.
- Codificación entrópica:
    - Huffman: pasar de la codificación basada en Golomb-Rice que emplea AudioPak a una codificación basada en tablas de Huffman.
    - Mixta: emplear una codificación basada en ambos algoritmos Golomb-Rice o tablas de Huffman, según la necesidad.

II. ABSTRACT

The Thesis Project is part of the study of encoding and compression of audio signals.

Audio compression is a priority research topic in recent years. The reasons are obvious, the digitization of audio signals leads to large file that cause very high transmission rates, so signal compression is necessary. There have been developed different methods of compression to reduce file size, in this project, lossless compression is going to be studied.

We have done a series of experiments with different configurations in order to determine which produce a higher compression of the audio signal. This Project has been based on a basic lossless encoder AudioPak, and we have implemented several improvements. These improvements are:

- *Variable length framing: AudioPak is based on fixed frames, a variable length framing is proposed.*
- *Coefficients: Change integer coefficients in linear coding in AudioPak.*
- *Entropy coding: AudioPak is based in Golomb-Rice algorithms.*
    - *Huffman: encoding that use different tables based on Huffman algorithms.*
    - *Mixed: encoding using Huffman tables or Golomb-Rice algorithms*

III. AGRADECIMIENTOS

Quisiera aprovechar la oportunidad para agradecer a todas esas personas que me han apoyado tanto durante este tiempo, porque sin su ayuda todo habría resultado más difícil.

Agradecer a José M., tutor del proyecto, el darme la oportunidad de realizar este trabajo y por todo lo aprendido en los últimos meses. Además agradecerle toda la ayuda brindada y su tiempo dedicado durante el último año.

A toda mi familia, en especial mis padres Julio y MªCarmen, mi hermana Sara, por todo lo que han hecho para que pueda llegar a este punto.

A Sergio por estar ahí siempre regalándome una sonrisa y aguantarme en los momentos más difíciles.

Además, como olvidarse, de todos mis amigos y compañeros, tanto de fuera como de dentro de la universidad, que me apoyan y me hacen pasar tan buenos ratos.

Contenido

I. RESUMEN ..... **I**

IV. Índice de Imágenes ..... **VII**

V. Índice de Tablas ..... **VIII**

1. INTRODUCCIÓN **2**

1.1. Planteamiento del problema **2**

1.2. Objetivos **4**

1.3. Organización de la memoria **5**

2. ESTADO DEL ARTE **6**

2.1. Funcionamiento del oído **6**

2.2. Codificación con pérdidas **8**

2.2.1. Umbral absoluto de audición *8*

2.2.2. Enmascaramiento *9*

2.2.3. Bandas críticas *11*

2.2.4. Formatos de audio más comunes *14*

2.3. Codificación sin pérdidas **17**

2.3.1. Enventanado *20*

2.3.2. Decorrelación intracanal *23*

2.3.3. Codificación de entropía *29*

2.3.4. Ejemplos de algoritmos de compresión sin pérdidas *38*

2.3.5. Estudio de un algoritmo sencillo: AudioPak *40*

3. EXTENSIONES PROPUESTAS **44**

3.1. Longitud trama adaptativa **44**

3.2. Extensión en decorrelación intracanal **47**

3.3. Extensión en codificación fuente **48**

3.3.1. Codificación Huffman *48*

3.3.2. Codificación mixta: Huffman-Golomb *50*

4. EXPERIMENTACIÓN Y RESULTADOS 52

4.1. Enventanado 56

4.2. Decorrelación intracanal 66

4.3. Codificación fuente 68

4.3.1. Codificación Huffman 76

4.3.2. Codificación mixta: Huffman-Golomb 78

4.4. Extensión total 81

5. CONCLUSIONES 88

6. LÍNEAS DE TRABAJO FUTURAS 90

7. PRESUPUESTO 92

8. REFERENCIAS 94

IV. Índice de Imágenes

Figura 1. Representación del umbral auditivo 9

Figura 2. Ejemplo de enmascaramiento frecuencial 10

Figura 3. Enmascaramiento temporal 11

Figura 4: Esquema general de los compresores de audio sin pérdidas 19

Figura 5: Espectro de una señal y su promedio 24

Figura 6: Modelo de predicción 24

Figura 7: Estructura general de predicción 25

Figura 8: Estructura general de reconstrucción 26

Figura 9: Aproximación polinómica de los predictores usados en la decorrelación intracanal . 41

Figura 10: Gráfica longitud de trama 54

Figura 11: Aclaración gráfica de un tensor 58

Figura 12: Niveles de tramas para la longitud variable 60

Figura 13: Niveles y agrupaciones para longitud de trama 61

Figura 14: Gráfica de la frecuencia de los predictores lineales 66

Figura 15: Histograma, ejes logarítmicos, de los archivos de audio 69

Figura 16: Histograma de la Tabla 5 de Huffman 71

Figura 17: Histograma de la Tabla 9 de Huffman 71

Figura 18: Histograma de la Tabla 13 de Huffman 72

Figura 19: Histograma de la Tabla 16 de Huffman 72

Figura 20: Gráfica de frecuencia de uso de las tablas Huffman 73

V. Índice de Tablas

Tabla 1: Algoritmos de compresión sin pérdidas (método predictivo) 28

Tabla 2: Índice de canciones 52

Tabla 3: Número de bits empleados con la codificación AudioPak 55

Tabla 4: Matriz correspondiente a la canción 015; trama 100.000 57

Tabla 5: Matriz resultado tras calcular longitud de trama. Canción 018, tramas 5026-5030 62

Tabla 6: Número de tramas empleadas en AudioPak vs tramas empleadas con longitud variable 63

Tabla 7: Comparación AudioPak vs extensión con tramas adaptativas 64

Tabla 8: Variación de coeficientes enteros en los filtros de codificación lineal 67

Tabla 9: Rango de las tablas Huffman 70

Tabla 10: Tablas Huffman 1-4 75

Tabla 11: Resultados experimento Golomb-Rice vs Huffman 77

Tabla 12: Resultado experimento AudioPak vs codificación mixta 79

Tabla 13: Codificación mixta, tramas por Huffman vs tramas por Golomb 80

Tabla 14: Matriz general, audio 012, trama 100 82

Tabla 15: Matriz tras el codificador, audio 026, tramas 13-18 83

Tabla 16: Resultado experimento todas las extensiones vs AudioPak 84

Tabla 17: Resultado experimento final, características tramas codificadas 85

Tabla 18: Resultado experimento final, longitud de trama 86

Tabla 19: Resultado experimento final, predictor lineal. 87

1. INTRODUCCIÓN

1.1. Planteamiento del problema

Se vive en un mundo en el que el acceso a la información es cada vez más una necesidad, el envío y recepción de datos se efectúa a lugares y desde lugares en cualquier parte del mundo. Las limitaciones tecnológicas y económicas han impulsado la búsqueda de esquemas para transferir datos de una forma más eficiente y de esquemas para minimizar la cantidad de bytes utilizados para representar la información.

La información multimedia, por su propia naturaleza, demanda grandes cantidades de almacenamiento y de procesamiento. Hasta cierto punto, mantener este contenido de manera local en una estación de trabajo no será costoso, sin embargo, las transferencias de archivos de sonido o vídeo a través de Internet (o de cualquier red en general) son cada día más comunes. Por lo tanto, es necesario perfeccionar la forma en la que este contenido se almacena para así minimizar el ancho de banda necesario para completar dicha transferencia.

Por ejemplo, un segundo de sonido con calidad *Compact Disc* (CD) sin comprimir ocuparía:

44100 muestras/segundo x 16 bits/ muestra x 2 = 176,400 bytes/segundo

lo que representaría 50.46 MB para un archivo de sonido cuya duración sea de 5 minutos.

Como puede observarse, transferir esta cantidad de datos a través de una red es costoso en tiempo y, a menos que se tenga una conexión con una velocidad equivalente a 1.5Mbps, imposible de reproducir en tiempo real mientras es descargado.

Es necesario entonces, encontrar esquemas para reducir la cantidad de información necesaria para generar un contenido con la calidad que demanda la aplicación.

CODIFICACIÓN Y COMPRESIÓN DE AUDIO

La codificación es una representación de los datos y existen diferentes objetivos para los que es útil aplicarla. En este punto conviene diferenciar entre codificación fuente y codificación de canal.

Se dice que los símbolos son generados por una fuente y a la conversión o representación de estos datos se le llama codificación de fuente.

La finalidad de la codificación de canal es la detección y corrección de errores producidos en el canal de comunicación o en medios de grabación, como consecuencia del ruido y distorsión introducidos, tanto por el medio de propagación como por las no linealidades en el propio sistema de transmisión.

Otro término que conviene introducir es la compresión, se denomina compresión de datos al conjunto de técnicas que permiten que un conjunto de datos de una determinada longitud pueda ser reducido en su tamaño, sin alterar el significado de la información que contiene.

En este proyecto se va a tratar la codificación fuente y se van a intentar comprimir las señales mediante técnicas de eliminación de redundancia de la señal.

En particular, cuando se trata de compresión de audio es posible efectuarla de dos maneras:

- Compresión sin pérdida: que busca y elimina redundancias en la información, la señal reconstruida es idéntica bit-a-bit a la original.
- Compresión con pérdida: en la actualidad este esquema se enfoca a explotar las características del oído humano para que las pérdidas de la señal sean imperceptibles. Se basa en la psicoacústica.

Cada esquema tiene ventajas y desventajas propias. La compresión sin pérdida logra mejores resultados si lo que se pretende es preservar la fidelidad del origen, a costa de un sacrificio en la relación de compresión. La compresión con pérdida no siempre logra el objetivo de ser perceptualmente transparente, lo cual disminuye drásticamente la calidad (al menos desde un punto de vista subjetivo) de la señal reconstruida, aunque su gran ventaja son las altas relaciones de compresión.

En compresión, la señal puede ser codificada utilizando esquemas de tasa de transferencia constante (CBR, del inglés *Constant Bit Rate*) o variable (VBR, del inglés *Variable Bit Rate*), siendo VBR una opción más natural, pues la entropía de una señal varía en el tiempo. En el caso de CBR, se necesita la misma cantidad de bytes para almacenar un periodo de tiempo determinado; por otro lado, en VBR, la tasa se ajusta con relación a las demandas de la señal, pues diferentes partes requieren más información que otras, por ejemplo, un silencio prolongado requerirá muy pocos bits.

La decisión de utilizar CBR o VBR depende del objetivo de la codificación. Un esquema CBR es útil cuando se tiene un canal con un ancho de banda limitado y se desea que el proceso de transmitir en tiempo real ocupe exactamente una porción del ancho de banda. Los cambios abruptos en la tasa requerida para transmitir un archivo VBR pueden causar interrupciones en dispositivos que no estén diseñados para manejar esta situación.

En CBR se mantiene una tasa de datos constante y la calidad del contenido es variable. La motivación para utilizar VBR radica no solo en el hecho de que puede ser un codificador más eficiente, sino en la posibilidad de mantener un contenido con calidad constante.

1.2. Objetivos

El objetivo principal de este proyecto es realizar un estudio sobre la compresión de audio sin pérdidas y mejorar los factores de compresión que dan otros modelos.

Para ello, se toma como referencia el modelo de AudioPak, que se explicará en capítulos posteriores, y que es un modelo sencillo de codificación de audio sin pérdidas.

Una vez elegido este modelo como referencia, se han introducido pequeñas modificaciones o innovaciones en las distintas fases de este codificador de audio. Estas modificaciones son emplear una longitud variable en el enventanado, variar los coeficientes enteros de la codificación linear y emplear tablas basadas en algoritmos de Huffman para la codificación entrópica.

Se realizan distintas pruebas que permitan comprobar el funcionamiento de las modificaciones introducidas en el sistema de codificación. Estas pruebas servirán como estudio de viabilidad de la implementación de un codificador con las extensiones propuestas.

1.3. Organización de la memoria

El cuerpo de la memoria está dividido en tres apartados claramente diferenciados:

ESTADO DEL ARTE: Se introducirán conceptos sobre la codificación de audio, los distintos enfoques existentes, sus aplicaciones y los distintos sistemas y estándares existentes.

EXTENSIONES INTRODUCIDAS: Se describirán los fundamentos teóricos de las extensiones que se van a introducir en este proyecto en las diferentes fases de la codificación (enventanado, decorrelación intracanal y codificación de la fuente)

EXPERIMENTACIÓN Y RESULTADOS: Se describirán las adaptaciones llevadas a cabo en las distintas fases de codificación con los diferentes experimentos realizados. Incluye también los resultados de las pruebas realizadas para comprobar el efecto de las modificaciones introducidas.

En último lugar se incluyen las conclusiones extraídas del proyecto, las líneas propuestas como trabajo futuro, un presupuesto y las referencias consultadas.

2. ESTADO DEL ARTE

Este apartado tiene como finalidad conocer los distintos métodos de codificación de audio que existen en la actualidad. Profundizando en el área de codificación sin pérdidas que es el que se trabaja en este proyecto.

En primer lugar, se van a estudiar las características del sonido y de la percepción que se tiene del mismo, para poder desarrollar los temas posteriores. Es decir, conocer el sonido desde un punto de vista físico, más concretamente entender la manera de percibir los sonidos por los seres humanos [1].

El análisis de la percepción de los sonidos (la psicoacústica) permite extraer conocimientos esenciales a la hora de procesar digitalmente los sonidos de forma más eficaz, lo que tiene su aplicación en codificación de audio y reconocimiento automático de sonidos y voz.

En codificación de audio con pérdidas se aprovechan las limitaciones del sistema auditivo humano para comprimir los datos de audio. Se usan diferentes métodos para eliminar los datos inaudibles.

En codificación de audio sin pérdidas se emplean algoritmos basados en la eliminación de la redundancia de la señal de audio, y por lo tanto, basados en el grado de predictibilidad de la información[2].

2.1. Funcionamiento del oído

El sentido del oído humano depende de una serie de procesos acústicos, mecánicos, hidráulicos nerviosos y mentales que tienen lugar en el oído y el cerebro humanos. Estructuralmente el oído se descompone en tres partes: oído externo, que pre-procesa y acondiciona la señal acústica, oído medio, que se encarga de transformar esa señal acústica en nerviosa y oído interno, parte en la que se realiza el procesamiento sensorial del sonido [1].

- **Oído externo:** Se encarga de recoger el sonido y conducirlo a través del canal auditivo hacia el tímpano haciendo que vibre siguiendo las ondas de presión produciendo a su vez cierto filtrado del mismo. El conducto auditivo se puede modelar como un tubo de longitud 2 cm abierto por un único lado. Un tubo de estas características tiene una frecuencia de resonancia de 4 kHz, frecuencia para la que el oído humano tiene mayor sensibilidad.

- **Oído medio:** Convierte los movimientos de aire del canal auditivo en movimientos del fluido de la cóclea. Para ello utiliza una estructura ósea formada por el tímpano, martillo, yunque, estribo y ventana oval que transforma grandes movimientos de aire con poca energía en pequeños movimientos del fluido de la cóclea de mayor energía. Realiza, por tanto, una labor de adaptación de impedancias, que es máxima para frecuencias de alrededor de 1 kHz. Alrededor de esta frecuencia la transmisión de energía será máxima y contará por tanto con una buena sensibilidad.
- **Oído interno:** Formado por la cóclea, realiza un análisis espectral de las vibraciones de su fluido interno transformándolas en impulsos nerviosos que se envían a través del nervio auditivo hacia el cerebro. La cóclea es un tubo largo y fino que se enrolla sobre sí mismo de forma espiral unas dos veces y media. Otra parte importante del oído interno es la membrana basilar, compuesta por 3000 células receptoras que detectan la vibración del oído interno. Estas células transforman movimientos del fluido en impulsos eléctricos que van al nervio auditivo. Los sonidos de una frecuencia particular hacen que la membrana basilar vibre con amplitud máxima en un punto particular. Distintas zonas en la membrana responden a distintas frecuencias. El estímulo más fuerte en un área local de la membrana es el que se envía al cerebro.

2.2. Codificación con pérdidas

Algoritmos importantes, como MP3, se basan en la llamada compresión con pérdidas (*lossy compression*), consistente en la eliminación de la parte de señal que no es audible por el oído humano, se aprovechan las limitaciones del sistema auditivo humano. Para ello, hace uso de los principios psicoacústicos que se exponen a continuación [1][3].

2.2.1. Umbral absoluto de audición

La energía o potencia de un sonido se mide físicamente mediante el nivel de presión sonoro (*sound pressure level*, SPL) en decibelios. La percepción de la energía o potencia de un sonido depende del nivel de presión sonoro, pero la relación entre ambas no es estrictamente lineal, sino que depende de forma compleja del nivel de presión sonoro y de la frecuencia.

El umbral absoluto de audición, también conocido como mínimo umbral auditivo, corresponde al sonido de intensidad más débil que se puede escuchar en un ambiente silencioso. El mínimo umbral auditivo no tiene un comportamiento lineal; se representa por una curva de Intensidad (dB) contra Frecuencia (Hz), que posee niveles mínimos entre 2 y 5 KHz, los cuales corresponden a la parte más sensitiva del oído humano. Esta representación del umbral auditivo se puede observar en la Figura 1.

Es una curva extremadamente importante en codificación de audio ya que las componentes de una señal de audio que caigan bajo este umbral son inaudibles y no se necesitan codificar ni transmitir. Además, el ruido introducido en una señal, por ejemplo el ruido de codificación, será inaudible siempre que se encuentre por debajo del umbral auditivo. Dada su importancia en la codificación se ha estudiado ampliamente la dependencia del umbral auditivo con la frecuencia.

- Típicamente es de unos 50 dB en 50 Hz.
- Baja a casi 0 dB en 500 Hz y se mantiene hasta los 2000 Hz.
- Entre 2 y 5 kHz baja por debajo de 0dB.
- Por encima de 5 kHz hay picos y valles que varían mucho con el sujeto.
- A partir de 16 kHz el umbral sube muy rápidamente.

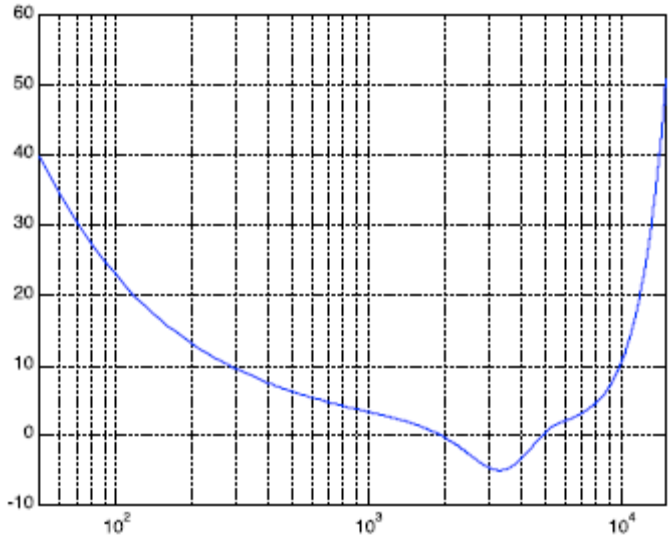


Figura 1. Representación del umbral auditivo

2.2.2. Enmascaramiento

El enmascaramiento es un fenómeno por el cual los sonidos débiles son anulados por otros sonidos más fuertes. Existen dos tipos de enmascaramiento: el simultáneo o frecuencial y el temporal.

El que resulta más común, es el simultáneo o frecuencial en el que el sonido enmascarador (un tono de alta energía, por ejemplo) excita la misma región de la membrana basilar de la cóclea que el sonido enmascarado (un tono próximo y en frecuencia y de menor energía, por ejemplo). Esto se produce por ejemplo cuando se está hablando y un ruido no deja escuchar lo que dice el interlocutor. El umbral de enmascaramiento en frecuencias indica el nivel de presión mínimo que debe tener una señal para ser audible en presencia del tono enmascarador.

Se puede calcular la relación entre señal y máscara (*signal to mask ratio*) como la diferencia de niveles de presión en decibelios entre una componente de una señal y el umbral de enmascaramiento en la frecuencia de dicha componente. Si el valor obtenido es negativo dicha componente será inaudible. Si por el contrario es positiva será audible, y tanto más cuanto mayor sea.

A modo de ejemplo, en la Figura 2 se pueden observar dos señales que aparecen por debajo del umbral de enmascaramiento. Estas señales serán por tanto inaudibles, aunque estén por encima del umbral auditivo:

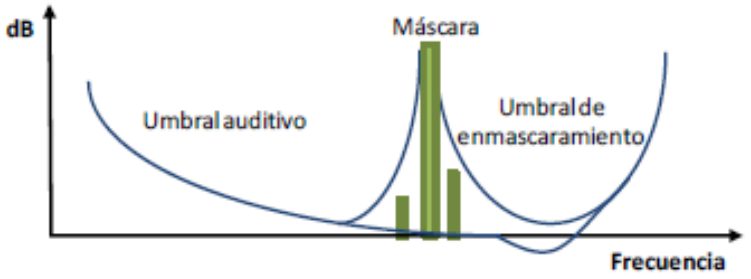


Figura 2. Ejemplo de enmascaramiento frecuencial

El efecto del enmascaramiento en frecuencias se puede emplear en la codificación de igual manera que el umbral auditivo:

- Las señales enmascaradas totalmente no son audibles y por tanto no tienen por qué ser codificadas y transmitidas.
- Si el ruido queda por debajo del umbral de enmascaramiento tampoco será audible y no supondrá un problema.

Por otro lado, el enmascaramiento temporal se produce entre sonidos que no están presentes simultáneamente y se explica por la inercia de los elementos físicos involucrados en el proceso auditivo.

