UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

FACULTAD DE INGENIERÍA

IMPLEMENTACIÓN DE UN CODIFICADOR-DECODIFICADOR DE VIDEO MJPEG XR EN UN DSP

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERÍA

INGENIERÍA ELÉCTRICA - PROCESAMIENTO DIGITAL DE SEÑALES

P R E S E N T A:

ING. JORGE ARMANDO RODRÍGUEZ VERA

TUTOR:
M. I. LARRY HIPÓLITO ESCOBAR SALGUERO

2012

Jurado asignado:

Presidente:	Dr. García Ugalde Francisco Javier
Secretario:	Dra. Medina Gómez Lucía
1^{er} Vocal:	M.I. Escobar Salguero Larry
1^{er} Suplente:	Dr. Pérez Alcázar Pablo Roberto
2^{o} Suplente:	Dr. Psenicka Bohumil

Lugar donde se realizó la tesis:

LABORATORIO DE PROCESAMIENTO DIGITAL DE SEÑALES

POSGRADO DE INGENIERÍA- UNAM

TUTOR DE TESIS:

M.I. Larry Hipólito Escobar Salguero

FIRMA

Resumen

El objetivo de este trabajo es implementar un codificador y decodificador MJPEG XR basado en el novedoso estándar JPEG XR, que fue aprobado en el año 2009 por la International Telecomunication Union. Para tal propósito, se abordan algunos conceptos necesarios para entender el tratamiento de imágenes y video, se hace una descripción de los estándares de video actuales así como de los métodos de compresión comunmente utilizados. El codificador MJPEG XR es descrito ampliamente y su implementación es realizada en un DSP, marca Texas Instruments, con alto desempeño, de la familia C6000, que tiene como propósito principal procesar imágenes y video. Los algoritmos utilizados durante la implementación de estr trabajo fueron escritos en lenguaje C. La evaluación del codificador MJPEG XR se realizó obteniendo secuencias de video utilizando un sistema de adquisición previo, mostrando tanto el desempeño de la calidad visual como de la cantidad de información que logra comprimir. Para este propósito se utilizan herramientas de software como el Code Composer Studio de Texas Instruments y Matlab de Mathworks. Finalmente, se hacen algunas comparaciones entre el compresor MJPEG, desarrollado con anterioridad, y el MJPEG XR.

Índice general

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

Desde la aparición de internet, la popularidad e incremento de usuarios de la red ha creado la necesidad de aumentar el ancho de banda para la transmisión de datos o de crear dispositivos que puedan almacenar esta inmensa cantidad de información. El video es una señal con gran cantidad de información que requiere de una gran cantidad de espacio para almacenamiento o de gran ancho de banda para su transmisión. Sin embargo, el aumento del ancho de banda o de almacenamiento en la mayoría de casos no es posible y se tiene que optar por otra forma de cubrir esta necesidad; una alternativa práctica es el uso de la compresión, ya que permite la reducción de la información. Si la información a tratar se repite un número determinado de veces en una señal se dice que hay *redundancia* y por lo tanto el objetivo de la compresión es disminuirla. Especificamente, en video, tema principal de la presente tesis, existen diferentes técnicas para reducir la redundancia que se encuentra entre los pixeles vecinos, entre los diferentes planos de color y a su vez entre las diferentes secuencias de imágenes. Estas técnicas son abordadas y utilizadas en los estándares que engloban a cada una de ellas. Las imágenes son las componentes fundamentales de una secuencia de video y por tal motivo algunos grupos de trabajo como el Joint Photographic Experts Group (JPEG), se han dedicado a producir estándares en junto con las tres mayores organizaciones de estandarización: la International Organization for Standardization (ISO), la International Electrotechnical Comission (IEC), y la International Telecomunication Union (ITU-T). JPEG es mundialmente reconocido como el comité lider para los formatos de compresión de imágenes y es responsable de las familias más populares de estándares de imágenes como JPEG y JPEG 2000.

El estándar JPEG surgió por la necesidad de cubrir algunas exigencias tecnológicas como los avances en las aplicaciones multimedia que requieren una alta relación de compresión y una gran calidad visual. La alta calidad, la alta tasa de compresión de una imagen digital y el bajo costo computacional son factores importantes en muchas áreas de consumo electrónico, que van desde la fotografía digital hasta equipos de visualización tales como la cámara digital y los marcos digitales. Estos requerimientos usualmente envuelven algoritmos computacionalmente intensos imponiendo ventajas y desventajas entre calidad, recursos computacionales y rendimiento.

JPEG se ha convertido en uno de los estándares más usados en el mundo y a sus veinte años de haberse iniciado la tecnología está llegando a sus límites, comenzando a estar por debajo

del desarrollo de características inovadoras y tendencias en la fotografía digital actual. Recientemente el comité JPEG ha producido el estándar JPEG 2000, introduciendo novedosas funcionalidades. Sin embargo, tiene una notable demanda de recursos computacionales y no ha logrado un gran impacto en móviles y en ambientes embebidos como la fotografía digital de consumo. Fotógrafos serios han optado por usar el codificador de imágenes *raw* por las limitaciones que han encontrado en el estándar JPEG, pero el codificador *raw* tiene una alta demanda de espacio y es generalmente para una cámara específica [8] [19]. JPEG XR es un nuevo estándar de codificación de imágenes, aprobado en 2009, diseñado para cubrir las limitaciones de los formatos de hoy y simultáneamente mejorar la alta calidad de la imagen mientras disminuye la demanda de recursos computacionales y capacidad de almacenamiento. La abreviatura XR, que forma parte del nombre JPEG XR, hace referencia al *intervalo extendido* de aplicaciones. En el caso de una secuencia de video, se utiliza el codificador MJPEG XR [8].

Justificación

Los *Procesadores Digitales de Señales* (DSP's) son muy utilizados en la industria de la telefonía y la computación. En el laboratorio de procesamiento digital de señales, del departamento de procesamiento de señales, perteneciente a la División de Ingeniería Eléctrica, de la Facultad de Ingeniería en la UNAM se cuenta con una variedad de DSP's, en los cuales se realizan pruebas e implementan algoritmos que buscan mejorar el rendimiento de algunas soluciones que ya se encuentren en el mercado.
Esencialmente en este trabajo se hace uso del nuevo compresor MJPEG XR para manejar una secuencia de cuadros y obtener un video, de duración determinada, con una buena calidad y tasa de compresión.

Objetivo

Analizar e implementar el compresor MJPEG XR, con base en el nuevo estándar JPEG XR, en un DSP de la famila C6000 de Texas Instruments (TI) y aplicarlo a una secuencia de video, para su codificación y decodificación.

Organización del trabajo

En el capítulo 1 se da una introducción, acotando la línea de desarrollo de la tesis y planteando su objetivo así como la metodología a seguir. El capítulo 2 contiene los conceptos básicos necesarios para entender el video digital. El capítulo 3 aborda diversas técnicas conocidas sobre compresión de video digital haciendo énfasis en aquellas que son utilizadas en la tesis. El capítulo 4 describe cada una de las etapas que forman el estándar MJPEG XR. El capítulo 5 contiene una descripción de los elementos del DSP que fueron necesarios para poder utilizar el estándar MJPEG XR. El capítulo 6 tiene la medición de los recursos del

DSP ocupados por el estándar MJPEG XR así como un análisis de los resultados en cuanto a calidad visual y compresión. Posteriormente se proporcionan las conclusiones y un resumen más concreto de los resultados obtenidos. Al final se tiene un apéndice de abreviaturas y anexos.

Capítulo 2

Generalidades de video digital

Las imágenes digitales son componentes elementales de una secuencia de video, digital. Por tal motivo, es necesario hacer una descripción de los espacios de color, del procesamiento de video así como de algunos estándares y formatos existentes, que son los conceptos fundamentales para realizar el desarrollo e implementación del compresor MJPEG XR.

2.1. Imágenes digitales

Una *imagen* es una fotografía de una escena en un instante de tiempo, mientras que una *secuencia de video* representa una escena sobre un periodo de tiempo. Una forma muy común para describir una imagen I es representarla como una matriz rectangular llamada matriz de imagen como se muestra en la ecuación (2.1).

$$I = [s(x, y)] \tag{2.1}$$

Lo anterior quiere decir que el valor de una fila, junto con el valor de una columna, define una pequeña área de la imagen llamada *pixel* (por las siglas *Picture Element*), el cual contiene un valor que representa la intensidad o nivel del pixel [19]. En la figura 2.1 se observa una sección amplificada de una imagen, donde se aprecia un conjunto de pixeles que forman parte de ésta.

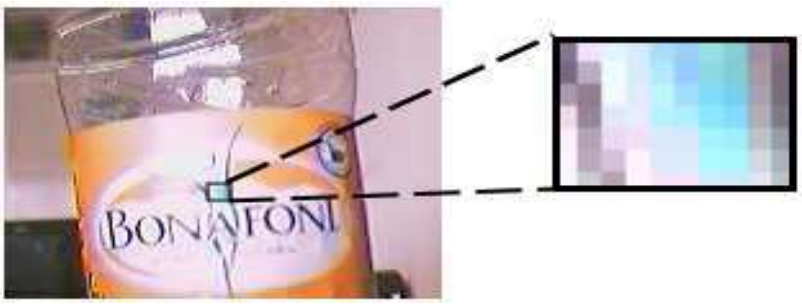


Figura 2.1: Pixeles como elementos de una imagen

La *profundidad de bits* es el número de bits que se le asigna a un pixel para representar la información de la imagen. Esto quiere decir que con una mayor profundidad de bits la imagen contendrá más información y por lo tanto se tiene un mayor número de tonos y colores.

- Si se tiene una profundidad de un solo bit, solo se pueden tener dos *niveles* o *tonos*. En este caso tenemos una *imagen binaria* que consta únicamente de valores 0 y 1 (blanco y negro).
- Si se tiene una profundidad de ocho bits, es posible tener 256 *niveles* o *tonos*.

Los niveles o tonos que tiene una imagen se obtienen mediante la relación 2^L, siendo L el número de profundidad de bits. Lo anterior quiere decir que entre mayor sea la profundidad de bits mejor será la aproximación a un tono real. Una imagen digital tiene diferentes representaciones; por mencionar una, cualquier imagen tiene una representación en escala de grises, donde el valor que tiene un pixel está en el intervalo de 0 a 255 [26], donde el 0 representa al negro y 255 al blanco.

2.2. Imágenes a color

Una imagen de color es una matriz formada por *pixeles de color* y mantiene un tamaño de N x M x 3, donde cada pixel de color es una terna de valores que corresponde a los componentes rojo, verde y azul en una localización espacial específica como se aprecia en la figura 2.2.
Cuando una imagen se despliega sobre la pantalla, cada elemento es mapeado a un pixel único de la pantalla. La combinación de los valores en las componentes R (Red), G (Green), B (Blue) dan como resultado al color final de pixel en la imagen.

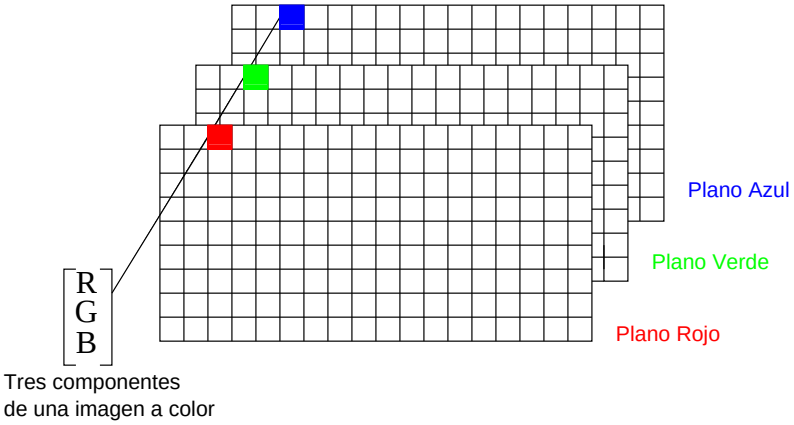


Figura 2.2: Componentes de una imagen de color RGB

2.3. Espacios de color

El color se define como una reacción cerebral a los estímulos visuales, que lo hace extremadamente subjetivo y personal [9]. Por tal motivo, tratar de atribuirle números es muy

complicado. Un *espacio de color* es un método por el cual se puede especificar, crear y visualizar el color. Los espacios de color pueden ser clasificados en función de la linealidad de la percepción, la intuición y la dependencia del dispositivo. Los diferentes espacios de color son seleccionados de acuerdo a la aplicación con que se relacione. Por ejemplo, en algunos equipos existen factores que dictan el tamaño y el tipo de espacio de color que puede ser utilizado.

2.3.1. Espacio RGB

El espacio RGB (Red, Green, Blue) es el más comunmente utilizado en cada sistema de cómputo, televisión y video [9] [26]. En la figura 2.3 se muestra un cubo formado por el espacio de coordenadas— RGB. Dado que cada componente tiene la misma importancia para obtener la representación de color final en la imagen, los sistemas RGB usualmente representan a cada componente con la misma precisión. Por ejemplo, si cada componente utiliza 8 bits, en total se tienen 3x8=24 bits para representar un pixel. En la figura 2.4 se muestra una imagen con sus respectivas componentes R, G, y B.

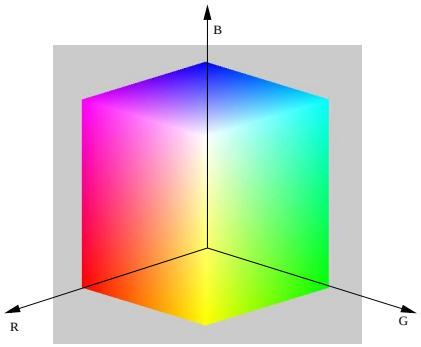


Figura 2.3: Espacio de color RGB

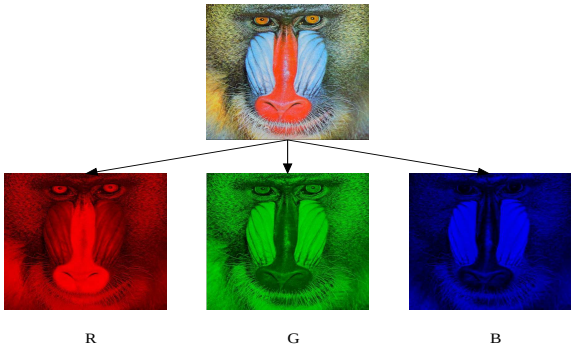


Figura 2.4: Componentes R, G y B de una imagen

Debido a que este espacio de color mantiene una alta correlación entre las tres componentes R, G y B, un significativo porcentaje de no linealidad en la percepción visual, una dependencia al dispositivo de despliegue y una mezcla de la información de luminancia, entonces la representación RGB no es una elección favorable para el procesamiento de imágenes. Cabe destacar que el costo computacional es considerable. Lo anterior da origen a la busqueda de otros espacios de color para ser utilizado en el procesamiento de imágenes [9].

2.3.2. Espacio de color YUV

Las imágenes a color son usualmente transformadas de RGB a una representación de luminancia-crominancia. Cada vector de pixeles $f(n)$ es transformado por una matriz de transformación reversible a un vector equivalente de luminancia-crominancia $g(n)$, como se muestra en la figura 2.5. Para el caso concreto del espacio YUV, la relación existente entre

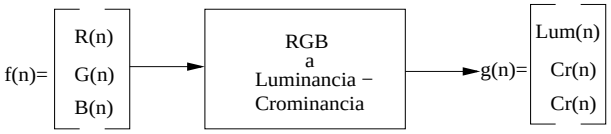


Figura 2.5: Transformación de espacio RGB a Luminancia-Crominancia

el espacio RGB y YUV está dada por la ecuación (2.2).

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.613 & -0.515 & 0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.2}$$

El propósito de la transformación es eliminar la correlación existente entre las bandas espectrales visuales debido a que pueden ser tratadas de forma separada. En la figura 2.6 se muestra el resultado de haber pasado del espacio RGB a YUV y sus diferencias.

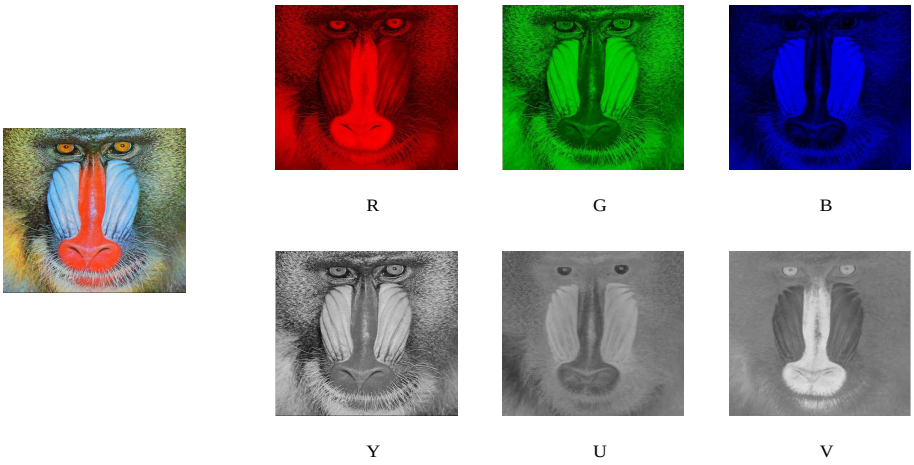


Figura 2.6: Transformación RGB a YUV

Es importante mencionar que los espacios de color luminancia-crominancia son los más utilizados en los estándares de compresión debido a que favorecen este proceso.

2.4. Procesamiento de video

La señal de video es interpretada en su forma analógica de manera similar a como lo haría el sentido de la vista. Para poder ser tratada por dispositivos digitales es necesario convertirla a un formato adecuado, lo cual es realizado mediante un *codec*, que es un dispositivo o programa capaz de codificar y decodificar una señal de video.
El procesamiento global de una señal de video, desde su adquisición por una cámara hasta su envío a la red para ser almacenado o visualizado en el otro extremo, se muestra en la figura 2.7, en la cual se observa que la señal obtenida de la cámara de video es adquirida por un

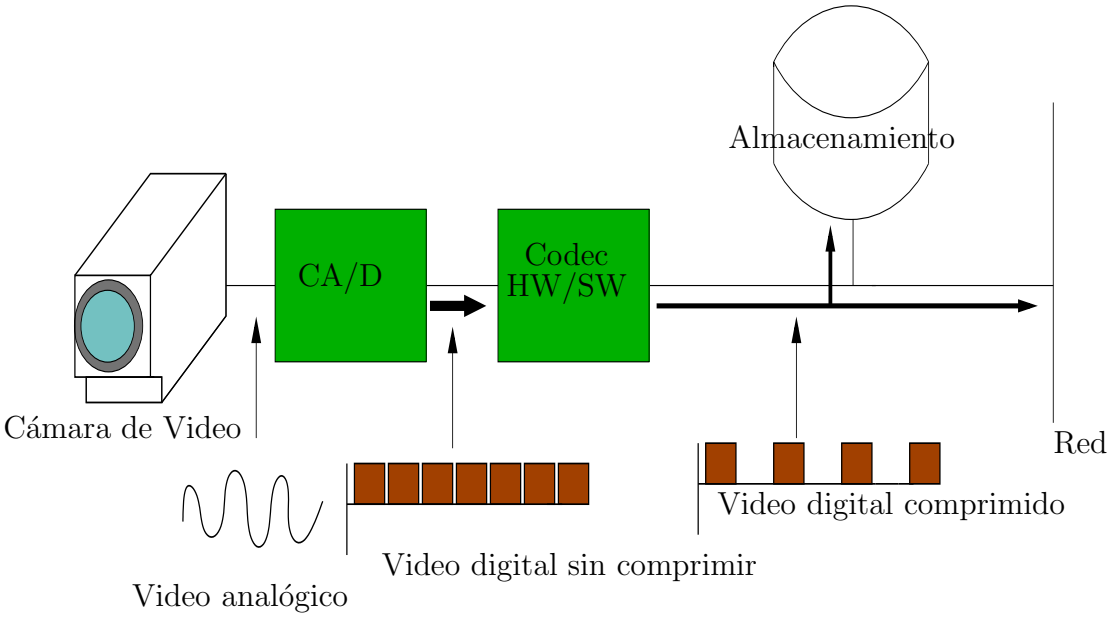


Figura 2.7: Proceso General de digitalización de Video

dispositivo CA/D (Convertidor Analógico Digital) que se encarga de digitalizar la señal de video para posteriormente ser procesada y comprimida por un codec (que puede ser basado en Software o Hardware) además de añadirle un retardo. El formato de video resultante tiene la posibilidad de ser almacenado o enviado por la red para posteriormente ser reproducido.

2.4.1. Cámara de video

El ojo humano capta luz y color que tiene una longitud de onda comprendido entre los 400 y 700 nanómetros, por encima y por debajo de estas longitudes de onda, las señales son invisibles para el ojo humano. El funcionamiento de una cámara de video en general trata de emular el funcionamiento del ojo humano, existiendo similitudes entre los componentes de una cámara y las partes de un ojo, como se aprecia en la figura 2.8.

COMPONENTES CÁMARA	PARTES DEL OJO
Lente	Cristalino, córnea
Objetivo, disparador	Iris, pupila
Película	Retina
Cable para transferir imágenes	Nervio óptico

Figura 2.8: Similitud entre Cámara y Ojo humano

2.4.2. Luz y Color. Luminancia y Crominancia

En esta sección se hace énfasis en algunos conceptos mencionados anteriormente pero que no han sido definidos.
En el ojo humano, la retina contiene fotorreceptores, de dos tipos:

- Conos: captan el color, existen conos más sensibles al rojo, al verde y al azul.
- Bastones: captan la intensidad de la luz, no detectan el color.

La sensación de percepción del color se divide en dos factores principales:

- Luminancia (brillo, intensidad).
- Crominancia (aspectos del color).
  - Tono o matiz del color.
  - Profundidad o pureza del color.

Así, una señal de video capturada por una cámara es descompuesta, mediante técnicas de filtros, en tres señales, correspondientes a cada uno de los tres colores primarios. Estas señales son sometidas a diversos procesos según la aplicación.

2.4.3. La televisión Analógica

En la televisión analógica, las señales RGB obtenidas de los dispositivos sensores se transforman en señales basadas en luminancia y crominancia. Esto se diseñó principalmente por dos motivos principales:

- Para mantener la compatibilidad entre la televisión a color y la televisión en blanco y negro (en esta última se ignora la información de crominancia).

- Para un mejor aprovechamiento de ancho de banda debido a que R, G y B son redundantes. Se transforma el modelo RGB en YCr(n1)Cr(n2), parámetros no correlacionados, de manera que se requiere menor ancho de banda de transmisión para Cr(n1) y Cr(n2) (información de crominancia) que para Y (información de luminancia).

Con los avances de la televisión en la segunda mitad del siglo XX se establecieron diferentes sistemas estandarizados, cada uno con características propias de tratamiento y presentación de las señales de video. Concretamente, los tres sistemas principales de televisión existentes en el mundo son:

- NTSC (1953): *National Television Systems Committee*, implantado en Estados Unidos y Japón.
- PAL (1967) *Phase Alternate Line*, implantado principalmente en Alemania.
- SECAM (1967): *Sequentiel Couleur Avec Memoire*, implantado principalmente en Francia y Rusia.

Estos sistemas de televisión a su vez presentan variantes, que difieren en algunas características, tales como la frecuencia de portadora o algunos parámetros de modulación de la señal. En la tabla 2.1 se muestra una comparativa de las características principalmente de NETSC, PAL y SECAM (es una variante de PAL, cuya principal diferencia está en la modulación de las señales portadoras de información) y NTSC.
Todos los sistemas de televisión anteriores emplean el modo entrelazado: cada trama está for-

CARACTERÍSTICAS DE LOS ESTÁNDARES	NTSC	PAL	SECAM
TRAMAS/s	29.97	25	25
LÍNEAS/TRAMA	525	625	625
LÍNEAS/s	15,734	15,625	15,625
ESPACIO DE COLOR	YIQ	YUV	YDbYDr
ANCHO DE BANDA LUMINANCIA (MHz)	4.2	5.0	6
ANCHO DE BANDA CROMINANCIA (MHz)	1.5	1.3	1

Tabla 2.1: Comparación de NTSC, PAL y SECAM

mada por dos campos, conteniendo cada uno de ellos las filas pares o las filas impares. En ocasiones este método es más sencillo de implementar para efectos de presentación en pantalla ya que el ojo humano no es capaz de detectar las diferencias entre las imágenes por segundo y campos por segundo (no detecta la falta de líneas intermedias en cada campo enviado).
Las imágenes de video analógico son almacenadas en formato *raster* (unidimensional). Cada trama o imagen está compuesto por un número de líneas, formándose señales unidimensionales consistentes en líneas escaneadas sucesivamente en la imagen.
En la figura 2.9 se muestra la diferencia entre modo entrelazado (*interlaced*, basado en campos) y modo progresivo (*progressive*, basado en tramas).

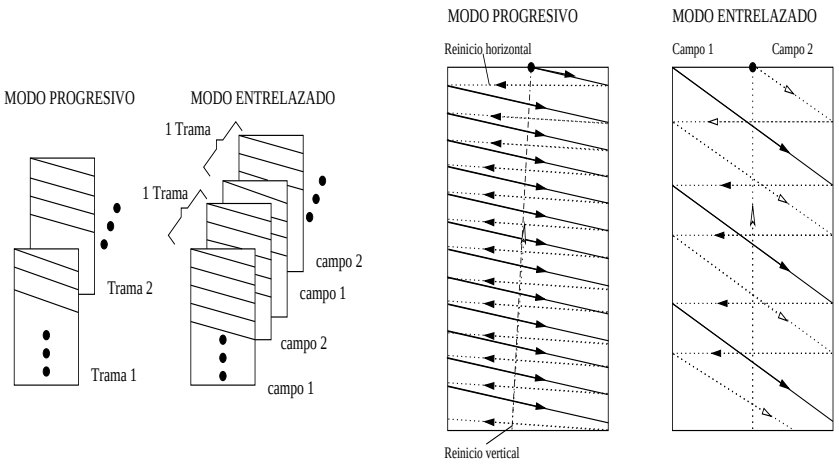


Figura 2.9: Modo Progresivo y Modo entrelazado

2.4.4. Digitalización de señales de video

En 1982, en el seno de CCIR (actual ITU-R) nace la recomendación 601 (actual recomendación BT.601), que define la digitalización de las señales analógicas de video para televisión. Se decide obtener un formato digital común a PAL/SECAM y a NTSC, por lo que se tienen ambos sistemas para la definición del estándar.

2.4.5. Resoluciones de pantalla

En el proceso de digitalización se considera, en ambos casos, un área activa, el resto del área no se considera válida para efectos de transmitir información, y este tiempo se emplea para dar tiempo al haz de electrones a recorrer la pantalla y para transmitir información adicional, como el servicio de teletexto. El área activa considerada en ambos casos se muestra en la figura 2.10 .

Por tanto, el área activa considerada, es de 720x480 en NTSC y de 720x576 en PAL/SECAM. El espacio de color utilizado se denomina YCrCb, y es un modelo escalado de YUV. La ecuación (2.3) describe al espacio de color YCrCb en términos de los componentes R, G y B.

$$\begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & 0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.3}$$

Las componentes Cb y Cr son las crominancias azul y roja, y son codificadas en formato bipolar, con valores entre 16 y 240 con el cero correspondiendo al nivel 128. Con objeto de reducir el ancho de banda necesario para la transmisión de estas señales, se emplean técnicas de submuestreo. Estas técnicas se basan en la menor sensibilidad del ojo humano a la crominancia respecto de la luminancia para descartar información sin que el ojo humano lo perciba. En la figura 2.11 se muestran algunas de ellas de manera esquemática.

