Universidad Carlos III de Madrid

Ingeniería de telecomunicación

PROYECTO FIN DE CARRERA

DISEÑO DE UN ALGORITMO RÁPIDO PARA DECISIÓN DE MODO EN H.264

Autor: Miguel Ángel Pellón López
Tutor: Eduardo Martínez Enríquez
Cotutor: Fernando Díaz de María

Mayo, 2009

DEPARTAMENTO DE TEORÍA DE LA SEÑAL Y COMUNICACIONES

AGRADECIMIENTOS

Resumen

El último estándar de codificación de vídeo H.264 supone una mejora respecto a sus predecesores en los que respecta a eficiencia de codificación, pero lo hace a costa de un aumento del coste computacional. Esto supone una dificultad para su utilización en aplicaciones de codificación en tiempo real.

El presente Proyecto Fin de Carrera pretende lograr una reducción en el tiempo de codificación de una secuencia de vídeo, intentando que este proceso no suponga un deterioro significativo en la calidad, ni un aumento en la tasa necesaria para su codificación. Para ello se centrará en el problema de la decisión de modo.

Abstract

The last standard in video coding, the H.264, represents a significant improvement in comparison with its predecessors in terms of coding efficiency but at the expense of a higher computational cost, which means a drawback in its use in real time coding applications.

The objective of this Master Thesis is to obtain a reduction in the time needed to code video, without a noticeable loss in quality, and avoiding an increase in the bitrate needed. In order to do so, it will tackle the problem of mode decision.

Índice de contenidos

1. Introducción y objetivos 1
1.1 Introducción 1
1.2 Objetivos 1
1.3 Estructura del documento 1
2. Codificación de vídeo 3
2.1 Introducción 3
2.2 Captura de secuencias de vídeo 3
2.3 Espacios de color 4
2.4 Resolución 6
2.5 Calidad de las secuencias de vídeo 8
2.6 Codificación DPCM 8
2.7 La transformación 9
2.8 Codificación entrópica 10
2.9 El modelo temporal 10
2.9.1 Estimación y compensación de movimiento 11
2.9.2 Flujo de datos en el codificador 12
2.9.3 Flujo de datos en el decodificador 13
2.10 Modelo espacial 14
2.11 Resumen y consideraciones 15
3. El estándar H.264 17
3.1 Introducción 17
3.2 Estructura del estándar 18
3.2.1 Perfiles y niveles 18
3.2.2 Formato de datos 19
3.2.3 Imágenes de referencia 19
3.2.4 Tiras 19
3.2.5 Macrobloques 20
3.2.6 Orden de codificación y visualización 21
3.3 Predicción Intra 22
3.4 Predicción Inter 24
3.4.1 Compensación de movimiento basada en una estructura en árbol 24
3.4.2 Selección de referencia 25
3.4.3 Vector de movimiento 27
3.4.4 *Skip* 29
3.4.5 Modo *Directo* 29
3.5 Transformación 30
3.5.1Transformada DCT 30
3.5.2 Transformada Hadamard 31
3.6 Cuantificación 32
3.7 Reescalado 33
3.8 Codificación entrópica 34
3.8.1 Codificación adaptativa según el contexto de longitud variable (CAVLC) 34
3.8.2 Codificación Aritmética binaria adaptable al contexto (CABAC) 36
4. La decisión de modo en el estándar H.264 37
4.1 El problema de la tasa-distorsión 37
4.2 Estado del arte 41
4.2.1 Algoritmos estudiados 41
4.2.2 Resumen 44

4.3 Conclusión del estudio del estado del arte 45
5. Solución propuesta 46
5.1 Introducción 46
5.2 Estructura general del diseño 46
5.2.1 Clasificador basado en perceptrones monocapa 46
5.3 Selección de variables 47
5.3.1 Descripción de las variables de prueba 47
5.3.2 Normalización de las variables de entrada 50
5.4. Entrenamiento 50
5.4.1. Conjunto de entrenamiento 51
5.4.3 Función de coste 51
5.4.3.1 Normalización de la función de coste: 53
5.5 Caracterización de los algoritmos 53
5.6 Entorno de trabajo y realización de las pruebas 55
5.7 Análisis previo de probabilidades a priori 56
6. Experimentos y resultados 62
6.1 El software de referencia 62
6.2 Diseño general 64
6.3 Diseño del decisor 1 66
6.3.1 Entrenamiento 66
6.3.2 Validación de parámetros 68
6.4 Diseño del decisor 2 70
6.4.1 Entrenamiento 71
6.4.2 Validación de parámetros 73
6.5 Test 75
7. Conclusiones y líneas futuras 83
7.1 Conclusiones 83
7.2 Líneas futuras 84
Anexo A Fichero de configuración utilizado en el software de referencia JM10.2 85
Anexo B Script utilizado en las pruebas 94
Anexo C Descripción de los vídeos utilizados 97
Anexo D Consideración sobre resultados 109
Anexo E Resultados para todas las secuencias analizadas 111
Referencias 126

Índice de figuras

Figura 2.1. Secuencia de vídeo entrelazado 4
Figura 2.2 YCbCr 4:4:4 5
Figura 2.3 YCbCr 4:2:2 5
Figura 2.4 YCbCr 4:2:0 6
Figura 2.5 Cuadro muestreado a distintas resoluciones 6
Figura 2.6. Codificador DPCM 9
Figura 2.7 Decodificador DPCM 9
Figura 2.8 Predicción y residuo 11
Figura 2.9 Esquema codificador DPCM/DCT 12
Figura 2.10 Esquema decodificador DPCM/DCT 13
Figura 2.11 Esquema codificador Intra 14
Figura 2.12 Esquema decodificador Intra 14
Figura 3.1 Esquema de un codificador H.264 17
Figura 3.2 Esquema de un decodificador H.264 17
Figura 3.3 Perfiles definidos en H.264 y sus funciones. 18
Figura 3.4 Sintaxis de una tira 20
Figura 3.5 Sintaxis de un macrobloque 20
Figura 3.6 Ejemplo de orden de codificación 21
Figura 3.7 Modos de predicción 4x4 luma 22
Figura 3.8 Modos de predicción Intra 23
Figura 3.9 Modos Inter 24
Figura 3.10 Distintas selecciones de referencia 26
Figura 3.11 Distintos tipos de resolución de los vectores de movimiento 27
Figura 3.12 Bloque a codificar y vecinos con el mismo tipo de partición 28
Figura 3.13 Bloque a codificar y vecinos con distinto tipo de partición 28
Figura 3.14 Ejemplo de cuantificador 32
Figura 3.15 Reordenamiento de coeficientes en zigzag 34
Figura 4.1 Distintos puntos de trabajo R-D 38
Figura 4.2 Minimización del coste para tres valores de λ 38
Figura 4.3 Esquema de funcionamiento del estándar H.264 40
Figura 4.4 Diagrama de flujo de [2] 42
Figura 5.1 Arquitectura del perceptrón monocapa con decisor 47
Figura 5.2 Perceptrón con función hiperbólica 51
Figura 5.3 Probabilidad a priori de cada modo 57
Figura 5.4 Probabilidad a priori de cada modo en tiras B 57
Figura 5.5 Probabilidad a priori de cada modo QP 28 tiras P 58
Figura 5.6 Probabilidad a priori de cada modo QP 36 tiras P 58
Figura 5.7 Probabilidad a priori de cada modo QP 40 tiras P 59
Figura 5.8 Probabilidad a priori de cada modo QP 28 tiras B 59
Figura 5.9 Probabilidad a priori de cada modo QP 32 tiras B 60
Figura 5.10 Probabilidad a priori de cada modo QP 36 tiras B 60
Figura 5.11 Probabilidad a priori de cada modo QP 40 tiras B 61
Figura 6.1 Esquema de funcionamiento del software de referencia 62
Figura 6.2 Esquema del decisor 1 en tiras P 67
Figura 6.3 Esquema del decisor 1 en tiras B 68
Figura 6.4 Esquema del decisor 2 en tiras P 72
Figura 6.5 Esquema del decisor 2 en tiras B 73
Figura 6.6 Gráfica de PSNR vs Tasa 79