- Si el tono enmascarado se empieza a producir antes de que comience la señal enmascarada se está hablando de preenmascaramiento. Comienza varias decenas de milisegundos antes y es fuerte sólo varios milisegundos después.
- Si por el contrario se deja de producir después de que termine la señal enmascaradora se habla de postenmascaramiento. Puede durar hasta varios cientos de milisegundos y es un fenómeno más fuerte que el preenmascaramiento.

La Figura 3 muestra el enmascaramiento temporal, indicando las zonas que corresponden a pre-enmascaramiento, enmascaramiento simultáneo o post-enmascaramiento.

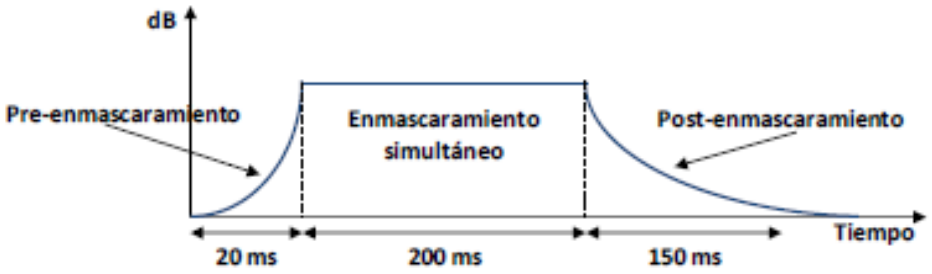


Figura 3. Enmascaramiento temporal

2.2.3. Bandas críticas

Estudios de la discriminación en frecuencia del oído han demostrado que en las bajas frecuencias, tonos con unos cuantos Hercios de separación pueden ser distinguidos; sin embargo, en las altas frecuencias para poder discriminar los tonos se necesita que estén separados por cientos de Hercios.

En cualquier caso, el oído responde al estímulo más fuerte que se presente en sus diferentes regiones de frecuencia; a este comportamiento se le da el nombre de bandas críticas. Los estudios muestran que las bandas críticas son mucho más estrechas en las bajas frecuencias que en las altas; lo que implica que el oído recibe más información en las bajas que en las altas frecuencias. Las bandas críticas tienen un ancho de aproximadamente 100 Hz para las frecuencias de 20 a 400 Hz; este ancho aumenta de manera logarítmica a medida que aumenta la frecuencia.

Las bandas críticas son comparables a un analizador de espectro con frecuencia central variable. Más importante aún es el hecho de que las bandas críticas no son fijas; son continuamente variables en frecuencia y cualquier tono audible creará una banda crítica centrada en él. Analizándolo desde otro punto de vista, el concepto de la banda crítica es un fenómeno empírico, una banda crítica es el ancho de banda al cual las respuestas subjetivas cambian abruptamente.

Estudiando las curvas de enmascaramiento frecuencial se descubrió la existencia de un pequeño margen de frecuencia alrededor de la frecuencia enmascaradora en el que el umbral de enmascaramiento en lugar de decrecer es plano, las bandas críticas. Esto es válido para cualquier señal enmascarante y enmascarada aunque con distinto nivel de enmascaramiento según el caso.

Esta conclusión llevó a un modelo auditivo como un banco de filtros paso banda que se superponían en frecuencia y con anchos de banda iguales a las bandas críticas. Aunque existen varias expresiones y fórmulas para las bandas críticas, la más aceptada es una expresión que describe la variación de las bandas críticas como una función de la frecuencia central enmascaradora.

Del enmascaramiento y las bandas críticas se puede por tanto concluir que:

- Su valor máximo se produce cerca de la frecuencia de la señal enmascaradora.
- Se mantiene constante en una pequeña banda (banda crítica) alrededor de la frecuencia central de la señal enmascaradora.
- Decrece rápidamente al alejarnos de la frecuencia central de la señal enmascaradora
- La forma de sus curvas depende de forma compleja de la señal enmascaradora, frecuencia y nivel.
- Comienza algunos milisegundos antes de que comience la señal enmascaradora (preenmascaramiento) y puede terminar cientos de milisegundos después de que termine la señal enmascaradora (postenmascaramiento).

Se puede concluir la codificación con pérdidas indicando que parte de la señal puede ser, descartada en la codificación final, debido a estos principios psicoacústicos. El método se dice que es con pérdidas ya que la señal descomprimida es distinta de la original.

Hay, por tanto, una disminución de calidad de sonido, que podrá ser mayor o menor en función del método de compresión empleado y del factor de compresión, definiéndose éste como sigue:

$$\text{Factor de Compresión} = \frac{\text{Longitud Señal Original}}{\text{Longitud Señal Comprimida}} \quad \textbf{(1)}$$

El nivel de compresión se puede controlar y depende de varios factores como de la calidad que se quiera obtener, el tamaño del fichero, el ancho de banda de la red o el tiempo de compresión.

Usualmente se utilizan compresiones máximas para transmisiones, especialmente cuando son servicios en directo como telefonía (telefonía IP o celular) o reproducciones en directo como *podcasting* (radio por Internet o programas de audio por Internet).

2.2.4. Formatos de audio más comunes

Existen varios estándares de codificación de audio con pérdidas, los más conocidos están descritos a continuación [4]:

- **WMA** (*Windows Media Audio*): desarrollada por Microsoft. Es la Versión de Windows para comprimir audio, muy parecido a MP3. No solo reduce el tamaño de archivos grandes, sino que también se adapta a diferentes velocidades de conexión en caso de que se necesite reproducir en Internet en Tiempo Real. Se basa en codificación mediante transformada de coseno discreta (MDCT). Frecuencias de muestreo compatibles, 32; 44,1; 48 (kHz) y admite VBR (48 a 192 kbps) [6].

- **Ac3 Codecs**: creados por los laboratorios Dolby. AC-3, es la versión más común que contiene hasta un total de 6 canales de sonido. Cada canal es independiente para cada altavoz y reproduce todo tipo de frecuencias, menos el sexto, que solo se encarga de las más bajas. El AC-3 es uno de los formatos denominados de compresión perceptual. Sigue los principios del modelo psicoacústico [6].

- **Musepack (MPC)** es un formato de compresión de audio con pérdida de código libre con gran énfasis en alta calidad, licenciado bajo LGPL o bajo la Licencia BSD. Musepack está considerado como uno de los mejores códecs para *bitrates* medios/altos. Sus raíces se basan en el algoritmo MPEG-1 *Audio Layer*-2 / MP2 [7].

- **OGG VORBIS:** Forma parte del proyecto Ogg y entonces es llamado Ogg Vorbis o también conocido como ogg por ser el códec más comúnmente encontrado en el contenedor Ogg. Vorbis es un códec de audio perceptivo de fines generales previsto para permitir flexibilidad máxima del codificador, permitiéndole escalar competitivamente sobre una gama excepcionalmente amplia de *bitrates*. Vorbis utiliza la Transformada de coseno discreta modificada (MDCT). Lo que diferencia a Ogg Vorbis del resto de grupo es que es gratuito, abierto y no está patentado. Su principal atractivo es la importante reducción que hace de un archivo de audio sin restarle calidad. Así mismo, se distingue por su versatilidad para reproducirse en prácticamente cualquier dispositivo y por ocupar muy poco espacio [6].

- **ATRAC:** desarrollado por Sony, basado en principios psicoacústicos, que ofrece distintas tasas de compresión, según la calidad de sonido. Actualmente se utiliza para guardar información de señales de audio, en MiniDisc y otros productos reproductores de audio propietarios de Sony [8].

- **MPEG** (*Moving Pictures Experts Group* - Grupo de Expertos en Imágenes en Movimiento) [9]: referido comúnmente como MPEG, es un grupo de trabajo del ISO/IEC encargado de desarrollar estándares de codificación de audio y vídeo. MPEG trabaja por fases: MPEG-1, MPEG-2, MPEG-4, dentro de cada fase hay niveles: "*Layers* I, II y III".

    - MPEG-1 (ISO/UEC 11172-3) (publicado en 1993): Frecuencias de muestreo: 32, 44.1 y 48 kHz. Modos de funcionamiento: mono, estéreo, dual y estéreo conjunto. Consta de tres capas o niveles:

        - *Layer*-1 (capa 1): 32 Subbandas iguales de 750 Hz de ancho, filtros de orden 511, empleo del modelo psicoacústico que se obtiene con FFT de 512 muestras. El tamaño de trama es 384 y emplea la codificación tipo mantisa y exponente.

        - *Layer*-2 (capa 2): 32 Subbandas iguales de 750 Hz de ancho, la FFT de 1024 muestras (el doble), el tamaño de trama es 1152 (triple que en *Layer* I). Además, cada subbanda genera 36 muestras, se agrupan en 3 grupos de 12 que comparten factor de escala.

        - *Layer*-3 (capa 3) MP3: emplea un banco de filtros híbrido, 32 filtros PQMF 750 Hz de ancho y MDCT. El tamaño de ventana es variable entre 576 (18 coeficientes MDCT) y 192 (6 coeficientes MDCT), de esta manera, combate preecos. Además, existen ventanas especiales de transición corto/largo y largo/corto. Es un códec de audio muy extendido. Su peculiaridad es su tamaño de compresión, 11 a 1.

- MPEG-2 BC (ISO/IEC 13818-3) (publicado en 1995): Extensión a sonido multicanal compatible con MPEG-1. Consta de 5 canales principales más uno de baja frecuencia (LFE). Extensión del MPEG-1 hacia menores regímenes binarios, frecuencias de muestreo: 16, 22.05 y 24 kHz y velocidades: 32-256 kb/s (MP-I) y 8-160 kb/s (MP-2 y 3)

- **AAC** (*Advanced Audio Coding*- Codificación de Audio Avanzada): Codificación estándar para audio reconocida por ISO en el patrón MPG-2. El AAC utiliza una frecuencia de bits variable (VBR), es un algoritmo de codificación de banda ancha de audio que tiene un rendimiento superior al del MP3. Se basa en la eliminación de redundancias de la señal acústica, así como en compresión mediante la transformada de coseno discreta modificada (MDCT) [5].

2.3. Codificación sin pérdidas

A lo largo de este trabajo, se van a estudiar los algoritmos de compresión sin pérdidas (*lossless compression*). Tras descomprimir la señal tratada, se obtiene exactamente la original, es decir, si se llama x a la señal de audio, x[n] serían cada una de sus muestras. En el caso de la codificación sin pérdidas se tiene que $x_{descomprimida}[n] = x_{original}[n]$.

Dado que la calidad de la señal no se altera, se trata de algoritmos útiles en aplicaciones para sistemas de alta fidelidad, empleados tanto en lo personal (almacenamiento de audio en CD y DVD), pero sobre todo en el ámbito profesional (estudios de música, productoras de cine, etc.).

A modo de ejemplo, es interesante la aplicación que propone [2], que es combinar ambos tipos de algoritmos en la descarga de ficheros de audio por Internet: escuchar una muestra de archivos comprimidos con pérdidas (bajo tamaño) y, una vez seleccionado el de interés, descargarlo sin pérdidas.

Existen ciertos programas que comprimen los ficheros sin pérdidas. Algunos son tan conocidos como WinZip. Es evidente que si el archivo a comprimir es, por ejemplo, un ejecutable o una librería dinámica, el hecho de tener errores en el fichero descomprimido puede suponer el no funcionamiento de la aplicación en sí con toda seguridad. Sin embargo, si se trata de comprimir un fichero de audio con este programa, se observa claramente que la tasa de compresión no es muy buena. La clave está en que las señales de audio tienen fuentes de redundancia específicas y propias de la señal, que hacen que sea necesaria la aplicación de nuevos algoritmos, distintos de los clásicos.

Como es de esperar, los factores de compresión en el caso sin pérdidas son bastante inferiores a los del caso con pérdidas. En concreto, mientras estos últimos algoritmos pueden comprimir en relación 12:1, los algoritmos sin pérdidas raramente superan la compresión 3:1.

A continuación, se comenzarán a ver los principios básicos de la compresión sin pérdidas, analizando matemáticamente las señales y los procesos de interés que tienen lugar. Después, se enumerarán los algoritmos ya implementados que existen en la actualidad y se desarrollará a lo largo de los capítulos posteriores un algoritmo sencillo llamado AudioPak, que servirá como punto de partida para este proyecto.

Principios básicos de la compresión sin pérdidas y fases de desarrollo

Las técnicas empleadas en la compresión sin pérdidas se basan en eliminar la redundancia de la señal y codificar el resultado con un esquema digital eficiente.

La redundancia es una propiedad de los mensajes, consistente en tener partes predictibles a partir del resto del mensaje y que por tanto en sí mismo no aportan nueva información o repiten parte de la información.

Para realizar la codificación se utilizan algoritmos basados en la eliminación de la redundancia de la señal de audio, y por lo tanto en el grado de predictibilidad de la información. Si la señal tiene patrones repetitivos, éste es redundante y por lo tanto fácil de predecir.

Usualmente los patrones repetitivos de señal son más evidentes en otros dominios (temporal, frecuencial...), por este motivo la transformación de la señal en función de estos patrones permitirá reducir o eliminar la redundancia.

Comprimir significa reducir la redundancia. En audio, hay tres fuentes de redundancia:

- Intercanal: redundancia entre canales. Se da en los sistemas con más de un canal, estéreo o multicanal. En estos sistemas las señales de los distintos canales pueden ser muy parecidas, por ello es una fuente de redundancia que se podría combatir. Este proyecto se ha basado en audio de un solo canal, por tanto, no habrá que abordar la problemática de este tipo de redundancia.

- Intracanal: redundancia temporal, que se manifiesta en una correlación entre muestras continuas. Se ha demostrado que las muestras de señales naturales, como voz y audio, pueden predecirse con gran exactitud a partir de una combinación lineal de muestras adyacentes en el espacio o en el tiempo. Para ello se emplean los códigos de predicción lineal, para combatir este tipo de redundancia. [10]

- Estadística: este tipo de redundancia se produce por no usar un código fuente óptimo, se basan en los valores estadísticos de la señal para eliminar esta redundancia. Cuando una señal tiene una estadística bien definida, en la que predominan unos pocos valores que se presentan con mucha probabilidad, frente a otros con una probabilidad baja, se suelen usar códigos de longitud variable, dónde se codifica con pocos bits los símbolos probables y con cadenas de bits más largas los símbolos menos probables. Esto es lo que se denomina codificación entrópica, que se desarrollará más adelante.

Una vez vistas las tres fuentes de redundancia, hay que intentar combatirlas. Las técnicas de reducción de la redundancia son el fundamento de muchos de los métodos de codificación de señales.

La mayoría de los compresores sin pérdidas se basan en el esquema mostrado en la Figura 4.

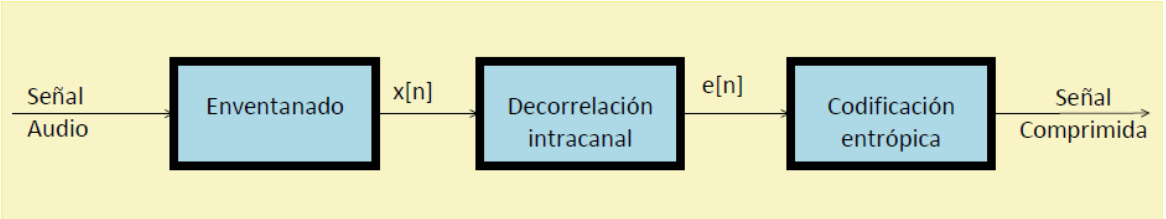


Figura 4: Esquema general de los compresores de audio sin pérdidas

En esta figura se muestra el procesamiento o esquema de un solo canal, aunque el número de canales depende del número de señales de audio simultáneas que contiene el flujo de datos. Pueden ser mono (1 canal), estéreo (2 canales) o multicanal como 5.1 (seis canales) o 7.1 (ocho canales).

En el actual proyecto se centra en la codificación de audio de un solo canal, aunque resultaría de interés tener en cuenta la codificación de audio sin pérdidas para sistemas estéreo o multicanal. Estudiar si la codificación de los canales podría hacerse de forma independiente, o en cambio, habría que buscar una relación entre canales para evitar redundancias e incrementar la compresión. Para esto se tendrían que estudiar o tener en cuenta diversos parámetros[11].

Uno de estos parámetros es la correlación cruzada entre canales, por ejemplo, en señales estéreas la correlación cruzada entre el canal L y el canal R. Otro parámetro es la relación de potencia de los canales. En los llamados codificadores espaciales paramétricos de audio estos y otros parámetros son extraídos de la señal de audio original para producir una señal de audio que tenga un número reducido de canales, por ejemplo, sólo un canal único, más un conjunto de parámetros que describen las propiedades espaciales de la señal de audio original.

Por tanto, el trabajo desarrollado en este proyecto se centra en los tres bloques la codificación de audio sin pérdidas, que se puede diferenciar en la Figura 4. Intentando mejorar cada uno de ellos para conseguir un grado de compresión mayor, es decir, para intentar combatir las diferentes fuentes de redundancia. En los siguientes puntos se hará un resumen de los fundamentos teóricos de las diferentes fases de desarrollo del proyecto.

2.3.1. Enventanado

Las señales de audio son señales no estacionarias, por tanto es recomendable realizar un análisis localizado de la señal. La aplicación de técnicas de enventanado, surge como solución necesaria a esta problemática, permitirán el análisis de tramos estacionarios de la señal a codificar.

El proceso de enventanado o *framing* es utilizado como primer paso en la compresión para que la edición de los ficheros de audio se produzca con mayor facilidad. El enventanado o segmentación de la señal consiste en agrupar un número de muestras consecutivas. Cada segmento se procesa individualmente. Si no se divide la señal, para poder tratar el fichero de audio, sería necesario descomprimir toda la señal, lo cual es un inconveniente en cuanto a tiempo y carga computacional se refiere. Desde este punto de vista, se necesitarían tener tramas (o *frames*) lo más pequeños posibles. Sin embargo, es importante notar que cada trama tendrá que tener asociada una cabecera con información para la descompresión, por lo que bloques muy pequeños generarían un exceso de cabeceras (*overhead*), dando como resultado algo que, en ciertos casos, puede neutralizar el propósito de la compresión, es decir, aumentar el número de bits necesarios para la codificación.

Por otro lado, tramas de longitud elevada llevan asociadas problemas de editabilidad y de poder ajustar correctamente la señal enventanada, es decir, poder calcular los parámetros de codificación óptimos para esa trama. Por tanto, es necesario llegar a una solución de compromiso para decidir la longitud de trama sin que se produzca *overhead* ni haya problemas de editabilidad.

El principio de incertidumbre de Heisenberg, tiene su versión más conocida en la mecánica cuántica. Expone que no se puede determinar, en términos de la física clásica, simultáneamente y con precisión absoluta, ciertos pares de variables o propiedades físicas, como son, por ejemplo, la posición y el momento lineal (cantidad de movimiento) de un objeto dado. En otras palabras, cuanta mayor certeza se busca en determinar la posición de una partícula, menos se conoce su cantidad de movimiento lineal y, por tanto, su velocidad.

Aunque el principio de incertidumbre para el análisis de señales no está relacionado con el tema para el que originalmente fue desarrollado, se toma el mismo nombre debido a la analogía entre ambos, pues en definitiva se trata de dos variables relacionadas entre sí donde existe alguna propiedad que no se puede cumplir al mismo tiempo para ambas, y un efecto de mejora en la primera variable implica forzosamente un empeoramiento en la segunda, en terminología matemática, se dice que los operadores asociados a dichas variables no conmutan.

En este caso, nuestras variables son el tiempo y la frecuencia. Este principio supone que una mejora en la resolución que se obtiene para una de las variables empeora la resolución de la segunda variable, y viceversa.

Si se aplica este principio al problema de la longitud de las ventanas se obtendrá que si se emplea una longitud de ventana amplia se obtiene una buena resolución en frecuencia pero a la vez, una pobre resolución en el tiempo. Esto se pude justificar mediante la transformada de Fourier, si se emplea una longitud de ventana amplia se tendrán muchos puntos o muestras para poder calcular la transformada de Fourier. Mientras que por el contrario, una ventana de longitud más corta da como resultado una buena resolución en el tiempo y una pobre resolución en frecuencia.

También puede ser explicado con referencia al muestreo y a la frecuencia de Nyquist. Si se toma una ventana de N muestras del valor real de una señal arbitraria con una tasa de muestreo de fs. Tomando la transformada de Fourier se produce N coeficientes complejos. De estos coeficientes solo la mitad son útiles. Estos N/2 coeficientes representan las frecuencias 0 a fs/2 (Nyquist) y dos coeficientes consecutivos son espaciados aparte por fs/N Hz.

Para incrementar la resolución en frecuencia de la ventana, la frecuencia de espaciado de los coeficientes necesita ser reducida. Hay solo dos variables, pero el disminuir fs (y mantener N constante) causará que el tamaño de la ventana aumente, debido a que ahora hay menos muestras por unidad de tiempo. La otra alternativa es incrementar N, pero esto causa de nuevo que el tamaño de la ventana se incremente. Cualquier intento de incrementar la resolución en frecuencia causa un mayor tamaño de la ventana y por lo tanto una reducción en la resolución del tiempo y viceversa.

Esta propiedad supone una restricción importante para las representaciones tiempo-frecuencia, ya que nunca se podrán obtener resultados totalmente ajustados tanto en el dominio temporal como espectral.

Matemáticamente, la acción de enventanar una señal discreta se define como la multiplicación en el tiempo de la señal por la función ventana, que es otra señal que toma valores no nulos únicamente en un intervalo entre $0 \leq n \leq N-1$. Los parámetros fundamentales que determinan el enventanado de una señal son su longitud temporal, es decir, el tamaño del intervalo extraído, sobre el cual se ha discutido anteriormente, y la función que define la ventana. Desde el punto de vista puramente teórico, lo ideal es tener ventanas de longitud elevada, ya que, de esta forma, las señales enventanadas son más parecidas a las reales, y aumenta la resolución frecuencial del sistema.