De la figura 2.11 se observa cómo la modalidad 4:4:4 implica no aplicar submuestreo: por

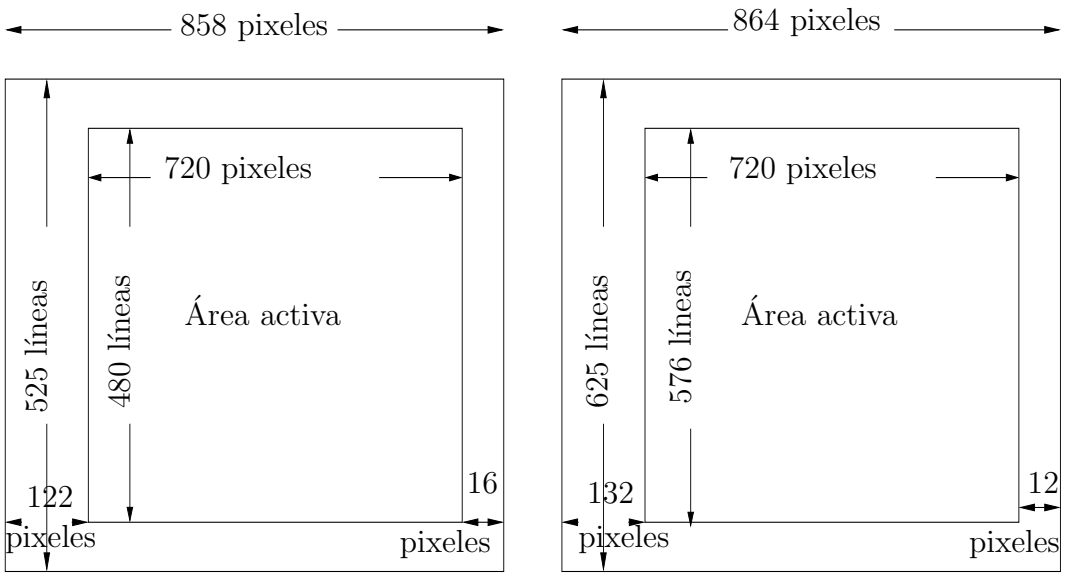


Figura 2.10: Digitalización PAL/SECAM y NTSC

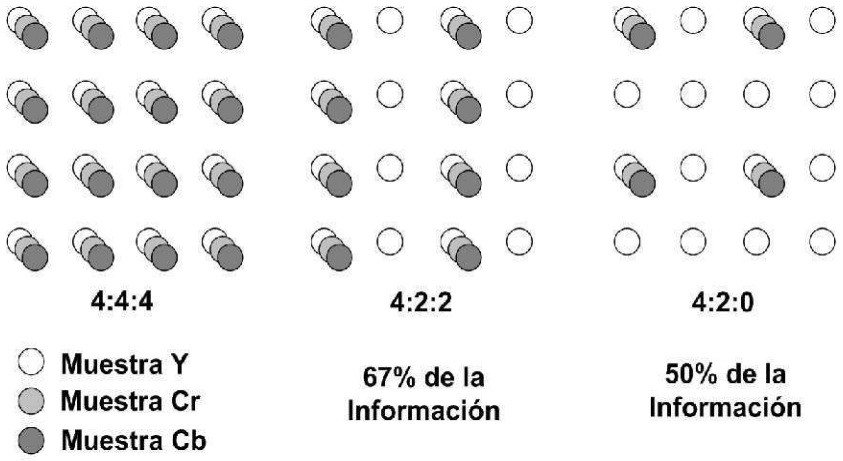


Figura 2.11: Técnicas de submuestreo

cada 4 componentes de luminancia, se envían 4 de crominancia azul y otras 4 de crominancia roja. Sin embargo, en las modalidades 4:2:2 y 4:2:0 no se envían determinadas componentes de Cb y Cr, disminuyendo por tanto el ancho de banda requerido para transmitir imágenes [16].

En la recomendación ITU-R BT.601 se utiliza 4:2:2, esto significa que por cada 4 pixeles con componente Y, sólo dos de ellos se transmiten con componente Cb y Cr. Si no se aplicase submuestreo, de cada 4 pixeles se transmiten sus 3 componentes Y, Cb y Cr, con lo que se transmiten 12 componentes, a 8 bits/componente, y por tanto a 24 bits/pixel.

Si se aplica 4:2:2, de cada 4 pixeles hay 2 que no llevan componentes Cb y Cr, por lo que en total se transmiten 8 componentes (4 de Y, 2 de Cb, 2 de Cr) en lugar de 12. Esto reduce a 64 el número de bits necesarios cada 4 pixeles, lo que en promedio supone 16 bits/pixel.

Aplicando estos datos a PAL/SECAM o NTSC se tiene:

$$30 \text{ x } 720 \text{ x } 480 \text{ x } 24 = 25 \text{ x } 720 \text{ x } 576 \text{ x } 24 = 248,832 Mbps$$

- Sin submuestreo:
  $25\frac{tramas}{s}$ x $864\frac{pixeles}{línea}$ x $625\frac{líneas}{trama}$ x $24\frac{bits}{pixel}$ = 29.97x858x525x24 = $324 Mbps$
- Con submuestreo 4:2:2:
  $25\frac{tramas}{s}$ x $864\frac{pixeles}{línea}$ x $625\frac{líneas}{trama}$ x $16\frac{bits}{pixel}$ = 29.97x858x525x16 = $216 Mbps$

Se observa cómo este ancho de banda necesita reducirse para poder transmitirse o almacenarse. Para ello se emplean otros tipos de técnicas como la reducción de la resolución.
De esta forma, a raíz de ITU-R BT.601, los estándares MPEG definen el formato SIF (Source Input Format) [16], cuyas características son las siguientes:

- Mantiene la tasa de transmisión de NTSC (29.97) y PAL (25).
- Reduce a la mitad la resolución del área activa: 360x240 en NTSC, 360x288 en PAL/SECAM.
- Aplica submuestreo 4:2:0.

Dado que los algoritmos de codificación de MPEG trabajan con macrobloques de 16x16 pixeles y 360 no es múltiplo de 16, se utilizan más las siguientes versiones:

- NTSC: 352x240 pixeles; 29.97 tramas/s; submuestreo 4:2:0.
- PAL: 352x288 pixeles; 25 tramas/s; submuestreo 4:2:0.

Para realizar lo anterior, se eliminan 4 columnas izquierdas y las 4 columnas derechas de cada trama. Las relación de aspecto en SIF es de 4:3. Para mantener esta relación en pantalla, los pixeles son rectangulares.
En la industria de las computadoras personales, sin embargo, SIF se ha redefinido, ya que el monitor de una computadora personal sí ofrece pixeles cuadrados. Para este caso, SIF equivale a 320x240 de QVGA (Quarter Video Graphics Array) en NTSC y a 384x288 en PAL/SECAM, manteniendose así la relación de aspecto 4:3.
Se han definido además diversas variantes SIF, según la aplicación:

- QSIF (Quarter SIF):176x120 (NTSC), 176x144 (PAL/SECAM); en computadoras personales, 160x120 (NTSC), 192x144 (PAL/SECAM).
- 2SIF:352x480.

Este formato SIF, procedente a su vez de las versiones digitales de NTSC y PAL/SECAM, ha sido la base para la definición de resoluciones estándares que son utilizadas por ejemplo en videoconferencias. De este modo, en 1990 surge la familia de estándares H.320 para videoconferencia sobre RDSI (Red Digital de Servicios Integrados), con el estándar H.261 para la codificación de video. En esta recomendación se establecen dos formatos de resoluciones para videoconferencia [16].

- CIF (Common Intermediate Format), opcional:
  - 352x288, tramas progresivas, submuestreo 4:2:0, velocidades 7, 5, 10, 15, 20 y 29.97 tramas/s.
- QCIF (Quarter CIF), obligatorio:
  - Mismas características que CIF, pero con resolución 175x144.

Estas dos posibilidades se ven ampliadas en los codecs posteriores. En H.263, H.263++ y H.264 se definen y utilizan diversas resolucines adicionales:

- SQCIF (Sub QCIF): 128x96.
- 4CIF (Super CIF): 740x576.
- 16CIF: 1408x1152.

Todos ellos con las características:

- Tramas progresivas.
- 29.97 tramas/s y derivados.
- Submuestreo 4:2:0.
- Codificación en YCbCr.

Los pixeles deben ser rectangulares para mantener la relación 4:3. En las computadoras personales, es necesario hacer una conversión, ya que los pixeles son cuadrados.

2.4.6. Televisión de Alta Definición

La primera estandarización del formato de video de alta definición aparece en la recomendación ITU-R BT.709. Ésta especifica las características de las señales de alta definición, para su captura y transferencia. Se compone de dos partes, una primera -ya obsoleta- que codifica sistemas de 1125 y 1250 líneas, y una segunda codifica sistemas basados en 1080 líneas verticales. En este caso se especifícan resoluciones 1920x1080, con pixeles cuadrados, para una relación de aspecto de 16:9 [16].
El concepto de televisión de alta definición ha sido adoptado por diferentes estándares, como ATSC (Advanced Televisión Standards Committee) o DVB (Digital Video Broadcasting). De acuerdo a ATSC, la televisión digital agrupa 18 formatos específicos. Algunos de estos estándares se refieren a los de SDTV (Standard Definition TV) y HDTV (High Definition TV).
ATSC establece la resolución vertical de cada uno de estos modos, la notación para referir diferentes formatos de video en este formato sigue la norma:
Resolución horizontal (opcional) x Resolución vertical + Modo tramas + Velocidad de trama (opcional). Lo anterior, permite definir en la televisión digital tradicional (SDTV) los dos estándares siguientes:

- 480i: 704x480 entrelazado, 30 tramas/s.
- 480p: 704x480 progresivo, 60 tramas/s.

Para HDTV se definen los tres estándares siguientes:

- 720p: 1280x720 progresivo, 60 tramas/s.
- 1080i:1920x1080 entrelazado, 30 tramas/s.
- 1080i: 1920x1080 progresivo, 60 tramas/s.

Se dice que un dispositivo está preparado para HD (HD ready) si es capaz de mostrar señales HDTV, y se dice que tiene HD integrado si incluye un codec de HD en su interior. En el caso de 1080p (es decir, 1980x1080 pixeles, modo progresivo, 60 tramas/s), se denomina también *Full HD*.
Los estándares HDTV tienen una relación de aspecto 16:9, en contraposición con la relación 4:3 de los estándares NTSC y PAL/SECAM. Por otro lado, en HDTV soportan las tasas de transmisión siguientes: 60, 59.94, 50, 30, 29.97, 25, 24 y 23.98.
En al figura 2.12 se muestra una gráfica comparativa de las resoluciones de los diferentes formatos, desde el inicial SQCIF de H.261, hasta 1080p de alta definición.

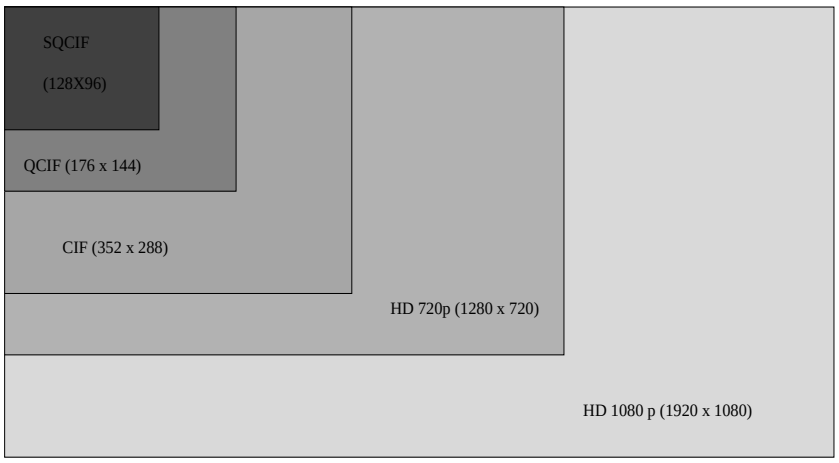


Figura 2.12: Resoluciones de formatos de video

Resumen

En este capítulo se abordaron los conceptos más generales y fundamentales que se requieren para entender la interpretación que se le da al video digital. El video está compuesto por una secuencia de imágenes que comparten características muy similares. En el espacio de color que ocupa la luminancia y crominancia se aprovecha la baja sensibilidad que tiene el ojo humano hacia la crominancia para disminuir el espacio que puede ocupar el video ya sea para almacenar o transmitir. Dependiendo de la aplicación y el interés que se tenga con el video se elige algun tipo de estándar, que rige tanto el tamaño o resolución que se desplegará en pantalla así como la cantidad de tramas de imagen que se envían y el tipo de submuestreo más adecuado.
En el presente trabajo se ocupa el formato QCIF para aplicarlo a imágenes y de esta forma obtener resultados que permitan probar la flexibilidad de MJPEG XR aplicado a una secuencia de imágenes.

Capítulo 3

Compresión de video digital

Los estándares existentes sobre compresión de video digital están compuestos por diferentes etapas, las cuales se encargan de reducir la redundancia que se encuentra en una imagen o entre una secuencia de imágenes.
Los tres tipos de algoritmos de compresión de video digital, que se pueden enunciar de acuerdo a la reducción de redundancia, son los siguientes [16]:

- Espacial: este algoritmo aprovecha principalmente el hecho de que el ojo humano no es capaz de distinguir pequeñas diferencias de color tan fácilmente como pequeñas diferencias en brillo. También, es importante destacar que la redundancia a explotar es únicamente dentro de una imagen.

- Temporal: el algoritmo aprovecha la redundancia de información existente entre imágenes consecutivas y se utiliza principalmente en el tratamiento de imágenes en movimiento, esto es, en los estándares de video. Este tipo de compresión lleva incluida la compresión espacial, debido a que es necesario comparar dentro de una sola imagen fija antes de comparar con imágenes consecutivas. El hecho en que se basa principalmente esta técnica, consiste en que para una secuencia en movimiento muchos de los pixeles de una imagen a otra no varían, codificandose sólo aquellos que hayan cambiado.

- Espectral: para el caso de este algoritmo, aprovecha la correlación existente entre diferentes aspectos de color o bandas de frecuencias.

De forma general, existen dos grandes grupos de técnicas de compresión de video digital [16]: sin pérdidas y con pérdidas. En los primeros, la descompresión tiene como objetivo recuperar exactamente la información original de la imagen de tal forma que el porcentaje de compresión es muy modesta, mientras que en los segundos el proceso de compresión permite una pérdida de información que no se recupera en la descompresión [5].

3.1. Técnicas de compresión sin pérdidas

Dentro de las técnicas de compresión sin pérdidas destacan las siguientes:

- Técnicas basadas en mapeos de bits.
- Técnicas basadas en modelos estadísticos para los datos de entrada (modelos predictivos).

Las técnicas basadas en mapeos de planos de bits utilizan codificación entrópica de longitud variable (VLC, Variable Lenght Coding). La codificación entrópica es un esquema de compresión sin pérdidas que es independiente de la naturaleza del medio y de los datos de entrada, siendo la entropía el número medio mínimo de bits requerido para codificar los símbolos de entrada.
Uno de los principales tipos de codificación entrópica asigna códigos a símbolos de tal forma que se relaciona la longitud de los códigos asignados con la probabilidad de aparición de los símbolos. De esta manera, los símbolos más frecuentes se transmiten con menos bits, produciéndose compresión sin pérdidas; las longitudes de los códigos pueden variar inversamente con la probabilidad de ocurrencia de los diferentes símbolos. La tasa de bits requerida para codificar estos símbolos es el inverso del logaritmo de probabilidad p, en base 2 (bits), por ejemplo $log_2 p$. La entropía se calcula mediante la ecuación (3.1) [15]:

$$H(x) = -\sum_{i=1}^{n} p_i log_2(p_i) \tag{3.1}$$

Dos de las técnicas más comunes para VLC son la codificación Huffman y la codificación aritmética, ambas utilizadas en codecs de video como MPEG y H.261 por dar un ejemplo. La codificación Huffman es el método VLC más utilizado y consiste en asignar un código de salida con un número de bits para cada símbolo igual a $-log_2 p$, donde p es la probabilidad del símbolo.
La codificación aritmética usa una escala en que los intervalos de codificación son números reales que están representados entre 0 y 1. Tiene dos modalidades para la distribución de probabilidades: fija (probabilidades asignadas previamente) y adaptiva (cambiantes en el tiempo).
Por su parte, las técnicas basadas en modelos predictivos elaboran modelos matemáticos para diseñar un predictor, tal que dadas unas imágenes de entrada, estimen cuál es el valor más probable de los pixeles de salida. Este predictor puede ser:

- Local: si el modelo emplea los mismos coeficientes para una imagen fija.
- Global: si el modelo emplea coeficientes que pueden variar entre una imagen y otra.
- Adaptivo: si el modelo emplea coeficientes que varían dentro de cada imagen.

3.2. Técnicas de compresión con pérdidas

En general, las técnicas de compresión con pérdidas consisten en buscar propiedades de la señal de entrada tales que al aplicar transformaciones sobre la misma se obtenga una señal parecida y mucho más acorde para ser almacenada o transmitida por una red.
Algunos esquemas generales son:

- Cuantización de vectores.
- Codificación de subbandas.
- Aplicación de transformadas.

La compresión basada en *cuantización de vectores* implica la comparación de los vectores (grupos de símbolos) que llegan con otros previamente almacenados (que pertenecen a un diccionario), y la selección del más parecidos de éstos con el original. Este método es computacionalmente intensivo, por la elevada cantidad de comparaciones a realizar. Además, implica también un enorme conjunto de datos almacenados, para contemplar todos los posibles patrones de comparación. Por otro lado, si se transmite por una red es necesario agregar cabeceras para que en el destino puedan interpretar el vector si éste no es conocido. Por todo ello, no es utilizado en los codecs de transmisión de video por red.
La compresión mediante *subbandas* consiste en aplicar un escalamiento a una imagen, obteniendo versiones de la imagen más pequeñas a partir de la original, tratando cada una de ellas de manera independiente. Se utiliza submuestreo, y en la reconstrucción en destino se utiliza sobremuestreo. Para desarrollar la técnica descrita se utilizan las *wavelets* [13][16].
La teoría de subbandas se apoya en el concepto de multiresolución, de manera que distintas partes de la misma imagen pueden ser tratadas con diferentes resoluciones. En la figura 3.1 se muestra de forma general el funcionamiento de esta técnica. Dicha técnica tiene la dificultad para compaginar con la compensación del movimiento, por lo que no es muy empleada en la transmisión de video (excepto, por ejemplo, para comprimir imágenes fijas que no necesiten compensación de movimiento, como en el estándar JPEG 2000).
La técnica de compresión de imágenes mediante *transformadas*, consiste en la aplicación de operaciones matemáticas en el dominio del tiempo o la frecuencia sobre los datos de entrada para obtener unos datos de salida codificados de tal manera que ocupen mucho menor ancho de banda. El objetivo que se persigue es agrupar la mayor parte de la energía de la señal en el menor número de coeficientes, sean computacionalmente rápidas y sean independientes del contenido a comprimir.
Las dos transformadas más conocidas son la DFT (*Discrete Fourier Transform* y la DCT (*Discrete Cosine Transform*) [16], en ambas existen algoritmos para obtener resultados computacionalmente rápidos, aunque la primera trabaja con componentes reales e imaginarios (amplitud y fase), y la segunda únicamente con componentes reales. Una de las ventajas de la DCT es la propiedad de concentrar la mayor parte de la energía en pocos coeficientes.
Los estándares de imágenes y video más comunes (por ejemplo JPEG, MJPEG y MPEG-4) ocupan la DCT como base para reducir la redundancia espacial. Por tal motivo merece un estudio más detallado.

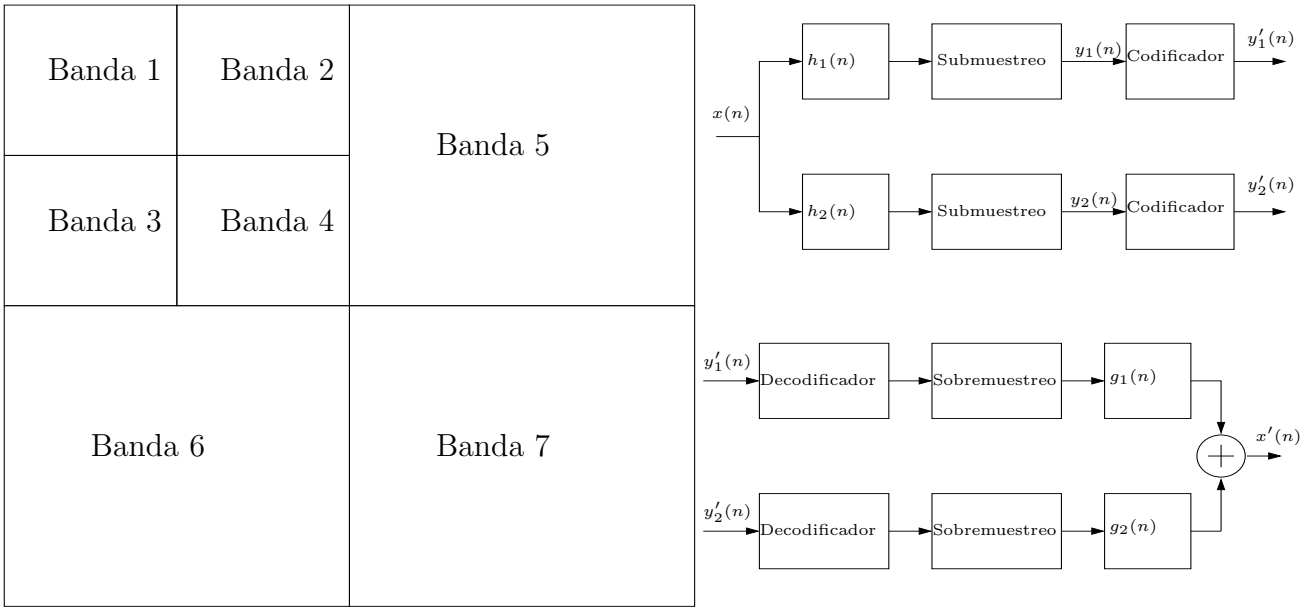


Figura 3.1: Compresión mediante subbandas

3.2.1. Transformada Coseno Discreta

En 1974 Ahmed, Natarajan y Rao propusieron la DCT [20], desde entonces, ésta ha sido la transformada más popular para la codificación de imágenes y video, y es una parte fundamental de en la compresión de imágenes JPEG [20]. Hay ocho variantes estándares, de las que la más común es la DCT tipo II, a menudo denominada como DCT. Su inversa, la DCT tipo III, es denominada DCT inversa o IDCT.
En los codecs de video se emplean DCTs (de tipo II) bidimensionales sobre bloques de 8x8 pixeles, para obtener coeficientes en el dominio de la frecuencia. El resultado es una matriz de 8x8 cuyo componente (0,0) es la componente continua de la señal (frecuencia nula o DC) y los valores para posiciones mayores en horizontal y vertical en la matriz representan valores más altos de frecuencia contenidos en la señal transformada y codificada.
La aplicación de la DCT y la IDCT implica una operación matemática que no genera pérdidas de información; su reconstrucción es exacta. Las pérdidas en los codecs de video se generan en la cuantización posterior de los coeficientes, y en los procesos asociados a la codificación de las imágenes de video.
En codecs de imágenes y video se aplica la DCT bidimensional a subbloques de la imagen de tamaño 8x8, y se aplica por separado a la crominancia y a la luminancia.
La ecuacion (3.2) describe a la DCT tipo II, en una dimensión mientras, que la inversa es la DCT tipo III que es descrita por la ecuación (3.3).

$$F(u) = \sqrt{\frac{2}{N}} C(u) \sum_{x=0}^{N-1} f(x) cos \left[\frac{(2x+1)u\pi}{2N} \right] u = 0, 1, ..., N-1 \tag{3.2}$$

$$f(x) = \sqrt{\frac{2}{N}} \sum_{x=0}^{N-1} C(u) F(u) cos \left[\frac{(2x+1)u\pi}{2N} \right] x = 0, 1, ..., N-1 \tag{3.3}$$

donde

$C(u) = \frac{1}{\sqrt{2}}$ para $u, v = 0$

$C(u) = 1$ para $u > 0$

$f(x)$ representa la intensidad del pixel x

$F(u)$ representa los N coeficientes de la transformada

La transformada coseno discreta puede extenderse de forma directa en dos dimensiones. Las ecuaciones (3.4) y (3.5) definen a la DCT directa y la IDCT en 2-D respectivamente para un bloque de 8x8.

$$F(u,v) = \frac{C(u)}{2}\frac{C(v)}{2}\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y) cos\left[\frac{(2x+1)u\pi}{16}\right]\left[\frac{(2y+1)v\pi}{16}\right] \tag{3.4}$$
$$u, v = 0, .., 7$$

$$f(x,y) = \sum_{u=0}^{7}\sum_{v=0}^{7}\frac{C(u)}{2}\frac{C(v)}{2}F(u,v) cos\left[\frac{(2x+1)u\pi}{16}\right] cos\left[\frac{(2y+1)v\pi}{16}\right] \tag{3.5}$$
$$u, v = 0, ..., 7$$

donde

$f(x, y)$ es el valor del pixel en la posición (x, y) dentro del bloque

$F(u, v)$ es el valor del coeficiente DCT (u, v)

$C(u)$, $C(v) = \frac{1}{\sqrt{2}}$ para $u, v = 0$

$C(u)$, $C(v) = 1$ para $u, v > 0$

La aplicación de la DCT es una técnica bien conocida en el manejo de imágenes y video, sin embargo, el rendimiento de la transformada comienza a disminuir significativamente a una baja tasa de bits. Los *artefactos* resultan de la codificación independiente de cada bloque de 8x8 y se manifiestan así mismos como discontinuidades en los bordes de cada bloque. Por tal motivo, se propuso la técnica LOT para resolver este problema [2].