Figura 6.7 Gráfica de mejora en tiempo vs PSNR 80
Figura 6.8 Gráfica de incremento de tasa vs PSNR 80
Figura 6.9 Gráfica de PSNR vs Tasa 81
Figura 6.10 Gráfica de mejora en tiempo vs PSNR 81
Figura 6.11 Gráfica de incremento de tasa vs PSNR 82
Figura C.1. Secuencia Akiyo. 97
Figura C.2. Secuencia Bridge-far. 97
Figura C.3. Secuencia Bus. 98
Figura C.4. Secuencia Carphone. 98
Figura C.5. Secuencia Claire. 99
Figura C.6. Secuencia Coastguard. 99
Figura C.7. Secuencia Container. 100
Figura C.8. Secuencia Football. 100
Figura C.9. Secuencia Foreman 101
Figura C.10. Secuencia Garden 101
Figura C.11. Secuencia Grandma. 102
Figura C.12. Secuencia Highway. 102
Figura C.13. Secuencia Miss-America. 103
Figura C.14. Secuencia Mobile. 103
Figura C.15. Secuencia Mother_daughter 104
Figura C.16. Secuencia News. 104
Figura C.17. Secuencia Paris. 105
Figura C.18. Secuencia Salesman 105
Figura C.19. Secuencia Silent. 106
Figura C.20. Secuencia Stefan. 106
Figura C.21. Secuencia Suzie. 107
Figura C.22. Secuencia Tempete 107
Figura C.23. Secuencia Waterfall 108
Figura D.1 Porcentaje de tiempo utilizado para el cálculo del coste RD en cada modo 109
Figura D.2 Porcentaje corregido de tiempo utilizado para el cálculo del coste RD en cada modo 110
Figura E.1 Gráfica de PSNR vs Tasa Bridge-far QCIF IP 116
Figura E.2 Gráfica de mejora en tiempo vs PSNR Bridge-far QCIF IP 116
Figura E.3 Gráfica de incremento de tasa vs PSNR Bridge-far QCIF IP 116
Figura E.4 Gráfica de PSNR vs Tasa Claire QCIF IP 117
Figura E.5 Gráfica de mejora en tiempo vs PSNR Claire QCIF IP 117
Figura E.6 Gráfica de incremento de tasa vs PSNR Claire QCIF IP 117
Figura E.7 Gráfica de PSNR vs Tasa Coastguard QCIF IP 118
Figura E.8 Gráfica de mejora en tiempo vs PSNR Coastguard QCIF IP 118
Figura E.9 Gráfica de incremento de tasa vs PSNR Coastguard QCIF IP 118
Figura E.10 Gráfica de PSNR vs Tasa Grandma QCIF IP 119
Figura E.11 Gráfica de mejora en tiempo vs PSNR Grandma QCIF IP 119
Figura E.12 Gráfica de incremento de tasa vs PSNR Grandma QCIF IP 119
Figura E.13 Gráfica de PSNR vs Tasa Miss-america QCIF IP 120
Figura E.14 Gráfica de mejora en tiempo vs PSNR Miss-america QCIF IP 120
Figura E.15 Gráfica de incremento de tasa vs PSNR Miss-america QCIF IP 120
Figura E.16 Gráfica de PSNR vs Tasa Bridge-far QCIF I2B 121
Figura E.17 Gráfica de mejora en tiempo vs PSNR Bridge-far QCIF I2B 121
Figura E.18 Gráfica de incremento de tasa vs PSNR Bridge-far QCIF I2B 121

Figura E.19 Gráfica de PSNR vs Tasa Claire QCIF I2B 122
Figura E.20 Gráfica de mejora en tiempo vs PSNR Claire QCIF I2B 122
Figura E.21 Gráfica de incremento de tasa vs Claire QCIF I2B 122
Figura E.22 Gráfica de PSNR vs Tasa Coastguard QCIF I2B 123
Figura E.23 Gráfica de mejora en tiempo vs PSNR Coastguard QCIF I2B..... 123
Figura E.24 Gráfica de incremento de tasa vs PSNR Coastguard QCIF I2B..... 123
Figura E.25 Gráfica de PSNR vs Tasa Grandma QCIF I2B 124
Figura E.26 Gráfica de mejora en tiempo vs PSNR Grandma QCIF I2B 124
Figura E.27 Gráfica de incremento de tasa vs PSNR Grandma QCIF I2B 124
Figura E.28 Gráfica de PSNR vs Tasa Miss-america QCIF I2B 125
Figura E.29 Gráfica de mejora en tiempo vs PSNR Miss-america QCIF I2B 125
Figura E.30 Gráfica de incremento de tasa vs PSNR Miss-america QCIF I2B 125

Índice de tablas

Tabla 2.1 Formatos de cuadro de una secuencia de vídeo 6
Tabla 3.1 Modos de predicción Intra4x4 luma 23
Tabla 3.2 Ejemplo de lista 0 25
Tabla 3.3 Ejemplo de lista 1 26
Tabla 3.4 Relación entre Q_{step} y QP 33
Tabla 5.1 Tipos de errores 52
Tabla 6.1 Resultados de validación decisor 1 IP 69
Tabla 6.2 Resultados de validación decisor 1 I2B 69
Tabla 6.3 Resultados de validación decisor 2 IP 74
Tabla 6.4 Resultados de validación decisor 2 I2B 74
Tabla 6.5 Resultados de test para IP 76
Tabla 6.6 Resultados de test para I2B 78
Tabla E.1 Resultados totales para resolución CIF y patrón IP 112
Tabla E.2 Resultados totales para resolución QCIF y patrón IP 113
Tabla E.3 Resultados totales para resolución CIF y patrón I2B 114
Tabla E.4 Resultados totales para resolución QCIF y patrón I2B 115

Índice de ecuaciones

Ecuación 2.1 PSRN (Peak Signal to Noise Ratio) 8
Ecuación 3.1 Bipredicción..... 26
Ecuación 3.2 Operación básica de cuantificación 32
Ecuación 3.3 Operación de cuantificación con factor de post-escalado..... 33
Ecuación 3.4 Operación de reescalado 33
Ecuación 3.5 Operación de reescalado con factor de post-escalado 33
Ecuación 4.1 Condicion de optimización 37
Ecuación 4.2 Función de coste con multiplicadores de Lagrange..... 37
Ecuación 4.3 Minimización de la función de coste 38
Ecuación 4.4 Función de coste para la estimación de movimiento 39
Ecuación 4.5 Función de coste para la decisión de modo 39
Ecuación 5.1 Función de salida del perceptrón 46
Ecuación 5.2 Desviación típica 49
Ecuación 5.3 Vector gradiente..... 49
Ecuación 5.4 Fase del vector gradiente 49
Ecuación 5.5 Módulo del vector gradiente 49
Ecuación 5.6 Aproximación del módulo del gradiente 49
Ecuación 5.7 Función de coste para el entrenamiento..... 51
Ecuación 5.8 Función de coste normalizada para el entrenamiento..... 53
Ecuación 6.1 Función de coste para el decisor 2 70

1. Introducción y objetivos

1.1 Introducción

El Moving Picture Experts Group (MPEG) y el Video Coding Experts Group (VCEG) han desarrollado un estándar que supera a sus predecesores MPEG-4 y H.263, consiguiendo una mejora en la compresión de las secuencias de vídeo. Lo consigue gracias a un aumento en la complejidad de codificación, adoptando un número importante de adelantos técnicos que permiten un aumento en la eficiencia de codificación.

El presente documento tratará sobre la codificación de vídeo con el estándar H.264, centrándose en la etapa de decisión de modo.

1.2 Objetivos

El objetivo que se intenta alcanzar es una reducción significativa en la carga computacional necesaria para la codificación. Para ello, se intentará realizar una decisión prematura de modo, evitando así la realización del proceso completo. Es evidente que esta decisión prematura conllevará una perdida en la calidad del vídeo, así como un aumento en la tasa necesaria para la codificación, ya que el modo elegido no será el óptimo en todos los casos.

Nuestro objetivo será reducir el tiempo de codificación intentando que la calidad de los resultados se vea afectada lo mínimo posible. Este tipo de avances será especialmente útil para aplicaciones de codificación en tiempo real.

1.3 Estructura del documento

El capítulo 2 introducirá los principales conceptos de codificación de vídeo, así como los bloques presentes en la mayoría de codificadores y los modelos de codificación más utilizados.

