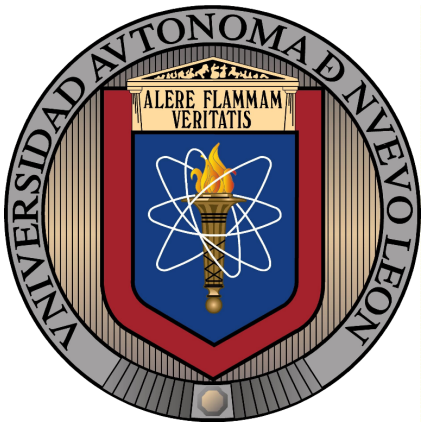
Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado



Codificación conjunta de canal inalámbrico y fuentes de información para la transmisión de imágenes

por

Yandy Ariel Yanes Baño

en opción al grado de

Maestría en Ciencias

de la Ingeniería Eléctrica

San Nicolás de los Garza, Nuevo León 21 de enero de 2021

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado

Codificación conjunta de canal inalámbrico y fuentes de información para la transmisión de imágenes

por

Yandy Ariel Yanes Baño

en opción al grado de

Maestría en Ciencias

de la Ingeniería Eléctrica

San Nicolás de los Garza, Nuevo León 21 de enero de 2021

Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica
Subdirección de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis "Codificación conjunta de canal inalámbrico y fuentes de información para la transmisión de imágenes", realizada por el alumno Yandy Ariel Yanes Baño, con número de matrícula 1988534, sea aceptada para su defensa como requisito para obtener el grado de Maestría en Ciencias en Ingeniería Eléctrica.

El Comité de Tesis

Dr. José Ramón Rodríguez Cruz
Director

Dr. Cornelio Posadas Castillo
Co-Director

Dr. Miguel Ángel Platas Garza
Revisor

Dr. Ernesto Zambrano Serrano
Revisor

Vo. Bo.

Dr. Simón Martínez Martínez
Subdirector de Estudios de Posgrado

103

San Nicolás de los Garza, Nuevo León, diciembre de 2020

Ciudad Universitaria Pedro de Alba s/n, C.P. 66455. A.P. 076 Suc. "F"
San Nicolás de los Garza, Nuevo León, México. Tels: (81) 8332 0903 /
Conm.: 8329 4020 / Fax: (81) 8332 0904

AGRADECIMIENTOS

Agradezco a Dios, quien ha sido mi fuerza y sostén en todo tiempo y sin Él nada de esto fuera posible.

A mi esposa Lianet Guevara por ser mi mejor amiga, por permitirme confiarles mis sentimientos, mis esperanzas y mi sueños, tu amor y apoyo significan para mí más de lo que nunca sabrás.

A mi madre Yaneysi Bano que siempre estuvo en todos mi logros apoyándome y enseñándome que todo en la vida tiene un propósito.

A mi difunto padre Ariel Yanes que se convirtió en el ángel de mi vida, por impulsarme a esforzarme y alcanzar mis metas siempre creyendo que lo lograría exitosamente.

A mis abuelos María Teresa y Diogenes por ser mis segundos padres, por su amor y cuidados siempre que lo necesité.

A mi asesor Dr José Ramón Rodríguez Cruz por su dedicación y disponibilidad para guiarme y aconsejarme durante todo el proyecto de tesis.

Al resto de los miembros del comité de tesis Dr Cornelio Posadas Castillo, Dr Miguel Angel Platas Garza y Dr Ernesto Zambrano Serrano por su tiempo y dedicación para corregir el manuscrito final y por sus valiosos consejos.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por el apoyo brindado mediante la beca de maestría.

Índice general

Resumen

Yandy Ariel Yanes Baño.

Candidato para el grado de Maestro en Ciencias en Ingeniería Eléctrica.

Universidad Autónoma de Nuevo León.

Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio:

Codificación conjunta de canal inalámbrico y fuentes de información para la transmisión de imágenes

Número de páginas: 111.

Firma del asesor: _____

Dr.José Ramón Rodríguez Cruz

Capítulo 1

Introducción

1.1 Motivación

Con el acelerado crecimiento de la tecnología y el despliegue de la infraestructura de comunicación de datos, ha habido una demanda creciente de servicios de comunicación multimedia digital en los últimos años [1]. Los sistemas inalámbricos se han convertido en el medio de comunicación predominante y la transmisión de imágenes a través de redes inalámbricas se ha vuelto cada vez más popular [2]. Recibiendo una gran atención la transmisión de datos en forma de imágenes en las últimas dos décadas [3]. Sin embargo, el medio de comunicación inalámbrico es ruidoso, está abierto a intrusos y tiene un ancho de banda limitado. Se requiere un nivel adicional de protección y seguridad contra errores para que la red inalámbrica sea confiable y segura [4]. Por otro lado, las fuentes multimedia suelen contener una cantidad significativa de redundancia.

Debido a estos impedimentos, lograr una transmisión eficiente y robusta de imágenes por canales inalámbricos constituye un gran desafío. Por lo que la codificación de fuente y de canal son componentes indispensables para la transmisión eficiente de los datos. La codificación de fuente elimina la redundancia presente en la información, es decir, intenta obtener representaciones digitales compactas de la salida de la fuente; mientras que la codificación de canal añade de manera controlada cierta redundancia para combatir los efectos de ruido e interferencias producidos durante la transmisión de la señal a través de canales ruidosos.

Durante las últimas décadas, se han logrado mejoras significativas en estas dos técnicas de manera separada. Sin embargo, estos dos esquemas de codificación actúan de manera contraria por lo que la idea de una codificación conjunta de fuente y canal (CCFC) ha ido ganando una creciente atención en los últimos años. La CCFC puede mejorar significativamente el desempeño de un sistema en cascada tradicional de codificador de fuente y canal [5].

1.2 Antecedentes

La investigación sobre la mejora del rendimiento de las comunicaciones a través de canales ruidosos aumenta con la creciente demanda de la transmisión rápida y eficiente de multimedia [6]. Siendo la CCFC objeto de una activa investigación por años. En la literatura se han propuesto diversos esquemas de CCFC.

En [5] se presenta un esquema de codificación conjunta de fuente y canal mediante el uso de códigos de longitud variable. Los autores utilizan la paquetización para limitar el efecto de propagación de errores y usan la redundancia residual en la salida del codificador de la fuente para proporcionar la corrección de errores a la salida codificada de la fuente con longitud variable.

En [7] examinan una forma de utilizar la capacidad de detección de errores para proporcionar corrección de errores. En este trabajo presentan un esquema concatenado en el que el papel del código interno y externo es asumido por diferentes aspectos del codificador de fuente. Para el código interno, usan un decodificador secuencial que utiliza la información de la salida del canal. La redundancia de la fuente está incrustada en la codificación de la fuente reservando espacio para un símbolo que no está en el alfabeto de la fuente para los códigos aritméticos.

En [8] se considera la decodificación iterativa de fuente y canal. Los autores analizan las dependencias entre las variables que intervienen en la codificación aritmética mediante una red Bayesiana. Esto proporciona un marco adecuado para el diseño de

un algoritmo de decodificación suave que ofrece una alta resistencia a los errores.
La decodificación conjunta de fuente y canal basada en la redundancia residual de la fuente es tratada en [9]. En este trabajo la redundancia residual se utiliza por el decodificador como protección implícita a los errores causados por el canal, dicho decodificador actúa como estimador estadístico de la secuencia transmitida, proporcionando una capacidad de corrección de errores. Con este enfoque se logran beneficios sustanciales para los canales ruidosos. Se probaron los métodos de codificación DPCM para imágenes sobre un canal simétrico binario. También se comparó un sistema compuesto por la codificación de canales basado en codificación convolucional no binaria y la decodificación conjunta de fuente y canal con un sistema más convencional basado en la decodificación de canales de codificación convolucional binaria y Viterbi estándar. Se demostró que con el sistema basado en codificación convolucional no binaria se logra un rendimiento superior.
En [10] se propone un esquema de codificación conjunta de fuente y canal de longitud fija para la transmisión de imágenes a través de canales inalámbricos. El esquema se basa en una óptima codificación conjunta de fuente y canal desarrollada para fuentes gaussianas generalizadas y una remodelación de la fuente de filtrado de paso total. Se derivan funciones generales de tasa de distorsión para tres modelos de canales: canales simétricos binarios y canales AWGN para canales sin memoria, y canales Gilbert-Elliott para canales de ráfaga.
En [11] se presenta un esquema de codificación conjunta de fuente y canal para la transmisión robusta de imágenes que, a diferencia de los esquemas de codificación en tándem estándar, donde la redundancia se introduce después de la codificación de la fuente, se introduce antes de la codificación de la fuente utilizando códigos BCH reales. Se presenta un modelo de canal conjunto correspondiente a una mezcla sin memoria de ruido Gaussiano y de Bernoulli-Gaussiano.
En [1] se proponen algoritmos de codificación conjunta de fuente y canal para la transmisión de imágenes basados en la norma de codificación de imágenes JPEG2000. Además, consideran algoritmos de codificación conjunta de fuente y canal para la transmisión de múltiples fuentes a través de un canal común que comparta un deter-

minado ancho de banda total. Los algoritmos explotan la diversidad de la distorsión de la velocidad entre múltiples fuentes para distribuir de manera óptima una velocidad binaria total dada por el canal. El autor propone un esquema de transmisión de imágenes para los canales de borrado de paquetes. El algoritmo se basa en las funcionalidades de resistencia a lo errores proporcionadas por JPEG2000 para una decodificación robusta de la fuente.

Una técnica de codificación conjunta de fuente y canal para la transmisión inalámbrica de imágenes que no se basa en códigos explícitos para la compresión ni para la corrección de errores; sino que, en su lugar, mapea directamente los valores de los píxeles de la imagen a los símbolos de entrada del canal de valor real/complejo se propone en [12]. En este trabajo las funciones de codificación y decodificación se parametrizan mediante dos redes neuronales convolucionales, que se entrenan conjuntamente, y pueden considerarse como un autoencoder con una capa no entrenable en el centro que representa el canal de comunicación ruidoso. Se consideran tanto los canales de ruido blanco Gaussiano (AWGN) invariables en el tiempo como los que se desvanecen, y se compara el rendimiento del algoritmo propuesto con los algoritmos de compresión de última generación combinados con códigos de canal. Se muestra a través de experimentos que la solución propuesta logra un rendimiento superior en regímenes de baja relación señal/ruido (SNR) y para un ancho de banda de canal limitado, sobre un canal AWGN invariante en el tiempo.

1.3 Planteamiento del problema

En los sistemas de comunicaciones inalámbricos nos enfrentamos a canales que son ruidosos y de ancho de banda limitado, por lo que se requiere tanto la protección de errores como la compresión de datos, esto se puede lograr mediante la codificación de canal y fuente. El diseño del codificador de fuente y el codificador de canal se han realizado por separado. El teorema de separación de Shannon [13] justifica teóricamente dicha separación siempre que la entropía de la fuente sea menor que la capacidad del canal y exista una fuente separable y un esquema de codificación

del canal que permita la transmisión con una probabilidad de error arbitrariamente baja, suponiendo una complejidad de codificación infinita y una demora infinita. Además, se requiere que el estado del canal sea conocido por el transmisor antes de la codificación. Sin embargo, cuando se opera en escenarios con condiciones restrictivas la separación no tiene tanto sentido, pues estas suposiciones no se cumplen para la mayoría de los sistemas prácticos. Por ejemplo para aplicaciones en tiempo real, un gran retardo no es aceptable y los canales de comunicaciones usados por el Internet y las redes inalámbricas sufren constantes cambios por lo que es difícil predecir las condiciones del canal antes y durante la transmisión. También suele ser demasiado costoso, en cuanto a ancho de banda y retardo, implementar una decodificación de canal perfecta en canales con mucho ruido, interferencias y desvanecimientos. Además se puede demostrar que el teorema de separación no es válido para todos los canales [14]. Por lo que el diseño de un esquema de codificación conjunta de fuente y canal es adecuado para sistemas prácticos.

1.4 Hipótesis

Utilizando estadísticas de archivos que contengan imágenes es posible construir un modelo de fuente que permita una eficiente compresión. Usando el modelo matemático del canal, es posible introducir algoritmos de control de errores optimizados al modelo de fuente. Implementar un esquema de codificación conjunta de fuente y canal puede reducir la distorsión y ofrecer una solución eficiente de ancho de banda.

1.5 Objetivos

1.5.1 Objetivo general

Optimizar los parámetros de transmisión de imágenes a través de canales inalámbricos y verificar su desempeño, haciendo uso del Radio Definido por Software.

1.5.2 Objetivos particulares

- Desarrollar un conjunto de instrumentos virtuales (VI) en LabVIEW que permitan transmitir y recibir una imagen.

- Evaluar el rendimiento de la transmisión de la imagen frente a técnicas de codificación de canal y compresión.

- Adaptar el sistema de transmisión a un esquema que combine la codificación de fuente y el control de errores de manera conjunta.

1.6 Metodología

- Recopilación de información e investigación previa de las tecnologías a utilizar en el desarrollo del trabajo.
- Extracción de estadísticas de las imágenes.
- Implementación de un módem digital de banda base con Modulación de Desplazamiento de Fase Binaria (BPSK) o Modulación de Desplazamiento de Fase en Cuadratura (QPSK).
- Implementación de las etapas de codificación y decodificación de canal y de fuente separadas.
- Optimización de la compresión y el control de errores mediante la implementación de un esquema con codificación conjunta de fuente y canal.
- Transmisión de imágenes cifradas y evaluación del rendimiento del sistema.

1.7 Contribuciones

Desarrollo e implementación de un sistema de comunicación digital para la transmisión de imágenes a través de un canal inalámbrico real con codificación conjunta de fuente y canal considerando la protección desigual de errores.
Evaluación y análisis del rendimiento del sistema en la transmisión de imágenes frente a las técnicas de compresión y codificación EEP y UEP.
Publicación de artículo de conferencia en el congreso IEEE Autumn Meeting on Power, Electronics and Computing (ROPEC 2020).

1.8 Organización del trabajo

En el capítulo 1 se hace una introducción al tema de investigación, exponiendo la motivación, los antecedentes, el problema que da origen a la realización del proyecto, así como los objetivos que se persiguen y la metodología a implementar.
En el capítulo 2 se realiza una revisión de la codificación de fuente para imágenes. Abarcando las diferentes técnicas de compresión de imágenes tanto sin pérdidas como con pérdidas. Se exponen, además, algunos de los estándares de compresión de imágenes existentes así como la compresión de imágenes en el reconocimiento facial.
En el capítulo 3 se presentan los diferentes esquemas de codificación de canal: los códigos de bloques lineales, códigos cíclicos, códigos convolucionales y los turbo códigos.
En el capítulo 4 se discute la codificación conjunta de fuente y canal. Brindando una clasificación de estos esquemas y ejemplos de ellos.
En el capítulo 5 se detalla el diseño y la implementación del sistema de comunicación digital inalámbrico propuesto. También se realiza una evaluación del desempeño del sistema frente a las técnicas de codificación de canal y compresión.
Por último, en el capítulo 6 se presentan las conclusiones y los trabajos futuros.

Capítulo 2

Codificación de fuente para imágenes

2.1 Codificación de fuente

Los sistemas de comunicaciones digitales tienen como objetivo principal enviar con la máxima eficiencia y fiabilidad, una secuencia de bits desde el emisor hasta el receptor. Para lograr que sea eficiente dicha transmisión se debe intentar comprimir la secuencia de bits que representa la información. Este proceso se conoce como codificación de fuente y consiste en generar una secuencia de bits comprimida con la menor redundancia posible, siempre que el receptor pueda recuperar la señal original con suficiente precisión.

La fuente es el origen de la información y puede ser discreta o continua. Una fuente discreta se presenta como una secuencia de bits por lo que puede ser transmitida de manera directa a través de un sistema de comunicación digital. Mientras que las fuentes analógicas son señales que varían de forma continua en el tiempo, requiriendo que sea transformada en forma digital para ser transmitida por el sistema de comunicación digital.

La razón por la que necesitamos la compresión de datos es que cada vez generamos y utilizamos más información que se encuentra en forma digital, en números representados por bytes de datos. El número de bytes necesarios para representar datos multimedia puede ser muy grande [15]. Por ejemplo, si hay una imagen en color de

tamaño 512x512 y la profundidad de color total es de 24 bits (8 bits rojo, 8 bits verde y 8 bits azul), la imagen sin comprimir necesita un espacio de almacenamiento de más de 786K bytes (512x512x24), que es un gran tamaño de datos para una transmisión y almacenamiento eficientes. Por lo tanto, para reducir el tamaño de la imagen y ahorrar espacio en los medios de almacenamiento y ancho de banda en los canales de comunicación, es necesaria la compresión [16].

Las imágenes suelen ocupar más espacio en un disco duro, o ancho de banda en un sistema de transmisión, que las palabras. Por lo que, en el amplio campo del procesamiento de señales, un área de muy alta actividad es la investigación de representaciones eficientes de imágenes. [17].

En el caso de las imágenes, se identifican varias formas de representación. De acuerdo a la forma utilizada, se aplican al menos tres tipos para reducir el número de datos redundantes: eliminar código redundante, eliminar píxeles redundantes y eliminar redundancia visual. El objetivo de eliminar código redundante es utilizar el menor número de símbolos para representar la imagen. En el caso de la eliminación de píxeles redundantes, la mayoría de las imágenes presentan semejanzas o correlaciones entre píxeles debido a las estructuras similares en las imágenes. De esta manera, el valor de un píxel puede emplearse para predecir el de sus vecinos. El ojo humano responde con diferente sensibilidad a la información visual que recibe. La información a la que es menos sensible se puede descartar sin afectar a la percepción de la imagen, suprimiéndose lo que se conoce como redundancia visual, y produciéndose a la vez la pérdida de ciertas características de la imagen. La eliminación de la redundancia está relacionada con la cuantificación de la información, proceso que conlleva a una pérdida de información irreversible [18].

La eficiencia de un algoritmo de codificación de fuente está relacionada con una cantidad llamada entropía, que es una función del alfabeto y la descripción probabilística de la fuente. Shannon demostró que esta entropía de la fuente es exactamente el número mínimo promedio de bits por símbolo que se requiere para la codificación de la fuente, tal que la decodificación correspondiente puede recuperar de manera única la información original a partir de la cadena de bits codificada. Para una fuente

discreta, la entropía corresponde a la longitud media de las palabras de código en el mejor algoritmo de compresión de datos sin pérdidas posible. Considerando una fuente que puede tomar valores de un alfabeto con m elementos distintos, cada uno con probabilidad $p_1, p_2, ..., p_m$. Entonces la entropía se define como [19]:

$$H = -\sum p_i log_2(p_i) \tag{2.1}$$

La entropía de fuente discreta se maximiza cuando todos los símbolos son igualmente probables, es decir cuando $p_i = \frac{1}{m}$ para $i = 1, 2, 3, ...m$.

2.2 Técnicas de compresión de imágenes

Las técnicas de compresión de imágenes se pueden dividir en dos grandes clases: sin pérdida de información y con pérdida de información. De acuerdo al tipo de imagen que se esté tratando se aplicará una u otra. Es decir, en algunas imágenes no es permisible la pérdida de información en el proceso de compresión, como por ejemplo en imágenes médicas o legales. Mientras que en otras imágenes es posible permitir cierto grado de error, aunque manteniendo la calidad de la imagen, con la finalidad de optimizar la compresión de imágenes, por ejemplo en imágenes de videoconferencias. Las técnicas de compresión no siempre se utilizan de manera separada. Por el contrario, para lograr una compresión más eficiente las técnicas de compresión con pérdida se complementan con técnicas de compresión sin pérdida [18].

2.2.1 Compresión sin pérdidas

Las técnicas de compresión sin pérdidas, como su nombre indica, no implican ninguna pérdida de información. Si los datos han sido comprimidos sin pérdidas, los datos originales pueden ser recuperados exactamente a partir de los datos comprimidos [15]. Los métodos de compresión sin pérdidas generalmente no dan una muy buena relación de compresión (típicamente en el orden de 3 a 4 veces el tamaño de

los datos originales), pero la imagen reconstruida es una réplica exacta de la imagen original y en algunas áreas de aplicación se prefiere la réplica exacta de los datos originales a una alta tasa de compresión. Por ejemplo, en aplicaciones donde las imágenes se utilizan para la extracción de alguna información específica, tales como: la obtención de imágenes médicas, el sistema de archivo de imágenes, el análisis exacto de imágenes, la teleobservación, el patrimonio cultural, el sector científico e industrial, etc [16]. La compresión de datos sin pérdidas probablemente explota la redundancia estadística para expresar los datos con mayor precisión sin pérdida de información [20].

La técnica de compresión sin pérdidas puede clasificarse a grandes rasgos en dos clases [21]:

- Codificación basada en la entropía: En este proceso de compresión el algoritmo cuenta primero la frecuencia de aparición de cada valor numérico de tono en la imagen. Luego, la técnica de compresión reemplaza los píxeles con el píxel generado por el algoritmo. Estos píxeles generados se fijan para un determinado píxel de la imagen original; y no depende del contenido de la imagen. La longitud de los píxeles generados es variable y varía en función de la frecuencia del píxel determinado de la imagen original [21].

- Codificación basada en el diccionario: Este proceso de codificación también se conoce como codificación de sustitución. En este proceso el codificador mantiene una estructura de datos conocida como diccionario. Esto es básicamente una colección de cadenas. El codificador hace coincidir las subcadenas elegidas del píxel original y las encuentra en el diccionario; si se encuentra una coincidencia exitosa, entonces los píxeles son reemplazados por una referencia al diccionario en el archivo codificado [21].

Existen diversas técnicas de compresión de imágenes sin pérdidas de información, entre estas se destacan algunas como:

- Codificación RLE (Run Length Encoding)

- Codificación de Huffman
- Codificación aritmética
- Lempel-Ziv-Welch

Codificación RLE

El método de compresión RLE es una de las formas más simples de compresión de datos. El principio de la RLE es explotar los valores que se repiten en una fuente. El algoritmo cuenta la cantidad de repetición consecutiva de un símbolo y utiliza ese valor para representar la secuencia. En la RLE, la cadena de caracteres que se repiten se almacenan como un valor y un recuento de datos únicos, en lugar de como la secuencia original. Esto es más útil en los datos cuya secuencia de caracteres se repiten: por ejemplo, imágenes gráficas simples como íconos, dibujos lineales y animaciones. No es útil con archivos que no tienen muchas repeticiones, ya que podría aumentar grandemente el tamaño del archivo. Es un método que permite la compresión de datos para información en la que los píxeles se repiten constantemente. Se basa en el hecho de que el píxel repetido puede ser sustituido por un número que indica cuántas veces se repite el píxel y por el propio píxel [20].

El algoritmo consiste en codificar un primer elemento al dar el número de veces que se repite un valor y luego el valor que se repite. Por ejemplo, la cadena "YYYYYYAAAAAAAAAAAAAAAAAA", cuando se comprime por medio de este algoritmo da como resultado "6Y19A", con una ganancia de compresión del 80 %. Sin embargo para la secuencia "CADENA", donde hay poca repetición de caracteres, al aplicar el algoritmo se obtiene como resultado "1C1A1D1E1N1A", que, como se puede observar, se logra una ganancia de compresión negativa, pues la secuencia que se obtiene es de mayor tamaño.

Codificación de Huffman

Para el esquema de compresión, en el presente trabajo se emplea la codificación de Huffman como codificación entrópica. La codificación de Huffman fue propuesta por el Dr. David A. Huffman en 1952. Los códigos generados con esta técnica o procedimiento se llaman códigos Huffman. Estos son códigos de prefijo y son óptimos para un modelo dado. El procedimiento Huffman se basa en dos observaciones sobre los códigos de prefijos óptimos [15]:

- En un código óptimo, los símbolos que se producen con mayor frecuencia (tienen una mayor probabilidad de producirse) tendrán palabras clave más cortas que los símbolos que se producen con menor frecuencia.
- En un código óptimo, los dos símbolos que aparecen con menor frecuencia tendrán la misma longitud.

El procedimiento de Huffman añade un requisito simple a estas dos observaciones. Este requisito consiste en que las palabras clave correspondientes a los dos símbolos de menor probabilidad difieran sólo en el último bit [15].
La idea de Huffman es sustituir los códigos de longitud fija por códigos de longitud variable, asignando palabras de código más cortas a los símbolos que aparecen con más frecuencia y disminuyendo así la longitud total de los datos. Cuando se utilizan palabras de código de longitud variable, es conveniente crear un código de prefijo, evitando la necesidad de un separador para determinar los límites de las palabras de código [20].
El algoritmo consiste en la creación de un árbol binario que tiene cada uno de los símbolos por hoja, y construido de tal forma que siguiéndolo desde la raíz a cada una de sus hojas se obtiene el código Huffman asociado a él. El proceso de construcción del árbol consiste en ordenar los caracteres únicos por frecuencia relativa, de menor a mayor y de izquierda a derecha. Ponemos cada uno de estos y sus frecuencias relativas en nodos conectados por ramas. Posteriormente se forma un nodo intermedio que

agrupa a los dos nodos hoja que tienen menor peso (frecuencia de aparición). El nuevo nodo intermedio tendrá como nodos hijos a éstos dos nodos hoja y su campo peso será igual a la suma de los pesos de los nodos hijos. Los dos nodos hijos se eliminan de la lista de nodos, sustituyéndolos por el nuevo nodo intermedio. El proceso se repite hasta que sólo quede un nodo en la lista. Éste último nodo se convierte en el nodo raíz del árbol de Huffman [22]. En resumen, el algoritmo para construir el árbol de Huffman es:

- A cada símbolo se crea un nodo hoja, asociando un peso según su frecuencia de aparición e insertarlo en la lista ordenada ascendentemente.
- Mientras haya más de un nodo en la lista:

  Eliminar los dos nodos con menor probabilidad de la lista.

  Crear un nuevo nodo interno que enlace a los nodos anteriores, asignándole como peso la suma de los pesos de los nodos hijos.

  Insertar el nuevo nodo en la lista, (en el lugar que le corresponda según el peso).
- El nodo que quede es el nodo raíz del árbol.

Después de construir el árbol de Huffman, el algoritmo crea un código de prefijo para cada símbolo del alfabeto simplemente recorriendo el árbol binario desde la raíz hasta el nodo que corresponde al símbolo. Asigna un 0 a una rama izquierda y un 1 a una rama derecha. Junto con la salida comprimida, se debe almacenar el árbol de Huffman con los códigos Huffman para los símbolos o sólo las frecuencias de losvalores de intensidades de color que se utilizan para crear el árbol de Huffman. Esta información es necesaria durante el proceso de decodificación y se coloca en la cabecera del archivo comprimido [20].

Codificación aritmética

Es más eficiente generar palabras de códigos para grupos o secuencias de símbolos que generar una palabra de código separada para cada símbolo en una secuencia. Sin embargo, este enfoque se vuelve impracticable cuando intentamos obtener códigos Huffman para largas secuencias de símbolos. Para encontrar la palabra de código de Huffman para una determinada secuencia de longitud m, necesitamos palabras clave para todas las posibles secuencias de longitud m. Por lo que se necesita una forma de asignar palabras clave a secuencias particulares sin tener que generar códigos para todas las secuencias de esa longitud. La técnica de codificación aritmética cumple con este requisito [15].