Existen numerosos tipos de ventanas: rectangular, triangular, Hanning, Hamming, Blackman, etc. La elección de la función ventana determinará como influirán los transitorios que se dan en los extremos del segmento de señal enventanada sobre el resultado final, la densidad espectral de potencia.

Las ventanas tienen propiedades muy variadas, por ejemplo las ventanas de Hamming y rectangular, a pesar de poseer un lóbulo principal muy definido, producen frecuencias parásitas (lóbulos secundarios), no tan atenuadas como otras ventanas similares (Hanning y Blackman). Al comparar, otras ventanas: Hanning y Blackman, se van a encontrar que existen resultados similares en cuanto a atenuación de frecuencias parásitas, amplitud y ancho de banda del lóbulo principal, por lo que para la elección de cada una de ellas se deberá tener en cuenta la aplicación correspondiente. No existe ninguna ventana que pueda considerarse la más óptima en términos absolutos, todas ellas implican tomar una decisión de compromiso entre la resolución frecuencial y temporal que se desea obtener, lo cual, depende del tamaño de los lóbulos principal y secundarios de la transformada de cada una de ellas.

2.3.2. Decorrelación intracanal

La siguiente etapa en el proceso de compresión se denomina decorrelación intracanal y sirve, fundamentalmente, para intentar conseguir que el espectro de la señal se transforme en uno más plano, es decir, realizar una especie de promedio frecuencial. Esta etapa del decodificador está basada en la redundancia temporal de las muestras de la señal de audio. Mediante códigos de predicción lineal se trata de combatir esta redundancia.

Si los valores futuros de las muestras de audio pueden ser estimados, entonces sólo será necesario transmitir las reglas de predicción a lo largo de la señal con la diferencia entre los valores estimados y la señal real. Esta es la función del bloque de decorrelación intracanal. Es útil considerar como actúa la predicción en el dominio de frecuencia (Shannon).

Se reduce la redundancia intracanal mediante la estimación eficiente de la señal de entrada x[n]. Dicha estimación tiene la finalidad de obtener, mediante pocos parámetros, una aproximación de la entrada. Gracias a esta estimación resulta un error de aproximación o residuo e[n] con menor varianza (es decir, menor potencia) que la entrada, resultando por ende un conjunto de datos más fácilmente comprimible, es decir, un espectro más plano.

La Figura 5 muestra el espectro de un fragmento de música. Si este espectro fuese plano, un filtro de predicción lineal no daría ningún tipo de ganancia. Sin embargo, el espectro de la señal de audio está lejos de ser plano, por lo que bloque del decorrelador puede obtener mejoras en la compresión mediante el aplanamiento de la señal; dejando la señal de diferencia con un espectro lo más plano posible.

El teorema de Gerzon y Craven [12] indican que el nivel óptimo de la señal decorrelada está dado por la media del espectro de la señal original, cuando se representa como nivel espectral (dB) frente a la frecuencia lineal. Como se ilustra en la Figura 5, este promedio de dB puede tener mucha menos energía que la señal original, por lo tanto, se tiene una reducción de la tasa datos. De hecho, esta reducción de potencia representa la descripción de la información de la señal según la definición de Shannon [13].

En resumen, el teorema de Gerzon y Craven, dice que cuanto más plano es el espectro de una señal, con menos bits se podrá codificar [14]. En definitiva, se trata de eliminar la redundancia presente en la señal mediante el método de la decorrelación aplicado a todas las muestras de cada trama.

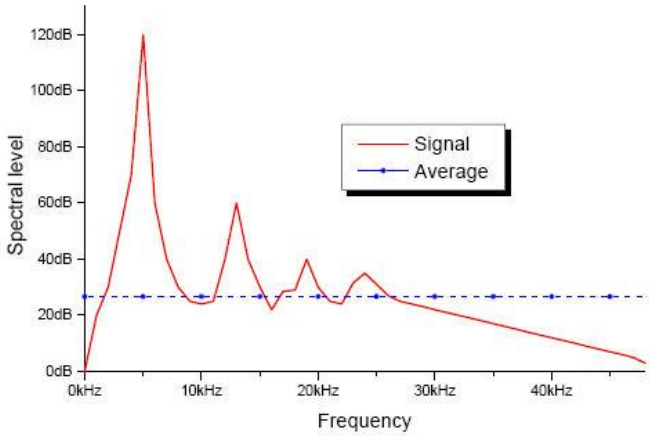


Figura 5: Espectro de una señal y su promedio

Los codificadores sin pérdidas actuales utilizan método del modelo predictivo para realizar esta etapa.

MODELO PREDICTIVO

Se basa en utilizar un sistema de predicción lineal que genera, para cada trama, una señal discreta de error, e[n]. Los parámetros del predictor representan, por tanto, la redundancia que es eliminada de la señal. Se trata del modelo más utilizado en la actualidad y en el que se basa este proyecto.

El principio fundamental del modelo predictivo es ser capaz de predecir el valor de una muestra x[n] a partir del valor de las muestras precedentes: x[n − 1], x[n − 2], etc. El esquema se muestra en la Figura 6.

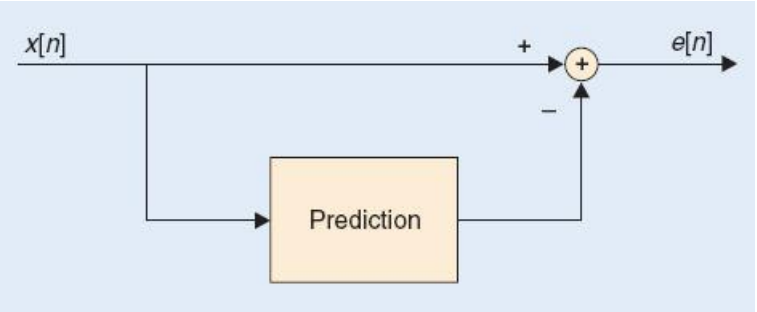


Figura 6: Modelo de predicción

La idea es que, si el predictor es óptimo, la señal de salida e[n] está incorrelada y, por tanto, tiene un espectro plano. Análogamente, e[n] tomará, de media, valores más bajos que x[n], por lo que menos bits serán requeridos en la posterior codificación.

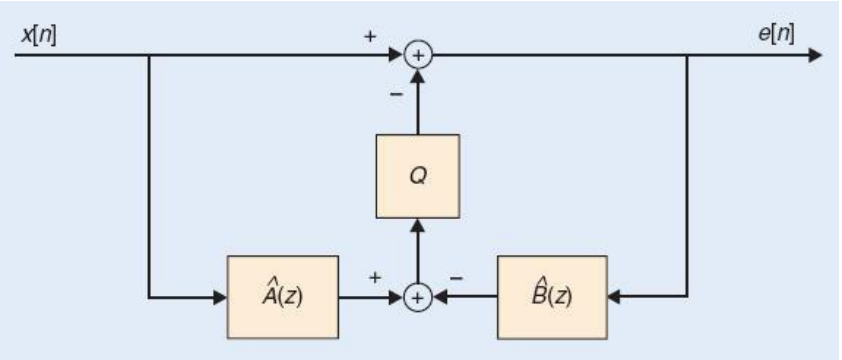


Figura 7: Estructura general de predicción

En la Figura 7 se muestra la estructura general de predicción. De la misma se deduce inmediatamente la expresión matemática (2), donde Q{} denota la operación de cuantificación.

$$e[n] = x[n] - Q\left\{\sum_{k=1}^{M} \hat{a}_k x[n-k] - \sum_{k=1}^{N} \hat{b}_k e[n-k]\right\}, \quad \textbf{(2)}$$

Aparte, se definen los polinomios del predictor como sigue:

$$\hat{A}(z) = TZ\left\{\sum_{k=1}^{M} \hat{a}_k x[n-k]\right\} = \sum_{k=1}^{M} \hat{a}_k z^{n-k} \quad \textbf{(3)}$$

$$\hat{B}(z) = TZ\left\{\sum_{k=1}^{N} \hat{b}_k e[n-k]\right\} = \sum_{k=1}^{N} \hat{b}_k z^{n-k} \quad \textbf{(4)}$$

Nótese que la estructura de (2), si se elimina el efecto del cuantificador, se corresponde con un filtro digital [15], que puede ser de respuesta al impulso finita (FIR, si $\hat{B}(z) = 0$) o infinita (IIR, si $\hat{B}(z) \neq 0$).

La operación de cuantificación que aparece en la Figura 7 añade al predictor global la propiedad de no linealidad. Sin embargo, dado que se trata de una operación con una precisión muy elevada (16 bits), es razonable eliminar el efecto del cuantificador a la hora de analizar efectos de primer orden y para desarrollar métodos de estimación para los parámetros del predictor. El cuantificador es algo, sin embargo, necesario, puesto que las señales en última instancia deben ser completamente digitales.

La reconstrucción de la señal es sencilla. A partir de (2), se deduce lo que sigue:

$$x[n] = e[n] + Q\left\{\sum_{k=1}^{M} \hat{a}_k x[n-k] - \sum_{k=1}^{N} \hat{b}_k e[n-k]\right\}, \quad \textbf{(5)}$$

lo cual se muestra en la Figura 8.

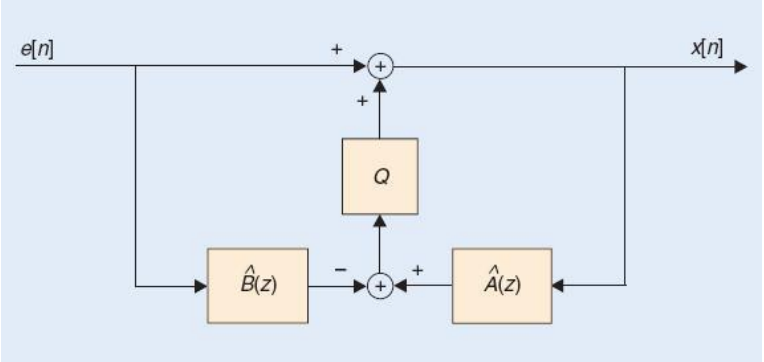


Figura 8: Estructura general de reconstrucción

Los predictores de tipo lineal son ampliamente utilizados en sistemas de procesamiento de audio y voz [2]. En la mayoría de las aplicaciones, se utilizan predictores FIR, y $\hat{A}(z)$ es calculado por minimización del error cuadrático medio de predicción. Si se elimina el cuantificador y se emplea un esquema FIR ($\hat{B}(z) = 0$), los coeficientes pueden encontrarse mediante la resolución de un simple sistema lineal de M ecuaciones con M incógnitas [16]. Se define el error cuadrático medio de predicción como sigue:

$$E_n = \sum_m (x[n+m] - e[n+m])^2 = \sum_m \left(x[n+m] - \sum_{k=1}^{M} \hat{a}_k x[n-k] \right)^2, \quad \textbf{(6)}$$

Para minimizar esta expresión, se aplica la propiedad de nulidad de la primera derivada a cada uno de los coeficientes del polinomio $\hat{A}(z)$, resultando lo que sigue:

$$\sum_{m} x[n+m-i]x[n+m] = \sum_{k=1}^{M} \hat{a}_k \sum_{m} x[n+m-i]x[n+m-k], \qquad 1 \le i \le M \tag{7}$$

Por simplicidad, se definirán las siguientes funciones:

$$\Phi_n(i,k) = \sum_{m} x[n+m-i]x[n+m-k], \tag{8}$$

simplificando el sistema de M ecuaciones con M incógnitas a resolver:

$$\sum_{k=1}^{M} \hat{a}_k \Phi_n(i,k) = \Phi_n(i,0), \qquad 1 \le i \le M, \tag{9}$$

Tras realizar operaciones matemáticas detalladas en [16], se puede llegar a la Ecuación (10). Si se examina esta expresión se deduce que error mínimo total consta de una componente fija y de una componente que depende de los coeficientes del predictor.

$$E_n = \Phi_n(0,0) - \sum_{k=1}^{M} \hat{a}_k \Phi_n(0,k) \tag{10}$$

Por lo tanto, se ha llegado a que los coeficientes del polinomio, imponiendo la condición de mínimo error cuadrático medio, se pueden calcular resolviendo el sistema (9), que es un sistema lineal de M ecuaciones con M incógnitas. La solución es, aparentemente, muy simple, pero hay que destacar que los parámetros del sistema hay que calcularlos para cada trama. Esto no tiene por qué ser sencillo ni rápido. Nótese que en aplicaciones en tiempo real no se puede emplear demasiado tiempo en estos cálculos. Sin embargo, también es cierto que este proceso de la codificación aunque es costoso sólo se realiza una vez.

Es importante a recalcar que los coeficientes del predictor no son únicos, sino que cambian dinámicamente de acuerdo con las características y estructura de la señal. En este aspecto, los sistemas más sofisticados y potentes calculan constantemente errores cuadráticos medios para deducir los coeficientes, mientras que los más simples los eligen de librerías que almacenan en memoria. De hecho, la mayoría de los compresores actuales utilizan esta última técnica, dado que la primera exige un gasto considerable en tiempo y recursos de computación. La técnica de las librerías cuenta, además, con la ventaja de que la simple identificación del predictor es lo único necesario en la trama para ser codificado.

En cuanto a IIR, se puede decir que, si bien pueden trabajar con un rango espectral superior al del caso FIR, su relación de compresión no es demasiado grande. Además, el cálculo de los coeficientes por el método de minimización de error cuadrático medio es enormemente costoso y complejo desde el punto de vista computacional.

En cuanto a la implementación a nivel *hardware* de estos sistemas, se apunta brevemente el hecho de que, en general, los coeficientes de $\hat{A}(z)$ son fraccionarios, por lo que es necesaria su codificación en punto flotante. Sin embargo, en algunos casos, se trata de números enteros, por lo que la implementación física es mucho más simple y económica, ya que la codificación también es más simple [17].

Para terminar con este apartado, se recogen en la siguiente tabla algunos algoritmos sin pérdidas utilizados en la actualidad, clasificados según los criterios expuestos.

Modelo FIR	Shorten, Sonarc, WA, Philips
Modelo IIR	OggSquish, DVD
Coeficientes enteros (FIR)	Shorten, HEAR, WA, MUSICompress, AudioPaK

Tabla 1: Algoritmos de compresión sin pérdidas (método predictivo)

2.3.3. Codificación de entropía

En esta fase del codificador se trata la redundancia estadística, que es la que se produce por no usar un código fuente óptimo. Por tanto, para buscar ese código fuente óptimo, se toman como referencia los valores estadísticos de la señal para eliminar este tipo de redundancia.

La información que transmite un mensaje no está relacionada con su longitud. Se pueden tener dos mensajes con distinta longitud y que transmitan la misma información. El concepto de información está muy relacionado con el concepto de probabilidad. Cuanto más probable es un mensaje menos información contiene.

La información que aporta un determinado valor (símbolo), x_i, de una variable aleatoria discreta X se define como:

$$I(x_i) = \log_2 \frac{1}{p(x_i)} \quad \textbf{(11)}$$

El objetivo de la codificación es obtener una representación eficiente de los símbolos de la señal. Para que la codificación sea eficiente es necesario tener un conocimiento de las probabilidades de cada uno de los símbolos.

El dispositivo que realiza esta tarea es el codificador de la fuente. Este codificador debe cumplir el requisito de que cada palabra de código debe decodificarse de forma única, de forma que la secuencia original sea reconstruida perfectamente a partir de la secuencia codificada.

Los bloques anteriores, tanto la predicción como la codificación con pérdidas no aportaban ningún tipo de compresión a la señal. Lo único que se ha hecho ha sido preparar la misma para que tenga unas características adecuadas para poder ser codificada. Quería codificar cada una de las muestras de la señal decorrelada de la manera más eficiente posible. La codificación de entropía, es el nombre que recibe este proceso, tiene como objetivo eliminar la redundancia de la señal obtenida, e[n]. Es importante recalcar que, dado que se está en codificación sin pérdidas, no hay pérdida de información en este proceso.

Un concepto muy ligado al de cantidad de información es el concepto de entropía.

El concepto básico de entropía en la teoría de la información, desarrollada por Shannon, tiene mucho que ver con la incertidumbre que existe en cualquier experimento o señal aleatoria. Es también la cantidad de "ruido" o "desorden" que contiene o libera un sistema. De esta forma, podremos hablar de la cantidad de información que lleva una señal.

Shannon ofrece una definición de entropía que satisface las siguientes afirmaciones:

- La medida de información debe ser proporcional (continua). Es decir, el cambio pequeño en una de las probabilidades de aparición de uno de los elementos de la señal debe cambiar poco la entropía.
- Si todos los elementos de la señal son equiprobables a la hora de aparecer, entonces la entropía será máxima.

La entropía determina el límite máximo al que se puede comprimir un mensaje usando un enfoque símbolo a símbolo sin ninguna pérdida de información (demostrado analíticamente por Shannon), el límite de compresión (en bits) es igual a la entropía multiplicada por el largo del mensaje. También es una medida de la información promedio contenida en cada símbolo del mensaje. Su cálculo se realiza a partir de su distribución de probabilidad p(x) mediante la siguiente fórmula (12):

$$H(V) = -\sum_{i=1}^{n} P(x_i) \log_a [P(x_i)] = \sum_{i=1}^{n} P(x_i) \log_a \left[\frac{1}{P(x_i)} \right], \quad \textbf{(12)}$$

Esta magnitud H(V), se conoce como la entropía de la variable aleatoria (V), y $P(x_i)$ representa la probabilidad de símbolo. La base del logaritmo, *a*, dependerá de la variable *X* con que estemos trabajando, es decir, para una variable binaria usaremos la base 2, para una ternaria la base 3.

Se puede decir que la entropía de una variable aleatoria es el número medio de bits que se necesitarán para codificar cada uno de los estados de la variable, suponiendo que se exprese cada suceso empleando un mensaje escrito en un alfabeto binario. La entropía H está acotada superiormente cuando es máxima, es decir, que todos los elementos son equiprobables, y no supone perdida de información.

La codificación entrópica es esencial en codificación multimedia de todo tipo. Su principio básico es que se asignan códigos más cortos a símbolos con mayor probabilidad de aparición. Por tanto, la longitud del código será:

$$\bar{L} = \sum p(x_i)L(x_i), \tag{13}$$

Dónde $L(x_i)$ representa la longitud (bits) del símbolo codificado.

Es evidente que la longitud media del código tendrá un límite inferior mayor que cero, ya que los datos no se pueden comprimir infinitamente. Se tiene por tanto que $\bar{L} \geq L_{min}$, pero para acotar la longitud del código inferiormente hay que basarse en el primer teorema de Shannon. Este establece que dada una fuente digital sin memoria con entropía H(S), la longitud media de código $\bar{L}$ para cualquier código sin pérdidas está acotada inferiormente por $\bar{L} > H(S)$. Por tanto, un código será tanto mejor cuanto más se aproxime a su longitud media a la entropía de la fuente.

Existen diferentes técnicas de codificación basadas en la entropía:

- Codificación de Huffman: independiente de la estadística del alfabeto de símbolos, cumple que: $H(S) \leq L(S) \leq H(S)+1$
- Codificación Aritmética: adecuada para codificar secuencias de símbolos.
- Codificación de Golomb-Rice: muy adecuada para datos con estadística Laplaciana, es decir, para codificación de audio.

Codificación Huffman

La codificación Huffman es un algoritmo usado para compresión de datos. El término se refiere al uso de una tabla de códigos de longitud variable para codificar un determinado símbolo, donde la tabla ha sido rellenada de una manera específica basándose en la probabilidad estimada de aparición de cada posible valor de dicho símbolo [1].

La codificación Huffman usa un método específico para elegir la representación de cada símbolo, que da lugar a un código prefijo. Es decir, la cadena de bits que representa a un símbolo en particular nunca es prefijo de la cadena de bits de un símbolo distinto. Este código prefijo representa los caracteres más comunes usando las cadenas de bits más cortas, y viceversa.

Huffman fue capaz de diseñar el método de compresión más eficiente de este tipo: ninguna representación alternativa de un conjunto de símbolos de entrada produce una salida media más pequeña cuando las frecuencias de los símbolos coinciden con las usadas para crear el código.

Para conseguir esta asignación óptima, los símbolos se representan con códigos cuya longitud es inversamente proporcional a la probabilidad del símbolo. De esta forma, los símbolos menos probables se representan con códigos más largos, y los más probables con códigos más cortos. El proceso de asignación de códigos se lleva a cabo mediante la construcción de un árbol binario, desde las hojas hacia la raíz, de manera que los nodos hoja son los símbolos del alfabeto.

En la construcción del árbol, los nodos menos probables se unen sucesivamente entre sí para formar otro nodo de mayor probabilidad, de forma que cada uno de los enlaces añade un bit al código de los símbolos que se están juntando. Este proceso finaliza cuando sólo se dispone de un nodo, de tal forma que éste representa la raíz del árbol que se ha creado.

Esto significa que para poder realizar la codificación Huffman es necesario conocer las probabilidades de aparición de los símbolos o sus pesos correspondientes. El objetivo es poder encontrar un código binario prefijo (un conjunto de elementos del código) con longitud de palabra esperada mínima (expresado de otra manera, un árbol con longitud de camino mínima). Además de tratarse de un código biunívoco, es decir, aquel código decodificable de forma única.

Como consecuencia del teorema de codificación de fuente de Shannon, que se expuso en capítulos anteriores, la entropía es una medida de la longitud de palabra más pequeña del código que es teóricamente posible para un alfabeto dado con unos pesos asociados. El código de Huffman no es óptimo únicamente en el sentido de que ningún otro código posible funciona mejor, conociendo los pesos, sino que además está muy cercano al límite teórico establecido por Shannon.

$$H(x) \leq L_H(x) \leq H(x) + 1$$

Además, el código de Huffman no necesita ser único, pero sí es siempre uno de los códigos que minimiza la longitud del código.

