COMPRESIÓN Y DESCOMPRESIÓN DE VOZ MEDIANTE TECNICAS DE PROCESAMIENTO DIGITAL DE IMÁGENES UTILIZANDO WAVELETS

MAURICIO CASTILLO ROZO
GERARDO MENESES PÉREZ

Tesis de grado presentada como requisito para obtener el título de Ingeniero en Electrónica y Telecomunicaciones

Director
HAROLD A. ROMO ROMERO
INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE TELECOMUNICACIONES
GRUPO I+D EN NUEVAS TECNOLOGÍAS EN TELECOMUNICACIONES
POPAYÁN
2008

COMPRESIÓN Y DESCOMPRESIÓN DE VOZ MEDIANTE TECNICAS DE PROCESAMIENTO DIGITAL DE IMÁGENES UTILIZANDO WAVELETS

MAURICIO CASTILLO ROZO
GERARDO MENESES PÉREZ

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE TELECOMUNICACIONES
GRUPO I+D EN NUEVAS TECNOLOGÍAS EN TELECOMUNICACIONES
POPAYÁN
2008

CONTENIDO

Pág.

LISTADO DE FIGURAS

Pág.

LISTADO DE TABLAS

Pág.

LISTADO DE TÉRMINOS

Los diferentes términos (siglas y acrónimos) que se tratarán en este trabajo se muestran a continuación:

- *ADPCM: Adaptive Differential Pulse Code Modulation*
- *ASCII: American Standard Code for Information Interchang*
- *BMP: Bit Mapped Picture*
- *CELP: Code excited linear prediction*
- *CRF: Canon Research Center France*
- *DWT: Discrete wavelet transform*
- *EBCOT: Embedded Block Coding with Optimized Truncation*
- *EPFL: Ecole Polytechnique Fédérale de Lausann*
- *EZW: Embedded Zerotree Wavelet*
- *GIF: Graphics Interchange Format*
- *GSM: Groupe Spécial Mobile*
- *ICT: Irreversible Color Transform*
- *JPEG: Joint Pictures Expert Group*
- *JPEG2000: Joint Pictures Expert Group 2000*
- *LPC: Linear Predictive Coding*
- *MOS: Mean opinion score*
- *MPEG: Moving Picture Experts Group*
- *PAM: Portable Any Ma*
- *PBM: Portable Bit Map*
- *PGM: Portable Grey Map*
- *PPM: Portable Pixel Map*
- *PR-UMDFB: Perfect Reconstruction-Uniformly Maximally Decimated Filter Bank*
- *RCT: Reversible Color Transform*
- *RDSI: Red Digital de Servicios Integrados*

- *RIFF: Resource Interchange File Format*
- *ROI: Region of interest*
- *SPIHT: Set Partitioning in Hierarchical Trees*
- *UBC: University of British Columbia*
- *VoIP: Voice over IP*
- *WAV: Waveform Audio Format*

1. INTRODUCCIÓN

Las bases de la teoría de la información provienen de las telecomunicaciones más tempranas, desde los momentos en que las redes telegráficas apenas se estaban estableciendo. Esta teoría matemática se creó en respuesta a la necesidad de un sistema de comunicación más eficiente. La Teoría de la información sigue manteniéndose como un tema vigente, donde cada vez cobra más fuerza la necesidad de reducir toda información redundante en una fuente de datos.

La teoría de la información se centra en los problemas referentes al envío de un mensaje entre un transmisor y un receptor, pasando el mensaje por un medio. El problema fundamental de la comunicación es tener la capacidad de asegurar que el receptor logre recuperar toda la información transmitida a través del medio. Por lo general, los mensajes enviados tienen un significado, un sentido, por lo que tienen una lógica interna relacionada con un sistema físico o conceptual.

Para atender estos problemas se han generado modelos matemáticos que atienden los problemas pertinentes a la comunicación, que se definen en encontrar el mejor método para presentar un mensaje, con la menor redundancia posible, en establecer el mejor método para distinguir la información entre el ruido inherente a un medio de transmisión y de cómo establecer el límite de información que puede enviarse a través de un medio específico [1].

En el proceso de comunicación es posible distinguir por lo menos tres niveles de análisis diferentes: el técnico, el semántico y el pragmático [2]. En el nivel técnico se analizan aquellos problemas que surgen en torno a la fidelidad con que la información puede ser transmitida desde el emisor hasta el receptor, en el nivel semántico se estudia todo aquello que se refiera al significado del mensaje y su interpretación, y en el nivel pragmático se analizan los efectos conductuales de la comunicación, la influencia o efectividad del

mensaje en tanto da lugar a una conducta. Es importante destacar que la teoría de la información se desarrolla como una respuesta a los problemas técnicos del proceso de comunicación, aun cuando sus principios puedan aplicarse en otros contextos [3].

En el área de la ingeniería podemos pasar por alto los aspectos semánticos y pragmáticos, y enfocarnos en torno al tratamiento de datos en la fuente para eliminar cualquier información innecesaria, y a las técnicas de tratamiento de señales en el receptor para poder recuperar la información enviada a través de un canal ruidoso. Esta rama ha generado un amplio nicho de investigación enfocado en el procesamiento de señales, entre los cuales uno de los mayores aportes ha sido la aplicación de la transformada de Fourier.

La transformada de Fourier convierte una señal en una serie de componentes periódicos en el dominio de la frecuencia, una combinación lineal de senos y cosenos que al sumarse dan la representación de una señal en cualquier instante de tiempo. Esta transformada ha sido el fundamento del procesamiento de señales, ampliamente usada en los campos de las telecomunicaciones por su capacidad para sintetizar señales periódicas. Sin embargo, ciertas señales cuya amplitud varía en forma rápida y abrupta en el tiempo, o señales cuyo contenido de frecuencia es variable y no periódico, las cuales son más conocidas como señales no estacionarias, no son analizadas debidamente mediante la transformada de Fourier debido a ciertas limitaciones de este análisis en los dominios tiempo - frecuencia. Es en estos casos es donde entra en juego la teoría de las *wavelets* como herramienta matemática de análisis [4].

La aparición de la teoría de las *wavelet* como herramienta matemática es relativamente reciente, aunque las ideas esenciales en las que se basa ya eran objeto de análisis mucho tiempo antes de que se plasmaran de forma analítica. Se trata de una transformación lineal al igual que la transformada de Fourier, sin embargo a diferencia de ésta, proporciona la localización en el dominio del tiempo de las diferentes componentes en frecuencia presentes en una señal dada. La transformada de Fourier ventaneada consigue parcialmente la identificación tiempo - frecuencia, cuyo principio consiste en multiplicar la señal a analizar por una función de restricción llamada ventana para calcular su transformada de Fourier; de modo que trasladando dicha función a lo largo de la señal, se obtiene información local de la misma en el ancho de esta ventana. El tamaño fijo de la ventana

supone una limitación ya que puede resultar demasiado grande para analizar frecuencias altas y demasiado pequeñas para analizar frecuencias bajas [4].

Diversas revistas de carácter investigativo alrededor del mundo dan constancia de sorprendentes resultados producto del desarrollo y aplicación de la *teoría de wavelets* en los últimos años. Es así como actualmente se puede encontrar una gran variedad de estudios referentes a la aplicación de esta teoría para afrontar problemas relacionados con el tratamiento de señales. Tal es el caso del proceso de reducción de ruido en señales de audio, detección de irregularidades locales en ciertos tipos de señales (electrocardiogramas, vibraciones de motores...), reconocimiento de patrones, y específicamente en el ámbito de las imágenes digitales, se puede destacar su aplicación en realce de bordes, reducción de ruido, filtrado de imágenes, detección de irregularidades locales, y la compresión de imágenes fijas para su almacenamiento en formato digital, siendo ésta última quizá la de mayor renombre. Contrariamente a la creencia popular, la transformada *wavelet* no comprime la información de por sí, lo que hace es analizar la señal y representarla en una serie de datos para que puedan ser comprimidos con más facilidad. Es más un medio que un fin, donde su tarea está definida en permitir una síntesis de datos apropiada [5].

Puntualmente, el *Joint Pictures Expert Group* (JPEG) ha desarrollado un estándar de compresión de imágenes basado en la transformada *wavelet* conocido como JPEG2000, el cual aventaja en rendimiento y capacidad de compresión a su predecesor, el afamado formato JPEG. El codificador del estándar JPEG2000 se basa en una técnica de compresión por la transformada *wavelet* en sub-banda. Maneja una compresión con o sin pérdidas basada en una transformada *wavelet* traspuesta a una o varias matrices de referencia, fundamentándose en los esquemas de códigos de bloque embebidos de truncado optimizado (*Embedded Block Coding with Optimized Truncation* - EBCOT) [5-6].

Un estudio hecho en la Universidad Austral de Chile propuso un modelo de compresión de tramas de voz usando el estándar JPEG, el cual mostró resultados muy provechosos por las similitudes en la estructura de datos entre una imagen y una trama de voz [7]. Considerando los alcances de la tecnología actual y las posibilidades que brindan las *wavelets* para la síntesis de datos, se propone un modelo de codificación de tramas de voz usando el estándar JPEG2000.

Considerando los conceptos previamente establecidos se propone un sistema de compresión y descompresión de tramas de voz utilizando el estándar JPEG2000, enfocados en generar medios de transmisión aptos para una conversación de voz sobre IP (VoIP). El proyecto entonces de divide en dos aspectos independientes del problema a solucionar, la implementación del sistema de interfaz entre una trama de voz y una imagen transformada única, y el diseño de una aplicación que cumpla los requerimientos necesarios para establecer un envío de tramas en un protocolo de VoIP.

En la primera fase, donde se plantean las interfaces de transformación de voz a imagen y viceversa se deben lidiar con los problemas de la asignación apropiada de los campos de datos, la extracción de la información contenida en los archivos de voz, la elección de un medio de compresión, y la transformación de un campo de información vectorial a uno matricial. Después de hacer las definiciones apropiadas del sistema de compresión y descompresión se deben hacer comparaciones de rendimiento con otros métodos de compresión en cuanto a radio de compresión y energía preservada respecto a la señal original.

En la segunda fase se trabajará sobre los aspectos técnicos de la transmisión de paquetes de voz en una sesión de VoIP, definiendo los tiempos de muestreo de la voz, el tamaño promedio de los archivos, el tiempo de procesamiento requerido en la comunicación. Con estas especificaciones se tienen los conceptos adecuados para luego comparar esta arquitectura con otros métodos de transmisión de voz.

Para dar una idea general de lo que se va a tratar en cada uno de los capítulos siguientes, se tiene: En el capítulo 2 en donde se trata la **Definición del problema** en el que se analizan los formatos utilizados en el proyecto, y se enumeran las características de los archivos de voz e imagen. Se hace referencia a las técnicas de compresión y descompresión de señales digitales. Para el capítulo 3 **Definición del Sistema Compresión y Descompresión** se define el Sistema de Compresión y Descompresión con el cual se va a desarrollar este proyecto, incluyendo y analizando cada una de los siguientes bloques: *capturar la voz, pasarla a un formato de imagen única, comprimirla, encapsularla en una trama de VoIP, transmitirla.* De igual manera se analiza el proceso inverso. En el capítulo 4 **Aportes Realizado por los Estudiantes** aquí se resaltan los diferentes aportes desarrollados por los

estudiantes, y se analizan los subproductos de cada fase del sistema. El capitulo 5 **Conclusiones y Recomendaciones** resumen los resultados obtenidos y se enumeran posibles líneas de investigación del proyecto para explorar en trabajos futuros.

2. DEFINICIÓN DEL PROBLEMA

Durante mucho tiempo y ahora más que nunca las telecomunicaciones han tomado una gran importancia en el desarrollo creciente que está haciendo la humanidad, no solo porque con ellas se nos facilita el comunicarnos con los demás sino que cada día que pasa se quiere integrar más las tecnologías a las necesidades que el hombre tiene. Uno de los campos más importantes de las telecomunicaciones está relacionado con la teoría de la información, y más específicamente con el tratado de señales.

Este capítulo se encarga de mencionar los sistemas de almacenamiento de voz e imagen tratados en este trabajo y en explicar en detalle la compresión de imágenes, definiendo el punto de partida para el desarrollo de este proyecto y con el que se desarrollará la respectiva simulación del sistema. Estas explicaciones se dan para fundamentar los objetivos con los que se va a crear un algoritmo de codificación de tramas de voz usando métodos de procesamiento de imágenes.

2.1. SISTEMA GENERAL.

La red de Internet es la más grande que existe en el mundo, la transmisión de voz por este medio está actualmente en estudio. La Voz sobre Ip (VoIP) [8] es una aplicación muy útil en el entorno de las telecomunicaciones y bajo la cual se están desarrollando estándares para brindar la fidelidad de la voz. Es aquí donde este proyecto se enlaza para hacer parte de esta importante investigación. El aporte se enfocará en buscar otra forma de comprimir voz mediante la utilización de *wavelets*, cuya teoría es actualmente una de las más estudiadas y modernas con que se busca dar cabida a la creación de nuevas metodologías para desarrollar compresiones de voz, con mejores características que los ya existentes tales

como ADPCM, GSM, LPC, CELP 4.5K, CELP 3.0K, LPC-10, CELP 2.3K, OpenLPC1.8K, OpenLPC 1.4K, etc.

Los diferentes códecs que actualmente existen para la compresión de voz no han sufrido modificaciones importantes, basándose en trabajos anteriores que muestran la posibilidad de utilizar los codificadores de imágenes para la respectiva compresión de voz en [7], la cual hace uso de la teoría de Fourier y algunos compresores de imágenes tales como JPEG. Para este trabajo se hace uso del estándar JPEG2000, una poderosa herramienta que se utiliza para la compresión de imágenes basándose en la transformada *wavelet*; que mediante la herramienta JJ2000 se puede manipular el codificador y decodificador.