En el capítulo 3 se realizará una descripción del estándar H.264, extendiendo el análisis de los bloques introducidos en el capítulo anterior, y haciendo hincapié en aquellos aspectos que serán especialmente relevantes para este proyecto.

Una vez presentado el estándar, en el capítulo 4 definiremos en qué consiste la decisión de modo y se planteará el problema de tasa-distorsión. Además se expondrá el estado del arte, detallando algunos de los algoritmos presentes en la literatura de decisión de modo.

El capítulo 5 se encargará de describir el algoritmo, las condiciones en las que se realizarán las pruebas, y cómo analizaremos los resultados.

Finalmente, en el capítulo 6 se explicará en primer lugar el funcionamiento del software de referencia con el que se trabajará. A continuación se presentará el trabajo realizado, exponiendo los resultados obtenidos, y comparándolos con un algoritmo aceptado por el VCEG, e incluido en el software de referencia utilizado.

Por último, en el capítulo 7 se presentarán las conclusiones obtenidas tras la realización del proyecto y las posibles líneas futuras de investigación a seguir.

2. Codificación de vídeo

2.1 Introducción

Codificar un vídeo consiste en compactar una secuencia de imágenes digitales en un menor número de bits. Actualmente se puede encontrar todo tipo de dispositivos que manejan vídeos, como móviles, televisores, reproductores… Un vídeo sin codificar, a calidad media, ocupa una gran cantidad de memoria, por lo que es difícil de reproducir, almacenar o enviar. Es por ello que el flujo de vídeo se codifica (comprime) con el fin de que este sea manejable por cualquier aplicación.

El proceso de compresión comprende un par de sistemas complementarios, un codificador (*encoder*) que será el encargado de convertir los datos de forma que ocupen menos bits, y un decodificador (*decoder*) encargado de transformar los datos comprimidos en el vídeo original. El par *encoder-decoder*, es conocido como CODEC.

La reducción de tamaño es en parte posible gracias a las redundancias existentes entre los datos que componen un vídeo. Eliminando estas redundancias estadísticas, se consigue reducir el número de bits que ocupa el vídeo, sin que se pierda calidad. Por desgracia, utilizando solamente redundancia estadística no es posible alcanzar los niveles de compresión que se requieren en la mayoría de aplicaciones. Es por ello que normalmente se trabaja con codificación con pérdidas, es decir, el vídeo codificado no será exactamente igual al original, ya que parte de la información se perderá en el proceso de compresión.

2.2 Captura de secuencias de vídeo

Una escena de vídeo es espacial y temporalmente continua, por lo que representarla digitalmente implica realizar un doble proceso de muestreo: espacial (con una rejilla rectangular sobre cada plano del vídeo) y temporal (como una serie de planos tomados a intervalos de tiempo regulares). Una vez hecho esto, cada muestra espacio-temporal (píxel) será representada mediante un número o conjunto de números que describirá su brillo y color.

Para obtener una imagen muestreada en 2D, una cámara enfoca una proyección de la escena de vídeo en un sensor, como por ejemplo un *array* de CCD (*Charge Coupled Devices*). En el caso de que la captura se haga en color, cada componente de color es separada mediante filtros, y proyectada en un *array* independiente. Hay que destacar que la calidad visual de la imagen es muy dependiente del número de puntos que contenga dicho *array*.

Para el muestreo temporal, se toma una sucesión de capturas a intervalos regulares de tiempo. Al reproducir esta secuencia de imágenes fijas se logra una ilusión de movimiento, que será más fluida cuanto mayor sea la frecuencia de muestreo. El problema que esto conlleva es la necesidad de una mayor cantidad de memoria (en el caso de almacenamiento) o de un mayor ancho de banda (en el caso de transmisión). Es por ello que la frecuencia de muestreo se fijará en función de la aplicación. Para aplicaciones de videoconferencia, en las que no importa que el movimiento se vea poco

fluido, la frecuencia de muestreo puede ser inferior a las diez imágenes por segundo. Con una frecuencia de entre 10 y 20 imágenes por segundo, se consigue un movimiento más fluido, pero que puede sufrir de saltos en el caso de movimientos rápidos. En el caso de imágenes de televisión, el muestreo suele ser de entre 25 y 30 imágenes por segundo.

También existe la posibilidad de mejorar la calidad visual suavizando el movimiento entre imágenes sucesivas mediante la técnica del entrelazado de vídeo. La idea es dividir cada imagen o cuadro en dos campos distintos: campo superior, que contiene las líneas pares; y campo inferior, que contiene las impares. Cada uno de ellos cuenta con la mitad de muestras de la imagen original, por lo que el número de muestras totales permanece constante.

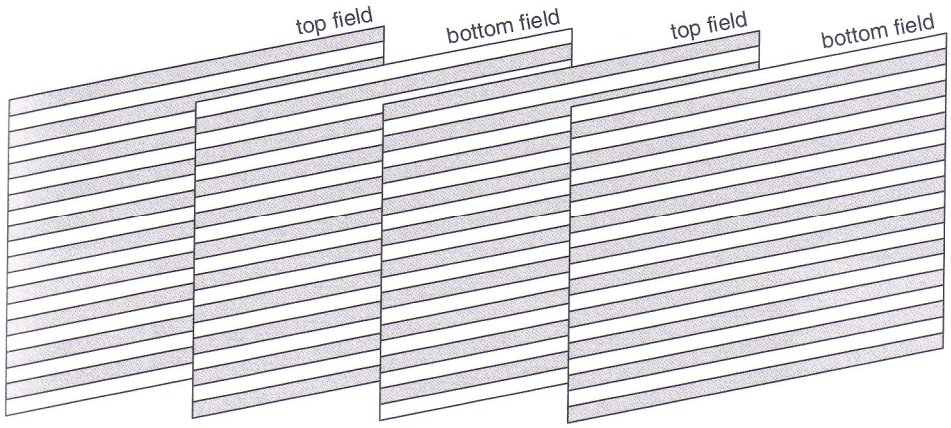


Figura 2.1. Secuencia de vídeo entrelazado

La mejora se encuentra en que al representar la alternancia entre campos, de una imagen a la siguiente sólo cambian la mitad de las líneas, percibiéndose una mayor fluidez. Una secuencia de 50 campos por segundo tiene mejor apariencia que una de 25 imágenes por segundo, aunque ambas precisen del mismo número de bits.

2.3 Espacios de color

La mayoría de aplicaciones de vídeo actuales trabajan con secuencias en color, y por tanto necesitan un método para capturar y representar la información del color. En el caso de una secuencia de vídeo monocromo, basta con un simple número para indicar la luminancia o brillo de cada píxel. Las imágenes en color, en cambio, requieren de al menos tres. El método elegido para representar el brillo y color es llamado espacio de color.

Uno de los más utilizados es el RGB (Red Green Blue), que mediante tres números indica las proporciones de rojo, verde y azul, los tres colores primarios. Este método es utilizado por las pantallas de tipo CRT y LCD para representar el color. Sin embargo, el ojo humano es menos sensible al color, que a la luminancia (brillo) por lo que separando la información sobre el color de la del brillo, y aplicando una mayor resolución para la luminancia, se obtendrá un mejor resultado visual. Es por ello que varios estándares de codificación utilizan el método YCbCr. En este espacio, Y es la

componente de luminancia, una versión monocroma de la imagen en color, y se calcula como una media ponderada de R,G y B:

$$Y = k_r \mathrm{R} + k_g \mathrm{G} + k_b \mathrm{B}$$

Donde k son los factores de ponderación. La información sobre el color se representa como "diferencia de color" o componentes de crominancia, que se definen como la diferencia entre R, G o B y la luminancia Y:

$$Cr=R-Y$$
$$Cb=B-Y$$
$$Cg=G-Y$$

A primera vista parece que utiliza cuatro valores en lugar de los tres utilizados por RGB, pero el valor de Cr+Cb+Cg es una constante, por lo que sólo serán necesarios dos (Cb y Cr son los utilizados) de los tres componentes de crominancia.

En este espacio de color son comunes los siguientes formatos:

- YCbCr 4:4:4. La luminancia y las crominancias tienen la misma resolución.