Existen muchas variaciones del código de Huffman, algunos que utilizan Huffman como algoritmo, y otros que encuentra el código prefijo óptimo. Algunos ejemplos serían:

Código Huffman n-ario: El algoritmo n-ario de Huffman usa el alfabeto {0, 1, …., n-1} para codificar el mensaje y construir un árbol n-ario. Este enfoque fue considerado por Huffman en su enfoque originario.

Código Huffman adaptable: La variación llamada código de Huffman adaptable calcula dinámicamente la probabilidad de la frecuencia de la cadena de origen basada en antiguas apariciones. Está relacionado con la familia de algoritmos LZ.

La codificación de Huffman se utiliza a menudo en algún otro método de compresión. Como la deflación y códec multimedia como JPEG y MP3 que tienen una cuantificación digital basada en la codificación de Huffman. Además, el estándar de MP3 estable 32 tablas para la codificación de los símbolos basadas en los principios de la codificación de Huffman.

Se va a hacer una introducción sobre cómo se realiza la creación de los códigos binarios de Huffman con un ejemplo sencillo, diferente de las señales de audio.

Se supone que se va a codificar la siguiente cadena de símbolos:

S={aabaacc}

que usa el alfabeto:

$$A=\{a, b, c\}$$

La probabilidad de cada uno de los símbolos vendrá dada por las siguientes expresiones:

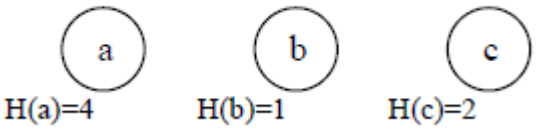
$$P(a)=4/7 \quad P(b)=1/7 \quad P(c)=2/7$$

El símbolo “a”, que se repite mucho, es interesante representarlo con el menor número de bits posibles.

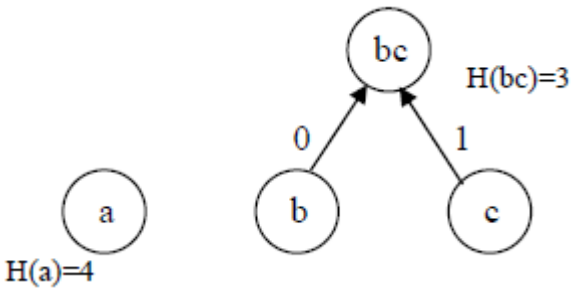
El primer paso del algoritmo será plantear un grafo no conexo de nodos que representan cada uno de los símbolos del alfabeto, junto con su probabilidad asociada. Para mejorar su comprensión y facilitar el cálculo, en lugar de usar directamente las probabilidades de los símbolos P(a), P(b), P(c), se emplea una cuenta del número de repeticiones, a modo de histograma, de forma que:

$$H(a)=4 \quad H(b)=1 \quad H(c)=2$$

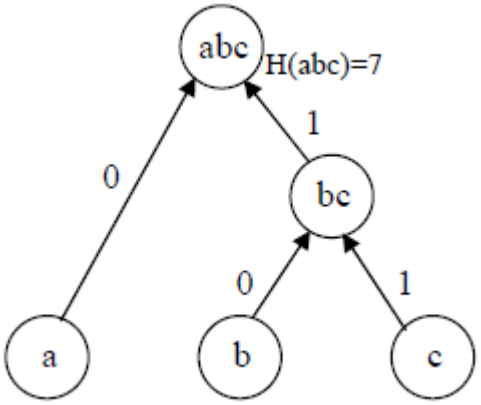
El grafo inicial será el siguiente:



El primer paso será juntar los nodos menos probables en un nuevo nodo, asignando un bit distinto de cada uno de los enlaces. El grafo resultante es el siguiente:



A continuación quedan dos nodos por unir, se repite la misma operación y se obtiene ya el árbol final:



Para obtener los códigos a usar en la codificación, simplemente se debe recorrer el árbol de la raíz a cada una de las hojas, asignando a cada símbolo el código resultante de unir las etiquetas asociadas a cada uno de los enlaces que se han recorrido.

De esta forma, los códigos finales en el ejemplo son los siguientes:

C(a)=0 C(b)=10 C(c)=11

Y por tanto la cadena original S={aabaacc} quedaría codificada como sigue:

C(aabaacc)= 0 0 10 0 0 11 11

El resultado final de la compresión es que se han empleado 10 bits para codificar los 7 símbolos originales, así que se han usado 10 bits/7 símbolos = 1,43 bits por símbolo.

Si se generase una tabla para futuras codificaciones con una distribución de los pesos, probabilidades igual o parecida. La tabla sería la siguiente:

SÍMBOLO	**CODIFICACIÓN**
A	0
B	10
C	11

Codificación Golomb-Rice

La codificación Golomb es un tipo de codificación de entropía inventada por Solomon W. Golomb que es óptima para alfabetos que siguen una distribución geométrica, lo que significa que los valores bajos son mucho más comunes que los altos. Si se considera la señal como discreta se denomina distribución geométrica, en cambio si se trata de una señal continua, sigue una distribución exponencial.

La codificación Rice es un caso especial de codificación de Golomb. Fue descrita primero (e inventada) por Robert F. Rice. La codificación de Rice se usa como codificación de entropía en una gran cantidad de algoritmos de compresión de imágenes y audio sin pérdida.

AudioPaK se basa en los códigos Golomb-Rice para codificar las muestras salientes de la fase de decorrelación. Se trata de códigos óptimos para distribuciones de carácter exponencial de números enteros positivos.

Dado un único parámetro m, el código Golomb divide un número entero n, en dos partes: la representación binara del resto de n/m y el máximo entero inferior del cociente de dicha división.

Es fácil ver que cuanto mayor es m, más probable es el símbolo y, por tanto, menos bits serán requeridos.

Si m = 2^K, algo que suele ocurrir para codificaciones PCM, la palabra código para n consiste en sus k bits menos significativos, más el número formado por los bits restantes más significativos de n en representación unaria y un bit de parada. La longitud será $k + 1 + \frac{n}{2^K}$.

Pero nótese que la metodología expuesta no es suficiente dado que los códigos de Golomb son útiles únicamente para representaciones de números positivos, y los residuos e_I [n] no siempre lo son, por lo que es necesario elaborar un mapeo como sigue:

$$M(e_i[n]) = \begin{cases} 2e_i[n] & \text{si } e_i[n] \geq 0 \\ 2 \mid e_i[n] \mid -1 & \text{otros} \end{cases} \quad (14)$$

El parámetro k está relacionado con la esperanza matemática de los residuos de manera muy similar a la ya vista para los códigos Rice:

$$k = \log_2(E\{\mid e_i[n] \mid\}) \quad (15)$$

Es un parámetro constante a lo largo de la trama. Puede demostrarse, además, que se puede estimar sin necesidad de acudir a operaciones de punto flotante [2].

Véase un resumen del algoritmo de formación del código de Golomb. Se parte de una muestra x[n] un número entero, que puede ser positivo o negativo. Para codificar este número se divide en cuatro partes diferenciadas y se aplican diferentes operaciones para realizar la codificación. Las cuatro partes son:

- Bit de signo. Se usa 0 cuando es un número positivo y 1 cuando se trata de un número negativo.
- Codificación con K bits del residuo de la división siguiente, Es decir, se codifica el resto de $\frac{|x[n]|}{2^K}$.
- *Overflow*: se añaden tantos ceros consecutivos como resulte el cociente de la división anterior.
- Bit terminación. Al final, se añade un bit igual a 1 para indicar el fin y completar el código.

Véase un ejemplo numérico para aclarar este método, se codifica la muestra x[n]=-73 con K=5.

- En primer lugar, habrá que codificar el signo "-" por tanto se pone un 1 al principio por ser negativo.
- El resto de la división $\frac{|x[n]|}{2^K}$ sería $\frac{|-73|}{2^5}=\frac{73}{32}\rightarrow$ resto=9, por tanto, se codifica el número 9 en binario con 5 bits (K=5), quedaría 01001.
- Se codifica el *overflow* $\frac{73}{32}=2$, como el cociente de la división es 2, se añaden 2 ceros 00.
- Se completa la secuencia con el bit de terminación, se añade un bit igual a 1 final y se completa el código.

Por tanto, el número x[n]=-73, codificado por Golomb-Rice con una K de valor igual a 5 será: 101001001.

2.3.4. Ejemplos de algoritmos de compresión sin pérdidas

A continuación se realiza una breve descripción de algunos algoritmos de compresión sin pérdidas utilizados en la actualidad [2].

- **LTAC.** Se trata de un algoritmo muy particular puesto que es uno de los pocos que utiliza el modelo de codificación con pérdidas para la fase de decorrelación intracanal. El código Rice se usa para codificar tanto los coeficientes de la transformada como el error residual [18].

- **MUSICompress.** Este algoritmo utiliza una estructura predictiva adaptativa. Los datos se representan en punto flotante y el código de entropía es tipo Huffman [19].

- **OggSquish.** Se puede decir que OggSquish en realidad no es exactamente un codificador de audio, sino más bien un esquema basado en DSP que puede codificar en muchos tipos de datos, como puede ser incluso MPEG. En el caso que interesa, se destacan como características principales el hecho de que utiliza predicción lineal con filtros IIR y codificación Huffman[20].

- **Philips.** Se trata de un algoritmo que utiliza un predictor lineal tipo FIR de orden 10, junto con un esquema de codificación Rice [2].

- **Shorten.** Este algoritmo presenta dos versiones. La más simple consta de un predictor FIR con coeficientes enteros. La segunda versión, más sofisticada, emplea predictores FIR de orden 10 que incorporan el algoritmo del error cuadrático medio anteriormente descrito [21].

- **Sonarc**. Utiliza predicción lineal tipo FIR y codificación Huffman. La versión más sencilla utiliza unidades de enteros en punto fijo para calcular los coeficientes de los filtros. La otra versión, más avanzada, utiliza métodos más complejos de cálculo, como ya se expuso en apartados anteriores [22].

- **Waveform Archiver (WA).** Se trata de un algoritmo que utiliza predicción lineal tipo FIR para la decorrelación intracanal. Existen varias versiones, según el parámetro empleado en la codificación Rice, variando éste desde 1 hasta 5. Se ha comprobado que, al aumentar este parámetro, si bien los resultados no mejoran drásticamente, sí lo hacen los recursos de computación necesarios [23].

- **AudioPaK.** Un algoritmo de compresión sin pérdidas muy sencillo pero que genera buenos resultados. Se basa en la utilización de varios predictores, de los que se seleccionará el más adecuado en función de lo óptimo que sea, según filtros FIR. En función del predictor elegido, se transferirá más o menos energía a las altas frecuencias, con el fin de cumplir con el objetivo de espectro plano. Una propiedad importante de AudioPaK es que sus predictores están ajustados para que los cálculos requieran únicamente de sumas, evitando las multiplicaciones, que son las operaciones más costosas [2]. Sus fundamentos teóricos serán expuestos en el Capítulo 2.3.5, puesto que es el punto de partida de nuestras extensiones.

2.3.5. Estudio de un algoritmo sencillo: AudioPak

A continuación se va a presentar AudioPaK [2]. Se trata de un algoritmo de compresión sin pérdidas sencillo, de baja complejidad. Puede ser bastante apropiado para la transmisión de audio por Internet. El esquema de bloques que tiene es el característico visto en la Figura 4, además, aclarar que AudioPak se basa en PCM (*Pulse Code* Modulation).

Una trama PCM es una representación digital de una señal analógica en donde la magnitud de la onda analógica es tomada en intervalos uniformes (muestras), cada muestra puede tomar un conjunto finito de valores, los cuales se encuentran codificados.

Este algoritmo es el punto de partida para este proyecto.

ENVENTANADO

Al igual que otros codificadores de audio sin pérdidas, AudioPaK divide la señal de entrada en tramas independientes. La primera decisión de ingeniería es el número de muestras por cada trama ya que, como es lógico, se trata de un parámetro que afecta al resto del sistema. Un valor estándar y bastante útil para la comunicación con otros sistemas es 192 o múltiplos de este. En general, se puede decir que para una tasa de muestreo de 44.1 KHz, con 16 bits por muestra, tramas de 1152 muestras son las más aceptables.

DECORRELACIÓN INTRACANAL

Esta etapa es crítica en toda compresión puesto que se trata, sin lugar a dudas, de la fase en la que se requiere de una mayor carga computacional. Por lo tanto, para obtener un algoritmo eficiente, es necesario simplificar al máximo sus métodos de decorrelación.

En AudioPaK, esta operación utiliza un método de predicción basado en filtros FIR y, además, adaptativo. Usa también coeficientes enteros, como se puede ver a continuación:

$$\begin{aligned}\hat{x}_0 &= 0\\ \hat{x}_1 &= x[n-1]\\ \hat{x}_2 &= 2x[n-1]-x[n-2]\\ \hat{x}_3 &= 3x[n-1]-3x[n-2]+x[n-3]\end{aligned} \tag{16}$$

Nótese que, efectivamente, se trata de un predictor FIR puesto que los valores se calculan a través de un conjunto finito de muestras en tiempos anteriores. Se puede hablar de unos polinomios característicos, definidos como aquellos en los que al evaluar la muestra enésima, se obtiene el valor $\hat{x}[n]$. En la Figura 9 se observa un ejemplo gráfico de esta aproximación.

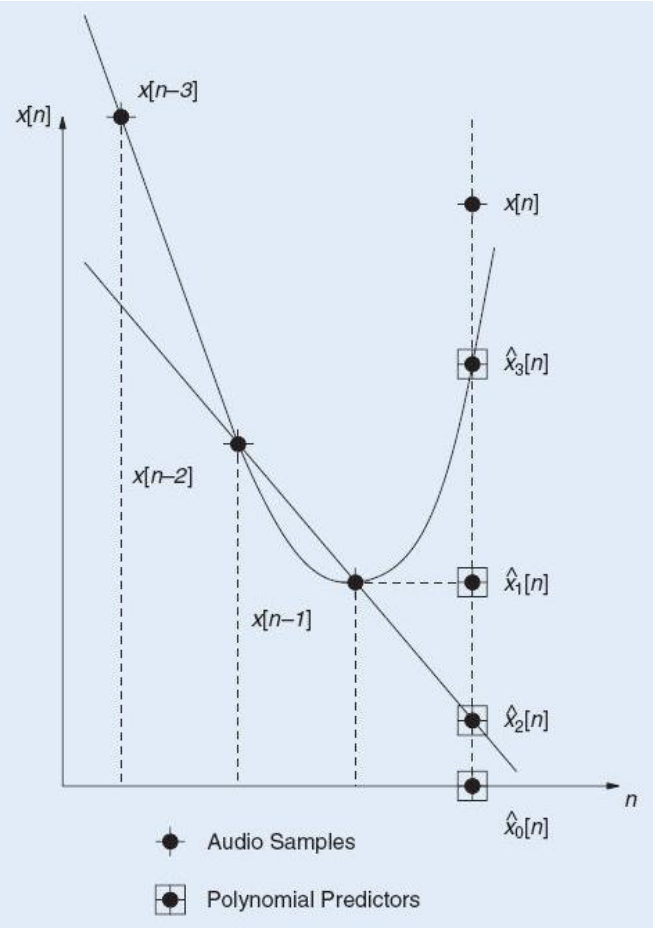


Figura 9: Aproximación polinómica de los predictores usados en la decorrelación intracanal

Pero estos polinomios no sólo tienen propiedades de interés en el ámbito del cálculo de coeficientes, sino también en el cálculo de las señales residuales, puesto que éstas, definidas como sigue: $e_i = x[n] - \hat{x}_i[n]$, pueden ser calculadas de manera recursiva como se muestra a continuación:

$$\begin{aligned} e_0 &= x[n] \\ e_1 &= e_0[n] - e_0[n-1] \\ e_2 &= e_1[n] - e_1[n-1] \\ e_3 &= e_2[n] - e_2[n-1] \end{aligned} \tag{17}$$

En otros términos, se pueden calcular las cuatro señales residuales $e_0[n]$, $e_1[n]$ $e_2[n]$ y $e_3[n]$ simplemente con sumas, sin necesidad de realizar ninguna multiplicación, operación que es computacionalmente mucho más costosa que las operaciones que se han expresado en la Ecuación (17).

El proceso que sigue AudioPaK es repetitivo para cada trama, el codificador calcula los valores absolutos de los cuatro errores residuales, esto lógicamente se debe realizar para cada muestra de la trama. Después se hace una media a lo largo de la trama y aquel predictor cuyo error residual sea menor es el que se usará.

Puesto que únicamente se tienen cuatro predictores, se podrán usar dos bits de información, que serán suficientes para codificar el predictor utilizado en la posterior cabecera de trama.

CODIFICACIÓN DE LA ENTROPÍA

AudioPaK realiza esta etapa de codificación de la entropía empleando la codificación de Golomb-Rice, descrita en el Capítulo 2.3.3.

3.EXTENSIONES PROPUESTAS

En este capítulo de la memoria se van a enunciar las diferentes extensiones que se han estudiado en la codificación, con relación al modelo de codificación de AudioPak que es el que se ha tomado de referencia.

Se han propuesto diferentes extensiones dependiendo de la etapa de codificación:

- Extensión en el enventanado: longitud de trama adaptativa.
- Extensión en decorrelación intracanal: coeficientes variables de predicción.
- Extensión en codificación fuente:
    - Codificación tablas Huffman
    - Codificación mixta Golomb Rice-Huffman

Los diferentes fundamentos teóricos en los que están basadas estas extensiones también se exponen en los siguientes apartados. Se evaluará en la parte de experimentación si las extensiones propuestas del codificador suponen una mejora en la codificación.

3.1. Longitud trama adaptativa

El proceso de enventanado consiste en agrupar un número de muestras consecutivas, como se explica en el estado del arte. Se debe conseguir un equilibrio para decidir el tamaño de trama, si es muy pequeño se tendrán demasiadas cabeceras, en cambio, si la longitud es demasiado grande las propiedades calculadas para la trama no serán las óptimas, por tanto, su compresión será menor que con un tamaño de trama más pequeño. En una trama muy larga no se cumple la premisa de que la señal sea estacionaria a lo largo de la misma, esto es lo que explica que las propiedades calculadas no sean óptimas.

La propuesta que se estudia en este proyecto es realizar un enventanado adaptativo de la señal, es decir, que la longitud de la ventana vaya variando en función de la señal de audio que se está tratando de codificar. De esta forma, se asegura no malgastar bits en codificar pasajes de transición o silencios.

Se tendrá un bloque en la codificación que permita adaptar la longitud de las ventanas que se utilizan, según la señal de entrada.

Desde un punto de vista teórico, la solución óptima, sería que la longitud de estas tramas fuese totalmente arbitraría, pero desde la práctica esta solución no es posible puesto que el conjunto de todas las particiones de un fragmento de miles de muestras de audio y en tramas es de 2 elevado a la longitud de la trama. Además, se tendría que mandar en la cabecera de la trama la longitud exacta de la trama, para cada una de ellas. Por tanto, al incrementar la cabecera, para enviar esta longitud, los bits necesarios para la codificación podrían no reducirse, al neutralizarse con las cabeceras, lo que alejaría del propósito de la compresión.

Existen diversos algoritmos de codificación que este bloque del enventanado lo hacen con ventanas de longitud variable, como MPEG o ATRAC (*Adaptive TRansform Acoustic Coding*), dónde existe un bloque de longitud adaptable que escoge la longitud de las ventanas de acuerdo con las características de la señal. En este caso existen dos modos de ventana: modo corto y modo largo [8].

La solución que se ha estudiado en este proyecto es parecida a la presentada por ATRAC, ya que como se ha explicado un enventanado totalmente adaptativo no sería rentable. Por ello, se ha elegido unas longitudes de ventana preestablecidas, en este caso se tratará de cuatro longitudes de trama posibles.

Desde el punto de vista de la cabecera, sólo se requerirán 2 bits adicionales para indicar cuál de estas cuatro longitudes de ventana ha sido empleada para codificar la trama correspondiente.

A priori se puede pensar, que dos tramas consecutivas con propiedades de codificación o parámetros de codificación parecidos, se podrían unir en una trama más larga, para eliminar una cabecera y seguir con las mismas propiedades de codificación. Con lo que se conseguiría una compresión más alta, ya que se están eliminando los bits correspondientes a esa cabecera. Si se tienen cuatro longitudes de ventana diferentes, este proceso se puede realizar cuatro veces, de tal forma que se obtendrán tramas de longitud L, 2L, 4L y 8L. Lo que quiere decir que en el mejor de los casos, se conseguirían ahorrar siete cabeceras.

Se va partir desde una longitud de trama corta, y fusionarán tramas consecutivas sólo si la longitud global se reduce. Estaremos empleado un proceso *bottom-up,* iniciando con tramas pequeñas, de longitud L, y fusionándolas hasta una longitud 8L, siempre que se reduzca la longitud global.

Basándose en este principio, se decide realizar un preprocesado de la señal. Esto es necesario ya que para poder unir las tramas según lo expuesto, se tienen que conocer los parámetros de codificación de la misma.

En el estudio que se ha realizado tras este bloque de preprocesado se incluye el bloque de longitud adaptable, dónde se decidirá la longitud de la trama. En primer paso de este proceso, consiste en realizar un enventanado con una longitud fija, tras ello se calcularán el número de bits necesarios para codificar la trama, con los diferentes parámetros de codificación.

Tras este preprocesado, cuando ya se tienen el número de bits necesarios para la codificación de la trama para los diferentes parámetros, se realiza el bloque de longitud adaptable, este se trata de un bucle, donde se evalúa si la trama tiene que tener la longitud inicial o por el contrario si se une a la trama siguiente se conseguiría una mejor compresión. Este proceso se repite varias veces hasta evaluar las tramas de longitud 8L.