La definición general del sistema a tratar se resume en el diagrama de bloques, mostrado en la *figura 2.1*:

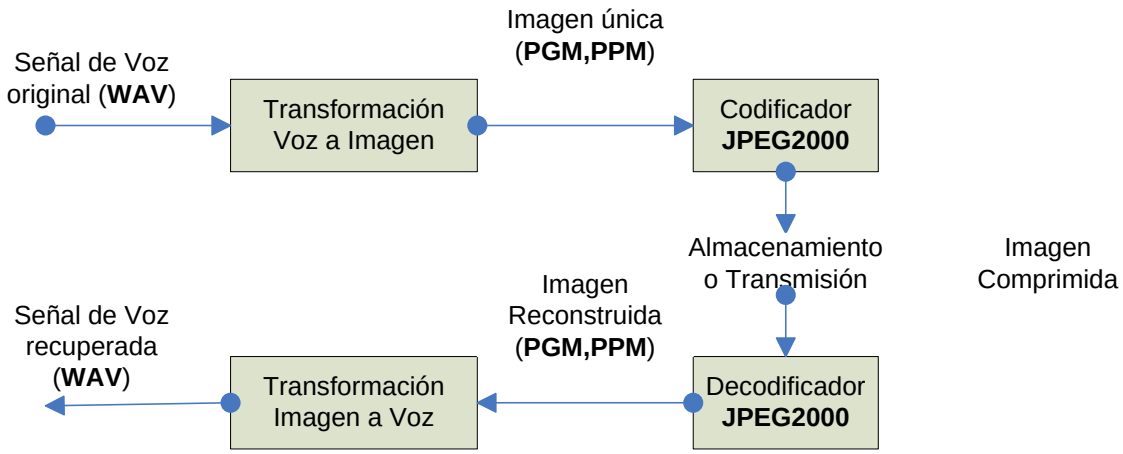


Figura 2.1. Formato del Sistema General Compresión /Descompresión.

La función de cada uno de los bloques se define de acuerdo con el proceso que realizan, mencionando así los siguientes bloques:

Transformación voz a imagen: Este bloque sirve de interfaz, el cual convierte en formato de imagen los datos del archivo de voz para que el compresor de imágenes la pueda procesar. Este bloque debe analizar el archivo de voz para extraer sus características, y reorganizar las muestras como una imagen única equivalente.

Codificador JPEG2000: Este bloque procesa la imagen obtenida para entregar un archivo compreso usando las especificaciones del estándar JPEG2000.

Decodificador JPEG2000: Contrario al bloque anterior, este bloque se encarga de revertir los efectos de la compresión, recuperando una imagen única a partir de los datos compresos.

Transformación de imagen a voz: Este último bloque retoma los datos del archivo de voz, y reconvierte la imagen recuperada en el archivo de voz inicial.

La necesidad de almacenar los datos de voz en un formato base de tal manera que no los comprima sino que simplemente los capture para luego manejarlos fácilmente en el proceso de conversión de vector a matriz, además de utilizar un estándar genérico y que los terminales a utilizar (Computadores) su sistema operativo lo tenga por defecto, precisamente todas estas características las posee el formato WAV.

2.1.1. Formato de audio WAV PCM

A la entrada del sistema desarrollado (*figura 2.1*) se tiene la señal de audio que se va a comprimir, la cual corresponde a archivo en formato *WAV*. Se elige el formato WAV como formato de prueba ya que las tramas de voz se presentan sin compresión alguna, lo que permite medir directamente la eficiencia del programa respecto a datos redundantes.

El formato **WAV** ("WAVEform Audio Format") es un estándar de almacenamiento digital de datos de audio definido por Microsoft e IBM. El formato WAV no es, en realidad, un formato de codificación sino un formato contenedor de audio. Los archivos WAV están basados en el estándar RIFF (“Resource Interchange File Format”), que define una estructura para almacenar datos multimedia.

Estructura

A continuación se explica la estructura del encabezado del archivo WAV para futura referencia. El formato WAV define una estructura de archivo para la recopilación de datos de audio en forma de fragmentos. Es en esta parte donde se entra a realizar el análisis respectivo para diseñar una forma de convertir estos datos en formatos de imagen, los cuales deben ser únicos para cada rango de voz grabada. En la siguiente tabla se muestra la estructura global del archivo WAV.

Tabla 2.1. Estructura global del archivo WAV.

Dirección (byte)	Nombre	Tamaño (bytes)	Descripción
00h	*rID*	4h	Palabra “RIFF”
04h	*rLen*	4h	Tamaño del fragmento
08h	*rData*	rLen	Datos del fragmento

El segmento *rData* contiene los datos que se desean manipular para realizar la respectiva compresión. La codificación y el formato del *rData* se muestra en la *tabla 2.2* y *tabla 2.3* respectivamente.

Tabla 2.2. Codificación del Fragmento rData.

Dirección (byte)	Nombre	Tamaño (bytes)	Descripción
00h	*wID*	4	Término "WAVE"
04h	Formato del fragmento	18	Formato utilizado
1Ch	Datos WAVE del fragmento	variable	Datos

Tabla 2.3. Formato del fragmento rData.

Cambio (byte)	Nombre	Tamaño (bytes)	Descripción
00h	*fId*	4	Término "fmt " (el espacio es necesario)
04h	*fLen*	4	
08h	*wFormatTag*	2	Formato (generalmente 1, para la modulación de código de pulso de Microsoft)
0Ah	*nChannels*	2	Número de canales (1=mono, 2=estéreo)

0Ch	*nSamplesPerSec*	4	Índice de muestreo (en Hz)
10h	*nAvgBytesPerSec*	4	nChannels * nSamplesPerSec * (nBitsPerSample/8) Para estimar el tamaño requerido por el búfer
14h	*nBlockAlign*	2	nChannels * (nBitsPerSample / 8) Para la alineación del búfer
16h	*FormatSpecific*	2	Medida del muestreo, en bits (8 o 16)

Tabla 2.4. Datos WAVE del fragmento

Dirección (byte)	Nombre	Tamaño (bytes)	Descripción
00h	*dId*	4	Término "datos"
04h	*dLen*	4	Longitud del campo dData (en bytes)
08h	*dData*	dLen	Datos de muestra del sonido

El campo *dData* se compone de los siguientes bits:

- En 8 bits mono: Cada byte representa una muestra
- En 8 bits estéreo: 2 bytes para cada muestra (canal izquierdo, canal derecho)
- En 16 bits mono cada palabra (byte bajo, byte alto) representa una muestra
- En 16 bits estéreo: 2 palabras para cada muestra (baja izquierda, alta izquierda, baja derecha, alta derecha)

2.1.2. Formato PPM y PGM

Para poder procesar la voz como una imagen se debe conocer el tipo de imágenes que maneja el codificador. Esto se hace para definir el proceso de interfaz, y conocer el formato que va a tomar la imagen final.

El decodificador del JJ2000 recibe los archivos de imagen no compresas, y entrega un archivo compreso con extensión *.j2k* . Para recuperar los datos en el archivo se utiliza el decodificador, el cual realiza el proceso inverso.

Las imágenes juegan un papel importante en el desarrollo de este proyecto ya que se está utilizando el estándar JPEG2000. Los formatos de archivo de imagen soportados por JJ2000 son PBM ("Portable Bit Map") para imágenes en blanco y negro, PGM ("Portable Grey Map") para imágenes en la escala de grises y PPM ("Portable Pixel Map") para imágenes en color.

Estos tres formatos almacenan información por cada punto de la imagen (llamado píxel). Las imágenes representadas mediante este tipo de formatos tienen la ventaja de que son muy sencillas de manipular, pero tienen el inconveniente de ocupar mucho espacio de memoria. Este inconveniente es la razón de que estos formatos no se utilicen tanto como los formatos JPG o GIF, que ocupan un espacio de memoria muy inferior.

Hay dos versiones de formato PPM (también existen estas dos versiones para PBM y PGM): la versión en ASCII y la versión en binario (*raw*). En modo ASCII los ficheros pueden editarse con un editor de texto cualquiera (En el modo binario esto no es posible, pero los ficheros ocupan un espacio cuatro veces menor).

Un archivo de imágenes en formato PGM almacena las imágenes en escala de grises, es ampliamente utilizado por investigadores en el tema del procesamiento de imágenes, por su simplicidad y eficiencia. A continuación se describe su formato:

- **Primera Línea:** Cadena de caracteres que identifica el tipo de formato. Las cadenas validas son:

    a) **P2**: Los valores de los píxeles vienen en formato ASCII (es decir como cifras numéricas enteras entre 0 y 255).

    b) **P5**: Los valores de los píxeles vienen en formato Binario (es decir la información de cada píxel viene expresada en un byte).

- **Segunda Línea**: aparecen caracteres que dependiendo de ellos se puede definir su función. Si traen como primer carácter **"#"**, son consideradas comentarios.

- **Línea Siguiente al último comentario**: En este campo hay dos números enteros separados por un espacio en blanco y que corresponden al ancho y al alto de la imagen en píxeles.
- **Línea Siguiente**: en este campo viene un número entero que indica la máxima cantidad de niveles de grises que se maneja en la imagen. Generalmente es 255 que indica el máximo espectro de niveles de gris.
- **Campo de datos:** contiene la información de cada uno de los píxeles de la imagen, estos valores pueden estar representados en decimal o ASCII.

Por ejemplo si los valores que aparecen son: **64 40 230,** esto significa que el primer píxel tiene un valor de gris igual a 64, el segundo píxel tiene un valor de gris igual a 40, el tercer píxel tiene un valor de gris igual a 230.

Si la cadena de la primera línea es **P5**, viene una serie de bytes (uno tras otro sin ningún tipo de separador), la cantidad de byte es igual a Ancho x Alto píxeles.

Si los bytes que aparecen son: **"A" "<" "µ",** en este caso aparece el caracter **'A'**, que corresponde al valor ASCII 65 que será el valor del nivel de gris del primer píxel, el caracter **"<"** le corresponde el valor ASCII 40 que será el nivel de gris del segundo píxel, el tercer byte es el carácter **"µ"** corresponde al valor ASCII 230 que será el valor del nivel de gris del tercer píxel.

Observe que en este caso solo se necesita un byte para almacenar los valores de los píxeles, valores que se encuentran entre 0 que corresponde al negro y 255 que corresponde al blanco.

Los dos ejemplos entregan los mismos valores de gris para los píxeles, pero en el caso **P5** se hace un mejor uso del espacio utilizado para almacenar la imagen, con los tres píxeles del ejemplo en el formato **P2** se utilizan 9 bytes, mientras que el formato **P5** sólo se necesitan 3 bytes para almacenar la misma información.

2.1.3. Diagrama de bloques del estándar JPEG2000

Para explicar más coherentemente el sistema de compresión se muestra la estructura del codificador y decodificador de imágenes según el estándar del JPEG2000 [9]. El codificador pasa por varios procesos en cascada, cada uno tomando el subproducto entregado por el bloque anterior.

Los bloques del codificador y decodificador son los que se muestran a continuación: 1) pre-procesamiento/post-procesamiento, 2) transformada intercomponente, 3) transformada intracomponente, 4) cuantización/decuantización, 5) codificación de primer orden, 6) decodificación de segundo orden, y 7) control de la tasa de compresión. Cada bloque del decodificador ha sido diseñado para invertir el proceso generado por el codificador, por lo que al mostrar el proceso de codificación se puede deducir el proceso inverso. Se supone que en el proceso ya se han efectuado los arreglos de formato necesarios, lo cual no es replicado en el programa por razones de practicidad. El diagrama del codificador y del decodificador se muestra en la *figura 2.2.*

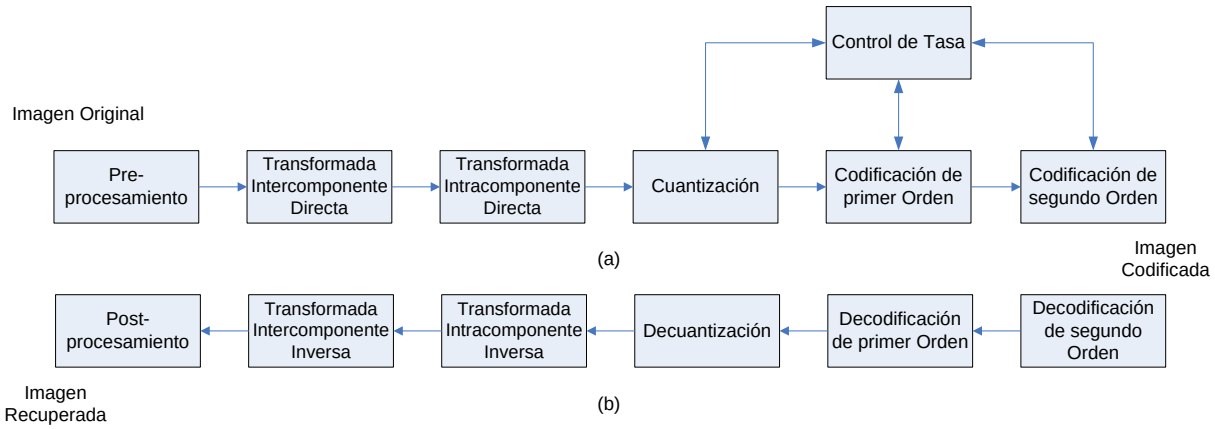


Figura 2.2. a) Estructura del codificador. b) Estructura del decodificador.

Se presenta información más detallada de cada uno de los bloques funcionales en el ANEXO B de esta monografía, donde cada uno de los procesos es explicado. Los procedimientos desarrollados para la compresión de la voz se describirán en el capítulo 3 donde se detalla las modificaciones al JJ2000 importantes para la creación del algoritmo deseado.

3. DEFINICION DEL SISTEMA DE COMPRESION Y DESCOMPRESION

En este capítulo se explica el proceso de desarrollo del sistema de compresión y descompresión de voz utilizando el estándar JPEG2000, dividido en los módulos de *Preprocesamiento, Transformada Wavelets, Compresión, Control de Tasa y Organización de Datos*. Este proceso se dividirá en las fases de análisis, donde se asentarán las decisiones técnicas, la fase de diseño, donde se mostrarán los prototipos obtenidos, y la fase de implementación, donde se explican cada una de las decisiones hasta llegar al sistema final.

3.1. Fase de Análisis

El grupo JPEG, al realizar las recomendaciones del estándar JPEG2000, implementó dos sistemas software de referencia, explicado en la parte 5 del estándar [23], como modelos de verificación e implementación del estándar JPEG2000 y para corregir retroactivamente los defectos en la recomendación. Estos sistemas se pueden utilizar como plataformas de desarrollo de la herramienta, ya que son la implementación explícita del estándar JPEG2000. Ambos sistemas presentan versiones funcionales del estándar y serán presentados a continuación.

3.1.1. Introducción a JasPer

El proyecto JasPer es una iniciativa de código libre para proveer una implementación referencial abierta del codificador especificado en la primera parte del estándar JPEG2000 [9]. Este proyecto ha sido dirigido por un esfuerzo conjunto entre Image Power Inc. y la UBC ("University of British Columbia"). En el momento, el mantenimiento y desarrollo del sistema JasPer está siendo coordinada por su autor principal, Michael Adams [10], el cual está afiliado con el grupo de Procesamiento Digital de Imágenes del Departamento de Ingeniería Eléctrica y Computacional en la Universidad de Victoria.