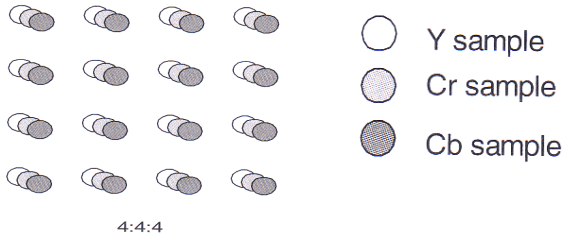


Figura 2.2 YCbCr 4:4:4

- YCbCr 4:2:2. Las crominancias tienen la mitad de resolución horizontal que la luminancia.

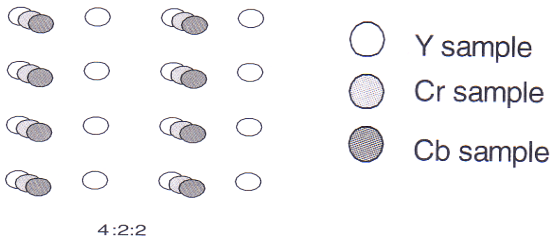


Figura 2.3 YCbCr 4:2:2

- YCbCr 4:2:0. Las crominancias tienen la mitad de resolución que la luminancia tanto horizontal como verticalmente. Este formato es muy utilizado en aplicaciones de usuario como videoconferencia, televisión digital y almacenamiento en DVD.

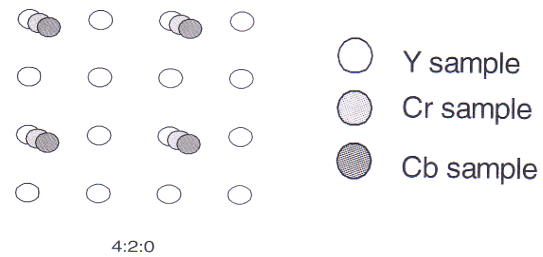


Figura 2.4 YCbCr 4:2:0

2.4 Resolución

Denominamos resolución de un vídeo al par formado por el número de píxeles utilizados por línea horizontal y vertical, tanto para la luminancia como para la crominancia. A continuación se muestra una tabla con algunos de los distintos formatos de resolución que puede seguir un cuadro de una secuencia de vídeo.

Formato	Resolución de la luminancia (horizontal x vertical)	Bits necesarios por cuadro (YCbCr 4:2:0, 8bits por muestra)
Sub-QCIF	128 x 96	147456
Quarter CIF (QCIF)	176 x 144	304128
CIF	352 x 288	1216512
4CIF	704 x 576	4866048

Tabla 2.1 Formatos de cuadro de una secuencia de vídeo

La figura 5 muestra una imagen con los distintos formatos expuestos.

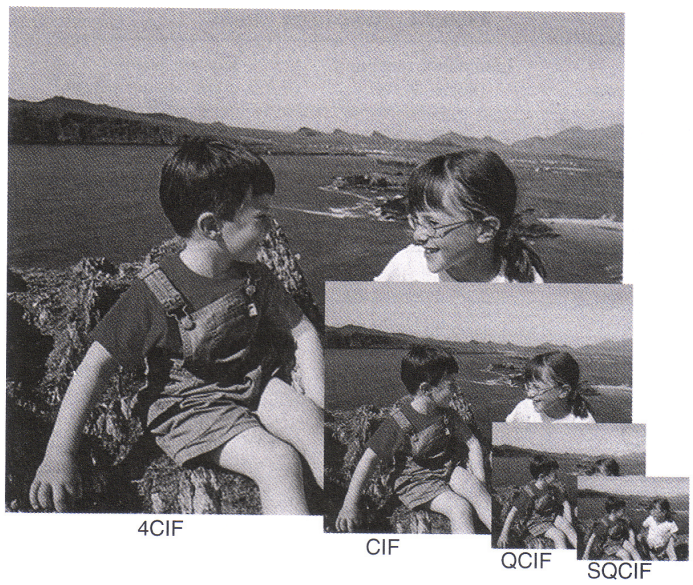


Figura 2.5 Cuadro muestreado a distintas resoluciones

La elección de dichos tamaños dependerá generalmente de la aplicación y de la capacidad de transmisión o almacenamiento disponible. Por ejemplo, 4CIF es apropiado para televisión o almacenamiento en DVD, CIF y QCIF para aplicaciones de videoconferencia y QCIF y SQCIF para aplicaciones móviles multimedia, donde tanto la resolución de la pantalla como el régimen binario están limitados.

Tomando como ejemplo una secuencia 4CIF, utilizada en televisión y DVD, y considerando una frecuencia de 25 imágenes por segundo, se necesitaría transmitir cerca de 15MB/s. Este dato deja clara la necesidad de la compresión de las secuencias de vídeo.

2.5 Calidad de las secuencias de vídeo

Con el fin de caracterizar, evaluar y comparar los distintos sistemas de codificación de vídeo, se hace necesario definir una medida de la calidad de una secuencia de vídeo.

El problema aparece en que medir la calidad visual de una secuencia vídeo no es una tarea sencilla. Esto se debe a que la calidad del vídeo es, ante todo, una variable subjetiva, y, a pesar de que están estandarizados algunos procedimientos para la evaluación de la calidad subjetiva, una prueba de este tipo presenta una gran cantidad de problemas prácticos.

Medir la calidad de un sistema de vídeo de un modo objetivo proporciona resultados precisos y repetibles, pero hasta el momento, ninguna medida de este tipo ha sido capaz de tener en cuenta los aspectos preceptúales del sistema visual humano. A pesar de todo, como se trata de medidas relativamente fáciles de realizar y que nos pueden dar una cierta información sobre la calidad de la secuencia, se utilizan.

Un ejemplo de medida objetiva de la calidad es la PSNR (Peak Signal to Noise Ratio). La PSNR se mide en escala logarítmica y está relacionada con el error cuadrático medio (MSE) entre la señal original y la imagen afectada por el sistema (codificada y decodificada).

$$PSNR_{dB} = 10 \cdot \log_{10} \frac{\left(2^N - 1\right)^2}{MSE}$$

Ecuación 2.1 PSRN (Peak Signal to Noise Ratio)

Donde N es el número de bits por cada muestra de la imagen y el numerador representa el cuadrado del máximo valor que puede tomar un píxel de la imagen. Esta fórmula es fácil de calcular y se ha extendido como medida de la calidad en todos los trabajos relacionados con codificación de vídeo.

A pesar de los inconvenientes mencionados, se utilizará la PSNR como medida de calidad a lo largo de todo el proyecto, ya que es la medida más fiable de la que se dispone.

2.6 Codificación DPCM

Para la eliminación de la redundancia, la mayoría de los compresores de vídeo utilizan sistemas basados en DPCM (Differential Pulse Code Modulation). Este método de codificación consiste en que cada señal de entrada (en nuestro caso una señal de vídeo), se codifica como la diferencia entre esa señal y una predicción de la misma. Esta predicción puede obtenerse de diversas formas, aprovechando la correlación temporal o espacial existente en los datos de vídeo. Los estándares H.261, H.263, H.264 y los MPEG 1,2 y 4-Visual, siguen este modelo, pero con implementaciones muy diversas.

A continuación se muestra el esquema general en el que se basa la codificación DPCM:

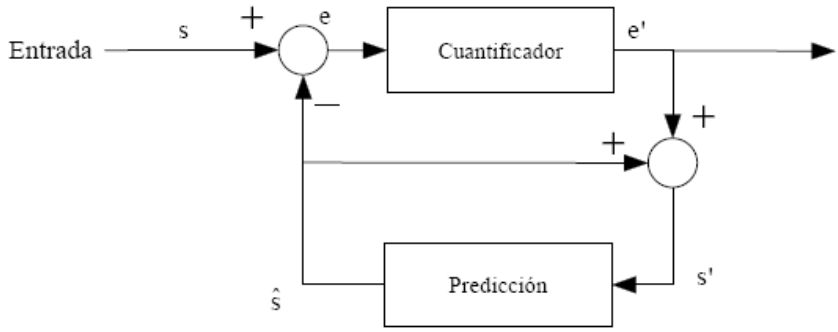


Figura 2.6. Codificador DPCM

A la entrada se tienen los datos a codificar (s) a los cuales se les resta la predicción (ŝ) obteniendo así el error en la predicción (e), que en adelante llamaremos residuo. A continuación se lleva a cabo el proceso de cuantificación, que consiste en el paso de un conjunto de valores de entrada a otro conjunto de valores prefijados más reducido. Este proceso será explicado en mayor detalle en apartados posteriores, ya que es muy importante en el proceso de codificación.