La codificación aritmética es una técnica poderosa para codificar estáticamente sin pérdidas. En la codificación aritmética, se genera un identificador o etiqueta única para la secuencia a codificar. Esta etiqueta corresponde a una fracción binaria, que se convierte en el código binario de la secuencia. En la práctica, la generación de la etiqueta y el código binario son el mismo proceso. Sin embargo, el enfoque de codificación aritmética es más fácil de entender si dividimos conceptualmente el enfoque en dos fases. En la primera fase, se genera un identificador o etiqueta única para una determinada secuencia de símbolos. A esta etiqueta se le da entonces un código binario único. Se puede generar un código aritmético único para una secuencia de longitud m sin necesidad de generar palabras clave para todas las secuencias de longitud m [15].

Es decir, en lugar de codificar cada símbolo individualmente, se asigna una sola palabra de código aritmético a toda la secuencia de imágenes. Se define una palabra de código del intervalo 0 a 1. El resultado de un proceso de codificación aritmética es un solo número menor que 1 y mayor o igual a 0. Este solo número puede ser decodificado de manera única para crear el flujo exacto de símbolos que se utilizó en su construcción. Para construir el número de salida, los símbolos se definen como un conjunto de probabilidades [20].

El procedimiento para realizar la codificación aritmética es el siguiente [18]:

- Tomar un alfabeto de n símbolos, cada símbolo tiene una frecuencia de aparición asociada o probabilidad de aparición del símbolo, la probabilidad en la secuencia de símbolos no necesariamente se presenta en orden.

- Calcular la probabilidad acumulativa para cada símbolo de una secuencia, probabilidad que se torna para cada símbolo procesado en un valor cada vez más pequeño.

- Asignar a cada símbolo su rango, que tienen como límite superior su probabilidad acumulativa y como límite inferior la probabilidad del símbolo anterior de la secuencia o cero sí es el primer símbolo.

- Traducir el resultado a código binario. Los símbolos con mayor probabilidad utilizan pocos bits, por ejemplo, 0.875 se traduce a 1110, mientras que 0.25 a 01.

Codificación Lempel-Ziv-Welch

Lempel-Ziv-Welch (LZW) es un algoritmo universal de compresión de datos sin pérdidas creado por Abraham Lempel, Jacob Ziv y Terry Welch. Fue publicado por Welch en 1984 como una implementación mejorada del algoritmo LZ78 publicado por Lempel y Ziv en 1978. El algoritmo es simple de implementar y tiene el potencial de un muy alto rendimiento en implementaciones de hardware. Es una técnica de compresión basada en un diccionario que permite mapear una longitud variable de la secuencia de imágenes a una longitud fija de código.
El algoritmo LZW registra el patrón en el diccionario. Las primeras 255 entradas contienen el valor de ASCII; por lo tanto, la asignación real del índice a la cadena comienza a partir del índice 256. El principio de funcionamiento del algoritmo LZW utiliza las múltiples ocurrencias de la secuencia de bits para una imagen dada que necesita ser codificada. El algoritmo LZW construye un diccionario sustituyendo las múltiples ocurrencias del patrón por un código de índice. Como es una técnica adaptativa, no es necesario transmitir el diccionario. En el lado del receptor, el

diccionario será reconstruido durante el proceso de decodificación [20].

2.2.2 Compresión con pérdidas

En la compresión con pérdida, como su propio nombre afirma, sí hay pérdida de datos de alguna manera. La imagen descomprimida no es la misma que la imagen original. La compresión con pérdida tiene una tasa de compresión por encima de las técnicas sin pérdida con alguna pérdida de datos. Para comprobar la calidad de la imagen, se comprueba la variación de color de los píxeles en los valores de color. La variación es tan pequeña que el ojo humano no puede distinguirla. El ejemplo más común de compresión con pérdida es el JPEG y la codificación Wavelet. La compresión con pérdida se utiliza más comúnmente para comprimir datos multimedia como audio, vídeo e imágenes fijas, especialmente en aplicaciones como la transmisión de medios de comunicación [23].

En esta técnica de compresión, la imagen o secuencia reconstruida es más o menos diferente de la imagen original. Se emplean principalmente cuando las imágenes tienen información redundante susceptible de ser eliminada o reducida, por ejemplo, el color del cielo en una foto suele ser uniforme y azul. En éstas técnicas, a veces también resulta interesante codificar el nivel de brillo de una muestra (luminancia o componente Y) y las diferencias de color (crominancias azul, roja y verde, o componentes Cb, Cr, Cg) [18].

La reducción se hace utilizando técnicas de codificación basada en la fuente, las cuales codifican los datos basándose en las características y propiedades de sus imágenes, permiten tasas de compresión alta y generalmente son para propósitos específicos. Entre estas técnicas se encuentran las siguientes:

- Cuantificación vectorial
- Codificación fractal
- Códificación por transformación

Cuantificación vectorial

La cuantificación es uno de los métodos de compresión con pérdidas más sencillos, aunque no por ello deja de proporcionar resultados aceptables. La cuantificación vectorial es una generalización de la cuantificación escalar . En la cuantificación escalar se representa cada valor mediante un índice a una tabla fija formada por un subconjunto de valores representativos denominado libro de códigos. Por ejemplo, si tenemos una secuencia de datos cada uno de ellos de 16 bits y consideramos solo los 8 bits más significativos de cada elemento, obtendremos una aproximación a los datos originales al perder precisión. En este caso la tabla de códigos estaría formada por todos los números de 16 bits divisibles por 256. El rango del intervalo que separa los dos valores más próximos que se pueden representar mediante esta codificación se denomina cuanto. En la cuantificación vectorial el índice referencia a un libro de códigos formado no por valores individuales, sino por vectores. Un ejemplo típico es una imagen en color en la que cada pixel viene representado por un triplete con los valores RGB. En la mayor parte de las imágenes tales tripletes no cubren todo el espacio RGB, sino que tienden a concentrarse en determinadas zonas de él. Por ejemplo, una imagen de un bosque tendrá normalmente una gran cantidad de verdes. Podemos, entonces, seleccionar un subconjunto relativamente pequeño de colores representativos y aproximar cada pixel representándolo mediante un índice al color más cercano del libro de códigos [24].

Codificación fractal

La aplicación de técnicas fractales para la compresión de imágenes digitales fue introducida por Michael Barnsley y Arnaud Jacquin en 1988. Jacquin propuso considerar las imágenes como una colección de transformaciones afines de pequeños dominios de imagen; mientras que Barnsley sugiere que las imágenes sean almacenadas como una colección de transformadas, cuyo número determina la tasa de compresión. Un fractal es un objeto semigeométrico cuya estructura básica, fragmen-

tada o irregular, se repite a diferentes escalas, por ejemplo, las nubes, las montañas, el sistema circulatorio, las líneas costeras o los copos de nieve son fractales naturales, muchas imágenes son como estos objetos [18]. La codificación fractal se basa en comprimir imágenes utilizando el hecho de que muchas imágenes presentan autosemejanza. Una de las diferencias fundamentales de este tipo de técnicas es que el código de compresión no guarda píxeles, por lo que es libre de escalas. De esta forma se puede descomprimir a cualquier escala sin tener problemas de resolución [25].
La codificación fractal introduce la idea de la descomposición de una imagen en segmentos mediante el uso de métodos estándar de procesamiento de imágenes como la separación de colores, la detección de bordes y el análisis de texturas. Cada segmento se almacena en una biblioteca de fractales [26].
Existen varias técnicas de compresión de imágenes fractales, una de ellas consiste en encontrar un sistema de funciones iteradas (IFS) que genera un conjunto de transformaciones que lleva la figura completa en cada una de sus partes autosemejantes. La información sobre la imagen quedará codificada en el IFS, donde la aplicación reiterada de las transformaciones permiten obtener una imagen bastante cercana a la original [18].

Codificación por transformación

La idea base en esta técnica es utilizar una transformada para hacer corresponder la imagen con un conjunto de coeficientes de la transformada [18]. Los algoritmos típicos de compresión de imágenes constan de tres etapas: transformada, cuantificación y codificación entrópica, dicho esquema será implementado en el presente trabajo.
La primera etapa consiste en transformar los datos linealmente mediante una transformada diseñada principalmente con el objetivo de producir coeficientes incorrelados y reducir así la redundancia entre píxeles vecinos de la imagen de entrada, permitiendo de esta forma la compresión de los datos en posteriores fases del proceso de codificación. Esta operación generalmente es reversible [27]. En este proyecto se em-

plea la transformada Wavelet.

La segunda etapa consiste en una cuantificación escalar independiente de cada coeficiente reduciendo así la precisión de los datos transformados. El escalón de cuantificación que se aplica a cada coeficiente, varía en función de la sensibilidad del sistema de visión humano al efecto subjetivo del error de cuantificación de los mismos. Esta operación es irreversible, y por esta razón no es posible recuperar de forma exacta la imagen de entrada original [27].

Por último, la tercera etapa consiste en la codificación entrópica de los coeficientes cuantificados formando una secuencia de bits. Normalmente, antes de aplicar la codificación entrópica, se aplica un reordenamiento de los coeficientes cuantificados con el objetivo de agrupar los datos con valores significativos y obtener una codificación entrópica más eficiente. En la mayoría de los casos el codificador entrópico asigna códigos de longitud variable a los coeficientes cuantificados e indexados, de forma que los valores que ocurren con más frecuencia son representados con palabras de código más pequeñas. La operación, por supuesto, es reversible [27]. Como ya se mencionó anteriormente, el algoritmo de compresión que se implementa en este proyecto utiliza la codificación de Huffman.

2.3 Transformadas utilizadas en la compresión de imágenes

Se han propuesto muchas transformadas para la compresión de imágenes, pero las más populares se pueden agrupar en dos categorías según la forma de aplicarse a la imagen: transformadas de bloque y transformadas de imagen. Las transformadas de bloque no se aplican a la imagen completa, sino que la imagen se divide en subimágenes denominadas bloques, y se transforma individualmente cada una de estas subimágenes. Algunos ejemplos de transformadas de bloque típicamente usadas en codificación de imágenes son, la Transformada de Walsh-Hadamard (WHT), la Transformada Discreta de Coseno (DCT) y la Transformada de Karhunen-Loéve

(KLT). Sin embargo, las transformadas de imagen operan con la imagen entera. La más popular es la Transformada Wavelet Discreta (DWT). Está demostrado que las transformadas de imagen, como la DWT, obtienen mejores resultados que las transformadas de bloque, pero tienden a necesitar mayores requerimientos de memoria ya que la imagen completa es procesada como una unidad [27].

2.3.1 Transformada discreta de coseno

La transformada discreta de coseno representa un secuencia finita de puntos en forma de suma de cosenos. La transformada de coseno es una transformada relacionada con Fourier, concretamente la DCT está relacionada con la DFT que es la transformada discreta de Fourier con la diferencia que la Transformada del Coseno solamente continene números reales y la Transformada de Fourier contiene reales y complejos. Esto se debe al uso de únicamente de cosenos en la DCT.
La DCT unidimensional está dada por la siguiente expresión:

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) cos\left(\frac{(2x+1)u\pi}{2N}\right), \ u = 0, 1, ..., N-1 \tag{2.2}$$

y la transformada inversa se define como:

$$f(x) = \sum_{u=0}^{N-1} \alpha(u) C(u) cos\left(\frac{(2x+1)u\pi}{2N}\right), \ x = 0, 1, ..., N-1 \tag{2.3}$$

donde:

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{, para } u = 0 \\ \sqrt{\frac{2}{N}} & \text{, para } u = 1, 2, ..., N-1 \end{cases} \tag{2.4}$$

Existe una versión extendida del caso unidimensional: la DCT bidimensional, la cual se define por:

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) cos\left(\frac{(2x+1)u\pi}{2N}\right) cos\left(\frac{(2y+1)u\pi}{2N}\right) \tag{2.5}$$

y la transformada inversa bidimensional viene dada por:

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\alpha(v) C(u,v) cos\left(\frac{(2x+1)u\pi}{2N}\right) cos\left(\frac{(2y+1)u\pi}{2N}\right) \tag{2.6}$$

El proceso de compresión de imagen utilizando DCT se realiza de la siguiente manera [28]:

- La imagen primero se divide en bloques de 8x8 píxeles.
- La DCT se aplica a cada bloque, de izquierda a derecha y de arriba a abajo.
- Cada bloque se comprime usando la tabla de cuantificación.
- El conjunto de bloques comprimidos que componen la imagen se almacena en una cantidad reducida de espacio.

La reconstrucción de la imagen se realiza a través de descompresión que es un proceso que utiliza la transformada discreta del coseno inversa (IDCT).
En las Fig. 2.1 y Fig. 2.2 se muestran las funciones bases de la transformada discreta de coseno para 4x4 y 8x8 respectivamente .

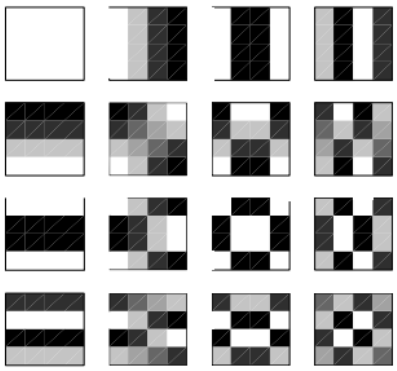


Figura 2.1: Funciones bases de la transformada discreta de coseno para 4x4 [29].

2.3.2 Transformada discreta Wavelet

La transformada Wavelet es la descomposición de $f(x)$ en una base de funciones formada por la traslación y dilatación de una misma función. Las funciones bases de la Transformada Wavelet, se conocen como wavelets, estas son generadas a partir de una función wavelet básica. Con ayuda de estas funciones, será posible

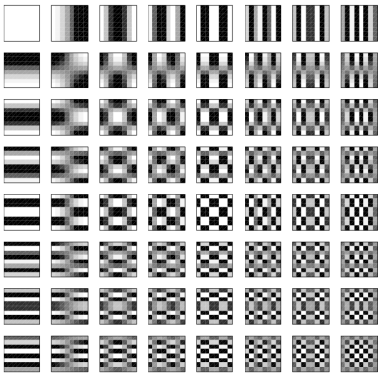


Figura 2.2: Funciones bases de la transformada discreta de coseno para 8x8 [29].

reconstruir la señal original a través de la Transformada de la Wavelet Inversa. Gracias al estudio de la Transformada Wavelet, se han logrado avances en el campo de procesamiento digital de imágenes, así como en reconocimiento de patrones y otras áreas de importancia [30]. La transformada wavelet se define por:

$$W_f(s,\tau) = \int f(t)\Psi_{s,\tau}(t)dt, \tag{2.7}$$

donde la wavelet madre o base está dada por:

$$\Psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\Psi(\frac{t-\tau}{s}). \tag{2.8}$$

Siendo s el factor de escala y τ el factor de traslación.

Para aplicar la transformada wavelet a una serie de datos numéricos, se hace necesario implementar una transformada discreta. La idea fue desarrollada por Mallat en 1989. Mallat diseñó un algoritmo basado en un banco de filtros que permite obtener una transformada wavelet, de forma poco costosa, a partir de los datos de interés [31].

En la transformada wavelet unidimensional la señal se pasa a través de dos filtros, uno paso-bajo y uno paso-alto, y las dos señales que se obtienen se diezman por un factor de 2, constituyendo la transformada de nivel uno. Los siguientes niveles se consiguen repitiendo el proceso de filtrado y de diezmado sólo en la señal filtrada paso-bajo. El proceso normalmente se lleva a cabo para un número finito de niveles, y los coeficientes resultantes se llaman coeficientes wavelet. Este enfoque puede ex-

tenderse para aplicarse a una señal bidimensional, usando flitros wavelet separables. Con filtros separables las transformadas bidimensionales pueden calcularse primero aplicando una transformada unidimensional a las filas, y después repitiéndolo en todas las columnas, tratando filas y columnas como señales unidimensionales. Cada fila de una imagen 2D se filtra con un filtro pasa-bajo y uno pasa-alto (Lx y Hx) y la salida de cada filtro se submuestrea en un factor de dos para producir las imágenes intermedias L y H. L es la imagen original pasa-bajo filtrada y muestreada hacia abajo en la dirección x y H es la imagen original pasa-alto filtrada y muestreada hacia abajo en la dirección x. A continuación, cada columna de estas nuevas imágenes se filtra con filtros de paso bajo y paso alto (Ly y Hy) y se muestrea hacia abajo en un factor de dos para producir cuatro sub-imágenes (LL, LH, HL y HH). Estas cuatro imágenes de "sub-banda"pueden combinarse para crear una imagen de salida con el mismo número de muestras que la original. La 'LL' es la imagen original, de paso bajo filtrada en direcciones horizontales y verticales y submuestra por un factor de 2. La 'HL' es de paso alto filtrada en la dirección vertical y contiene frecuencias residuales verticales, la 'LH' es de paso alto filtrada en la dirección horizontal y contiene frecuencias residuales horizontales y la 'HH' es de paso alto filtrada en ambas direcciones, horizontal y vertical. Entre ellas, las cuatro imágenes de sub-banda contienen toda la información presente en la imagen original, pero la naturaleza escasa de las sub-bandas LH, HL y HH las hace susceptibles de ser comprimidas [29].
En la Fig. 2.3 se muestra el diagrama en bloque del proceso de descomposición para la transformada wavelet bidimensional.

2.3.3 Transformada de Karhunen-Loéve

La transformada Karhunen-Loève (KLT) se basa en el análisis por componentes principales de Hotelling (PCA). La KLT es una transformada dependiente de los datos, es decir, que su núcleo (o matriz) de transformación dependerá de los datos que se quieran transformar, al contrario que la transformada DCT, que tiene un núcleo de transformación fijo. Esto hace que la KLT sea la transformada más

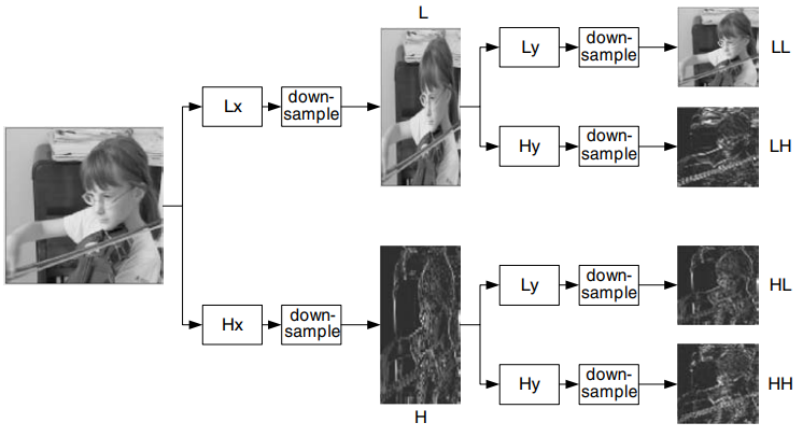


Figura 2.3: Descomposición de la transformada wavelet bidimensional [29].

apropiada para utilizar en codificación adaptativa, ya que según varían las propiedades estadísticas de los datos, la KLT se adapta a estas variaciones, mientras que la DCT no. Por otro lado, esto supone un inconveniente, y es que el decodificador necesita, para aplicar la transformada inversa, la misma matriz de transformación que utilizó el codificador para transformar los datos, por lo que es absolutamente necesario hacerle llegar dicha matriz, aumentando de esta forma la cantidad de bits enviados al decodificador y por consiguiente, disminuyendo la compresión total de los datos. En compresión de imágenes, la DCT se aplican normalmente en su forma bidimensional a los bloques de las imágenes, atacando primero la correlación existente entre las columnas del bloque y seguidamente la existente entre las filas, mediante dos transformadas unidimensionales respectivamente. Por el contrario la KLT, aunque también se aplica a los bloques de la imagen, ataca la correlación de los píxeles del bloque con una única transformación unidimensional. El motivo de esto es porque el núcleo de transformación de la KLT, en general, no es separable, por lo que cuando se aplica a bloques, estos han de ser vectorizados [27].

2.4 Estándares de compresión de imágenes

Existen varios formatos de imágenes, entre ellos se destacan algunos más conocidos como: JPEG (Joint Photographic Experts Group), GIF (Graphics Interchange Format), PNG (Portable Network Graphics), TIFF (Tagged Image File Format) y BMP (Windows bitmap). A continuación detallaremos las diferentes versiones del estándar JPEG.

2.4.1 JPEG

El nombre JPEG es un acrónimo cuyo origen proviene del Joint Photographic Experts Group. Este fue un esfuerzo conjunto del CCITT y la ISO (International Standards Organization) quienes comenzaron en junio de 1987 y produjeron el primer borrador de la propuesta del JPEG en 1991. El estándar JPEG ha demostrado ser exitoso y se ha utilizado ampliamente para la compresión de imágenes, especialmente en páginas web. JPEG ha sido diseñado como un método de compresión para imágenes de tono continuo. Los principales objetivos de la compresión JPEG son los siguientes [32]:

1. Altos índices de compresión, especialmente en los casos en que la calidad de la imagen se juzga de muy buena a excelente.

2. El uso de muchos parámetros, que permiten a los usuarios conocedores experimentar y lograr el equilibrio deseado entre compresión y calidad.

3. La obtención de buenos resultados con cualquier tipo de imagen de tono continuo, independientemente de las dimensiones de la imagen, los espacios de color, las proporciones de aspecto de los píxeles u otras características de la imagen.

4. Un método de compresión sofisticado, pero no demasiado complejo, que permite la implementación de software y hardware en muchas plataformas.

5. Varios modos de funcionamiento:

   - Un modo secuencial en el que cada componente de la imagen (color) se comprime en una sola exploración de izquierda a derecha y de arriba a abajo.
   - Un modo progresivo en el que la imagen se comprime en múltiples bloques para ser vista desde los detalles más gruesos hasta los más finos.
   - Un modo sin pérdidas que es importante en los casos en que el usuario decide que no se pierdan píxeles (la contrapartida es una baja relación de compresión en comparación con los modos con pérdidas).
   - Un modo jerárquico en el que la imagen se comprime en múltiples resoluciones que permiten visualizar los bloques de menor resolución sin tener que descomprimir primero los siguientes bloques de mayor resolución.

En el modo secuencial basado en DCT, se divide primero una imagen en bloques de 8 x 8 píxeles. Los bloques son procesados de izquierda a derecha y de arriba abajo. A cada bloque se le aplica la DCT directa y los 8 x 8 coeficientes DCT son cuantificados. Finalmente, los coeficientes DCT cuantificados son codificados entrópicamente, obteniendo los datos de la imagen comprimida. En las Fig. 2.4 y Fig. 2.5 se muestran los principales pasos de procesamiento para la codificación y decodificación basada en la DCT [33].

2.4.2 JPEG 2000

El estándar internacional JPEG 2000 representa los avances en la tecnología de compresión de imágenes, en la que el sistema de codificación de imágenes está optimizado no solo por su eficiencia, sino también por su escalabilidad e interoperabilidad en entornos de red y móviles. Los mercados y aplicaciones mejor atendidos

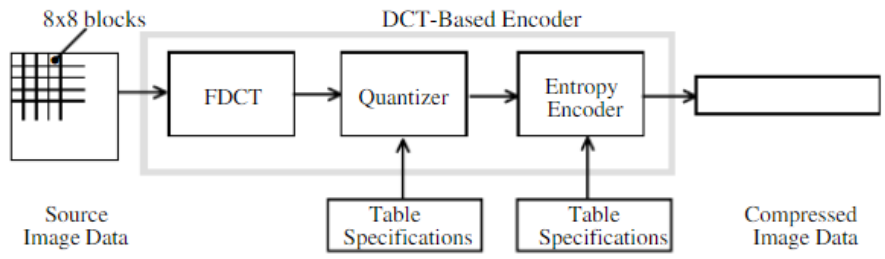


Figura 2.4: Codificador JPEG basado en DCT [33].

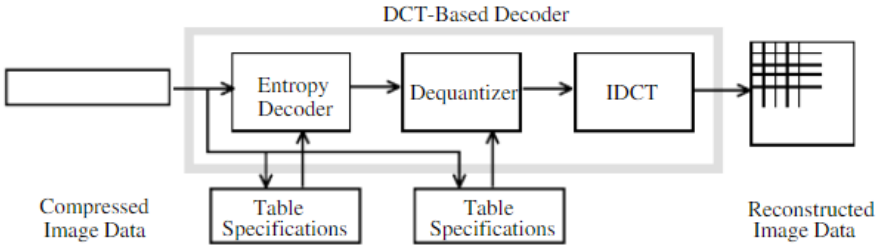


Figura 2.5: Decodificador JPEG basado en DCT [33].

por el JPEG 2000 son: Internet, el facsímil en color, la impresión, el escaneado (de consumo y de preimpresión), la fotografía digital, la teledetección, la telefonía móvil, las imágenes médicas, las bibliotecas/archivos digitales y el comercio electrónico. Algunas de las características más importante de este estándar son [34]:

- Rendimiento superior a baja velocidad de bits.
- Compresión de tono continuo y binivel.
- Compresión sin pérdidas y con pérdidas.
- Transmisión progresiva por la precisión y resolución de los píxeles.
- Codificación de la región de interés (ROI).
- Arquitectura abierta.
- Robustez a errores de bits.

En la Fig. 2.6 se muestra el diagrama en bloques del funcionamiento del estándar JPEG 2000. Primero, en el codificador, se aplica la transformada discreta en los datos de la imagen fuente. Posteriormente, los coeficientes de transformación se cuantifican y codifican entrópicamente para formar el flujo binario de código de salida.

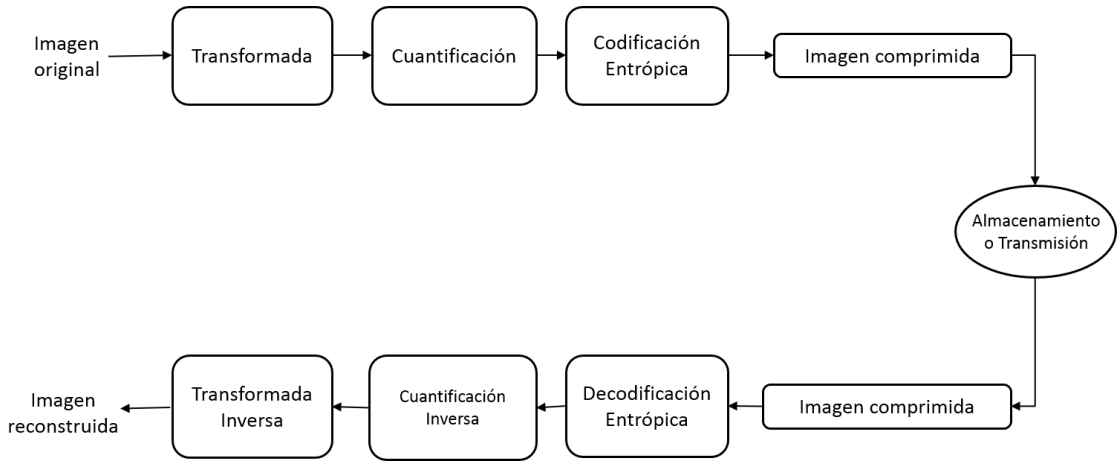


Figura 2.6: Diagrama en bloques general del proceso de codificación y decodificación en JPEG 2000.