En términos simples, JasPer es un kit de herramientas de software para el manejo de datos de imágenes. Este software entrega al desarrollador diversos medios de representación de imágenes y facilita la manipulación de datos de imágenes, como la importación/exportación de datos en varios formatos (JPEG2000, JPEG, BMP, Sun Rasterfile, etc). La funcionalidad de la importación detecta automáticamente el formato de imagen, eliminando la necesidad de identificar explícitamente el formado de la imagen codificada. Además posee un motor de manejo de color para la representación de imágenes multicromáticas [11].

El software JasPer consiste en una librería y varias aplicaciones construidas bajo esa librería. El código ha sido escrito en el lenguaje de programación C, éste lenguaje ha sido elegido principalmente por la amplia gama de plataformas de desarrollo en los entornos actuales de computación. Por el momento, JasPer está conformado por 40K líneas de código aproximadamente. Aunque se encuentre escrito en C, la biblioteca de JasPer puede ser fácilmente integrable en aplicaciones escritas en el lenguaje de programación C++.

3.1.2. Bloques funcionales de JasPer

La base del software JasPer se encuentra en su librería. De hecho, la mayor parte del código en JasPer está asociado a esta librería. La librería JasPer provee varias clases para la representación de imágenes, definiciones de los planos de color y otras entidades relacionadas. Cada una de estas clases tiene una interfaz definida con la que una aplicación puede interactuar para manejar los objetos de estas clases. La librería puede usarse para manipular imágenes, importar o exportar datos de imágenes en una variedad de formatos y realizar operaciones de control básicas [12].

Conceptualmente, la librería de JasPer está estructurada como se muestra en la *Figura 3.1*. La librería consiste en dos tipos de código, el código básico y los controladores de los códec. El código básico provee la información sobre la que la librería está armada, mientras que los controladores de códec entregan los métodos de codificación y decodificación de datos de imágenes en varios formatos. Todas las aplicaciones tratan directamente con el código básico. Los controladores de los códec solo pueden ser invocados por el código básico, nunca por una aplicación.

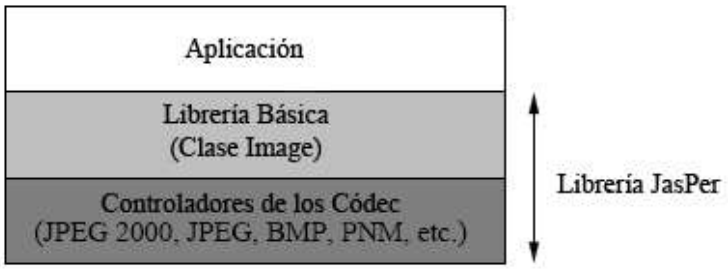


Figura 3.1: Modelo de bloques del sistema JasPer

El soporte de los códec en la librería JasPer es modular y extensible. Existe una interfaz entre el código fuente y los controladores de los códec. Más aún, se puede agregar el soporte para un nuevo formato de imagen sin tener que modificar la librería. Para esto, se tiene que implementar un nuevo controlador de códec especificando el formato a usar. De esta manera, una aplicación puede evitar el costo del consumo aumentado de memoria en controladores de códec que no van a ser utilizados.

3.1.3. Introducción a JJ2000

JJ2000 es un proyecto creado por el trabajo conjunto de EPFL ("Ecole Polytechnique Fédérale de Lausanne"), ERICSSON y CRF ("Canon Research Center France") con el objetivo de implementar una aplicación de java compatible con el estándar JPEG2000. Es una herramienta eficiente, donde convergen todos los requerimientos del estándar asegurando la funcionalidad requerida en este sistema.

3.1.3.1. Objetivos del JJ2000

El estándar JJ2000 tiene como objetivos [13]:

- Funcionar como el software oficial de referencia para el JPEG2000.
- Acelerar la convergencia del proceso de estandarización del JPEG2000.
- Generar las pruebas de verificación del JPEG2000 en términos de rendimiento y capacidades.
- Asegurar la integridad y claridad de las especificaciones del texto en el estándar.

- Facilitar el uso de JPEG2000 y el desarrollo de los productos basados en JPEG2000 en el momento de la estandarización.

- Promover JPEG2000 como un estándar de compresión de imágenes.

Y estos objetivos se alcanzan:

- Desarrollando una implementación en Java del estándar JPEG2000; esto permite una interoperabilidad entre plataformas que ayuda a acelerar la prueba y adopción de JPEG2000.

- Desarrollando una implementación en Java del estándar JPEG2000 directamente desde su descripción técnica; de esta manera, las descripciones técnicas incompletas pueden ser identificadas y corregidas, mejorando la descripción textual del estándar.

- Proveyendo versiones actualizadas del software y la documentación al público.

- Trabajando con los chequeos de interoperabilidad de los grupos ad-hoc apropiados y con intercambios de código fuente entre implementaciones independientes de software tanto para identificar errores de codificación como para verificar la integridad y claridad de las especificaciones.

- Desarrollando demostraciones simples de la eficiencia obtenida con JPEG2000 y publicándolas en una página web.

- Proveyendo materiales tutoriales del JPEG2000 accesibles a los grupos interesados.

3.1.3.2. ¿Por qué una implementación en Java?

La selección de Java como el lenguaje de desarrollo para JJ2000 se basó en las capacidades de independencia de plataforma y portabilidad que tiene Java. El soporte entre diferentes sistemas operativos acelera sus pruebas y por consiguiente acelera la adopción de JPEG2000 como un estándar de compresión. Una implementación en Java también facilita la integración de JPEG2000 en los exploradores y su uso en las aplicaciones de internet, las cuales son clave para que el estándar JPEG200 tenga éxito y acogida. Más aún, el desarrollo de máquinas virtuales de Java en dispositivos como cámaras digitales, PDA,

teléfonos inteligentes e impresoras, facilita una penetración más amplia de JPEG2000 en nichos y aplicaciones clave. [13]

JJ2000 está compuesta por grupos e individuos con un papel clave y experiencia en JPEG2000 como en otros estándares importantes, tales como MPEG. También han participado en varias actividades relacionadas con JPEG2000 dentro de la estructura de proyectos europeos (Eurostill, Spear) y varios consorcios internacionales. Todos los grupos han estado involucrados con la actividad del JPEG2000 desde sus principios. EPFL funciona como el grupo ad-hoc líder en requerimientos y perfiles de JPEG2000 y está a cargo de la definición de las funcionalidades de JPEG2000 en aplicaciones potenciales. Ericsson y CRF actúan como coeditores del estándar JPEG2000. Ericsson además está a cargo de la edición del Modelo de Verificación JPEG2000 y sirve de coeditor de la parte V del estándar JPEG2000, la cual trata del software de referencia. Los grupos de este proyecto también forman parte del consorcio del WG1, encargado de la actividad del JPEG2000. EPFL, Ericsson y CRF han tomado un papel importante en JPEG2000 al proponer e incluir varias tecnologías en el estándar. La siguiente lista resume algunas características de JJ2000:

- El software desarrollado por JJ2000 ha sido elegido como una de las dos implementaciones oficiales del software de referencia para JPEG2000. Como tal, contribuye a la parte V del estándar (software de referencia).

- El software JJ2000 promueve la adopción de JPEG2000 al proveer la única implementación de JPEG2000 con todos los requerimientos del estándar.

- JJ2000 ha sido confirmado con el modelo de verificación de software del WG1, el cual ha sido usado por el comité de JPEG2000 para desarrollar el estándar. En este sentido, la validez del software con el estándar ha sido ampliamente verificada.

- JJ2000 ha jugado un papel importante para identificar tanto inconsistencias en el texto del estándar como errores e inconsistencias en el modelo de verificación de software del comité de JPEG2000.

- JJ2000 tiene un papel importante en la verificación de las características y capacidades de JPEG2000, tratadas en la parte IV del estándar.

3.1.3.3. Bloques funcionales del JJ2000

En esta sección se mostrarán a grandes rasgos los objetos que cumplen con las funciones de codificación y decodificación. Para definir el sistema con más claridad, se explican los bloques funcionales del codificador y, tras esto, se tratan las diferencias que existan entre el codificador y el decodificador.

El sistema usa las clases de los paquetes *"jj2000.j2k.encoder"* y *"jj2000.j2k.decoder"* para controlar todo el proceso, verificar el formato apropiado de las imágenes, definir los límites de teselas, asignar áreas de interés y entregar todos los datos que necesita cada bloque para hacer su proceso. Los programas principales están situados en las clases *"CmdLnEncoder"* y *"CmdLnDecoder",* respectivamente. Se resumen las funciones de estos programas en recibir las instrucciones del usuario, como las direcciones de los archivos de entrada y salida, la tasa de compresión, el número de teselas[1], entre otros.

Las especificaciones son enviadas a los objetos "*Encoder*" y "*Decoder"*, donde tomarán la imagen de la fuente especificada, instanciarán todos los bloques de procesamiento y procederán a arrancar el proceso de codificación o decodificación. Las especificaciones son adquiridas a través de una lista de parámetros ("*ParameterList"*) de los programas principales. Luego invoca el método de arranque de un hilo y ejecuta el proceso requerido. Al terminar su trabajo, lanza un código de salida, el cual es cero si no ocurrió ningún error.

En el caso del codificador, los módulos son insertados en el siguiente orden:

- ***ImgData:*** Esta interfaz define todos los métodos de acceso a los atributos de la imagen, tales como sus dimensiones, el número de teselas, el número de componentes, etcétera. Esta interfaz está implementada por todos los módulos aquí mencionados, entregando a todas las clases la capacidad de manejar la imagen.

[1] Proceso de subdivisión de un área en segmentos regulares, usado para reducir el tiempo de procesamiento.

- ***ImgReader*:** Esta es la interfaz genérica implementada para todos los lectores de archivos, nominalmente para las extensiones PGM y PPM. Un "*ImgReader"* se comporta como un objeto de "*ImgData"*, el cual es el objeto básico que representa una imagen. Esta clase asigna su origen en el punto (0,0) en la malla de componentes, asume que no hay un teselado inicial (entiéndase, que la única tesela equivale a la imagen) y que el sistema no posee muestreo inferior de componentes (que todos los componentes tengan el mismo tamaño). Si es necesario, se pueden sobrecargar las funciones para que puedan admitir los otros requerimientos.

- ***ImgDataJoiner*:** Esta clase implementa la interfaz de "*ImgData"* permitiendo obtener datos de diferentes fuentes. El uso más típico de este objeto se da cuando se quieren codificar varios componentes de diferentes archivos de entrada (entiéndase, de múltiples "*ImgReader"*). Estos dos objetos cubren el bloque funcional de pre-procesamiento, adaptando la imagen a los requerimientos necesarios.

- ***ForwCompTransf*:** Este objeto tiene como tarea el cumplir con el bloque funcional de la transformada intercomponente. Como fue explicado, el estándar JPEG2000 ha definido dos transformadas intercomponente, las cuales son la RCT ("Reversible Color Transform") y la ICT ("Ireversible Color Transform").

- ***Tiler*:** A diferencia del estándar, se hace una teselación de la imagen una vez está en proceso. Se entiende por teselación como el proceso de subdivisión de una imagen para reducir cargas computacionales. Este objeto está encargado de alinear la imagen, teselarla y calcular las conversiones de transformadas si es necesario. Todas las teselas son rectangulares, no superpuestas y la unión de todas las teselas cubre la imagen. Sin embargo y como fue previamente explicado, las teselas no tienen que ser del mismo tamaño por los efectos de borde.

- ***ImgDataConverter*:** Esta clase es implementada como una interfaz de adaptación con el objeto de evitar problemas de compatibilidad e intercambiando la fuente de los datos si se requiriese.

- ***ForwardWT*:** Esta clase es la que se encarga de las funciones del bloque de transformada intracomponente, donde reduce los bloques de datos a las subbandas necesarias usando el PR-UMDFB antes explicado. Algo importante de notar es que el "*ForwardWT"* es una clase abstracta implementada con un objeto llamado "*ForwardWTFull"*, el cual puede usar otros métodos de síntesis si es necesario.

- ***Quantizer*:** Este bloque se encarga del proceso de cuantización según se ha definido en el estándar JPEG2000. Como se explica en el anexo, el objeto "*Quantizer"* toma como entrada las subbandas de la transformada wavelet y entrega los índices de cuantización en la notación de signo-magnitud con zona de nulidad.

- ***ROIScaler*:** Luego de tener los índices de cuantización de las diferentes subbandas, se activan los algoritmos de región de interés. Esta traslación de los datos más importantes marca la primera fase de la codificación de primer orden, para ser enviados al codificador de entropía.

- ***EntropyCoder*:** El codificador de entropía engloba todas las acciones de las que el codificador de primer orden se encarga. La salida de este módulo son los grupos de pasos de codificación resultantes de los pasos de significancia, refinamiento y limpieza.

- ***PostCompRateAllocator*:** Este objeto está encargado de cumplir con las áreas del bloque de control de tasa. Debe poder crear las capas de diferenciación de acuerdo con los rangos de compresión deseados, como seleccionar los pasos de codificación relevantes para ser empaquetados por el módulo de codificación de segundo orden.

El codificador usa un modelo en cascada. Esto significa que cada módulo usa los datos del módulo anterior como base de su proceso. Luego de comprimir los datos, usa los objetos "*HeaderEncoder"* y "*CodestreamWriter"* para escribir el código final; estos dos objetos engloban la funcionalidad de la codificación de segundo orden.

Gran parte de los módulos se comportan diferentemente dependiendo de los componentes de tesela. Las especificaciones de su comportamiento son guardadas en extensiones de sus

clases conocidas como "*ModuleSpec"*. Todos estos módulos son accedidos por una instancia de la clase "*EncoderSpec"* (la cual está en el paquete "*jj2000.j2k.encoder"*).

El decodificador trabaja de la misma manera que el codificador. Dependiendo si se desea guardar la imagen o sólo se desea ver su contenido, el decodificador elije uno de dos módulos finales. El proceso de decodificación usa la siguiente lista de objetos en cascada:

- ***BitstreamReaderAgent*:** Este objeto se encarga de leer el código empaquetado y actuar como decodificador de segundo orden, obteniendo la información relevante del archivo y entregando datos importantes para el proceso, especificados en el encabezado de la imagen.

- ***EntropyDecoder*:** Análogamente, el *EntropyEncoder*está encargado del módulo de la decodificación de primer orden, entregando los índices de cuantización de la imagen recuperada.

- ***ROIDeScaler*:** Si se ha hecho una traslación de un área de interés, este objeto invierte el proceso para obtener la imagen apropiada.