Una vez realizada la cuantificación, el resultado (e') se suma a la predicción ŝ, teniendo así la señal reconstruida s', que podrá ser utilizada como base para una predicción posterior.

El diagrama de decodificación es el siguiente:

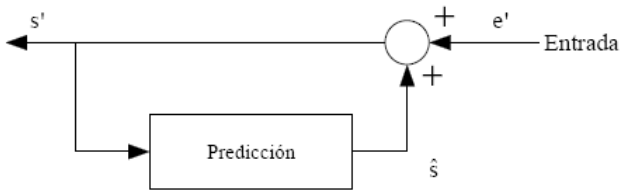


Figura 2.7 Decodificador DPCM

El residuo cuantificado generado en el codificador se toma como entrada, y es sumado a la predicción (ŝ) generada en el decodificador. Como resultado se tiene la señal descodificada s'. Se puede observar que el proceso de generar una señal reconstruida se lleva a cabo tanto en el codificador como en el decodificador. Esto se debe a que la predicción generada en el codificador y el decodificador deberá ser siempre igual, y ésta se obtiene a partir de la señal reconstruida, y no la señal original, que sólo se encuentra disponible en el codificador.

2.7 La transformación

Otro de los procesos que permiten una compresión alta en los CODEC de vídeo es la utilización de transformadas. Existe una gran variedad de transformaciones matemáticas que pasan un conjunto de datos de un sistema de medición a otro. En algunos casos, la representación de los datos en el nuevo sistema, posee propiedades que facilitan la compresión de los mismos. Por ejemplo, en el dominio espacial, las muestras de una imagen se encuentran altamente correladas. El objetivo al realizar la transformación será que en el nuevo dominio la correlación sea menor, dejando un

número pequeño de coeficientes que sean importantes para la imagen, y un gran número de coeficientes que no afecten tanto a la visualización.

Es importante resaltar que el proceso de transformación no supone una compresión per se, sino que ésta se consigue eliminando los coeficientes menos importantes, teniendo por tanto una codificación con pérdidas.

La transformada más utilizada en codificación de vídeo es la DCT (*Discrete Cosine Transform*). Sus propiedades más interesantes son linealidad e invertibilidad, lo cual es imprescindible en la codificación. Además puede ser aplicada a bloques de diversos tamaños, y de una manera muy eficiente ya que existen varias fórmulas para un cálculo rápido de la transformada.

2.8 Codificación entrópica

El objetivo de la codificación es compactar los datos de entrada. Para ello, además de tener en cuenta las características propias del vídeo, como la correlación temporal o espacial en las imágenes, hemos de tener presente que tratamos con vídeo digital, es decir, los datos son binarios. Un bloque a codificar, después de haber descartado los coeficientes de la DCT que sean menos importantes, tendrá pocos valores distintos de cero y utilizando métodos de compresión estadística, se conseguirá mejorar la eficiencia en la codificación.

Para ello, en primer lugar los valores son reordenados, ya que los coeficientes distintos de cero tienden a estar concentrados en la esquina izquierda superior, en la zona de bajas frecuencias de la transformada DCT. Es por ello que un escaneo en zigzag, permite poner al principio los valores más significativos.

A continuación, a estos valores se les aplica una codificación run-level, que consiste en codificar cada coeficiente distinto de cero como un par, en el que el primer valor es el del número de ceros que preceden al coeficiente, y el segundo el número es el valor del coeficiente actual.

Por último, se hace uso de una codificación entrópica, que utilizará palabras de bits de longitud variable para representar los distintos valores de entrada, en función de la probabilidad de aparición de dicha entrada. Una entrada muy repetida, tendrá asignada una palabra corta, mientras que a una entrada que no sea frecuente, se le asignará una palabra más larga.

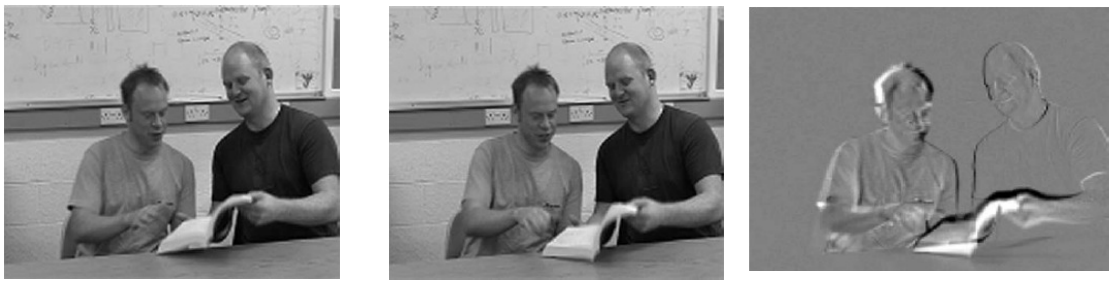
2.9 El modelo temporal

El modelo temporal es una implementación de la codificación DPCM, en el cual la predicción se genera a partir de bloques pertenecientes a imágenes previamente codificadas. Cuando se hace uso del modelo temporal hablaremos de codificación Inter.

Una escena de vídeo es en realidad una sucesión de imágenes, que expuestas a suficiente velocidad, parecen fluidas. Este efecto se consigue en torno a las 25/30 imágenes por segundo por lo que el contenido de imágenes sucesivas tiende a ser muy

parecido, a no ser que ocurra un cambio de escena. Es por ello que en vez de enviar todas las imágenes completas, se enviará la diferencia entre imágenes consecutivas. Se usará por tanto, la correlación temporal existente entre dichas imágenes para comprimir.

Para ello, se genera una predicción de la imagen a codificar, la cual es restada a la imagen original obteniendo el residuo, o diferencia de imágenes. Se puede ver que si en el decodificador se obtiene la misma predicción, bastará con codificar y enviar el residuo, ya que con sumar éste a la predicción se obtendrá en el decodificador la imagen reconstruida.



a) b) c)

Figura 2.8 Predicción y residuo

El método más simple de predicción temporal es la utilización de la imagen anterior (b en la figura 2.8) como predicción de la actual (a). El problema es que con una predicción de este tipo quedará mucha energía en le residuo (c), lo que significa que una vez realizada la compresión temporal, hay una importante cantidad de datos a codificar. La mayor parte de esta energía suele deberse a movimientos entre las dos imágenes. Es por esto por lo que se utilizan técnicas de compensación de movimiento para obtener una predicción más precisa.

2.9.1 Estimación y compensación de movimiento

El proceso de estimación de movimiento crea un modelo del plano actual basado en la información disponible de planos previamente codificados, llamados planos de referencia. El objetivo de la estimación será el de modelar el plano actual de la manera más precisa posible (ya que esto resultará en una mejor eficiencia de compresión), manteniendo una complejidad computacional aceptable. Para ello se dividirá la imagen que está siendo procesada en unidades denominadas macrobloques (16x16 píseles), los cuales a su vez pueden dividirse en bloques de diferentes tamaños. Para cada uno de estos bloques se aplicará un algoritmo de estimación de movimiento que buscará un área del plano de referencia que sea lo más parecida, es decir, que minimice la diferencia de energía entre el bloque actual y el bloque referencia. Hay que tener en cuenta que no se podrá realizar esta búsqueda en todo el plano de referencia, ya que sería demasiado costoso computacionalmente. Es por ello que suele realizarse en un entorno reducido alrededor de la posición del macrobloque a codificar, ya que es de esperar que el desplazamiento sea pequeño entre dos cuadros.

Una vez seleccionado el bloque en el plano de referencia, se le restará el bloque actual, obteniendo el bloque de diferencias, o residuo. Junto con el residuo, para cada bloque deberá enviarse el vector de movimiento, que indica la posición de la región de referencia utilizada para la compensación. En el decodificador, se sumará el residuo al

bloque de referencia apuntado por el vector de movimiento, obteniendo una reconstrucción del bloque original.