3.2.2. Lapped Orthogonal Transform (LOT)

Casserau, Staelin, and Jager (1989) propusieron una transformada por solapamiento de bloques, llamada Lapped Orthogonal Transform (LOT), la cual utiliza los pixeles de los bloques vecinos para suavizar las discontinuidades en los bordes de cada bloque. Malvar y Staelin (1989) propusieron una nueva estructura de la LOT que utiliza las funciones base de la transformada coseno discreta [3][7][2].
La LOT es construida mediante el diseño de una transformada unitaria aplicada a una señal de

una dimensión. La transformada está caracterizada por el traslape de una matriz rectangular. Más específicamente, la matriz de transformación tiene la forma de la ecuación (3.6) [3].

$$T = \begin{bmatrix} P & & & 0 \\ & P & & \\ & & \ddots & \\ & & & P \\ 0 & & & P \end{bmatrix} \tag{3.6}$$

Donde P es una matriz de tamaño M x L. Ahora la relación entre la señal de entrada y su salida puede ser descrita mediante la ecuación (3.7) utilizando la matriz T

$$y = T'x \tag{3.7}$$

donde y representa el vector transformado de la señal, x representa al vector de la señal de entrada y ($'$) representa la transpuesta de la matriz de transformación. De forma similar la ecuación (3.8) permite obtener la señal de entrada a partir del vector transformado.

$$x = Ty \tag{3.8}$$

Para entender la manera en que se aplica la ecuación, suponiendo que se está examinando una imagen de 512 por 512 y se desea transformar una columna, se toma a P como una matriz de 8 x 16, con los bloques de entrada tomados de la imagen teniendo un tamaño de 16 x 1 y, por tal motivo, los bloques de salida serán de tamaño 8 x 1. Por lo tanto, el bloque DCT, contenido en P, estaría tomando cuatro puntos adicionales en cada lado del bloque vecino. En otras palabras, se sigue aplicando un bloque DCT para la misma cantidad de datos, pero ahora se toman datos vecinos. Por otro lado, la síntesis consiste en combinar los 8 vectores base con una longitud de 16 elementos transformados para obtener una porción de los bloques (la contribución de los bloques vecinos se utiliza para rellenar el resto). El solapamiento ayuda a disminuir el efecto de los artefactos de la DCT debido a que las fronteras del bloque ya no existen en la síntesis.
Imponiendo la condición que T debe ser una transformación unitaria, lleva a destacar algunas condiciones que tienen que ser satisfechas por P. Las columnas de P, las cuales representan las funciones base, deben ser ortogonales. Esta condición se puede expresar por medio de la ecuación (3.9).

$$P'P = I \tag{3.9}$$

y debido al solapamiento, las funciones de los bloques vecinos también deben ser ortogonales. La ecuación (3.10) muestra la relación anterior.

$$P_0 W P_0 = 0 \tag{3.10}$$

donde W es una matriz definida por la ecuación (3.11)

$$W = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \tag{3.11}$$

La LOT definida por Malvar, en la cual su matriz P cumple con las condiciones anteriormente impuestas es ortogonal [7]. La matriz P se define mediante la ecuación (3.12).

$$P = \begin{bmatrix} I_M & 0 \\ 0 & V_R \end{bmatrix} \begin{bmatrix} D_e - D_0 & J_{M/2}(D_e - D_o) \\ D_e - D_0 & -J_{M/2}(D_e - D_o) \end{bmatrix} \tag{3.12}$$

Donde M representa el tamaño del bloque de la DCT que se va a utilizar, D_e y D_o representan a las filas pares y a las filas impares de un bloque DCT de tipo III, respectivamente. El tamaño tanto de la matriz D_e y D_o es de $M/2$ x M, J es una matriz cuadrada de tamaño $M/2$ la cual se denomina *contra identidad* y se puede expresar mediante el ejemplo presentado en la ecuación (3.13).

$$J_4 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.13}$$

Para el ejemplo anterior el valor de M es 8 y por tal motivo el tamaño de la matriz J es $M/2$ x $M/2$=4x4. La matriz I de la ecuación 3.12 representa una matriz identidad de tamaño MxM. Mientras que la matriz V_R está definida por la ecuación (3.14)

$$V_R = D'_{IV} D' \tag{3.14}$$

donde D_{IV} es la matriz DCT tipo IV la cual tiene a cada componente definida por la ecuación (3.15).

$$d_{ij}^{IV} = \sqrt{\frac{2}{M}} cos\left(\frac{(2x+1)(2y+1)}{4M}\right) \tag{3.15}$$

y D corresponde a la DCT tipo III de la ecuación 3.3. En general dado que la LOT está basada en la DCT, sus funciones bases son muy parecidas entre sí pero con un ligero incremento en la amplitud para las funciones base de la LOT.
Aunque la LOT solventa muchos de los problemas en la DCT, tiene el inconveniente de que no decae suavemente a cero en los bordes. Lo anterior da como resultado la aparición de artefactos pero con mucho menor notoriedad que en las transformadas sin translape. Sin embargo, dado que se busca disminuir la aparición de discontinuidades dentro del proceso de reconstrucción de una imagen, se propuso otra técnica que suaviza el decaimiento en los bordes, llamada Lapped Biorthogonal Transform y será tratada en la siguiente sección.

3.2.3. Lapped Biorthogonal Transform (LBT)

Dado que la LOT no elimina del todo la aparición de los artefactos, Malvar propuso una modificación a la LOT, creando la Lapped Biorthogonal Transform (LBT) o transformada biortogonal solapada [27]. La matriz de la LBT es muy parecida a la LOT y es descrita por la ecuación (3.16).

$$P_{LBT} = \begin{bmatrix} I_M & 0 \\ 0 & V_R \end{bmatrix} \begin{bmatrix} D_e - \gamma D_o & J_{M/2}(D_e - \gamma D_o) \\ D_e - \gamma D_o & -J_{M/2}(D_e - \gamma D_o) \end{bmatrix} \tag{3.16}$$

donde γ es una matriz diagonal definida por $\gamma = diag\{\sqrt{2}, 1, 1, ..., 1\}$. Se debe notar que de acuerdo a la ecuación 3.16 solo una componente de cada DCT de salida es multiplicada por una constante. Al obtener la inversa de la matriz P_{LBT} quedaría, parecida a la ecuación 3.16 excepto que γ se modificaría quedando $\gamma = \{\frac{1}{\sqrt{2}}, 1, 1, ..., 1\}$. El efecto de este nuevo factor después de invertir la matriz P_{LBT} es una reducción en la amplitud de las bases DCT lo cual implica una reducción en la caída de los bordes en cada bloque. La recuperación de una imagen mediante la aplicación de una LBT mejora respecto a una que fue recuperada con una LOT [7] .

3.3. Cuantización

La *cuantización* es el proceso de conversión de una señal con un intervalo de X valores a una señal con un intervalo reducido de Y valores. Esto hace posible que la señal cuantizada se pueda representar con menos bits que la original, dado que el intervalo de valores es más pequeño [21].
En la compresión de imágenes y video, el proceso de codificación por transformadas no realiza la compresión por sí mismo. La transformación ocasiona que las partes de energía más significativas de la imagen se concentren en los componentes de baja frecuencia, provocando que la mayoría de los coeficientes tengan poca energía. La cuantización y la codificación de longitud variable de los coeficientes de la transformada son los que se encargan de reducir la tasa de bits, siendo el propósito de la cuantización el de eliminar aquellos coeficientes que no son relevantes en la apariencia visual de la imagen. Durante el proceso inverso a la cuantización, los coeficientes removidos ya no pueden ser recuperados, de tal forma que la compresión es con pérdidas. En los codecs de compresión de imágenes y video, la operación de cuantización normalmente se realiza en dos partes: un *cuantizador directo* en el codificador y un *cuantizador inverso* en el decodificador.
Para iniciar con el proceso de cuantización es necesario definir un parámetro denominado *tamaño de escalón QP*. Si el tamaño del escalón es grande, el intervalo de valores cuantizados será pequeño, por lo que puede ser representado eficientemente durante la transmisión (compresión alta), sin embargo, los valores cuantizados serán una aproximación muy pobre de la señal original. Si el tamaño del escalón es pequeño, los valores cuantizados serán mucho más parecidos a la señal original, pero se tendrá un intervalo de valores cuantizados más grande, lo que reduce la eficiencia de la compresión [21]. La ecuación (3.17) ejemplifica el proceso de cuantización por truncamiento de un número fraccionario tomando unicamente la parte entera. El proceso descrito es con pérdidas debido a que no es posible determinar el valor exacto del número fraccionario original a partir del valor redondeado.

$$S_q = Parte\ entera\left(\frac{X}{QP}\right) \tag{3.17}$$

donde:

X es el valor de la muestra original

S_q es el valor de la muestra cuantizada

QP es el tamaño de escalón de cuantización

La ecuación (3.18) describe el proceso inverso de cuantización expresado en la ecuación (3.17).

$$Y = S_q QP \tag{3.18}$$

donde Y representa al valor de la muestra obtenido después de haber aplicado el proceso inverso de cuantización.

3.4. Estimación y compensación de movimiento

En la transmisión de video, una de las formas de obtener una mayor compresión es aprovechar el hecho de que entre una imagen y la siguiente, muchos de los pixeles no varían. La metodología a seguir consiste en estimar el movimiento y codificar solo aquellas partes donde se detecte movimiento en la imagen, dejando el resto inalterado.
El proceso de estimación de movimiento es el de mayor carga computacional de aquellos que realiza el codec. Para cada bloque en la imagen original, el codec busca, en el entorno equivalente de la imagen de referencia, bloques con una alta correlación con el original.
Para determinar cuál es el mejor vector de movimiento (estimación del desplazamiento horizontal y vertical de cada región o bloque de una imagen dada con respecto a una o varias secuencias de imágenes), se calcula el error de correlación para todos lo vectores candidatos, y estos se comparan entre sí, seleccionándose aquel que presente un menor error de correlación.
El proceso de estimación de movimiento es aplicado sólo a las componentes de luminancia. Los vectores de movimiento de la crominancia, se obtienen dividiendo por dos los vectores hallados para la luminancia.
Para realizar la compresión entre imágenes, los algoritmos trabajan con grupos de imágenes (GOP, *Group of Images*), de tal manera que según la compresión aplicada, se definen diversos tipos de *cuadros* (llamemos cuadro a una imagen independiente dentro de una secuencia de imágenes, como se muestra en la figura 3.2):

- Cuadro I (*Intra*): solo incluyen compresión espacial; su codificación no depende de otros cuadros.
- Cuadro P (*Predictive*): cuadros referidos al cuadro P/I anterior, incluyen compresión temporal.
- Cuadros B (*Bidireccional predictive*): cuadros referidos al P/I anterior o posterior.

Los cuadros anteriores son los más comunes en la mayoría de codecs que se utilizan. Ahora, un GOP comienza con un cuadro I, que es codificado sin referencia a ningun otro cuadro. Cada n-ésimo cuadro de este GOP, dependiendo de los parámetros de configuración del codec, se codifica como un cuadro P, el primero relativo al cuadro I del GOP, el segundo al primer cuadro P y así sucesivamente.
Los cuadros B intermedios entre 2 cuadros P/I se codifican con predicción hacia adelante o hacia atrás, dependiendo cuál dé mejores resultados. Cada k cuadros se vuelve a codificar un cuadro como I, volviendo a comenzar con un nuevo GOP, todo el proceso, que se muestra en

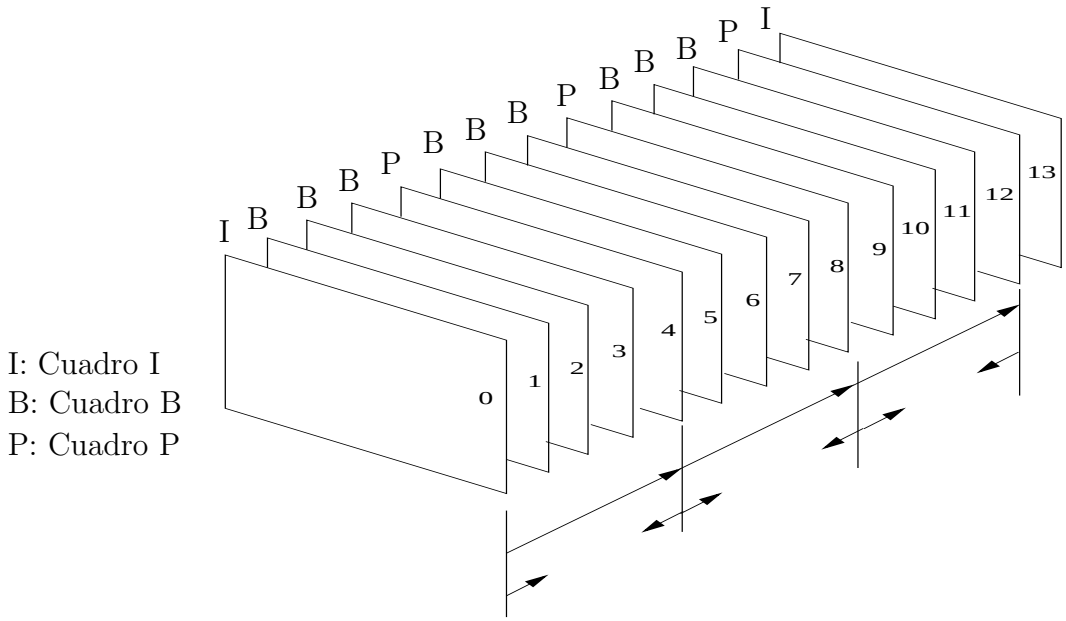


Figura 3.2: Bloques I, P y B en la estimación de movimiento

la figura 3.2. Del esquema anterior, la propagación de errores se mantiene únicamente dentro de cada GOP, ya que cada cuadro I vuelve a comenzar todo el proceso de predicción. Por otro lado, la existencia de cuadros B implica que la predicción se puede hacer utilizando cuadros que ya han sido recibidos codificando un cuadro anterior en el tiempo, debido a que están almacenados en un *buffer*.
Para la estimación del movimiento, la imagen se divide en macrobloques (unidad básica en el proceso de estimación) de NxN pixeles y la búsqueda de porciones similares en la imagen de referencia se hace pixel a pixel en una determinada vecindad [16] .

3.5. Codecs de video

El esquema general de un codec de video por cuadro para su transferencia o almacenamiento se muestra en la figura 3.3. Durante el preprocesado en la codificación se trata de eliminar la información no perceptible por el ojo humano. En el postprocesado, durante la decodificación, intenta eliminar discontinuidades resultantes del submuestro de la imagen.

El proceso de transformación, durante la codificación, convierte la imagen de video de entrada del dominio espacial al dominio de la frecuencia. Habitualmente se divide la imagen en bloques y se realiza la transformada de cada uno de esos bloques. Las transformadas comprimen la imagen, debido a que por un lado la imagen codificada en el dominio de la frecuencia necesita menos bits para transmitir, y por otro lado el codec aprovecha la mayor sensibilidad del ojo humano a los componentes de baja frecuencia para reducir la precisión de los coeficientes de alta frecuencia. La transformada inversa, en la decodificación, consiste en realizar la operación contraria a la transformación, obteniendo una matriz de pixeles a

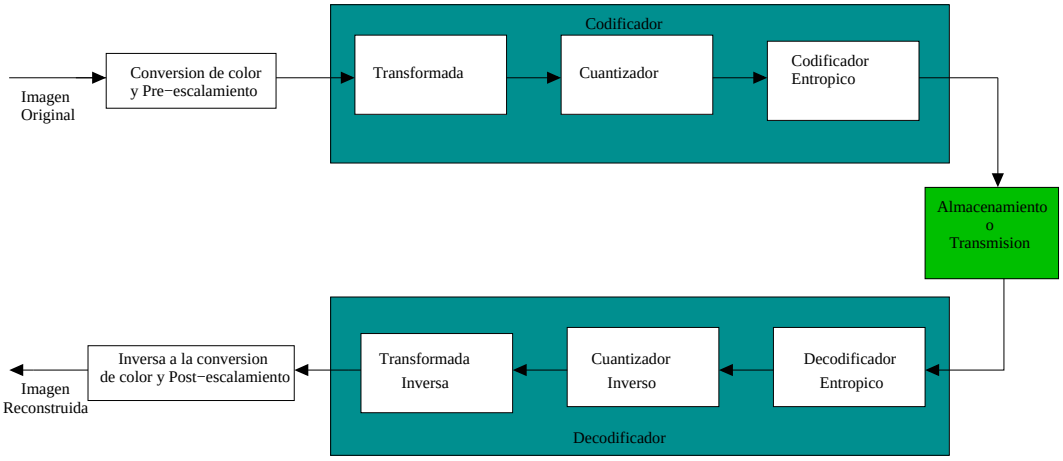


Figura 3.3: Esquema de codificación general de video cuadro por cuadro de imagen

partir de una matriz de coeficientes de frecuencia.
La cuantización, durante la codificación, es el proceso que se encarga de reducir la precisión de los coeficientes del dominio de la frecuencia, dividiendo los valores y redondeando al entero más cercano. Durante la decodificación se realiza el proceso inverso a la cuantización, multiplicando y truncando cada valor cuantizado.
La codificación entrópica reduce la tasa de bits necesaria, eliminando redundancia en el flujo de bits (utilizando menos bits para cada símbolo), minimizando la entropía.
Los codecs de compresión de video se pueden clasificar en [16]:

- Estándares internacionales
- Formatos de propietarios
- Estándares abiertos

Los estándares internacionales frecuentemente utilizan tecnología patentada, alguna autoridad se encarga de controlar la recolección de cuotas en nombre de los propietarios de la patente. Los estándares abiertos son libres para todo usuario [4].
A continación se presentan las características generales de algunos de los codecs de video más populares.

3.5.1. H.261

H.261 de ITU es considerado como el primer estándar para codecs de video en lo que respecta a videoconferencias, siendo completamente pionero en sus conceptos y sentando las bases que han servido a todos los estándares posteriores. La última versión es de marzo de 1993, y aunque ya ha sido superada por los codecs posteriores, como H.263 y H.264, es posible ocuparlo en equipos antiguos [16].
En la codificación se consigue la compresión temporal mediante la predicción entre cuadros de imágen, y la compresión espacial mediante la codificación de cada cuadro de imagen en

donde se emplea cuantización escalar y codificación Huffman.
H.261 puede utilizarse para comunicación unidireccional o bidireccional y define cuadros de imágenes no entrelazadas que tienen una frecuencia aproximada de 30 Hz.
Las imágenes son codificadas utilizando el espacio de color Y, Cb y Cr, disponiendo una relación de aspecto de 4:3. Soporta dos formatos de imagen, CIF (opcional) y QCIF (obligatorio). Para aumentar la predicción, el decodificador tiene además la capacidad de compensación de movimiento, siendo opcional su aplicación [16].
También se aplica la transformada discreta coseno de 8x8 para cada imagen individual. La estructura de datos en H.261 se dispone en una estructura jerárquica de cuatro capas: imagen, grupo de bloques, macrobloque y bloque.
Cada imagen se divide en grupos de bloques, cada grupo de bloques se divide en un número de macrobloques y cada macrobloque comprende cuatro bloques de luminancia (Y) y un bloque para cada crominancia (Cr y Cb).

3.5.2. MPEG-1

MPEG-1 representa el inicio de la saga de estándares de video MPEG (Moving Pictures Expert Group), definido por el ISO/IEC 11172 en 1992, está pensado para una calidad de video a 1.5 Mbps.
MPEG-1 utiliza el formato de resolución SIF, definido para ser compatible con PAL y NTSC. Cada imagen se transmite sólo en modo progresivo, no entrelazado. Es utilizado en aplicaciones típicas de baja resolución y baja tasa de bits, aunque admite en el estándar resoluciones de hasta 4095x4095. Es utilizado en formatos como VCD y DVD.

3.5.3. MPEG-2 y H.262

MPEG-2 incluyó muchas mejoras sobre su antecesor, definido por el ISO/IEC 13818 en 1996, el estándar está dividido en partes [16]. Algunas de estas partes son las siguientes:

- Parte 1 (Sistemas). Define dos contenedores de información:
  - Transport Stream: muy utilizado en difusión (broadcast); por ejemplo en televisión por cable o televisión por satélite.
  - Program Stream: definido por medios fiables de almacenamiento como DVD.
- Parte 2 (Video). De concepción similar a MPEG-1, le añade mejoras, admitiendo mayores velocidades (3 Mbps y superiores). Admite video entrelazado y añade tramas B, pudiendo utilizar submuestreos 4:2:0, 4:2:2 y 4:4:4. Mantiene compatibilidad con MPEG-1.

MPEG-2 define 6 perfiles y 4 niveles, combinando las posibles configuraciones de los diversos parámetros: tasas de bit, resoluciones, etc.
En cuanto a H.262, llamado "Tecnología de la información - Codificación genérica de imágenes en movimiento e información de audio asociada: Video", es idéntico en contenido a la parte

de video de ISO/IEC MPEG-2. La primera versión surge en julio de 1995, y la versión más reciente fue actualizada en el año 2000 [16].
H.262 fue desarrollado en un grupo conjunto de trabajo entre ITU-T e ISO/IEC, siendo publicado conjuntamente como estándar de ambas organizaciones. Fue diseñado para codificación de video en entornos de alto ancho de banda y para altas resoluciones.

3.5.4. MPEG-3

MPEG-3 fue concebido y diseñado para manejar señales para televisón de alta definición, en el intervalo de los 20 y 40 Mbps. Simultáneamente al desarrollo de este estándar, en MPEG-2 se obtuvieron avances similares para el campo de la televisión de alta definición, por lo que MPEG-3 no continúa desarrollandose [16].

3.5.5. MPEG-4

MPEG-4 comprende un conjunto de 23 partes de estándares para la compresión de audio y de video, especificados por el ISO/IEC 14496-1,...,ISO/IEC 14496-28 a partir de 1999 y se encuentra en continua evolución en el presente. Se utiliza en difusión de entornos web (streaming), en CD, en videoconferencia y en difusión de televisión (broadcast).
MPEG-4 absorbe MPEG-1 y MPEG-2, añadiendo mejoras sobre los mismos, por ejemplo la gestión de derechos digitales o escenas basadas en objetos. Utiliza video entrelazado y vector de movimiento [16] .

3.5.6. H.264

La recomendación H.264, denominada "codificación de video avanzada para los servicios genéricos audiovisuales", es un trabajo conjunto de ITU-T e ISO/IEC y la primera versión completada en 2003. Publicado también como la parte 10 de MPEG-4, es conocido también como AVC (Advanced Video Coding) [16].
H.264 está definido de manera flexible, tal que permite una mayor versatilidad de operación que sus antecesores: desde 56 kbps para telefonía 3G, hasta varios Mbps para televisión de alta definición. Debido a su flexibilidad es un codec válido para altas y bajas resoluciones, para altas y bajas tasas de bit, y para distintos tipos de aplicaciones y sistemas.
Dos de las características destacables de la recomendación H.264 son:

- Vectores de movimiento aplicables a bloques de hasta sólo 4x4 pixeles.
- Codificación entrópica adaptable

Podría decirse que el codec H.264 es hasta ahora el más reciente y más eficiente de los estándares internacionales; sin embargo, para fines del presente trabajo se hará referencia a los codecs no estandarizados [16].

3.5.7. MJPEG

El MJPEG (JPEG en Movimiento) es un método de compresión no estandarizado que se encarga de codificar a una secuencia de video como una serie de imágenes JPEG, donde cada una corresponde a un cuadro de video. JPEG surgió en 1992 y utiliza el esquema de compresión con pérdidas. La compresión JPEG está basada en la transformada coseno discreta aplicada a bloques de 8x8, seguida por la cuantización, reordenamiento y la codificación de longitud variable.
Originalmente JPEG no estaba destinado para ser utilizado en compresión de video, sin embargo MJPEG se ha vuelto muy popular y es utilizado en varias aplicaciones de transmisión y almacenamiento de video [20]. Las ventajas que presenta MJPEG son:

- Baja complejidad: la complejidad del algoritmo, los requisitos de hardware, procesamiento y almacenamiento son bajos comparados con algún codec que ocupa estimación de movimiento.

- Tolerancia al error: en el caso de MPEG-1 y MPEG-2 la propagación de errores es inevitable en comparación con MJPEG, donde cada cuadro de imagen es independiente.

- Posición en el mercado: JPEG es quizás el estándar de compresón de imágenes más conocido y utilizado en el mercado, de tal forma que los usuarios potenciales ya están familiarizados con la tecnología JPEG en Movimiento. Debido a su baja compresión, MJPEG solo es adecuado para comunicaciones con gran ancho de banda (como redes dedicadas). Paradójicamente, los usuarios generalmente tienen una buena experiencia de MJPEG debido a que las instalaciones tienden a sufrir de los problemas de retardos y ancho de banda encontrados en los codecs que ocupan estimación de movimiento cuando son utilizados en redes de datos (como internet) o canales con baja tasa de bits. Por otro lado, MJPEG es popular para aplicaciones como captura de video, edición de video en PC y cámaras de seguridad [20].

Debido a que MJPEG presenta los mismos problemas que JPEG por cuadro de imagen, se han buscado otras alternativas que mantengan la simplicidad de implementación pero que mejoren la calidad visual y la compresión de video.

3.5.8. MJPEG-2000

Otro método de compresión de video no estandarizado, similar a MJPEG, es el MJPEG-2000 (JPEG-2000 en Movimiento), donde la secuencia de video se comprime como una secuencia de cuadros individuales, aplicando a cada cuadro JPEG-2000 [4].
JPEG-2000 surgió en el año 2000 y presenta muchas ventajas sobre JPEG como es la eliminación de artefactos debido a que la transformada que ocupa es la DWT (Discrete Wavelet Transform) permitiendo su aplicación a una imagen completa sin la necesidad de dividir en bloques. Tiene una muy buena relación señal a ruido, lo cual implica una mejor calidad de la imagen. El resultado es un estándar de compresión con un desempeño de compresión significativamente mejor que JPEG. Para la misma calidad de imagen, JPEG-2000 puede comprimir al menos dos veces más que JPEG. Esta ganancia se logra a costa de incrementar

la complejidad y los requisitos de almacenamiento durante la codificación y decodificación [20] [24].

3.5.9. MJPEG XR

Debido a la baja aceptación de JPEG-2000 en el video, surgió la necesidad de realizar un nuevo estándar que mejorara tanto a JPEG y JPEG-2000. MJPEG XR (JPEG XR en Movimiento) es un codec no estandarizado basado en el más reciente estándar JPEG XR que data desde 2009. Al igual que MJPEG y MJPEG-2000 el codec MJPEG XR aplica a cada imagen individual el codificador JPEG XR [8] .
Las especificaciones sobre JPEG XR se encuentran en el documento T.832 de ITU-T. Las principales características de JPEG XR es la flexibilidad, ya que no está restringida a un solo tipo de espacio de color, relación de aspecto de imagen o hasta en la compresión se puede elegir con pérdidas o sin pérdidas. La transformada que ocupa es la LBT mejorando la velocidad con que se codifica y decodifica una sola imagen comparativamente con respecto a JPEG 2000 y elimina los problemas de artefactos alcanzando una mejor relación de compresión y superando a JPEG.
En el presente trabajo se eligió utilizar el codec MJPEG XR y aplicarlo a una secuencia de video. El capítulo 4 describe su modo de operación e implementación.

Resumen

En el presente capítulo se abordaron las principales características que comparten la mayoría de los codecs de video así como una breve descripción de su evolución en cuanto a las mejoras que presentan respecto a algunas aplicaciones para las cuales fueron concebidas. Es importante hacer énfasis que la base de los codecs más utilizados y aceptados en el mercado son los que hacen uso de la DCT como transformada.
El estándar JPEG XR es el estándar de compresión de imágenes más actual y que forma base de MJPEG XR. Para este trabajo, el enfoque se realizará sobre este codec aprovechando las mejoras que presenta sobre sus predecesores.

Capítulo 4

Compresión de video MJPEG XR

En este capítulo se describe el codec MJPEG XR dado que el objetivo general de este trabajo es su implementación. Debido a que la base de MJPEG XR es el estándar JPEG XR, se hace una descripción de cada uno de los bloques que lo componen y se hace énfasis en las características que son utilizadas en la implementación del codec.

4.1. Método de compresión MJPEG XR

MJPEG XR es un método de compresión no estandarizado que consiste en aplicar el estándar JPEG XR a cada cuadro de imagen, que compone a una secuencia de video, de manera individual. Dos de las aplicaciones en las que se ha probado a MJPEG XR es en videocámaras de vigilancia [28] y transmisión de video sobre redes inalámbricas [17]. MJPEG XR tiene muchas ventajas como codec de video, las cuales son:

- MJPEG XR soporta acceso de cuadros de imagen aleatorio ya que no utiliza predicción de movimiento.
- Al procesar las imágenes de manera independiente se disminuye el efecto del error, ya que si se presenta un error en una imagen, éste no se propaga al resto de las imágenes.
- La implementación de MJPEG XR requiere de poco hardware, procesamiento y almacenamiento bajos en comparación con los codecs que utilizan predicción de movimiento.

A pesar de sus bondades también presenta las siguientes desventajas:

- MJPEG XR no explota la redundancia temporal de movimiento en una secuencia de video, por lo que su desempeño de compresión es bajo comparado con codecs que si resuelven esta característica.
- MJPEG XR no está estandarizado, por lo que puede haber problemas de compatibilidad con los archivos de salida de distintos fabricantes.