- ***Dequantizer*:** Este objeto se encarga de invertir el proceso de cuantización, retornando las subbandas de la imagen.

- ***InverseWT*:** Como se infiere de su nombre, el *inverseWT* está encargado de recuperar la imagen, revertiendo la transformada intracomponente. Este objeto tiene la capacidad de controlar el nivel de resolución al cual reconstruir la imagen, que no tiene que ser necesariamente el nivel máximo de reconstrucción. Se asume por simplicidad computacional que todas las teselas serán reconstruidas al mismo nivel.

- ***ImgDataConverter*:** Este es el único objeto que tienen en común el codificador y el decodificador. Su función es exclusivamente como un adaptador de los datos internos. Aquí también se asignan los datos de múltiples fuentes, para ser unidas.

- ***InvCompTransf*:** Este objeto se encarga del segmento de la transformada inversa intercomponente, en caso de haber sido utilizada.

- ***ImgWriter*:** Suponiendo que el usuario haya designado un archivo de salida, el "*ImgWriter"* funciona como bloque de adaptación, entregando un archivo de imagen sin compresión, con la información de los componentes.

- ***BlkImgDataSrcImageProducer*:** En caso de no haberse designado un archivo de salida, la interfaz de despliegue se activa mostrando la imagen resultante en el plano de la escala de grises o en el plano RGB.

3.1.4. Directrices de Selección

Considerando las dos plataformas de desarrollo sobre las que se puede implementar el sistema de compresión de tramas de voz, se enumeran las directrices sobre las que se hace la selección del entorno. Estas directrices se establecen alrededor del objetivo principal del sistema, el cual es el desarrollar un sistema de compresión y descompresión de tramas de voz sobre una plataforma de compresión y descompresión de imágenes basada en el estándar JPEG2000. Por lo tanto, las directrices que rigen la elección del sistema son:

- Debe ser un sistema abierto, modular y flexible, que permita la vista de los procesos internos del sistema.

- Se debe preservar la integridad del sistema de codificación de manera que, al implementar la aplicación, el codificador mantenga la capacidad de comprimir y descomprimir imágenes.

- El nivel de abstracción debe ser preferiblemente bajo, para no tener que realizar procesos redundantes en la implementación de la interfaz.

- Se debe procurar que la aplicación trabaje transparentemente hacia el usuario, sin que tenga que influir en el proceso final.

Considerando estas guías, se ha decidido usar el sistema de JJ2000 pues, si bien la flexibilidad del JasPer permitiría una implementación más rápida del codificador, el nivel de abstracción de JJ2000 permite ver el proceso de compresión y descompresión de imágenes hasta el punto donde podemos controlar sus métodos internos.

3.2. Fase de Diseño

Habiendo descrito el sistema de compresión de datos del JPEG2000, se pasa a mostrar el proceso de desarrollo para implementar una aplicación híbrida de compresión de voz utilizando las herramientas del JJ2000, además de las pruebas que se tomarán para su respectiva evaluación.

Para mayor claridad del proceso elegido, se sintetiza el sistema de compresión y descompresión especificado en [14] en los bloques de: Adaptación de señal, compresión y codificación de archivo mostrada en la *figura 3.2*.

- **Adaptación de señal:** El sistema JJ2000 recibe como entrada válida archivos de imagen monocromáticos o multicromáticos sin compresión (PGM o PPM), para convertirlos en una imagen interna. Este bloque toma las características de la imagen para poder elegir un camino de codificación apropiado.

- **Compresión:** El codificador y el decodificador se comportan tal como es definido en el ANEXO B, pasando por los módulos de preprocesamiento, teselación, codificación intercomponente, codificación intracomponente, cuantificación y codificación en primer orden para obtener un archivo compreso equivalente a la imagen.

- **Codificación del Archivo:** Este bloque trabaja con el módulo de codificación de segundo orden, anteriormente mostrado en las especificaciones del estándar. Más específicamente se usará la herramienta de codificación del archivo y encabezado.

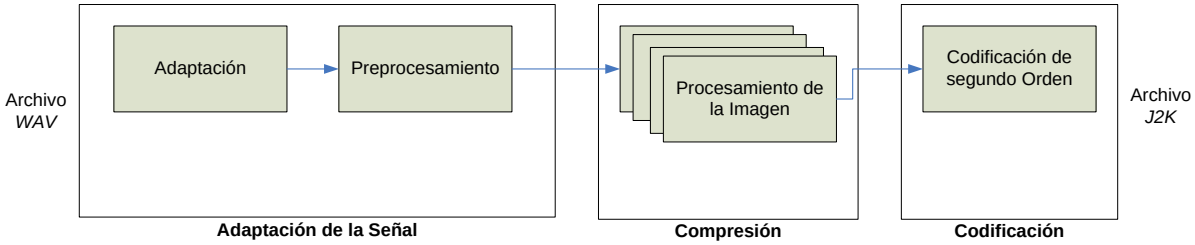


Figura 3.2. Diagrama de Bloques del sistema.

Al buscar el procedimiento apropiado para codificar los archivos de voz en imágenes compresas de información equivalente, se tiene que tratar con tres temas importantes: La construcción de una interfaz que permita al JJ2000 reconocer el archivo de voz, la diferenciación entre los datos internos, más precisamente entre las muestras de los diversos canales, y la diferenciación entre los datos y el encabezado del archivo de voz, en la *figura 3.3*. se mostrará la definición de un nuevo objeto en la parte de adaptación.

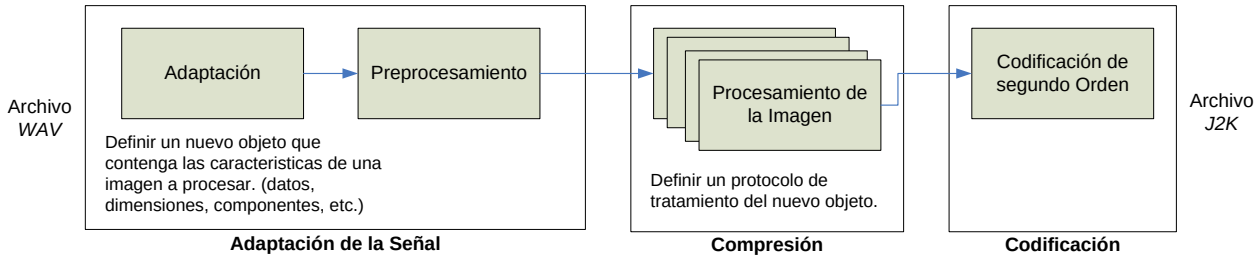


Figura 3.3. Primera propuesta del sistema de compresión.

Al implementar la interfaz de datos, se plantean dos posibles opciones a seguir. La primera es una modificación directa de la herramienta, manipulando los bloques de adaptación y codificación de tal manera que pueda reconocer y tratar un archivo de voz. La otra opción es una implementación externa a la aplicación, donde se hace un tratado del archivo con los datos de voz para convertirlo en una imagen, y con esta imagen dejar que la herramienta procese lo necesario, los módulos se muestran en la *figura 3.4.* Se eligió la última de las opciones, pues a pesar de que la primera opción ofrece una mejora en el rendimiento, lo consigue a costa de la eliminación de la integridad del código hasta el punto donde dejaría de funcionar para su propósito inicial.

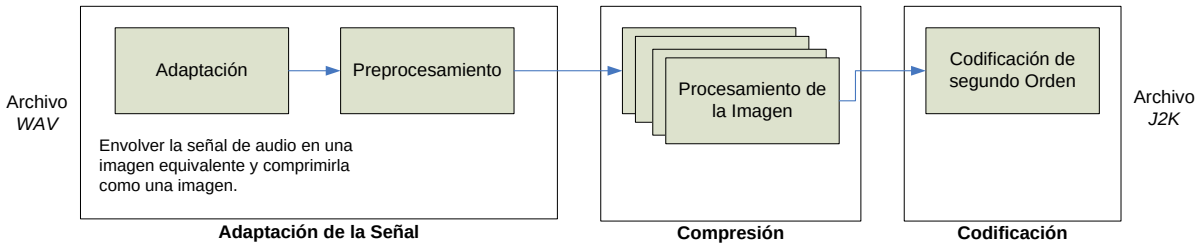


Figura 3.4. Segunda propuesta del sistema de compresión.

En el aspecto de la diferenciación de datos en la codificación y decodificación, se observó un aspecto interesante en el funcionamiento del codificador, que consiste en que la capacidad de compresión del JJ2000 aumenta entre más coherencia haya en la imagen, dando mejores resultados. En el formato de organización de un archivo WAV, las muestras de cada canal se entregan secuencialmente, donde cada "n" muestras consecutivas representan a los "n" canales en un instante de tiempo, como se muestra en la *figura 3.5*. Considerando estos factores, se eligió hacer un proceso de división de datos por canal, para que cada canal tuviese su información respectiva en el transcurso del tiempo requerido. De esta manera, los datos de una muestra en un lugar cualquiera mostraban una coherencia mayor entre ellos. Adicionalmente, por las restricciones internas de la herramienta, una muestra no puede sobrepasar el rango entre 0 y 255, eliminando muchas posibilidades en el ámbito de almacenamiento de muestras. Con esta restricción se debe hacer una división adicional por el número de bits por muestra, agrupando los bits más significativos y los menos significativos en zonas separadas.

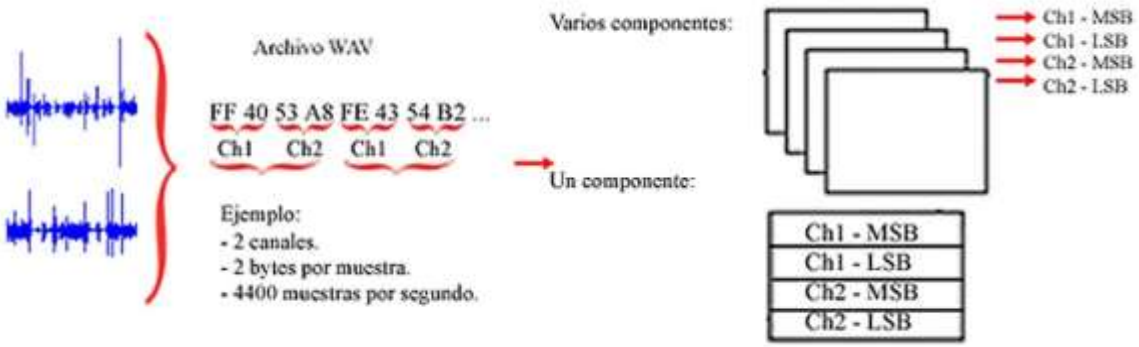


Figura 3.5. División de una imagen por canales.

Otro aspecto importante a tener en cuenta es que mientras los datos de voz pueden ser procesados para reducir la redundancia, los datos del encabezado se deben preservar exactamente, ya que un bit errado podría dañar el funcionamiento del archivo de voz o volverlo un archivo no válido. Para esto se pasó por tres fases distintas. Las tres fases consisten en tratar al archivo como un todo ("wrapping"), diferenciar el encabezado de los datos y procesar dos veces, e integrar el encabezado en un segmento aparte a los datos.

Inicialmente, el archivo PGM es empaquetado completamente como una imagen única. El proceso es análogo a la segunda propuesta del sistema mostrado e la figura 3.4. El segmento de adaptación mide el tamaño total del archivo, luego calcula una matriz bidimensional que pueda albergar el archivo y, finalmente, segmenta el archivo y rellena la matriz a procesar. Este método permite un proceso rápido y sencillo, pero no se pueden usar compresiones con pérdida sin comprometer los datos del encabezado.

Considerando la poca eficiencia que entrega el primer prototipo, se decide aislar los datos del encabezado y comprimirlos en archivos separados, donde se adquieren los datos del número de canales, bits por muestra y muestras totales del encabezado para ser usadas como bases para albergar y diferenciar los datos. Este proceso adquiere la ventaja de comprimir con pérdidas los datos respecto al primer prototipo, pero la eficiencia se reduce al tener que trabajar el proceso de compresión dos veces, especialmente cuando el tamaño del archivo del encabezado termina siendo más grande que la información que alberga, los módulos para esta etapa se muestran en la figura 3.6.

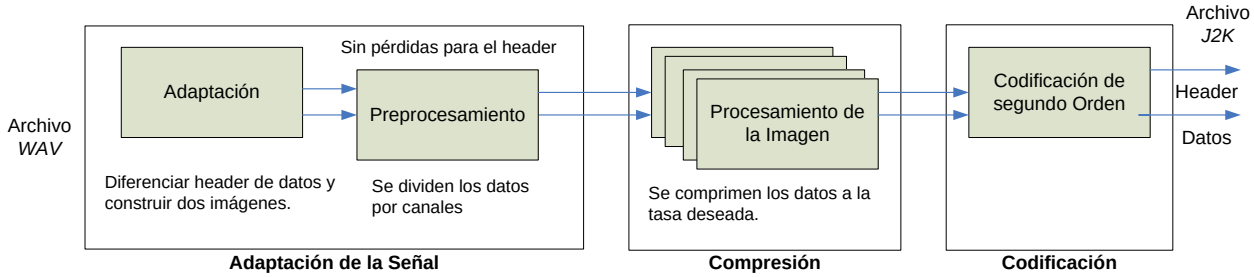


Figura 3.6. Tercera propuesta del sistema de compresión.

Posteriormente, se decide modificar directamente el funcionamiento interno del programa. Al tomar esta decisión, se tuvo que tener en cuenta que, como se había estipulado previamente, uno de los objetivos era cuidar la integridad del código, para que no se perdiesen las funcionalidades básicas del sistema.

Pensando en esto, se modificaron las listas de parámetros de la herramienta, indicándose la localización de los datos del encabezado para ser agregados, introduciendo un cambio en el módulo de procesamiento de segundo orden para que usase la caja de comentarios del estándar JPEG2000 como alojamiento de dichos datos. Este método permite generar un proceso único y, al usar un parámetro opcional controlado externamente, asegura que la

herramienta continúe funcionando para lo que está diseñada, en la figura 3.7 se muestra la *Propuesta Final del Sistema de Compresión.*

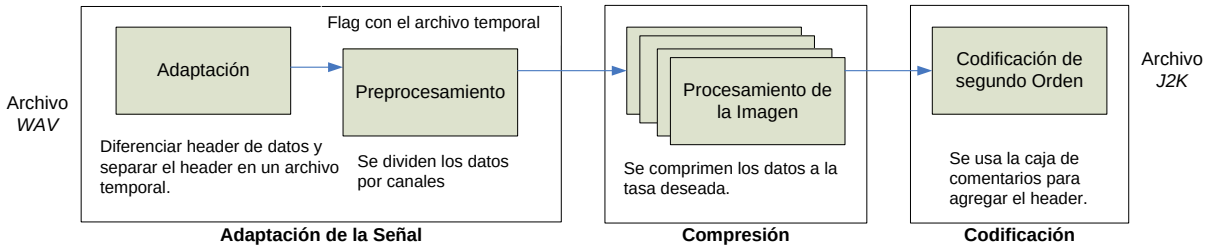


Figura 3.7. Propuesta final del sistema de compresión.