El decodificador realiza el proceso inverso al codificador. El código binario se decodifica entrópicamente, a los coeficientes transformados obtenidos se les aplica la cuantificación inversa y luego la transformada discreta inversa, para obtener la imagen reconstruida. Aunque este diagrama de bloques general se parece al del JPEG convencional, hay diferencias radicales en todos los procesos de cada bloque del diagrama [34].

2.4.3 JPEG XR

JPEG XR es el más reciente estándar de codificación de imágenes del comité JPEG. Se dirige principalmente a la representación de imágenes fijas de tono continuo como las imágenes fotográficas y logra una alta calidad de imagen, a la par de JPEG 2000, mientras que requiere bajos recursos computacionales y capacidad

de almacenamiento. Además, responde eficazmente a las necesidades de las nuevas aplicaciones de imágenes de alto rango dinámico (HDR) al incluir soporte para una amplia gama de formatos de representación de imágenes. La arquitectura del JPEG XR refleja los nuevos requisitos específicos de las funcionalidades de alto rango dinámico. El formato de codificación JPEG tradicional utiliza una profundidad de bits de ocho para cada uno de los tres canales de color rojo, verde y azul (RGB), lo que da como resultado 256 valores por canal o 16.777.216 colores por píxeles. Sin embargo, las aplicaciones más exigentes pueden requerir una profundidad de bits de 16, proporcionando 65.536 niveles por cada canal. El comité JPEG comenzó la estandarización de la tecnología JPEG XR en julio de 2007. JPEG XR se basa en un diseño de transformación de bloques, y utiliza algunos de los mismos bloques de construcción de alto nivel que en la mayoría de los esquemas de compresión de imágenes, como la conversión de colores, la transformación espacial, la cuantificación escalar, el escaneo de coeficientes y la codificación de la entropía. El algoritmo proporciona soporte nativo para los tipos de color RGB y CMYK, convirtiendo estos formatos de color a un formato interno de luminancia-crominancia mediante el uso de una transformación de color reversible. Además, se admiten los formatos de color YUV, monocromo y cualquier otro de n componentes. Soporta múltiples profundidades de bits. Se admiten formatos de 8 y 16 bits, así como otros formatos de bits empaquetados especializados, tanto para la compresión con pérdidas como para la compresión sin pérdidas. El formato de 32 bits es soportado sólo para el modo con pérdidas. Si bien para el procesamiento interno sólo se utiliza la aritmética de los enteros, también se admite la codificación con pérdidas y sin pérdidas para los datos de imágenes de punto fijo y de punto flotante, así como para los formatos de imágenes de enteros. La transformación espacial convierte los datos de la imagen en una representación en el dominio de la frecuencia. Se utiliza una transformación jerárquica reversible del tipo Lapped Biorthogonal Transform (LBT). La transformación requiere sólo un pequeño número de operaciones de procesamiento de números enteros tanto para la codificación como para la decodificación. Es exactamente invertible en aritmética de enteros y por lo tanto soporta la representación de imágenes sin pérdidas. La

transformación se basa en dos operaciones básicas: la transformación del núcleo y el filtrado de superposición opcional. La transformación del núcleo es similar a la transformada discreta de coseno (DCT) y puede explotar la correlación espacial dentro de una región en forma de bloque. Por el contrario, el filtro de superposición está diseñado para explotar la correlación a través de los límites de los bloques, así como para mitigar los artefactos de bloqueo [35].

Para optimizar cuantificación basada en la sensibilidad del sistema visual humano y las estadísticas de coeficientes, el JPEG XR utiliza un enfoque flexible de cuantificación de coeficientes controlado por parámetros de cuantificación (QP), en el que la selección de un QP puede variar de forma flexible en diferentes regiones espaciales, bandas de frecuencia y canales de color. Para mejorar aún más la capacidad de compresión, la predicción del coeficiente de interbloqueo también se aplica a los coeficientes de transformación para eliminar la redundancia entre bloques. El escaneo adaptativo de coeficientes se utiliza para convertir el conjunto bidimensional de coeficientes de transformación dentro de un bloque en un vector unidimensional para ser codificado. Los patrones de escaneo se adaptan dinámicamente basados en las estadísticas locales de los coeficientes codificados. Finalmente, los coeficientes de transformación son codificados entrópicamente. Para ello se utiliza codificación de longitud variable (VLC) en el que se selecciona una tabla VLC entre un pequeño conjunto de tablas predefinidas fijas, y la selección de la tabla se controla de forma adaptativa basándose en las estadísticas locales. En la Fig. 2.7 se ilustran las diversas etapas del proceso de decodificación JPEG XR, el cual es básicamente una inversión paso a paso del proceso de codificación [35].

2.5 Compresión de imagen en el reconocimiento facial

La tecnología de reconocimiento de rostros ha avanzado rápidamente y se ha utilizado ampliamente en diversas aplicaciones. En el reconocimiento facial a gran

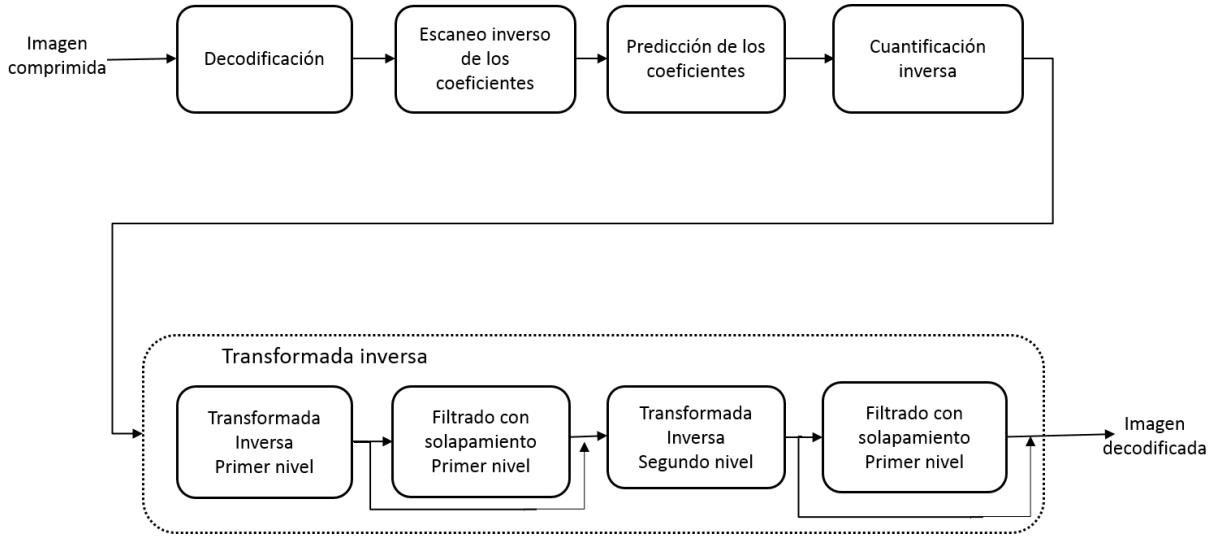


Figura 2.7: Diagrama en bloques del decodificador JPEG XR.

escala se necesita comparar la imagen obtenida con los cientos de millones de imágenes de rostros de la base de datos del sistema. Un número tan grande de imágenes de rostros no sólo ocupa una gran cantidad de recursos de almacenamiento, sino que también gasta una gran cantidad de recursos informáticos. Por lo que es importante un enfoque de compresión de imágenes eficiente que pueda adaptarse a tareas como el reconocimiento de rostros y la clasificación de imágenes. Un sistema de reconocimiento de rostros incluye dos partes: la preservación y el procesamiento de las imágenes de los rostros. Los datos de la imagen original se comprimen, codifican, decodifican y restauran mediante el códec de imagen. Luego, la imagen reconstruida es procesada por la red de reconocimiento facial para obtener los rasgos profundos. Los rasgos profundos son, a menudo, en forma de tensores de primer orden. El trabajo básico del reconocimiento facial es comparar la diferencia entre los rasgos profundos de dos imágenes de rostros. Las imágenes de las caras en muchas aplicaciones de reconocimiento facial se comprimen y reconstruyen mediante esquemas convencionales como el JPEG [36].

La limitación a una familia específica y estrecha de imágenes aumenta su redundancia espacial combinada, y esto permite que los algoritmos especialmente adaptados para la tarea de compresión de imágenes faciales, superen a los algoritmos de com-

presión de propósito general. Estos tipos de algoritmos conducen a un rendimiento muy superior al JPEG2000, varios trabajos en la bibliografía lo han demostrado [37].

2.5.1 Compresión a baja tasa de bits de imágenes faciales

En [38] se introduce un método eficaz para la compresión facial. Las imágenes con las que tratan son fotos tipo pasaporte: rostro completo, vista frontal, fondo liso, sin lentes oscuros, sin sombreros y otra ropa no estándar. Para la implementación del algoritmo adoptan un enfoque basado en los siguientes conceptos: canonización geométrica, agrupación y tratamiento Jerarquial Multiescala. Cada uno de estos conceptos se explica a continuación.

Canonización geométrica : las imágenes manipuladas se deforman geométricamente en una forma canónica, en la que los rasgos faciales se ubican en las mismas ubicaciones espaciales. Usando un procedimiento de detección de características simple, la imagen se divide para disjuntar y cubrir un conjunto de triángulos, cada uno deformado usando una deformación afín diferente.

Agrupación : el tratamiento general de las imágenes es local, dividiendo la imagen en mosaicos. Cada mosaico se codifica mediante cuantificación vectorial. Se permite una asignación de bits flexible, utilizando el árbol de cuantificación vectorial (VQ).

Tratamiento Jerarquial Multiescala : La codificación se realiza sobre una representación piramidal de la imagen, procesando la información de gruesa a fina, y en cada etapa operando sobre la imagen residual.

La codificación y decodificación en este algoritmo se realizan de la siguiente manera. El proceso de codificación comienza con la alineación: los rasgos faciales se detectan y la imagen se deforma a su forma canónica. Luego, cada mosaico se cuantifica usando cuantificación vectorial, encontrando el vector de representación más cercano (en el sentido MSE) entre los vectores VQ almacenados para este mosaico. El índice del vector VQ elegido sirve como representación del bloque. El flujo de bits que representa la imagen comprimida contiene datos sobre las coordenadas de las características

faciales y los índices de los vectores VQ. En este esquema no implementan una etapa de codificación de entropía. Para la decodificación, los pasos descritos anteriormente se realizan en orden inverso, es decir, el flujo de bits se analiza sintácticamente en las coordenadas de los puntos característicos y los índices vectoriales VQ. Como la asignación de bits y el orden de los campos en el flujo de bits son conocidos y fijos, no se requieren símbolos de formato adicionales para el análisis sintáctico. Para un mosaico dado, la imagen se recupera mediante el índice del conjunto de vectores VQ almacenado para esta ubicación. Se promedian los píxeles de bloque superpuestos y el proceso se repite para los tres canales de imagen. Finalmente, la imagen se deforma utilizando una etapa de alineación inversa.

Este método, con una deformación geométrica en una forma canónica, seguida de una codificación eficiente para cada bloque, permite un rendimiento de compresión mucho mejor en comparación con el JPEG-2000 para velocidades de bits muy bajas (en el rango de 0,01 a 0,03 bpp) [38].

2.5.2 Compresión de imágenes utilizando el Diccionario Iteración Ajustado y Alineado

Un codificador de imágenes que utiliza el Diccionario Iteración Ajustado y Alineado (ITAD) como una transformación para codificar bloques de imágenes tomados sobre una cuadrícula regular es presentado por [39]. En este algoritmo introducen un códec de imagen para clases de imagen basado en la transformación ITAD entrenada para la clase de imagen en particular. En el códec, la transformación ITAD se utiliza para codificar los bloques de imagen eliminados con la media, mientras que la media de los bloques se codifica utilizando una disposición común basada en DPCM. Los coeficientes de la transformada ITAD se codifican mediante una combinación de codificador de entropía / cuantificador uniforme simple, mientras que los índices de átomos se codifican mediante un código de longitud fija. El códec ITAD propuesto supera los estándares de compresión JPEG y JPEG2000 en evaluaciones cuantitativas y cualitativas cuando se usa en clases específicas de imágenes, como imágenes

faciales, para las cuales la transformación puede adaptarse bien.

2.5.3 Compresión de imágenes faciales usando transformada Wavelet adaptable basada en el orden de parches

Otro algoritmo de compresión es presentado en [37], el cual explota la transformada wavelet basada en árbol redundante (RTBWT). Para los procesos de codificación y decodificación se asume que se reciben conjuntos de entrenamiento y prueba que contienen imágenes faciales alineadas. Las imágenes en el conjunto de entrenamiento se promedian y se le restas la imagen media. Se calcula un diccionario a partir de las imágenes resultantes, y luego se codifica cada una de estas imágenes usando el siguiente procedimiento:

- Aplicar codificación dispersa a la imagen usando el algoritmo OMP para obtener un pequeño conjunto de valores de coeficientes y sus índices correspondientes.
- Reemplazar los índices de coeficientes por las diferencias entre los índices de coeficientes consecutivos, dividir los valores de los coeficientes en rangos bajos y altos, y aplicar una cuantificación uniforme a los valores en cada rango.
- Calcular dos tablas de Huffman diferentes, una para los valores de los coeficientes y la otra para sus índices, confiando en las estadísticas de sus ocurrencias en las escasas representaciones de todas las imágenes en el conjunto de entrenamiento.

La decodificación de la imagen se obtiene aplicando primero la decodificación de entropía, obteniendo así los valores de los coeficientes cuantificados y las diferencias de índice. Se recuperan los índices de coeficientes de sus diferencias, y los correspondientes valores de coeficientes cuantificados para construir una representación dispersa.

La imagen se reconstruye simplemente aplicando la reconstrucción RTBWT a este vector disperso y agregando la imagen de la cara media al resultado.

Capítulo 3

Codificación de canal

Claude E. Shannon demostró, en 1948, que mediante la codificación apropiada de la información, los errores que introduce un canal ruidoso pueden reducirse a cualquier nivel deseado, sacrificando el tiempo y la complejidad de tal procesamiento. [13]. Desde entonces se ha invertido mucho esfuerzo en el diseño de una codificación eficiente, así como en los métodos de decodificación para el control de errores en ambientes ruidosos. Los desarrollos recientes han contribuido a lograr confiabilidad en los sistemas digitales de alta velocidad actuales, y el uso de la codificación para el control de errores se ha convertido en una parte integral en el diseño de los sistemas de comunicaciones modernos y los sistemas computacionales digitales [40].

La codificación de canal se refiere al uso de técnicas que pretenden mejorar el rendimiento de la comunicación añadiendo redundancia en la señal transmitida para proporcionar resistencia a los efectos de las degradaciones de los canales. La idea básica de la codificación del canal es añadir cierta redundancia de manera controlada a la secuencia de bits de información. Esta redundancia puede utilizarse en el receptor para la detección o corrección de errores [19].

3.1 Clacificación de los códigos de canal

Existen diferentes códigos que se pueden emplear para la corrección de errores. Estos se pueden clasificar en dos clases principales: códigos de bloques y códigos convolucionales. La característica que distingue estas clasificaciones es la presencia

o no de memoria en los codificadores.

El codificador del canal, para generar un código de bloque, acepta una secuencia de información dividida en bloques de k-bits; a cada bloque, añade $n-k$ bits redundantes que están algebráicamente relacionados con los k bits de mensaje, produciendo así un bloque codificado global de n bits, donde $n > k$. El bloque de n-bits se denomina palabra de código, y n es su longitud. Los bits son producidos por el codificacdor a razón de $R_0 = (\frac{n}{k})R_s$, donde R_s es la tasa de bits de la fuente de información. La tasa de código se define por la relación $r = \frac{k}{n}$, donde $0 < r < 1$ [41]. Los esquemas de codificación en bloque no tienen memoria. Después de que se codifica y transmite una palabra de código, el sistema recibe un nuevo conjunto de k bits de información y los codifica utilizando la cartografía definida por el esquema de codificación. La palabra clave resultante depende únicamente de los actuales k bits de información y es independiente de todas las palabras de código transmitidas anteriormente [42].

Los códigos convolucionales se describen en términos de máquinas de estado finito. En estos códigos, en cada instancia de tiempo i, k bits de información entran en el codificador, causando n símbolos binarios generados en la salida del codificador y cambiando el estado del codificador de σ_{i-1} a σ_i. El conjunto de posibles estados es finito y está denotado por Σ . Los n símbolos binarios generados en la salida del codificador y el siguiente estado σ_i dependen de los k bits de entrada así como de σ_{i-1}. Un código convolucional se puede representar por un registro de desplazamiento de longitud Kk como se muestra en la Fig. 3.1.

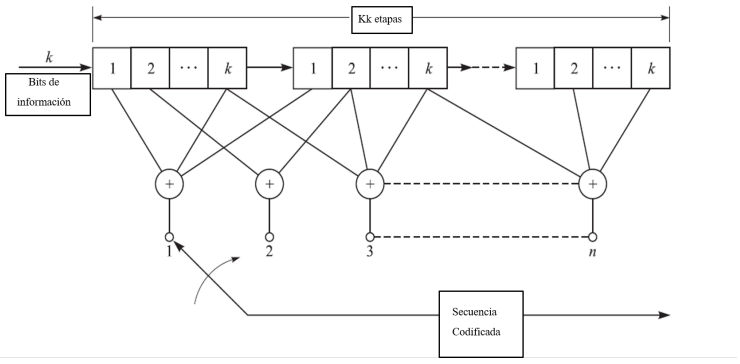


Figura 3.1: Codificador convolucional.

En cada instancia de tiempo, entran k bits en el codificador y el contenido del registro de desplazamiento se desplaza a la derecha por k elementos de memoria. El contenido de los k elementos más a la derecha del registro de desplazamiento sale del codificador. Después de que los k bits han entrado en el registro de desplazamiento, los n sumadores añaden el contenido de los elementos de memoria a los que están conectados generando así la secuencia de código de longitud n que se envía al modulador. El estado de este código convolucional viene dado por el contenido de los primeros $(K-1)k$ elementos del registro de desplazamiento [42].

De la concatenación en paralelo de dos codificadores convolucionales recursivos y sistemáticos con un entrelazador entre ellos resultan los turbo códigos. Los turbo códigos heredan muchas de las ventajas de un código convolucional, incluido el aumento de la redundancia en el bloque codificado aumentando la memoria de los codificadores. Este código es robusto ante ráfagas de errores debido al entrelazador aleatorio [40].

3.2 Códigos de bloques lineales

Un código de bloque C consiste en un conjunto de M vectores de longitud n denotado por $c_m = (c_{m1}, c_{m2}, ..., c_{mn})$, con $1 \leq m \leq M$, denominados palabras de códigos, cuyos componentes se seleccionan de un alfabeto de q elementos. Cuando el alfabeto consta de dos símbolos, 0 y 1, el código es un código binario. Cuando q es una potencia de 2, es decir, $q = 2^b$ donde b es un número entero positivo, cada símbolo tiene una representación binaria equivalente que consiste en b bits; así, un código no binario de longitud de bloque N puede ser transformado en un código binario de longitud de bloque $n = bN$. Hay 2^n posibles palabras de código en un código de bloque binario de longitud n. De estas se pueden seleccionar $M = 2^k$ palabras de código (k ¡n) para formar un código. Así, un bloque de k bits de información se convierte en una palabra de código de longitud n seleccionada del conjunto de palabras de código $M = 2^k$ [42].

Los códigos de bloques lineales son un subconjunto de los códigos de bloque. Se dice que un código es lineal si en la aritmética del módulo 2 se pueden añadir dos palabras de código para producir una tercera palabra de código. En un código de bloque lineal, la asignación del conjunto de $M = 2^k$ secuencias de información de longitud k a las correspondientes 2^k palabras de código de longitud n se representa mediante una matriz k x n G denominada matriz generadora como:

$$c_m = u_m G \qquad 1 \leq m \leq 2^k, \tag{3.1}$$

donde u_m es un vector binario de longitud k que denota la secuencia de información y c_m es la palabra de código correspondiente. Las filas de G están denotadas por g_i, $1 \leq i \leq k$, que denotan las palabrasde código correspondientes a las secuencias de información $(1, 0, ..., 0), (0, 1, 0, ..., 0), ..., (0, ..., 0, 1)$.

$$G = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_k \end{bmatrix} \tag{3.2}$$

por lo que

$$c_m = \sum_{i=1}^{k} u_{mi} g_i. \tag{3.3}$$

Si la matriz generadora G tiene la estructura siguiente:

$$G = [I_k | P], \tag{3.4}$$

donde I_k es una matriz idéntica k x K y P es una matriz k x $(n - k)$, el código de bloque lineal resultante se denomina sistemático. En los códigos sistemáticos, los primeros k componentes de la palabra de código son iguales a la secuencia de información, y los siguientes $n - k$ componentes, llamados bits de comprobación de paridad, proporcionan la redundancia para la protección contra los errores. Puede demostrarse que cualquier código de bloque lineal tiene un equivalente sistemático [42]. En la práctica encontramos algunos códigos de bloque lineales tales como: códigos de repetición, códigos de Hamming, códigos de longitud máxima, códigos de Hadamard, entre otros.

3.2.1 Códigos de repetición

Un código de repetición binaria es un código $(n, 1)$ con dos palabras de código de longitud n. Una palabra de código es todo-cero, y la otra es la palabra de código todo-uno. Este código tiene una tasa de$Rc = \frac{1}{n}$ y una distancia mínima de $d_{min} = n$. El dual de un código de repetición es un código $(n, n-1)$ que consiste en todas las secuencias binarias de longitud n con paridad uniforme.

3.2.2 Códigos de Hamming

Los códigos de Hamming son uno de los primeros códigos estudiados en la teoría de la codificación. Son códigos de bloques lineales con parámetros $n = 2^m - 1$ y $k = 2^m - m - 1$, para $m \geq 3$. Los códigos Hamming se describen mejor en términos de su matriz de comprobación de paridad H, que es una matriz $(n-k)$ x $n = m$ x $(2^m - 1)$. Las $2^m - 1$ columnas de H consisten en todos los vectores binarios posibles de longitud m, excluido el vector todo-cero. La tasa de un código Hamming viene dada por:

$$R_c = \frac{2^m - m - 1}{2^m - 1}, \tag{3.5}$$

Dado que las columnas de H incluyen todas las secuencias no nulas de longitud m, la suma de dos columnas cualesquiera es otra columna. En otras palabras, siempre existen tres columnas que son linealmente dependientes. Por lo tanto, para los códigos de Hamming, independientemente del valor de m, $d_{min} = 3$. El polinomio de distribución de peso para los códigos Hamming (n, k) se expresa como [42]:

$$A(Z) = \frac{1}{n+1}[(1+z)^n + n(1+z)^{\frac{n-1}{2}} + (1-Z)^{\frac{n+1}{2}}] \tag{3.6}$$

3.2.3 Códigos de Hadamard

Un código Hadamard se obtiene seleccionando como palabras de código las filas de una matriz Hadamard. Una matriz Hadamard M_n es una matriz n x n (n es un entero par) de 1 y 0 con la propiedad de que cualquier fila difiere de cualquier otra

fila en exactamente $\frac{n}{2}$ posiciones. Una fila de la matriz contiene todos los ceros. Las otras filas contienen cada una$\frac{n}{2}$ ceros y $\frac{n}{2}$ unos [42].

3.2.4 Códigos LDPC

Los códigos de verificación de paridad de baja densidad (LDPC) fueron inventado por Gallager en 1962, estos han sido descuidados en gran medida por la comunidad científica durante varias décadas hasta el notable éxito de los códigos Turbo que invocaron el redescubrimiento de los códigos LDPC [43]. Son códigos de bloque con matrices de comprobación de paridad que contienen solo un número muy pequeño de entradas diferentes de cero.

Un código LDPC se define como el espacio nulo de una matriz MxN de control de paridad dispersa. Puede representarse mediante un gráfico bipartito, entre los M nodos de verificación en un conjunto y los N variables en el otro conjunto. Un código LDPC se puede decodificar mediante el algoritmo de decodificación de paso de mensajes. Después de que los nodos variables se inicializan con los mensajes del canal, los mensajes de decodificación se calculan iterativamente por todos los nodos variables y nodos de control e intercambiados a través de los bordes entre los nodos vecinos. Presenta un excelente rendimiento de corrección de errores [43].

3.3 Códigos cíclicos

Los códigos cíclicos fueron introducidos por primera vez por Prange en 1957. Estos forman una subclase importante de los códigos lineales. Son muy atractivos puesto que la estructura adicional construida en esta familia de códigos hace posible su decodificación algebraica con una complejidad de cálculo reducida [42].

Los códigos cíclicos satisfacen la siguiente propiedad de desplazamiento cíclico: si $c = (c_{n-1}, c_{n-2}, ..., c_1, c_0)$, es una palabra de código de un código cíclico, entonces $(c_{n-2}, c_{n-3}, ..., c_0, c_{n-1})$, también es una palabra de código obtenida por un desplazamiento cíclico de los elementos de c. Es decir, todos los desplazamientos cíclicos de c

son palabras clave. A causa de la propiedad cíclica, los códigos poseen una considerable cantidad de estructura que puede ser explotada en las operaciones de codificación y decodificación. Se han ideado varios algoritmos eficientes de codificación y decodificación de decisiones duras para los códigos cíclicos que permiten aplicar códigos de bloque largo con un gran número de palabras de código en sistemas de comunicación prácticos.

Para desarrollar las propiedades algebraicas en los códigos cíclicos, es conveniente asociar con una palabra de código $c = (c_{n-1}, c_{n-2}, ..., c_1, c_0)$ un polinomio $c(X)$ de grado máximo $n-1$ definido como:

$$c(X) = c_{n-1}X^{n-1} + c_{n-2}X^{n-2} + ... + c_1X^2 + c_0X \tag{3.7}$$

Si $c(X)$ representa una palabra de código en un código cíclico, entonces $X^i c(X) \quad mod(X^n + 1)$ es también una palabra de código del código cíclico. Por lo tanto:

$$X^i c(X) = Q(X)(X^n + 1) + c^i(X), \tag{3.8}$$

donde el polinomio $c^i(X)$ representa una palabra de código del código cíclico que corresponde a i desplazamientos cíclicos de c a la derecha y $Q(X)$ es el cociente.
Para generar un código cíclico se utiliza el polinomio generador $g(X)$ de grado $n-k$. El polinomio generador de un código cíclico (n, k) es un factor de $X^n + 1$ y tiene la forma general:

$$g(X) = X^{n-k} + g_{n-k-1}X^{n-k-1} + ... + g_1X + 1. \tag{3.9}$$

El polinomio de mensaje se define como:

$$u(X) = u_{k-1}X^{k-1} + u_{k-2}X^{k-2} + ... + u_1X + u_0, \tag{3.10}$$

donde $(u^{k-1}, u^{k-2}, ..., u_1, u_0)$ representan los k bits de información. Las palabras de código que posean la propiedad cíclica se pueden generar multiplicando los 2^k polinomios de mensajes por el polinomio generador $g(X)$ del código cíclico (n, k) que divide a $X^n + 1$ y con grado $n - k$. Un código cíclico (n, k) puede existir solo si se puede encontrar un polinomio $g(X)$ de grado $n-k$ que divida $X^n + 1$. Por lo tanto, el problema de diseñar códigos cíclicos equivale al problema de encontrar factores de $X^n + 1$.

3.3.1 Codificadores para códigos cíclicos

Las operaciones de codificación para generar un código cíclico se pueden realizar mediante un registro de desplazamiento de retroalimentación lineal basado en el uso del polinomio generador o del polinomio de paridad.