4.2. Descripción del estándar JPEG XR

JPEG XR es el más reciente codificador de imágenes del comité JPEG, el cual tiene como objetivo principal la representación en tonos continuos de imágenes fijas como las fotografías, consiguiendo una alta calidad de imagen similar a JPEG 2000 mientras que requiere un bajo costo de recursos computacionales y capacidad de almacenamiento. Por otra parte, el estándar se extiende a una gran cantidad de formatos y aplicaciones que han surgido a lo largo del tiempo [8].
Con la rápida evolución de las tecnologías sobre imágenes digitales se ha obtenido un gran éxito del uso de la fotografía digital por los profesionales y consumidores. En el centro de este éxito se encuentra el codificador estandarizado JPEG (ITU-T T.81 — ISO/IEC 10918-1), el cual ha jugado un papel clave. Sin embargo, mientras que se ha convertido en uno de los estándares más utilizado en el mundo, con sus ya más de 20 años desde su creación, JPEG alcanzó sus límites y ha comenzado a estar por debajo del desarrollo de inovadoras características y mejoras en el rendimiento de la fotografía digital [8]. Recientemente, el comité JPEG produjo el estándar JPEG 2000 (UIT-T T.800 — ISO/IEC 15.444-1), introduciendo una serie de novedades y nuevas funcionalidades. Sin embargo, contiene un notable aumento de recursos computacionales y no ha tenido un gran impacto en aplicaciones como la telefonía móvil, en los entornos embebidos y principalmente en el mercado de la fotografía digital. Los fotógrafos profesionales han optado por utilizar imágenes en crudo para evitar las limitaciones que se mantienen en la línea base de JPEG, pero la codificación cruda requiere de una gran capacidad de almacenamiento y es generalmente diseñada para una cámara específica, además de carecer de interoperabilidad y de una documentación publicada.
JPEG XR (ITU-T T.832 — ISO/IEC 29199) es un nuevo estándar de codificación de imágenes dirigido principalmente a la representación de una imagen en tonos continuos como las obtenidas con una cámara fotográfica. Está diseñado para superar las limitaciones de su predecesor y lograr al mismo tiempo alta calidad en imágenes así como minimizar recursos computacionales y de almacenamiento.
El término "XR" hace referencia a una amplia "gama de aplicaciones" más allá de las capacidades que mantiene JPEG.

Estructura del estándar JPEG XR

El estándar de codificación de imágenes JPEG XR, a cargo del comité JPEG, está dividido en las siguienes partes:

- Parte 1. Arquitectura del sistema: ésta es la parte técnica no normativa que da una visión general de los diferentes componentes que constituyen a JPEG XR y ofrece algunas recomendaciones sobre las mejores prácticas de codificación y decodificación.

- Parte 2. Especificación del codificador de imágenes: en esta parte se especifica el formato de la imagen JPEG XR. Fue aprobado como norma final tanto por ITU-T (donde es conocido como ITU-T recomendación T.832) en marzo del 2009 y en ISO/IEC(como

ISO/IEC 29199-2) en junio del 2009.

- Parte 3. Motion JPEG XR: En esta parte se especifica el uso de JPEG XR en la codificación de secuencias de imágenes asociadas al movimiento. El formato del archivo Motion JPEG XR se basa en el estándar ISO de formatos de archivos multimedia. Aun se encuentra en proceso de aprobación por el ISO/IEC.

Existen dos partes más que constituyen al estándar JPEG XR sin embargo, todavía continuan en revisión.

4.3. Proceso de codificación JPEG XR

El proceso de codificación JPEG XR consiste en diversas etapas encargadas de tratar los datos de una imagen introducida en formato crudo, regularmente obtenidos desde una cámara fotográfica o de video. Cada etapa tiene un propósito específico y utiliza técnicas que aprovechan características que mejoran el rendimiento de la totalidad del codificador, así como la calidad de la imagen si se compara con respecto a los codificadores predecesores de JPEG XR. La figura 4.2 muestra a cada una de las etapas que componen al codificador.

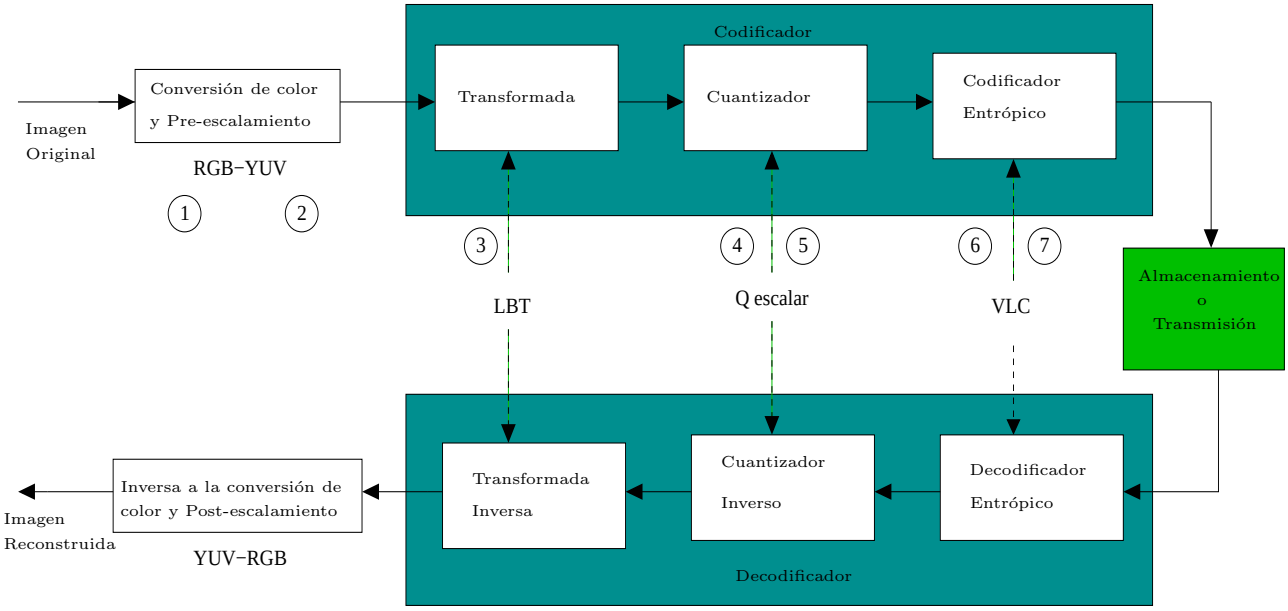


Figura 4.1: Etapas del proceso de codificación JPEG XR

Una breve descripción de cada etapa se presenta a continuación:

1. La primera etapa consiste en transformar la imagen original que se encuentra en un espacio de color RGB al espacio YUV para posteriormente aplicar submuestreo.

2. Adicionalmente, cada componente Y, U y V es dividido en diferentes porciones de imagen, correspondientes a tres niveles jerárquicos: *azulejos*, *macrobloques* y *bloques*. El tamaño de un mozaico está relacionado con el uso de memoria y los resultados de compresión [28]. Un macrobloque está predefinido a una matriz de 16 x 16 pixeles, mientras que el tamaño de un bloque está predefinido a 4 x 4 pixeles.

3. Posteriormente en la codificación, se aplica la transformada LBT que en JPEG XR está compuesta de dos operadores importantes: POT (Photo Overlapp Transform) y PCT (Photo Core Transform). Donde POT es un operador opcional y tiene como finalidad disminuir los artefactos que aparecen en los bordes entre cada bloque. PCT se encarga de obtener los coeficientes de frecuencia en cada bloque.

4. Para remover aquellos coeficientes de menor importancia, se introduce una etapa de cuantización. Los parámetros de cuantización (QP, por sus siglas en inglés) tienen un alto impacto en el resultado de la calidad final de la imagen. Si se elige un parámetro de cuantización muy grande se pierden muchos coeficientes dando origen a la compresión con pérdidas.

5. Por otro lado, se utiliza predicción para remover aquellos parámetros de cuantización que contribuyen a la redundancia entre cada bloque.

6. Existe un proceso previo y necesario para realizar la codificación entrópica, que consiste en convertir una matriz de dos dimensiones a una matriz unidimensional. Al proceso descrito en JPEG XR se le denomina exploración adaptable y depende principalmente de la dirección de la predicción y de un promedio estadístico.

7. Finalmente, la codificación entrópica consiste en disminuir la información para poderla almacenar o transmitir.

Para la decodificación, a cada proceso descrito se asocia un proceso inverso con la finalidad de regresar a la imagen "original".
En las siguientes secciones se hace una descripción más profunda de cada proceso haciendo énfasis en lo que será utilizado en el presente trabajo.

4.3.1. Conversión de Color y Pre-escalamiento

En la sección 2.3 se trataron aspectos que relacionaban los espacios de color YUV y RGB, mientras que en la sección 2.4.5 se trataron algunas técnicas de submuestreo existentes. El codificador JPEG XR utiliza la conversión de color de RGB-YUV y soporta el submuestreo 4:4:4, 4:2:2 y 4:2:0. Para el caso del presente trabajo se utiliza un submuestreo 4:2:0 debido a que se aprovecha un sistema de adquisición con el cual ya se contaba [26].

4.3.2. Particionado de la imagen

JPEG XR divide a una imagen en partes pequeñas, denominadas jeraquías, las cuales pueden ser procesadas de forma individual más facilmente. Las tres jerarquías que existen para dividir a una imagen son: azulejos, macrobloques y bloques. La figura 4.2 muestra las jerarquías utilizadas.

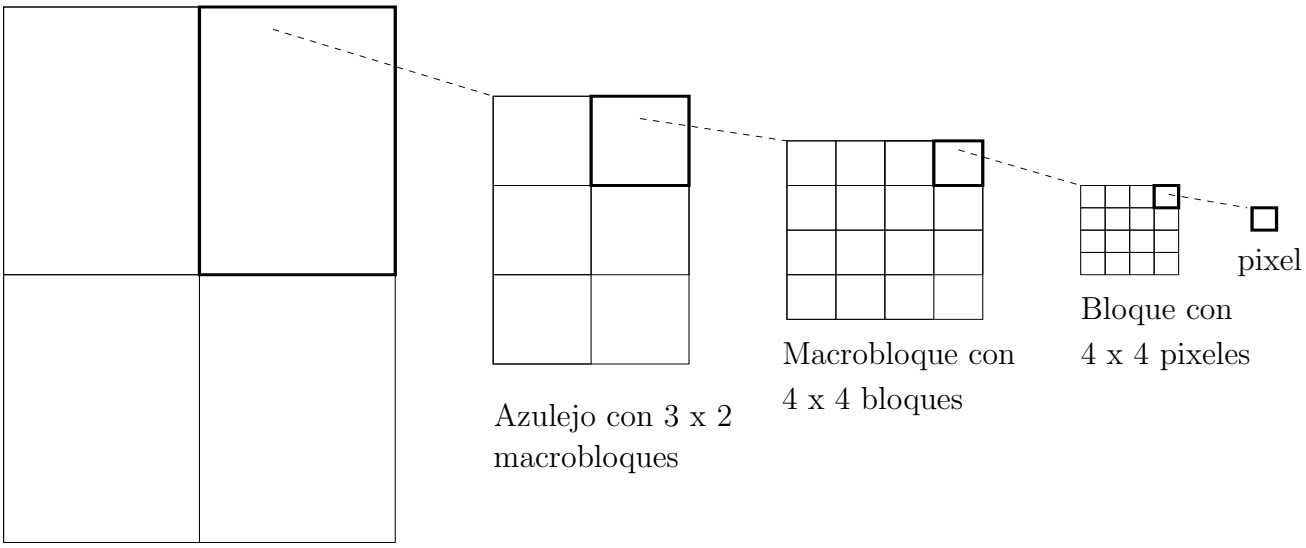


Figura 4.2: División jerárquica de una imagen

Azulejos

El tamaño de un azulejo debe de ser fijado al inicio de la compresión JPEG XR. Cada azulejo en una imagen es procesado independientemente, como si fuera una pequeña imagen. Dividir a una imagen en pequeños azulejos tiene las siguientes ventajas:

- Flexibilidad: los azulejos en una misma imagen pueden tener diferentes configuraciones en los procesos. Por ejemplo, diferentes parámetros de cuantización, distinto orden de exploración adaptable y diferentes tablas de codificación. Ésta es una buena característica cuando una persona está interesada solamente en alguna parte específica de la imagen.

- Robustez: si ocurre un error en alguno de los azulejos, los restantes no serán afectados.

Las dos características antes expuestas hacen a JPEG XR más flexible y adecuado para su implementación en hardware, especialmente cuando la limitación de espacio en memoria es prioridad.
Sin embargo, utilizar muchos azulejos puede decrementar la eficiencia de compresión debido a que la redundancia entre cada azulejo no puede ser extraída. En otras palabras, la relación de compresión decrese cuando el número de azulejos aumenta. Para obtener la mejor relación de compresión se ha considerado, en este trabajo, a toda la imagen como un solo azulejo.

Macrobloques

El macrobloque es una unidad base de JPEG XR y el tamaño es predefinido directamente por el estándar. En el plano de luminancia, cada macrobloque contiene a 16 x 16 pixeles y a su vez se obtiene un componente DC (Direct Current), 15 componentes LP (Low Pass) y 240 HP (High Pass). En los planos de Crominancia, para YUV 4:2:2 y 4:2:0, cada macrobloque es de 8 x 16 y 8 x 8, respectivamente. Las etapas PCT, POT, cuantización y predicción están diseñadas para procesar un solo macrobloque cada instante.

Bloque

Un bloque es la unidad más pequeña en JPEG XR. Cada bloque consiste de 4 x 4 pixeles y un macrobloque contiene 4 x 4 bloques.

4.3.3. Transformada

La transformada LBT es utilizada en JPEG XR, debido a que es una de las técnicas más novedosas para procesar imágenes. LBT en JPEG XR está integrada por dos partes: El prefiltrado, que es realizado mediante el operador POT, y la obtención de coeficientes de frecuencia mediante el operador PCT. A continuación se explican cada uno de los operadores.

Prefiltrado

Cuando se aplica la transformación de los pixeles a coeficientes de frecuencia, no se toma en cuenta que existe redundancia entre los bordes de cada bloque dando origen a los artefactos. POT es una nueva característica implementada en JPEG XR que se encarga de disminuir o "suavizar" los artefactos que aparecen en los bordes de los bloques. POT puede ser utilizada de forma opcional, surgiendo tres posibilidades:

- Sin solapamiento: no utiliza ningún tipo de prefiltrado y disminuye el tiempo de procesamiento. Sin embargo, se obtiene una baja relación de compresón.
- Un nivel de solapamiento: ocupa una etapa POT. Tiene una mejor relación de compresión pero la desventaja es que complica el cómputo y aumenta el tiempo de procesamiento.
- Dos niveles de solapamiento: utiliza dos etapas POT que implica una muy alta complejidad en cómputo, demanda de memoria y tiempo de procesamiento. Sin embargo, al mismo tiempo se obtiene la mejor relación de compresión a una baja tasa de bits [28] .

Para el presente trabajo se realizaron pruebas sin solapamiento para aprovechar mejor los recursos del DSP.
En comparación con PCT, que sólo se realiza dentro de cada bloque de 4 x 4 pixeles, POT es más complejo y consume más memoria debido a que no sólo se realiza dentro de cada macrobloque, sino también en los límites de los bloques adyacentes [28]. POT consiste en dos operaciones principales:

- Prefiltrado 4 x 4: se aplica a todas las uniones existentes entre macrobloques y totalmente en las áreas interiores. POT consiste de 4 suboperaciones: transformada Hadamard T_H, rotación hacia adelante T_R de 2 puntos, escalamiento directo de 2 puntos T_S y la transformada Odd odd T_{oddodd}.
- Prefiltrado de 4 puntos: Consiste en aplicar el prefiltrado a los bordes de la imagen.

Tramsformada Hadamard

La matriz Hadamard T_H está definida por la ecuación (4.1).

$$T_H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{4.1}$$

Sin embargo, la ecuación anterior define a una transformada Hadamard de una dimensión para poder obtener la matriz Hadamard en dos dimensiones es necesario definirla por medio del producto de Kronecker como se muestra en la ecuación (4.2)

$$T_{HH} = T_H \otimes T_H = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{4.2}$$

El símbolo $\otimes$ es el operador conocido como producto de Kronecker.

Rotación hacia adelante

La Rotación hacia adelante es una operación que rota al vector de entrada sobre un plano en un ángulo θ y tiene como representación matricial la mostrada en la ecuación (4.3)

$$T_R = \begin{bmatrix} -cos\theta & sen\theta \\ sen\theta & cos\theta \end{bmatrix} \tag{4.3}$$

donde θ en la ecuación 4.3 tiene un valor de $\frac{\pi}{8}$, lo cual da como resultado la matriz (4.4)

$$T_R = \begin{bmatrix} -0.9239 & 0.3827 \\ 0.3827 & 0.9239 \end{bmatrix} \tag{4.4}$$

Escalamiento Directo

Escalamiento directo es una operación que escala un vector de entrada por un factor S. Esta operación puede ser descrita por la ecuación (4.5)

$$T_S = \begin{bmatrix} S^2 & S^2 & 1 & 1 \\ S^2 & S^2 & 1 & 1 \\ 1 & 1 & S^{-2} & S^{-2} \\ 1 & 1 & S^{-2} & S^{-2} \end{bmatrix} \tag{4.5}$$

Transformada Odd odd

La Transformada Odd odd se obtiene mediante el producto de Kronecker de dos T_R, como se muestra en la ecuación (4.6).

$$T_{oddodd} = T_R \otimes T_R = \begin{bmatrix} -cos\theta & sen\theta \\ sen\theta & cos\theta \end{bmatrix} \otimes \begin{bmatrix} -cos\theta & sen\theta \\ sen\theta & cos\theta \end{bmatrix} \tag{4.6}$$

Transformada Photo Core (PCT)

Posterior al prefiltrado, la transformada PCT se encarga de pasar los datos del dominio espacial al dominio de la frecuencia. Esto es similar a la forma como se realizaba mediante la transformada DCT en el estándar JPEG. Las diferencias entre la DCT y la PCT son:

- PCT define tres bandas de frecuencia DC, LP y HP, las cuales no existen en la DCT. Al definir tres bandas se puede realizar su procesamiento de forma independiente.
- En JPEG XR la PCT se aplica sobre bloques de 4 x 4 pixeles la cual hace una unidad más pequeña que en JPEG donde se utiliza una DCT de 8 x 8 pixeles.

PCT es mucho más simple que POT dado que la entrada es siempre aplicada a un bloque de 4 x 4 pixeles. Las tres suboperaciones utilizadas por la PCT son la T_H, T_R y T_{oddodd}. La figura 4.4 muestra el proceso de aplicar los operadores en la PCT.

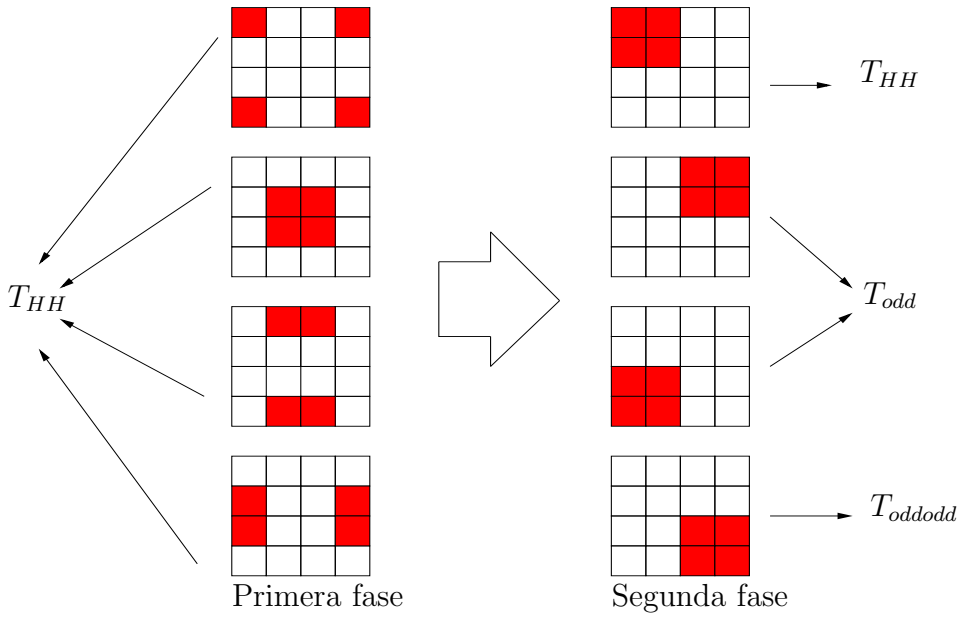


Figura 4.3: Fases de la PCT

El proceso para aplicar PCT consiste en dos fases. En la primera fase, se aplica una transformada Hadamard de 2 x 2 a las esquinas, en los límites y en el área interior de un bloque. En la segunda fase, una T_H se aplica a la esquina superior izquierda mientras que una transformada Odd se aplica a la esquina superior derecha y a la esquina inferior izquierda. La esquina superior derecha es procesada por una transformada Odd odd.

Transformada Odd

La T_{odd} se obtiene mediante el producto de Kronecker de una T_R y una T_H como se muestra en la ecuación (4.7)

$$T_{odd} = T_R \otimes T_H = \begin{bmatrix} -cos\theta & sen\theta \\ sen\theta & cos\theta \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{4.7}$$

Etapas de POT y PCT

POT y PCT son concatenados apropiadamente y su combinación forma la transformada LBT. En la figura 4.4 se muestra un esquema general de las posibles combinaciones entre POT y PCT. De la figura 4.4 se observa que POT es opcional y por tal motivo podemos omitirlo, quedando PCT como el único operador aplicado sobre cada macrobloque. PCT se calcula en dos etapas y de acuerdo a cada etapa se obtienen los coeficientes pertenecientes a una banda de frecuencia distinta. En la primera etapa, PCT es aplicada a cada bloque que compone a un macrobloque. El resultado es la obtención de un coeficiente DC ubicado en la parte superior izquierda de cada bloque y 15 coeficientes HP por bloque. En total se obtienen 16 coeficientes DC y 240 coeficientes HP. Para la segunda etapa, los 16 coeficientes DC son

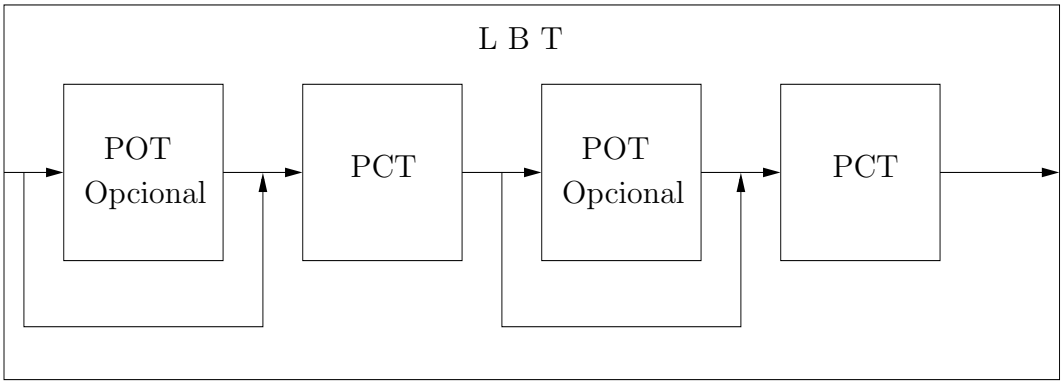


Figura 4.4: Combinación de la POT y PCT

ordenados como una sola matriz de 4 x 4 y se aplica nuevamente PCT obteniéndose un sólo coeficiente DC y 15 coeficientes LP. Esta vez, el coeficiente DC queda ubicado en la parte superior izquierda del macrobloque y los 15 coeficientes LP son reposicionados a su ubicación inicial. La figura 4.5 es un esquema del procedimiento antes descrito.

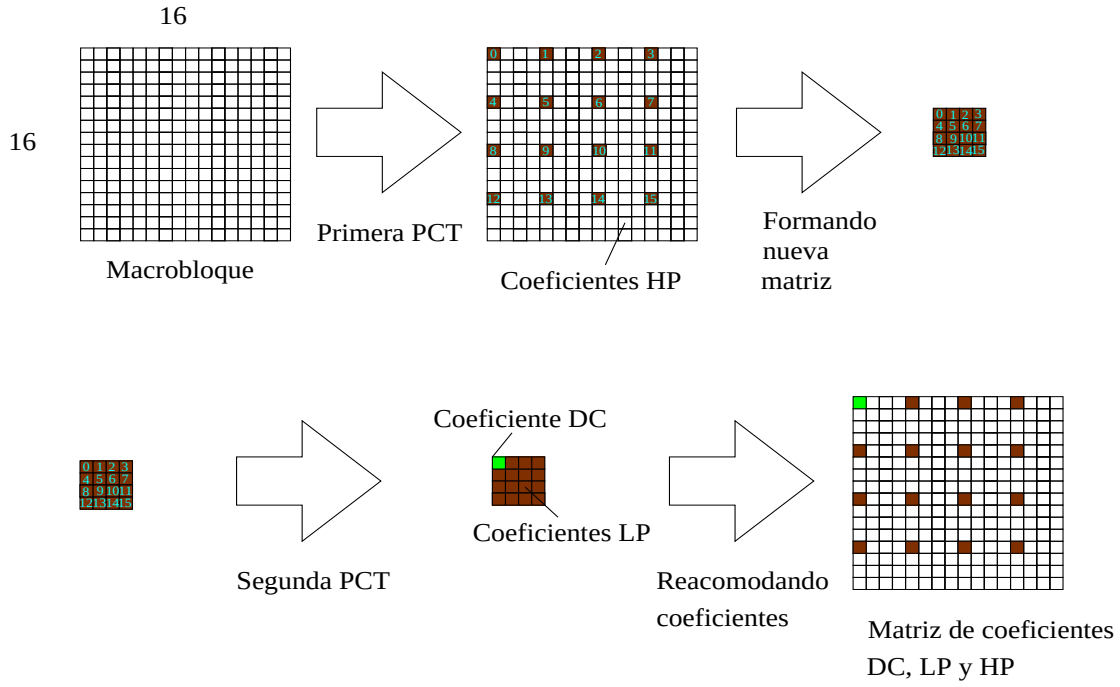


Figura 4.5: Obtención de coeficientes de frecuencia

4.3.4. Cuantización

Después de haber aplicado la transformada LBT, los coeficientes de alta frecuencia menos importantes son agrupados y pueden ser removidos por medio de la cuantización. En la cuantización cada coeficiente obtenido de la transformada LBT es dividido por un parámetro de cuantización y redondeado a un entero llamado valor cuantizado. Es en esta etapa donde se puede introducir distorsión a la imagen, esto sucede cuando el parámetro de cuantización es mayor que uno. JPEG XR soporta la compresión con pérdidas y sin pérdidas, la cual debe ser decidida en esta etapa; si la compresión es con pérdidas, el valor resultante de la cuantización no puede ser recuperado en el decodificador. Esto sucede comúnmente en la banda HP.

Parámetro de cuantización

El valor de QP está en el intervalo de 1 a 255. También puede variar a través de los distintos planos de color, azulejos y las diferentes bandas de frecuencias [22] [28]. La amplia gama de valores que puede obtener QP ofrece flexibilidad en la precisión al codificador para controlar la calidad. Hay que tomar en cuenta que la etapa posterior a la cuantización es la predicción y solo puede ser calculada para un mismo QP. Para este trabajo se eligió trabajar con un único QP para toda la imagen.

4.3.5. Predicción

En la mayoría de los casos, los pixeles entre cada bloque que forman una pequeña área de la imagen mantienen valores relativamente similares. Se dice que existe redundancia espacial cuando hay una gran similitud entre los pixeles de bloques contiguos. Por lo tanto, se vuelve más eficiente almacenar las diferencias que los valores reales debido a que se generan varios ceros que son aprovechados en la codificación entrópica.
La predicción se realiza tomando a un macrobloque y comparándolo con dos macrobloques vecinos de tal forma que la dirección de la predicción se toma del macrobloque con más similitud al actual. La dirección de la predicción puede provenir de la izquierda, superior y superior izquierda. Además, el esquema de la predicción cambia de acuerdo a las bandas de frecuencia. En los siguientes párrafos se trata a la predicción para DC, LP y HP.