2.9.2 Flujo de datos en el codificador

A continuación se muestra el esquema de un codificador con sus elementos más comunes.

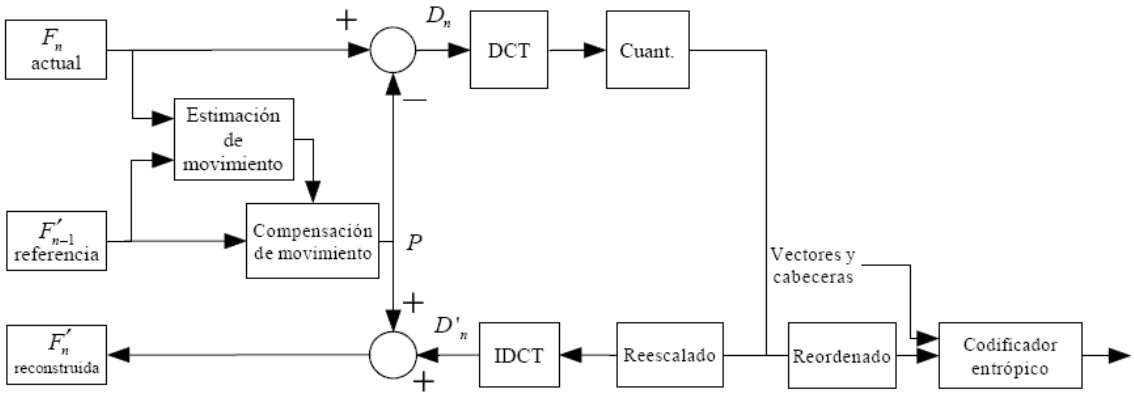


Figura 2.9 Esquema codificador DPCM/DCT

El flujo de codificación es el siguiente:

1.- A la entrada se tiene un cuadro F_n, el cual es procesado en macrobloques, que contienen las muestras correspondientes de luminancia y crominancia. Estos a su vez pueden ser procesados en subunidades de hasta 4x4 píxeles, pero en este apartado, con el fin de tratar el problema de forma genérica, hablaremos de macrobloque como única unidad de procesado.

2.- El cuadro de entrada es comparado con un cuadro de referencia, F'_{n-1} que ha sido codificado con anterioridad, y mediante una función de estimación de movimiento, se busca una región 16x16 en F'_{n-1}, que sea lo más parecida posible al macrobloque de F'_n que está siendo procesado. La diferencia de posiciones entre ambos macrobloques se denomina vector de movimiento (MV).

3.- En base a este vector de movimiento, y utilizando la referencia F_{n-1}, se realiza la compensación de movimiento, generando una predicción P (esta predicción es la región 16x16 que había sido elegida por la función de estimación de movimiento).

4.- La predicción P se resta del macrobloque que está siendo procesado, produciendo un residuo, o macrobloque de diferencias D_n.

5.- Este macrobloque de diferencias es transformado utilizando la DCT. Normalmente este proceso se realiza en subbloques de 8x8 o 4x4, y utilizando aproximaciones que faciliten el cálculo de la transformada.

6.- Los coeficientes del residuo son cuantificados y reordenados.

7.- Debido a la estimación de movimiento, es de esperar que los coeficientes de la transformada DCT sean en su mayoría cero, por lo que son codificados de manera que se mejore la compresión.

8.- Finalmente los coeficientes, los vectores de movimiento, y demás información, es codificada entrópicamente, produciendo así la cadena de bits codificada.

El flujo de reconstrucción es el siguiente:

1.- Cada macrobloque cuantificado es reescalado, y se le realiza la transformada inversa, produciendo un residuo D' decodificado. Hay que tener en cuenta, que al ser la cuantificación un proceso no reversible, D' será diferente a D.

2.- A cada macrobloque del cuadro D' se le aplica la compensación de movimiento, es decir, se le añade P, produciendo un cuadro reconstruido, F'_n, que podrá ser utilizado como referencia para codificaciones posteriores.

2.9.3 Flujo de datos en el decodificador

A continuación se muestra el esquema de un decodificador con sus elementos más comunes.

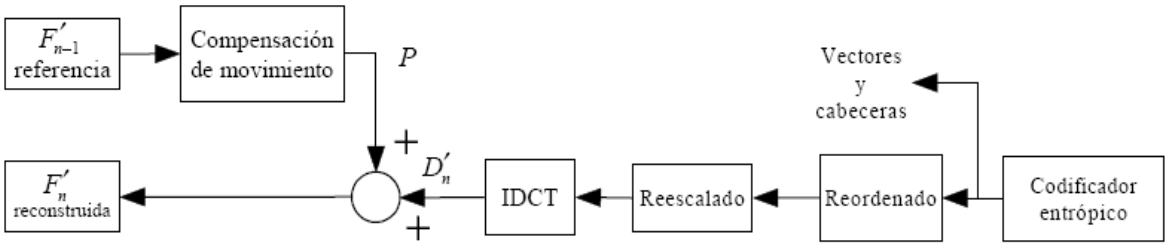


Figura 2.10 Esquema decodificador DPCM/DCT

1.- Al decodificador entra una cadena comprimida de bits, que es decodificada entrópicamente para extraer los coeficientes, vectores de movimiento, y cabeceras para cada macrobloque.

2.- A continuación se realiza el proceso inverso de la codificación y reordenamiento, obteniendo así el macrobloque cuantificado y transformado.

3.- Reescalando, y realizando la transformada inversa, se obtiene el residuo D', que como en el caso del codificador, será diferente del residuo D, debido al proceso no reversible de la cuantificación.

3.- El vector de movimiento decodificado se usa para localizar la región 16x16 (P) en la referencia F'_{n-1} del decodificador que corresponde a la compensación de movimiento.

4.- A cada macrobloque P, se le suma el macrobloque residuo D' correspondiente, produciendo finalmente el cuadro reconstruido F'_n.

2.10 Modelo espacial

Dentro de una imagen, los píxeles adyacentes también se ven sujetos a una gran correlación, siendo normalmente los valores de muestras vecinas muy parecidos, por lo que se puede utilizar esta correlación espacial con el fin de generar una predicción para un sistema DPCM. Cuando se hace esto, se hablará de codificación Intra.

En el caso de cambios de escena o con la aparición de nuevos objetos en la imagen, la codificación Inter da como resultado una predicción bastante diferente de la imagen original, por lo que el residuo a enviar puede ser demasiado pesado. Además, cuando se codifica la primera imagen de una secuencia, no se dispone de imágenes de referencia, por lo que la única opción será o enviar la imagen entera, cosa poco deseable por el gran tamaño en bits, o utilizar la correlación espacial para codificar.

A continuación se muestra el esquema de este tipo de codificación:

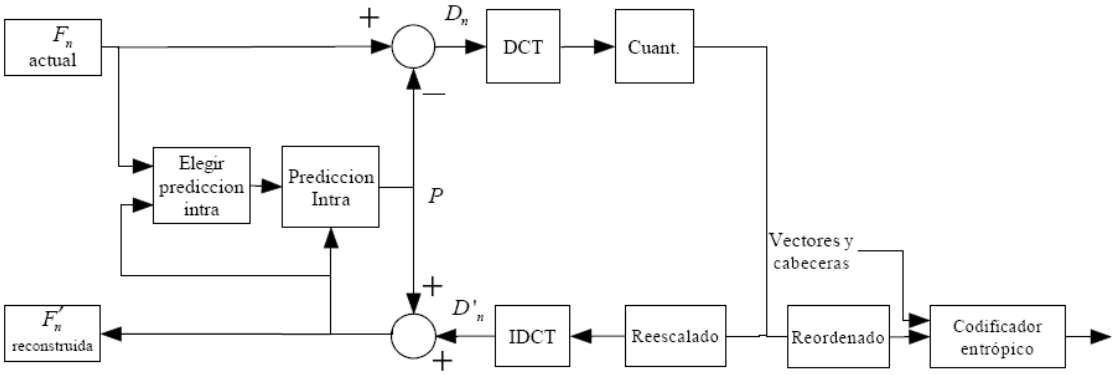


Figura 2.11 Esquema codificador Intra

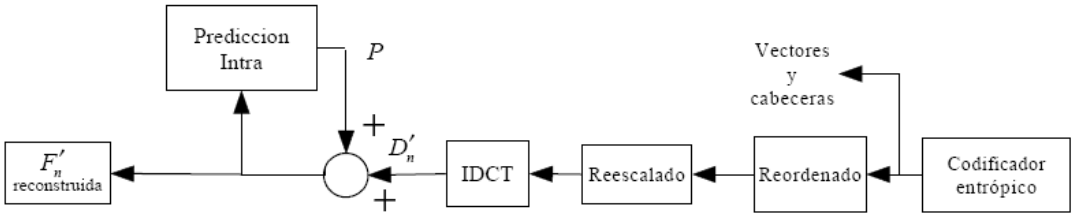


Figura 2.12 Esquema decodificador Intra

Se puede observar que el esquema de funcionamiento es parecido al de codificación Inter, sustituyendo la parte en la que se genera la predicción mediante estimación y compensación de movimiento, por otra en la que se utiliza una predicción Intra. A continuación explicamos esta fase, ya que el resto funcionan de manera análoga a la descrita en el apartado de modelo temporal.