3.3. Fase de Implementación

Habiéndose descrito el proceso de trabajo, se pasa a explicar cada una de las fases iterativas que se dieron durante la creación del sistema y las decisiones tomadas en cada una de las fases de trabajo. Se dividirá esta sección en tres partes, las cuales son *implementación de la interfaz, diferenciación de los datos a comprimir y diferenciación del encabezado y los datos*, de acuerdo a los tres aspectos del problema mencionados en el proceso.

3.3.1. Implementación de la interfaz

Inicialmente se descubre en las etapas exploratorias de la plataforma de desarrollo JJ2000 que se permiten varias modificaciones al tratamiento desde las listas de parámetros externas, asignando uno u otro bloque a este proceso. Aun así, la mayor parte de decisiones se toman por defecto si no se agregan los parámetros que digan lo contrario. Tomando en cuenta estos casos, se propone inicialmente el generar un protocolo paralelo creando internamente un objeto de imagen, implementando la interfaz "*ImgData",* que represente al archivo de voz, guiando el proceso hasta que el sistema pueda seguir sin intervención alguna.

La propuesta inicial consta de tres segmentos: un lector de archivos WAV con los procesos y protocolos, una matriz de datos que implemente la interfaz "*ImgData"* y se comporte como una imagen, y un modificador en las listas de parámetros que indique un cambio en el proceso interno. Por cada uno de esos módulos de codificación, existe un módulo análogo

para invertir el proceso. Los diseños básicos de la herramienta se establecen hasta el punto donde se tiene que manipular el proceso interno de la máquina, descubriendo que se hacen múltiples verificaciones a lo largo de cada decisión para asegurarse que está manipulando el archivo deseado, cuya información es llevada a los codificadores de primer y segundo orden para generar un archivo con su información compresa. Esto implica que si se toma la decisión de anular estas verificaciones, sobrescribiendo los comandos internos, deja de funcionar como un codificador de imágenes, más aún, se saltaría muchos pasos del estándar para poder comprimir una imagen falsa.

Tras haber detectado este problema, se prefiere retroceder en el proceso y usar dos módulos de adaptación, externos al sistema de compresión, que actuarían como un módulo de interfaz entre el sistema y la aplicación, recibiendo el archivo de voz indicado y entregando una imagen equivalente que contenga los datos a comprimir.

Para elegir el método de transformación de voz a imagen, se muestran seguidamente los objetivos a cumplir. Primero, un archivo de sonido no compreso en formato WAV se puede definir como "n" grupos de muestras unidimensionales, donde cada muestra especifica un tono en un instante de tiempo, siendo usualmente "n" igual a uno para la voz o mayor a uno si es un archivo de audio, cada grupo de muestras se asignan a un canal, donde cada muestra puede tener uno o más bytes de profundidad.

Una imagen son "k" grupos de muestras matriciales, donde cada muestra especifica una cromaticidad en un punto espacial definido. La variable "k" puede ser uno ($k = 1$) o tres ($k = 3$) dependiendo si la información es en escala de grises o a color. Cada matriz es conocida como un componente, y cada muestra en el componente puede albergar un byte de información. Se pueden notar las similitudes y divergencias entre los dos procesos, y qué factores se deben tomar en cuenta para diseñar una interfaz entre ambos formatos.

Considerando la conformación de los diferentes archivos, se define la transformación de vector a matriz. Debido a los requerimientos de eficiencia en procesamiento, se necesita que la transformación de vector a matriz sea lo más sencilla posible, para reducir la carga computacional y aumentar la eficacia. Con eso en mente, se decide hacer un proceso de

segmentación del vector, calculando una matriz en donde puedan albergarse los datos para luego extraerlos y recuperar la trama de voz directamente.

3.3.2. Diferenciación de los datos a comprimir

Una vez definida la metodología a seguir respecto a la transformación de las tramas de voz, se hacen varios procesos de prueba, utilizando tramas de voz segmentadas y envueltas en la caja de menores dimensiones $2^i \times 2^i$, donde "i" se calcula considerando el tamaño del archivo de voz. Se eligen potencias de 2 como tamaños por el funcionamiento interno del estándar. Esta técnica no genera grandes compresiones dado que el espacio vacío se debe llenar con información inútil, reduciendo su eficiencia. Considerando este factor, se decide modificar el algoritmo que elige los tamaños de la imagen por $2^i \times a$, donde "a" es la división entre el tamaño del archivo y 2^i.

Al experimentar con el sistema se descubre que, al mantener una mayor coherencia de los datos, se puede alcanzar una tasa de compresión más alta; por ello se decide hacer una separación de las muestras por canales, colocando cada canal en una zona diferenciada dentro de la matriz de datos. Adicionalmente, en los resultados obtenidos incide el formato elegido, en este caso PGM, el cual sólo puede albergar datos con profundidad de un byte por muestra. Al considerar esta situación, es necesario dividir las muestras no solamente por canales, sino también por número de bits por muestra.

Para poder identificar estos datos, se implementa un sistema que lea el encabezado del archivo de audio, usando el campo de alineamiento de bloque. Este campo es el producto del número de canales con los bits por muestra. Conociendo este dato, se procede a dividir el componente por el número de canales y por el número de bits de profundidad.

3.3.3. Diferenciación del encabezado y los datos

Este aspecto es parte fundamental del proceso de diseño, hasta tal punto que las mejoras incluidas en este aspecto se pueden relacionar directamente con la eficiencia del sistema de compresión. El sistema pasa por tres etapas respecto a esta fase del proceso, entregando mejoras sustanciales al prototipo en cada iteración. Inicialmente, tanto para hacer pruebas de la herramienta como para establecer un punto inicial de trabajo, se propone un sistema

simplificado, en el que el módulo de adaptación de la herramienta toma la información del archivo WAV sin analizarse, para segmentar y reorganizar las muestras como un archivo PGM.

Al analizar este proceso se encuentra que, aunque era bastante ligero en términos de procesamiento, no suple las necesidades apropiadas de compresión, entregando archivos irrecuperables si se asignaba una compresión con pérdidas.

La segunda propuesta consiste en la diferenciación del encabezado de los datos, identificando la bandera DATA, la cual muestra el final del encabezado y el comienzo de los datos a comprimir. Cada uno de estos datos son comprimidos por separado, entregando dos archivos *jp2*. Este método posee las ventajas de permitir una compresión con pérdidas en los datos y la utilización de los datos del encabezado para generar una coherencia entre los canales de un componente, sin embargo resulta ser muy dispendioso en términos de procesamiento, teniendo que comprimir dos archivos por separado. Adicionalmente se nota que el encabezado era muy pequeño para tratar como un archivo independiente, el cual termina entregando un *jp2* más grande que el encabezado mismo, por los datos agregados que se incluyen en la codificación de segundo orden.

Finalmente se decide hacer una modificación paralela a la herramienta definida en el sistema de compresión base. Al leer las instrucciones de la codificación de segundo orden y los objetos de codificación de encabezado, se nota que la herramienta de compresión tiene una función por defecto en el campo de escritura de la caja de comentarios, entregando una información innecesaria acerca de la versión del codificador. Al haber descubierto esto, se modifica el proceso de codificación de segundo orden de la imagen para incluir el contenido del encabezado en la caja de comentarios. Esta información se entrega exclusivamente si se desea, por lo que se agrega un comando a la lista de parámetros de manera que la herramienta siga utilizando la función por defecto en caso de que no se necesite. Este procedimiento facilita la compresión de los datos, preservando la información del encabezado intacta, entregando un único archivo y un único proceso, el cual no interfiere con el funcionamiento usual de la herramienta.

3.4. Modelo del Sistema de Compresión.

Esta sección trata con el sistema de compresión y descompresión de datos, identificando los elementos que lo componen y explicando las funciones definidas a lo largo del proceso elegido. Es importante notar en este momento que, debido al diseño del modelo de compresión, existen dos programas principales aparte de los especificados en el estándar para englobar las funcionalidades del sistema sin impedir el funcionamiento normal del JJ2000.

El programa se diseña en dos clases que manejen el proceso, "*PseudoMusicCoder"* y "*PseudoMusicDecoder"*, en donde se encuentran los módulos de adaptación y pre-procesamiento. Éstos objetos son invocados por las clases "*MainEncoder"* y "*MainDecoder"*, las cuales controlan el flujo de los datos, el rango de compresión, la localización de los archivos de entrada y salida, y cualquier otra característica pertinente al sistema JJ2000.

Los métodos implementados en la clase PseudoMusicCoder son los siguientes:

- **wrap:** Este método se genera para controlar el flujo de la interfaz. Este proceso instancia el archivo de audio especificado en "*wavfile*", llama a las funciones apropiadas para separar el encabezado de los datos, luego llama a las funciones que reorganizan el archivo como una imagen y finalmente arranca el codificador del JJ2000 con los parámetros de la imagen.

- **readHeaderWavFile:** Este método se encarga de encontrar el encabezado en el archivo de audio ingresado, buscando la bandera definida como el comienzo de los datos. Luego de encontrar la bandera apropiada, guarda el contenido del encabezado en un vector de bytes y los retorna para futura referencia.

- **readContentWavFile:** Luego de averiguar el contenido del encabezado, se invoca el proceso de lectura del archivo, el cual define las dimensiones de segmentación y retorna los datos para ser procesados.

- **writeHeader:** Esta función se encarga de asignar el contenido del encabezado en un archivo temporal, archivo que se accederá posteriormente para ser ingresado en el campo de comentarios.

- **writePGMFile:** Una vez tiene el contenido del archivo WAV, este método escribe la información de los datos, segmenta el archivo de audio, divide la información en el número de canales y bytes por muestra, y escribe la información en la imagen equivalente.

- **writePGMHeader:** El método aquí mencionado se encarga de recibir las dimensiones de segmentación y escribe el encabezado del archivo PGM.

- **runEncoder:** Por último, este método ensambla una lista de parámetros a pasar al codificador y procesa el archivo a ensamblar.

Análogamente, los métodos implementados en la clase PseudoMusicDecoder son:

- **runDecoder:** De la misma forma que el runEncoder, este método genera una lista de parámetros indicando el lugar en donde entregar el encabezado y los datos para ser rearmados.

- **unwrap:** Este proceso, paralelo al wrap del codificador, coordina las acciones en el sistema para tomar el contenido decodificado, recuperar las especificaciones del archivo, llamar las funciones de reestructuración de acuerdo a las especificaciones y retornar un archivo de voz los datos recuperados.

- **readHeader:** Esta función se encarga de recuperar la información guardada en el encabezado, la cual ha sido extraída de la caja de comentarios.

- **readLong:** Este proceso se tuvo que implementar para determinar la longitud total del archivo, al leer la información en el header, el cual manejaba datos sin signo y de big-endian.

- **readPGMFile:** Este método se encarga de tomar los datos en la imagen recuperada, reorganizando las muestras en el archivo y entregando el archivo recuperado.

4. APORTES REALIZADOS POR LOS ESTUDIANTES

A continuación se presentan los resultados obtenidos de las pruebas realizadas en donde se visualizan los resultados del código, detallando sus ventajas y mostrando además los inconvenientes que puede tener durante una transmisión [7]. Estas pruebas se realizaron utilizando MATLAB_7.1 software suministrado por la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca. Las respectivas muestras de voz son capturadas con el programa Camtasia Studio v5 el cual es de libre distribución.

4.1. Análisis de los Módulos del Sistema

Para observar la eficiencia del algoritmo llamado JPEG2K_VC se realizaron pruebas utilizando diferentes grabaciones de personas comunes, las cuales son de duración corta, mediana y extensa, enfocando el análisis en la comparación del tamaño original del archivo con su respectiva compresión, se variará la tasa de compresión para determinar hasta cuanto se puede llegar a codificar, determinando de esta manera la tasa más apropiada para la compresión de voz (monofónica). Además cabe destacar que se van a tomar muestras de sonidos de varios canales para hacer los análisis respectivos.

Las características de las grabaciones son: formato PCM, la frecuencia de muestreo 8KHz con 16 bits y con canal MONO. Estos valores se pueden cambiar utilizando la ventana que proporciona el programa de Camtasia.

El código diseñado en JAVA se ejecuta utilizando el programa de NetBeans v5.5 de libre distribución.

Como muestras a manipular, se ha conseguido tramas de voz de 361KB, 429KB, 883KB, 1.840KB, 2.666KB y 4.000KB, esta tiene un tamaño de 4MB, un tamaño considerable con el que se puede colocar a prueba la eficiencia del código diseñado. Para entender y explicar mejor la fase de pruebas se tiene en cuenta el esquema general del sistema de compresión y descompresión el cual se muestra en la *figura 4.1*,

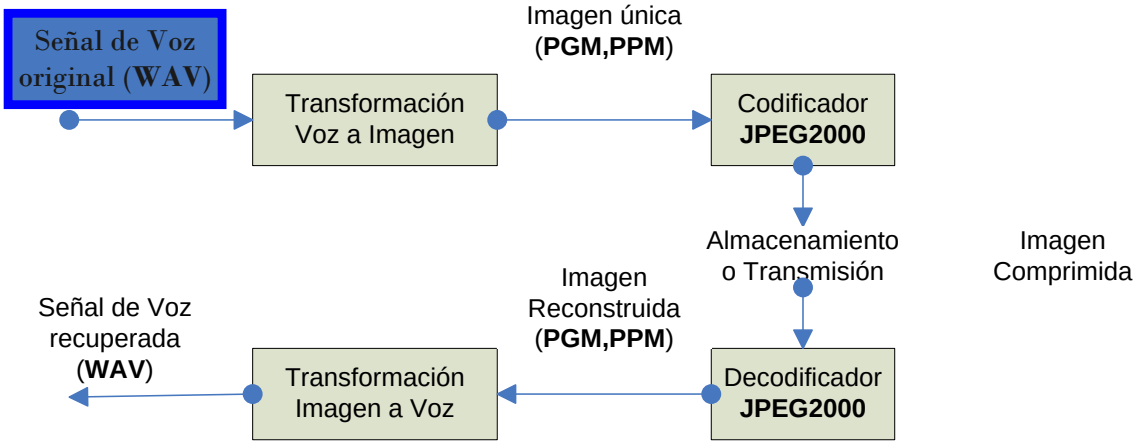


Figura 4.1. Sistema general de compresión y descompresión de voz (Modulo 1).