Para generar un código cíclico sistemático se realizan los siguientes tres pasos: multiplicar el polinomio del mensaje $u(X)$ por X^{n-k} , dividir el producto por $g(X)$ y sumar el resto a $X^{n-k}u(X)$. La división del polinomio $A(X) = X^{n-k}u(X)$ de grado $n-1$ por el polinomio $g(X) = g_{n-k}X^{n-k} + g_{n-k-1}X^{n-k-1} + ... + g_1X + g_0$ se puede realizar por el registro de desplazamiento de retroalimentación de $n-k$ etapas mostrado en la Fig. 3.2:

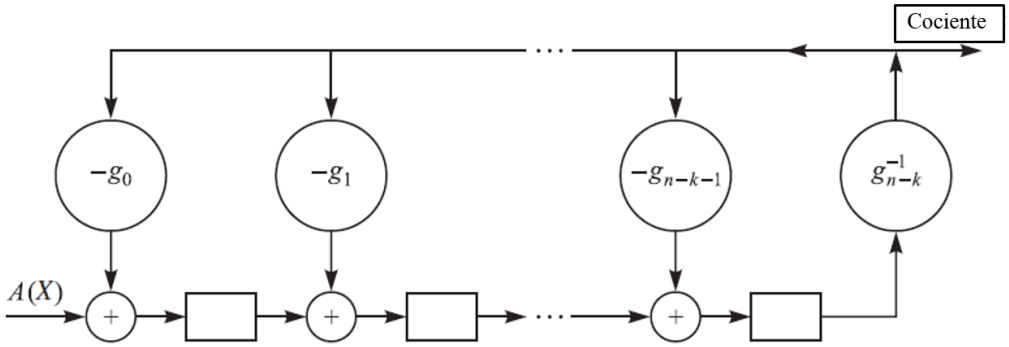


Figura 3.2: Registro de desplazamiento de retroalimentación para dividir el polinomio $A(X)$ por $g(X)$.

El registro de desplazamiento contiene todos los ceros inicialmente. Los coeficientes de $A(X)$ se registran un coeficiente a la vez, comenzando con los coeficientes de orden superior, es decir, con a_{n-1}, seguido de a_{n-2}, y así sucesivamente. Después del desplazamiento k, la primera salida no nula del cociente es $q_{k-1} = g_{n-k}a_{n-1}$. Para cada coeficiente de salida del cociente, se resta el polinomio $g(X)$ multiplicado por ese coeficiente. La resta se realiza mediante la parte de retroalimentación del registro de desplazamiento.

En a la Fig. 3.3 se muestra un codificador de código cíclico usando el polinomio generador $g(X)$. Los primeros k bits a la salida del codificador son simplemente los k bits

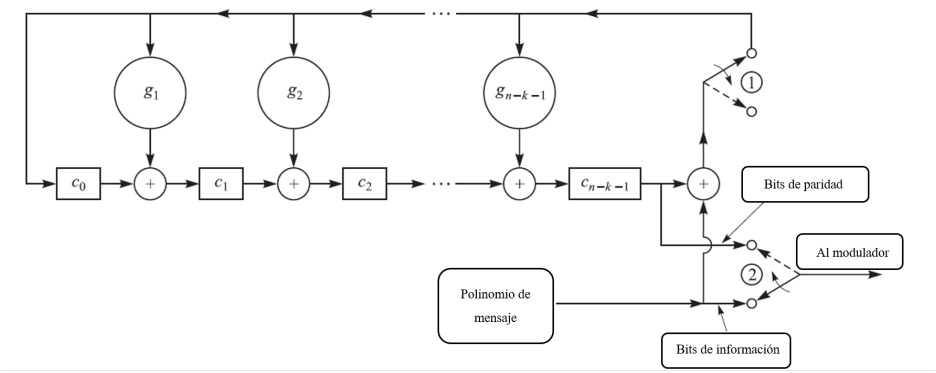


Figura 3.3: Codificador cíclico utilizando el polinomio generador.

de información. Estos k bits también se registran simultáneamente en el registro de desplazamiento, ya que el interruptor 1 está en posición cerrada. La multiplicación polinómica de X^{n-k} con $u(X)$ no se realiza explícitamente. Después de que los k bits de información se registran en el codificador, las posiciones de los dos interruptores se invierten. En este momento, el contenido del registro de desplazamiento son simplemente los bits de comprobación de paridad $n-k$, correspondientes a los coeficientes del polinomio restante. Estos $n-k$ bits se registran uno a uno y se envían al modulador.

Si en lugar del polinomio generador se utiliza, para la implementación del codificador, el polinomio de paridad $h(X) = X^k + h_{k-1}X^{k-1} + ... + h_1X + 1$ el codificador quedaría como se muestra en la Fig. 3.4.

Inicialmente, los k bits de información se desplazan al registro de desplazamiento

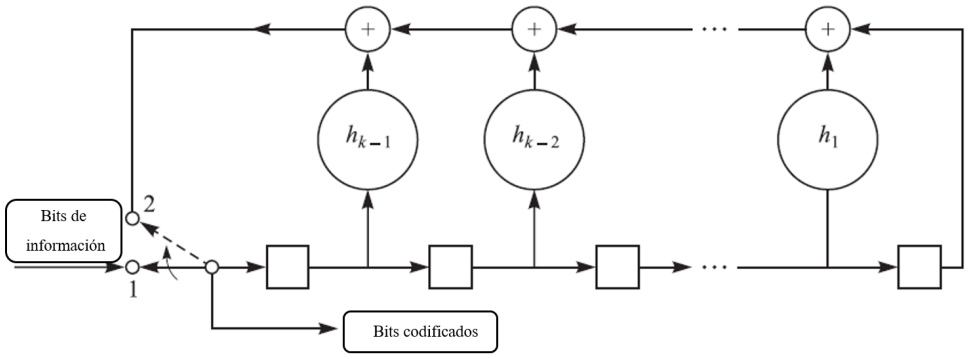


Figura 3.4: Codificador cíclico utilizando el polinomio de paridad.

y simultáneamente se incorporan al modulador. Después de que todos los k bits de información están en el registro de desplazamiento, el interruptor es puesto en la posición 2 y el registro de desplazamiento es registrado $n-k$ veces para generar los $n-k$ bits de verificación de paridad.

3.3.2 Código Bose-Chaudhuri-Hocquenghem (BCH)

Existen diferentes códigos cíclicos, entre ellos están los códigos cíclicos Hamming, códigos cíclicos de longitud máxima y códigos BCH. Estos últimos son la clase más importante de los códigos cíclicos.

Los códigos BCH comprenden una gran clase de códigos cíclicos que incluyen códigos en los alfabetos binarios y no binarios. Los códigos BCH tienen una rica estructura algebraica que hace posible su decodificación mediante el uso de eficientes algoritmos de decodificación algebraica. Los códigos BCH están entre los códigos más conocidos para longitudes de bloque bajas a moderadas.

Para cualesquiera dos enteros positivos $m \geq 3$ y $t < 2^{m-1}$ se puede diseñar un código BCH con los siguientes parámetros:

Longitud de bloque: $n = 2^m - 1$

Bits de paridad: $n - k \leq mt$

Distancia mínima: $d_{min} \geq 2t + 1$

Este código es capaz de corregir al menos t errores en un bloque de $n = 2^m - 1$ bits. Para diseñar un código BCH corrector de t errores se selecciona α, que es un elemento primitivo del campo de Galois GF(2^m). Luego el polinomio generador del código BCH, se define como el polinomio de menor grado $g(X)$ sobre GF(2) tal que $\alpha, \alpha^2, \alpha^3, ...,$ y α^{2t} son sus raíces. Sea $\phi_i(X)$ el polinomio mínimo de α, entonces $g(X)$ debe ser el Mínimo Común Múltiplo (MCM) de $\phi_{\alpha^1}(X), \phi_{\alpha^2}(X), ..., \phi_{\alpha^{2t}}(X)$, es decir:

$$g(X) = MCM[\phi_{\alpha^i}(X), \quad 1 \leq i \leq 2t] \tag{3.11}$$

Los $\phi_{\alpha^i}(X)$ para $i = 1, 2, 4, ...$ son los mismos ya que $\alpha, \alpha^2, \alpha^4, ...$ son conjugados y por lo tanto tienen el mismo polinomio mínimo. Lo mismo ocurre con $\alpha^3, \alpha^6, \alpha^{12}, ...$ Por lo tanto, el polinomio generador $g(X)$ de un código binario BCH puede reducirse a:

$$g(X) = MCM[\phi_\alpha(X), \phi_{\alpha^3}(X), \phi_{\alpha^5}(X), ..., \phi_{\alpha^{2t-1}}(X)] \tag{3.12}$$

Como el grado de cada polinomio $\phi_{\alpha^i}(X)$ es m o menor, el grado de $g(X)$ es a lo máximo mt. Por lo tanto, el número de bits de paridad del código es a lo sumo mt, es decir, $n - k \leq mt$.

Para cualquier polinomio de palabra de código $c(X)$ se tiene que:

$$c(\alpha^i) = 0 \qquad 1 \leq i \leq 2t, \tag{3.13}$$

Esta condición es necesaria y suficiente para que un polinomio de grado menor a n, sea un polinomio de palabra de código del código BCH.

3.4 Códigos convolucionales

Otra categoría importante de codificación de canales en las comunicaciones inalámbricas es la codificación convolucional. Este mecanismo de codificación es el que se usará en el presente trabajo. Un código convolucional se describe en general (k, n, m), donde k y n tienen el mismo significado que en los códigos de bloque y m es la longitud de restricción que define el número de k etapas almacenadas en el registro de desplazamiento de codificación, que se utiliza para realizar la convolución. En particular, las n-tuplas generadas por un código convolucional (k, n, m) son una función no sólo de la k entrada actual sino también de las k-tuplas de entrada anteriores $(m - 1)$. El número de elementos de memoria necesarios en el codificador convolucional es m, el cual desempeña un papel importante en el rendimiento del código y la consiguiente complejidad de la decodificación [19].

3.4.1 Codificador general para códigos convolucionales

En la Fig. 3.5 se muestra un esquema general de un codificador convolucional. Este codificador se emplea con un registro de desplazamiento de km etapas y n sumadores de módulo 2. La longitud de restricción representa el número de despla-

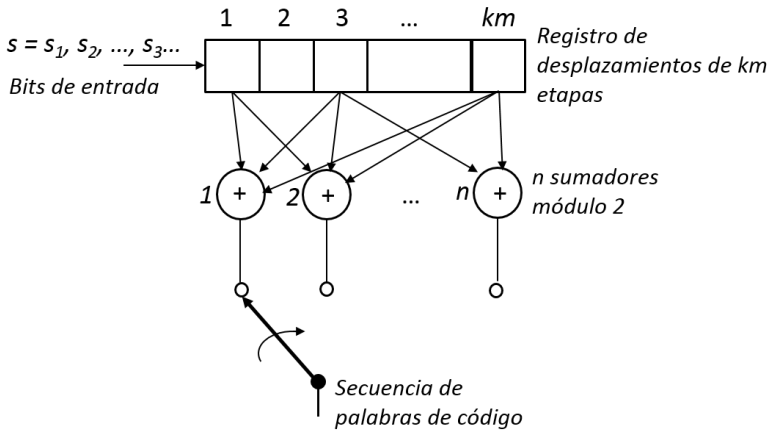


Figura 3.5: Esquema general de un codificador convolucional con longitud de restricción m y tasa de codificación $\frac{k}{n}$.

zamientos de k-bit sobre los que un solo bit de información puede influir en la salida del codificador. En cada unidad de tiempo, k bits se desplazan hacia las primeras k etapas del registro; todos los bits del registro se desplazan k etapas hacia la derecha, y las salidas de los n sumadores se muestrean secuencialmente para obtener los símbolos de código binario. Como hay n bits de código para cada grupo de entrada de k bits de mensaje, la tasa de código es $\frac{k}{n}$, donde $k < n$ [44].
Un codificador convolucional se puede describir mediante una matriz generadora o por otros métodos alternativos como: el diagrama de estado, diagrama de árbol y diagrama de espiral.

3.4.2 Representación mediante matriz generadora

La matriz generadora de un código convolucional, en gerneral, es semi-infinita, ya que la secuencia de entrada es de longitud semi-infinita. Como alternativa a la

especificación de la matriz generadora, se puede utilizar una representación funcionalmente equivalente en la que se especifica un conjunto de n vectores, un vector para cada uno de los n sumadores de módulo 2. Cada vector tiene mk dimensiones y contiene las conexiones del codificador con ese sumador de módulo 2. Un 1 en la posición i-ésima del vector indica que la etapa correspondiente en el registro de desplazamiento está conectada al sumador, y un 0 en una posición dada indica que no existe ninguna conexión entre esa etapa y el sumador de módulo 2 [42].

El codificador para un código binario $(3, 1, 3)$ se muestra en la Fig. 3.6. Inicialmente, se supone que el registro de desplazamiento está en el estado todo cero. Supongamos que el primer bit de entrada es un 1. Luego la secuencia de salida de 3 bits es 111. Supongamos que el segundo bit es un 0. Entonces la secuencia de salida será 001. Si el tercer bit es un 1, la salida será 100, y así sucesivamente. Suponiendo que las salidas de los generadores de funciones que generan cada secuencia de salida de 3 bits se enumeran como 1, 2 y 3, de arriba a abajo, y se enumeran de forma similar cada generador de funciones correspondiente. Entonces, como sólo la primera etapa está conectada al primer generador de funciones, el generador es $g_1 = [100]$, el segundo generador de funciones está conectado a las etapas 1 y 3, por lo tanto, $g_2 = [101]$ y $g_3 = [111]$. g_1,g_2 y g_3 son las respuestas impulso de la entrada del codificador a las tres salidas. Entonces si la entrada del codificador es la secuencia de información s, las tres salidas están dadas por:

$$c^{(1)} = s * g_1, \qquad c^{(2)} = s * g_2, \qquad c^{(3)} = s * g_3. \tag{3.14}$$

Para obtener la secuencia de código correspondiente c, se intercalan $c^{(1)}$, $c^{(2)}$ y $c^{(3)}$ de la siguiente manera:

$$c = (c^{(1)}, c^{(2)}, c^{(3)}, c^{(1)}, c^{(2)}, c^{(3)}, ...) \tag{3.15}$$

La operación de convolución equivale a la multiplicación en el dominio de la transformación. Sea D la transformada de s

$$s(D) = \sum_{i=0}^{\infty} s_i D^i \tag{3.16}$$

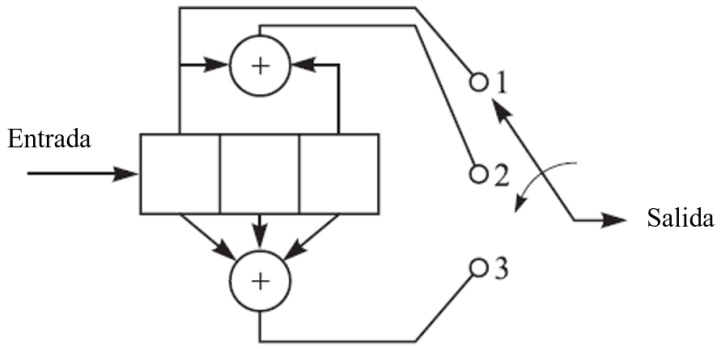


Figura 3.6: Codificador convolucional $(3, 1, 3)$.

y la función de transferencia para las tres respuestas de impulso g_1, g_2 y g_3

$$g_1(D) = 1, \qquad g_2(D) = 1 + D^2, \qquad g_3(D) = 1 + D + D^2. \tag{3.17}$$

Las transformadas de salida vienen dada por

$$c^1(D) = s(D)g_1(D), \qquad c^2(D) = s(D)g_2(D), \qquad c^3(D) = s(D)g_3(D). \tag{3.18}$$

La transformada de la salida del codificador c es

$$c(D) = c^1(D^3) + Dc^2(D^3) + D^2c^3(D^3) \tag{3.19}$$

3.4.3 Representación mediante diagrama de árbol

El diagrama de árbol para el codificador convolucional que se muestra en la Fig. 3.6 es el ilustrado en la Fig. 3.7. Suponiendo que el codificador se encuentra en el estado de todos los ceros inicialmente, el diagrama muestra que si el primer bit de entrada es un 0, la secuencia de salida es 000 y si el primer bit es un 1, la secuencia de salida es 111. Si el primer bit de entrada es un 1 y el segundo bit es un 0, el segundo conjunto de 3 bits de salida es 001. Continuando con el árbol, se puede apreciar que si el tercer bit es un 0, entonces la salida es 011, mientras que si el tercer bit es un 1, la salida es 100. Dado que una secuencia particular lleva a un nodo concreto del árbol, la regla de ramificación es seguir la rama superior si el siguiente bit de

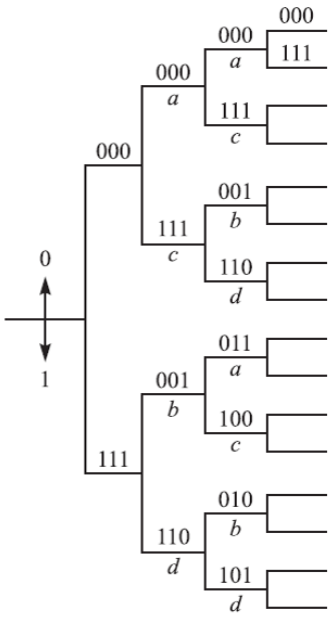


Figura 3.7: Diagrama de árbol para un codificador convolucional $(3, 1, 3)$ [42].

entrada es un 0 y la rama inferior si el bit es un 1. Así pues, se traza un camino particular a través del árbol que está determinado por la secuencia de entrada. La estructura se repite después de la tercera etapa. Este comportamiento es consistente con el hecho de que la longitud de la restricción $m = 3$. Es decir, la secuencia de salida de 3 bits en cada etapa está determinada por el bit de entrada y los 2 bits de entrada anteriores, es decir, los 2 bits contenidos en las dos primeras etapas del registro de desplazamiento. El bit de la última etapa del registro de desplazamiento se desplaza a la derecha y no afecta a la salida. Por lo tanto, la secuencia de salida de 3 bits para cada bit de entrada está determinada por el bit de entrada y los cuatro posibles estados del registro de desplazamiento, denotados como $a = 00$, $b = 01$, $c = 10$, $d = 11$ [42].

3.4.4 Representación mediante diagrama de trellis

Si se etiqueta cada nodo del árbol para que corresponda a los cuatro posibles estados del registro de desplazamiento, se aprecia que en la tercera etapa hay dos

nodos con etiqueta a, dos con etiqueta b, dos con etiqueta c y dos con etiqueta d. Todas las ramas que emanan de dos nodos que tienen la misma etiqueta son idénticas en el sentido de que generan secuencias de salida idénticas. Esto significa que los dos nodos que tienen la misma etiqueta pueden fusionarse. Al hacer esto con el árbol que se muestra en la Fig. 3.7, se obtiene otro diagrama, que es más compacto, conocido como diagrama de trellis. Siendo el diagrama de trellis para el codificador convolucional de la Fig. 3.6 el que se muestra en la Fig. 3.8. En el

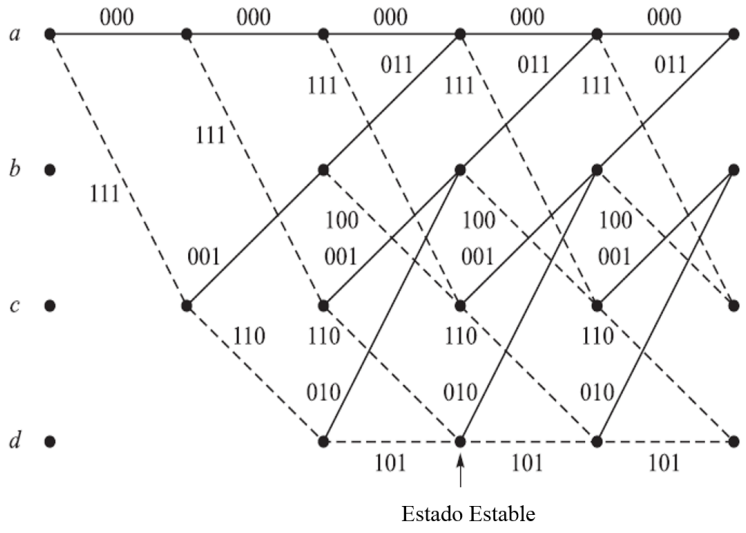


Figura 3.8: Diagrama de trellis para un codificador convolucional $(3, 1, 3)$.

dibujo de este diagrama, se usa la convención de que una línea sólida denota la salida generada por el bit de entrada 0 y una línea punteada la salida generada por el bit de entrada 1. En el ejemplo considerado, después del transitorio inicial, el diagrama contiene cuatro nodos en cada etapa, correspondientes a los cuatro estados del registro de desplazamiento, a, b, c y d. Después de la segunda etapa, cada nodo del trellis tiene dos vías de entrada y dos vías de salida. De los dos caminos de salida, uno corresponde al bit de entrada 0 y el otro al camino seguido si el bit de entrada es un 1 [42].

3.4.5 Algoritmo de decodificación de Viterbi

El algoritmo de decodificación de Viterbi fue descubierto y analizado por Viterbi en 1967. Este algoritmo esencialmente realiza la decodificación de máxima verosimilitud; sin embargo, reduce la carga de cálculo aprovechando la estructura especial en el código de trellis. La ventaja de la decodificación de Viterbi es que la complejidad de un decodificador de Viterbi no es una función del número de símbolos en la secuencia de palabras clave. El algoritmo consiste en calcular una medida de similitud, o distancia, entre la señal recibida, en el tiempo t_i, y todas las trayectorias de trellis que entran en cada estado en el tiempo t_i. El algoritmo de Viterbi elimina de la consideración aquellas trayectorias de trellis que no podrían ser candidatos a la elección de máxima probabilidad. Cuando dos caminos entran en el mismo estado, se elige el que tiene la mejor métrica; este camino se llama el camino superviviente. Esta selección de los caminos supervivientes se realiza para todos los estados. El decodificador continúa de esta manera para avanzar más profundamente en el trellis, tomando decisiones eliminando los caminos menos probables. El rechazo temprano de los caminos menos probables reduce la complejidad de la decodificación [44]. Por lo tanto, el algoritmo de Viterbi es óptimo en el sentido de que siempre encuentra la trayectoria de máxima verosimilitud a través del diagrama de trellis. La decodificación consiste pues en escoger las secuencias más semejantes o más cercanas en el sentido de Hamming (distancia de Hamming) para una decodificación de decisión dura (hard decision) y una distancia Euclideana m dimensional para una decodificación de decisión suave (soft decision) [40].

3.5 Turbo códigos

Los esquemas de codificación concatenada fueron propuestos por primera vez por Forney como un método para lograr grandes ganancias de codificación mediante la combinación de dos o más códigos, de componentes o bloques, de construcción relativamente simples. Los códigos resultantes tenían la capacidad de corrección de

errores mucho más largos, y estaban dotados de una estructura que permitía una decodificación relativamente fácil a moderadamente compleja. Una concatenación en serie de códigos se utiliza con mayor frecuencia para sistemas de potencia limitada, como los transmisores de las sondas del espacio profundo. Se puede pensar en un turbo código como un refinamiento de la estructura de codificación concatenada más un algoritmo iterativo para decodificar la secuencia de código asociada. Los turbo códigos fueron introducidos por primera vez en 1993 por Berrou, Glavieux y Thitimajshima. Los códigos se construyen utilizando dos o más códigos de componentes en diferentes versiones intercaladas de la misma secuencia de información. Mientras que, para los códigos convencionales, el paso final en el decodificador produce bits decodificados de decisión dura (o más generalmente, símbolos descodificados), para que un esquema concatenado, como un código turbo, funcione correctamente, el algoritmo de decodificación no debe limitarse a pasar decisiones duras entre los decodificadores. Para explotar mejor la información aprendida de cada decodificador, el algoritmo de decodificación debe efectuar un intercambio de decisiones suaves en lugar de decisiones duras. En el caso de un sistema con códigos de dos componentes, el concepto que subyace a la turbo descodificación es pasar las decisiones suaves de la salida de un decodificador a la entrada del otro decodificador, e iterar este proceso varias veces para producir decisiones más fiables [44].
En la Fig. 3.9 se muestra un turbo codificador básico. Este consiste de un codificador sistemático recursivo que emplea dos codificadores convolucionales sistemáticos recursivos en paralelo, en los que el segundo codificador va precedido de un intercalador.

El uso de dos codificadores convolucionales recursivos junto con el intercalador produce un código que contiene muy pocas palabras de código de bajo peso. Esta característica no implica necesariamente que la distancia libre del código concatenado sea especialmente grande. Sin embargo, el uso del intercalador en conjunción con los dos codificadores da como resultado palabras de código que tienen relativamente pocos vecinos más cercanos. Es decir, las palabras de código son relativamente escasas. Por lo tanto, la ganancia de codificación lograda por un turbo código se debe en parte

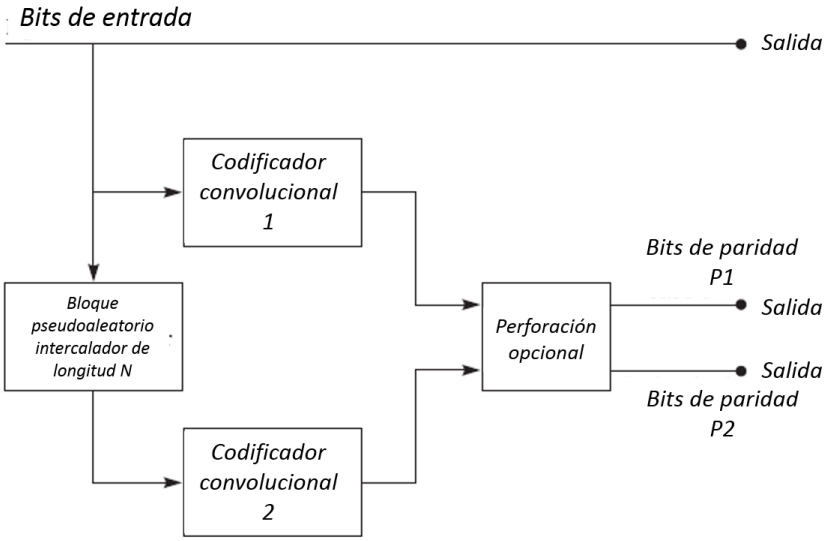


Figura 3.9: Esquema en bloques de un turbo codificador.

a esta característica, es decir, la reducción del número de palabras de código vecinas más cercanas, llamada multiplicidad, que resultan del intercalado. Un turbo código estándar, como el mostrado en la Fig. 3.9, se describe completamente mediante los códigos constitutivos, que suelen ser similares, y el patrón de intercalado, que suele denotarse con Π . Los códigos constitutivos, al ser recursivos y sistemáticos, vienen dados por su matriz generadora de la forma [42]:

$$G(D) = [1 \quad \frac{g_2(D)}{g_1(D)}], \tag{3.20}$$

donde $g_1(D)$ y $g_2(D)$ indican las conexiones de retroalimentación y de avance, respectivamente.