Una vez decidido la longitud de la trama se podría pasar a los siguientes bloques de la codificación de la señal. Y en la cabecera de cada trama quedaría reflejado, en dos bits, la longitud de la trama.

Esta forma de trabajo a posteriori de conocer los parámetros, puede parecer muy costosa y además aumenta la complejidad del codificador, pero en codificación multimedia, la complejidad del codificador es un tema que no es tan importante. Siempre que el decodificador sea lo suficientemente sencillo y se pueda implementar en tiempo real. El codificador solo tiene que trabajar una vez, por lo que no es tan importante el tiempo y los cálculos necesarios para hacerlo.

En el capítulo de experimentación y resultados, se expondrán los métodos de trabajo empleados más detalladamente, los experimentos realizados y los resultados que se obtuvieron basados en estas extensiones que se proponen para el enventanado.

3.2. Extensión en decorrelación intracanal

Se podría preguntar ahora hasta qué punto este algoritmo en el que se basa AudioPak es bueno. Es importante notar que se trata de un predictor muy simple, por lo que puede ocurrir que los resultados no sean demasiado buenos. Sin embargo, se ha demostrado en diferentes experimentos que no es así y, codificadores más complejos, con operaciones más complejas no dan mejores resultados.

Por ello, se ha decidido mantener este predictor tan sencillo para evitar cálculos más complicados como las multiplicaciones. Además como se basa en filtro FIR con coeficientes enteros, su implementación a nivel hardware de este sistema, es mucho más simple y económica que si fuesen coeficientes fraccionarios, ya que habría que utilizar codificación en punto flotante.

Basándose en este predictor, se realizarán diversas pruebas para comprobar si los coeficientes enteros que emplea son los más adecuados o en cambio habría otros que podrían dar unos resultados mejores. Para ello, se ampliarán los coeficientes del filtro por otros valores de números enteros, y se verán que resultados se obtienen.

Todo este proceso se ve reflejado en la parte de experimentación y resultados, dónde se verá qué coeficientes serán los más adecuados y si éstos coinciden con los valores de AudioPak o son distintos.

3.3. Extensión en codificación fuente

En esta fase del desarrollo del codificador se van a estudiar dos propuestas diferentes para la codificación de la fuente, en la fase de experimentación y resultados, se expondrá la comparativa con el modelo de AudioPak, y el modelo de Golomb-Rice que emplea.

La primera de las extensiones se basa en eliminar la codificación de Golomb-Rice y sustituirla por una basada en la codificación de Huffman, y la generación de tablas de codificación.

La segunda de las extensiones que se estudia, es un codificador basado en una tecnología mixta que decida, para cada trama, si se debe hacer por codificación Huffman o Golomb, decidiendo en cada caso la que de una codificación con un número de bits menor.

3.3.1. Codificación Huffman

La primera de las propuestas que se estudia en este bloque del proyecto es realizar una codificación basada en Huffman, ya que consigue una longitud de código más cercana a la entropía.

La primera opción para realizar esto, fue basarlo en las tablas de codificación MP3, para utilizar unas tablas preestablecidas, pero estas tablas están diseñadas para ser aplicadas en un ciclo interno que realiza el MP3 que no se puede aplicar al diseño que se ha propuesto en este proyecto.

Por tanto, se decidió crear unas tablas propias de codificación basadas en Huffman, nuevas. Estas tablas serán fijas tanto en el codificador como decodificador, y adaptadas a las señales de audio. Según la trama que se esté codificando se aplicará una tabla u otra, adaptándose a la señal.

Dado que para codificar una señal en AudioPak, es necesario especificar en la cabecera de la trama que valor de K se había utilizado para la codificación por Golomb-Rice. En esta parte del estudio se emplearán esos bits de cabecera para indicar que tabla de Huffman es la que se utiliza en la trama. Lo que conlleva a evitar una penalización en los datos de cabecera con bits extra.

Un inconveniente del proceso de decodificación Huffman es que es necesario disponer del árbol a partir del que se codifican los datos, tal y como se ha expuesto en el capítulo de estado del arte. Por lo tanto, no es suficiente con almacenar la cadena final, sino que también hay que comunicar al decodificador las probabilidades de la fuente (o el histograma asociado a los símbolos), de forma que el decodificador sea capaz de reconstruir el árbol, otra alternativa sería transmitir directamente la tabla de codificaciones.

Este proceso de transmisión no sería necesario si las tablas de correspondencia fuesen conocidas, tanto por el codificador como por el decodificador.

En este último argumento es en el que se ha basado la extensión, una vez que se generan las tablas de codificación Huffman, éstas serán fijas y conocidas tanto por el codificador y como el decodificador.

En este punto del proyecto hay que crear las nuevas tablas de Huffman

Como se ha podido observar, con el desarrollo del ejemplo del estado del arte, para poder generar las tablas, es necesario conocer el histograma o las probabilidades de los símbolos que se van a codificar. Este es el punto más complicado del problema, puesto que para cada archivo de sonido, el histograma será diferente.

Para abordar esta problemática se ha decido hacer un estudio previo, sobre los diferentes histogramas de diversos archivos de audio y juntarlos o estudiarlos como un conjunto de datos o un histograma común. Los datos y el procedimiento exacto se mostrarán en la parte de experimentación.

Una vez obtenido ese histograma común, se trata de realizar las diferentes tablas de Huffman. La forma y el proceso de distribución de las tablas de Huffman y el número de tablas que se van a proponer están también descritos en los siguientes capítulos de experimentación.

Resumiendo esta propuesta de extensión, se va a emplear una codificación basada en los algoritmos de Huffman en contraposición del algoritmo de Golomb-Rice que emplea AudioPak. Para ello se tienen que crear unas tablas de codificación fijas, para que codificador y decodificador las conozcan y las empleen, sin necesidad de ser transmitidas. Estas tablas se van a desarrollar a través de un estudio sobre un histograma común de varios archivos de audio o la suma de los histogramas de cada archivo.

3.3.2. Codificación mixta: Huffman-Golomb

En este apartado se pretende maximizar la compresión que aportan las dos técnicas usadas Huffman y Golomb Rice, para ello se ha planteado el desarrollo de una codificación mixta que emplee ambos métodos.

Esta codificación mixta consiste en emplear codificación Huffman o Golomb-Rice según sea más conveniente. Es decir, para cada una de las tramas se estudiarán cada uno de estos dos métodos y se evaluará cual es el que proporciona una compresión mayor de la señal y éste será el método que se emplee para codificar esa trama.

Esta codificación es más costosa computacionalmente puesto que se tiene que calcular la longitud que el código tendrá después de la codificación, tanto para Golomb como para Huffman y elegir la que sea menor en cada caso.

Al igual que en el enventanado se explicó, esta medida no afectará a la decodificación, puesto que en la propia cabecera vendrá indicado el método empleado y por tanto la decodificación será directa.

Además, existe un inconveniente, habrá que incluir un nuevo bit de cabecera de trama para indicar si las muestras de la misma se van a codificar mediante codificación de Golomb o por el contrario por codificación por tablas de Huffman. En el capítulo de experimentación y resultados se comprobará si esta codificación mixta aporta alguna mejora o si por el contrario, la penalización del bit adicional de la cabecera de trama provoca una peor compresión que con cualquiera de los dos métodos por separado.

En este punto hay que destacar que en este proyecto se estiman el número de bits que se tienen que emplear para codificar, por tanto no es tan costoso como codificar.

4.EXPERIMENTACIÓN Y RESULTADOS

En este capítulo se describirán los diferentes experimentos que se han llevado a acabo y se expondrán los resultados obtenidos. Todos los experimentos realizados se han comparado con la codificación AudioPak descrita en capítulos anteriores. Tanto la codificación AudioPak como las extensiones que se proponen han sido realizadas en un codificador de audio que se ha desarrollado en MatLab. Este codificador implementado no realiza la codificación de la señal, esto sería un proceso posterior, sino que calcula el número de bits que supondría realizar la codificación con los métodos o extensiones empleadas.

Para realizar los experimentos se eligieron 16 canciones de diferentes estilos y diferente duración. En la Tabla 2 se pueden observar las canciones empleadas, así como su género, duración, artista y título de la canción. Por mayor simplicidad para los experimentos y la visualización de sus resultados las canciones se han numerado y en las tablas posteriores y gráficos se distinguirán por su número de canción, es decir, la columna que aparece a la izquierda de la tabla.

Canción	Género	Duración	Artista	Título
011	*Rock* alternativo	5:17	Muse	*Showbiz*
012	*Minimal techno*	3:58	Caribou	Jamelia
013	*Rock*	4:37	Extremoduro	So payaso
014	*Indie rock*	2:23	Bombay Bicycle club	*Slow song (will you, won't you)*
015	*Hip Hop*	3:30	Black Eyed Peas	*Bebot*
016	*R&B*	3:47	Black Eyed Peas	*Where Is The Love*
017	*Reggae*	7:07	Bob Marley	*No Woman No Cry*
018	Merengue-Bachata	4:35	Celia Cruz	La Vida Es Un Carnaval
019	Electronica	2:59	Gorillaz	*Crystalised*
020	*Nu-metal*	3:33	Limp Bizkit	*Rollin*
021	*Jazz*	4:41	Louis Prima	*I'm just a gigolo*
022	Salsa	6:04	Mark Anthony	Vivir lo Nuestro
023	Ska	3:00	Obrint pas	*Viure*
024	*Techno*	2:17	Dj	Sesión aniversario
025	Tango	3:42	Shakira	*Objection*
026	*Synth pop*	3:54	Muse	*Undisclosed Desires*

Tabla 2: Índice de canciones

Todas las canciones se han codificado en primer lugar con AudioPak, y la longitud o el número de bits que se emplea para su codificación servirán como medida para evaluar las distintas extensiones que se propusieron.

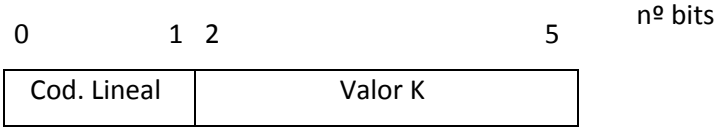
El capítulo de experimentación se va a estructurar de la siguiente manera:

- Descripción y experimentos de los parámetros básicos, con los que se van a comparar las extensiones, experimentos con AudioPak.
- Enventanado: explicación y experimentos realizados con relación a la longitud de trama variable.
- Decorrelación intracanal: experimentos para ajustar los coeficientes de la codificación lineal.
- Codificación fuente: creación tablas Huffman, experimentos sobre codificación con estas tablas y experimentos sobre codificación mixta Golomb-Rice y Huffman.
- Extensión total: las extensiones que resulten mejoras se fusionan, en este apartado, para realizar experimentos sobre cómo sería el codificador mejorado.

Todos estos subcapítulos se estructuraran de manera parecida, se evaluará la cabecera de la trama, se explicarán los experimentos realizados, se mostrarán los valores obtenidos y se obtendrán conclusiones.

AUDIOPAK

En primer lugar, se va a describir la cabecera de las tramas, porque emplea un papel muy importante ya que modificarla es ampliar el número de bits totales en la codificación de la canción. Una de las decisiones principales es la cabecera de trama, en la codificación con AudioPak se necesitan al menos 2 bits para indicar el codificador lineal empleado y se necesitan más bits para indicar el valor que adquiere K en la codificación, en este proyecto se ha codificado el valor de K con cuatro bits, es decir, K variará entre los valores 0 y 16. Por tanto, para AudioPak se tendrá una cabecera de 6 bits: dos para indicar el codificador lineal y 4 para indicar el valor de K.



Para realizar la codificación AudioPak se tiene que definir también un tamaño de longitud de trama, el número de muestras que tendrá cada trama. En este caso se ha realizado un pequeño experimento para ver cuál era la longitud de trama óptima para este codificador.

Se han codificado las señales de audio para distintas longitudes de trama, el siguiente gráfico indica el número de bits empleados en la codificación de la señal de audio, para diferentes longitudes de trama, desde 16-256, en pasos de 16.

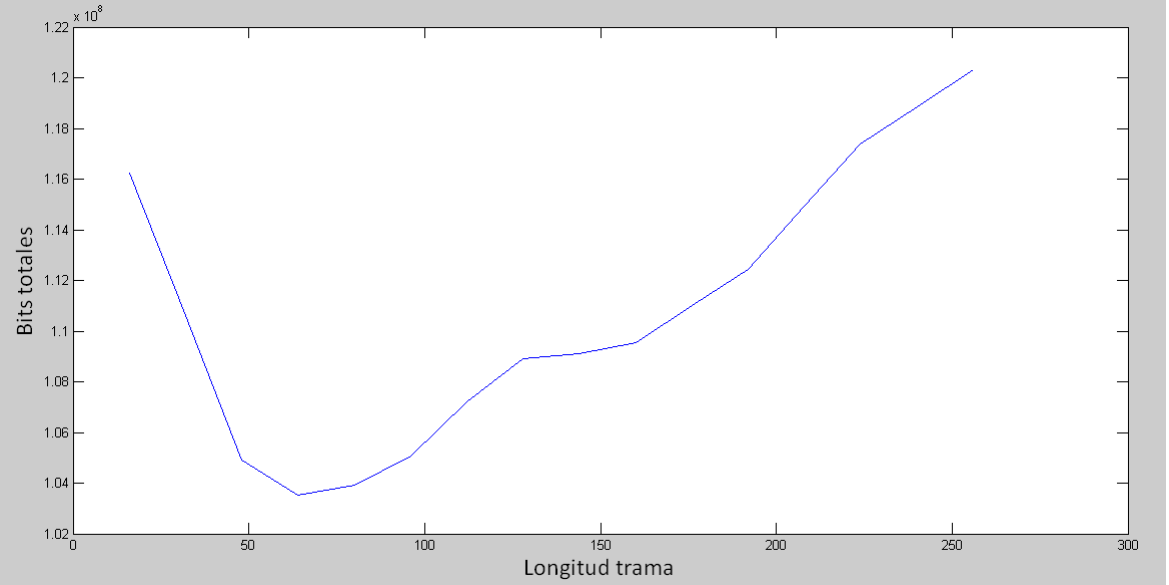


Figura 10: Gráfica longitud de trama

Por tanto, como se ve en el resultado la longitud de trama que se emplea, a lo largo del proyecto, será L=64 muestras. A partir de este punto ya se puede codificar la canción empleando un codificador basado en AudioPak. Este codificador, como se ha indicado al inicio está creado en MatLab. En primer lugar, Se divide la canción en tramas de 64 muestras, se aplican los 4 predictores de AudioPak predeterminados y se codifica en la secuencia de 2 bits de la cabecera cuál de los 4 proporciona un mejor resultado en términos de residuo. A continuación, se codifica la secuencia de audio de acuerdo con el algoritmo de Golomb-Rice, para cada trama se calcula el parámetro K óptimo, con la Ecuación (15) y se codifica este valor en los 4 bits de cabecera.

En la Tabla 3 se muestran los valores de longitud de archivo en bits, de las diferentes canciones, tras haber sido codificadas empleando AudioPak.

Canción	***nº bits***
011	166333502
012	108163625
013	149631363
014	64169821
015	102934754
016	112924647
017	186089707
018	146226539
019	68279730
020	119979313
021	140533787
022	174896997
023	104738209
024	71523444
025	119465932
026	94797276
PROMEDIO	109163556

Tabla 3: Número de bits empleados con la codificación AudioPak

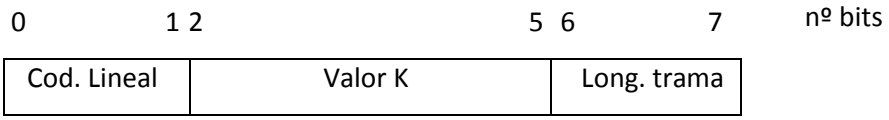
En las secciones siguientes se van a ir desarrollando los experimentos de las extensiones que se han introducido en los capítulos anteriores. En todas las tablas se añade una fila extra denominada “promedio”, que contiene el valor de la media aritmética de todas las canciones.

4.1. Enventanado

En este apartado se explicará y se verán los resultados de la extensión que se ha propuesto para el enventanado, es decir, la longitud de trama variable. Se han ilustrado varias tablas y esquemas que ayudarán a la comprensión de los mecanismos empleados.

Como ya se explicó en capítulos anteriores, la longitud de trama en AudioPak es fija, en este caso 64 muestras, y con la extensión se pretende que esa longitud sea adaptable a las características de la señal, es decir, que se pueda tener una longitud de trama variable. Se podrán tener tramas de 64 muestras (L), 128 muestras (2L), 256 muestras (4L) y 512 muestras (8L). Puesto que hay 4 posibles longitudes de trama, esto implica que se deben introducir dos bits adicionales en la cabecera de trama para codificar la longitud de la misma, por lo que la cabecera pasará de tener 6 bits a tener 8 bits.

El aumento de la longitud de cabecera penalizará el número de bits totales ya que se está hablando de cabeceras de trama, por tanto, se aumentan 2xnºtramas bits en las distintas señales de audio. Al final del capítulo, en los experimentos realizados, se puede observar si esta penalización se compensa con el número de bits que el codificador es capaz de reducir al emplear tramas de diferentes longitudes. La cabecera de trama, en este apartado, quedaría de la siguiente manera:



Como se enunció en el Capítulo 3.1 se trata de un proceso a posteriori, es decir, el tamaño de la longitud de trama se decidirá tras ver si es rentable, en términos de bits, o no realizarlo. Para decidir el tamaño de trama más adecuado se realizan una serie de pasos descritos a continuación.

En primer paso de este proceso, consiste en realizar un enventanado con una longitud fija L=64 muestras, tras ello se calculará el número de bits necesarios para codificar la trama, con los diferentes parámetros de codificación. Este resultado se almacena en una matriz de dimensiones 4x16, es decir, 4 codificadores lineales por 16 valores de K. No se eligen los parámetros óptimos para cada trama, sino que se evalúa o se calcula el número de bits que son necesarios para todos los codificadores lineales y para todos los valores de K, para cada par codificador lineal y valor de K.

En la Tabla 4 se puede observar un ejemplo de esta matriz, en concreto se trata de la trama número 100.000 de la canción 015, la matriz se muestra traspuesta para que se pueda visualizar correctamente.

C.Lineal / Valor K	1	2	3	4
1	490049	14080	15393	26680
2	245174	7189	7843	13489
3	122770	3774	4101	6923
4	61602	2097	2263	3677
5	31045	1292	1376	2084
6	15798	923	962	1322
7	8207	767	795	977
8	4444	726	747	841
9	2593	740	759	807
10	1699	787	799	823
11	1290	845	851	863
12	1112	906	909	915
13	1053	968	969	972
14	1053	1031	1031	1032
15	1095	1095	1095	1095
16	1159	1159	1159	1159

Tabla 4: Matriz correspondiente a la canción 015; trama 100.000

Como se observa en la Tabla 4 se almacena el número de bits necesarios para la codificación de la trama en la que se encuentra para cada par de valores [codificador lineal, valor K]. El valor mínimo de esta tabla, marcado en color gris, da los valores óptimos de los parámetros, en este caso, la codificación óptima se produciría con el codificador lineal 2 y un valor de K=8. Esta matriz la se calcula para una de las tramas de la señal de audio, quedando todas ellas almacenadas en un tensor de tamaño 4x16xnºtramas. El siguiente gráfico muestra o intenta clarificar visualmente cómo sería el tensor.

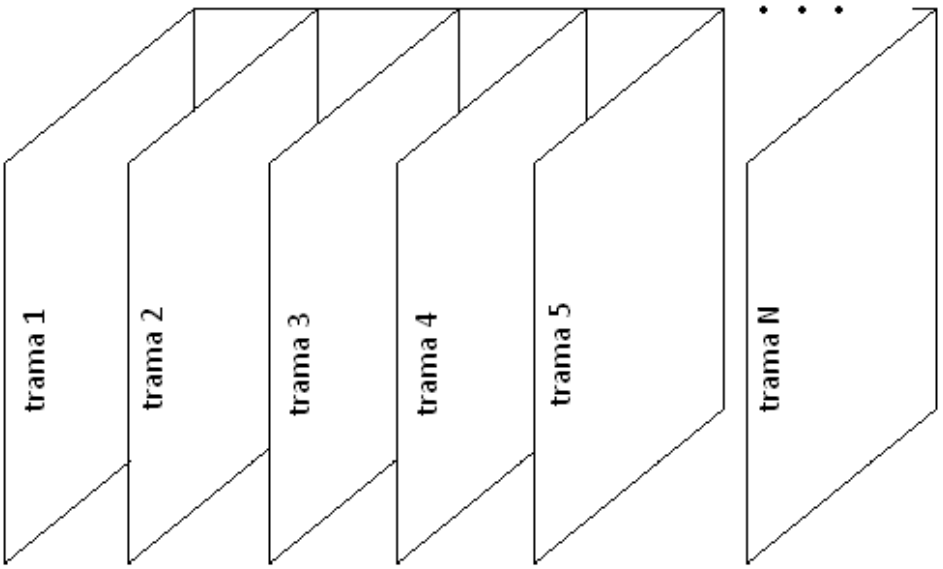


Figura 11: Aclaración gráfica de un tensor

Tras este preprocesado, se tendrán almacenados en este tensor, el número de bits necesarios para la codificación de la trama para los diferentes parámetros y en todas las tramas de la señal de audio. Esta forma de almacenar los datos se realiza porque se va a estudiar si es mejor fusionar tramas consecutivas, aunque no tengan los mismos parámetros, sólo para ver si lo que se pierde al unirlas compensa la eliminación de una o varias cabeceras.

Desde un punto de vista teórico la decisión de la longitud de trama se puede hacer siguiendo dos modelos, *bottom-up* o *top-down.*

La estrategia *top-down*, consiste en tomar longitudes de trama largas e irlas dividiendo para averiguar si es mejor en términos de longitud final. En cambio, la estrategia *bottom-up* consiste en iniciar con tramas de longitud inferior e intentar llegar a una más larga, es decir, fusionarlas si la longitud final extensión.