Para generar la predicción del macrobloque actual, el codificador hace uso de los bloques adyacentes pertenecientes al mismo plano. Existen varios modos de predicción Intra, según qué bloques adyacentes se utilicen, y como se operen sus píxeles. En el

próximo capítulo se explicará en detalle los distintos modos, y cómo se realiza la elección.

Una vez calculada la predicción, la forma de operar del codificador es análoga a la de la codificación Inter, y el proceso del decodificador también, salvo la obtención de la predicción, que en este caso será Intra. Sabiendo el tipo de predicción Intra que se ha llevado a cabo en el codificador, se puede obtener en el decodificador la misma predicción P que se tenía en el codificador haciendo uso de los bloques vecinos. A esta predicción se le suma el residuo decodificado, obteniendo así el bloque reconstruido

2.11 Resumen y consideraciones

En este apartado se ha explicado el proceso de captura de un vídeo digital y los distintos formatos de vídeo. Además se ha visto de manera general como opera un par codificador/ decodificador utilizando el sistema DPCM, tanto en su variante temporal, codificación Inter, como en su variante espacial, codificación Intra. También se han explicado los principales pasos en el proceso de codificación, como la transformación, cuantificación y reescalado.

A continuación, se muestra el proceso llevado a cabo en el codificador cuando va a operar sobre el plano F_n

Figura 2.13: Plano a codificar F_n

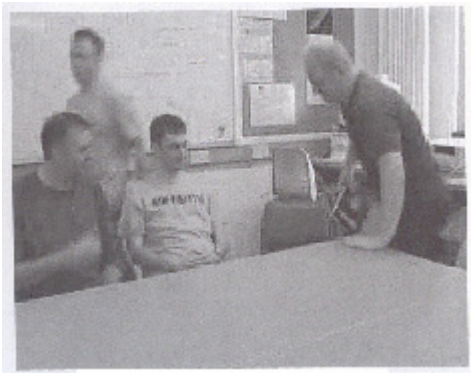


Figura 2.14: Plano reconstruido referencia F'_{n-1}

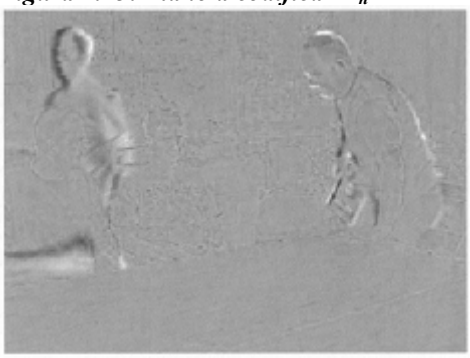


Figura 2.15: Residuo sin compensación de movimiento ($F_n - F'_{n-1}$)

Figura 2.16: Plano con vectores de movimiento

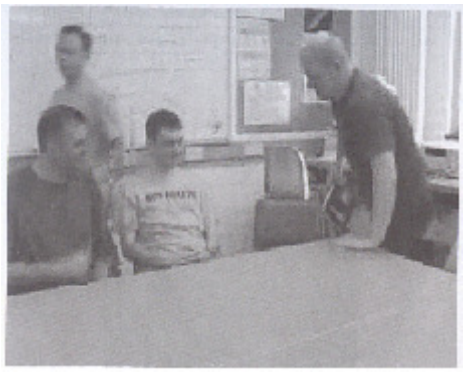


Figura 2.17: Plano referencia con compensación de movimiento o predicción P

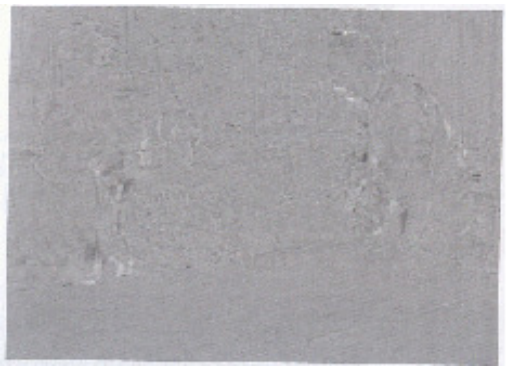


Figura 2.18: Residuo con compensación de movimiento D_n

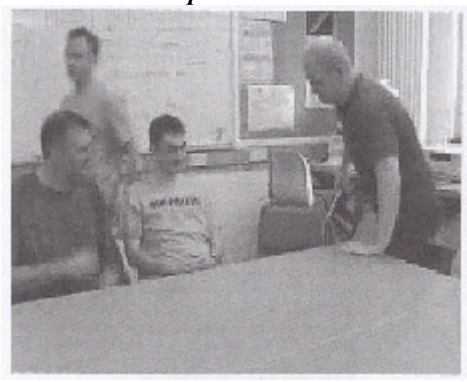


Figura 2.19: Plano reconstruido F'_n

En las dos primeras figuras se muestra el plano a codificar, y el que utilizaremos de referencia. Si no se utiliza compensación de movimiento para generar la predicción, el residuo resultante es el de la figura 2.15.

En la figura 2.16 se muestran los vectores de movimiento obtenidos en el proceso de estimación de movimiento para cada macrobloque, y en la siguiente figura la predicción obtenida. Se puede ver en la figura 2.18 que al realizar la compensación de movimiento, el residuo ha disminuido notablemente su energía respecto a la de la figura 2.15.

Por último, se muestra la figura reconstruida, que será almacenada tanto en el codificador como en el decodificador, con el fin de ser utilizada como referencia para planos codificados con posterioridad.

3. El estándar H.264

3.1 Introducción

El estándar H.264, desarrollado por el MPEG y el VCEG obtiene una mayor eficiencia de compresión que estándares anteriores, y lo consigue gracias un aumento en la complejidad de codificación, con el uso de bloques de varios tamaños, múltiples imágenes de referencia, etc. Todo esto hará que el tiempo empleado en el proceso de codificación sea mayor.

Como en estándares anteriores, no se define un CODEC (enCoder/DECoder). A cambio, se define la sintaxis que ha de tener una cadena de bits codificada, así como el método de decodificación de la cadena de bits. A pesar de esto, casi todos los codificadores y decodificadores incluyen los elementos mostrados en las figuras 3.2 y 3.4, siendo estos en su mayoría comunes con estándares anteriores. Las principales diferencias, más que en el diseño global, se darán dentro de cada bloque específico.