3.5.1 Decodificación para turbo códigos

El algoritmo de turbo decodificación se basa en el uso iterativo del algoritmo Log-APP o el algoritmo Max-Log-APP. Los valores L a posteriori pueden escribirse como la suma de tres términos como:

$$L(u_i) = L_c y_i^s + L^{(a)}(u_i) + L^{(e)}(u_i), \tag{3.21}$$

donde:

$$L_c y_i^s = \frac{4\sqrt{\varepsilon}_c y_i^s}{N_0}, \tag{3.22}$$

$$L^{(a)}(u_i) = \ln \frac{P(u_i = 0)}{P(u_i = 1)}, \tag{3.23}$$

$$L^{(e)}(u_i) = max^*_{(\sigma_{i-1},\sigma_i)\in S_1}\{\alpha_{i-1}(\sigma_{i-1}) + \frac{2y_i^p c_i^p}{N_0} + \beta_i(\sigma_i)\} - \tag{3.24}$$
$$max^*_{(\sigma_{i-1},\sigma_i)\in S_0}\{\alpha_{i-1}(\sigma_{i-1}) + \frac{2y_i^p c_i^p}{N_0} + \beta_i(\sigma_i).\}$$

El término $L_c y_i^s$ se denomina valor del canal L y denota el efecto de las salidas del canal correspondientes a los bits sistemáticos. El segundo término $L^{(a)}(u_i)$ es el valor L a priori y es una función de las probabilidades a priori de la secuencia de información. El término final, $L^{(e)}(u_i)$, representa el valor L extrínseco o información extrínseca que es la parte del valor L a posteriori que no depende de las probabilidades a priori y de la información sistemática en la salida del canal.

Sea $u = (u_1, u_2, ..., u_N)$ una secuencia de información binaria y $c^p = (c_1^p, c_2^p, ..., c_N^p)$ los bits de paridad en la salida. La secuencia de información se pasa por el entrelazador para obtener $u' = (u'_1, u'_2, ..., u'_N)$, y esta secuencia se aplica al segundo codificador para generar la secuencia de paridad $c'^p = (c_1'^p, c_2'^p, ..., c_N'^p)$. Las secuencias u, c^p y c'^p son moduladas por BPSK y transmitidas por un canal Gaussiano. Las correspondientes secuencias de salida se denotan por y^s , y^e, y y'^p. El decodificador MAP para el primer código constitutivo recibe el par (y^s, y^p). En la primera iteración el decodificador asume que todos los bits son equiprobables, y por lo tanto los valores L a priori se ponen a cero. Teniendo acceso a (y^s, y^p), el primer decodificador calcula los valores L a posteriori. A la salida del primer decodificador, se resta los valores L del canal a los valores L a posteriori para calcular los valores L extrínsecos. Estos valores se denotan por $L_{12}^{(e)}(u_i)$ y son permutables por el entrelazador Π. Luego son utilizados por el segundo decodificador como sus valores L a posteriori. Además de esta información, el segundo decodificador se suministra con y'^p y una versión

permutable de y^s después de pasarlo por el entrelazador Π. El segundo decodificador calcula los valores L extrínsecos denotados por $L_{21}^{(e)}(u_i)$ y después de permutarlos por Π^{-1} los suministra al primer codificador, que en la siguiente iteración utiliza estos valores como sus valores L a priori. Este proceso continúa ya sea durante un número fijo de iteraciones o hasta que se cumpla un determinado criterio. Después de la última iteración se utilizan los valores $L(u_i)$ a posteriori para tomar la decisión final. El bloque de construcción del turbo decodificador es un decodificador SISO con entradas ys , y^p, y $L^{(a)}(u_i)$ y salidas $L^{(e)}(u_i)$ y $L(u_i)$. En la decodificación iterativa, $L^{(a)}(u_i)$ se sustituye por los valores L extrínsecos proporcionados por el otro decodificador [42]. En la Fig. 3.10 se muestra el diagrama de un turbo decodificador.

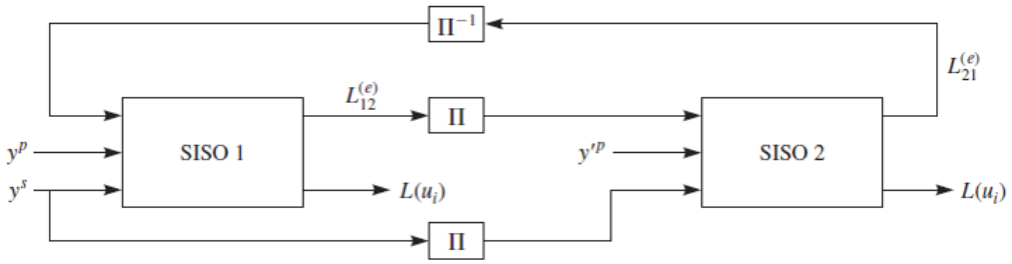


Figura 3.10: Esquema en bloques de un turbo decodificador [42].

Capítulo 4

Codificación conjunta de fuente y canal

Los diseños de los codificadores de fuente y canal, en la mayoría de los sistemas de transmisión, se consideran tradicionalmente de forma independiente debido al teorema de separación fuente-canal de Shannon; como se representa en la Fig. 4.1. Sin embargo, el teorema de Shannon asume efectivamente que el codificador de fuente elimina toda la redundancia y que el codificador de canal corrige todos los errores. En los sistemas prácticos suele haber redundancia residual en la salida del codificador de fuente. Por lo tanto, a menudo se puede mejorar el rendimiento considerando conjuntamente los diseños de la fuente y del canal [9]; ya que aplicar el principio de separación en un sistema con longitudes de bloque limitadas, produce algunos problemas tales como que [45]:

- Es imposible realizar códigos de canal que tengan una tasa de error de bits residuales de cero.
- Los bits de salida del codificador de la fuente no son independientes y están distribuidos uniformemente, es decir, los bits codificados contienen redundancias residuales.
- La distorsión en la salida del decodificador de la fuente depende de cuál bit tiene el error.

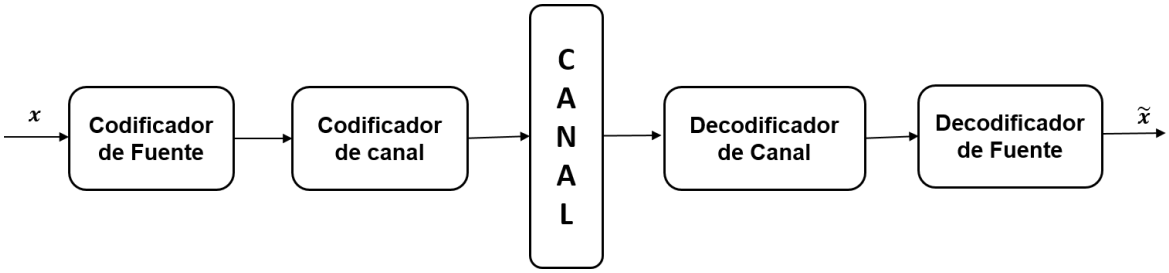


Figura 4.1: Principio de separación de Shannon.

En la Fig. 4.2 se muestra un esquema general de un sistema de comunicación digital que emplea codificación conjunta de fuente y canal. La codificación conjunta

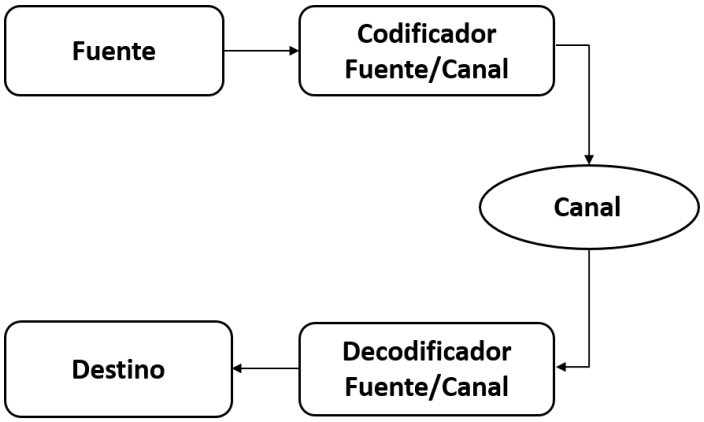


Figura 4.2: Sistema de comunicación digital con codificación conjunta de fuente y canal.

de fuente y canal se convierte en una alternativa interesante a los sistemas en cascada, porque en algunos casos no introduce redundancia en la fuente y trata de reducir el número de errores generados por el canal. La denominada codificación conjunta de fuente y canal se refiere al diseño de un codificador de fuente que tiene en cuenta las características del canal, o incluso al diseño de un decodificador de canal que tiene en cuenta la redundancia restante del codificador de fuente [40].

4.1 Clasificación de los esquemas de codificación conjunta de fuente y canal

En los últimos años, la idea de la codificación conjunta de fuente y canal ha ido ganando cada vez más atención en la comunidad de investigación [11]; debido a la creciente importancia de las comunicaciones inalámbricas, donde están presentes canales que son ruidosos y de ancho de banda limitado. Los enfoques de codificación conjunta de fuente y canal se pueden clasificar principalmente en cuatro categorías según el autor de [5], dichas categorías son:

1. Codificadores conjunto de fuente y canal
2. Codificadores concatenados de fuente y canal
3. Codificación de fuente y canal con protección desigual a errores
4. Codificadores de fuente y canal restringidos

La primera categoría se designa así porque las operaciones de codificación de fuente y canal están verdaderamente integradas. En la segunda categoría los codificadores se ponen en una configuración de cascada con codificadores de fuente y canal conocidos y con una tasa de transmisión fija para maximizar el rendimiento del sistema. En la tercera clase la salida del codificador de fuente recibe una protección desigual basada en el efecto de los errores en la secuencia reconstruida. La cuarta categoría se denomina así porque el codificador y/o decodificador de fuente se modifican para tener en cuenta la presencia de un canal ruidoso determinado. Dentro de esta última clase se encuentra otro subconjunto, aquellos que utilizan algún conocimiento de las propiedades de la fuente o del codificador de fuente para detectar errores de canal y compensar sus efectos. En el presente trabajo, el esquema de codificación conjunta de fuente y canal elegido es el correspondiente a la tercera clase, es decir, codificación de fuente y canal con protección desigual de errores.

4.2 Protección desigual contra errores y ocultación de errores

Debido a que los bits de salida del codificador de fuente no son igualmente sensibles a los errores de bits, algunos de los bits de datos provocan degradaciones de calidad más fuertes que otros si se produce un error, aunque de cualquier manera se utilizan para decodificar. La información de significación de la fuente (SSI) puede derivarse de eso y ser utilizada por el esquema de codificación de canal para aplicar una mayor protección contra errores a los bits más sensibles; este esquema se denomina protección desigual de errores (UEP) [45]. En la Fig. 4.3 se muestra dicho esquema.

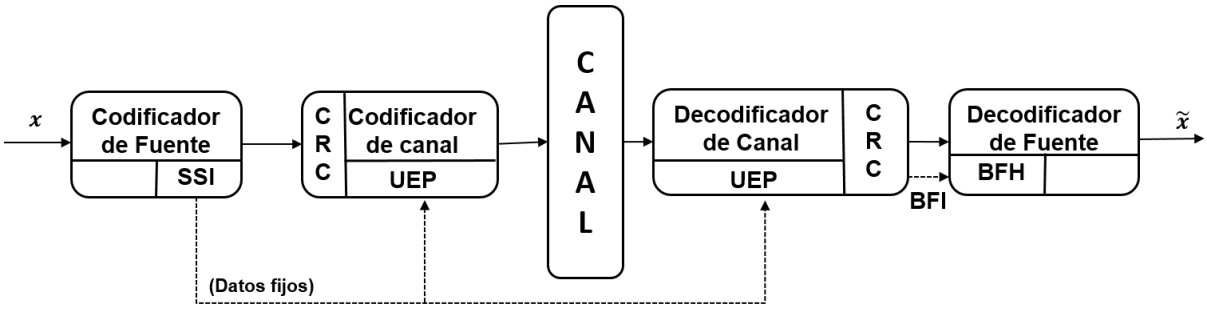


Figura 4.3: Protección desigual de errores y ocultamiento de errores. [45].

Las redundancias residuales en los bits de salida del codificador de fuente pueden ser explotadas para ocultar errores de bits, lo que es conocido como mal manejo de la trama (BFH), por ejemplo, mediante la repetición de los bits del bloque anterior; siempre y cuando los bits estén correlacionados en el tiempo. Para iniciar la ocultación de errores se requiere una detección de errores, que normalmente se realiza mediante un código de canal de detección de errores (comprobación de redundancia cíclica (CRC)). Si se detectan errores, este se comunica al decodificador de fuente mediante el indicador de trama defectuosa (BFI). Los bits fuertemente sensibles del codificador de fuente se codifican en el CRC y después la palabra de código del CRC

se codifica en el canal junto con todos los demás bits de datos restantes. Así, en combinación con la UEP, los bits del codificador de la fuente se dividen en varias clases, que están protegidas por cantidades individualmente ajustables de redundancia de código de canal. Afortunadamente, los bits más sensibles son típicamente los más fuertemente correlacionados al mismo tiempo, es decir, la protección desigual de errores y la ocultación de errores pueden combinarse muy eficientemente. Por lo que ambos esquemas se utilizan muy frecuentemente, por ejemplo, en todas las normas de radiocomunicaciones móviles [45].

Algunas condiciones para la generación de códigos de protección desiguales contra errores, según el autor de [46], se encuentran en los teoremas presentados a continuación. Enfocándose principalmente en las propiedades de la matriz de comprobación de paridad H de los códigos UEP. Considerando los dígitos correspondientes a las columnas de H que dependen linealmente de $2f_i$ o $2f_i + 1$ otras columnas de H. Los cuales se denominan f_i protegidos. Se denota por f_1 el nivel de protección más bajo y por f_z el nivel de protección más alto y $f_1 = f_1 + j - 1$. La $i - ésima$ columna de H se denominará h_i y el síndrome de un vector v se denota $S(v)$.

Teorema 1

Consideremos un código V que consiste en todos los vectores en el espacio nulo de una matriz de comprobación de paridad H. El código V puede corregir cualquier patrón de peso f_1 o menos. Además, si se produce cualquier patrón de error de peso f_i o menos, cualquier dígito protegido f_i o mayor puede ser decodificado correctamente.

Prueba: Dado que no puede existir una relación de dependencia lineal que implique menos de $2f_1 + 1$ columnas, el código V debe ser capaz de corregir cualquier error de peso menor o igual a f_1.

Sea $\omega(E)$ el peso de Hamming del vector E. Para probar la segunda parte del teorema basta con mostrar que el síndrome de cualquier patrón de error E_1, $0 < \omega(E_1) \leq f_i$, el cual contiene un error en el p-ésimo dígito que es f_i protegido único tanto para el síndrome de cualquier patrón de error E_2, $0 < \omega(E_2) \leq f_i$, que no incluye un error en el p-ésimo dígito, como para el síndrome de cualquier patrón de error E_3, $0 < \omega(E_3) \leq f_i$, que contiene un error diferente en el p-ésimo dígito.

Sea v cualquier palabra de código en V

$$E_1 = [e_{11}, e_{12}, e_{13}, ...e_{1p} \neq 0, ...e_{1n}] \tag{4.1}$$

$$E_1 = [e_{21}, e_{22}, e_{23}, ...e_{21p} \neq 0, ...e_{2n}] \tag{4.2}$$

$$E_1 = [e_{31}, e_{32}, e_{33}, ...e_{3p} \neq 0, ...e_{3n}] \tag{4.3}$$

Entonces

$$S(v) = 0, \qquad S(v + E_1) = S(E_1), \qquad S(v + E_2) = S(E_2), \tag{4.4}$$

de modo que

$$S(E_1) = \sum_{i=1}^{n} e_{1i} h_i \tag{4.5}$$

y

$$S(E_2) = \sum_{i=1}^{n} e_{2i} h_i \tag{4.6}$$

Asumiendo que existe un patrón de error E_1 y un patrón de error E_2 tal que $S(E_1) = S(E_2)$, entonces

$$\sum_{i=1}^{n} (e_{1i} - e_{2i}) h_i = 0 \tag{4.7}$$

Como $e_{1p} \neq 0$ y $e_{2p} = 0$, entonces un conjunto de $2f_i$ columnas como máximo de H, incluyendo la columna h_p, es linealmente dependiente, lo que es contrario a la hipótesis. por lo tanto la suposición de que los patrones de errores E_1 y E_2 pueden ser encontrados tal que $S(E_1) = S(E_2)$ es falsa.

Asumiendo que existe un patrón de error E_1 y un patrón de error E_3 tal que $S(E_1) = S(E_3)$, entonces

$$\sum_{i=1}^{n} (e_{1i} - e_{3i}) h_i = 0 \tag{4.8}$$

Dado que e_{3i} es desigual tanto de 0 como de e_{1p}, entonces un conjunto de $2f_i - 1$ columnas como máximo de H, incluyendo la columna h_p, contrario a la hipótesis. por lo tanto la suposición de que los patrones de errores E_1 y E_3 pueden ser encontrados tal que $S(E_1) = S(E_3)$ es falsa.

Así, si se produce cualquier patrón de error de peso inferior o igual a f_i que afecte

al p-ésimo dígito, el síndrome especificará el error particular que se produjo en este p-ésimo dígito. Además, si se produce un patrón de error de peso menor o igual a f_i que no involucre el p-ésimo dígito, el hecho de que el p-ésimo dígito sea correcto puede determinarse a partir del síndrome. Por lo tanto, el teorema está probado.

Los códigos descritos en el Teorema 1 tienen las siguientes propiedades. Si un patrón de error de peso no mayor que f_i pero mayor que f_1 involucra el dígito p-ésimo, el síndrome identificará el error en el dígito p-ésimo, permitiendo así la corrección de este dígito. Sin embargo, algunos dígitos incorrectos pueden seguir siendo incorrectos y algunos dígitos correctos pueden ser erróneamente corregidos para convertirse en dígitos incorrectos. Esto puede ocurrir ya que el síndrome puede ser el mismo que el de algún otro patrón de error de peso menor o igual a f_i que tenga el mismo error en el dígito p-ésimo. También puede ser cierto que si se produce una pauta de error de peso mayor que f_1 pero no mayor que f_i que no implique el p-ésimo dígito, el síndrome resultante puede indicar erróneamente una pauta de error incorrecta pero no indicará un error en el dígito p-ésimo [46].

El teorema 2 considera las condiciones para que una palabra entera sea decodificada correctamente.

Teorema 2

Sea un código de grupo lineal con los t_1 dígitos f_1 protegidos, los t_2 dígitos f_2 protegidos, ..., los t_z dígitos f_z protegidos. Además de la protección de dígitos discutida en el Teorema 1, cualquier patrón de error puede ser corregido siempre que haya f_i o menos errores en los f_i dígitos o menos protegidos, $i = 1, 2, ..., z$.

Prueba: Es suficiente con demostrar que el síndrome de cualquier patrón de error E_1, que satisface las condiciones anteriores es único de: 1) el síndrome de cualquier otro patrón de error E_2 que también satisface las condiciones anteriores, y 2) el síndrome de cualquier patrón de error, E_3, que no satisface necesariamente la condición anterior, donde $\omega(E_3) \leq \omega(E_1)$ y E_3 es distinto de E_1.

Suponiendo que los patrones de error E_1 y E_2 existen tal que $S(E_1) = S(E_2)$. Entonces

$$\sum_{i=1}^{n}(e_{1i} - e_{2i})h_i = 0 \tag{4.9}$$

Dado que E_1 y E_2 difieren, debe haber un dígito muy protegido en el que difieren. Suponiendo que el dígito más protegido en el que E_1 y E_2 difieren es μ protegido. Por la hipótesis, E_1 y E_2 no pueden cada uno tener más que μ errores en los μ o menos dígitos que están protegidos. Por lo tanto, la relación anterior describe un conjunto de a lo sumo 2μ columnas de H, incluyendo una que corresponde a un μ dígito protegido, que es linealmente dependiente, contrariamente a la hipótesis. Por lo tanto, no se puede encontrar ningún dígito altamente protegido en el que E_1 y E_2 difieran y por lo tanto si $S(E_1) = S(E_2)$, entonces $E_1 = E_2$.

Suponiendo que los patrones de error E_1 y E_3 existen tal que $S(E_1) = S(E_3)$. Entonces

$$\sum_{i=1}^{n}(e_{1i} - e_{3i})h_i = 0 \tag{4.10}$$

Sea $\omega_1 = \omega(E_1)$, $\omega_3 = \omega(E_3)$. Asumiendo que el dígito más protegido en el que E_1 y E_3 difieren es μ protegido.Y E_1 y E_3 no difieren en los ψ dígitos que no son cero y que están protegidos por μ o más. Por lo tanto, hay a lo sumo $\omega_1 - \psi$ e_{1i} no cero que no suman a cero con el correspondiente e_{3i}. Puede haber como máximo $\omega_3 - \psi$ e_{3i} no cero que no se suman a cero con la correspondiente e_{ij}. Pero $\omega_1 - \psi \leqslant \mu$, y como $\omega_3 \leqslant \omega_1$ sabemos $\omega_3 - \psi \leqslant \mu$. Así la suma anterior indica que existe un conjunto de 2μ o menos columnas de H, incluyendo una columna correspondiente a un μ dígito protegido que es linealmente dependiente, contrariamente a la hipótesis. Por lo tanto, no puede haber ningún dígito protegido más alto en el que E_1 y E_3 difieran y, por lo tanto, no se pueden encontrar patrones de error E_1 y E_3 de tal manera que $S(E_1) = S(E_3)$. Por lo tanto, el teorema está probado.

Otra cuestión a considerar es el requisito mínimo de protección contra errores de los dígitos de control en un código UEP. Esta cuestión surge cuando se contempla la generación de códigos con dígitos de información altamente protegidos y dígitos de control poco protegidos. Las columnas de control son linealmente independientes, por lo que cada dígito de control está implicado en alguna relación de dependencia lineal que involucra dígitos de información. Por lo tanto, cada dígito de control está tan protegido como el dígito de información menos protegido con el que está involucrado en una relación de dependencia lineal [46].

A fin de proporcionar una protección adicional contra los errores para un dígito de un código de grupo, hay que hacer que la columna de la matriz H correspondiente a ese dígito sea de mayor independencia lineal que las demás columnas de H. Un método para lograr esto en el caso de los códigos binarios es añadir un número de unos a la columna elegida y añadir un número de columnas de manera que cada columna esté implicada en al menos una relación de dependencia lineal [46].

4.2.1 Método para generar el código

Un procedimiento para construir los códigos UEP, basado en la observación de que las ρ columnas de la matriz de comprobación de paridad correspondientes a las ρ posiciones de comprobación deben formar una base para el espacio de columnas. Elijiendo k el número de dígitos de información, y b_i la protección contra errores para el i-ésimo dígito de información, $i = 1$ a k. Selecciona un conjunto de ρ tuplas linealmente independientes para formar la base. Luego se elije una combinación lineal de $2b_i$ o más elementos base para la i-ésima columna de información de H, de manera que ninguna combinación lineal de t columnas que involucre a la i-ésima columna de información sea en sí misma una combinación de menos de $2b_i + 1 - t$ vectores base [46].

En la Fig. 4.4 se muestra la matriz de verificación del código UEP construido por el método de base para $k = 5$ dígitos de información.

El dígito m_1, está protegido contra hasta 1 error

El dígito m_2, está protegido contra hasta 2 errores

El dígito m_3, está protegido contra hasta 3 errores

El dígito m_4, está protegido contra hasta 4 errores

El dígito m_5, está protegido contra hasta 5 errores.

Este método de elementos bases se hace difícil para más de cinco dígitos de información cuando se intenta utilizar el menor número posible de dígitos de control [46].

$$H = \begin{pmatrix}
c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & m_5 & m_4 & m_3 & m_2 & m_1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1
\end{pmatrix}$$

Figura 4.4: Matriz de verificación del código UEP construido por el método de base [46].

4.3 Decodificación de canal controlado por la fuente

La idea con este enfoque, es explotar las redundancias residuales, más sistemáticamente que con la protección desigual contra a errores, como información apriori en el procedimiento de decisión de los bits de datos en la decodificación de canal para reducir la tasa de error de los bits. Este concepto puede aplicarse muy eficientemente para la decodificación de códigos convolucionales binarios; además, la idea puede extenderse también a los decodificadores de salida suave los cuales proporcionan información confiable para los bits de datos. Estas confiabilidades pueden

ser explotadas por los decodificadores fuente basados en estimaciones [45]. En la Fig. 4.5 se representa el esquema básico para la decodificación de canal controlado por la fuente y la decodificación de fuente basado en estimación. Para una correcta ponderación de la información apriori y de las salidas de canales suaves, se debe disponer de una información de estado del canal (CSI), por ejemplo, la variación del ruido.

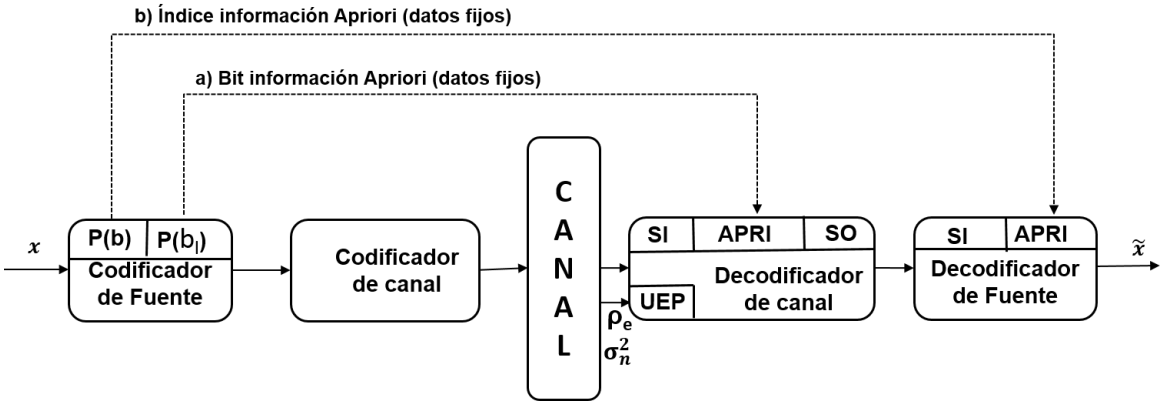


Figura 4.5: a) Decodificación de canal controlado por la fuente y b) decodificación de fuente basado en estimación.

En [47] se presenta este método de decodificación de canales controlados por la fuente usando información a priori y a posteriori sobre los bits de la fuente. En dicho enfoque la información a priori y a posteriori (SAI) de la fuente le dice al decodificador con una cierta probabilidad cuál será el siguiente bit a decodificar. Es mucho mejor dar al decodificador del canal toda la información conocida sobre los bits que se van a decodificar, es decir, sobre los valores de correlación o de sesgo, en lugar de dejar que cometa errores mediante una decodificación a ciegas e inducirlo a ocultar estos errores posteriormente. Mientras sea posible el ocultamiento o el post-procesamiento, sigue habiendo cierta correlación en la señal de origen. Esta correlación debería ser utilizada por los decodificadores de canales. El objetivo de la decodificación del canal controlado por la fuente es reducir la tasa de errores del decodificador del canal suministrando al decodificador del canal información de la fuente.