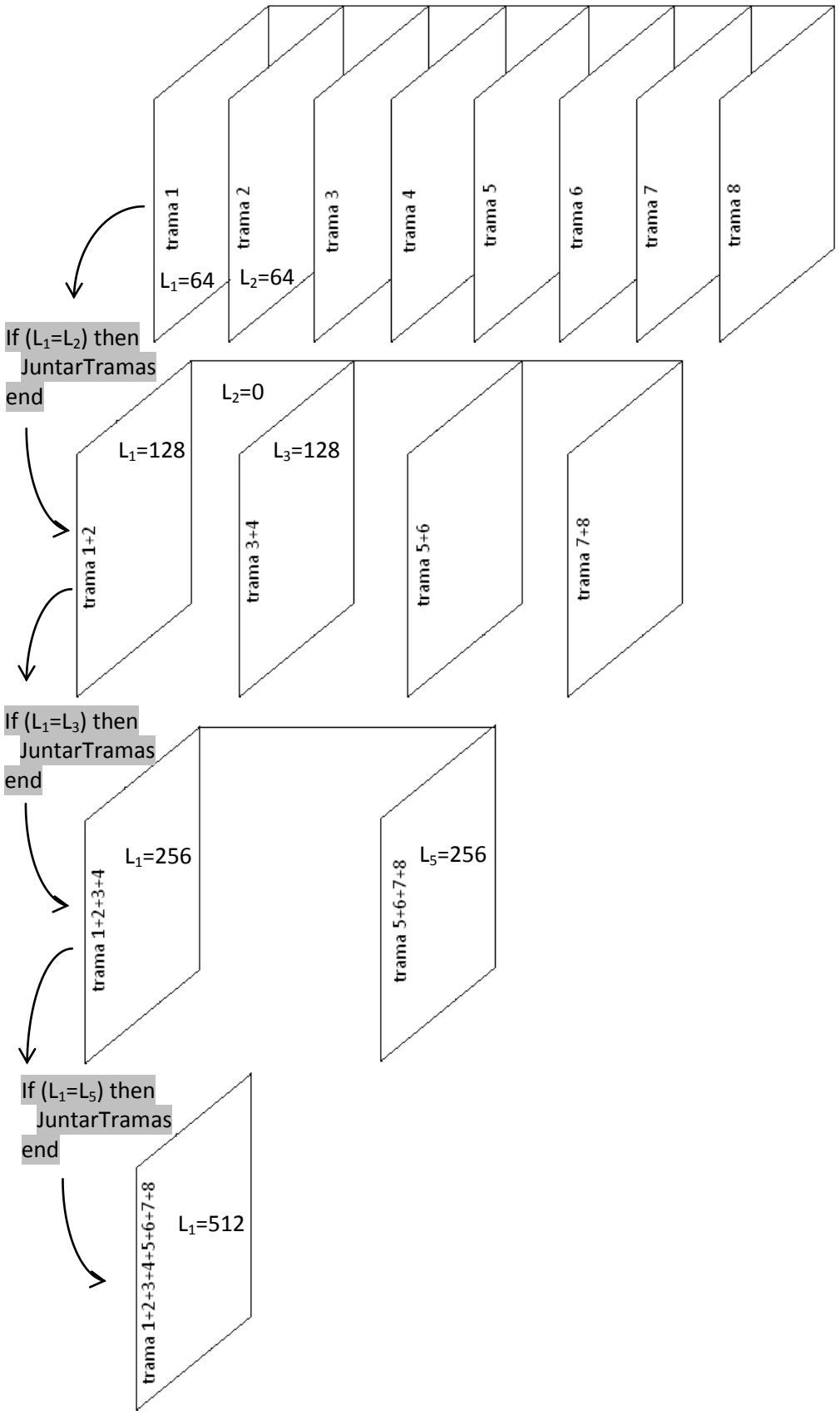
En este proyecto, como ya se mencionó anteriormente, se emplea la estrategia *bottom-up*. Partiremos de tramas de 64 muestras y las fusionaremos en caso que la longitud global disminuya.

Se realiza el bloque de longitud adaptable, este se trata de un bucle, donde se evalúa si la trama tiene que tener la longitud inicial o por el contrario si se une a la trama siguiente se conseguiría una mejor compresión. Este proceso se repite varias veces hasta evaluar las tramas de longitud 8L.

Es decir, se ha desarrollado un bucle que vaya cogiendo las tramas de 8 en 8, ya que 8L es el valor máximo que se ha decidido puedan tomar la longitud de las tramas. En primer lugar, se calcula el número de bits óptimo con el que se codificaría cada trama, esto es el valor mínimo de la matriz que se ha presentado anteriormente, Tabla 4.

La idea principal es ver si la suma de esos valores mínimos puede ser reducida juntando las tramas, en tramas de longitud mayor. Para esto se suman por pares las 8 matrices que se introducen en el bucle, es decir, se sumaría la matriz 1+2, la 3+4, la 4+5, la 5+6 y la 7+8. Obteniendo cuatro nuevas matrices, cuatro nuevas súper-tramas, de estas se calcula el mínimo y se le resta el valor de una cabecera (8 bits), se compara este valor mínimo que ha salido, con la suma de los valores óptimos de las tramas por separado. Si el valor de la matriz suma es menor, se seguiría en el bucle hacia la siguiente unión, así sucesivamente hasta completar la longitud 8L. En el momento que juntar las tramas no reduzca el número de bits, no se sigue en el bucle.

La Figura 12 intenta clarificar la explicación anterior:



NIVEL 1:

8 tramas longitud 64

8 matrices

8 valores mínimos

NIVEL 2:

Tramas longitud 128

Sumamos matrices, calculamos el mínimo de las mismas. Si cumplen la condición de igualdad.

Si éste es inferior a la suma de los dos valores mínimos anteriores, continuamos.

Sólo continúan las tramas que cumplan este requisito.

NIVEL 3:

Igual que el anterior

NIVEL 4:

Igual que el anterior.

Figura 12: Niveles de tramas para la longitud variable

Las tramas sólo podrán unirse como muestra el esquema, y sólo se unirán cuando el mínimo de la matriz suma tenga un valor inferior a la suma de los valores mínimos por separado.

En el nivel 1 se encuentran las tramas de longitud 64 muestras, en el nivel dos es una longitud de 128 muestras, en el nivel tres una longitud 256 muestras y por último el nivel 4 muestra las tramas de longitud 512 muestras.

Podría suceder que algunas de estas 8 tramas puedan unirse y otras no, es decir, se puede tener un grupo de tramas con diferentes longitudes, pertenecientes a diferentes niveles. Aunque siempre se tendrá que seguir el esquema anterior, es decir, la trama 4 nunca podrá unirse a la trama 5, a no ser que se esté en el nivel 4, es decir, en la supertrama con longitud L=512 muestras. En la siguiente figura se muestra un ejemplo de cómo podría ser esta agrupación.

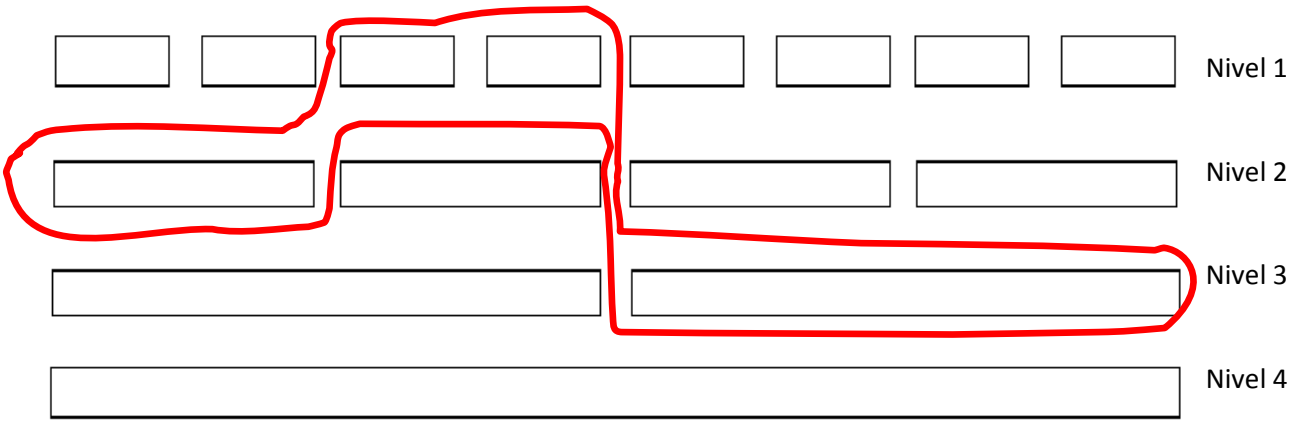
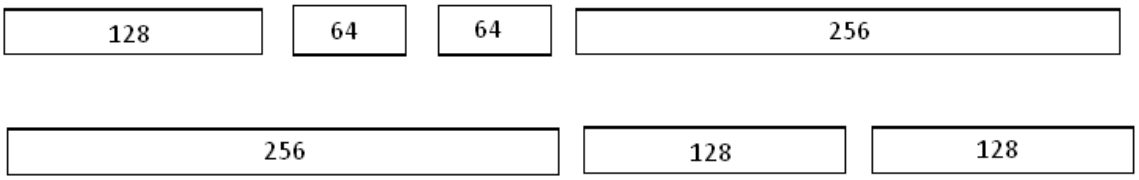


Figura 13: Niveles y agrupaciones para longitud de trama

De esta forma, pueden aparecer supertramas que formasen diferentes patrones, por ejemplo:



…

Tras realizar este bucle en toda la señal de audio, se obtiene como respuesta una nueva matriz que indica la longitud que tiene la trama (L, 2L, 4L o 8L), los parámetros K y el codificador lineal, y el número de bits resultantes de codificar esa trama a la longitud indicada, con la K y el codificador lineal indicados.

En la Tabla 5 se puede observar una parte de esta matriz resultado, se trata de la canción 018 de las tramas 5026 a la trama 5030.

	5026	***5027***	***5028***	***5029***	***5030***
Longitud trama	256	64	64	128	512
Bits	3117	787	779	1602	6244
Cod. Lineal	3	3	4	3	3
K	9	9	9	9	9

Tabla 5: Matriz resultado tras calcular longitud de trama. Canción 018, tramas 5026-5030

En este ejemplo, la trama 5026 de la canción 018, tiene una longitud de trama L=256 muestras, está codificada con el 3er codificador lineal y con una K=9, con estas características, la trama se codifica en 3117 bits.

Además, al tener tramas con una longitud mayor, se disminuye el número de tramas totales. En la Tabla 6 se muestra esta disminución del número de tramas para cada una de las canciones.

Canción	*AudioPak*	*Tramas Adaptativas*
011	218520	112951
012	164466	84505
013	191304	79322
014	99054	75696
015	144936	95030
016	156906	98277
017	294624	200214
018	190026	88853
019	123372	86241
020	147150	63195
021	194202	100379
022	251460	164929
023	124416	50894
024	94554	48580
025	153648	71083
026	161244	86759
PROMEDIO	**169367**	**94181**

Tabla 6: Número de tramas empleadas en AudioPak vs tramas empleadas con longitud variable

Tras observar los resultados, se observa que las tramas que se necesitan para codificar con AudioPak, son aproximadamente el doble que las tramas que son necesarias con la extensión propuesta de longitud adaptativa. Esto conlleva que se están ahorrando muchas cabeceras, aunque en las que existan se penalicen con dos bits adicionales.

Una vez decidida la longitud de la trama se puede pasar a los siguientes bloques de la codificación de la señal. Y en la cabecera de cada trama quedaría reflejado, en dos bits, la longitud de la trama.

Para ver si esta medida va a resultar satisfactoria se calculan el número de bits totales necesarios para codificar la canción con este modelo de longitud adaptativa y se comparan a los que se emplearían para AudioPak.

En la Tabla 7 se muestran los resultados obtenidos:

Canción	*AudioPak*	*Tramas Adaptativas*	*Diferencia*	*Porcentaje (%)*
011	166333502	166124843	208659	0,1254%
012	108163625	108002895	160730	0,1486%
013	149631363	149302231	329132	0,2200%
014	64169821	64231472	-61651	-0,0961%
015	102934754	102932353	2401	0,0023%
016	112924647	112901078	23569	0,0209%
017	186089707	186123016	-33309	-0,0179%
018	146226539	145985153	241386	0,1651%
019	68279730	68303957	-24227	-0,0355%
020	119979313	119745481	233832	0,1949%
021	140533787	140365202	168585	0,1200%
022	174896997	174897931	-934	-0,0005%
023	104738209	104542263	195946	0,1871%
024	71523444	71442250	81194	0,1135%
025	119465932	119265758	200174	0,1676%
026	94797276	94666005	131271	0,1385%
PROMEDIO	**120668040**	**120551993**	**116047**	**0,09%**

Tabla 7: Comparación AudioPak vs extensión con tramas adaptativas

En la tabla anterior, la columna de AudioPak representa el número de bits necesarios para la codificación empleando este algoritmo. La columna de tramas adaptativas, representa el número de bits necesarios para la codificación con la extensión de la longitud de trama adaptativa. Diferencia, es la mejor de bits que existe al usar tramas adaptativas en lugar de AudioPak, es decir, la diferencia entre el número de bits empleados al usar AudioPak menos el número de bits empleados con la extensión; es decir, si el número es positivo significa que es mejor emplear tramas adaptativas, si por el contrario es negativo significa que no es conveniente usarlas.

Porcentaje es el porcentaje que representa la diferencia con respecto a la codificación AudioPak, se calcula mediante la Ecuación (18):

$$\text{Porcentaje} = \frac{\text{Diferencia}}{\text{AudioPak}} x100 \quad \textbf{(18)}$$

Se puede observar en los resultados que en la mayoría de los casos se consigue reducir el número de bits empleados, pese a tener una penalización de dos bits adicionales en la cabecera que indican la longitud de la trama.

Los casos negativos no implican ningún problema, puesto que siempre se puede volver a modelo sencillo de AudioPak, es decir, el codificador evalúa si es mejor el modelo sencillo o el de las extensiones. Sólo hay que añadir un bit adicional a la cabecera del archivo para indicar si se usa el modelo sencillo o con las extensiones. Lo que quiere decir, que estos casos negativos se codificarían con el modelo sencillo de AudioPak.

Por consiguiente, se puede decir que si se emplea la extensión propuesta de longitud de trama adaptativa, en lugar de una longitud de trama fija, se consigue reducir, mayoritariamente, el número de bits empleados para la codificación de una señal de audio.

4.2. Decorrelación intracanal

En este capítulo se verán los experimentos realizados para evaluar la decorrelación intracanal o redundancia temporal. Ya se ha visto que AudioPak emplea unos predictores lineales muy sencillos y que ha dado muy buenos resultados. Por tanto, los experimentos se van a basar en ellos para intentar mejorar esta etapa.

Se ha estudiado para las diferentes canciones qué predictores son los más empleados, en este gráfico se ve los que más se emplean.

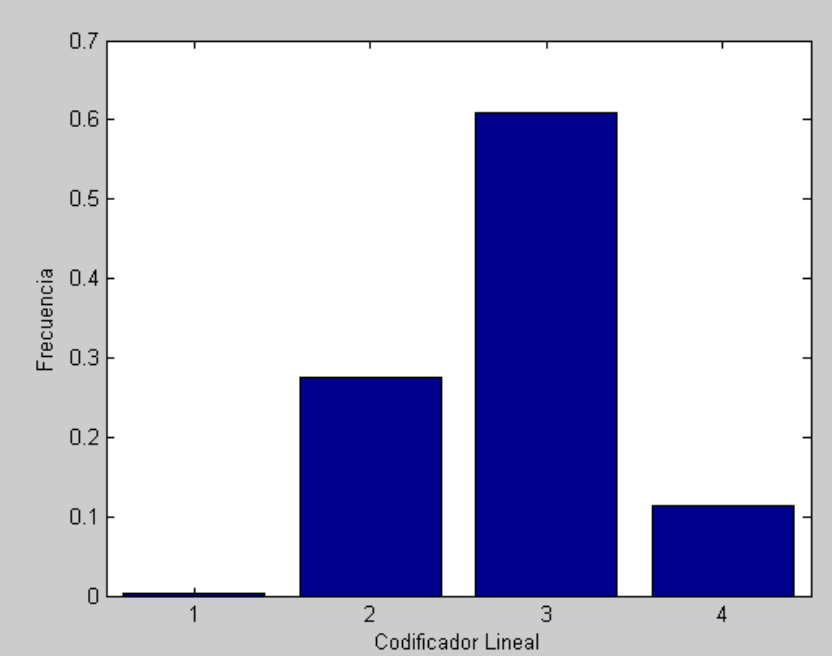


Figura 14: Gráfica de la frecuencia de los predictores lineales

Los predictores lineales que más emplean son el predictor de orden 1 y el de orden 2.

$$\hat{x}_1 = x[n-1]$$
$$\hat{x}_2 = 2x[n-1] - x[n-2]$$

Se van a realizar diversas pruebas para comprobar si los coeficientes enteros que emplea son los más adecuados o en cambio habría otros que podrían dar unos resultados mejores. Para ello, se ampliarán los coeficientes del filtro por otros valores de números enteros, y se verá que resultados se obtienen.

Los coeficientes enteros actuales son [0 1] para el codificador 1 y [2 -1], para el codificador 2. Se han hecho pruebas con otros coeficientes enteros. En la siguiente tabla se ve el porcentaje de tramas que determinan qué predictor es el más adecuado.

X[n-1] / X[n-2]	**-1**	**0**	**1**	**2**
-1	0,150%	0,215%	0,589%	47,588%
0	0,000%	0,000%	32,565%	4,951%
1	2,859%	0,580%	0,158%	4,386%
2	1,857%	0,250%	2,350%	1,501%

Tabla 8: Variación de coeficientes enteros en los filtros de codificación lineal

Según lo que se puede observar en la Tabla 8, los valores que más se emplean en la codificación lineal, son los que provocan un menor residuo, son los correspondientes a los valores que tenían los predictores de AudioPak.

Esto tiene sentido puesto que tal y como se ha visto, en la Figura 9, estos coeficientes tienen un significado geométrico.

Por tanto, en este apartado no se ha conseguido ningún tipo de mejora, puesto que los predictores lineales de AudioPak aportan un buen compromiso entre simplicidad de diseño y resultados.

4.3. Codificación fuente

En este capítulo se van a desarrollar dos extensiones en relación con la codificación fuente, en primer lugar, se realizará la fase de codificación fuente exclusivamente con codificación Huffman. Después, se realizará la combinación de codificación Golomb-Rice y tablas de Huffman.

Como se enunció en el Capítulo 3.3.1 se ha basado en la idea del MP3, dónde se crean unas tablas de Huffman, conocidas tanto por el codificador como el decodificador, que da la correspondencia entre los símbolos y su codificación binaria. Por tanto estas tablas serán fijas y no variarán ni habrá que transmitirlas. En este apartado se va a explicar el desarrollo seguido para la creación de esas tablas.

Como no es conveniente añadir más bits a la cabecera de trama, para no penalizarla más, se van a crear 16 tablas de Huffman, se necesitan emplear 4 bits de cabecera, los mismos que se emplearían para codificar el valor de K, empleado por AudioPak. Por tanto, se aprovechan los 4 bits de cabecera que indicaban del valor de K, para codificar el valor de la tabla de Huffman correspondiente.

La codificación de Huffman se basa en la frecuencia de aparición de los símbolos a codificar, como ésta no es conocida a priori, se tendrá que hacer un estudio de estas frecuencias de aparición de los símbolos.

En primer lugar, se realiza el histograma de todas las canciones que se han elegido para este proyecto, el histograma se realiza tras haber realizado la codificación lineal, es decir, con los símbolos que habría que codificar.

En la Figura 15 se puede apreciar que la mayoría de los símbolos corresponden a valores en las posiciones más bajas.

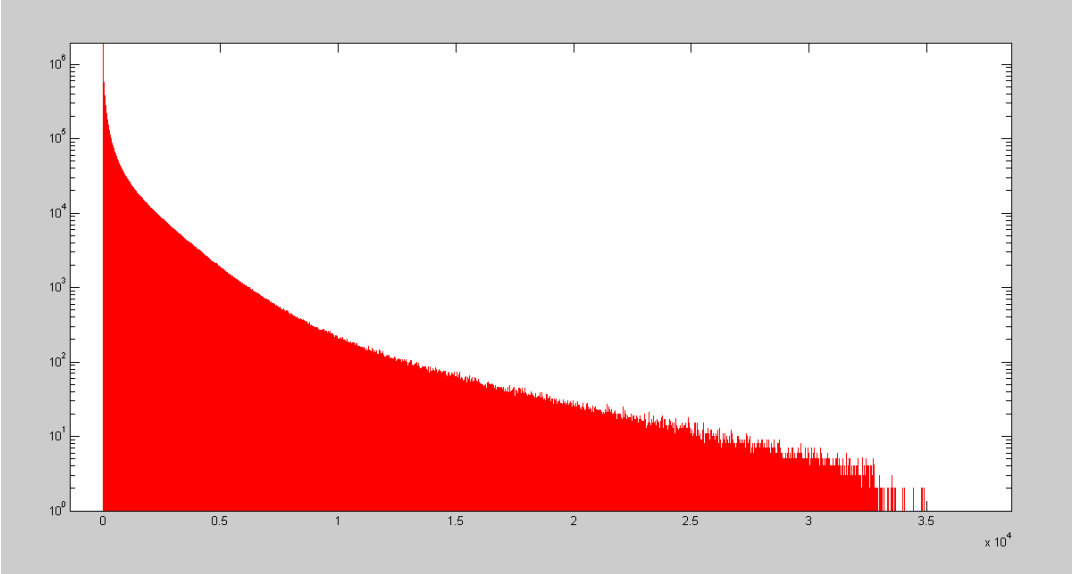


Figura 15: Histograma, ejes logarítmicos, de los archivos de audio

Se ha realizado una representación logarítmica del eje y, para poder apreciar mejor la distribución de los símbolos, ya que la frecuencia de aparición de los símbolos en zonas bajas es mucho más alta que la frecuencia de aparición de los símbolos en zonas altas, y la función que seguiría esta distribución tiende a converger.