En la entrada del primer módulo están las señales de voz grabadas, archivos nombrados *señaldevoz1*, *señaldevoz2*, *señaldevoz3*, *señaldevoz4* y *señaldevoz5*, cuya grafica en el dominio del tiempo vienen dadas por las graficas (a), (b), (c), (d) y (e) respectivamente, mostradas a continuación,

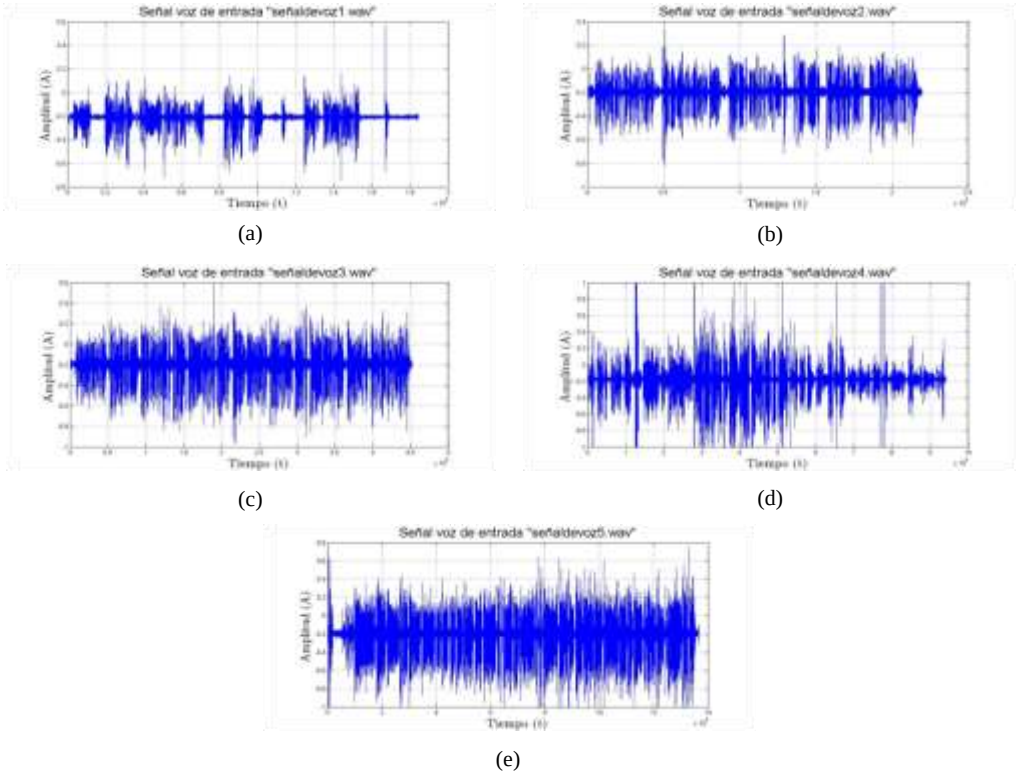


Figura 4.2. (a),(b),(c),(d) y (e) señales de voz a la entrada del sistema general.

Hasta aquí se han mostrado las representaciones de las señales que se van a comprimir, ahora se pasa al análisis de estos archivos en formato PGM, con lo que ya se pasa al modulo 2 del esquema general del sistema (*figura 4.3*)

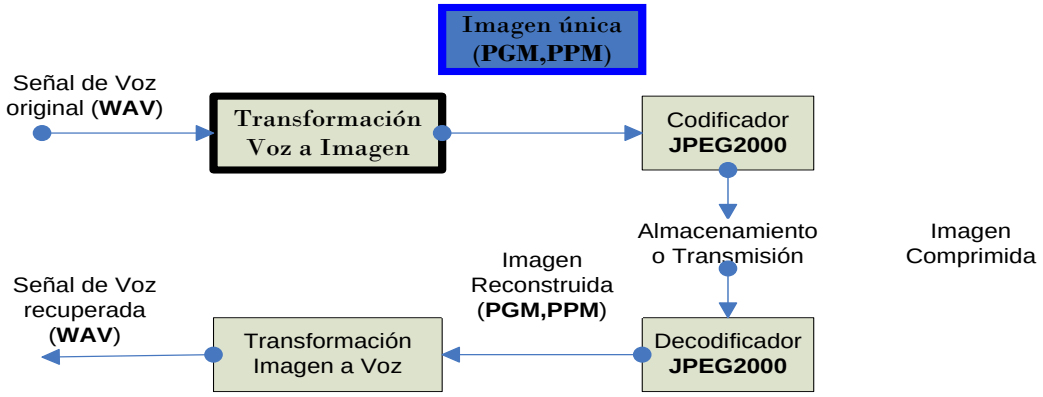


Figura 4.3. Sistema general de compresión y descompresión de voz (Modulo 2).

Los respectivos archivos *.pgm* que se obtienen luego de que las tramas de voz pasan por el módulo de adaptación, cuya función es el de convertir la voz (unidimensional) a imagen (bidimensional), se muestran en las *figuras 4.4 a 4.8*, en donde se observa que cada archivo de voz le corresponde una imagen única. Se mostrará entonces las imágenes con extensión *.pgm* utilizando un visualizador de imágenes llamado *ImageEye* de libre distribución, cabe mencionar que el tamaño del archivo de voz original ha aumentado al transformarse en una imagen PGM, debido al proceso de agregado necesario para transformarlo en imagen,

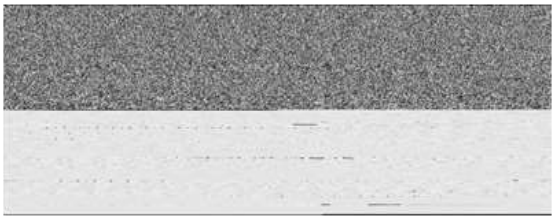


Figura 4.4. Imagen PGM única de la señal de voz (señaldevoz1.wav)

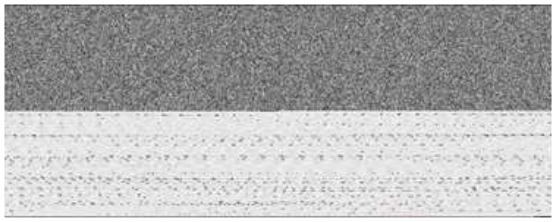


Figura 4.5. Imagen PGM única de la señal de voz (señaldevoz2.wav)

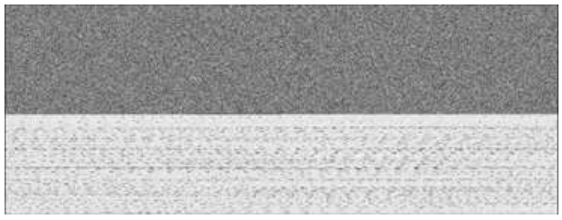


Figura 4.6. Imagen PGM única de la señal de voz (señaldevoz3.wav)

Figura 4.7. Imagen PGM única de la señal de voz (señaldevoz4.wav)

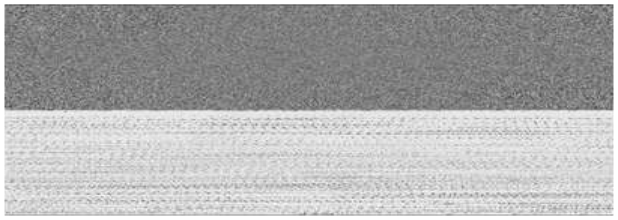


Figura 4.8. Imagen PGM única de la señal de voz (señaldevoz5.wav)

Seguidamente, las imágenes *PGM* pasan a ser tratadas por el *codificador* modificado del JJ2000, el cual recibe el formato de imagen única de voz en su respectivo formato *PGM*, para aplicarle la compresión arrojando el archivo de voz en formato *J2K*. Siguiendo el modelo que describe el sistema en la *figura 4.9* se puede ver la parte en donde está ubicado

el respectivo codificador y los respectivos archivos en formato *.j2k* se pueden ver en las *figuras 4.10 a 4.14.*

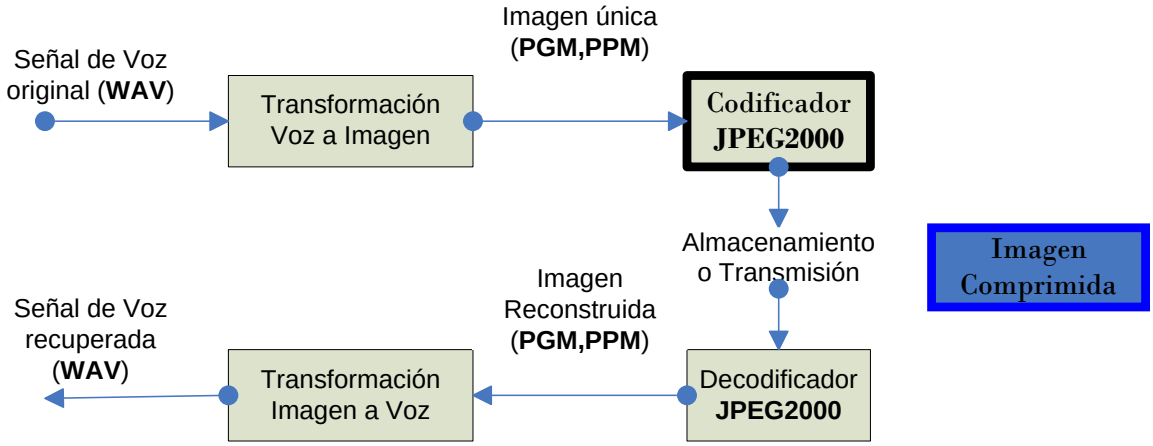


Figura 4.9. Sistema general de compresión y descompresión de voz (Modulo 3-4).

Figura 4.10. Imagen J2K única de la señal de voz (señaldevoz1.wav)

Figura 4.11. Imagen J2K única de la señal de voz (señaldevoz2.wav)

Figura 4.12. Imagen J2K única de la señal de voz (señaldevoz3.wav)

Figura 4.13. Imagen J2K única de la señal de voz (señaldevoz4.wav)

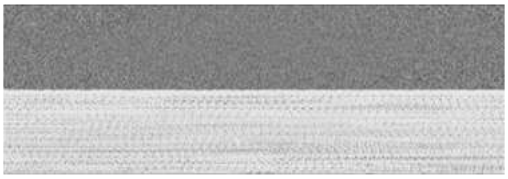


Figura 4.14. Imagen J2K única de la señal de voz (señaldevoz5.wav)

Si detallamos minuciosamente las graficas J2K se pueden observar pequeños cambios en su lineamiento, estas imágenes tienen un mayor brillo que las obtenidas en PGM, además de adicionarle contraste, lo cual muestra el ruido de cuantización agregado, aunque este varía con relación a la tasa de compresión que se le aplique.

Una vez adquirida la imagen codificada se almacena temporalmente para luego invertir el proceso, aplicado a este archivo para recuperar la información inicial. Estos procesos forman parte de los bloques de decodificación y transformación de imagen a voz.

La *figura 4.15* muestra los subproductos del proceso de compresión del archivo *señaldevoz1.wav,* desde la señal original, mostrando la imagen equivalente, el archivo compreso, la imagen recuperada y el archivo de voz restaurado de la imagen. Estas gráficas muestran el proceso aplicado luego de una compresión con pérdidas sin tasa asignada, con lo que el codificador asume una compresión mínima y una fidelidad máxima.

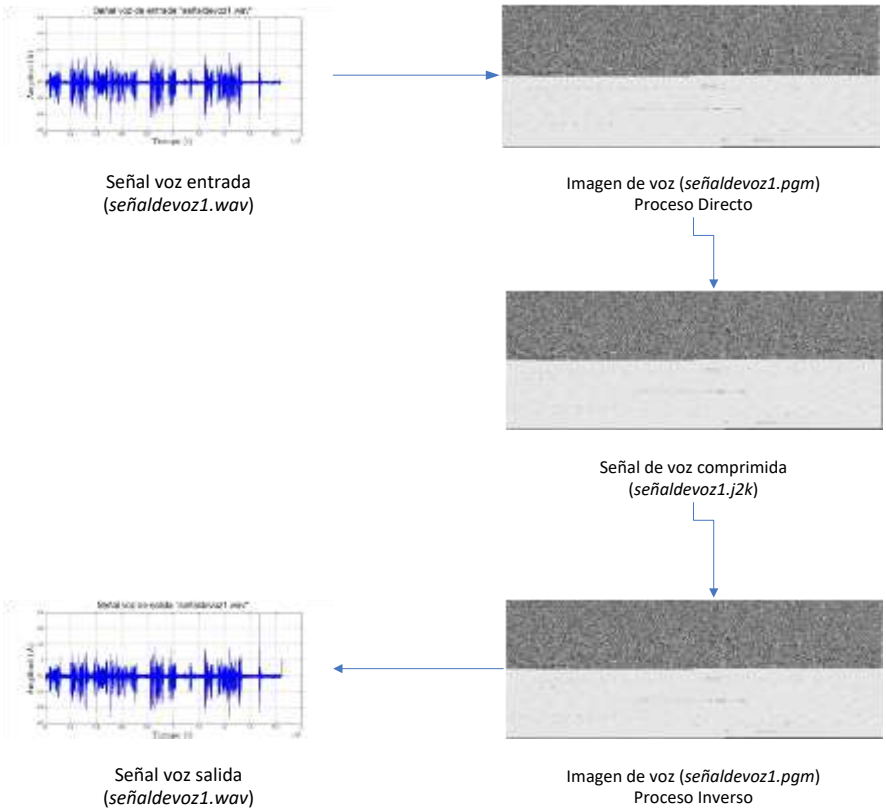


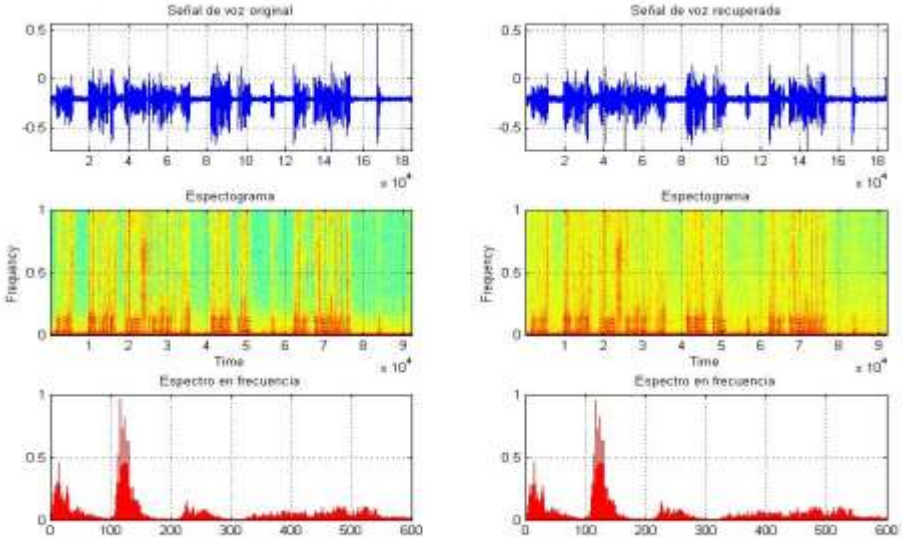
Figura 4.15. Proceso detallado de la compresión y descompresión de voz.