Predicción en coeficientes DC

La predicción de los coeficientes DC se realizan entre macrobloques y puede provenir de arriba, de la izquierda o de la parte superior izquierda. El nivel de DC que se parezca más al actual es seleccionado y se calcula su diferencia. En la figura 4.6 se presenta un ejemplo de la predicción de un coeficiente DC.
El esquema de la figura 4.6 muestra un ejemplo de cómo sucede la predicción entre macrobloques. Existe solamente un coeficiente de DC en cada macrobloque, y los cuatro valores representan a cuatro macrobloques. Del ejemplo de la figura 4.6, el valor 64 corresponde a la

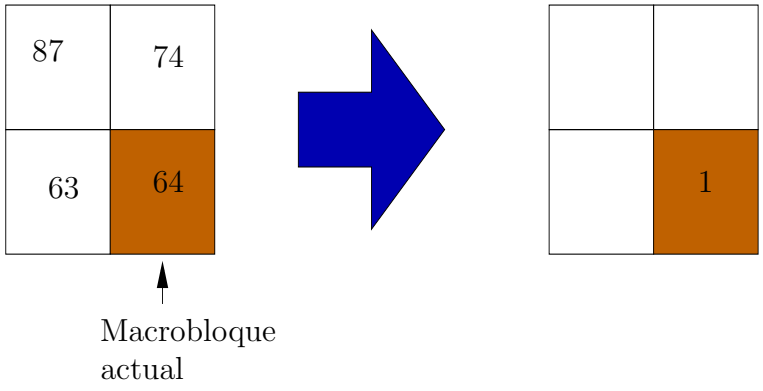


Figura 4.6: Ejemplo de predicción DC

componente DC del macrobloque actual. Comparando el valor del componente actual en las tres direcciones se obtiene que el valor 63 del macrobloque de la izquierda, mantiene mayor similitud y se elige como sustraendo. El resultado de la diferencia para este caso es 1 y es utilizado finalmente en la siguiente etapa de *exploración adaptable*.

Predicción en coeficientes LP

Al igual que la predicción en DC, la predicción en LP se realiza entre macrobloques, pero solo se predicen dos direcciones las cuales son: a la izquierda y arriba. La dirección de predicción de LP es decidida por el modo de predicción DC, si los macrobloques involucrados tienen el mismo parámetro de cuantización, de lo contrario se omite todo el procedimiento. La figura 4.7 muestra un ejemplo de la predicción LP.

Predicción en coeficientes HP

A diferencia de la predicción en DC y LP, la predicción en HP se realiza dentro de cada macrobloque. La intención es almacenar las diferencias de cada bloque que se encuentra dentro de un macrobloque. La predicción de la dirección de HP puede provenir de la izquierda o de arriba, esto es decidido de acuerdo a los coeficientes LP obtenidos dentro del macrobloque analizado. En la figura 4.8 se muestra un ejemplo de la predicción en HP.

4.3.6. Exploración adaptable

El propósito de la exploración es reordenar el arreglo de dos dimensiones en una matriz unidimensional. El orden de los coeficientes debe ser ajustado de manera que los coeficientes iguales a cero estén juntos tanto como sea posible, de esta manera la codificación entrópica se vuelve más eficiente. En JPEG, es utilizado el ordenamiento por zizag mientras que JPEG XR utiliza un ordenamiento dinámico, esto quiere decir que la exploración se va modificando para adaptarse al cambio de los coeficientes distintos de cero de tal forma que los coeficientes iguales a cero queden siempre juntos. Hay que hacer notar que la exploración adaptable se realiza sol amente dentro de cada bloque y que para el caso de los macrobloques se utiliza el

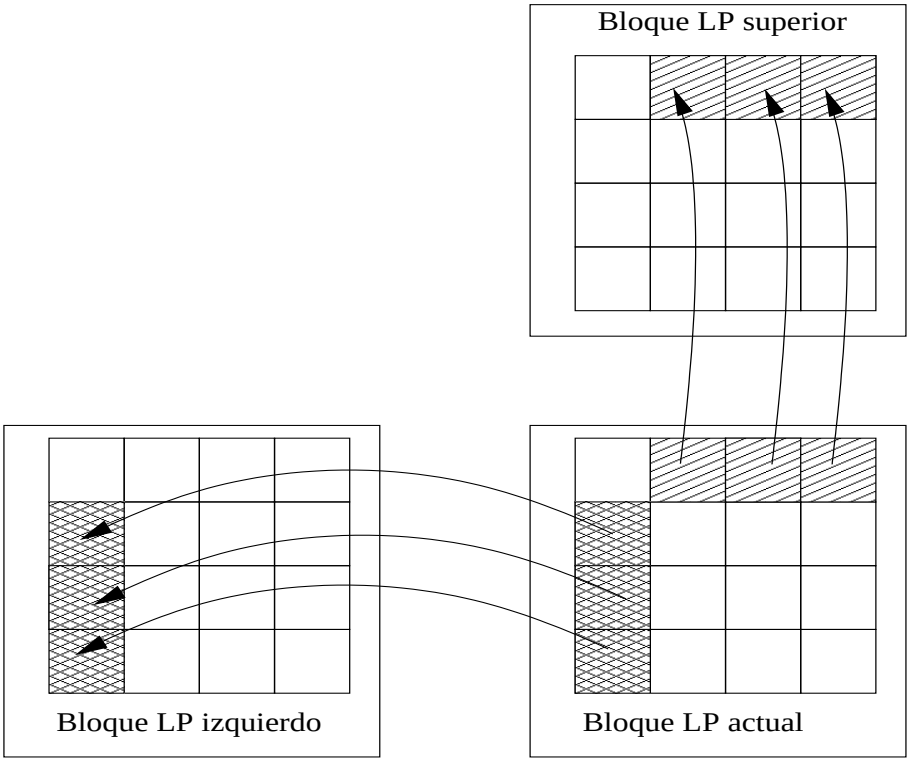
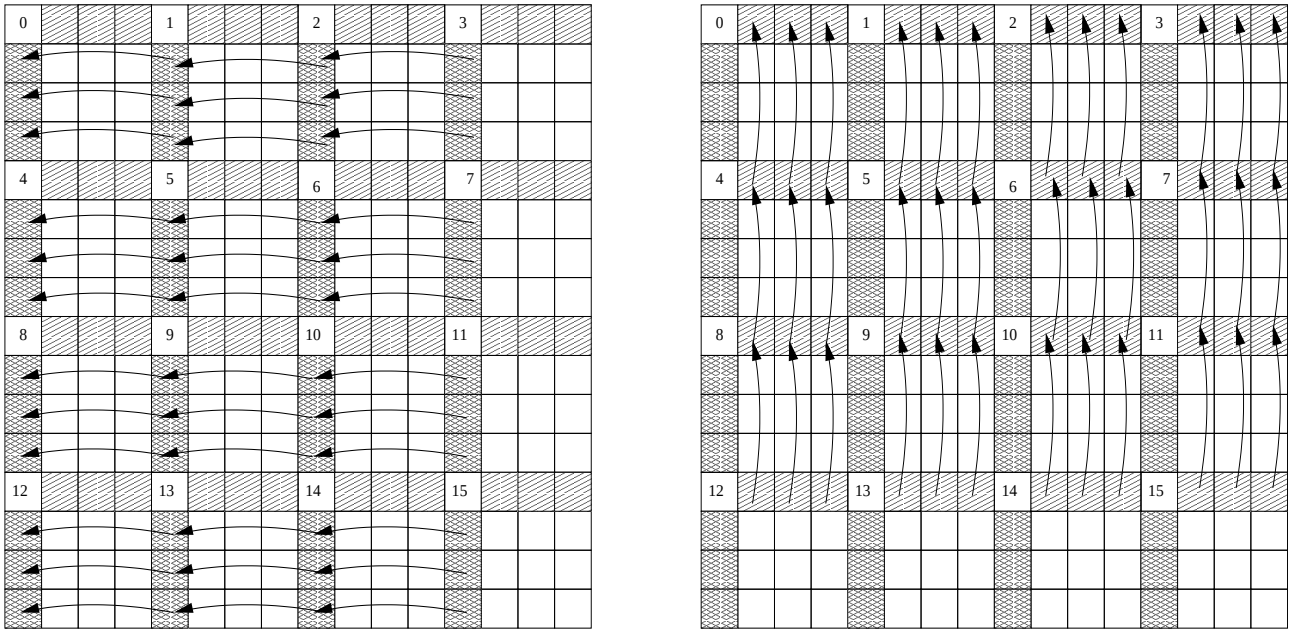


Figura 4.7: Ejemplo de predicción LP



a) Predicción HP a la izquierda b) Predicción HP hacia arriba

Figura 4.8: Ejemplo de predicción HP

tradicional barrido de tramas [28].

4.3.7. Codificador Entrópico

En teoría de la información, la entropía es la cantidad de información contenida en un mensaje. Si la entropía es baja, se necesita menos espacio para almacenar el mensaje. Cuando se desea transmitir o almacenar la información de la imagen, los bits deben formar un vector e idealmente se esperará obtener al primer bit en "1"seguido por tantos ceros como sea posible [28].
JPEG XR utiliza dos técnicas para codificar los bits resultantes de la exploración adaptable para ello utiliza una combinación de *codificación larga de niveles (RLE)* y de VLC.

Codificación larga de niveles (RLE)

La Codificación larga de niveles está basado en el algoritmo denominado *codificación de longitud larga* (CLL) [28]. La idea básica es codificar la información de acuerdo al número de veces que se repite. En la figura 4.9 se muestra un ejemplo del CLL. El formato que se utiliza para codificar consiste en colocar el número de veces que se repite un valor seguido por su respectivo valor. Para el caso del ejemplo presentado, cada dígito es representado por un byte y se observa que existe una gran redundancia en el flujo de bits original. Después de codificar, los 19 bytes iniciales se reducen a solamente 8 bytes.

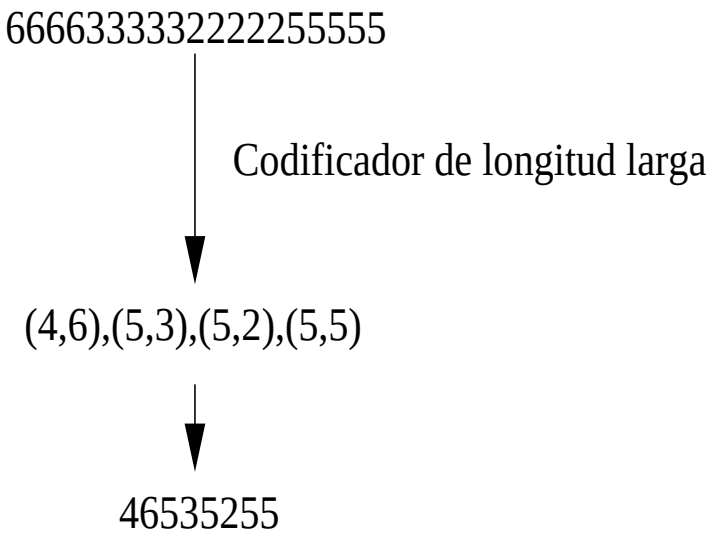


Figura 4.9: Ejemplo de Codifiación de longitud larga

La codificiación larga de niveles tiene un concepto similar al CLL, pero en este caso se utiliza cuando el codificador produce una larga secuencia de ceros entre coeficientes distintos de cero. Un ejemplo sobre esta situación se presenta en la figura 4.10 . Para este caso, se coloca el número de ceros que preceden a un valor seguido por el respectivo valor. Dado que después de cuantizar los coeficientes transformados y haber realizado la predicción adaptable, existe un gran número de ceros generados por las etapas previas, la codificación larga de niveles se vuelve muy eficiente.

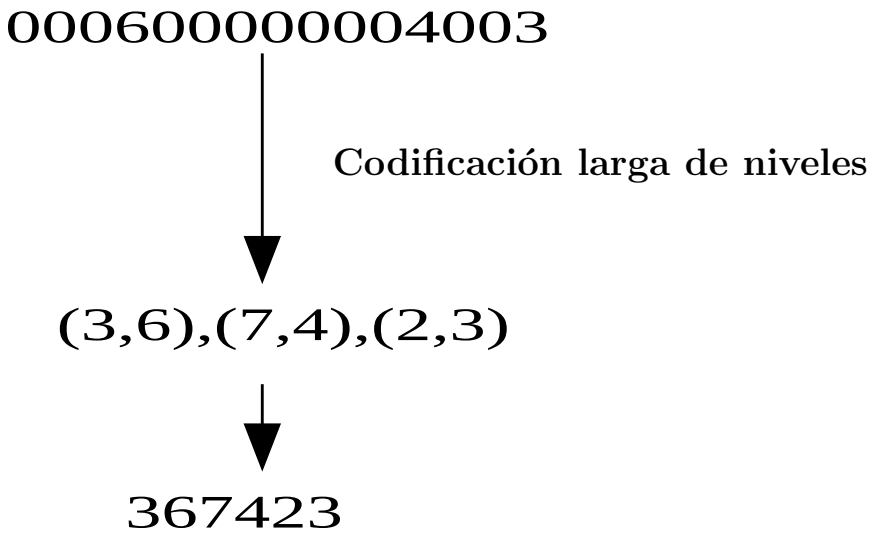


Figura 4.10: Ejemplo de codificación larga de niveles

4.3.8. Codificación VLC

Un codificador de longitud variable mapea símbolos de entrada a una serie de palabras de código VLCs. Cada símbolo mapea una palabra de código y dicha palabra de código varía su longitud pero cada una contiene un número entero de bits [11]. Los símbolos con mayor frecuencia de ocurrencia son representados mediante VLCs cortos, mientras que los símbolos menos comunes son representados con VLCs más largos. Con un mayor número de símbolos asociados a un número mucho menor de códigos permite obtener la compresión de los datos [21].

Para el propósito de este trabajo se utilizan tablas VLC con codificación Huffman preestablecidas que fueron manejadas en trabajos previos y se ajustan al estándar JPEG XR [23] [26]. El modelo estadístico Huffman para la codificación de los coeficientes DC agrupa estos coeficientes dentro de un conjunto de categorás cuyas longitudes se incrementan de forma cuasi logarítmica. Cada una de estas categorías representa un símbolo al cual se le asigna un código Huffman. El modelo tiene un alfabeto de símbolos muy pequeños, sin embargo no es lo bastante óptimo en términos de la entropía. La tabla 4.1 muestra algunos códigos Huffman para los coeficientes DC correspondientes a la luminancia, con precisión de 8 bits [23] [26].

Una vez que se ha clasificado a los coeficientes DC y se ha identificado el código Huffman que le corresponde, a dicho código se le concatenan *SSSS bits adicionales* para identificar el

Categoría SSSS	Coeficiente DC	Código Huffman
0	0	00
1	-1 , 1	010
2	-3,-2,2,3	011
3	-7,...,-4,4,...,7	100
4	-15,...,-8,8,...,15	101
5	-31,...,-16,16,...,31	110
6	-63,...,-32,32,...,63	1110
7	-127,...,-64,64,...,127	11110
8	-255,...,-128,128,...,255	111110
9	-511,...,-256,256,...,511	1111110

Tabla 4.1: Clasificación de coeficientes DC en categorías SSSS y sus códigos Huffman

signo y la magnitud respectiva.
Los pasos a seguir para concatenar los bits adicionales al valor del código Huffman son los siguientes:

- Si el coeficiente DC es positivo, se concatenan los *SSSS bits* menos significativos del mismo.
- Para el caso en que el coeficiente DC es negativo, se concatenan los *SSSS bits* menos significativos del mismo pero en complemento a uno.

Los *SSSS bits* adicionales se concatenan al primer bit menos significativo del código Huffman que le corresponde. La tabla 4.2 muestra las secuencias de bits adicionales para unas cuantas categorías de coeficientes DC. En la tabla se observa que el primero de los bits adicionales es 1 si el coeficiente es positivo y 0 si el coeficiente es negativo. Las tablas de codificación

Categoría SSSS	Coeficiente DC	SSSS bits adicionales
0	0	-
1	-1 , 1	0,1
2	-3,-2,2,3	00,01,10,11
3	-7,...,-4,4,...,7	000,...,011,100,...,111
4	-15,...,-8,8,...,15	0000,...,0111,1000,...,1111
5	-31,...,-16,16,...,31	00000,...,01111,10000,...,11111
6	-63,...,-32,32,...,63	000000,...,011111,100000,...,111111
7	-127,...,-64,64,...,127	0000000,...,0111111,1000000,...,1111111
8	-255,...,-128,128,...,255	00000000,...,01111111,10000000,...,11111111
9	-511,...,256,256,...,511	000000000,...,011111111,100000000,...,111111111

Tabla 4.2: Codificación huffman de los coeficientes DC

Huffman utilizadas para la codificación de la luminancia y crominancia de los coeficientes se

encuentran anexadas a este trabajo. Debido al efecto de la cuantización y predicción, hay muchos coeficientes LP y HP con valor de cero, el codificador Huffman utiliza símbolos que combinan las longitudes de ceros (el número de ceros que preceden a un coeficiente diferente de cero debido a la etapa de escaneo) con las magnitudes de los coeficientes distintos de cero; dado que estas categorías se incrementan logaritmicamente del mismo modo que los coeficientes DC hace que la estrategia de codificación sea muy similar en ambos casos. La tabla 4.3 muestra las categorías SSSS para los coeficientes LP y HP distintos de cero [23] [26] [25] [18]. Ahora, para definir la longitud de ceros se tiene a la categoría RRRR. Como en el

Categoría SSSS	Coeficientes LP y HP
0	0
1	-1 , 1
2	-3,-2,2,3
3	-7,...,-4,4,...,7
4	-15,...,-8,8,...,15
5	-31,...,-16,16,...,31
6	-63,...,-32,32,...,63
7	-127,...,-64,64,...,127
8	-255,...,-128,128,...,255
9	-511,...,256,256,...,511

Tabla 4.3: Clasificación de los coeficientes HP y LP en categorías SSSS

caso de las tablas para codificación de los coeficientes DC, las tablas de coeficientes HP y LP se han desarrollado basándose en el promedio estadístico de un gran conjunto de imágenes con precisión de 8 bits. En la figura 4.11 se muestra una tabla donde la intersección de los valores RRRR y SSSS proporcionan el valor del código Huffman que representa el coeficiente HP o LP distinto de cero y la longitud de ceros que le preceden [18] [23] [26].

Una vez obtenido el código Huffman a éste se le concatenan SSSS bits adicionales, para determinar la magnitud y el signo del coeficiente distinto de cero. El formato para los bits adicionales es el mismo que en la codificación de los coeficientes DC. El valor de la categoría SSSS proporciona el número de bits adicionales requeridos para especificar el signo y la amplitud del coeficiente. Los SSSS bits adicionales pueden ser los bits menos significativos cuando el coeficiente LP o HP es positivo o los bits menos significativos en complemento a uno cuando el coeficiente es negativo. Las tablas de codificación Huffman para los coeficientes LP o HP se pueden consultar en el anexo B.
De la figura 4.11, las entradas marcadas con N/A no aplican en JPEG XR.

Inserción de marcadores

En JPEG XR los datos comprimidos deben de contener marcadores que permitan identificar el inicio y final de la imagen. Entre estos dos marcadores hay típicamente dos o más segmentos marcadores que permiten saber el tamño de la imagen, parámetros de cuantización y modo de predicción, mismos que serán utilizados para la decodificación [28]. Sin embargo,

		SSSS										
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
RRRR	**0**	**EOB**	01	02	03	04	05	06	07	08	09	0A
	1	**N/A**	11	12	13	14	15	16	17	18	19	1A
	2	**N/A**	21	22	23	24	25	26	27	28	29	2A
	3	**N/A**	31	32	33	34	35	36	37	38	39	3A
	4	**N/A**	41	42	43	44	45	46	47	48	49	4A
	5	**N/A**	51	52	53	54	55	56	57	58	59	5A
	6	**N/A**	61	62	63	64	65	66	67	68	69	6A
	7	**N/A**	71	72	73	74	75	76	77	78	79	7A
	8	**N/A**	81	82	83	84	85	86	87	88	89	8A
	9	**N/A**	91	92	93	94	95	96	97	98	99	9A
	10	**N/A**	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA
	11	**N/A**	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA
	12	**N/A**	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA
	13	**N/A**	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA
	14	**N/A**	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA
	15	**N/A**	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA

Figura 4.11: Intersección de las categorías RRRR y SSSS

dada la flexibilidad del estándar es posible utilizar únicamente un marcador *Inicio de imagen (SOI)* y terminar con un marcador *Fin de imagen (EOI)*. A una marcador SOI le siguen los datos comprimidos.

Resumen

En este capítulo se describió el método de compresión de video MJPEG XR y se resaltaron sus ventajas y desventajas con respecto a otros métodos de compresión de video. Por otro lado, dado que el método MJPEG XR está basado en el estándar de compresión JPEG XR, en este capítulo se estudiaron las características más importantes de este estándar para el desarrollo de este trabajo, dentro de las cuales es de importancia mencionar la transformación de la imagen aplicando LBT, la cuantización, la predicción, la exploración adaptable y la codificación entrópica. En el capítulo cinco se aborda la implementación y desarrollo del compresor con la finalidad de evaluar su desempeño.

Capítulo 5

Implementación del compresor MJPEG XR en el DSP

Este capítulo comprende la implementación del compresor de video MJPEG XR, haciendo una descripción detallada de cada etapa y tomando como base teórica los capítulos 1 al 3, así como los algortimos necesarios para poder realizar la compresión en el DSP. Se realizan algunos ejemplos a manera de mostrar el funcionamiento en general del compresor.

5.1. Descripción general del sistema

La implementación de la adquisicón de una secuencia de video está basada en un trabajo previo realizado en el laboratorio [26]. En esta sección se explica de forma general la estructura del sistema.
La adquisición de datos para el compresor MJPEG XR se realiza por medio de un sensor de video y un hardware de alto desempeño. Los *procesadores digitales de señales* (DSPs) son microprocesadores de propósito especial con un conjunto de instrucciones diseñados especialmente para el procesamiento de señales, principalmente para aplicaciones en tiempo real. La familia de procesadores digitales de señales TMS320C6000 de Texas Instruments es adecuada para realizar cálculos numéricos intensivos y es muy utilizada para el procesamiento de imágenes y video [6].
El sistema de aquisición implementado consta de las siguientes etapas:

- Adquisición de video: Se lleva a cabo utilizando un sensor de video, el cual se encarga de adquirir una secuencia de video en formato digital y almacenarla en un buffer de video.

- Compresión de video MJPEG XR: En esta etapa, se aplica el método de compresión MJPEG XR, el cual consiste en comprimir cada imagen almacenada en el buffer de video mediante el estándar JPEG XR, utilizando el DSP6416.

- Descompresión de video MJPEG XR: Para el caso de esta etapa se aplica el proceso inverso a cada una de las etapas que consta el compresor MJPEG XR y se utiliza nuevamente al DSP6416 para realizar ésta tarea.

- Despliegue de la secuencia de video: Se analiza cuadro por cuadro a través del ambiente Code Composer Studio *(CCS)*, que es una interfaz gráfica que permite realizar la programación de la tarjeta.

El esquema de la adquisición se muestra en la figura 5.1.

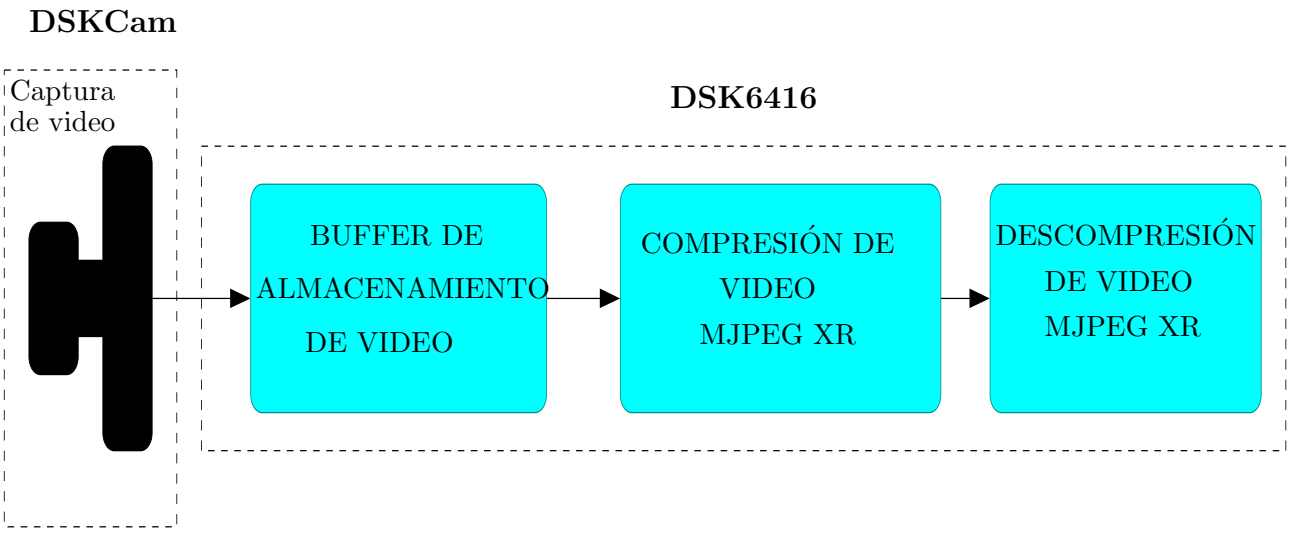


Figura 5.1: Esquema de adquisición y procesamiento

5.2. Procesador Digital de Señales DSP TMS320C6416

Las arquitecturas de los DSPs actuales proporcionan una solución conveniente que combina flexibilidad y potencia computacional. El diseño de la familia de DSPs TMS320C6xxx de la marca Texas Instruments está orientada al procesamiento digital de imágenes. El DSP utilizado para la elaboración del presente trabajo es el C6416 de punto fijo y alto desempeño en su versión de 1 GHz, el cual está basado en la tecnología VelociTI, con arquitectura de *palabra de instrucción muy larga (VLIW)*, desarrollado por Texas Instruments, en la cual varias palabras de datos son capturadas y procesadas simultáneamente. La tecnología VelociTI es una modificación de la arquitectura VLWI que reduce el tamaño de código e incrementa el desempeño cuando las instrucciones están almacenadas en memoria externa [1] [10].
Dentro de las características más importantes con las que cuenta el DSP C6416 se tienen las siguientes:

- Alcanza un desempeño de 8000 *Millones de Instrucciones por Segundo (MIPS)*, con una frecuencia de reloj de 1 GHz, por lo que tiene un ciclo de instrucción de 1 ns.
- Puede ejecutar ocho instrucciones de 32 bits/ciclo y 28 operaciones/ciclo.
- Su núcleo consta de 64 registros de propósito general con una longitud de palabra de 32 bits y ocho unidades funcionales independientes de 2 multiplicadores para un resultado de 32 bits, además de 6 *Unidades Lógicas Aritméticas (ALUs)*.

- Es capaz de procesar 4 *Multiplicaciones Acumulaciones (MACs)* de 16 bits por ciclo para un total de 4000 *Millones de MACs por segundo (MMACS)*, o bien 8 MACS de 8 bits por ciclo para un total de 8000 MMACS.

Además de las características anteriormente mencionadas, el DSP utilizado en el presente trabajo forma parte de una tarjeta de desarrollo, la cual consta de diferentes periféricos que a su vez permiten distintas funcionalidades, entre ellas la capacidad de comunicarse con una cámara de video externa. En las siguientes secciones se hace una descripción general de la tarjeta de desarrollo y de la cámara de video.