Con este método presentado en [47] se realiza una modificación del algoritmo de decodificación de Viterbi (VA) para trellises binarios que utiliza información a priori o a posteriori sobre la probabilidad de bits de la fuente para una mejor decodificación, además de entradas suaves e información de estado del canal. El algoritmo se combina con el algoritmo de viterbi de salida suave (SOVA) y un estimador para la correlación residual de los bits de fuente para lograr la decodificación de canal controlada por fuente para bits de fuente enmarcados.

4.3.1 Rederivación del Algoritmo de Viterbi con información suave a priori y a posteriori.

El algoritmo de Viterbi (VA) suele derivarse como un estimador de secuencia de máxima verosimilitud. Se comienza un paso antes de la estimación a posteriori de la secuencia de estados de un trellis binario y se deriba el VA más general usando la notación: las ramas del trellis están etiquetadas con el bit de información u_k y con los bits codificados $X_k = (x_{k1}, ..., x_{kn}, ..., x_{kN})$ de un código convolucional de tasa $\frac{1}{N}$. El código podría ser perforado, en cuyo caso no se transmitirían todos los bits codificados y se lograría una tasa mayor. También se podría usar cualquier código de bloque que tenga una representación de trellis binaria. El decodificador busca la secuencia de estado $s^{(m)}$ y por lo tanto la secuencia de información deseada $u^{(m)}$ maximizando la probabilidad a posteriori [47]:

$$\max_m P(s^{(m)}|y). \tag{4.11}$$

Como y no depende de m, se puede maximizar de forma equivalente

$$\max_m p(y|s^{(m)})P(s^{(m)}). \tag{4.12}$$

Existe independencia estadística de la u_k pertinente dentro de la región de decisión del VA. Fuera de esta región de decisión la u_k puede estar correlacionada. Así con

$$P(s_k) = P(s_{k-1})P(u_k) \tag{4.13}$$

se obtiene

$$\max_m \left\{ P(s_{k-1}^m) \Pi_{l=0}^{k-1} p(y_l | s_{l-1}^m, s_l^m) P(u_k^m) p(y_k | s_{k-1}^m, s_k^m) \right\} \tag{4.14}$$

$$p(y_k | s_{k-1}^m, s_k^m) = p(y_k | x_k^m) = \Pi_{n=1}^N p(y_{k,n} | x_{k,n}^m) \tag{4.15}$$

El máximo no cambia si se toma el logaritmo, se multiplica por 2 y se añaden dos constantes que son independientes de m. Denotando la primera parte del algoritmo de la ecuanción 4.16 por el valor métrico $\frac{M_{k-1}^m}{2}$ y sustituyendo el logaritmo del producto por la suma del logaritmo; la maximización es ahora:

$$\max_m \{M_k^m\} = \max_m \left\{ M_{k-1}^m + [2 \log P(u_k^m) - C_u] + 2 \sum_{n-1}^{N} [2 \log p(y_{k,n} | x_{k,n}^m) - C_y] \right\} \tag{4.16}$$

Se selecciona por conveniencia las constantes:

$$C_u = \log P(u_k = +1) + \log P(u_k = -1) \tag{4.17}$$

$$C_y = \log(p(y_{k,n} | x_{k,n} = +1)) + \log(p(y_{k,n} | x_{k,n} = -1)) \tag{4.18}$$

y se obtiene:

$$M_k^m = M_{k-1}^m + \sum_{n=1}^{N} x_{k,n}^m \log \frac{p(y_{k,n} | x_{k,n} = +1)}{p(y_{k,n} | x_{k,n} = -1)} + u_k^m \log \frac{u_k = +1}{u_k = -1} \tag{4.19}$$

Cambiando la notación de la ecuación 4.19 quedaría:

$$M_k^m = M_{k-1}^m + \sum_{n=1}^{N} x_{k,n}^m L_{c_{k,n}} y_{k,n} + u_k^m L(u_k) \tag{4.20}$$

Esta ligera modificación de la métrica del algoritmo de Viterbi en 4.20 incorpora la información a priori o a posteriori sobre la probabilidad de los bits de la fuente. Este algoritmo de Viterbi modificado se llama APRI-VA. Si el canal es muy bueno $L_{c_{k,n}}$ será mayor que $|L(u_k)|$ y la decodificación se basa en los valores del

canal recibido. Si el canal es malo, como durante un desvanecimiento profundo, la decodificación se basa en la información APRI $L(u)$. Es similar como en un filtro de Kalman donde un valor de predicción es actualizado por las métricas [47].

4.4 Decodificación de la fuente basada en estimaciones

En los sistemas convencionales, para la decodificación de la fuente se suelen aplicar procedimientos sencillos, por ejemplo, tableros de cuantificación. Sin embargo, si la codificación de la fuente no es perfecta y quedan redundancias residuales, por ejemplo, correlaciones en el tiempo, en los bits de datos, este conocimiento a priori puede explotarse para una mejor decodificación de la fuente realizando estimaciones (óptimas) de la entrada del codificador de la fuente [45]. Este esquema, de manera general, se representa en la Fig. 4.5.

Esta redundancia residual es usada para el diseño de codificadores conjuntos fuente/canal en [48]. En este enfoque parten de un esquema de decodificación que incorpore la corrección de errores. La corrección de errores usando códigos convolucionales es posible limitando explícitamente las posibles transiciones de código a código basadas en la entrada de código anterior y la estructura del codificador. En el receptor, un decodificador compara el flujo de datos recibido con la información a priori sobre la estructura del código. La salida del decodificador es la secuencia que más probablemente sea la secuencia transmitida. La salida del codificador fuente contiene redundancia; esto, a su vez, implica que habrá ciertas transiciones de código a código que serán más probables que otras. Por lo tanto,se puede diseñar un decodificador que aproveche este hecho para corregir los errores que se produzcan en el canal [48]. La estructura del sistema para este enfoque se muestra en la Fig. 4.6, donde el codificador de fuente actúa ahora como un codificador conjunto de fuente y canal. El bloque decodificador actúa como un decodificador de canal. La ausencia de un codificador de canal explícito significa que no se producen gastos generales debidos

a la codificación del canal; la estructura que utiliza el decodificador se debe a la redundancia residual en la salida del codificador de la fuente.

Para el diseño del bloque decodificador, como se desea corregir los errores incurridos

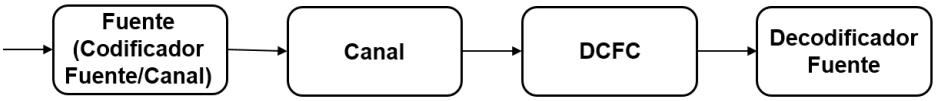


Figura 4.6: Codificador de fuente actuando como codificador conjunto de fuente y canal

en el canal, se puede ver como el diseño de un decodificador que aumenta la probabilidad de tomar una decisión correcta. En la teoría de la comunicación clásica, el receptor óptimo, en el sentido de maximizar la probabilidad de tomar una decisión correcta, selecciona α_j para maximizar $P[\theta_i = \alpha_j|\hat{\theta}_i = \alpha_n]$ donde α_i son elementos del alfabeto de entrada del canal, θ_i es el símbolo transmitido en el tiempo i, y $\hat{\theta}_i$ es el valor recibido corrupto en el tiempo i. Es decir, el receptor maximiza la probabilidad aposteriori de recibir θ_i. Se puede apreciar que la secuencia de entrada del canal es simplemente la salida del codificador de la fuente, y debido a la redundancia de esta secuencia, se incluye el símbolo anterior en la cantidad a maximizar. Así, el decodificador maximiza la cantidad

$$L(j, m, n) = P[\theta_i = \alpha_j|\theta_{i-1} = \alpha_m, \hat{\theta}_i = \alpha_n]. \tag{4.21}$$

Esta métrica no es la óptima, sin embargo, el uso de ella puede llevar a una cantidad significativa de corrección de errores [48]. Debido a la memoria inherente en esta expresión, el decodificador tiene que examinar las secuencias en lugar de decodificar un símbolo a la vez. Así, el decodificador maximiza $\Pi_{i=1}^{N}\{P[\theta_i|\theta_{i-1}, \hat{\theta}_i]\}$ o equivalentemente $\log \Pi_{i=1}^{N}\{P[\theta_i|\theta_{i-1}, \hat{\theta}_i]\}$. Notando que

$$\log \Pi_{i=1}^{N}\{P[\theta_i|\theta_{i-1}, \hat{\theta}_i\} = \sum_{i=1}^{N} \log\{P[\theta_i|\theta_{i-1}, \hat{\theta}_i]\}. \tag{4.22}$$

la cantidad anterior es similar en forma a la métrica de la trayectoria de un decodificador convolucional. La métrica de la rama es simplemente $\log(L(j, m, n))$. Se puede

escribir L en términos de las probabilidades de transición del codificador de la fuente y las probabilidades de transición del canal. Estas probabilidades de transición pueden estimarse por separado, y luego las estimaciones pueden combinarse para obtener L. Utilizamos la definición de probabilidad condicional se puede escribir L como:

$$L = \frac{P[\theta_i = \alpha_j | \theta_{i-1} = \alpha_m, \hat{\theta}_i = \alpha_n]}{P[\theta_{i-1} = \alpha_m, \hat{\theta}_i = \alpha_n]} \tag{4.23}$$

Suponiendo un canal sin memoria, y utilizando la independencia condicional

$$P[\hat{\theta}_i = \alpha_n | \theta_i = \alpha_j, \theta_{i-1} = \alpha_m] = P[\hat{\theta}_i = \alpha_n | \theta_i = \alpha_j] \tag{4.24}$$

Así L se convierte en

$$L = \frac{P[\hat{\theta}_i = \alpha_n | \theta_i = \alpha_j] P[\theta_i = \alpha_j | \theta_{i-1} = \alpha_m]}{P[\hat{\theta}_i = \alpha_n | \theta_{i-1} = \alpha_m]} \tag{4.25}$$

El numerador de L está en términos de las probabilidades de transición del codificador de fuente y las probabilidades de transición del canal. Para obtener el denominador de L en la misma forma se hace la siguiente afirmación.

$$P[\hat{\theta}_i = \alpha_n | \theta_{i-1} = \alpha_m] = \sum_l \{P[\hat{\theta}_i = \alpha_n | \theta_i = \alpha_l] P[\theta_i = \alpha_l | \theta_{i-1} = \alpha_m]\} \tag{4.26}$$

Prueba de la afirmación

$$D = \sum_l \{P[\hat{\theta}_i = \alpha_n | \theta_i = \alpha_l] P[\theta_i = \alpha_l | \theta_{i-1} = \alpha_m]\} \tag{4.27}$$

$$D = \sum_l \{P[\hat{\theta}_i = \alpha_n | \theta_i = \alpha_l] \frac{P[\theta_i = \alpha_l, \theta_{i-1} = \alpha_m]}{P[\theta_{i-1} = \alpha_m]}\} \tag{4.28}$$

$$D = \frac{1}{P[\theta_{i-1} = \alpha_m]} \sum_l \{P[\hat{\theta}_i = \alpha_n | \theta_i = \alpha_l] P[\theta_{i-1} = \alpha_m, \theta_i = \alpha_l]\} \tag{4.29}$$

Asumiendo un canal sin memoria

$$P[\hat{\theta}_i = \alpha_n | \theta_i = \alpha_l] = P[\hat{\theta}_i = \alpha_n | \theta_i = \alpha_l, \theta_{i-1} = \alpha_m]\}. \tag{4.30}$$

Por lo que

$$D = \frac{1}{P[\theta_{i-1} = \alpha_m]} \sum_l \{P[\theta_{i-1} = \alpha_m, \theta_i = \alpha_l] P[\hat{\theta}_i = \alpha_n | \theta_i = \alpha_l, \theta_{i-1} = \alpha_m]\} \tag{4.31}$$

$$D = \frac{1}{P[\theta_{i-1} = \alpha_m]} \sum_l \{P[\hat{\theta}_i = \alpha_n, \theta_{i-1} = \alpha_m, \theta_i = \alpha_l]\} \tag{4.32}$$

$$D = \frac{P[\hat{\theta}_i = \alpha_n, \theta_{i-1} = \alpha_m]}{P[\theta_{i-1} = \alpha_m]} \tag{4.33}$$

que por la definición de probabilidad condicional prueba la afirmación.

Así la métrica $L(j, m, n)$ puede escribirse como

$$L = \frac{P[\hat{\theta}_i = \alpha_n | \theta_i = \alpha_j] P[\theta_i = \alpha_j | \theta_{i-1} = \alpha_m]}{\sum_l P[\hat{\theta}_i = \alpha_n | \theta_i = \alpha_l] P[\theta_i = \alpha_l | \theta_{i-1} = \alpha_m]} \tag{4.34}$$

La expresión para L está compuesta por dos conjuntos de probabilidades condicionales:

$$P[\theta_i = \alpha_j | \hat{\theta}_i = \alpha_n], \qquad j, n = 1, ..., N \tag{4.35}$$

y

$$P[\theta_i = \alpha_j | \theta_{i-1} = \alpha_m], \qquad j, m = 1, ..., N \tag{4.36}$$

donde N es el tamaño del alfabeto de salida del codificador de fuente. Por lo tanto, para obtener L se necesita estimar cantidades de $2N^2$. Para la mayoría de los sistemas de compresión de datos, este número suele ser bastante pequeño. El primer conjunto de probabilidades condicionales depende sólo de las estadísticas del canal, y el segundo conjunto de probabilidades de transición depende sólo del codificador y de las estadísticas de la fuente. Asumiendo un modelo simétrico binario para el canal, entonces dado el código binario usado sobre el canal y una estimación de la probabilidad de error del canal, es una tarea simple obtener el primer conjunto de probabilidades de transición. Para obtener el segundo conjunto de probabilidades de transición, se puede utilizar una secuencia de entrenamiento [48].

4.5 Decodificación iterativa de canal y fuente

En la Fig. 4.7 se muestra el esquema para un sistema con decodificación iterativa de canal y fuente, aunque es similar al de los sistemas de la Fig. 4.5, el concepto teórico es diferente: el transmisor se interpreta como un esquema de codificación

de canal concatenado en serie; los códigos constitutivos son las redundancias residuales implícitas dentro de los vectores de bits de salida del codificador de fuente y la redundancia explícita del código de canal. Esto exige el uso del principio turbo para la decodificación. Como en todos los esquemas de decodificación iterativa, debe disponerse de decodificadores para los dos códigos constituyentes, capaces de intercambiar información extrínseca sobre los bits de datos dentro de las iteraciones. Entre los esquemas de decodificación realizables, la decodificación iterativa del canal y fuente funciona mejor, pero al mismo tiempo es el enfoque más complejo [45].

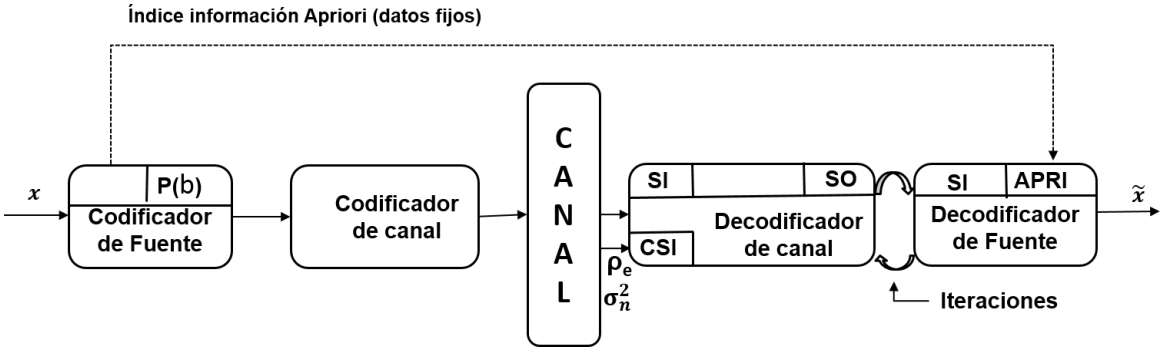


Figura 4.7: Sistema con decodificación iterativa de canal y fuente.

En [49] se presenta un procedimiento iterativo para la aproximación de la solución óptima, que se basa en el principio de decodificación iterativa de turbo códigos. El esquema del decodificador iterativo conjunto de fuente y canal se ilustra en la Fig. 4.8.

El índice I_k^j consta del siguiente conjunto ordenado de bits:

$$I_k^j = \{i_{l,k}^j \in \{0,1\} : l = 1,2,...,N^j\} \tag{4.37}$$

Una realización del bit $i_{l,k}^j$ se denota por $\lambda_l^j, \mu_l^j \in \{0,1\}$. Usando estas definiciones, el índice de la probabilidad a posteriori (APP) viene dada por:

$$P^{(C)}(i_{l,k}^j = \lambda_l^j | \tilde{V}_k) = B_k^{(C,1)} P_e^{(C)}(i_{l,k}^j = \lambda_l^j) p(\tilde{i_{l,k}^j}) | i_{l,k}^j = \lambda_l^j) P_a^{(C)}(i_{l,k}^j = \lambda_l^j), \tag{4.38}$$

donde las probabilidades de bits extrínsecos se calculan como

$$P_e^{(C)}(i_{l,k}^j = \lambda_l^j) = B^{(C,2)_k} \sum ... \sum p(\tilde{C}_k | C_k^{(\mu)}) \Pi_{m=1,m\jmath}^{M} \Pi_{\xi=1}^{N^m} p(i_{\xi,k}^{\tilde{m}} | i_{\xi,k}^m = \mu_\xi^m) \tag{4.39}$$

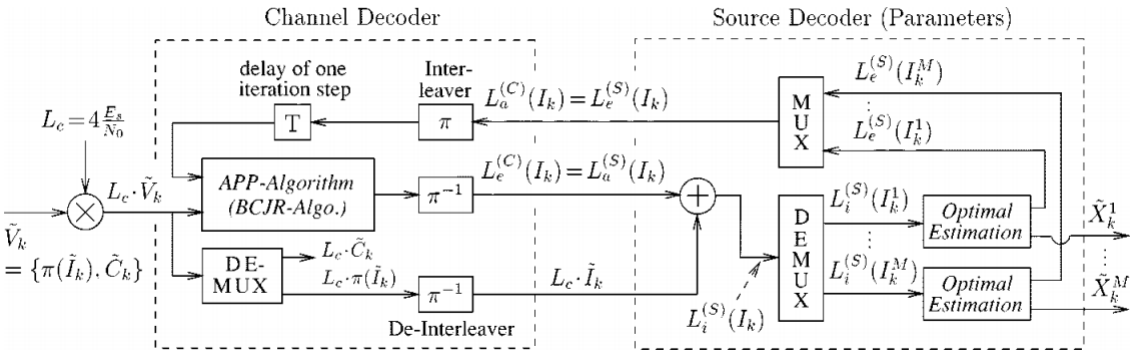


Figura 4.8: Decodificación iterativa conjunta de fuente y canal [49].

$$P_a^{(C)}(i_{\xi,k}^m = \mu_\xi^m)\Pi_{v=1,v\not=}p(i_{v,k}^{\tilde{j}}|i_{v,k}^j = \mu_v^j)P_a^{(C)}(i_{v,k}^j = \mu_v^j)$$

y las probabilidades a priori de los bits $P_a^{(C)}(i_{l,k}^j)$ se derivan de las probabilidades a priori del índice calculando las distribuciones marginales

$$P_a^{(C)}(i_{l,k}^j = \lambda_l^j) = \sum P_k(\mu^j|\tilde{V}_{k-1}), \qquad j = 1, ..., M \qquad l = 1, ..., N^j. \tag{4.40}$$

Como sólo se dispone de probabilidades extrínsecas para los bits, se utiliza la siguiente aproximación. La idea es simplificar el cálculo del término extrínseco manteniendo el resto del algoritmo óptimo [49].

$$P_k(\lambda_k^j = \lambda^j) \approx \Pi_{l=1}^{N^j} P_e^{(C)}(i_{l,k}^j = \lambda_l^j) \tag{4.41}$$

Por lo que se obtiene

$$P_k(\lambda^j|\tilde{V}_k) = B_k^{(S)}\Pi_{l=1}^{N^j}[P_e^{(C)}(i_{l,k}^j = \lambda_l^j)p(i_{l,k}^{\tilde{j}}|i_{l,k}^j = \lambda_l^j)]P_k(\lambda^j|\tilde{V}_{k-1}) \tag{4.42}$$

donde el término de canal se descompuso en factores para los bits. El producto entre paréntesis puede interpretarse como un nuevo término de canal, es decir, la información extrínseca del decodificador de canal produce virtualmente un canal más fiable para los índices de bits.

Como la probabilidad extrínseca se factoriza, las correlaciones mutuas del índice de bits no se explotan. Para compensar esta pérdida de información se adopta a partir de la decodificación iterativa de los códigos de canal concatenados: a partir de los

valores temporales de los APP para los índices según 4.42 se calculan nuevos APP para los índices de bits y se deriva nueva información a priori para una segunda ejecución del algoritmo APP para la decodificación del canal. Los APP de los bits vienen dados por los APP marginales de los índices [49]

$$P^{(S)}(i_{l,k}^{j} = \lambda_l^j | \tilde{V_k}) = \sum P_k(\mu^j | \tilde{V_k}). \tag{4.43}$$

Usando 4.42, se puede reescribir como

$$P^{(S)}(i_{l,k}^{j} = \lambda_l^j | \tilde{V_k}) = B_k^{(S,1)} P_e^{(S)}(i_{l,k}^{j} = \lambda_l^j) p(\tilde{i_{l,k}^{j}} | i_{l,k}^{j} = \lambda_l^j) P_a^{(S)}(i_{l,k}^{j} = \lambda_l^j), \tag{4.44}$$

con las probabilidades extrínsecas

$$P_e^{(S)}(i_{l,k}^{j} = \lambda_l^j) = B_k^{(S,2)} \sum P_k(\mu^j | \tilde{V_{k-1}}) \Pi_{v=1,v\neq l} p(\tilde{i_{v,k}^{j}} | i_{v,k}^{j} = \mu_v^j) P_a^{(S)}(i_{v,k}^{j} = \mu_v^j) \tag{4.45}$$

Los factores $B_k^{(S,1)}$ y $B_k^{(S,2)}$ son constantes que se normalizan. Las probabilidades a priori $P_a^{(S)}(i_{v,k}^{j} = \mu_v^j)$ están definidas por

$$P_a^{(S)}(i_{v,k}^{j} = \mu_v^j) = P_e^{(C)}(i_{l,k}^{j} = \lambda_l^j) \tag{4.46}$$

La probabilidad extrínseca $P_e^{(S)}(i_{l,k}^{j} = \lambda_l^j)$, que se genera a partir de las correlaciones de la fuente, no contiene la información a priori $P_a^{(S)}(i_{v,k}^{j} = \mu_v^j)$ del decodificador del canal para este bit y viceversa. En ambas probabilidades extrínsecas el término del canal $p(\tilde{i_{l,k}^{j}} | i_{l,k}^{j} = \lambda_l^j)$ no está contenido. Por lo tanto, las probabilidades extrínsecas pueden utilizarse como la nueva información a priori en 4.40 para un segundo y posteriores ejecuciones del algoritmo APP (para la decodificación del canal). El proceso de utilizar alternativamente las probabilidades extrínsecas como las nuevas probabilidades a priori se repite varias veces [49]. La decodificación iterativa conjunta de fuente y canal se resume en los siguientes pasos.

- Cálculo del índice de probabilidades a priori utilizando los valores anteriores del índice APP de la etapa anterior $k-1$.
- Cálculo de las probabilidades a priori del bit inicial.

- Decodificador de canal: Cálculo de las probabilidades extrínsecas de los índice de bits xplotando las probabilidades a priori de los bits. Uso de un algoritmo APP eficiente.

- Decodificador de fuente: Cálculo de las probabilidades extrínsecas de los índice de bits usando las probabilidades extrínsecas de los bits, que se calcularon en el paso 3, como la información a priori. Las nuevas probabilidades extrínsecas se utilizan como las nuevas probabilidades a priori para una repetición del Paso 3.

- Después de varias iteraciones de los pasos 3 y 4 los valores finales para los APP de los índices se calculan.

- Estimación de los vectores de los parámetros utilizando el índice APP del paso 5.

- Poner $k := k + 1$ y retornar al paso 1.

Capítulo 5

Diseño e implementación del sistema de comunicación con CCFC

5.1 Descripción del sistema de comunicación propuesto

Los elementos que conforman nuestro sistema de comunicación digital inalámbrico se muestran en la Fig. 5.1, el cual consta de tres procesos fundamentales en el transmisor y sus operaciones inversas en el receptor. El esquema de codificación conjunta de fuente y canal empleado corresponde a la clasificación dada en el capítulo anterior y tomada de [5], denominada codificación conjunta de fuente y canal con protección desigual de errores. La entrada al sistema es una imagen con extensión BMP, la cual se convierte a un vector de bits para su transmisión. La primera etapa del esquema es la codificación de fuente. Mediante esta se logra comprimir la información aprovechando la redundancia existente en los datos. El próximo paso en la transmisión es la codificación de canal, la cual se utiliza para controlar los errores que pueden ocurrir durante la propagación de la señal, dicha protección se realiza de manera desigual. Finalmente se realiza la modulación. Cada uno de estos bloques se explicará detalladamente a continuación.

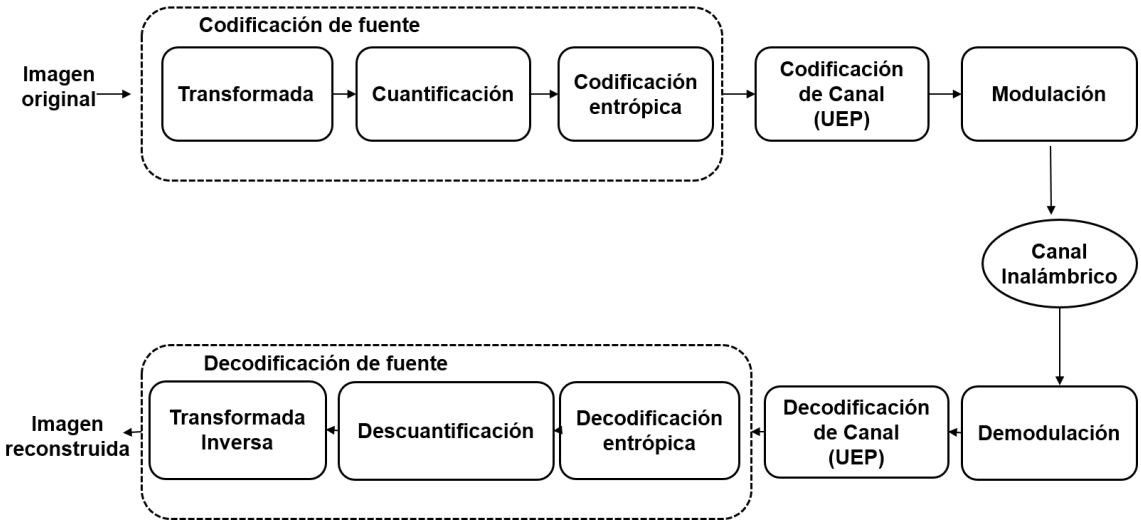


Figura 5.1: Sistema de comunicación digital inalámbrico propuesto.

5.1.1 Compresión de la imagen

El proceso de compresión o codificación de fuente consta de las etapas de transformación, cuantificación y codificación. En nuestro sistema se usan la Transformada Wavelet Discreta, la cuantificación mediante el algoritmo de Offset Binario y la codificación de Huffman. Con la Transformada Wavelet se busca representar la imagen en una base matemática distinta para que la correlación y la redundancia existentes se manifiesten para eliminar aquellos coeficientes donde no está presente la información [50].