Se tienen que crear 16 tablas de Huffman, éstas tienen que centrarse en la distribución que ha dado el histograma, por tanto, se emplearán más tablas para codificar la zona de valores bajo y menos tablas que cubran los valores más altos. La Tabla 9 muestra la distribución de las tablas que se han elegido, y el rango que alcanza cada una de ellas.

	Rango valores
Tabla 1	0-1
Tabla 2	0-3
Tabla 3	0-7
Tabla 4	0-15
Tabla 5	0-31
Tabla 6	0-63
Tabla 7	0-128
Tabla 8	0-255
Tabla 9	0-511
Tabla 10	0-1023
Tabla 11	0-2047
Tabla 12	0-4095
Tabla 13	0-8191
Tabla 14	0-16383
Tabla 15	0-32767
Tabla 16	0-65535

Tabla 9: Rango de las tablas Huffman

Aún no se generan la tablas, pero se puede observar que la tabla 1 al tener sólo dos valores 0 y el 1 sólo necesitará un bit para codificarlos, cuanto más se aleje de esta tabla más bits serán necesarios para codificar los diferentes símbolos.

Se empleará una tabla u otra dependiendo del valor máximo que tengan los símbolos de cada trama, por ejemplo, si el valor máximo a codificar en una trama es el 6, toda la trama la se podrá codificar con la tabla 3. También podrían ser codificados con las tablas de la 4-16, pero es la tabla 3 la que dará una codificación óptima para esos valores.

Por tanto, se empleará cada una de las tablas según el valor máximo a codificar, ya que serán las que menor número de bits necesiten. Para poder calcular bien las tablas de Huffman, se necesitan precisar las frecuencias de los símbolos, si no se van a emplear todas las tablas por igual, sino que dependen del valor máximo de trama, lo mejor es calcular nuevos histogramas. Se tendrán que calcular 16 histogramas correspondientes a las 16 tablas, además, se podrá observar cuales son las tablas más empleadas. Es decir, cada trama contribuirá a crear el histograma solamente de la tabla que sea la óptima para ella.

Desde la Figura 16 hasta la Figura 19 se muestran algunos de los histogramas que se van a emplear para la creación de las tablas. En este caso se han elegido la tabla 5 [0-31], la tabla 9 [0-511], la tabla 13 [0-8191] y la tabla 16 [0-65535].

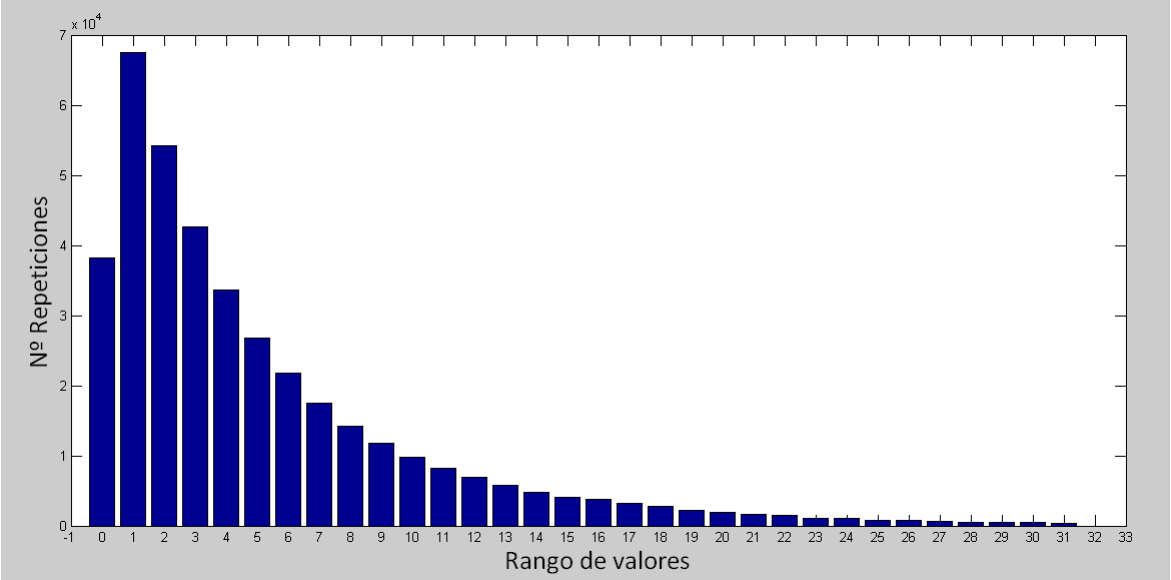


Figura 16: Histograma de la Tabla 5 de Huffman

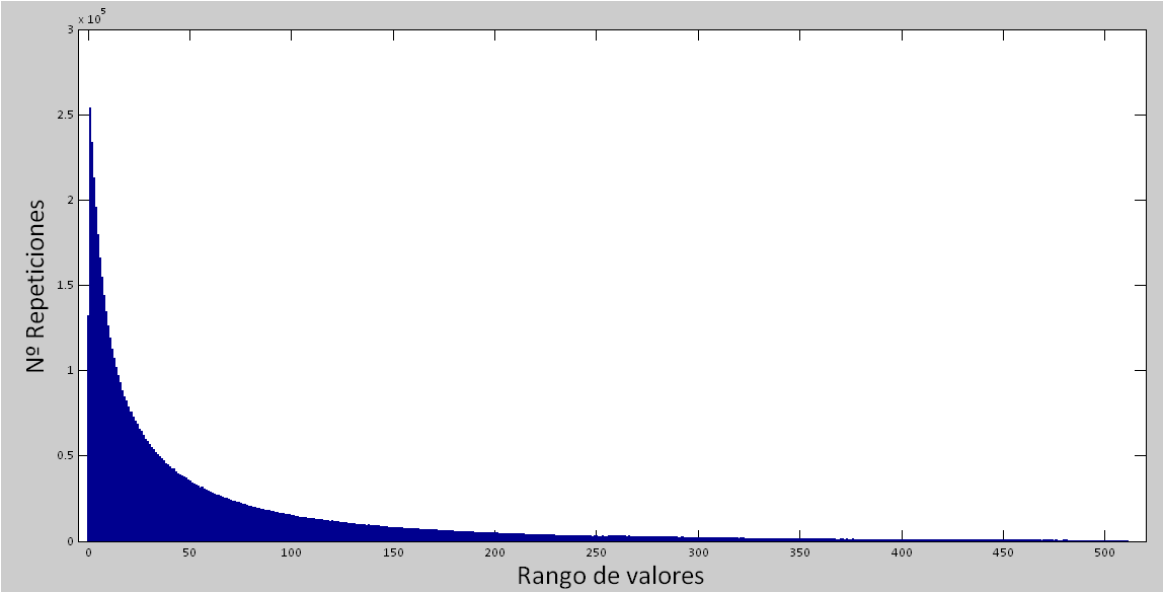


Figura 17: Histograma de la Tabla 9 de Huffman

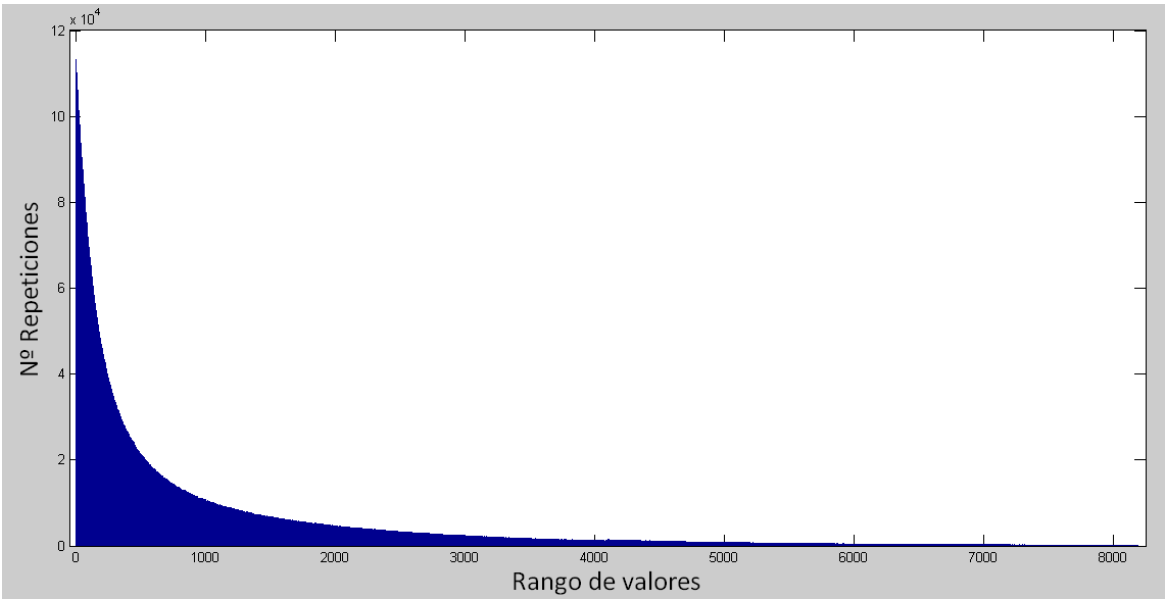


Figura 18: Histograma de la Tabla 13 de Huffman

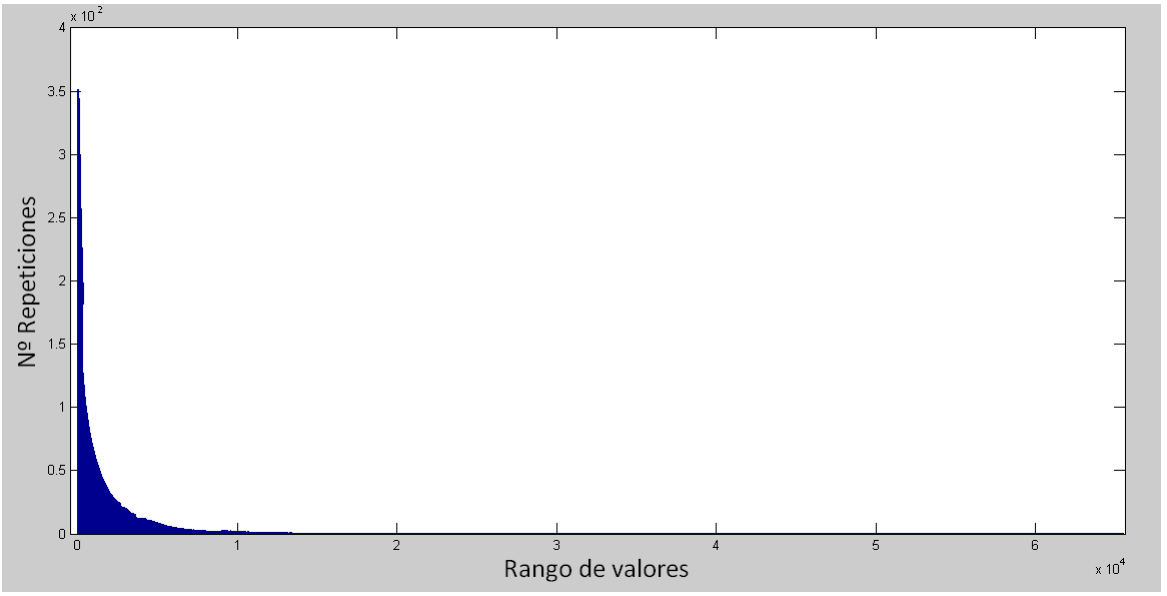


Figura 19: Histograma de la Tabla 16 de Huffman

En la Figura 20 se muestra el número de veces que se emplea cada tabla de Huffman que se va a crear:

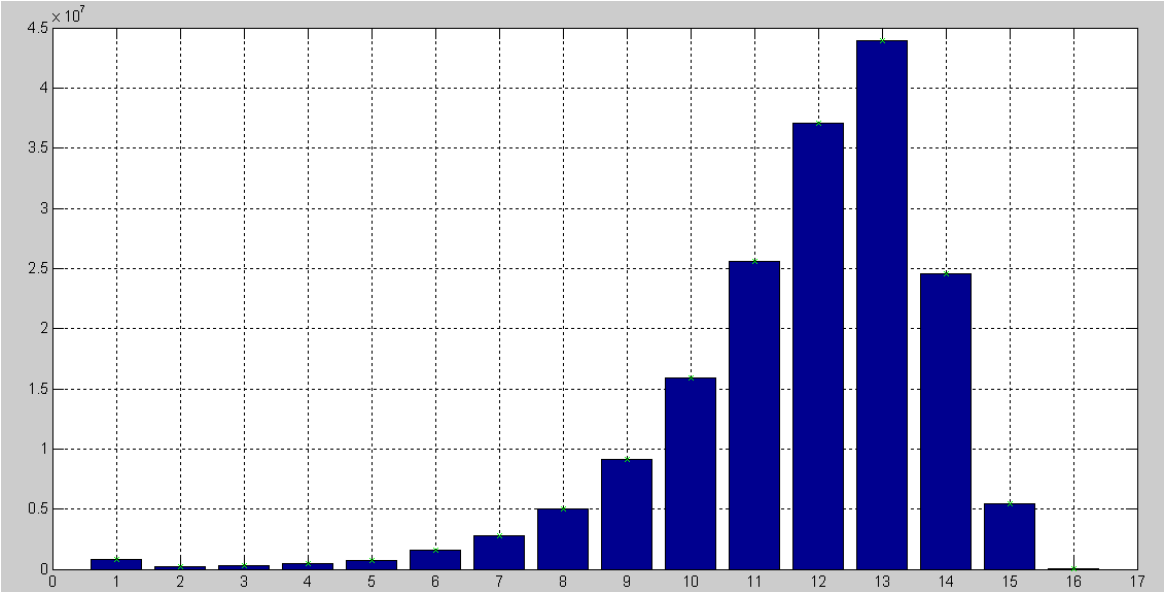


Figura 20: Gráfica de frecuencia de uso de las tablas Huffman

Se puede observar que las tablas que más se emplean son la 12 y la 13, esto guarda relación con los valores que suele emplear la codificación de Golomb-Rice para su parámetro K. También se ve que la tabla 1 es ligeramente superior a las posteriores, lo que quiere decir que posiblemente muchas tramas puedan codificarse con esta tabla que da el menor número de bits.

Para crear las tablas de Huffman se ha empleado un programa de MatLab, donde introduciendo las frecuencias de los símbolos, devuelve el número de bits necesarios para su codificación. Este programa se basa en ir creando ramas según la frecuencia de aparición de los símbolos, así símbolos menos frecuentes tienen asignados códigos de longitud mayores y símbolo con una aparición mayor tienen los códigos menores.

Como ejemplos se van a poner las cuatro primeras tablas creadas, dónde aparece el símbolo a codificar y el número de bits que se necesitan para ello, en este punto hay que aclarar que las tablas de Huffman codifican valores absolutos, el codificador empleará un bit adicional para la codificación.

Tabla 1

Símbolo	***Nº Bits***
0	1
1	1

Tabla 2

Símbolo	***Nº Bits***
0	1
1	2
2	3
3	3

Tabla 3

Símbolo	***Nº Bits***
0	2
1	2
2	2
3	3
4	4
5	5
6	6
7	6

Tabla 4

Símbolo	***Nº Bits***
0	3
1	2
2	3
3	3
4	3
5	4
6	4
7	5
8	5
9	6
10	6
11	7
12	7
13	7
14	8
15	8

Tabla 10: Tablas Huffman 1-4

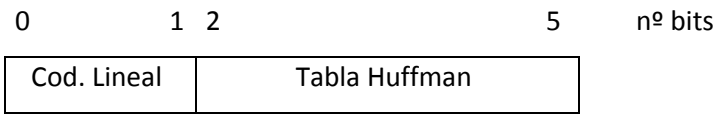
…

Estas 16 tablas serán fijas y conocidas tanto para el codificador como el decodificador.

4.3.1. Codificación Huffman

En este apartado se va a comparar la codificación de AudioPak, Golomb-Rice, con la codificación de la fuente basada en las tablas que se han creado siguiendo los principios de Huffman.

En este caso la cabecera no emplea bits adicionales, respecto a la de AudioPak, ya que los 4 bits que se empleaban para codificar el valor de K, se van a emplear para codificar el número de tabla empleada. En este caso, la cabecera quedaría de la siguiente forma:



Para cada una de las tramas, tras haber realizado la codificación lineal, y haber codificado el codificador lineal empleado, se calcula el valor absoluto de las muestras. Las tablas sólo muestran valores absolutos, y para ver que tabla es la que se debe emplear se tiene que calcular el valor máximo, que es el que va a fijará la tabla que da la codificación óptima. Una vez seleccionada la tabla a usar, se codifica el valor en los 4 bits correspondientes de la cabecera.

Para codificar las muestras, se debe codificar en primer lugar el bit de signo y después se buscan los valores absolutos de las mismas en la tabla seleccionada. El codificador que se ha creado en MatLab devuelve una matriz dónde muestra el valor del codificador lineal, la tabla de Huffman empleada y la longitud de la trama codificada con esos parámetros.

Tras haber realizado la codificación de todas las canciones con nuestras tablas de Huffman, se obtienen los siguientes resultados, comparándolos con los que se tenían de AudioPak, resulta la tabla siguiente:

Canción	AudioPak	Huffman	Ganancia	Porcentaje
011	166333502	168020250	-1686748	-1,0141%
012	108163625	109127438	-963813	-0,8911%
013	149631363	152182663	-2551300	-1,7051%
014	64169821	63981215	188606	0,2939%
015	102934754	103554621	-619867	-0,6022%
016	112924647	113941266	-1016619	-0,9003%
017	186089707	186608778	-519071	-0,2789%
018	146226539	147706689	-1480150	-1,0122%
019	68279730	68525052	-245322	-0,3593%
020	119979313	121261974	-1282661	-1,0691%
021	140533787	142367389	-1833602	-1,3047%
022	174896997	176207654	-1310657	-0,7494%
023	104738209	105729052	-990843	-0,9460%
024	71523444	72343000	-819556	-1,1459%
025	119465932	120741666	-1275734	-1,0679%
026	94797276	96369373	-1572097	-1,6584%
PROMEDIO	**120668040**	**121791755**	**-1123714**	**-0,90%**

Tabla 11: Resultados experimento Golomb-Rice vs Huffman

Como se puede observarse, la codificación de Huffman, en la mayoría de los casos provoca que se tenga que emplear un mayor número de bits para codificar los archivos de audio. Esto es debido, tal como se vio en la Figura 20, el gráfico indicaba que las tablas más empleadas eran las tablas 12 y 13 y estas tablas emplean un número muy elevado de bits para codificar los símbolos. Tal y como se dijo en el estado del arte, Golomb-Rice es muy adecuada para datos con estadística Laplaciana, es decir, señales de audio.

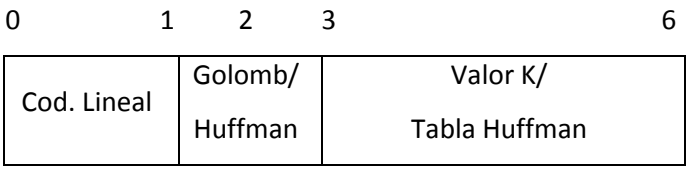
Esta propuesta no conlleva ningún beneficio en el número de bits, pero está claro que las tablas de Huffman son útiles si los símbolos a codificar pudiesen hacerse siempre con las tablas más pequeñas. Esto lleva a pensar que si se combinasen codificación de Huffman y codificación de Golomb, quizá se podrían obtener mayores beneficios. Esto se estudiará en el apartado siguiente.

4.3.2. Codificación mixta: Huffman-Golomb

Así como se ha introducido en el apartado anterior, ahora se intentará combinar las dos codificaciones con las que se ha ido trabajando, la codificación de Huffman, con las tablas que se han creado y la codificación de Golomb-Rice que emplea AudioPak.

En este caso la cabecera de la trama se tendrá que modificar, se ha de añadir un bit adicional, que indique si la trama la se va a codificar mediante tablas de Huffman o por el contrario se va a emplear la codificación de Golomb-Rice.

De tal forma que la cabecera de la trama para este apartado constaría de 7 bits, dos para indicar la codificación lineal empleada, uno para codificar si se emplea codificación Huffman o codificación Golomb-Rice y cuatro bits para indicar o bien el valor de la tabla o bien el valor del parámetro K.



Para la codificación se tiene que decidir el codificador lineal a emplear, el de menor residuo, se codifica en dos bits de la cabecera. Para verificar si se emplea codificación Golomb-Rice o Huffman, se codifica la trama con el valor óptimo de K y para la tabla óptima de Huffman, el que menor número de bits necesite será el método empleado, se codifica, por tanto, el bit que indica si se emplea Huffman o Golomb-Rice y además se codifican los 4 bits que indican el valor de K, en el caso de Golomb-Rice, y el número de tabla en el caso de Huffman.

En la siguiente tabla se comparan los valores que se han obtenido al combinar estos tipos de codificaciones, con el modelo base de AudioPak. Hay que tener en cuenta que la cabecera de las tramas tiene ahora un bit adicional.