5.2.1. Tarjeta de desarrollo DSKC6416T

La tarjeta de desarrollo DSK6416 es una plataforma que permite al usuario evaluar y desarrollar aplicaciones sobre el DSP C6416. Los principales componentes de la tarjeta utilizados en este trabajo, son:

- Un procesador digital de señales DSP TMS320C6416T, con una frecuencia de operación de 1 GHz.
- 16 Mbytes de memoria externa SDRAM.
- Conectores de expansión estándar para el uso de una tarjeta externa.

Una de las características más importantes de la tarjeta es su capacidad de almacenamiento, debido a que tiene una memoria externa SDRAM con 16 Mbytes y de esta manera se puede almacenar 400 cuadros en tamaño QCIF a color, con submuestreo 4:2:0, lo cual es suficiente para almacenar una secuencia de video y aplicar el compresor MJPEG XR con la finalidad de evaluar tiempos de procesamiento. Cabe mencionar que la tarjeta DSK6416 cuenta con conectores de expansión para realizar la comunicación con una tarjeta externa.

5.2.2. Tarjeta de expansión DSKam

La adquisición de video se realiza por medio de la tarjeta de expansión *DSKCam*. Esta tarjeta tiene un sensor de video que puede ser programado para manejar imágenes con resolución VGA o QVGA, y espacio de color RGB y YUV con submuestreo 4:2:2. La tasa de cuadros por segundo capturados por el sensor puede ser de 30 o menor, con una resolución de video VGA, o bien se puede tomar una sola imagen mediante un solo disparo. Para el caso de este trabajo, por conveniencia, se utiliza la resolución QVGA y posteriormente se pasa a una resolución QCIF, mediante software, en un submuestreo 4:2:0, para poder aplicar el proceso de compresión MJPEG XR.

5.3. Proceso de compresión MJPEG XR en el DSP

La descripción del proceso inicia con la adquisición de las imágenes de video por medio del sensor que tiene la DSKCam los canales son posteriormente transferidas y almacenadas en una sección de memoria externa SDRAM del DSP, en un formato QCIF con submuestreo 4:2:0, y sus componenctes YUV se ordenan por separado, de tal manera que las imágenes queden listas para la compresión JPEG XR, como se observa en la figura 5.2 .
Los bloques presentados en la figura 5.2 describen la compresión JPEG XR de cada una de

Figura 5.2: Etapas implementadas

las imágenes de video dando como resultado la compresión MJPEG XR. En las siguientes secciones se describe cada etapa implementada en el DSP.

5.3.1. Preparación de macrobloques

Como se explicó en la sección 4.3.2, es necesario particionar la imagen para poder ser procesada. En el caso del presente trabajo se considera a cada componente de color como un sólo azulejo y dentro de cada azulejo se obtiene un número determinado de macrobloques. Cada macrobloque tiene un total de 16x16 muestras. El orden de codificación de macrobloques es de izquierda a derecha y de arriba abajo. La figura 5.3 muestra el orden de codificación de los macrobloques para la componente de luminancia de una imagen en tamaño QCIF.

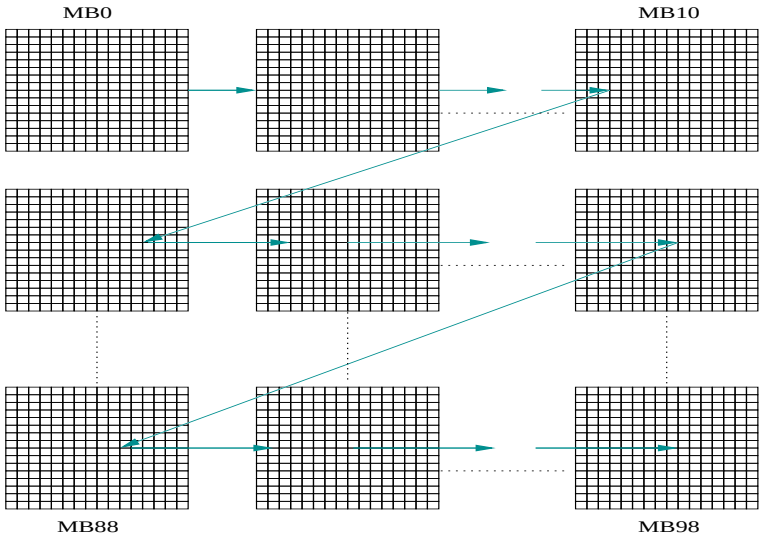


Figura 5.3: Orden de codificación de macrobloques

5.3.2. Algoritmo PCT

Una vez identificados los macrobloques que conforman el azulejo, es posible operar a cada uno de forma independiente. En la tabla 5.1 se muestra un macrobloque tomado como ejemplo de manera aleatoria que corresponde a la componente de luminancia de un cuadro de imagen obtenido por la cámara en formato QCIF y con submuestreo 4:2:0, al cual se le aplicará el algoritmo de la PCT.

178	157	138	165	122	129	167	158	144	142	139	173	129	139	148	128
166	173	168	183	149	127	115	144	154	154	158	162	127	119	119	115
162	163	153	172	132	96	118	142	164	162	159	163	119	127	122	116
158	162	173	144	145	151	147	143	154	147	146	137	117	126	126	121
128	127	127	138	167	159	161	156	139	140	152	138	167	149	152	155
130	122	129	131	168	154	149	148	115	135	128	133	168	159	127	165
131	133	156	167	153	156	166	165	128	132	148	111	150	155	164	148
167	166	161	164	155	136	129	144	135	143	146	168	114	126	161	161
164	131	181	165	153	137	147	143	135	116	144	145	152	138	135	136
175	172	177	186	158	151	152	119	113	108	146	128	168	143	166	158
173	164	170	178	157	150	148	130	113	131	126	143	173	181	165	155
152	124	134	155	131	109	132	131	131	124	97	154	161	176	169	152
129	160	165	126	138	131	161	149	132	142	117	120	130	128	137	138
155	124	167	142	137	146	153	153	121	120	120	115	128	123	129	126
77	62	96	165	152	151	149	152	117	120	118	124	127	134	167	171
171	155	142	142	146	140	145	154	117	126	130	125	173	168	166	159

Tabla 5.1: Macrobloque de ejemplo

Ahora, a este macrobloque se aplica la primera etapa de la PCT mediante el algoritmo descrito en la sección 4.3.3 que consiste en aplicarlo a cada bloque de tamaño 4x4 pixeles contenido dentro del macrobloque. El resultado lo observamos en la tabla 5.2.

654	-16	5	26	546	35	2	-3	615	-24	1	13	500	18	15	-2
11	-2	13	-13	22	-34	2	-18	11	6	5	9	-13	-8	-8	9
2	8	-7	6	-18	4	17	-23	-6	-3	-6	-4	2	2	-3	-3
18	-8	-13	3	-13	8	5	-19	0	-4	-10	-21	-8	3	6	2
570	20	-53	3	617	-13	21	-8	548	33	-9	-6	606	-13	19	22
9	-1	0	4	11	7	-10	20	-14	14	-12	0	9	-4	-17	9
-18	-7	1	-21	9	-2	2	-18	-13	-1	19	17	-11	15	16	5
16	1	-1	0	7	5	3	5	-12	4	4	9	-23	0	-11	33
650	-47	26	-12	562	-20	22	-4	514	10	8	-17	632	-22	-35	21
24	10	-3	4	-1	18	3	13	18	25	-1	12	-4	-3	-6	-1
-15	16	7	0	20	4	-5	-3	-30	-9	-12	0	16	-6	-15	11
-3	21	17	-6	-24	23	-3	8	10	-1	35	-4	2	7	9	-2
545	50	14	-42	590	-7	-5	-8	491	14	3	2	576	24	-52	3
9	-36	14	-68	1	4	1	3	-5	-5	3	3	0	1	0	11
-21	-13	-20	-47	-16	9	2	5	4	-4	-7	0	-17	-4	4	-30
51	17	12	2	-5	8	10	-9	5	-7	-8	16	21	11	-1	-2

Tabla 5.2: Ejemplo de la primera etapa PCT

De la tabla 5.2, es importante notar que los coeficientes de más alta energía se concentran en la esquina superior izquierda de cada bloque. Ahora, se aplica la segunda etapa PCT que consiste en ocupar a los coeficientes de más alta energía y aplicar nuevamente el algoritmo para obtener los coeficientes LP y el coeficiente DC. De la tabla 5.3 se puede notar que el coeficiente de más alta energía se encuentra alojado en la esquina superior izquierda del macrobloque, mientras que los coeficientes LP se encuentran en la esquina superior izquierda de los 15 bloques restantes. Y por último es necesario resaltar que los 240 coeficientes de más baja energía corresponden a los HP.

2304	-16	5	26	-45	35	2	-3	33	-24	1	13	-47	18	15	-2
11	-2	13	-13	22	-34	2	-18	11	6	5	9	-13	-8	-8	9
2	8	-7	6	-18	4	17	-23	-6	-3	-6	-4	2	2	-3	-3
18	-8	-13	3	-13	8	5	-19	0	-4	-10	-21	-8	3	6	2
63	20	-53	3	-45	-13	21	-8	55	33	-9	-6	24	-13	19	22
9	-1	0	4	11	7	-10	20	-14	14	-12	0	9	-4	-17	9
-18	-7	1	-21	9	-2	2	-18	-13	-1	19	17	-11	15	16	5
16	1	-1	0	7	5	3	5	-12	4	4	9	-23	0	-11	33
50	-47	26	-12	44	-20	22	-4	57	10	8	-17	42	-22	-35	21
24	10	-3	4	-1	18	3	13	18	25	-1	12	-4	-3	-6	-1
-15	16	7	0	20	4	-5	-3	-30	-9	-12	0	16	-6	-15	11
-3	21	17	-6	-24	23	-3	8	10	-1	35	-4	2	7	9	-2
36	50	14	-42	-35	-7	-5	-8	99	14	3	2	45	24	-52	3
9	-36	14	-68	1	4	1	3	-5	-5	3	3	0	1	0	11
-21	-13	-20	-47	-16	9	2	5	4	-4	-7	0	-17	-4	4	-30
51	17	12	2	-5	8	10	-9	5	-7	-8	16	21	11	-1	-2

Tabla 5.3: Segunda etapa PCT

5.3.3. Cuantización

Después de aplicar la transformación al macrobloque de ejemplo, los coeficientes de menor energía son agrupados y pueden ser removidos por medio de la cuantización. De acuerdo a lo expresado en la sección 3.3, la cuantización utilizada en la implementación es escalar y consiste solamente en realizar una división de cada coeficiente entre un parámetro de cuantización definido y tomar la parte entera del resultado obtenido. Como ejemplo se elige un QP = 8 y se sustituye el valor del coeficiente de mayor energía de la tabla 5.3 en la ecuación (3.17) quedando como resultado la ecuación 5.1.

$$S_q = Parte\ entera\left(\frac{2304}{8}\right) = 288 \tag{5.1}$$

Aplicando el procedimiento anterior para el resto de coeficientes que pertenecen al macrobloque, queda como resultado la tabla 5.4. Del resultado anterior se observa para el caso del bloque que contiene al coeficiente DC que aparecen seis ceros, lo cual es aprovechado en la etapa de codificación entrópica. Hay que hacer notar que en esta etapa es donde se obtienen pérdidas de tal forma que se afecta a la calidad de la imagen.

288	-2	0	3	-5	4	0	0	4	-3	0	1	-5	2	1	0
1	0	1	-1	2	-4	0	-2	1	0	0	1	-1	-1	-1	1
0	1	0	0	-2	0	2	-2	0	0	0	0	0	0	0	0
2	-1	-1	0	-1	1	0	-2	0	0	-1	-2	-1	0	0	0
7	2	-6	0	-5	-1	2	-1	6	4	-1	0	3	-1	2	2
1	0	0	0	1	0	-1	2	-1	1	-1	0	1	0	-2	1
-2	0	0	-2	1	0	0	-2	-1	0	2	2	-1	1	2	0
2	0	0	0	0	0	0	0	-1	0	0	1	-2	0	-1	4
6	-5	3	-1	5	-2	2	0	7	1	1	-2	5	-2	-4	2
3	1	0	0	0	2	0	1	2	3	0	1	0	0	0	0
-1	2	0	0	2	0	0	0	-3	-1	-1	0	2	0	-1	1
0	2	2	0	-3	2	0	1	1	0	4	0	0	0	1	0
4	6	1	-5	-4	0	0	-1	12	1	0	0	5	3	-6	0
1	-4	1	-8	0	0	0	0	0	0	0	0	0	0	0	1
-2	-1	-2	-5	-2	1	0	0	0	0	0	0	-2	0	0	-3
6	2	1	0	0	1	1	-1	0	0	-1	2	2	1	0	0

Tabla 5.4: Cuantización

5.3.4. Predicción

De acuerdo a la teoría expuesta en la sección 4.3.5, esta etapa se encarga de mejorar la compresión ocupando los coeficientes vecinos entre cada macrobloque, comparándolos y guardando la menor diferencia existente entre ellos, de tal forma que se remueven a los coeficientes redundantes que fueron resultado de los coeficientes transformados y a su vez cuantizados [14]. Continuando con el macrobloque de ejemplo, en la figura 5.4 se muestra localizado el coeficiente DC marcado con azul (288) y a los coeficientes DC de los macrobloques vecinos identificados de color amarillo (295, 269 y 248). Tomando los coeficientes respectivos se obtiene la tabla 5.5.

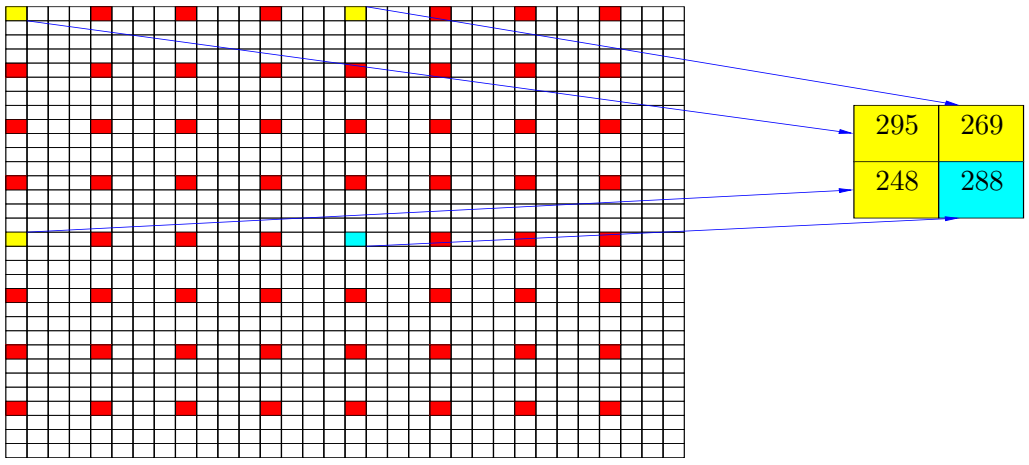


Figura 5.4: Ejemplo de tres macrobloques

295	269
248	288

Tabla 5.5: Predicción entre macrobloques

El algoritmo de la predicción consiste en obtener las diferencias entre los coeficientes vecinos al macrobloque de ejemplo para determinar la menor de ellas y por último decidir con cuál coeficiente se realizará la diferencia con el coeficiente del macrobloque de ejemplo. Por lo tanto, de la tabla 5.5 se obtiene que el vecino más parecido es 295 y se dice que la predicción proviene de la parte superior izquierda [14] [28]. El resultado de realizar la diferencia se muestra en la tabla 5.6. Para los coeficientes LP y HP la dirección de la predicción queda

295	269
248	-7

Tabla 5.6: Diferencia entre coeficientes DC

definida de acuerdo al resultado del coeficiente DC y el procedimiento es muy similar.

5.3.5. Exploración adaptable

La exploracióna adaptable, de acuerdo a la sección 4.3.6, consiste en reordenar los coeficientes y pasarlos de un arreglo de dos dimensiones a un arreglo unidimensional. El esquema de la figura 5.5 muestra el proceso de reordenamiento de los coeficientes para cada bloque de acuerdo a la posición de los índices que los identifican dentro del arreglo.

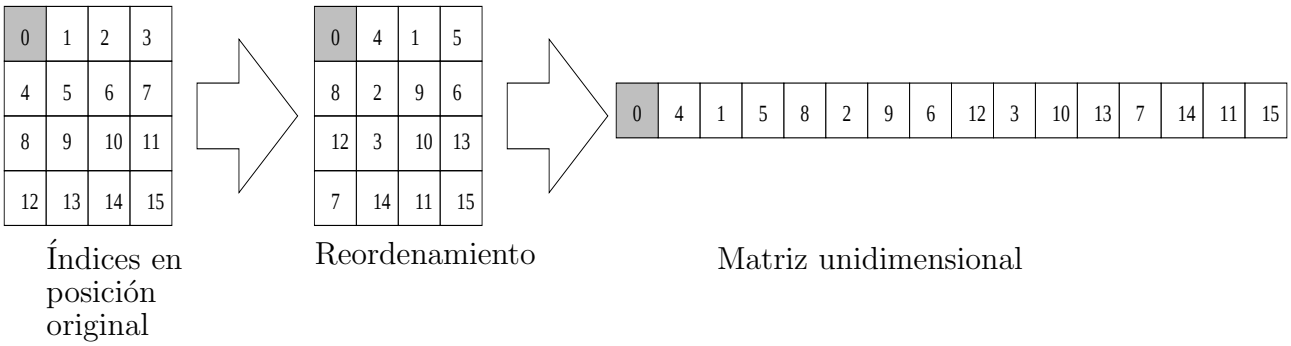


Figura 5.5: Reordenamiento de los coeficientes

Para ejemplificar la exploración, se toma al bloque mostrado en la tabla 5.7 y se le aplica el proceso mostrado en la figura 5.5, quedando como resultado la tabla 5.8.

Por último, el nuevo arreglo matricial de 4x4 es necesario colocarlo como un arreglo unidimensional de 1x16, tal como se muestra en la tabla 5.9. Hay que destacar que los ceros son agrupados junto con los coeficientes distintos de cero.

-7	-2	0	3
1	0	1	-1
0	1	0	0
2	-1	-1	0

Tabla 5.7: Bloque a ordenar

-7	0	0	1
-2	3	-1	2
1	1	0	-1
0	0	-1	0

Tabla 5.8: Reordenamiento del bloque

-7	0	0	1	-2	3	-1	2	1	1	0	-1	0	0	-1	0

Tabla 5.9: matriz unidimensional

5.3.6. Codificador Entrópico

La última de las etapas corresponde al codificador entrópico donde comienza la compresión de la información. Tomando como ejemplo el resultado mostrado en la tabla 5.9, se aplica el codificador entrópico identificando al primer coeficiente DC (sombreado en gris) que corresponde al valor de -7, ahora se obtiene su categoría SSSS y los bits adicionales de acuerdo a la tabla 5.10. Así, se obtiene que el coeficiente pertenece a la categoría 3 y por lo

Categoría SSSS	Coeficiente DC	Código Huffman	SSSS bits adicionales
...	...	...	...
2	-3,-2,2,3	011	00,01,10,11
3	-7,...,-4,4,...,7	100	000,...,011,100,...,111
...	...	...	...

Tabla 5.10: categorías SSSS y bits adicionales de los coeficientes DC

tanto su código Huffman es 100. Además, los bits adicionales que se concatenan al código Huffman son los tres bits del valor 7 pero en complemento a uno, es decir 000, por lo que el resultado final será el valor de *100000*. (Las tablas completas se muestran en las tablas 4.1 y 4.2 de la sección 4.3.8).
Una vez codificado el coeficiente DC, se procede a codificar los coeficientes LP y HP dentro de cada macrobloque.

Codificación de los coeficientes LP y HP

De acuerdo al procedimiento descrito en la sección 4.3.8, el codificador Huffman combina las *longitudes de ceros* (definidas con el valor RRRR) con las *categorías SSSS* de los coeficientes distintos de cero. Las longitudes de ceros indican el número de ceros que preceden a un coeficiente distinto de cero en la matriz unidimensional.
En el arreglo de la tabla 5.9, el primer coeficiente HP diferente de cero es 1 el cual corresponde a la categoría SSSS=1 de acuerdo a la tabla 5.11, y es precedido por dos coeficientes de valor cero, entonces se tiene un valor de longitud de ceros RRRR=2, por tal motivo le corresponde el código base 0x21 localizado en la fila 2 y la columna 1 de la tabla mostrada en la figura 5.6. Al código base 0x21 le corresponde el código Huffman 11100 de la tabla 5.12*(Las tablas completas se encuentran en el anexo B).*
Después de obtener el valor del código Huffman, se procede a concatenar los bits adicionales.

Categoría SSSS	Coeficientes LP y HP
...	...
1	-1 , 1
2	-3,-2,2,3
...	...

Tabla 5.11: Clasificación de los coeficientes HP y LP en categorías SSSS

Categoría	Longitud de código	Código Huffman
0x00 (EOB)	4	1010
0x01	2	00
0x02	2	01
0x03	3	100
...	...	...
0x21	5	11100

Tabla 5.12: Códigos base para los coeficientes HP de luminancia

Como el valor del coeficiente diferente de cero es 1, el cual pertenece a la categoría SSSS=1, al código Huffman se le concatena el bit menos significativo del valor 1 que corresponde a 1. Por lo tanto, el primer coeficiente HP codificado es *111001.*

El segundo coeficiente HP diferente de cero es -2, al cual le corresponde la categoría SSSS=2 y no es precedido por ceros, por tal motivo, le corresponde la el valor 0x02 en la tabla 5.12. El código Huffman proporcionado por la tabla es el valor 01. Como el valor del coeficiente diferente de cero es -2, el cual pertenece a la categoría SSSS=2, y es un valor negativo, al código Huffman se le concatenan los 2 bits menos significativos del valor 2 en binario en complemento a uno, teniendo como resultado el valor 01. Por lo tanto, el segundo coeficiente HP codificado es 0101.
Siguiendo el mismo procedimiento anterior para los 7 coeficientes HP distintos de cero

		SSSS										
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
RRRR	**0**	**EOB**	01	02	03	04	05	06	07	08	09	0A
	1	**N/A**	11	12	13	14	15	16	17	18	19	1A
	2	**N/A**	21	22	23	24	25	26	27	28	29	2A
	...	**...**	...	...	...	...	...	...	...	...	...	...

Figura 5.6: Codificación Huffman de los coeficientes HP y su longitudes de ceros

restantes se obtiene la tabla 5.13 donde se muestra el código Huffman concatenado a los bits adicionales de cada coeficiente distinto de cero.

Finalmente, el código correspondiente al primer bloque queda definido como la siguiente

Coeficiente distinto de cero	Código Huffman + SSSS bits
-7	100000
1	111001
-2	0101
3	10011
-1	000
2	0110
1	001
1	001
-1	11000
-1	111000
EOB	1010

Tabla 5.13: Códigos de coeficientes distintos de cero

secuencia *100000 111001 0101 10011 000 0110 001 001 11000 111000 1010*. A este código también se le conoce como *flujo de bits* [28].

Para determinar la compresión obtenida sobre le bloque, hay que considerar que un bloque de 4 x 4 requiere de un total de 128 bits antes de codificar, mientras que la secuencia codificada solo requiere de 49 bits. Si comparamos a los bits originales con los bits codificados se obtiene $(49/128)100\,\% = 38\,\%$. Por último, para indicar el inicio de la imagen es necesario concatenar el código SOI al cual le corresponde el valor 0xFFD8 y el final de la imagen se concatena el código EOI con un valor de 0xFFD9 [23] [26].

5.3.7. Descompresión de video MJPEG XR

La recuperación de la secuencia de video se realiza por medio de la descompresión. Este proceso inicia por la decodificación de la secuencia de bits generada en la etapa del codificador entrópico, obteniendo primeramente los coeficientes (DC,LP,HP) resultantes de predicción de cada bloque. Después de obtener dichos coeficientes, se aplica el algoritmo inverso del escaneo adaptable reordenando nuevamente los coeficientes para formar un arreglo de dos dimensiones. El nuevo arreglo forma un bloque que a su vez mantiene una posición determinada dentro de un macrobloque. Así, se determinan los macrobloques vecinos y se aplica el mismo algoritmo de predicción pero ahora se calcula la suma del coeficiente tomado como actual con el coeficiente que coincide en la dirección de la predicción. Posterior a la predicción, los coeficientes que se encuentran en el dominio de la frecuencia son tranformados al dominio espacial aplicando la transformada PCT inversa, primero a los coeficientes LP y DC y finalmente se aplica nuevamente la transformada PCT inversa por bloque. De esta manera se llega a los pixeles originales en el espacio de color YCrCb obtenidos por la cámara para cada cuadro de imagen. En el esquema mostrado en la figura 5.7 se observa la secuencia del proceso de la descompresión.

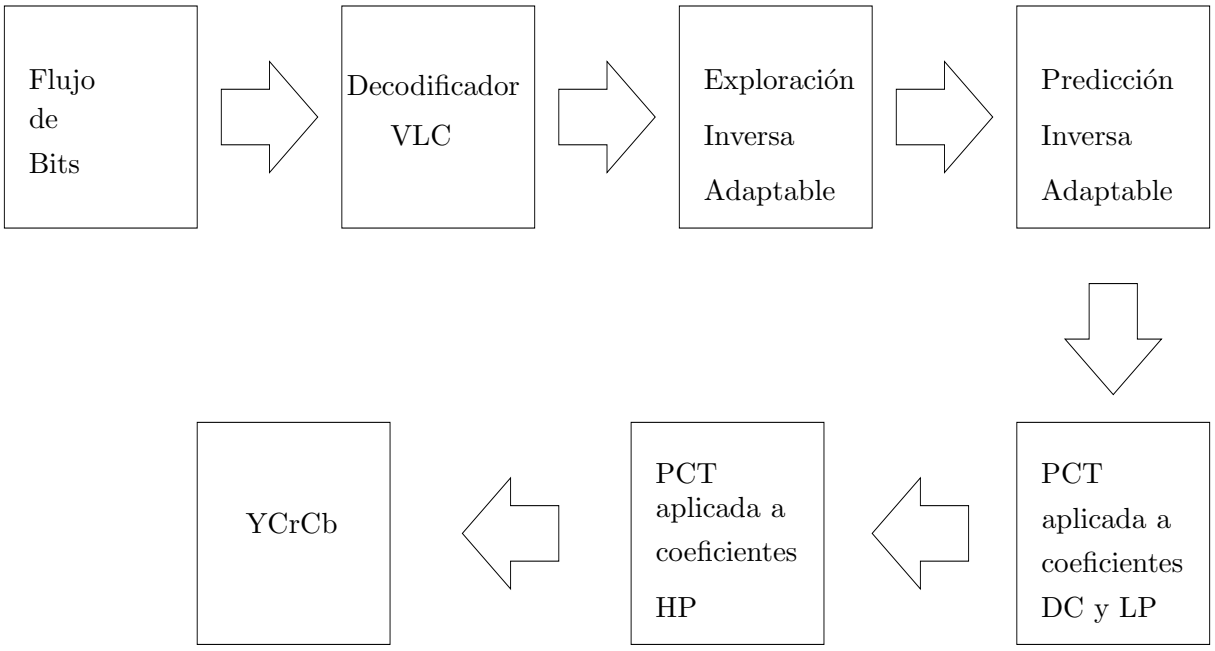


Figura 5.7: Esquema de la descompresión

Resumen

En este capítulo se explicó cómo se implementó la compresión de video MJPEG XR en el DSP, describiendo y ejemplificando cada una de las etapas de la compresión, desde la aplicación de la LBT hasta el codificador entrópico. También se hizo notar que una de las características principales de la LBT es la concentración de la energía en pocos coeficientes y que para llegar a este resultado se toman los bloques vecinos. Por otro lado, la cuantización es la etapa en donde se obtienen pérdidas y la predicción adaptable se encarga de disminuir la redundancia por cuadro de imagen. Por último, la exploración adaptable se encarga de reordenar los coeficientes y preparalos para la etapa de codificación entrópica donde se obtiene la compresión sin pérdidas. Cabe mencionar que en esta sección también se describió el proceso inverso de cada etapa ocupada para la compresión.
En el capítulo 6 se presenta el análisis de los resultados obtenidos en este trabajo.