La transformada wavelet se define por:

$$W_f(s,\tau) = \int f(t)\Psi_{s,\tau}(t)dt, \tag{5.1}$$

donde la wavelet base está dada por:

$$\Psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\Psi(\frac{t-\tau}{s}). \tag{5.2}$$

Siendo s el factor de escala y τ el factor de traslación. Para nuestro sistema se seleccionó la wavelet de Daubechies de orden 4 por ser wavelets ortogonales de soporte

compacto con un número máximo de momentos nulos, propiedades que las hacen adecuadas para el análisis y procesamiento de señales con soporte finito como son las imágenes [51].

A la salida de la etapa de transformación obtenemos una matriz de números reales de dimensión MxM que se debe cuantificar a una matriz binaria para su transmisión. Para lograr esto se implementó el algoritmo de Offset Binario propuesto en [52], de la siguiente manera:

- Encontrar la "escala completa" de la matriz denotada como E_c, donde E_c se define tal que todos los elementos de la matriz se encuentran en el intervalo $(-E_c/2, E_c/2)$.
- Si la resolución, que es el número de bits que tendrán las muestras, es L bits, el intervalo $[-E_c/2, E_c/2]$ se divide en 2^L intervalos, calculándose el tamaño del intervalo como: $T = \frac{E_c}{2^L}$.
- Definir la representación binaria de un número real A como:

$$Binario\left\{\frac{A+\frac{E_c}{2}}{T}\right\}. \tag{5.3}$$

Después de la cuantificación se obtienen una serie de muestras donde cada muestra es un número binario. Algunas de estas muestras se repetirán con mayor frecuencia que otras. Para explotar esta redundancia y comprimir aún más los datos, se utiliza un codificador de Huffman, donde la longitud de cada código depende de la frecuencia de aparición de cada muestra. A mayor frecuencia de menor longitud será su código asociado. Este codificador genera un árbol binario que tiene por hoja cada muestra, y al recorrer desde el nodo principal hasta cada hoja se obtiene el código Huffman asociado.

Para la descompresión en el lado receptor se decodifica el código de Huffman. La secuencia binaria obtenida se convierte a números reales y se aplica una transformación wavelet inversa.

5.1.2 Codificación de canal

Para la corrección de errores se utiliza la técnica de codificación convolucional como codificación de canal. El codificador convolucional genera una secuencia de bits codificada en función de una tasa de código, que es igual a la relación entre la longitud de la palabra de datos y la longitud de la palabra de código. En el codificador convolucional una secuencia de entrada se codifica a través de un conjunto de registros de desplazamiento y sumadores en módulo 2, por lo que la secuencia codificada depende tanto de la secuencia de mensaje actual como de las secuencias anteriores [53]. En esta etapa de codificación de canal se emplea la técnica UEP de la siguiente manera; la imagen de entrada al codificador es dividida en segmentos los cuales son codificados de forma desigual, los parámetros de los códigos utilizados se dan en secciones posteriores.
Por otro lado, el decodificador procesa el flujo de bits codificados convolucionalmente utilizando el algoritmo de Viterbi.

5.1.3 Modulación y demodulación digital

Mediante la etapa de modulación la secuencia de bits $b\,[n]$ se convierte a una secuencia de símbolos $s\,[n]$, donde cada valor $s\,[n]$ es un número complejo que pertenece a la constelación de la modulación digital. En la Fig. 5.2, se observa la constelación para la modulación QPSK, la cual se emplea en el sistema desarrollado.
La secuencia de símbolos $s\,[n]$ se convierte a pulsos mediante el filtro de conformación de pulsos coseno alzado que se muestra en (5.4)

$$g_{rc}\,(t) = \mathrm{sinc}(\frac{t}{T})\frac{\cos(\pi\beta t)}{1-\frac{4\beta^2t^2}{T^2}}, \tag{5.4}$$

donde $\beta = 0.5$ es el factor de roll-off. Luego se escala por el factor de $\sqrt{E_x}$ para producir la señal banda base compleja (5.5).

$$x(t) = \sqrt{E_x}\sum s\,[n]\,g_{rc}(t-nT), \tag{5.5}$$

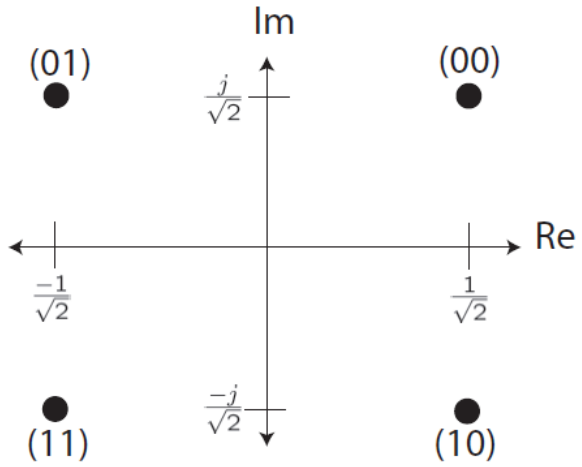


Figura 5.2: Constelación QPSK.

El escalar por el factor $\sqrt{E_x}$ se usa para modelar el efecto de añadir energía o potencia a la señal $x(t)$. La señal $x(t)$ se convierte a una señal pasa banda para viajar en una frecuencia portadora f_c, resultando la señal mostrada en (5.6).

$$x_p(t) = \text{Re}\{x(t)\}\cos(2\pi f_c t) - \text{Im}\{x(t)\}\,\text{sen}(2\pi f_c t). \tag{5.6}$$

Esta señal $x_p(t)$ es lanzada desde las antenas de transmisión al medio de propagación. En el receptor se usa un filtro acoplado con la misma respuesta al impulso que la del transmisor para filtrar la señal recibida. Se implementa un método para disminuir el efecto del error de sincronización de símbolos cuando la forma del pulso no se muestrea en el momento preciso, junto con la sincronización de trama para encontrar el inicio del dato.

Para la sincronización de símbolos se utiliza la función de costo de energía de salida máxima que intenta encontrar la demora que maximiza la energía de la señal que sale del filtro acoplado, la cual se define por J_{MOE} y cuya solución se muestra en (5.7).

$$\tau_d = \arg\max(J_{MOE}(\tau)), \tag{5.7}$$

Para la sincronización de trama se utiliza un correlacionador deslizante, el cual correlaciona la señal recibida con una secuencia de entrenamiento para calcular $R[n]$

como se muestra en (5.8).

$$R[n] = \sum t(k)^* y(n+k), \tag{5.8}$$

Entonces es posible encontrar la demora d que maximice esa correlación como se ilustra en (5.9)

$$\hat{d} = \arg\max(|R(n)|). \tag{5.9}$$

Posteriormente los símbolos se pasan a un bloque de detección para determinar el símbolo más probable para esa observación. Como detector se utiliza el criterio de Máxima Verosimilitud cuya solución se muestra en (5.10)

$$\hat{s}[n] = \arg\max f_{y|s}(y[n] \mid s[n] = s), \tag{5.10}$$

donde $f_{y|s}(.)$ es la distribución condicional de $y\,[n]$ dado $s\,[n]$.

Los símbolos detectados se pasan a un bloque de mapeo de símbolo inverso para producir una buena suposición de los bits transmitidos. El esquema general del modem digital se muestra en la Fig. 5.3.

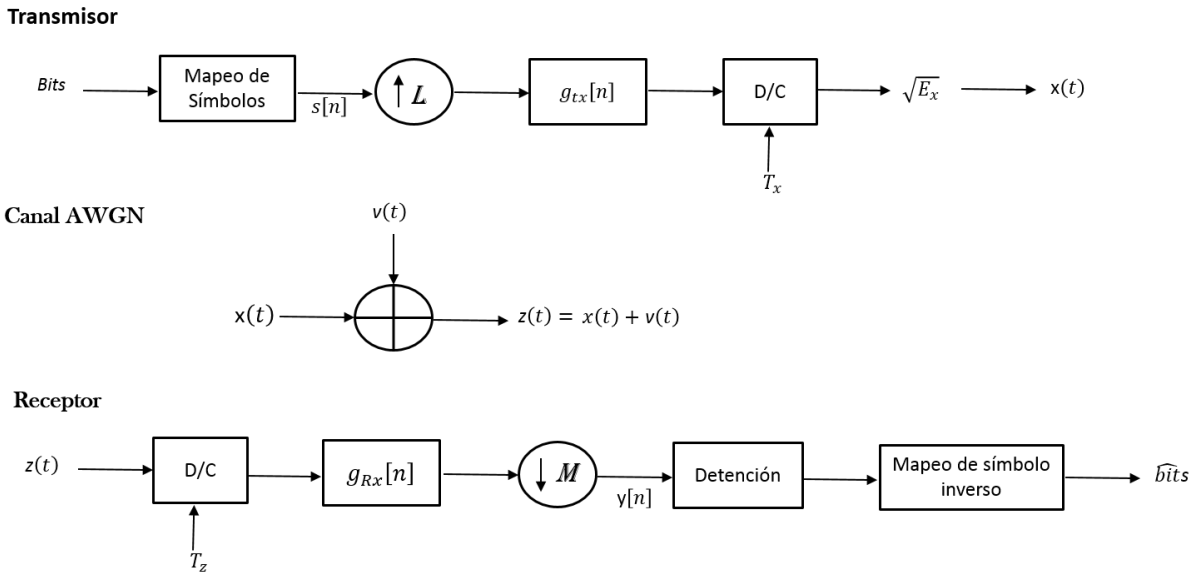


Figura 5.3: Esquema de modulación y demodulación.

5.2 Implementación en la plataforma LabView Communications

El sistema de comunicación digital inalámbrico se diseñó e implementó con el software NI LabVIEW Communications 2.0, utilizando dos módulos NI USRP-2920 y los Instrumentos Virtuales (VIs) programados. Un equipo USRP funciona como transmisor y el otro como receptor, interconectados entre sí con un cable para la sincronización de reloj y uno de ellos con un cable UTP hacia la computadora, como se observa en la Fig. 5.4.

Las imágenes se transmiten a una frecuencia de portadora de 915 MHz con una

Figura 5.4: Conexión física de los módulos NI USRP-2920.

ganancia de 10 dB, utilizando modulación QPSK y un factor de rolloff del filtro de conformación del pulso de 0.5, de manera inalámbrica sobre un canal AWGN.

Se consideraron los siguientes parámetros en la configuración del transmisor y el receptor:

- Factor de sobremuestreo Tx : 8
- Factor de sobremuestreo Rx : 4
- Longitud del paquete : 1000 Bits
- Tiempo de captura : 120 ms
- Tasa de muestreo Tx : 20M muestras/seg
- Tasa de muestreo Rx : 10M muestras/seg
- Filtro conformador de pulso: Raíz de coseno alzado

En la Fig. 5.5 se presenta el esquemático general que constituye el transmisor del sistema y en la Fig. 5.6 el correspondiente al receptor. Estos están constituidos por bloques operativos que a su vez están conformados por diferentes bloques funcio-

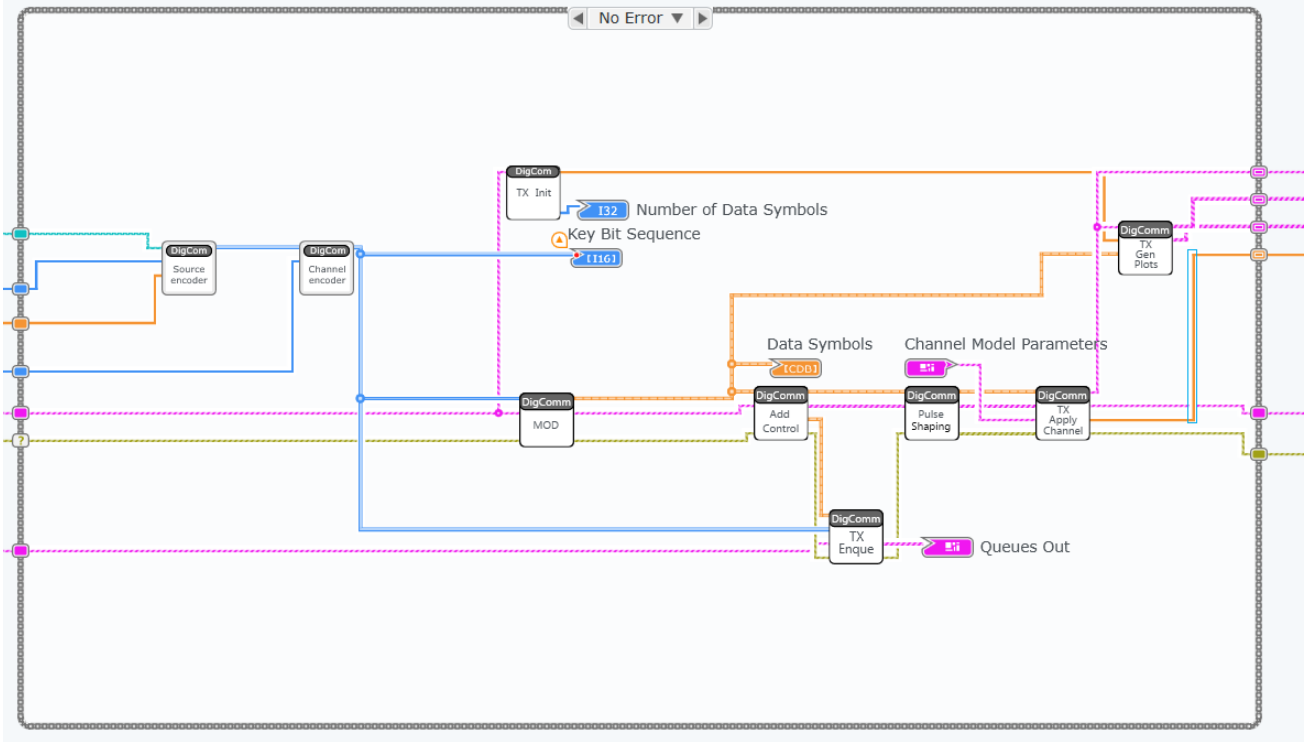


Figura 5.5: Diseño del transmisor en LabView.

nales. Para la implementación del modem digital se usó el manual de prácticas de laboratorio de [54].

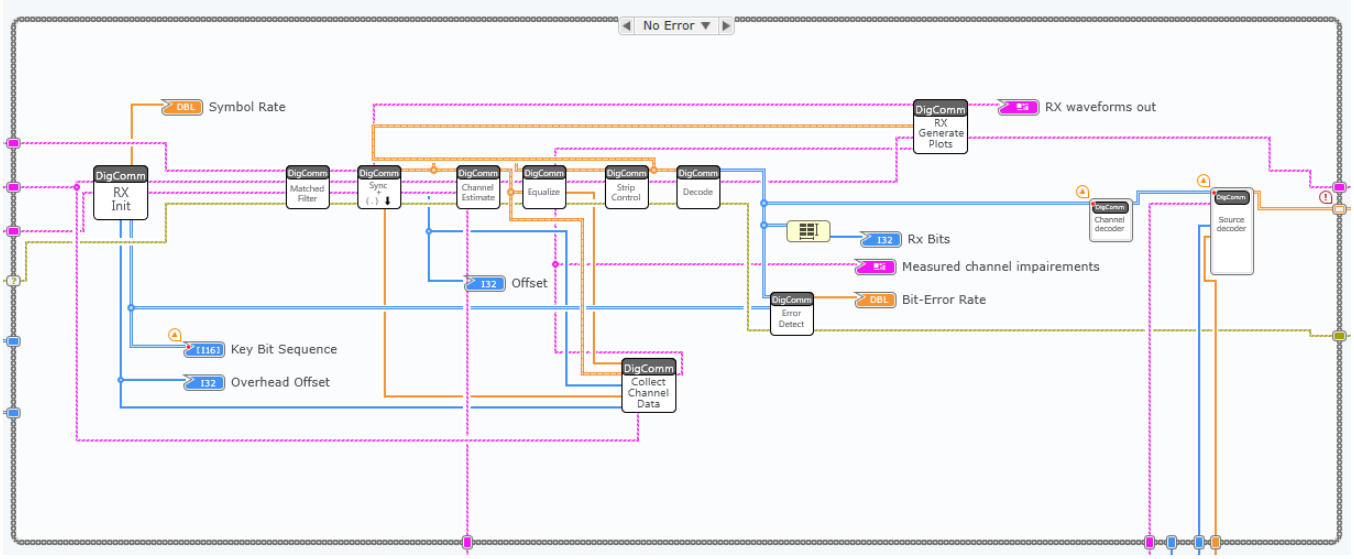


Figura 5.6: Diseño de receptor en LabView.

5.3 Transmisión de imágenes en ambiente real

Una vez diseñado e implementado el sistema de comunicación digital inalámbrico mediante LabView Communications se transmitieron imágenes a través de un canal real. Las imágenes son de tamaño 512x512 píxeles con extensión bmp. El proceso de transmisión y recepción para la imagen Lena se detalla a continuación. En la Fig. 5.7 se presenta la imagen Lena a transmitir.

El primer paso es leer el archivo de imagen mediante el bloque de LabView Read from Binary File, el cual lee datos binarios de un archivo cargado especificando su ruta y devuelve una matriz 1D de números enteros sin signo de 8 bits (U8).

Este conjunto de muestras es la entrada a la primera etapa del sistema, o sea al codificador de fuente, específicamente al bloque de transformación. Donde hacemos uso de la Transformada Wavelet Discreta. El objetivo que se persigue es tratar de comprimir el conjunto de muestras pues mediante la transformación se pueden despreciar varios coeficientes ya que la información solo está presente en algunos de ellos. Al aplicar la TWD obtenemos una serie de coeficientes. En la Fig. 5.8 se muestra gráficamente la salida de la transformación wavelet.

Como se mencionó en secciones anteriores, a la salida de la etapa de transformación

Figura 5.7: Imagen Lena transmitida.

obtenemos una serie de números reales que se cuantifican a una matriz binaria mediante el algoritmo de Offset Binario explicado anteriormente. Para ello se seleccionó una resolución de 16 bits, es decir, 16 bits para cada muestra. Como la cantidad de muestras transformadas son 262144, se obtienen a la salida del cuantificador 4194304 bits. Con el objetivo de eliminar redundancia, ya que estas muestras se repiten, y lograr una mayor compresión de la imagen dichas muestras binarias se codifican entrópicamente mediante el algoritmo de Huffman, que es la etapa final del codificador de fuente. A la salida del cuantificador de Huffman se reduce la cantidad de bits a 1422228. En la Tabla 5.1 se muestra un resumen del proceso de compresión de la imagen Lena.

Para proteger la imagen de errores durante la transmisión por el canal inalámbri-

Tabla 5.1: Compresión de la imagen Lena

Muestras Transformadas	Bits después de Cuantificación	Bits después de Huffman	Porcentaje de compresión
262144	4194304	1422228	66 %

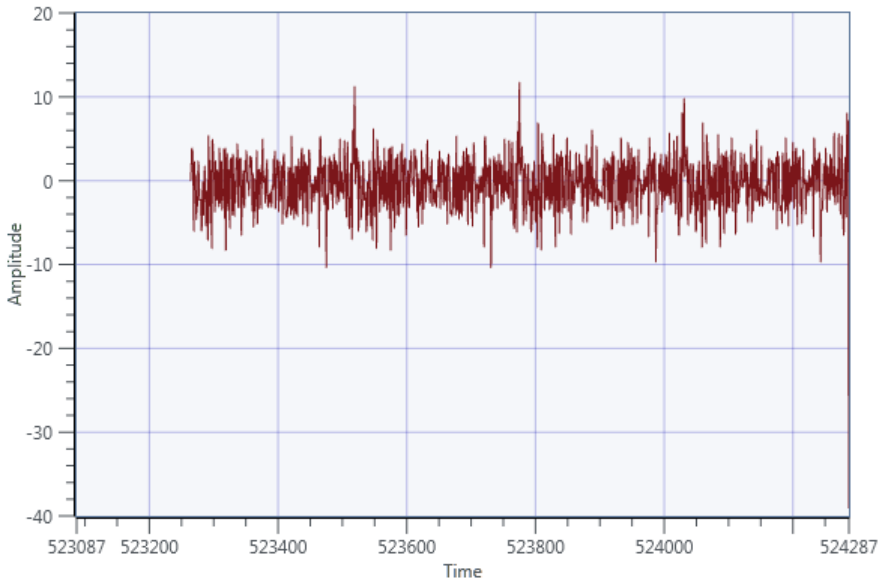


Figura 5.8: Transformada Wavelet Discreta de la imagen Lena.

co, se le agrega redundancia mediante la codificación de canal. Dicha protección se realizó aplicando la técnica UEP dividiendo la secuencia de salida del codificador de fuente en 8 segmentos y aplicando diferentes tasas de codificación, dándole una mayor protección a la parte central de la imagen puesto que es donde se encuentra la información más importante, o sea, el rostro. En la Tabla 5.2 se especifican los valores de codificación.

Tabla 5.2: Codificación convolucional desigual para la imagen Lena

Segmento	**Tasa de codificación**	**Longitud de restricción**
1	1/2	6
2	1/3	6
3	1/4	6
4	1/4	6
5	1/4	6
6	1/4	6
7	1/3	6
8	1/2	6

En la Fig. 5.9 se muestra la forma de onda de la señal que se transmite. La constelación de la señal transmitida y su diagrama de ojo se representan en las Fig. 5.10 y Fig. 5.11 respectivamente.

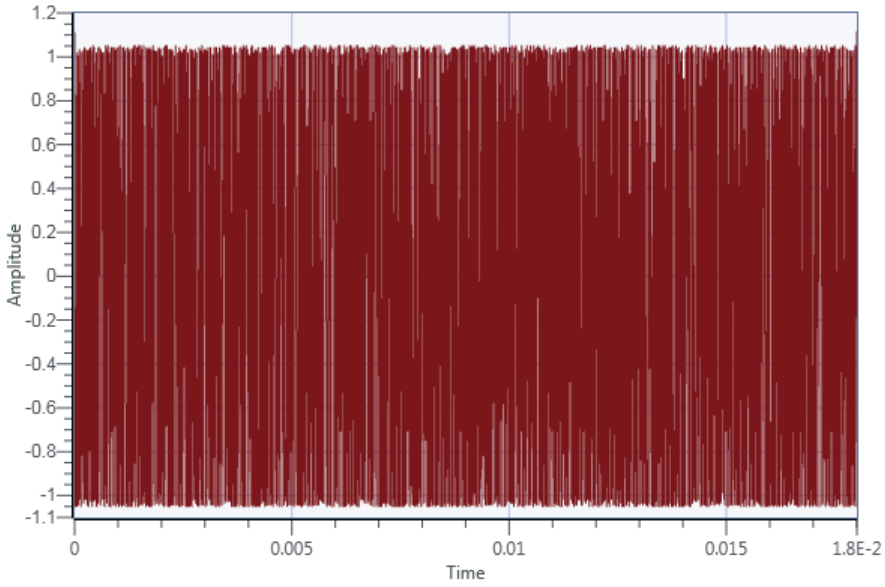


Figura 5.9: Forma de onda de la imagen Lena transmitida.

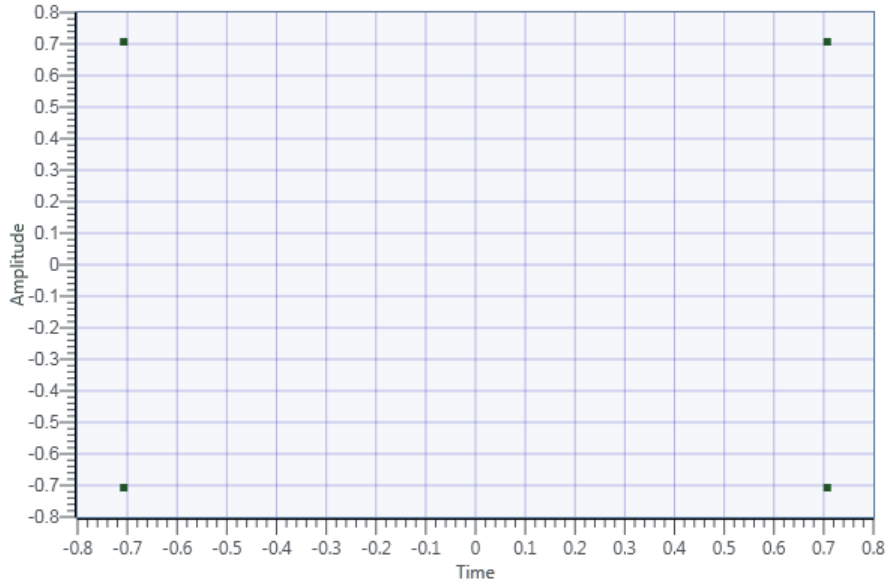


Figura 5.10: Constelación de la imagen Lena transmitida.

Por otro lado en la Fig. 5.12 se observa la forma de onda compleja de la señal

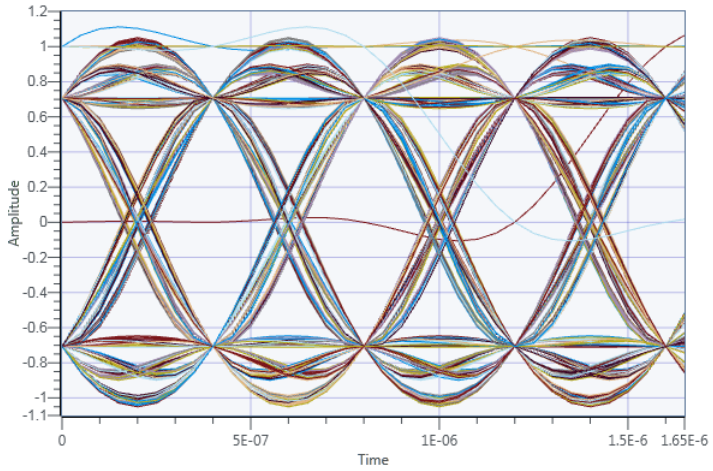


Figura 5.11: Diagrama de ojo de la imagen Lena transmitida.

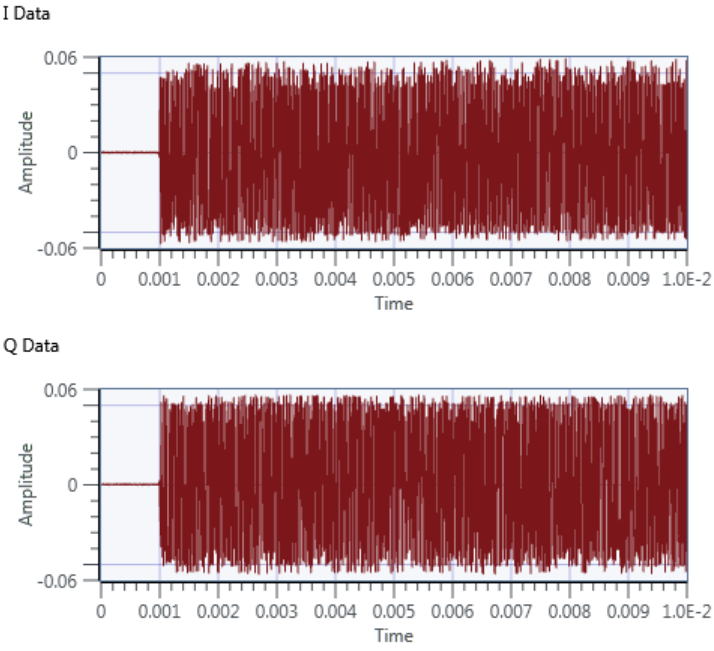


Figura 5.12: Forma de onda de la imagen Lena recibida.

recibida en sus dos componentes en fase y cuadratura y en las Fig. 5.13 y Fig. 5.14 la constelación de la imagen Lena recibida y su diagrama de ojo.