4.2. Análisis Espectral de las Señales.

Hasta ahora se han detallado los cambios que ha tenido la voz cuando pasa por cada uno de los módulos del sistema de compresión y descompresión, las señales se muestran en el dominio del tiempo y en formato de imagen. Ahora se utilizará la información de entrada (señal de voz grabada para ser procesada) y la de salida (señal de voz después de pasar por todos los submódulos del sistema), con el fin de mostrar la eficiencia del código diseñado.

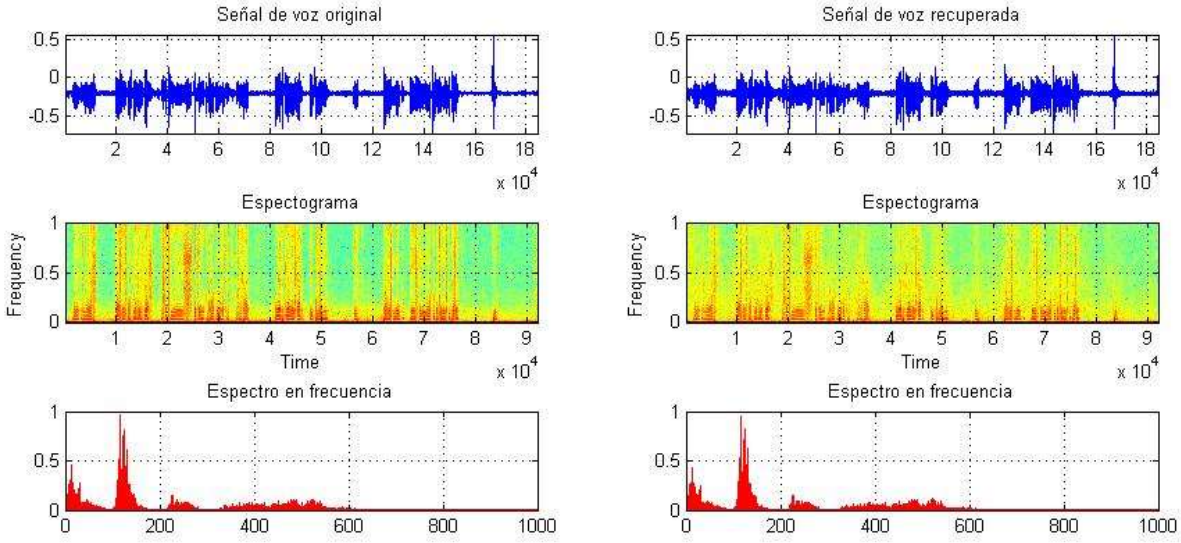
En este momento, se buscará una tasa de compresión óptima para el sistema. Para esto se varia la tasa de bits por muestra, y se comparan los archivos con la información compresa para ver su grado de compresión (observando la tasa que nos proporcione una compresión suficiente) y las muestras recuperadas (buscando los archivos que sean entendibles al oído humano).

Las señales que se están analizando tienen tamaños de 361 KB y 2.666KB, a una tasa de valor 6 se obtuvieron archivos J2K de tamaños 222KB y 1.999KB y con porcentajes de compresión de 38.5% y 25.02% respectivamente, aunque se está utilizando la misma tasa los porcentajes no son similares, con lo que se puede concluir que el tamaño del archivo de voz influye de manera inversa en estos valores de porcentaje. En las *figuras 4.16 y 4.17* se muestra la señal de voz de entrada (*señldevoz1.wav* sin comprimir) y la señal de voz de salida (*señaldevoz.wav* comprimida), a tasas de 6 bit por muestra y 4 bit por muestra respectivamente, mostrando su contenido en el dominio del tiempo, tiempo-frecuencia (espectrograma) y en el dominio de la frecuencia con el propósito de observar los cambios más significativos que se han tenido durante el proceso de compresión. Entre las características que se pueden destacar esta el ruido blanco que se le adiciona a la señal de salida.



(a) (b)

Figura 4.16. Muestras del archivo señadevoz1.wav a 6 bit por muestra (a) Análisis de la señal de voz de entrada. (b) Análisis de la señal de voz de salida.



(a) (b)

Figura 4.17. Muestras del archivo señadevoz1.wav a 4bpm (a) Análisis de la señal de voz de entrada. (b) Análisis de la señal de voz de salida.

Al utilizar una tasa de cuatro bits por muestra, la señal de salida presenta un incremento en el ruido, y una reducción en la energía total recuperada, mostrados en la *figura 4.18.*

4.3. Error Cuadrático Medio de la Señal Recuperada.

Consiste en la suma de las diferencias al cuadrado entre lo real (señal de entrada al sistema general) y lo proyectado (señal de salida al sistema general) por el modelo, matemáticamente se expresa: $errorcm = \sum_{i=0}^{n} (p_i - r_i)^2/n = \sum_{i=0}^{n} (e^2/n)$ de donde:

- p = valor proyectado
- r = valor real
- n = tamaño de la muestra

La utilización del CODEC implementado junto al filtro diseñado en Matlab 7.1, es una combinación muy estable para la compresión de voz. Cabe mencionar que los resultados obtenidos son excelentes comparados con los CODEC ya utilizados en la compresión de voz. En la *tabla 4.1* se muestra una comparación de la señal de entrada y la señal de salida después del filtro utilizando como referencia la energía de cada una de ellas, observando además la diferencia. Entre más pequeños los valores obtenidos en esta prueba la señal recuperada es más idéntica a la señal original. Se ha tomado la señal de referencia base para dicha prueba, la cual tiene un tamaño de 269KB.

Tabla 4.1. Valores del error cuadrático medio variando la tasa de muestreo.

Tasa de compresión	Tamaño del archivo comprimido (J2K)	Error cuadrático medio
8bpm	198KB	0.0311
7bpm	198KB	0.0311
6bpm	198KB	0.0311
5bpm	169KB	0.0316
4bpm	135KB	0.0321
3bpm	101KB	0.0324
2bpm	**68KB**	**0.0349**
1bpm	35KB	0.0450
0.5bpm	17KB	0.0477

4.4. Grado de Compresión del Sistema.

Ahora se analiza la eficiencia del código a nivel de compresión de datos, dependido del tamaño del archivo, se variará la tasa para el codificador a partir de 8bpm hasta llegar a la tasa óptima la cual es 2 bits por muestra. En las *tablas 4.2 a 4.5* se mostrarán los resultados obtenidos en la simulación desde compresión mínima hasta 2bpm.

El grado de compresión límite es automáticamente generado por el programa. Este grado de compresión muestra la eficiencia mínima de comprensión de un archivo sin tener que agregar pérdidas.

Tabla 4.2. Porcentaje de compresión utilizando una tasa de compresión con pérdidas.

Tamaño del archivo de voz	Grado de compresión límite	Tamaño del archivo comprimido (J2K)	Porcentaje de compresión
269KB	5.87bpm	198KB	26.39%
361KB	4.90bpm	222KB	38.50%
429KB	5.73bpm	308KB	28.20%
883KB	6.21bpm	686KB	22.31%
1.840KB	5.81bpm	1.336KB	27.39%
2.666KB	6.35bpm	2.116KB	20.63%

Tabla 4.3. Porcentaje de compresión utilizando una tasa de compresión de 6bpm.

Tamaño del archivo de voz	Tamaño del archivo comprimido (J2K)	Porcentaje de compresión
269KB	198KB	26.39%
361KB	222KB	38.50%
429KB	308KB	28.20%
883KB	662KB	25.03%
1.840KB	1.336KB	27.39%
2.666KB	1.999KB	25.02%

Tabla 4.4. Porcentaje de compresión utilizando una tasa de compresión de 5bpm.

Tamaño del archivo de voz	Tamaño del archivo comprimido (J2K)	Porcentaje de compresión
269KB	169KB	37.17%
361KB	222KB	38.50%
429KB	268KB	37.53%
883KB	552KB	37.49%
1.840KB	1.150KB	37.50%
2.666KB	1.667KB	37.47%

Tabla 4.5. Porcentaje de compresión utilizando una tasa de compresión de 4bpm.

Tamaño del archivo de voz	Tamaño del archivo comprimido (J2K)	Porcentaje de compresión
269KB	135KB	49.81%
361KB	182KB	49.58%
429KB	215KB	49.88%
883KB	451KB	48.92%
1.840KB	920KB	50.00%
2.666KB	1.333KB	50.00%

Debido a que el nivel de ruido incluido en las grabaciones ha sido bastante significativo, se asignó temporalmente el nivel de la tasa de compresión a 4bpm, ya que la relación entre la señal y el ruido no permitía forzar más el porcentaje de compresión. Al realizar las pruebas con grabaciones de voces hechas en estudios en donde se han utilizado dispositivos especiales para obtener una voz limpia se obtuvieron grandes mejoras. Con estos archivos se ha podido llegar a comprimir la voz con una tasa de 2bpm, lo que hace de este código una opción adicional para la compresión de voz. En la *tabla 4.6*. se resume el resultado de estas pruebas.

Tabla 4.6. Porcentaje de compresión utilizando una tasa de compresión de 2bpm.

Tamaño del archivo de voz	Tamaño del archivo comprimido (J2K)	Porcentaje de compresión
10KB	3KB	70.00%
64KB	16KB	75.00%
89KB	23KB	74.15%
124KB	31KB	75.00%
200KB	50KB	75.00%

La voz recuperada utilizando una tasa de 2bpm se le ha adicionado un ruido interno, entre ellos está el ruido blanco, para reducir el nivel de ruido se ha diseñado un ecualizador en Matlab7.1 para mejorar la calidad de la señal, para eliminar algunas componentes de alta frecuencia.

Los CODEC estandarizados para la voz proporcionan porcentajes de compresión elevados, para darse una idea de estos valores en la *tabla 4.7* se listan los CODEC más utilizados [17].

Tabla 4.7. Porcentaje de compresión comparado con 16 bit PCM

	Codificador	Decodificador	% de Compresión
u-law:	42K	40K	50.0%
ADPMC:	407K	330K	75.0%
GSM:	2.0M	950K	89.7%
LPC:	2.5M	1.0M	96.3%
CELP 4.5K:	24-52M	4.4M	96.5%
CELP 3.0K:	25-47M	4.0M	97.7%
LPC-10:	6.4M	3.5M	98.1%
CELP 2.3K:	24-45M	3.8M	98.2%
OpenLPC 1.8K:	2.9M	1.8M	98.6%
OpenLPC 1.4K:	2.9M	1.9M	98.9%

Comparando los resultados obtenidos del CODEC que se ha creado en lenguaje JAVA y los CODEC estandarizados, se puede observar que la diferencia de los porcentajes viene siendo de aproximadamente de 10%.

4.5. Pruebas de la Calidad de Voz Recuperada.

Una vez se han comparado las señales originales y recuperadas se averigua la calidad de la trama de voz recuperada frente a la trama de voz original. Para esta parte se utiliza la *Puntuación Media de Opinión* (MOS) [15] el cual proporciona una indicación numérica de la calidad percibida de los medios de comunicación después de recibida la compresión y / o transmisión de los datos respectivos, en la *tabla 4.8* se muestran la descripción de los valores de esta puntuación. El MOS se expresa como un número único en el rango de 1 a 5, donde 1 representa la más baja calidad percibida, y 5 es localidad más alta percibida.

El MOS es generado por un promedio de resultados obtenidos por una encuesta realizada y evaluada por 20 oyentes siendo una prueba subjetiva.

Tabla 4.8. Puntuación media de opinión

MOS	Calidad	Deterioro
5	Excelente	Imperceptible
4	Bueno	Perceptible, pero no molesto
3	Justo	Muy poco molesto
2	Pobre	Molesto
1	Malo	Muy molesto

Como ejemplo, la *tabla 4.9* muestra las puntuaciones medias de una opinión de los diferentes códec de aplicación [16]:

Tabla 4.9. Puntuaciones media de opinión de diferentes codecs.

CODE	Velocidad de transmisión de Datos [Kb/S]	MOS
GSM FR	12.5	3.5
G.723.1r53	5.3	3.65
G.729a	8	3.7
G.726 ADPCM	32	3.8
GSM EFR	12.2	3.8
G.723.1r63	6.3	3.9
G.723.1r63	8	3.92
iLBC	15.2	4.14
AMR	12.2	4.14
G.711 (RDSI)	64	4.3

Para analizar la efectividad del código y llevarlo a sus extremos, observemos el comportamiento de la variación de las tasas de compresión manteniendo constante el mismo archivo de voz. Se tomará el archivo *señaldevoz3.wav* con un tamaño de 883KB para hacer las mediciones respectivas.

La *tabla 4.10* presenta los resultados de las pruebas:

Tabla 4.10. Calidad de la voz variando la tasa de compresión.

Tasa de compresión	Tamaño del archivo comprimido (J2K)	MOS
8bpm	686KB	5
7bpm	686KB	5
6bpm	662KB	5
5bpm	552KB	4.5
4bpm	451KB	4
3bpm	344KB	3.5
2bpm	221KB	3.5
1bpm	113KB	2.5
0.5bpm	55KB	1.5
0.1	11KB	1
0.05	6KB	1
0.01	1KB	1

La calidad de la voz en el receptor (después de ser descomprimida) se clasifica de acuerdo al MOS encontrado, estas medidas se tomaron con relación a la señal de voz original, cuando se habla de calidad alta se hace referencia a que esta señal recuperada es muy similar a la original.