Capítulo 6

Evaluación y resultados del compresor MJPEG XR

En este capítulo se presentan los resultados obtenidos al adquirir varias secuencias de imágenes en QCIF, en el espacio YCrCb en formato de submuestreo 4:2:0 así como en formato QCIF en escala de grises. Además, se muestran los resultados de la compresión y descompresión MJPEG XR sobre una secuencia de video adquirida. La evaluación se realiza tanto de manera subjetiva, comparando la calidad visual para diferentes secuencias de video, como objetiva obteniendo curvas características de la relación de compresión.
El código utilizado en este capítulo tanto para la compresión, descompresión y despliegue de resultados puede consultarse en el apéndice C.

6.1. Evaluación de la calidad visual

Para evaluar la calidad visual se adquirieron dos secuencias de video de 24 cuadros de tamaño QCIF a color con formato de submuestreo 4:2:0 y se les aplicó la compresión MJPEG XR. La calidad de la imagen está directamente relacionada con la variación del factor de cuantización QP definido en la sección 4.3.4. En esta sección se hace una clasificación de una secuencia de video en buena, regular y mala de acuerdo a la percepción que se obtuvo de la misma.

Secuencia del refresco

La primera secuencia de video obtenida para realizar la evaluación del compresor consiste en pasar un refresco de lata frente a la cámara y comparar de manera subjetiva la secuencia original con tres secuencias recuperadas para un QP distinto. La figura 6.1 muestra la secuencia de video original, donde las figuras 6.1a, 6.1b y 6.1c representan al cuadro 1, cuadro 15 y cuadro 23 respectivamente de los 24 cuadros que forman a la secuencia de video completa. Por otro lado, las figuras 6.2, 6.3 y 6.4 representan los cuadros de imagen recuperados de la secuencia de video original con parámetros de cuantización que les corresponden los valores de 7, 10 y 15 de manera respectiva.

(a) cuadro 1 (b) cuadro 15 (c) cuadro 23

Figura 6.1: Primera secuencia de video original

(a) cuadro 1 (b) cuadro 15 (c) cuadro 23

Figura 6.2: Secuencia recuperada con QP = 7

(a) cuadro 1 (b) cuadro 15 (c) cuadro 23

Figura 6.3: Secuencia recuperada con QP = 10

(a) cuadro 1 (b) cuadro 15 (c) cuadro 23

Figura 6.4: Secuencia recuperada con QP = 17

Clasificación por calidad visual

Comparando la secuencia de video original, mostrada en la figura 6.1, con la secuencia recuperada, mostrada en la figura 6.2, no se observan muchos cambios perceptibles entre ambas.

Por tal motivo, se puede decir que la calidad de la secuencia de video es buena. Sin embargo, si comparamos nuevamente a la figura 6.1 con la mostrada en la figura 6.3 comienzan a notarse la aparición de algunos artefactos sobre todo en donde existen cambios de iluminación, como ejemplo más notorio se puede observar en la pared que se encuentra de lado derecho de la lata mostrado en la 6.3c. De lo anterior, se concluye que la secuencia de la figura 6.3 es de calidad regular. Al hacer la última comparación entre la secuencia original con la secuencia de la figura 6.4 se observan varios cambios notorios de lo cual se puede concluir que es de mala calidad.

Secuencia de video diferencia

Con el fin de sustentar la clasificación anterior, se obtuvieron secuencias de video diferencia. La secuencia de video diferencia se obtiene de la resta entre los pixeles de cada cuadro de imagen de la secuencia de video original con la secuencia de video recuperada. La expresión utilizada para obtener la diferencia entre cada cuadro de imagen es representada en la ecuación (6.1).

$$C_{dif}(x, y) = C_{or}(x, y) - C_{rec}(x, y) \tag{6.1}$$

Donde $C_dif(x, y)$ representa el pixel de la imagen resultante de la diferencia, $C_{or}(x, y)$ es el pixel de la imagen original y $C_{re}(x, y)$ es el pixel de la imagen recuperada. La ecuación 6.1 es aplicada a cada cuadro de la secuencia de video en el espacio de color RGB (sección 2.3). La figura 6.5 muestra la secuencia de video diferencia entre la secuencia de video original con la secuencia de video que tiene un parámetro de cuantización QP con valor 10. En las figuras 6.5a, 6.5b y 6.5c se observan imágenes visualmente poco perceptibles, lo cual implica que la diferencia entre la secuencia de video original con la secuencia recuperada es muy pequeña.
La figura 6.6 muestra a otra secuencia de video diferencia entre la secuencia de video original

(a) cuadro 1 (b) cuadro 15 (c) cuadro 23

Figura 6.5: Secuencia recuperada con QP = 10

con una secuencia de video que tiene un parámetro de cuantización con valor QP de 17. De las figuras 6.6a, 6.6b y 6.6c se observan imágenes más perceptibles si las comparamos con las figuras 6.5a, 6.5b y 6.5c. Lo anterior permite concluir que la diferencia entre la secuencia reconstruida de la figura 6.6 con la secuencia original es considerable.

(a) cuadro 1 (b) cuadro 15 (c) cuadro 23

Figura 6.6: Secuencia recuperada con QP = 17

6.2. Medida de distorsión y relación de compresión

Además de la evaluación subjetiva que se realizó, también se obtuvo una evaluación de la pérdida de información o medida de la distorsión [5]. Una de las medidas más utilizadas para tal propósito es la *relación señal a ruido pico* ($PSNR$) que es expresada por la ecuación (6.2) [5], [28].

$$PSNR = 20log_{10}\left(\frac{2^n}{\sqrt{MSE}}\right) \tag{6.2}$$

Donde n es el número de bits por pixel de la imagen y MSE es el *error cuadrático medio* que es descrito por la ecuación (6.3) [5], [28].

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}|Y(i,j) - \hat{Y}(i,j)|^2 \tag{6.3}$$

Donde $Y(i,j)$ representa un pixel de la componente luminancia para un solo cuadro de imagen de la secuencia de video original, mientras que $\hat{Y}(i,j)$ representa un pixel de la componente luminancia para un solo cuadro de imagen de la secuencia de video recuperada y MN representa el tamaño de la imagen lo que es equivalente a la expresión MxN. En este trabajo el valor de n es de 8 bits, con tamaño de cuadro de imagen MxN es 176 x 144 en formato QCIF y submuestreo 4:2:0.
Para hacer un análisis entre cada cuadro de imagen se obtuvo el PSNR para cada uno de los 24 cuadros que pertenecen a cada una de las secuencias de video recuperadas con parámetros de cuantización QP=7, QP=10 y QP=17, la gráfica resultante se muestra en la figura 6.7. A su vez se obtuvo el PSNR promedio para cada uno de los párametros de cuantización así como la variación existente que se muestra en la tabla 6.1.

QP	PSNR promedio [dB]	Variación [dB]
7	48.66	± 1.5
10	30.47	± 2
17	23.48	± 2

Tabla 6.1: PSNR promedio y variación

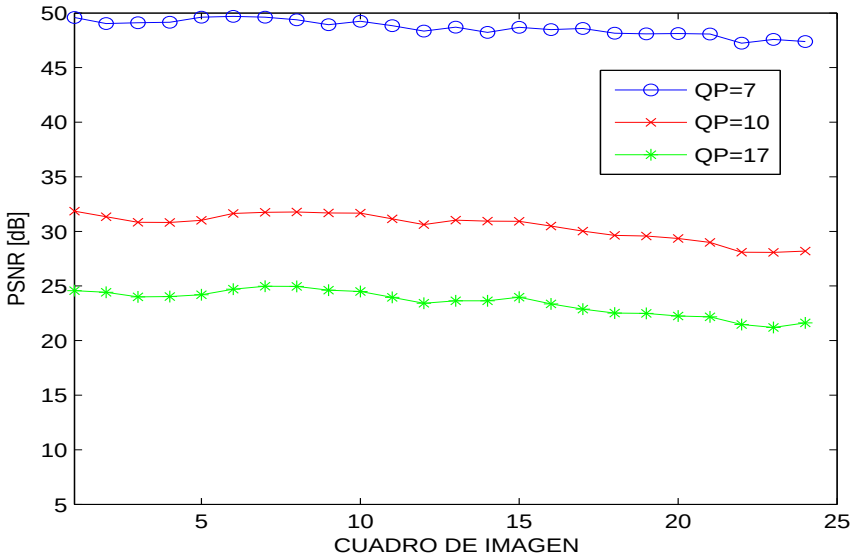


Figura 6.7: PSNR para tres secuencias de video con diferente parámetro de cuantización

Tomando como referencia los resultados obtenidos tanto de la gráfica 6.7 y la tabla 6.1 se pueden obtener las siguientes observaciones:

- La distorición entre cuadro y cuadro de imagen para una misma secuencia de video no varía de manera considerable.
- El parámetro de cuantización QP=7 mantiene muy poca distorsión dado que el PSNR es mayor si se compara con el PSNR de QP=10 y QP=17. Por otro lado, la mayor distorsión se obtiene para QP=17.

De acuerdo a las observaciones anteriores se pueden relacionar la calidad visual con la medida de distorsión de acuerdo a la tabla 6.2 [12]. Además de evaluar la distorción se obtuvo la

QP	PSNR promedio [dB]	Calidad visual
7	48.66	Buena
10	30.47	Regular
17	23.48	Mala

Tabla 6.2: PSNR promedio y calidad visual

relación existente entre la totalidad de bytes que ocupa la secuencia de video original y los bytes resultantes de la compresión. Dicha relación se obtiene mediante la ecuación 6.4 [26].

$$\%FC = \frac{BC}{BO} x100\,\% \tag{6.4}$$

Donde FC es el factor de compresión, BC son los bytes de compresión y BO son los bytes originales de la secuencia de video.

La tabla 6.3 muestra el factor de compresión para toda la secuencia de video completa respecto a cada parámetro de cuantización. Para finalizar, se obtuvo la gráfica mostrada

QP	Espacio Original [Bytes]	Espacio Comprimido [Bytes]	Factor de compresión
7	912384	138533	15.18 %
10	912384	116237	12.74 %
17	912384	89274	9.78 %

Tabla 6.3: Factor de compresión

en la figura 6.8 para relacionar el factor de compresión con el PSNR, de la cual se puede observar que una distorsión aceptable es de 30 [dB] con una compresión del 11 % y para este caso coincide con un parámetro de cuantización QP=12. Lo anterior implica que para un parámetro de cuantización mayor a QP=12 se obtienen secuencias de video de regular a mala calidad.

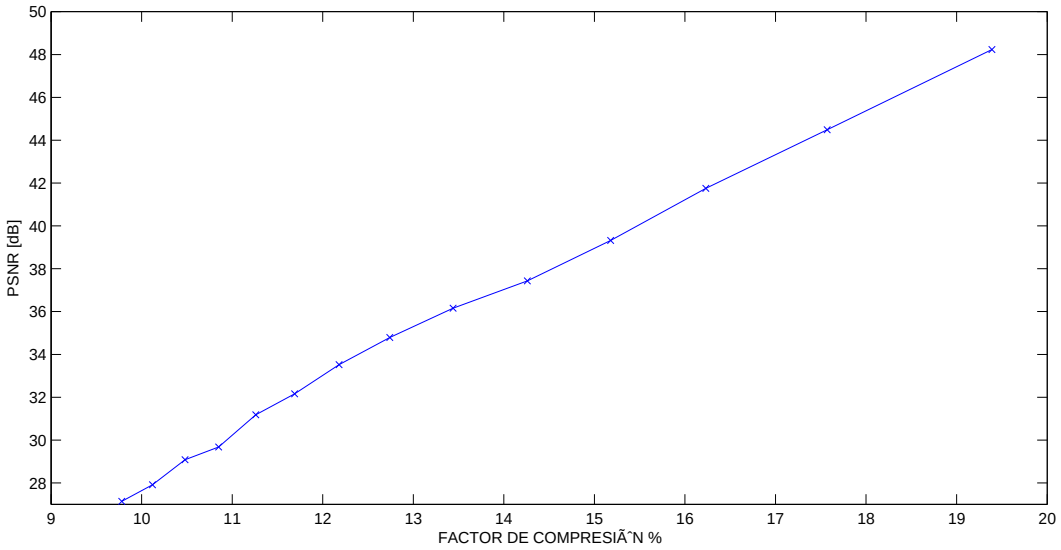


Figura 6.8: Relación del PSNR con el factor de compresión

Comparación de rendimiento

Como se especificó en sección 5.1 el sistema de adquisición de video está basado en un trabajo previo en el cual se implementó un compresor MJPEG [26]. Por tal motivo, en esta sección se hace una comparación entre el compresor MJPEG XR y el compresor MJPEG, obteniendo las características de cada uno.
La figura 6.9 muestra la secuencia original utilizada para probar el compresor MJPEG que consiste en pasar a un libro frente a la cámara de izquierda a derecha. A esta secuencia se le aplicó el compresor MJPEG, ver figura 6.10, y a su vez el compresor MJPEG XR,

ver figura 6.11. Ambas secuencias recurperadas en sus respectivo descompresor conservan características similares, PSNR < 31 dB y de calidad regular.

De la figura 6.10a y 6.11a se observa la aparición de artefactos pero con distinto aspecto.

(a) cuadro 1 (b) cuadro 15 (c) cuadro 23

Figura 6.9: Secuencia original del libro

(a) cuadro 1 (b) cuadro 15 (c) cuadro 23

Figura 6.10: Secuencia recuperada MJPEG

(a) cuadro 1 (b) cuadro 15 (c) cuadro 23

Figura 6.11: Secuencia recuperada MJPEG XR

En MJPEG XR de acuerdo a la sección 4.3.3 las unidades a codificar son de tamaño 16 x 16 mientras que en MJPEG las unidades son de tamaño 8 x 8 [26] lo que hace relativamente menos perceptibles la aparición de artefactos después de recuperar la secuencia MJPEG.
La tabla 6.4 muestra la comparación del factor de compresión así como del PSNR para la secuencia recuperada MJPEG y MJPEG XR.

Codificador	Fator de compresión	PSNR [dB]
MJPEG	5.37 %	21
MJPEG XR	11.81 %	30

Tabla 6.4: Distorsión y compresion en MJPEG y MJPEG XR

6.3. Tiempos de procesamiento para la compresión

Una vez que se ha realizado un análisis de la calidad de las secuencias de video resultantes así como de la compresión, es muy importante evaluar el desempeño del codificador para un cuadro de imagen en formato QCIF con submuestreo 4:2:0 y para una secuencia de video completa de 24 cuadros. Por tal motivo, en la tabla 6.5 se muestra el tiempo requerido por el DSP para comprimir un solo cuadro de imagen de 176 x 144.

Factor de cuantización	Tiempo de compresión [ms]
QP=4	25
QP=9	22

Tabla 6.5: Tiempo de compresión por cuadro de imagen

Por otro lado, se obtiene también el tiempo de compresión para una secuencia de video completa en el DSP. La tabla 6.6 muestra el tiempo para diferentes secuencias de video con factores de cuantización distinta.

Factor de cuantización	Tiempo de compresión [ms]
QP=4	600
QP=9	528

Tabla 6.6: Tiempo de compresión para una secuencia de video

Para una posterior mejora a la implementación realizada, se presentan los tiempos de procesamiento para cada una de las etapas que integran al compresor MJPEG XR, por cuadro de imagen y los resultados se muestran en la tabla 6.7.

De acuerdo a la tabla 6.7 la etapa que demanda más tiempo de procesamiento es la PCT en su primera etapa, mientras que la etapa de predicción es la que requiere de menos recursos del DSP. De forma similar se obtienen los tiempos equivalentes para la descompresión MJPEG XR.

Tanto en la tabla 6.8 como en la tabla 6.9 se observa que el tiempo requerido para descomprimir un cuadro de imagen y video no varía de forma significativa si la comparamos con la tabla 6.5 y la tabla 6.6 respectivamente. De acuerdo a la tabla 6.10 la etapa que requiere más

Factor de cuantización	Tiempo de compresión [ms]
Primera etapa PCT	11
Segunda etapa PCT	1.6
Cuantización	3.4
Predicción	0.93
Escaneo adaptable	5.0
VLC	2.7

Tabla 6.7: Tiempo de compresión de cada etapa por cuadro de imagen

Factor de cuantización	Tiempo de compresión [ms]
QP=4	30
QP=9	26

Tabla 6.8: Tiempo de la descompresión por cuadro de imagen

Factor de cuantización	Tiempo de compresión [ms]
QP=4	720
QP=9	624

Tabla 6.9: Tiempo de la descompresión para una secuencia de video

Factor de cuantización	Tiempo de compresión [ms]
Primera etapa PCT	13
Segunda etapa PCT	2.6
Cuantización	2.4
Predicción	0.93
Ecaneo adaptable	5.0
VLC	5.7

Tabla 6.10: Tiempo de la descompresión de cada etapa por cuadro de imagen

tiempo de procesamiento en la descompresión por cuadro de imagen es la PCT en su primera etapa, mientras que la predicción no requiere de mucho tiempo de procesamiento. Se puede destacar que la etapa del decodificador VLC es la que aumenta el número de operaciones de manera considerable, incrementando el tiempo de proceso de 2.7 ms a 5.7 ms por cuadro de imagen, tal como se observa si comparamos la tabla 6.7, con la tabla 6.10 respectivamente. Finalmente, la tabla muestra una comparación del tiempo total de compresión entre el codificador MJPEG [26] y el codificador MJPEG XR para una secuencia de video de 24 cuadros, en formato QCIF con submuestreo 4:2:0. De acuerdo a la tabla 6.11 se observa una notable mejora en el tiempo de procesamiento obtenido con el codificador MJPEG XR implementado

Codificador	Tiempo [s]
MJPEG	2.731
MJPEG XR	0.6

Tabla 6.11: Tiempo de compresión MJPEG y MJPEG XR

en este trabajo comparandolo con MJPEG.

Resumen

En este capítulo se presentó la manera de calcular el tiempo de procesamiento por etapa del codificador MJPEG XR, por cuadro de imagen en formato QCIF y con submuestreo 4:2:0 de una secuencia de video así como de los 24 cuadros en total. Se hizo un análisis de la calidad visual, haciendo una clasificación subjetiva y midiendo la distorsión de la secuencia de video. También se calculó el factor de compresión y se relacionó con la distorsión obtenida. Se compararon los resultados con la implementación del codificador MJPEG y se observaron notables mejoras en el tiempo de procesamiento.

Capítulo 7

Conclusiones

En este trabajo se implementó un sistema de compresión de video utilizando el codificador MJPEG XR, haciendo uso de una cámara de video digital y de la tarjeta de desarrollo DSP C6416 (DSK6416), el cual se permite destacar los siguientes puntos:

- Se obtuvieron tiempos de compresión adecuados para ocupar el sistema en tiempo real, en formato QCIF, dado que el tiempo total de procesamiento para secuencia de video fue de 600 [ms], lo cual implica que se puede obtener una tasa mayor a 24 cuadros por segundo.

- A su vez, se comparó la implementación actual de MJPEG XR con una implementación anterior, realizada en el laboratorio, de MJPEG obteniendose una notable mejora en tiempo procesamiento debido a que MJPEG XR requiere 30 % menos recursos para completar la compresión de video.

- El tiempo necesario para realizar la descompresión no aumenta de manera significativa respecto a la compresión y las etapas con mayor demanda de recursos son la PCT y el codificador entrópico VLC.

- Fue evaluada la calidad visual, compresión y distorsión para un cuadro de imagen y para los 24 cuadros de la secuencia de video en formato QCIF con submuestro 4:2:0.

- La distorsión se mantiene casi constante para cada uno de los cuadros de imagen de una misma secuencia de video, obteniendo solamente una variación de $\pm$ 2 [dB] aproximadamente.

- Se obtuvo una calidad visual buena para un PSNR de 48.66 [dB], regular con un PSNR de 30.47 [dB] y mala para un PSNR de 23.48 [dB].

- La compresión mínima alcanzada, considerando una distorsión de 30 [dB] de calidad visual regular, fue del 11.81 %.

- Se obtuvo una secuencia de video diferencia para resaltar los cambios que resultaron de la recuperación de la secuencia de video original, de lo cual se concluye que entre mayor sea la distorsión se logran percibir detalles de la secuencia de video original.

- La finalidad del sistema a futuro es que se pueda adquirir, comprimir, transmitir, y en otro módulo similar descomprimir y visualizar video a color en tiempo real, de tal forma que con el sensor de video y el DSP se adquiera y comprima video de forma independiente, y se transmita el video por una red TCP/IP, y en el otro lado del sistema, una PC reciba el video comprimido, lo descomprima y lo despliegue en el monitor, todo esto en tiempo real, y tratando de que el tamaño de video sea lo más grande posible, siendo el tamaño QVGA una buena opción.

Apéndices

Apéndice A

Glosario

ABREVIATURA	SIGNIFICADO
ALU:	Unidad Aritmética Lógica
ATSC:	Advanced Television Standards Commitee
BC:	Bytes de Compresión
BO:	Bytes Originales
CA/D:	Convertidor Analógico Digital
CIF:	Common Intermediate Format
CLL:	Codificación de Longitud Larga
dB:	decibel
DSP:	Procesador Digital de Senales
DVB:	Digital Video Broadcasting
DWT:	Discrete Wavelet Transform
EOI:	End Of Image
HDTV:	High Definition TV
FC:	Factor de Compresión
IEC:	International Electrotechnical Comission
ISO:	International Organization for Standarization
ITU:	International Telecomunication Union
JPEG:	Joint Photographic Experts Group
JPEG XR:	JPEG Extended Range
LOT:	Lapped Orthogonal Transform
LBT:	Lapped Biorthogonal Transform
NTSC:	National Televisión Systems Committee
MAC:	Multiplicacion Acumulador
Mbps:	Mega bits por segundo
MIPS:	Millones de Instrucciones por Segundo
MJPEG:	Motion JPEG
MJPEG XR:	Motion JPEG XR
MJPEG 2000:	Motion JPEG 2000
MSE:	Error Cuadrático Medio

PAL: Phase Alternate Line
PCT: Photo Core Transform
POT: Photo Overlap Transform
PSNR: Relación Señal a Ruido Pico
QCIF: Quarter CIF
QP: Quantization Parameters
QSIF: Quarter SIF
QVGA: Quarter Video Graphics Array
RDSI: Red Digital de Servicios Integrados
RLE: Run Level Encoding
SDTV: Standard Definition TV
SECAM: Sequentiel Couleur Avec Memoire
SIF: Source Input Format
SOI: Start Of Image
SQCIF: Sub QCIF
T_H**:** Hadamard Transform
T_R**:** Forward Rotation
T_{oddodd}**:** Odd odd Transform
T_S**:** Forward Scaling
VLC: Variable Lenght Coding
VLIW: Palabra de Instrucción muy Larga

Apéndice B

Tablas de codificación Huffman

Categoría SSSS	Longitud de código	Código Huffman
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110
11	9	111111110

Tabla B.1: Códigos Huffman para los coeficientes DC de luminancia

Categoría SSSS	Longitud de código	Código Huffman
0	2	00
1	2	01
2	2	10
3	3	110
4	4	1110
5	5	11110
6	6	111110
7	7	1111110
8	8	11111110
9	9	111111110
10	10	1111111110
11	11	11111111110

Tabla B.2: Códigos Huffman para los coeficientes DC de crominancia

Categoría	Longitud de código	Código Huffman	Categoría	Longitud de código	Código Huffman
0/0 (EOB)	4	1010			
0/1	2	00	4/1	6	111011
0/2	2	01	4/2	10	1111111000
0/3	3	100	4/3	16	1111111110010110
0/4	4	1011	4/4	16	1111111110010111
0/5	5	11010	4/5	16	1111111110011000
0/6	7	1111000	4/6	16	1111111110011001
0/7	8	11111000	4/7	16	1111111110011010
0/8	10	1111110110	4/8	16	1111111110011011
0/9	16	1111111110000010	4/9	16	1111111110011100
0/A	16	1111111110000011	4/A	16	1111111110011101

Tabla B.3: Tablas de codificación de los coeficientes AC de luminancia

Categoría	Longitud de código	Código Huffman	Categoría	Longitud de código	Código Huffman
1/1	4	1100	5/1	7	1111010
1/2	5	11011	5/2	11	11111110111
1/3	7	1111001	5/3	16	1111111110011110
1/4	9	111110110	5/4	16	1111111110011111
1/5	11	11111110110	5/5	16	1111111110100000
1/6	16	1111111110000100	5/6	16	1111111110100001
1/7	16	1111111110000101	5/7	16	1111111110100010
1/8	16	1111111110000110	5/8	16	1111111110100011
1/9	16	1111111110000111	5/9	16	1111111110100100
1/A	16	1111111110001000	5/A	16	1111111110100101

Tabla B.4: Tablas de codificación de los coeficientes AC de luminancia

Categoría	Longitud de código	Código Huffman	Categoría	Longitud de código	Código Huffman
2/1	5	11100	6/1	7	1111011
2/2	8	11111001	6/2	12	111111110110
2/3	10	1111110111	6/3	16	1111111110100110
2/4	12	111111110100	6/4	16	1111111110100111
2/5	16	1111111110001001	6/5	16	1111111110101000
2/6	16	1111111110001010	6/6	16	1111111110101001
2/7	16	1111111110001011	6/7	16	1111111110101010
2/8	16	1111111110001100	6/8	16	1111111110101011
2/9	16	1111111110001101	6/9	16	1111111110101100
2/A	16	1111111110001110	6/A	16	1111111110101101

Tabla B.5: Tablas de codificación de los coeficientes AC de luminancia

Categoría	Longitud de código	Código Huffman	Categoría	Longitud de código	Código Huffman
3/1	6	111010	7/1	8	11111010
3/2	9	111110111	7/2	12	111111110111
3/3	12	111111110101	7/3	16	1111111110101110
3/4	16	1111111110001111	7/4	16	1111111110101111
3/5	16	1111111110010000	7/5	16	1111111110110000
3/6	16	1111111110010001	7/6	16	1111111110110001
3/7	16	1111111110010010	7/7	16	1111111110110010
3/8	16	1111111110010011	7/8	16	1111111110110011
3/9	16	1111111110010100	7/9	16	1111111110110100
3/A	16	1111111110010101	7/A	16	1111111110110101

Tabla B.6: Tablas de codificación de los coeficientes AC de luminancia

Categoría	Longitud de código	Código Huffman	Categoría	Longitud de código	Código Huffman
8/1	9	111111000	C/1	10	1111111010
8/2	15	111111111000000	C/2	16	1111111111011001
8/3	16	1111111110110110	C/3	16	1111111111011010
8/4	16	1111111110110111	C/4	16	1111111111011011
8/5	16	1111111110111000	C/5	16	1111111111011100
8/6	16	1111111110111001	C/6	16	1111111111011101
8/7	16	1111111110111010	C/7	16	1111111111011110
8/8	16	1111111110111011	C/8	16	1111111111011111
8/9	16	1111111110111100	C/9	16	1111111111100000
8/A	16	1111111110111101	C/A	16	1111111111100001

Tabla B.7: Tablas de codificación de los coeficientes AC de luminancia

Categoría	Longitud de código	Código Huffman	Categoría	Longitud de código	Código Huffman
9/1	9	111111001	D/1	11	11111111000
9/2	16	1111111110111110	D/2	16	1111111111100010
9/3	16	1111111110111111	D/3	16	1111111111100011
9/4	16	1111111111000000	D/4	16	1111111111100100
9/5	16	1111111111000001	D/5	16	1111111111100101
9/6	16	1111111111000010	D/6	16	1111111111100110
9/7	16	1111111111000011	D/7	16	1111111111100111
9/8	16	1111111111000100	D/8	16	1111111111101000
9/9	16	1111111111000101	D/9	16	1111111111101001
9/A	16	1111111111000110	D/A	16	1111111111101010