Como se puede apreciar efectivamente la constelación recibida es QPSK. Podemos notar también que el diagrama de ojo tiene buena apertura vertical y horizontal, lo que indica que el sistema presenta inmunidad al ruido y a los errores en el instante de muestreo de la señal; aunque se reflejan los efectos de impedimentos comunes de

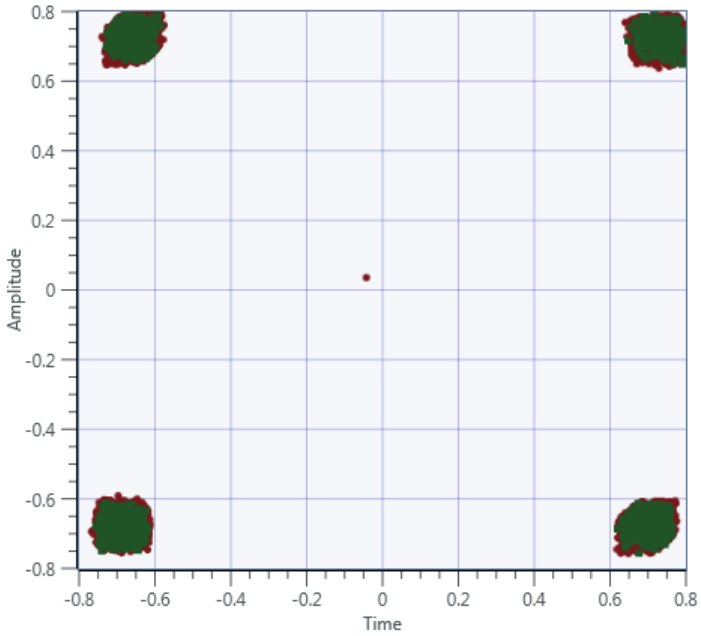


Figura 5.13: Constelación de la imagen Lena recibida.

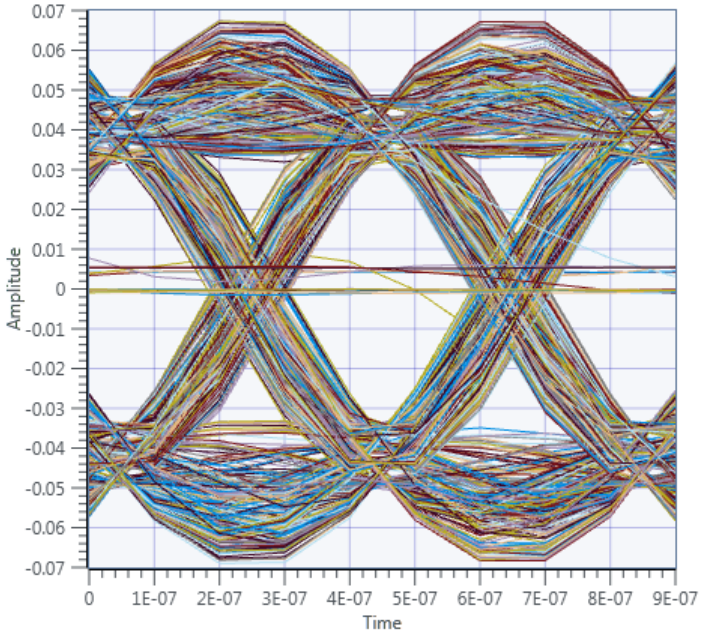


Figura 5.14: Diagrama de ojo de la imagen Lena recibida.

los sistemas digitales, como son lo errores de tiempo.

En el lado del receptor se realizan las operaciones inversas a las realizadas en el transmisor para reconstruir la imagen. Finalmente la secuencia obtenida se guarda

en un archivo creado, el cual se usa para visualizar y conformar la imagen con extensión bmp en el software MATLAB. La imagen Lena reconstruida se muestra en la Fig. 5.15.

Se transmitieron otras imágenes las cuales se presentan junto a su original en las

Figura 5.15: Imagen Lena recibida.

Fig. 5.16, Fig. 5.17 y Fig. 5.18. Como se puede apreciar todas las imágenes se reconstruyeron con calidad y sin distorsión.

5.4 Evaluación del desempeño de la codificación convolucional

En esta sección se analizará el rendimiento de la transmisión de imágenes a través de canales inalámbricos con y sin codificación convolucional y el uso de la técnica UEP sobre la Protección Igual de Errores (EEP).

Se hace una comparación entre el rendimiento del sistema codificado con el del sistema no codificado bajo el mismo canal. El rendimiento del sistema se define como la tasa de transmisión útil en bits por segundo teniendo en cuenta la pérdida debida a los errores de canal.

Figura 5.16: Imagen rostro de mujer a)transmitida y b) recibida.

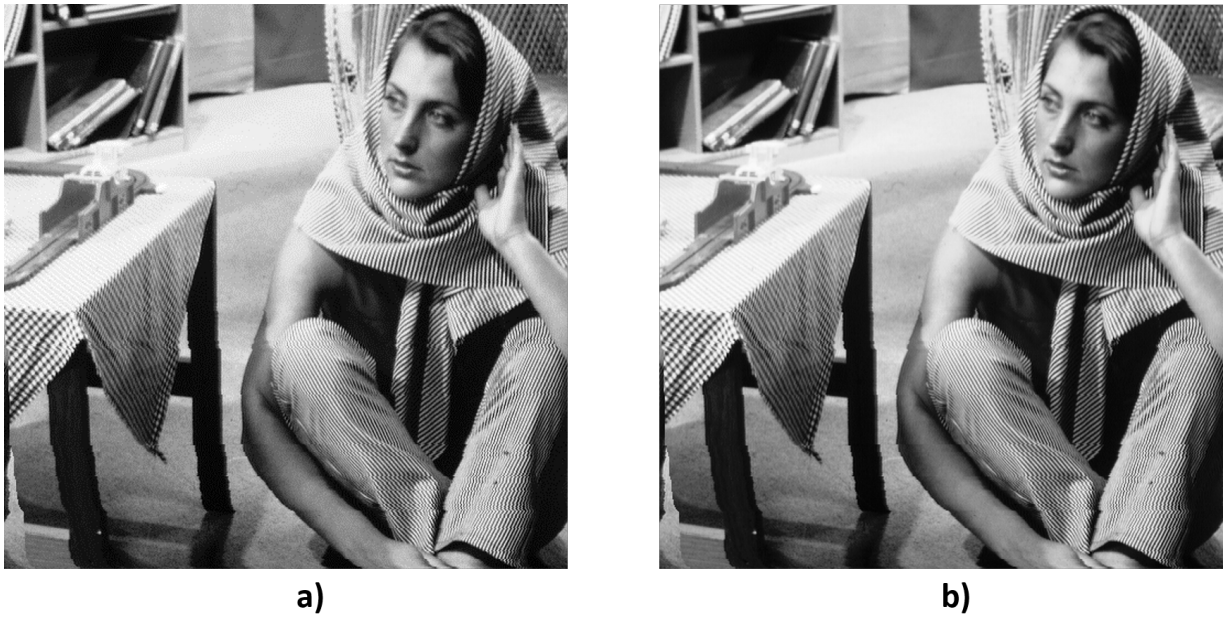


Figura 5.17: Imagen Barbara a) transmitida y b) recibida.

En las Fig. 5.19 y Fig. 5.20 se muestran el rendimiento de la tasa de error de bits (BER) para la transmisión de las imágenes Lena y rostro de mujer con y sin codificación convolucional a través de un canal inalámbrico y el desempeño del sistema frente a la aplicación de las técnica EEP y UEP.

Figura 5.18: Imagen rostro de bebé a) transmitida y b) recibida.

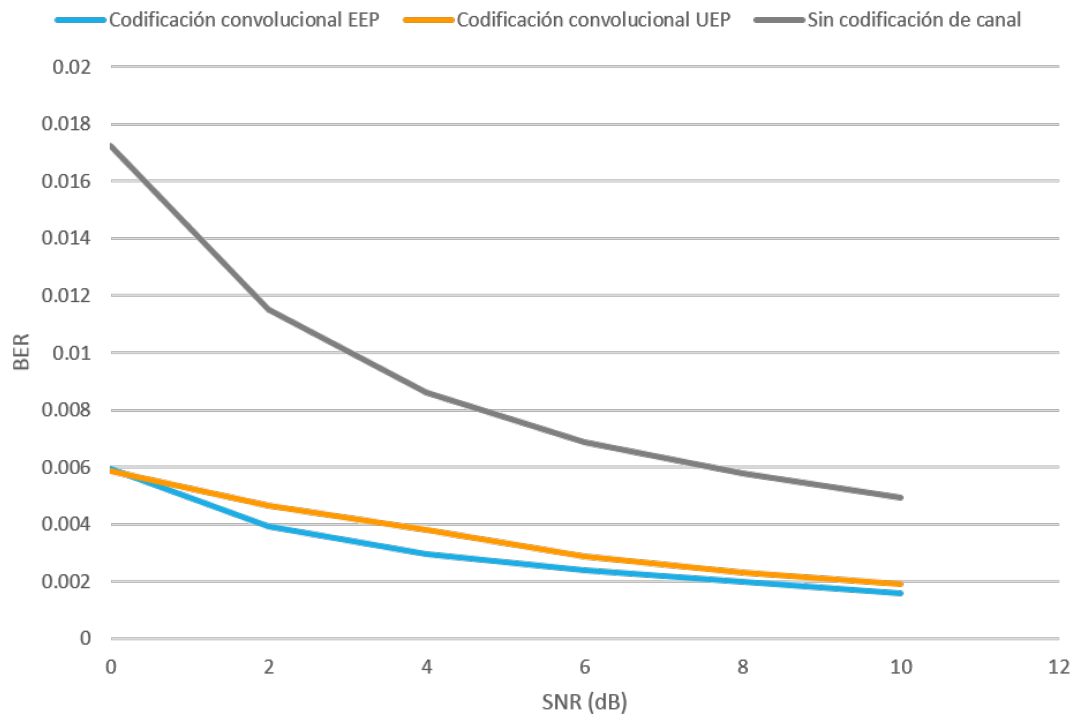


Figura 5.19: Desempeño de la tasa de error de bits para la imagen Lena

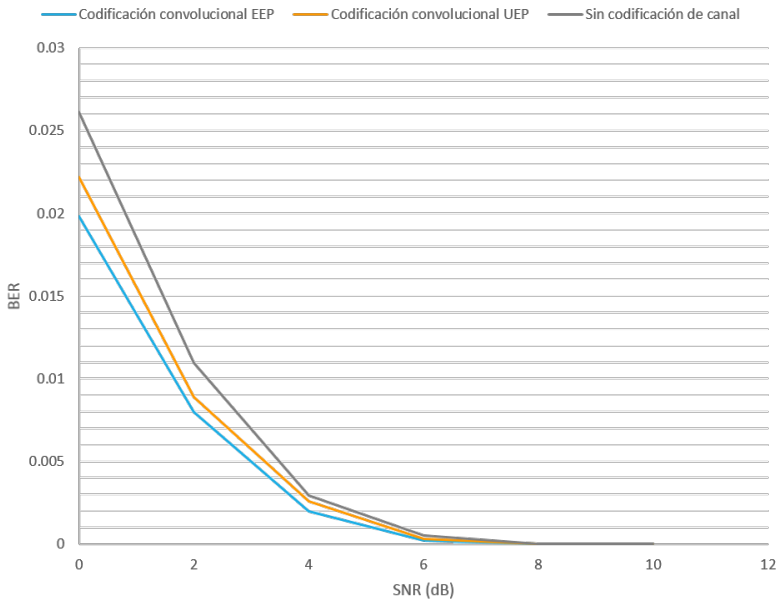


Figura 5.20: Desempeño de la tasa de error de bits para la imagen rostro de mujer

La codificación convolucional se ha utilizado como codificación de canal para la transmisión de las imágenes a través del canal inalámbrico. Como podemos apreciar en el gráfico comparativo de las Fig. 5.19 y Fig. 5.20 el uso de la codificación convolucional aumenta el rendimiento del sistema ya que disminuye la tasa de error de bits. En nuestro sistema empleamos la técnica UEP, que a diferencia de la EEP que protege los datos por igual sin importar su significado cualitativo, este esquema asigna la protección a los datos que se desean transmitir en función de la importancia relativa de sus bits, recibiendo mayor protección los bits más significativos. En la Tabla 5.3 se muestra una comparación entre la cantidad de bits codificados con UEP y EEP.

De los gráficos comparativos anteriores se puede observar que aunque con EEP se

Tabla 5.3: Codificación convolucional UEP y EEP

Esquema	**Bits resultantes**	**Tasa de redundancia**
Codificación convolucional EEP	5688912	75 %
Codificación convolucional UEP	4622246	69 %

logran valores de BER menores estos no difieren grandemente de los obtenidos con UEP. Por lo que, con el uso de la técnica UEP se logra disminuir la redundancia y por tanto el ancho de banda, sin deteriorar la calidad de la imagen en términos de tasa de error de bits.

Capítulo 6

Conclusiones y tabajos futuros

6.1 Conclusiones

Se diseñó e implementó un sistema de comunicación digital inalámbrico con codificación conjunta de fuente y canal mediante el uso de la plataforma de Radio Definido por Software para la transmisión de imágenes a través de un canal real. Dicho sistema comprende compresión, control de errores y modulación.

El algoritmo de compresión implementado mediante la codificación de fuente disminuye la redundancia presente en los datos, lo cual es muy beneficioso para superar el problema de ancho de banda de transmisión limitado.

La codificación de canal mediante códigos convolucionales permitió dotar la información de redundancia útil para combatir los errores durante la transmisión de la imagen por el canal inalámbrico, mejorando el rendimiento del sistema en comparación con esquemas no codificados.

Se aplicó el principio de protección desigual de errores para la codificación de control de errores, brindando mayor protección a los bits más significativos. Dicha implementación favoreció el desempeño del sistema de comunicación puesto que se lograron valores de tasa de error de bits muy cercanos a los obtenidos con la protección igual de errores con una menor redundancia, lo que se traduce a un menor ancho de banda, mateniendo la calidad de la imagen. Además, como al usar un esquema de protección desigual de errores se logra una codificación con menor tasa de redundancia la cantidad de bits resultante a transmitir es menor por tanto el tiempo de transmisión

de un USRP a otro también es menor.

6.2 Trabajos futuros

Con la finalidad de darle continuidad a la investigación se propone para trabajos futuros los siguientes aspectos:

- Diseñar una combinación alternativa del codificador de fuente sustituyendo la codificación entrópica de Huffman por codificación Aritmética para lograr una razón de compresión más alta, aprovechando de mejor forma la entropía de los coeficientes transformados y cuantizados.
- Emplear turbo códigos en el esquema de codificación de canal UEP para lograr un mejor desempeño y robustez del sistema.
- Adaptar el sistema con codificación conjunta de fuente y canal para transmitir imágenes cifradas para que la red inalámbrica sea segura al añadir un nivel adicional de protección y seguridad.

Nomenclaturas

ASCII American Standar Code for Information Interchange (Código Estándar Estadounidense para el Intercambio de Información).

AWGN Additive White Gaussian Noise (Ruido Blanco Guassiano Aditivo).

BCH Bose-Chaudruri-Hocquenghem.

BFI Indicador de Trama Defectuoso.

BMP Bit Map Picture.

BPSK Binary Phase Shift Keying (Modulación por Desplazamiento de Fase Binaria).

CCFC Codificación Conjunta de Fuente y Canal.

CRC Cyclic Redundancy Check (Código de Redundancia Cíclica).

CSI Channel Status Information (Información de Estado del Canal).

DCT Discrete Cosine Transform (Transformada Discreta de Coseno).

DFT Discrete Fourier Transform (Transformada Discreta de Fourier).

DPCM Differential Pulse Code Modulation (Modulación de Código de Impulso Diferencial)

DWT Discrete Wavelet Transform (Transformada Wavelet Discreta).

EEP Equal Error Protection (Protección Igual de Errores).

GIF Graphics Interchange Format (Formato de Intercambio de Gráficos).

DWT High Dynamic Range (Alto Rango Dinámico).

IDCT Inverse Discrete Cosine Transform (Transformada Discreta de Coseno Inversa).

IEEE Institute of Electrical and Electronics Engeneers (Instituto de Ingenieros Eléctricos y Electrónicos).

IFS Iterated Function System (Sistema de Funciones Iteradas).

ISO International Standards Organization (Organización Internacional de Normalización).

JPEG Joint Photographic Experts Group (Grupo Conjunto de Expertos en Fotografía).

KLT Karhunen Loeve Transform (Transformada de Karhunen Loeve).

LBT Lapped Biorthogonal Transform (Transformada Biortogonal Lapped).

LDPC Low-Density Parity-Check Codes (Códigos de Verificación de paridad de Baja Densidad).

LZW Lempel-Ziv-Welch.

OMP Orthogonal Matching Pursuit.

PNG Portable Network Graphics (Gráficos de Red Portátiles).

QPSK Quadrature Phase Shift Keying (Modulación por Desplazamiento de Fase en Cuadratura).

RGB Red Green Blue (Rojo Verde Azul).

RLE Run Length Encoding (Codificación por Longitud de Secuencia).

SNR Signal to Noise Ratio (Relación Señal a Ruido).

SSI Source Significance Information (Información de Significación de la Fuente).

TIFF Tagged Image File Format (Formato de Archivo de Imagen Etiquetada).

UEP Unequal Error Protection (Protección Desigual de Errores).

VA Viterbi Algorithm (Algoritmo de Viterbi).

VLC Variable Length Coding (Codificación de Longitud Variable).

VQ Vector Quantization (Cuantificación Vectorial).

WHT Walsh Hadamard Transform (Transformada de Walsh Hadamard).

Bibliografía

[1] Zhenyu Wu. Joint source/channel coding for image and video transmission. 2005.

[2] Salah A Aliesawi, Dena S Alani, and Abdullah M Awad. Secure image transmission over wireless network. *International Journal of Engineering & Technology*, 7(4):2758–2764, 2018.

[3] M Padmaja and Syed Shameem. Secure image transmission over wireless channels. In *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007)*, volume 4, pages 44–48. IEEE, 2007.

[4] MA El-Iskandarani, Saad M Darwish, and Saad M Abuguba. Combination of 2d chaotic encryption and turbo coding for secure image transmission. *IJCSNS*, 10(11):179, 2010.

[5] Khalid Sayood, Hasan H Otu, and Nejat Demir. Joint source/channel coding for variable length codes. *IEEE Transactions on Communications*, 48(5):787–794, 2000.

[6] Mahin Torki. *JPEG2000 transmission over wireless channels using unequal power allocation*. PhD thesis, School of Engineering Science-Simon Fraser University, 2009.

[7] Billy D Pettijohn, Michael W Hoffman, and Khalid Sayood. Joint source/channel coding using arithmetic codes. *IEEE Transactions on Communications*, 49(5):826–836, 2001.

[8] Thomas Guionnet and Christine Guillemot. Soft decoding and synchronization of arithmetic codes: Application to image transmission over noisy channels. *IEEE transactions on Image Processing*, 12(12):1599–1609, 2003.

[9] Moonseo Park and David J Miller. Joint source-channel decoding for variable-length encoded data by exact and approximate map sequence estimation. *IEEE Transactions on Communications*, 48(1):1–6, 2000.

[10] Jianfei Cai and Chang Wen Chen. Robust joint source-channel coding for image transmission over wireless channels. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(6):962–966, 2000.

[11] Abraham Gabay, Michel Kieffer, and Pierre Duhamel. Joint source-channel coding using real bch codes for robust image transmission. *IEEE transactions on Image Processing*, 16(6):1568–1583, 2007.

[12] Eirina Bourtsoulatze, David Burth Kurka, and Deniz Gündüz. Deep joint source-channel coding for wireless image transmission. *IEEE Transactions on Cognitive Communications and Networking*, 5(3):567–579, 2019.

[13] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[14] Sridhar Vembu, Sergio Verdu, and Yossef Steinberg. The source-channel separation theorem revisited. *IEEE Transactions on Information Theory*, 41(1):44–54, 1995.

[15] Khalid Sayood. *Introduction to data compression*. Newnes, 2012.

[16] Jaya Shukla, Manoj Alwani, and Anil Kumar Tiwari. A survey on lossless image compression methods. In *2010 2nd International Conference on Computer Engineering and Technology*, volume 6, pages V6–136. IEEE, 2010.

[17] Samir Kumar Bandyopadhyay, Tuhin Utsab Paul, and Avishek Raychoudhury. Image compression using approximate matching and run length. *International Journal of advanced Computer science and applications*, 2(6), 2011.

[18] Nora La Serna, Luzmila Pro Concepción, and Carlos Yañez Durán. Compresión de imágenes: Fundamentos, técnicas y formatos. *Revista de investigación de Sistemas e Informática*, 6(1):21–29, 2009.

[19] Robert W Heath Jr. *Introduction to Wireless Digital Communication: A Signal Processing Perspective*. Prentice Hall, 2017.

[20] Fouzia I Khandwani and PE Ajmire. A survey of lossless image compression techniques. *International Journal of Electrical Electronics & Computer Science Engineering*, 5(1), 2018.

[21] Walaa Z Wahba and Ashraf YA Maghari. Lossless image compression techniques comparative study. *Lossless Image Compression Techniques Comparative Study*, 3(2), 2016.

[22] Javier Lezama. Compresión de imágenes-codificación de huffman. *Revista de educación matemática*, 32(1), 2017.

[23] Diya Chudasama, Khushboo Parmar, Dipali Patel, Kruti J Dangarwala, and Shaishav Shah. Survey of image compression method lossless approach. *International Journal of Engineering Research & Technology*, 4(03):981–983, 2015.

[24] Juan Antonio Pérez Ortiz. *Codificación fractal de imágenes*. PhD thesis, Tesis de Master, Universidad de Alicante, 1998.

[25] Sebastian Pérez Becker. Compresión fractal de imágenes. *Foro-Red-Mat: Revista electrónica de contenido matemático*, 29(2):1, 2012.

[26] Akhilesh Kumar Singh AK Malviya. A survey on image compression methods. *International Journal of Engineering and Computer Science*, 6(5), 2017.

[27] Valentín Cruz Rodríguez. Diseño de un codificador de imágenes adaptativo multitransformada mediante el uso de la transformada karhunen-loève. B.S. thesis, 2012.

[28] Darwin Carrión and Danilo Barreno. La transformada discreta del coseno (dct) y su aplicación en la compresión de imágenes. In *Book of Proceedings I International Seminar on Teaching, Research and Linking, preliminary to. IV CONGRESS OF SCIENCE TECHNOLOGY, INNOVATION AND ENTREPRENEURSHIP*, page 65.

[29] Iain EG Richardson. H. 264 and mpeg-4 video compression. *H. 264 and MPEG-4 Video Compression*, pages 201–207, 2003.

[30] Trujillo and Desachy. *Compresión de imágenes mediante Wavelets: Haar y Daubechies*. PhD thesis, INSTITUTO POLITÉCNICO NACIONAL ,ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA, 2015.

[31] L Martínez. Computación paralela de la transformada wavelet; aplicaciones de la transformada wavelet al álgebra lineal numérica. *Universidad Politécnica de valencia, España*, 2009.

[32] David Salomon. *Data compression: the complete reference*. Springer, 2007.

[33] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[34] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5):36–58, 2001.

[35] Frédéric Dufaux, Gary J Sullivan, and Touradj Ebrahimi. The jpeg xr image coding standard [standards in a nutshell]. *IEEE Signal Processing Magazine*, 26(6):195–204, 2009.

[36] Nai Bian, Feng Liang, Haisheng Fu, and Bo Lei. A deep image compression framework for face recognition. In *2019 2nd China Symposium on Cognitive Computing and Hybrid Intelligence (CCHI)*, pages 99–104. IEEE, 2019.

[37] Idan Ram, Israel Cohen, and Michael Elad. Facial image compression using patch-ordering-based adaptive wavelet transform. *IEEE Signal Processing Letters*, 21(10):1270–1274, 2014.

[38] Michael Elad, Roman Goldenberg, and Ron Kimmel. Low bit-rate compression of facial images. *IEEE Transactions on Image Processing*, 16(9):2379–2383, 2007.

[39] Joaquin Zepeda, Christine Guillemot, and Ewa Kijak. Image compression using sparse representations and the iteration-tuned and aligned dictionary. *IEEE Journal of Selected Topics in Signal Processing*, 5(5):1061–1073, 2011.

[40] A. Alcocer Ochoa. *Codificación Conjunta de Fuente y Canal para el ITU T G.729 CS ACELP aplicando Turbo Códigos*. PhD thesis, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2003.

[41] S. Haykin. *Digital Communication Systems*. Wiley, 2013.

[42] J.G. Proakis and M. Salehi. *Digital Communications*. McGraw-Hill International Edition. McGraw-Hill, 2008.

[43] Hao Zhong and Tong Zhang. Block-ldpc: A practical ldpc coding system design approach. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 52(4):766–775, 2005.

[44] Bernard Sklar et al. *Digital communications: fundamentals and applications*. 2001.

[45] Norbert Goertz. *Joint source-channel coding of discrete-time signals with continuous amplitudes*, volume 1. World Scientific, 2007.

[46] Burt Masnick and Jack Wolf. On linear unequal error protection codes. *IEEE Transactions on Information Theory*, 13(4):600–607, 1967.

[47] Joachim Hagenauer. Source-controlled channel decoding. *IEEE Transactions on Communications*, 43(9):2449–2457, 1995.

[48] Khalid Sayood and Jay C Borkenhagen. Use of residual redundancy in the design of joint source/channel coders. *IEEE Transactions on Communications*, 39(6):838–846, 1991.

[49] N Gortz. On the iterative approximation of optimal joint source-channel decoding. *IEEE Journal on Selected Areas in Communications*, 19(9):1662–1670, 2001.

[50] Fernando Ruiz Vera. Aplicación de la transformada wavelet en compresión de imágenes. *1er Congreso Internacional de Ingenierías Eléctrica, Electrónica y de Sistemas. Universidad Industrial de Santander. Bucaramanga, Colombia. Ponencia estudiantil*, 2000.

[51] M. Domínguez Jiménez and G. Sansigre Vidal. La transformada wavelet: una introducción. *Apuntes de la Asignatura: Transformada wavelet y aplicaciones en Ingeniería del Doctorado en Matemáticas Aplicadas. Universidad Politécnica de Madrid*, 2005.

[52] L. Erik. Ee49 lab 1: Source coding lab: Cosine transform (dct), sample quantization, and huffman coding, 2016.

[53] Pintos Castro y Boris Ramos Sánchez Freddy Manuel, Orosco Villamagua y Claudia Sofía. Diseño, simulación e implementación de códigos de canal en sistemas ofdm. 2014.

[54] RW Heath Jr. Digital communications, physical layer exploration lab using the ni usrp platform, 2012.

Índice de figuras

Índice de tablas