5. CONCLUSIONES Y RECOMENDACIONES

En este capítulo trata con las consideraciones del proyecto total, de su alcance y sus limitaciones, de los objetivos logrados en este proyecto y las posibles líneas de trabajo que podrían seguir proyectos futuros.

5.1. Conclusiones

Inicialmente se hace un recuento de los objetivos planteados en este proyecto. El objetivo principal de este trabajo es el desarrollo de un sistema para la compresión y descompresión de voz mediante técnicas de procesamiento digital de imágenes utilizando *wavelets*. Para esto se utilizaron las técnicas de codificación de voz y datos haciendo énfasis en el estándar JPEG2000 [9]. Respecto al estándar, se investigaron los mecanismos de funcionamiento del JPEG2000, especificados en el ANEXO B. Considerando los procesos internos del estándar, se escogió la plataforma JJ2000 [13] para desarrollar la aplicación de codificación y decodificación de tramas de voz. Utilizando este método, se analizaron los resultados con medidas matemáticas y medidas subjetivas, basadas en la puntuación de opinión media [15].

Considerando cada una de las actividades realizadas, se puede afirmar que se completaron los objetivos específicos propuestos en el anteproyecto. En lo que respecta al estado del arte, se investigó sobre los métodos de compresión de voz y datos [4-7], especificados en el capítulo 2 y en los anexos. El proceso de familiarización con el estándar JPEG2000 proporcionó las herramientas adecuadas para poder elegir una plataforma de desarrollo apropiada, como lo es JJ2000. Se diseñó un sistema de interfaz sobre el cual la aplicación comprime tramas de voz. Sobre estos resultados, se han analizado los archivos recuperados frente a los archivos originales, obteniendo unos niveles de fidelidad objetivos y subjetivos.

Al analizar los archivos de audio, se puede ver claramente como el estándar JPEG2000 cumple satisfactoriamente con los objetivos planteados. Este estándar ha sido capaz de comprimir exitosamente tramas de voz, y por su particular funcionamiento, resulta ser ideal para la compresión de voz. La transformada *wavelet* ha demostrado ser altamente eficiente en el análisis de señales de voz, entregando tasas de compresión de hasta un 75%, usando un filtro sencillo para adaptar la salida.

El desempeño del sistema de compresión del JJ2000 ha sido altamente satisfactorio, donde se demuestra que el uso de la transformada wavelet es muy adecuado en el análisis de tramas de voz. Además, gracias a su proceso de compresión con tasas adaptables, donde se puede ajustar directamente el grado de compresión deseado frente a la calidad del archivo recuperado, se ofrece el posible diseño de un codificador de tramas de voz adaptable al medio.

La plataforma de compresión de datos JJ2000 proporciona una amplia gama de herramientas bastante útiles en el análisis de los archivos de imagen, lo que permite aclarar los pasos para mejorar la transformada de vector a matriz en el módulo de adaptación. Al tener medidas directas del efecto generado al manipular la matriz de datos, se pudieron definir los métodos para proporcionar una mejor compresión y una menor pérdida de datos.

Sobre la calidad de los archivos recuperados, se puede ver en el capítulo 4 que las diferencias de los archivos recuperados son muy bajas en los grados de compresión más altos, manteniendo un umbral de tolerancia al ruido bastante apropiado [15]. Más aún, al hacer un tratamiento básico de adaptación a la señal de salida, se pueden comprimir los datos a tasas muy bajas, manteniendo un grado de la señal bastante alto.

Tras la evaluación de los resultados, se puede notar que la implementación de un sistema de codificación de datos basado en la transformada *wavelet*, de código libre y compresión flexible y adaptable al medio, como el estándar JPEG2000, es bastante adecuado como codificador de tramas de voz, y su postulación como códec en una arquitectura de voz sobre IP entregaría muchos beneficios [8]. Los beneficios más claros es que el módulo de selección de códec no tendría que ver entre varios codificadores, modificando su

mecanismo para que elija el grado de compresión deseado, adaptándose al medio en el cual se instauraría la llamada.

Lamentablemente, al hacer las investigaciones apropiadas en la definición de códec de tramas de voz, no se pudo encontrar un método efectivo para poder asignar a este sistema como codificador de tramas de voz en una arquitectura de VoIP. La arquitectura del *PBX Asterisk*, a pesar de poder elegir los codificadores a utilizar varios códec, configurado con el *sip.conf*, carece de los métodos para agregar códec externos, por lo que las pruebas de eficiencia de este sistema en una red de VoIP solo podían hacerse de manera comparativa a los otros estándares de compresión de voz.

5.2. Futuras Investigaciones

Considerando las funcionalidades del sistema implementado, se pueden ofrecer muchas líneas de trabajo que nacen a partir de este.

- Desde el punto de vista de tratamiento de datos, se podría plantear una mejora en el módulo de adaptación, modificándose para que divida las muestras deseadas en diferentes componentes, o que use archivos con profundidades de byte superiores [14].
- Se pueden postular transformadas de vector a matriz que analicen el contenido y aumenten la coherencia entre los datos, mejorando el desempeño general del sistema [18].
- Con las herramientas apropiadas es posible desarrollar un módulo en el cual se pueda adaptar el sistema de compresión de datos para simular su rendimiento en una red de voz sobre IP. Se puede postular el diseño de un selector de códec de tramas de voz que no tenga que elegir entre varios, sino que use este sistema como base y elija dinámicamente la tasa de bits que desea enviar.
- Adicionalmente se puede proponer un estándar de compresión de voz no híbrido, basándose en el funcionamiento del estándar JPEG2000, aprendiendo de sus características para entregar un sistema basado en wavelets, que utiliza un banco de

filtros con decimado máximo uniforme de reconstrucción perfecta (*Perfect-reconstruction uniformly-maximally-decimated filter bank – PR-UMDFB)* como herramienta de análisis de la trama de voz, de codificación por zonas de significancia y de tasas de compresión ajustables.

Estos son apenas ejemplos del potencial que tiene esta línea de trabajo, mostrando varias posibilidades en las que grupos de investigación podrían explorar, para verificar su viabilidad. Son muchos los campos en que se puede enfatizar este trabajo, y se espera que los resultados obtenidos animen a más grupos a indagar en esta rama de investigación.

BIBLIOGRAFIA

[1] Claude E. Shannon. "*A Mathematical Theory of Communication*". The Bell System Technical Journal, Vol. 17 pp. 379-423 623-656. October 1948.

[2] López Alejandro, Parada Andrea, Simonetti Franco. "*Introducción a la psicología de la comunicación*" Ediciones Universidad Católica de Chile. 1995.

[3] Wiio Osmo A. "*Wiio's laws - and some others*". 1978.

[4] Faundez Pablo, Fuentes Álvaro. "*Procesamiento Digital de Señales Acústicas utilizando Wavelets*". Universidad Austral de Chile, Instituto de Matemáticas. Febrero 2006.

[5] Gómez Carolina Celeste. "*Compresión de imágenes con pérdidas. Método de la Transformada Wavelet*". Trabajo Práctico Final Universidad Nacional del Litoral. Junio 2005.

[6] B.E. Usevitch. "*A Tutorial on Modern Lossy Wavelet Image Compression: Foundations of JPEG2000*". IEEE Signal Processing Magazine. Vol. 18. pp. 22-35. Septiembre 2001.

[7] Kaschel Héctor C. Francisco Watkins, Enrique San Juan U. "*Compresión de voz mediante técnicas para el procesamiento de señales y aplicación de formatos de compresión de imágenes*". Tesis Universidad de Santiago de Chile, Departamento de Tecnologías Industriales. 14 de octubre de 2005.

[8] Pascual Alberto Escudero, Berthilson Louise. "*Una guía para crear una infraestructura de voz en regiones en desarrollo*". Diciembre 2006.

[9] Adams Michael D. "*The JPEG-2000 Still Image Compression Standard*". Dept. of Electrical and Computer Engineering University of Victoria. 3 Diciembre 2005.

[10] Adams Michael D. "*JasPer project home page*". http://www.ece.uvic.ca/ ˜mdadams/jasper. 2002.

[11] Adams Michael D. and F. Kossentini, "*JasPer: A software-based JPEG-2000 codec implementation*". Of IEEE International Conference on Image Processing, Vancouver, BC, Canada, Oct. 2000.

[12] Adams Michael D. "*JasPer software reference manual*". ISO/IEC JTC 1/SC 29/WG 1 N 2415, Dec. 2002, Documentation distributed with the JasPersoftware.

[13] Adams Michael D. "*The JPEG-2000 Still Image Compression Standard*". Last Revised: Dic. 2005.

[14] ISO/IEC 15444-1: Information technology-JPEG 2000 image coding system-Part 1: Core coding system, 2000.

[15] UIT-T P.800-Methods for Subjective Determination of Transmission Quality-August 1996.

[16] Mean opinion score - http://en.wikipedia.org/wiki/Mean_Opinion_Score

[17] HawkVoice™ codec comparation- http://www.hawksoft.com/hawkvoice/codecs.shtml - 19 October 2005.

[18] ISO/IEC 15444-2: Information technology-JPEG 2000 image coding system-Part 2: Extensions, 2002.

[19] Colom Ricardo J., Gadea Rafael, Sebastiá Ángel, Martínez Marcos, Herrero Vicente, Arnau Vicente. "*Transformada Discreta Wavelet 2-D para procesamiento de video en tiempo real*". Septiembre 2001.

[20] "ISO/IEC call for contributions, JPEG 2000," ISO/IEC JTC 1/SC 29/WG 1 N505, Mar. 1997.

[21] ISO/IEC 15444-3: Information technology-JPEG 2000 image coding system-Part 3: Motion JPEG 2000, 2002.

[22] ISO/IEC 15444-4: Information technology-JPEG 2000 image coding system-Part 4: Compliance testing, 2002.

[23] ISO/IEC 15444-5: Information technology-JPEG 2000 image coding system-Part 5: Reference software, 2002.

[24] ISO/IEC 15444-6: Information technology-JPEG 2000 image coding system-Part 6: Compound image file format, 2003.

[25] ISO/IEC 15444-12: Information technology-JPEG 2000 image coding system-Part 12: ISO base media file format, 2003.

[26] A. S. Lewis and G. Knowles, "*Image compression using the 2-D wavelet transform*". IEEE Trans. on Image Processing, vol. 1, no. 2, pp. 244–250, Apr. 1992.

[27] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "*Image coding using wavelet transform*". IEEE Trans. on Image Processing, vol. 1, no. 2, pp. 205–220, Apr. 1992.

[28] D. Taubman, "*High performance scalable image compression with EBCOT*". In Proc. of IEEE International Conference on Image Processing, Kobe, Japan, 1999, vol. 3, pp. 344-348.

[29] D. Taubman, "*High performance scalable image compression with EBCOT*". IEEE Trans. on Image Processing, vol. 9, no. 7, pp. 1158–1170, July 2000.

[30] D. Taubman, E. Ordentlich, M. Weinberger, and G. Seroussi, "*Embedded block coding in JPEG 2000*". Signal Processing: Image Communication, vol. 17, no. 1, pp. 49–72, Jan. 2002.

[31] A. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Wavelet transforms that map integers to integers". Applied and Computational Harmonic Analysis, vol. 5, no. 3, pp. 332–369, July 1998.

[32] M. D. Adams, Reversible Integer-to-Integer Wavelet Transforms for Image Coding, Ph.D. thesis, Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada, Sept. 2002, Available online from http://www.ece.uvic.ca/˜mdadams.

[33] M. J. Gormish, E. L. Schwartz, A. F. Keith, M. P. Boliek, and A. Zandi, "*Lossless and nearly lossless compression of high-quality images*". In Proc. of SPIE, San Jose, CA, USA, Mar. 1997, vol. 3025, pp. 62–70.

[34] I. H. Witten, R. M. Neal, and J. G. Cleary, "*Arithmetic coding for data compression*" Communications of the ACM, vol. 30, no. 6, pp. 520–540,1987.

[35] ISO/IEC 14492-1: Lossy/lossless coding of bi-level images, 2000.

[36] H. Chao, P. Fisher, and Z. Hua, "*An approach to integer wavelet transforms for lossless for image compression*". In Proc. of International Symposium on Computational Mathematics, Guangzhou, China, Aug. 1997, pp. 19–38.

[37] M. D. Adams, "*Reversible wavelet transforms and their application to embedded image compression*". M.A.Sc. thesis, Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada, Jan. 1998, Available from http://www.ece.uvic.ca/˜mdadams.

[38] M. D. Adams and F. Kossentini, "*Reversible integer-to-integer wavelet transforms for image compression: Performance evaluation and analysis*". IEEE Trans. on Image Processing, vol. 9, no. 6, pp. 1010–1024, June 2000.

[39] W. Sweldens, "*The lifting scheme: A custom-design construction of biorthogonal wavelets*". Applied and Computational Harmonic Analysis, vol. 3, no. 2, pp. 186–200, 1996.

[40] W. Sweldens, "*The lifting scheme: A construction of second generation wavelets*". SIAM Journal of Mathematical Analysis, vol. 29, no. 2, pp. 511–546, Mar. 1998.

[41] D. Le Gall and A. Tabatabai, "*Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques*". in Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing, New York, NY, USA, Apr. 1988, vol. 2, pp. 761–764.

[42] J. M. Shapiro, "*Embedded image coding using zerotrees of wavelet coefficients*". IEEE Trans. on Signal Processing, vol. 41, no. 12, pp. 3445–3462,Dec. 1993.

[43] C. Christopoulos, J. Askelof, and M. Larsson, "*Efficient methods for encoding regions of interest in the upcoming JPEG 2000 still image coding standard*". IEEE Signal Processing Letters, vol. 7, no. 9, pp. 247–249, Sept. 2000.

[44] J. Askelof, M. Larsson Carlander, and C. Christopoulos, "*Region of interest coding in JPEG 2000*". Signal Processing: Image Communication, vol. 17, no. 1, pp. 105–111, Jan. 2002.

[45] J. S. Houchin and D. W. Singer, "*File format technology in JPEG 2000 enables flexible use of still and motion sequences*". Signal Processing: Image Communication, vol. 17, no. 1, pp. 131–144, Jan. 2002.

[46] J. H. Kasner, M. W. Marcellin, and B. R. Hunt, "*Universal trellis coded quantization*". IEEE Trans. on Image Processing, vol. 8, no. 12, pp. 1677–1687, Dec. 1999.

[47] M. W. Marcellin, M. A. Lepley, A. Bilgin, T. J. Flohr, T. T. Chinen, and J. H. Kasner, "*An overview of quanziation in JPEG 2000*". Signal Processing: Image Communication, vol. 17, no. 1, pp. 73–84, Jan. 2002.