Tabla B.8: Tablas de codificación de los coeficientes AC de luminancia

Categoría	Longitud de código	Código Huffman	Categoría	Longitud de código	Código Huffman
A/1	9	111111010	E/1	16	1111111111101011
A/2	16	1111111111000111	E/2	16	1111111111101100
A/3	16	1111111111001000	E/3	16	1111111111101101
A/4	16	1111111111001001	E/4	16	1111111111101110
A/5	16	1111111111001010	E/5	16	1111111111101101
A/6	16	1111111111001011	E/6	16	1111111111110000
A/7	16	1111111111001100	E/7	16	1111111111110001
A/8	16	1111111111001101	E/8	16	1111111111110010
A/9	16	1111111111001110	E/9	16	1111111111110011
A/A	16	1111111111001111	E/A	16	1111111111110100

Tabla B.9: Tablas de codificación de los coeficientes AC de luminancia

Categoría	Longitud de código	Código Huffman	Categoría	Longitud de código	Código Huffman
B/1	10	1111111001	F/1	16	1111111111110101
B/2	16	1111111111010000	F/2	16	1111111111110110
B/3	16	1111111111010001	F/3	16	1111111111110111
B/4	16	1111111111010010	F/4	16	1111111111111000
B/5	16	1111111111010011	F/5	16	1111111111111001
B/6	16	1111111111010100	F/6	16	1111111111111010
B/7	16	1111111111010101	F/7	16	1111111111111011
B/8	16	1111111111010110	F/8	16	1111111111111100
B/9	16	1111111111010111	F/9	16	1111111111111101
B/A	16	1111111111011000	F/A	16	1111111111111110

Tabla B.10: Tablas de codificación de los coeficientes AC de luminancia

Categoría	Longitud de código	Código Huffman	Categoría	Longitud de código	Código Huffman
0/0(EOB)	2	00			
0/1	2	01	4/1	6	111010
0/2	3	100	4/2	9	111110110
0/3	4	1010	4/3	16	1111111110010111
0/4	5	11000	4/4	16	1111111110011000
0/5	5	11001	4/5	16	1111111110011001
0/6	6	111000	4/6	16	1111111110011010
0/7	7	1111000	4/7	16	1111111110011011
0/8	9	111110100	4/8	16	1111111110011100
0/9	10	1111110110	4/9	16	1111111110011101
0/A	12	111111110100	4/A	16	1111111110011110

Tabla B.11: Tablas de codificación de los coeficientes AC de crominancia

Categoría	Longitud de código	Código Huffman	Categoría	Longitud de código	Código Huffman
1/1	4	1011	5/1	6	111011
1/2	6	111001	5/2	10	1111111001
1/3	8	11110110	5/3	16	1111111110011111
1/4	9	111110101	5/4	16	1111111110100000
1/5	11	11111110110	5/5	16	1111111110100001
1/6	12	111111110101	5/6	16	1111111110100010
1/7	16	1111111110001000	5/7	16	1111111110100011
1/8	16	1111111110001001	5/8	16	1111111110100100
1/9	16	1111111110001010	5/9	16	1111111110100101
1/A	16	1111111110001011	5/A	16	1111111110100110

Tabla B.12: Tablas de codificación de los coeficientes AC de crominancia

Categoría	Longitud de código	Código Huffman	Categoría	Longitud de código	Código Huffman
2/1	5	11010	6/1	7	1111001
2/2	8	11110111	6/2	11	11111110111
2/3	10	1111110111	6/3	16	1111111110100111
2/4	12	111111110110	6/4	16	1111111110101000
2/5	15	111111111000010	6/5	16	1111111110101001
2/6	16	1111111110001100	6/6	16	1111111110101010
2/7	16	1111111110001101	6/7	16	1111111110101011
2/8	16	1111111110001110	6/8	16	1111111110101100
2/9	16	1111111110001111	6/9	16	1111111110101101
2/A	16	1111111110010000	6/A	16	1111111110101110

Tabla B.13: Tablas de codificación de los coeficientes AC de crominancia

Categoría	Longitud de código	Código Huffman	Categoría	Longitud de código	Código Huffman
3/1	5	11011	7/1	7	1111010
3/2	8	11111000	7/2	11	11111111000
3/3	10	1111111000	7/3	16	1111111110101111
3/4	12	111111110111	7/4	16	1111111110110000
3/5	16	1111111110010001	7/5	16	1111111110110001
3/6	16	1111111110010010	7/6	16	1111111110110010
3/7	16	1111111110010011	7/7	16	1111111110110011
3/8	16	1111111110010100	7/8	16	1111111110110100
3/9	16	1111111110010101	7/9	16	1111111110110101
3/A	16	1111111110010110	7/A	16	1111111110110110

Tabla B.14: Tablas de codificación de los coeficientes AC de crominancia

Categoría	Longitud de código	Código Huffman	Categoría	Longitud de código	Código Huffman
8/1	8	11111001	C/1	9	111111010
8/2	16	1111111110110111	C/2	16	1111111111011011
8/3	16	1111111110111000	C/3	16	1111111111011100
8/4	16	1111111110111001	C/4	16	1111111111011101
8/5	16	1111111110111010	C/5	16	1111111111011110
8/6	16	1111111110111011	C/6	16	1111111111011111
8/7	16	1111111110111100	C/7	16	1111111111100000
8/8	16	1111111110111101	C/8	16	1111111111100001
8/9	16	1111111110111110	C/9	16	1111111111100010
8/A	16	1111111110111111	C/A	16	1111111111100011

Tabla B.15: Tablas de codificación de los coeficientes AC de crominancia

Categoría	Longitud de código	Código Huffman	Categoría	Longitud de código	Código Huffman
9/1	9	111110111	D/1	11	11111111001
9/2	16	1111111111000000	D/2	16	1111111111100100
9/3	16	1111111111000001	D/3	16	1111111111100101
9/4	16	1111111111000010	D/4	16	1111111111100110
9/5	16	1111111111000011	D/5	16	1111111111100111
9/6	16	1111111111000100	D/6	16	1111111111101000
9/7	16	1111111111000101	D/7	16	1111111111101001
9/8	16	1111111111000110	D/8	16	1111111111101010
9/9	16	1111111111000111	D/9	16	1111111111101011
9/A	16	1111111111001000	D/A	16	1111111111101100

Tabla B.16: Tablas de codificación de los coeficientes AC de crominancia

Categoría	Longitud de código	Código Huffman	Categoría	Longitud de código	Código Huffman
A/1	9	111111000	E/1	14	11111111100000
A/2	16	1111111111001001	E/2	16	1111111111101101
A/3	16	1111111111001010	E/3	16	1111111111101110
A/4	16	1111111111001011	E/4	16	1111111111101111
A/5	16	1111111111001100	E/5	16	1111111111110000
A/6	16	1111111111001101	E/6	16	1111111111110001
A/7	16	1111111111001110	E/7	16	1111111111110010
A/8	16	1111111111001111	E/8	16	1111111111110011
A/9	16	1111111111010000	E/9	16	1111111111110100
A/A	16	1111111111010001	E/A	16	1111111111110101

Tabla B.17: Tablas de codificación de los coeficientes AC de crominancia

Categoría	Longitud de código	Código Huffman	Categoría	Longitud de código	Código Huffman
B/1	9	111111001	F/1	15	111111111000011
B/2	16	1111111111010010	F/2	16	1111111111110110
B/3	16	1111111111010011	F/3	16	1111111111110111
B/4	16	1111111111010100	F/4	16	1111111111111000
B/5	16	1111111111010101	F/5	16	1111111111111001
B/6	16	1111111111010110	F/6	16	1111111111111010
B/7	16	1111111111010111	F/7	16	1111111111111011
B/8	16	1111111111011000	F/8	16	1111111111111100
B/9	16	1111111111011001	F/9	16	1111111111111101
B/A	16	1111111111011010	F/A	16	111111111111111

Tabla B.18: Tablas de codificación de los coeficientes AC de crominancia

Apéndice C

Código fuente en C

Codificador MJPEG XR

L B T

```
//Transformada Hadamard
void _2x2T_h(short int*a,short int*b,short int*c,short int*d,short int R_flag)
  {
     short int t1,t2;
    *a +=*d;
    *b -=*c;
    t1= ((*a-*b+ R_flag)>>1);
    t2= *c;
    *c=t1-*d;
    *d=t1-t2;
    *a-=*d;
    *b+=*c;
  }
```

```
//Transformada Odd
void _T_odd(short int*a,short int*b,short int*c,short int*d)
{
  *b -=*c;
  *a +=*d;
  *c +=(*b+1)>>1;
  *d = ((*a+1)>>1) - *d;

  *b -= (*a * 3 + 4) >> 3;
  *a += (*b * 3 + 4) >> 3;
  *d -= (*c * 3 + 4) >> 3;
  *c += (*d * 3 + 4) >> 3;

  *d += *b >> 1;
  *c -= (*a + 1) >> 1;
  *b -= *d;
  *a += *c;
```

```
}
```

```
//Transformada Odd Odd
void _T_odd_odd(short int*a,short int*b,short int*c,short int*d)
{
  short int t1,t2;
  *b = -*b;
  *c= -*c;
  *d += *a;
  *c -= *b;
  t1= *d >> 1;
  t2= *c >> 1;
  *a -= t1;
  *b += t2;
  *a += (*b * 3 + 4) >> 3;
  *b -= (*a * 3 + 3) >> 2;
  *a += (*b * 3 + 3) >> 3;
  *b -= t2;
  *a += t1;
  *c += *b;
  *d -= *a;
}
```

Cuantización

```
void quant_var(short int*coef,short int nivel)
  {
    *coef=*coef/nivel;
  }
```

Predicción adaptable

```
      //j y k deben de ajustarse al ancho y alto de la imagen
      for(j=0; j < 22528  ;j+=2816)
      {
        k=256;
        for(i=3072; i< 5632; i+=256)
        {
          H_w=Yc[j+k-256]- Yc[k+j];
          V_w=Yc[j+k-256]-Yc[j+2816+k-256];
          if(abs(H_w) < 4*abs(V_w)  )
          {
            Ya[i+j]=Yc[i+j] - Yc[j+2816+k-256];
            for(l=1; l<256;l++)
```

```
            Ya[i+j+l]=Yc[i+j+l];
            for(l=16; l < 256; l+=16)
            Ya[i+j+l]=Yc[i+j+l] - Yc[j+2816+k-256+l];
          }
          else if (abs(V_w) < 4*abs(H_w))
          {
            Ya[i+j]=Yc[i+j]-Yc[k+j];
            for(l=1; l<256;l++)
            Ya[i+j+l]=Yc[i+j+l];
            for(l=16;l< 256; l +=16)
            Ya[i+j+l]=Yc[i+j+l]-Yc[k+j+l];
          }
          else
          {
            Ya[i+j] = Yc[i+j]-Yc[j+k-256];
            for(l=1; l<256;l++)
            Ya[i+j+l]=Yc[i+j+l];
          }
          k+=256;
        }
      }
```

Exploración adaptable

```
void _FwdPermute(short int*coeff)
{
    int fwd[16] = {0, 4, 1, 5,
        8, 2, 9, 6,
        12, 3, 10, 13,
        7, 14, 11, 15};
    short int t[16];
    short int idx;
    for (idx = 0 ; idx < 16 ; idx += 1)
        t[fwd[idx]] = coeff[idx];
    for (idx = 0 ; idx < 16 ; idx += 1)
        coeff[idx] = t[idx];
}
```

Codificador entrópico

```
//*****Codificador de Huffman **********************************
void Cod_Huffman_DC(short int coef)
{
SSSS=categoria_dc(coef);
longitud=l_codes_DC_Y[SSSS];
var_conca=0x00FF & codes_DC_Y[SSSS];
```

```
concatenacion();
longitud=SSSS;
var_conca=bits_adicionales(coef);
concatenacion();
}
```

```
//Codificador de coeficientes HP
void Cod_Huffman_HP(short int coef[])
{
unsigned char k1;
 RRRR=0;
  for(k1=0; k1 < NA; k1 ++)
    {
      if(coef[k1]==0)
      {
        RRRR++;
      }
      else
      {
        SSSS=categoria_hp(coef[k1]);
        longitud=l_codes_HP_Y[RRRR][SSSS-1];
        var_conca=codes_HP_Y[RRRR][SSSS-1];
        concatenacion();
        longitud=SSSS;
        var_conca=bits_adicionales(coef[k1]);
        concatenacion();
        RRRR=0;
      }

    }

  longitud=l_code_EOB;
  var_conca=code_EOB;
  concatenacion();

}
```

```
//Rutina para concatenar codigo
void concatenacion()
{
 resul = resul -longitud;
    if(resul<=0)
       if(resul==0)
     {
      reg_temp=reg_temp|var_conca;
      datos_comprimidos_Y[contador_Y]=reg_temp;
      resul=16;
      reg_temp=0;
      contador_Y=contador_Y+1;

     }else
     {
       reg_duplicado=var_conca;
     dif=-resul;
     var_conca=var_conca >> dif;
     reg_temp=reg_temp|var_conca;
     datos_comprimidos_Y[contador_Y]=reg_temp;
     resul=16;
     reg_temp=0;
     contador_Y++;
     resul=resul-dif;
     reg_duplicado=reg_duplicado << resul;
     reg_temp=reg_temp | reg_duplicado;
     }
    else
  {
    var_conca=var_conca << resul;
    reg_temp=reg_temp|var_conca;

  }

}
```

```
//Obtencion de categorias SSSS para coeficientes DC **
unsigned char categoria_dc(short int coef_DC)
{
unsigned short int comp=0,aux=2;
unsigned char catego;
if (coef_DC ==0)
catego=0;
else
{
 if (coef_DC<0)
   {
   comp=-coef_DC;
   catego=1;
   }
 else {
```

```
    comp=coef_DC;
17  catego=1;
 }
19
    while(comp>= aux)
21   {
      catego++;
23    aux=aux << 1;
     }
25 }
return catego;
27 }
```

```
1  //Obtencion de categorias SSSS para los coeficientes HP ******
   unsigned char categoria_hp(short int coef_HP)
3  {
   unsigned short int comp=0,aux=2;
5  unsigned char catego;
   if(coef_HP< 0)
7  {
      comp=-coef_HP;
9   catego=1;
   }
11 else
   {
13 comp=coef_HP;
   catego=1;
15 }
    while(comp>= aux)
17   {
      catego++;
19    aux=aux << 1;
     }
21 return catego;
   }
```

```
//Obtencion de los bits adicionales
signed short bits_adicionales(signed short cantidad)
{
if (cantidad <0)
{
cantidad=-cantidad;
cantidad=~cantidad;
cantidad=cantidad<<(16-SSSS);
cantidad=cantidad>>(16-SSSS);
}
else
cantidad=cantidad;
return cantidad;
} //bits adicionales
```

Decodificador MJPEG XR

L B T inversa

```
void _InvT_odd(short int*a,short int*b,short int*c,short int*d)
{
  *b += *d;
  *a -= *c;
  *d -= *b >> 1;
  *c += (*a + 1) >> 1;

  *a -= ((*b)*3 + 4) >> 3;
  *b += ((*a)*3 + 4) >> 3;
  *c -= ((*d)*3 + 4) >> 3;
  *d += ((*c)*3 + 4) >> 3;

  *c -= (*b + 1) >> 1;
  *d = ((*a + 1) >> 1) - *d;
  *b += *c;
  *a -= *d;
}
```

```
void _InvT_odd_odd(short int*a,short int*b,short int*c,short int*d)
{
  short int t1, t2;
  *d += *a;
  *c -= *b;
  t1 = *d >> 1;
  t2 = *c >> 1;
  *a -= t1;
  *b += t2;
```

```
  *a -= ((*b)*3 + 3 ) >> 3;
  *b += ((*a)*3 + 3) >> 2;
  *a -= ((*b)*3 + 4) >> 3;
  *b -= t2;
  *a += t1;
  *c += *b;
  *d -= *a;
  *b = -*b;
  *c = -*c;
}
```

Cuantización Inversa

```
void invquant_var(short int*coef,short int nivel)
  {

    *coef=*coef*nivel;
  }
```

Predicción adaptable Inversa

```
      //j y k deben de ajustarse al ancho y alto de la imagen
      for(j=0; j < 22528    ;j+=2816)
      {
        k=256;
        for(i=3072; i< 5632; i+=256)
        {
          H_w=Yc[j+k-256]- Yc[k+j];
          V_w=Yc[j+k-256]-Yc[j+2816+k-256];
          if(abs(H_w) < 4*abs(V_w) )
          {
            Yc[i+j]=Ya[i+j]+Yc[j+2816+k-256];
            for(l=1; l<256;l++)
            Yc[i+j+l]=Ya[i+j+l];

            for(l=16; l < 256; l+=16)
            Yc[i+j+l]=Ya[i+j+l] + Yc[j+2816+k-256+l];
          }
          else if (abs(V_w) < 4*abs(H_w))
          {
            Yc[i+j]=Ya[i+j]+Yc[k+j];
            for(l=1; l<256;l++)
            Yc[i+j+l]=Ya[i+j+l];
            for(l=16;l< 256; l +=16)
            Yc[i+j+l]=Ya[i+j+l]+Yc[k+j+l];
          }
```

```
            else
            {
              Yc[i+j] = Ya[i+j]+Yc[j+k-256];
              for(l=1; l<256;l++)
              Yc[i+j+l]=Ya[i+j+l];
            }
            k+=256;
          }
        }
```

Secuencia de video en C y Matlab

Acondicionamiendo a partir de archivos CCS

```
//Este programa se encarga de leer un archivo .dat de 32 bits y separa los
    datos en 8 bits
#include<stdio.h>
#include<stdlib.h>
main()
{
  int i,k,l,s;
  FILE *file_in;
  FILE *file_out;
  file_in=fopen("recQP9frames.dat","rb");
  file_out=fopen("libroQP9m.dat","wb");
  for(l=0; l < 912384;l++)
    {
    fscanf(file_in ,"%X\n",&i);
    unsigned short int j[4];
      k=i >> 24;
      j[0]= k & 0x000000FF;
      k= i & 0x00FF0000;
      k= k >> 16;
      j[1]=k;
      k= i & 0x0000FF00;
      k= k >> 8;
      j[2]= k;
      k= i & 0x000000FF;
      j[3]=k;

      for(s=3;s >=0; s--)
      fprintf(file_out ,"%d\n",j[s]);
    }
                fclose(file_in);
          fclose(file_out);

}
```

codigof/leeygenera.c

Secuencia diferencia

```
%Obtiene secuencia de video de la diferencia
clc
clear
video=mmreader('libro.avi');
vidFrames=read(video);
videorec=mmreader('libroQF2.avi');
vidFramesrec=read(videorec);
aviobj=avifile('librodQF2.avi','fps',6);
for k=1:24
    mov(k).cdata=vidFrames(:,:,:,k);
    movrec(k).cdata=vidFramesrec(:,:,:,k);
    movd(k).cdata=mov(k).cdata-movrec(k).cdata;
    movd(k).colormap=[];
    aviobj = addframe(aviobj, movd(k).cdata);
end
aviobj = close(aviobj);
```

codigof/vid2error.m

Genera frames determinados

```
%Obtencion de las secuencias necesarias
clc
clear
video=mmreader('libroQF05.avi');
vidFrames=read(video);

%video=mmreader('cocarQP10.avi');
%vidFramesQP10=read(videorec);

for k=[1 15 23]
    nombre='libroQF05Frame';
    im=vidFrames(:,:,:,k);
     nombre=[nombre int2str(k) '.bmp'];
    imwrite(im,nombre,'BMP');
end
```

codigof/videosec.m

Reconstrucción de video

```
%Construir imÃ¡genes
clc
c=uint8(salida);
k=1;
aviobj = avifile('librorQP9.avi','fps',6);

for s=1:24
```

```
for i=1:144
    for j=1:176
        m(i,j)=c(k);
        k=k+1;
    end
end
m=uint8(m);
Y=m;
m=[];

for i=1:72
    for j=1:88
        m(i,j)=c(k);
        k=k+1;
    end
end

m=uint8(m);
Cr=m;
m=[];
for i=1:72
    for j=1:88
        m(i,j)=c(k);
        k=k+1;
    end
end

m=uint8(m);
Cb=m;
Crn=[];
for i=1:72
    a=interp(Cr(i,:),2);
    Crn=[a ; Crn];
end

Crnn=[];
 for i=1:176
    a=interp(Crn(:,i),2);
    Crnn=[a Crnn];
 end
Crnn=uint8(Crnn(end:-1:1,end:-1:1));
Crnn=uint8(Crnn);

Cbn=[];
for i=1:72
    a=interp(Cb(i,:),2);
    Cbn=[a ; Cbn];
end

Cbnn=[];
 for i=1:176
    a=interp(Cbn(:,i),2);
```

```
    Cbnn=[a  Cbnn];
 end
Cbnn=uint8(Cbnn(end:-1:1,end:-1:1));
Cbnn=uint8(Cbnn);
imyuv=uint8(cat(3,Y,Crnn,Cbnn));
im3=ycbcr2rgb(imyuv);

mov(s).cdata=im3;
mov(s).colormap=[];
aviobj = addframe(aviobj, mov(s).cdata);
end
aviobj = close(aviobj);

hf=figure;
set(hf,'position',[150 150 176 144]);
movie(hf,mov, 1,8,[0 0 0 0])
```

codigof/consimg.m

```
%Construir imÃ¡genes
clc
c=uint8(salida);
k=1;
aviobj = avifile('libroQF2.avi','fps',6);

for s=1:24

for i=1:144
    for j=1:176
        m(i,j)=c(k);
        k=k+1;
    end
end
m=uint8(m);
Y(:,:,s)=m;
m=[];
end
c=uint8(salida2);
k=1;
for s=1:24

for i=1:72
    for j=1:88
        m(i,j)=c(k);
        k=k+1;
    end
end
m=uint8(m);
Cr=m;
m=[];
Crn=[];
 for i=1:72
     a=interp(Cr(i,:),2);
     Crn=[a ; Crn];
```

```
 end

 Crnn=[];
  for i=1:176
      a=interp(Crn(:,i),2);
      Crnn=[a Crnn];
  end
Crnn=uint8(Crnn(end:-1:1,end:-1:1));
 Crnn=uint8(Crnn);
 Crf(:,:,s)=Crnn;
 m=[];
end
c=uint8(salida3);
k=1;
for s=1:24
 for i=1:72
      for j=1:88
          m(i,j)=c(k);
          k=k+1;
      end
 end

 m=uint8(m);
 Cb=m;
 Cbn=[];
 for i=1:72
      a=interp(Cb(i,:),2);
      Cbn=[a ; Cbn];
 end

 Cbnn=[];
  for i=1:176
      a=interp(Cbn(:,i),2);
      Cbnn=[a Cbnn];
  end
 Cbnn=uint8(Cbnn(end:-1:1,end:-1:1));
 Cbnn=uint8(Cbnn);
 Cbf(:,:,s)=Cbnn;
 m=[];
end
for s=1:24
 imyuv=uint8(cat(3,Y(:,:,s),Cbf(:,:,s),Crf(:,:,s)));
 im3=ycbcr2rgb(imyuv);
 mov(s).cdata=im3;
 mov(s).colormap=[];
 aviobj = addframe(aviobj, mov(s).cdata);
 end
 aviobj = close(aviobj);

 hf=figure;
 set(hf,'position',[150 150 176 144]);
 movie(hf,mov, 1,8,[0 0 0 0])
```

codigof/consimgs.m

Bibliografía

[1] *TMS320C6414T, TMS320C6415T, TMS320C6416T, Fixed-Point Digital Signal Processors*. Texas Instruments, 2003.

[2] T. Aach. Fourier, block and lapped transforms. In P. W. Hawkes, editor, *Advances in Imaging and Electron Physics*, volume 128, pages 1–52, San Diego, 2003. Academic Press.

[3] A. N. Akansu and R. A. Haddad. *Multiresolution Signal Decomposition*. Academic Press, San Diego USA, 2nd edition, May 24, 2001.

[4] D. Austerberry. *The technology of Video and Audio Streaming*. Focal Press, E.U., 2a. edition, 2005.

[5] A. Bovik. *Handbook of Image and Video Processing*. Academic Press, 1st edition, June 14, 2000.

[6] R. Chassaing. *Digital Signal Processing and Applications with the C6713 and C6416 DSK*. Wiley, E.U., 2005.

[7] R. L. de Queiroz. *The Transforms and Applications Handbook*. CRC Press LLC, 2nd edition, 2000.

[8] G. J. Sullivan F. Dufaux and T. Ebrahimi. The jpeg xr image coding standard. *IEEE Signal Processing Magazine*, pages 195–204, Nov 2009.

[9] S. Asirvadam H. Asmare and L. Iznita. Color space selection for color image enhancement applications. *International Conference on Signal Acquisition and Processing*, pages 208–212, 2009.

[10] N. Kehtarnavaz. *Real-Time Digital Signal Processing*. Newnes, E.U., 2005.

[11] Y. Kim and J. W. Modestino. Adaptive entropy coded subband coding of images. *Image Processing, IEEE Transactions on*, 1(1):31 –48, jan 1992.

[12] A. Klein and J. Klaue. Performance study of a video application over multi hop wireless networks with statistic-based routing. In *Networking*, Alemania, 2009.

[13] E. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 2nd edition, 1999.

[14] A. Maloof and L. Chaker. Enhancing the intra-prediction in jpeg-xr by using edge information. In *2009 Fifth International Conference on Signal Image Technology and Internet Based Systems*, France, Chasseneuil, 2009.

[15] G. Mohammed. *Standards Codecs: Image Compression to Advanced Video Coding*. Institution of Electrical Engineers, Reino Unido, 2003.

[16] J. Luque Ordónez. *Videconferencia: Tecnología, sistemas y aplicaciones*. Alfaomega, México, 1st edition, 2009.

[17] F. Frescura L. Angelini P. Micanti, G. Baruffa and M. Caon. Backward-compatible robust error protection of jpeg xr compressed ideo. In *33rd IEEE Sarnoff Symposium 2010*, pages 1–5, Princeton, NJ, USA, April 2010.

[18] W. B. Pennebaker and J. L. Mitchell. *JPEG Still image Data Compression Estandard*. Chapman and Hall, New York, 1993.

[19] R. E. Woods R. C. Gonzalez and S. L. Eddins. *Digital Signal Processing Using MATLAB*. Prentice Hall, 2nd edition, 2004.

[20] I. E. G. Richardson. *Video Codec Design*. Wiley, E.U., 2002.

[21] I. E. G. Richardson. *H.264 and MPEG-4 Video Compression*. Wiley, Inglaterra, 2003.

[22] T. Richter. Spatial constant quantization in jpeg xr is nearly optimal. In *DCC '10 Proceedings of the 2010 Data Compression Conference*, Washington, DC, USA, 2010.

[23] M. J. Vitela Rodriguez. *Compresión MJPEG de video digital en un DSP*. Tesis de Maestría, UNAM, 2007.

[24] P. Jiménez Rosas. *Implementación de un Compresor-Descompresor de Imágenes con JPEG2000 en un DSP*. Tesis de Licenciatura, UNAM, 2009.

[25] Y. Q. Shi and H. Sun. *Image and Video Compresssion for Multimedia Engineering*. CRC, EU, 2000.

[26] E. A. Trueba. *Implementación de un Sistema de Adquisición y Compresión de Video MJPEG en un DSP*. Tesis de Maestría, UNAM, 2009.

[27] D. Veiner. Ee368b final report performance of the lapped orthogonal transform. `http://www.stanford.edu/class/ee368b/Projects/dveiner/node4.html`, ultima consulta el 19 de mayo de 2012.

[28] L. YU. *Evaluating and Implementing JPEG XR Optimized for Video Surveillance*. Masther Thesis, Linkoping Institute of Technology, 2010.