Canción	*AudioPak*	*Huffman /Golomb*	*Ganancia*	*Porcentaje*
011	166333502	165997822	335680	0,202%
012	108163625	107965235	198390	0,183%
013	149631363	149586114	45249	0,030%
014	64169821	63604554	565267	0,881%
015	102934754	102534125	400629	0,389%
016	112924647	112801904	122743	0,109%
017	186089707	185265205	824502	0,443%
018	146226539	146109450	117089	0,080%
019	68279730	67832140	447590	0,656%
020	119979313	119886963	92350	0,077%
021	140533787	140503903	29884	0,021%
022	174896997	174639990	257007	0,147%
023	104738209	104635188	103021	0,098%
024	71523444	71391440	132004	0,185%
025	119465932	119404159	61773	0,052%
026	94797276	94711065	86211	0,091%
PROMEDIO	120668040	120429329	238711	0,23%

Tabla 12: Resultado experimento AudioPak vs codificación mixta

Al igual que en tablas anteriores la columna de AudioPak es el número de bits necesarios para la codificación con el modelo usado de AudioPak. Huffman/Golomb, es el número de bits necesarios empleado la extensión explicada. Ganancia, es la diferencia entre el número de bits usado en AudioPak menos los usados con esta extensión. Porcentaje es el número de bits de ganancia, entre los bits del codificador AudioPak por cien.

Se puede observar que cuando se combinan las dos codificaciones se consigue ahorrar bits en la codificación, a pesar de ese bit adicional que se le añade a la cabecera.

Otro dato interesante que se ha estudiado, es el número de tramas que se codifican por los dos métodos, según los datos que se han obtenido en el apartado anterior la codificación Golomb debería ser la que más se emplease.

La siguiente tabla muestra el número de tramas que se codifican con estos dos métodos y el número de tramas totales que existen en la señal de audio. Además, se representa el porcentaje que supone cada una de ellas.

Canción	*Nºtramas total*	*Tramas Huffman*	*Tramas Golomb*	*% Huffman*	*% Golomb*
011	218520	51426	167094	23,534%	76,466%
012	164466	38041	126425	23,130%	76,870%
013	191304	18910	172394	9,885%	90,115%
014	99054	50365	48689	50,846%	49,154%
015	144936	50408	94528	34,779%	65,221%
016	156906	36018	120888	22,955%	77,045%
017	294624	113133	181491	38,399%	61,601%
018	190026	33415	156611	17,584%	82,416%
019	123372	47336	76036	38,369%	61,631%
020	147150	26521	120629	18,023%	81,977%
021	194202	23787	170415	12,249%	87,751%
022	251460	68884	182576	27,394%	72,606%
023	124416	19245	105171	15,468%	84,532%
024	94554	15114	79440	15,985%	84,015%
025	153648	22628	131020	14,727%	85,273%
026	161244	28123	133121	17,441%	82,559%
PROMEDIO	169367	40209	129158	23,80%	76,20%

Tabla 13: Codificación mixta, tramas por Huffman vs tramas por Golomb

En todas las señales de audio, el método más empleado para la codificación es el método de Golomb-Rice, ya que esta codificación es muy adecuada para situaciones en las que la ocurrencia de los valores pequeños en el flujo de entrada es mucho más probable que los valores grandes, como ocurre en el caso de las señales de audio.

Pero se puede observar que en el mejor de los casos dónde se han codificado los símbolos por igual entre tablas de Huffman y Golomb-Rice, por ejemplo en la canción 014, se tiene un porcentaje de mejora de 0.881% respecto a la codificación de AudioPak.

4.4. Extensión total

Tras haber explicado las diferentes extensiones por separado, se han realizado unas últimas pruebas combinando las que han resultado mejoras, desde el punto de vista del número de bits totales. Según los resultados vistos, se empleará la longitud de trama variable combinada con la codificación mixta entre Golomb-Rice y tablas de Huffman.

En este caso de este codificador se decidirá el tamaño de trama empleado, así como el método a emplear Huffman o Golomb-Rice.

El funcionamiento del codificador desarrollado en este punto es muy parecido a lo explicado en la parte del enventanado, salvo que las matrices en lugar de tener sólo los valores de los bits necesarios en la codificación de Golomb-Rice almacenarán también los valores de la codificación por tablas de Huffman.

En primer lugar, se tiene que calcular el número de bits necesarios para la codificación de la señal, para todos los codificadores lineales, empleando Golomb, es decir, para los 16 valores de K, y además para todos los codificadores lineales, empleando las tablas de Huffman.

Al igual que se explicó en la parte de enventanado, se crea una matriz, por cada trama de tamaño 8x16. Esta matriz se podría dividir en dos submatrices de 4x16, la primera submatriz correspondería a la codificación de Golomb-Rice, es decir, AudioPak. Mientras que la segunda almacenaría el número de bits necesario para la codificación por tablas de Huffman, para las cuatro codificaciones lineales y las 16 tablas de Huffman.

En este caso habrá símbolos que no sean codificables por algunas de las tablas de Huffman, en ese caso la casilla se rellenará con un valor alto, para que no afecte a la hora de calcular la longitud de trama.

En la siguiente tabla se muestra una de estas matrices, en concreto la relativa a la canción 012, trama 100. Se muestra traspuesta para su mejor visualización.

	Golomb-Rice				Huffman			
C.Lineal / K/Tabla	1	2	3	4	1	2	3	4
1	1045	352	388	516	100000000	100000000	100000000	100000000
2	668	323	342	407	100000000	100000000	100000000	100000000
3	512	341	356	387	100000000	100000000	100000000	100000000
4	468	391	396	408	100000000	331	100000000	100000000
5	474	455	455	456	100000000	324	343	100000000
6	519	519	519	519	526	340	348	389
7	583	583	583	583	497	362	367	402
8	647	647	647	647	498	377	391	411
9	711	711	711	711	505	417	424	439
10	775	775	775	775	522	460	468	476
11	839	839	839	839	533	521	524	521
12	903	903	903	903	583	575	572	574
13	967	967	967	967	648	649	652	648
14	1031	1031	1031	1031	712	713	716	712
15	1095	1095	1095	1095	712	712	712	712
16	1159	1159	1159	1159	876	859	846	874

Tabla 14: Matriz general, audio 012, trama 100

Como se ha explicado las tramas que no se pueden codificar por alguna tabla de Huffman tienen un valor muy alto para que no influya en los cálculos de la longitud de trama. En este caso se ha elegido el valor de 100000000 para que no interfiera con los cálculos de la longitud total de trama. Cuando se realiza el proceso de fusionar las tramas para ver si disminuye la longitud total, estas casillas siempre darán valores muy altos para no poder ser seleccionadas, puesto que no es posible codificar con esos valores de tabla.

En este caso, el valor mínimo sería para el codificador lineal segundo, con codificación Golomb-Rice de K=2.

Estas matrices se calculan para todas las tramas y se almacenan en un tensor.

Una vez que se tiene el tensor de matrices, se pasa a realizar el proceso que se expuso en el Capítulo 3.1, de bucles para calcular el tamaño de trama óptimo. El proceso es el mismo de ir buscando el menor número de bits en la codificación.

Tras realizar esta operación el codificador almacena para cada trama, la longitud de la misma, el codificador lineal empleado, y el valor de K o la tabla almacenada, además el número de bits necesarios para la codificación de la trama con estos parámetros. En la siguiente tabla se muestra la señal de audio 026, las tramas de la 13 a la 18.

	13	***14***	***15***	***16***	***17***	***18***
Longitud trama	64	64	128	64	64	128
Bits	388	536	1172	560	584	1200
Cod. Lineal	3	3	3	3	2	2
K/Tabla	5	9	22	22	11	11

Tabla 15: Matriz tras el codificador, audio 026, tramas 13-18

Se observan que hay tramas de diferentes longitudes, con diferentes codificadores lineales y el valor de K o de la tabla también va variando. Según como se distribuye la matriz que se ha examinado antes, los valores entre 1-16 corresponden a la K de la codificación Golomb, y los valores entre 17-32 corresponderían a las tablas de Huffman, es decir, la 17 corresponde a la tabla 1, el valor 18 a la tabla 2, etc.

Tras hacer todos estos cálculos se va a comparar la longitud total que tendrían las señales de audio codificadas con todas estas extensiones comparándolas con la codificación AudioPak base de la que se ha partido, los resultados se muestran en la Tabla 16.

Canción	*AudioPak*	*Todas Extensiones*	*Ganancia*	*Porcentaje*
011	166333502	165709491	624011	0,375%
012	108163625	107740984	422641	0,391%
013	149631363	149104754	526609	0,352%
014	64169821	63623655	546166	0,851%
015	102934754	102468788	465966	0,453%
016	112924647	112698080	226567	0,201%
017	186089707	185222095	867612	0,466%
018	146226539	145769228	457311	0,313%
019	68279730	67818441	461289	0,676%
020	119979313	119582555	396758	0,331%
021	140533787	140195093	338694	0,241%
022	174896997	174527045	369952	0,212%
023	104738209	104384117	354092	0,338%
024	71523444	71242099	281345	0,393%
025	119465932	119117242	348690	0,292%
026	94797276	94492160	305116	0,322%
PROMEDIO	**120668040**	**120230989**	**437051**	**0,39%**

Tabla 16: Resultado experimento todas las extensiones vs AudioPak

Como se puede observar se ha conseguido reducir el número de bits empleados en todos los archivos de audio, cuando se han combinado las extensiones propuestas.

Además, se han sacado datos estadísticos de otras características de las señales de audio, como el número de tramas que se codifican por Huffman o por Golomb, y el número de tramas que pasa a tener la señal de audio una vez se hace la longitud de trama variable.

Canción	*Nºtramas total*	*Tramas long. Variable*	*Tramas Golomb*	*Tramas Huffman*	*% Golomb*	*% Huffman*
011	218520	130787	43086	87701	32,94%	67,06%
012	164466	97936	47384	50552	48,38%	51,62%
013	191304	84195	27802	56393	33,02%	66,98%
014	99054	84839	41503	43336	48,92%	51,08%
015	144936	106717	44390	62327	41,60%	58,40%
016	156906	108515	50605	57910	46,63%	53,37%
017	294624	231534	136974	94560	59,16%	40,84%
018	190026	99626	29258	70368	29,37%	70,63%
019	123372	98820	61120	37700	61,85%	38,15%
020	147150	72499	10771	61728	14,86%	85,14%
021	194202	107458	46649	60809	43,41%	56,59%
022	251460	183866	96948	86918	52,73%	47,27%
023	124416	59260	2183	57077	3,68%	96,32%
024	94554	52217	16253	35964	31,13%	68,87%
025	153648	79334	17381	61953	21,91%	78,09%
026	161244	96614	76060	20554	78,73%	21,27%
PROMEDIO	**169367**	**105888**	**46772**	**59115**	**40,52%**	**59,48%**

Tabla 17: Resultado experimento final, características tramas codificadas

La segunda columna indica el número de tramas que quedan tras haber realizado la longitud variable de trama, y las tramas que emplean Golomb o Huffman también están calculadas tras elegir la longitud de trama.

En la siguiente tabla se puede ver la distribución de las longitudes de trama, es decir, cuantas tramas existen para cada señal de audio de longitud L, 2L, 3L o 4L.

Canción	*Tramas*	*% L=64*	*% L=128*	*% L=256*	*% L=512*
011	130787	65,002%	24,777%	7,311%	2,910%
012	97936	63,640%	25,836%	7,894%	2,630%
013	84195	54,813%	25,193%	9,484%	10,510%
014	84839	86,255%	12,705%	0,809%	0,231%
015	106717	72,834%	23,349%	3,565%	0,253%
016	108515	68,988%	25,396%	5,029%	0,587%
017	231534	78,340%	19,219%	2,264%	0,177%
018	99626	55,726%	29,299%	10,847%	4,128%
019	98820	80,239%	17,640%	1,912%	0,209%
020	72499	54,230%	29,234%	10,508%	6,029%
021	107458	60,734%	26,542%	8,721%	4,002%
022	183866	73,081%	22,708%	3,856%	0,355%
023	59260	50,179%	31,468%	12,497%	5,856%
024	52217	60,183%	26,920%	9,030%	3,867%
025	79334	56,299%	28,560%	10,219%	4,922%
026	96614	63,465%	26,712%	7,145%	2,678%
PROMEDIO	**105888**	**65,25%**	**24,72%**	**6,94%**	**3,08%**

Tabla 18: Resultado experimento final, longitud de trama

La mayoría de las tramas emplean una longitud de 64 muestras, esto indica, al igual que el primer experimento para elegir la longitud inicial de trama, que la longitud inicial elegida, L=64 muestras, no es demasiado larga.

También se ha extraído información sobre el tipo de codificación lineal empleada en las tramas.

Canción	*Tramas*	*% Predictor 1*	*% Predictor 2*	*% Predictor 3*	*% Predictor 4*
011	130787	0,297%	30,314%	64,901%	4,488%
012	97936	0,626%	21,948%	74,804%	2,622%
013	84195	1,669%	56,160%	38,133%	4,038%
014	84839	0,165%	14,318%	77,193%	8,324%
015	106717	0,027%	15,253%	61,667%	23,053%
016	108515	0,014%	5,838%	61,428%	32,720%
017	231534	0,051%	5,187%	66,992%	27,770%
018	99626	0,436%	25,083%	62,586%	11,895%
019	98820	0,034%	11,214%	82,541%	6,210%
020	72499	0,247%	34,333%	53,903%	11,517%
021	107458	0,417%	26,555%	63,803%	9,226%
022	183866	0,019%	2,760%	51,810%	45,411%
023	59260	0,329%	15,245%	72,256%	12,170%
024	52217	0,352%	31,149%	59,636%	8,863%
025	79334	0,407%	40,926%	54,409%	4,258%
026	96614	0,230%	30,543%	64,824%	4,403%
PROMEDIO	**105888,56**	**0,33%**	**22,93%**	**63,18%**	**13,56%**

Tabla 19: Resultado experimento final, predictor lineal.

Se observa que los codificadores lineales que más se emplean son los relativos al primer y segundo orden, predictores 2 y 3, respectivamente.

5.CONCLUSIONES

La codificación de audio sin pérdidas es un área difícil en el que conseguir grandes reducciones o rangos de compresión de la señal, puesto que tiene que ser reconstruirla perfectamente. Sólo se pueden explotar las redundancias de la señal de audio para conseguir un mayor ratio de compresión.

El objetivo básico del proyecto consistía en realizar un estudio sobre la codificación de audio sin pérdidas y proponer la estructura de un codificador que mejorase las prestaciones ofrecidas por el que se toma como punto de partida, AudioPak.

De todas las pruebas que se han realizado se puede extraer que:

- La longitud de trama adaptativa, da mejores prestaciones con relación al número de bits empleados y al ratio de comprensión que AudioPak, pero ralentiza mucho el proceso de codificación, ya que se ha de realizar un estudio previo.
- En cuanto a la redundancia temporal, AudioPak emplea unos predictores con un buen compromiso entre simplicidad y resultados.
- La codificación de la fuente, por tablas de Huffman no conlleva ninguna disminución de tamaño final, si se emplea individualmente. Si se combina con la codificación de Golomb-Rice se pueden obtener mejoras considerables en la reducción del tamaño de archivo codificado.
- Si se juntan las extensiones que han dado resultados satisfactorios: longitud de trama variable y codificación mixta Golomb-Rice y Huffman se consigue comprimir el tamaño de todos los archivos respecto a AudioPak, pero el coste computacional en la fase de codificación es muy elevado.

Destacar que con el resultado final obtenido, se ha mejorado la longitud final de todas las tramas, consiguiendo ahorrar como promedio 437051 bits/canción, aproximadamente un 0.4% de la longitud inicial. Además, gracias a la longitud variable de la trama se han conseguido reducir el número total de tramas, pasando de un promedio de 169367 tramas/canción a 105888 tramas/canción.

Teniendo en cuenta que habría que reducir el coste computacional en el codificador, volver a destacar que este proceso de codificación sólo se realiza una vez y que a la hora de decodificar las propuestas de extensión no suponen un coste adicional, puesto que toda la información va en la cabecera de la trama y ésta sólo se ha aumentado 3 bits en relación con la cabecera que se ha tomado como base en AudioPak.

Además, volver a destacar que si en algún caso las extensiones diesen resultados negativos siempre se podría volver al codificador básico de AudioPak. Para esto se añade un bit adicional en la cabecera de archivo, para indicar si se va a emplear la codificación básica, AudioPak, o la codificación con extensiones.

Aunque los avances en el área de la codificación sin pérdidas son lentos y con resultados peores en comparación con la codificación con pérdidas, se puede llegar a mejorar los algoritmos o códecs que existen.

6.LÍNEAS DE TRABAJO FUTURAS

Durante la realización del proyecto han surgido diferentes situaciones o problemas que se pueden mejorar en un futuro. El códec propuesto tiene posibilidades de crecimiento, en este capítulo se citan algunas de las extensiones que se le podrían realizar.

Los problemas básicos que se han encontrado durante el proceso, son principalmente problemas de rendimiento y optimización:

• Carga de la CPU: El codificador se ha implementado en código MatLab, es un algoritmo complejo que consume muchos recursos del microprocesador. Una posible extensión en este campo sería realizar la implementación real del codificador, con las pautas marcadas, en otro lenguaje que consuma menos recursos, por ejemplo C++, o incluso un *hardware* específico, como las DSP.

• Longitud de trama a posteriori: sería recomendable que se investigase la forma de elegir la longitud de trama, sin necesidad de realizar todo el estudio previo de la codificación de la señal de audio. Los algoritmos propuestos exigen la caracterización completa de la señal, para poder simplificar el proceso, es conveniente estudiar estadísticos y sacar relaciones entre ellos y la longitud de trama.

Además, se debería hacer un estudio comparativo entre los diferentes codificadores sin pérdidas que existen en el mercado para ver la relación que tienen con las extensiones propuestas en este proyecto.

Otra extensión importarte es incluir soporte en el codificador para el audio multicanal (en particular 5.1, 6.1 y 7.1), y estéreo, ya que en este proyecto sólo se ha trabajado con archivos en un solo canal. Tratar de explotar la alta correlación que puede existir entre canales para poder lograr una mayor compresión. Esta es la línea de trabajo futuro más importante para poder realizar un codificador de audio sin pérdidas completo, que explote todas las redundancias de la señal.

7.PRESUPUESTO

Durante la

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:
Sonia García Martín
2.- Departamento:
Teoría de la señal
3.- Descripción del Proyecto:

- Titulo	**Codificación audio sin pérdidas**
- Duración (meses)	**12**
Tasa de costes Indirectos:	20%

4.- Presupuesto total del Proyecto (valores en Euros):
22.183 Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (persona mes) [a]	Coste persona mes	Coste (Euro)
Jose M. Leiva		Ingeniero Senior	0,5	4.289,54	2.144,77
Sonia García Martín		Ingeniero	6	2.694,39	16.166,34
		Personas mes	**6,5**	**Total**	18.311,11

[a] 1 Personas mes = 131,25 horas. Máximo anual de dedicación de 12 personas mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 personas mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Ordenador Portátil	900,00	100	12	60	180,00
Disco de Backup	150,00	100	12	60	30,00
				Total	210,00

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	18.311
Amortización	210
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	3.662
Total	22.183

8. REFERENCIAS

[1] Ken C. Pohlmann. Principles of Digital Audio. Mc GrawHill, 4th edition, 2000.

[2] M. Hans and R.W. Schafer. Lossless compression of digital audio. IEEE Signal Processing Magazine, 18(4), July 2001.

[3] J.Watkinson, "The Art of Digital Audio", Ed. Focal Press, 3ª edition.

[4] http://es.wikipedia.org/wiki/Códec_de_audio

[5] http://es.wikipedia.org/wiki/Advanced_Audio_Coding

[6] http://www.mp3developments.com/article4.php

[7] http://www.musepack.net/

[8] http://es.wikipedia.org/wiki/Atrac

[9] P. Noll, "MPEG Digital Audio Coding", IEEE Signal Processing Magazine, Vol. 14, No. 5, pp.59-81, Sep. 1997

[10]María Carmen España Boquera. Servicios avanzados de telecomunicación. Ediciones Díaz de Santos, 2003.

[11]http://patentados.com/patente/conversion-de-canal-de-audio

[12]Craven, PG and Gerzon, MA. 'Optimal Noise Shaping and Dither of Digital Signals.' Presented at AES 87th Convention, New York (Preprint#2822) (October 1989)]

[13]Shannon,C.E. 'A Mathematical Theory of Communication',Bell System Technical Journal vol 27 pp.379-423,623-656(July, October 1948).

[14]J.R. Stuart, P.G. Craven, and M.J. Law. Lossless compression for dvd-audio. AES 9th Regional Convention Tokyo.

[15]A.V. Oppenheim and R.W. Schafer. Tratamiento de señales en tiempo discreto. Prentice Hall, 2nd edition, 2000.

[16]L.R. Rabiner and R.W. Schafer. Digital processing of speech signals. Prentice Hall, 1978.

[17]J.L. Hennessy and D.A. Patterson. Computer architecture: a quantitative approach. San Francisco, Morgan Kauffmann, 2nd edition, 1996.

[18]M. Purat, T. Liebchen, and P. Noll, “Lossless transform coding of audio signals,” in Proc. 102nd AES Conv., Munich, Germany, 1997, preprint 4414.

[19]A. Wegener, “MUSICompress: Lossless, low-MIPS audio compression in software and hardware,” in Proc. Int. Conf. Signal Processing Applications and Technology, 1997. Available: http://members.aol.com/sndspace.

[20]C. Montgomery. (1997, August). Personal Communications on OggSquish. Available: http://www.xiph.com

[21]T. Robinson, “SHORTEN: Simple lossless and near-lossless waveform compression,” Cambridge Univ. Eng. Dept., Cambridge, UK, Tech. Rep.156, 1994.

[22]R. Sprague, Private communication on Sonarc, Aug. 1997.

[23]D. Lee, Personal communications on waveform archiver, Aug. 1997.