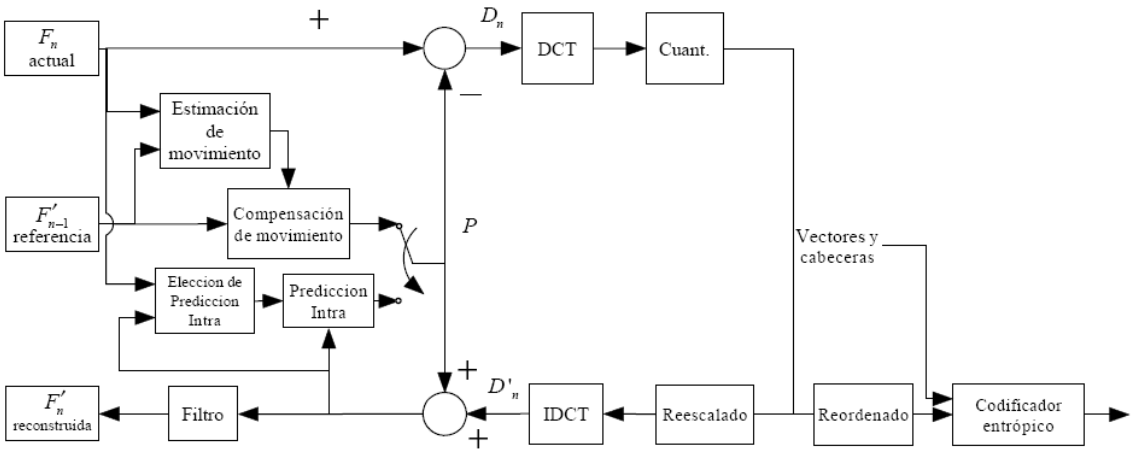


Figura 3.1 Esquema de un codificador H.264

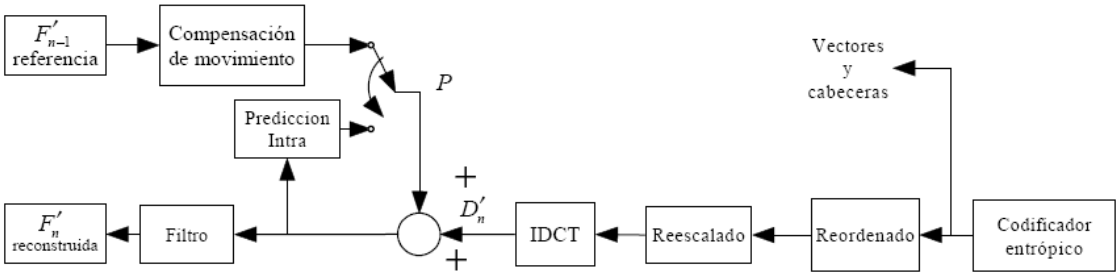


Figura 3.2 Esquema de un decodificador H.264

3.2 Estructura del estándar

3.2.1 Perfiles y niveles

El estándar define tres perfiles, cada uno de los cuales tiene un conjunto de funciones de codificación, y para los cuales se especifican los requisitos del codificador y decodificador para que cumpla con el perfil. Estos perfiles son: *Main Profile*, *Baseline Profile* y *Extended Profile*, y sus principales funciones pueden verse en la siguiente figura.

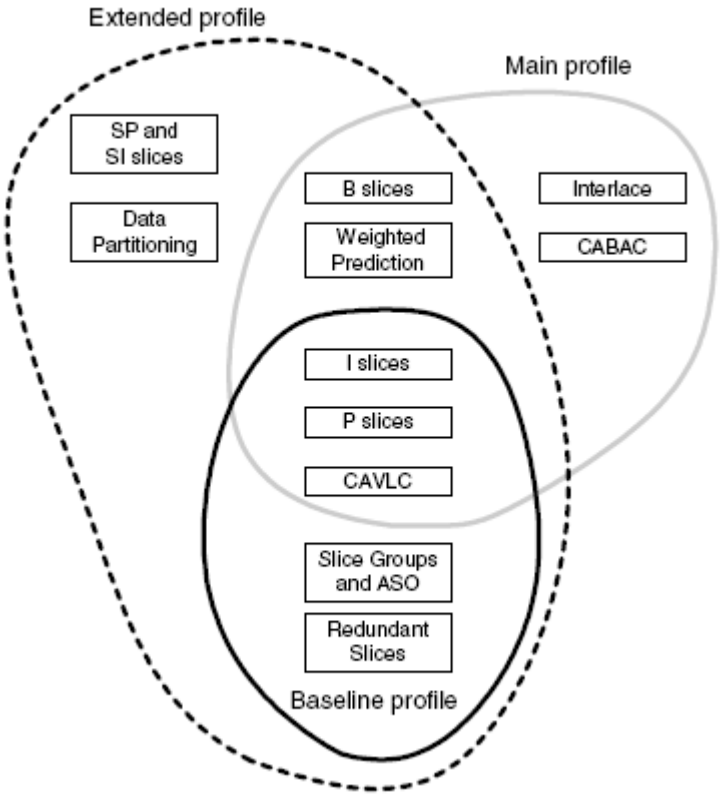


Figura 3.3 Perfiles definidos en H.264 y sus funciones.

El perfil *Baseline* proporciona codificación Intra e Inter y codificación entrópica con codificación adaptativa según el contexto de longitud variable (CAVLC).

El perfil *Main* incluye soporte para vídeo entrelazado, codificación Inter con tiras B, predicción ponderada, y codificación aritmética basada en contexto (CABAC).

El perfil *Extended* no soporta vídeo entrelazado, ni CABAC, pero tiene modos que permiten un eficiente cambio entre flujos codificados, y una mejor capacidad de recuperación ante errores.

La existencia de varios perfiles se debe a la gran cantidad de usos que tiene el vídeo actualmente, así como los aparatos que lo utilizan, haciendo que las funciones de codificación varíen según la aplicación. El perfil *Baseline* está pensado para su uso en videotelefonía, videoconferencia, y comunicaciones wireless. El perfil *Main* se orienta más a su uso en televisión y almacenamiento de vídeo, ya que aporta una mayor calidad de codificación y el perfil extended proporciona suficiente flexibilidad como para ser

utilizado en un amplio rango de aplicaciones. El perfil empleado durante la realización del proyecto será el *Main*.

3.2.2 Formato de datos

El estándar H.264 hace distinción entre dos capas de datos, una de vídeo, la *Video Coding Layer* (VLC) y otra de red, la *Network Abstraction Layer* (NAL). La salida del proceso de codificación serán datos VLC, los cuales son mapeados en NAL, antes de su transmisión o almacenamiento. Una secuencia de vídeo se representa como una secuencia de unidades NAL, que pueden ser transmitidas por una red, o guardadas en un archivo.

3.2.3 Imágenes de referencia

Un codificador H.264 puede usar varias imágenes, de las anteriormente codificadas, como referencia para la predicción por compensación de movimiento de cada macrobloque Inter que va a ser codificado. Esto permite al codificador buscar una predicción temporal del actual macrobloque más ajustada que si sólo pudiese utilizar la última imagen codificada.

Para que la predicción sea la misma tanto en el codificador como en el decodificador, ambos guardan una o dos listas de imágenes de referencia, que contienen imágenes que han sido previamente codificadas y reconstruidas. Según el número de referencias utilizadas para generar la predicción, hablaremos de tiras P, o de tiras B. Las predicciones codificadas de forma Inter en tiras P son generadas a partir de referencias en una sola lista, la lista 0. Las que pertenezcan a tiras B, serán obtenidas de dos listas, lista 0 y lista 1. En los siguientes apartados se explicarán en mayor profundidad los tipos de tiras que define el estándar, así como las características de cada lista de referencia.

3.2.4 Tiras

Un plano de vídeo se divide para su codificación en una o más tiras, cada una de las cuales contiene un número de macrobloques que puede variar de uno, al total de los que componen el plano. El número de macrobloques por tira no tiene por qué ser constante durante la codificación, ni siquiera dentro del mismo plano. Durante el desarrollo de este proyecto se ha trabajado solamente con tiras que ocupan la totalidad de un plano. Hay cinco tipos de tiras, y cada plano puede estar compuesto de distintos tipos de tira. A continuación los detallamos:

-Tiras I: Están compuestas por macrobloques I, cuya predicción se realiza mediante el uso de macrobloques previamente codificados pertenecientes a la misma tira, es decir, utilizando predicción Intra. Se encuentra definida en todos los perfiles.

-Tiras P: Están compuestas por macrobloques P y/o macrobloques I. Para cada macrobloque P la predicción se genera a partir de las referencias pertenecientes a la lista 0. Se encuentra definida en todos los perfiles

-Tiras B: Contiene macrobloques B y/o macrobloques I. Para cada macrobloque B, las predicciones se generan a partir de las referencias pertenecientes a la lista 0 y/o lista 1. Está definida en los perfiles *Main* y *Extended*.

Tiras SP: Están definidas en el perfil *Extended*, y por tanto no se usarán.

Tiras SI: Están definidas en el perfil *Extended*, y por tanto no se usarán.

A continuación, se detalla el contenido de una tira:

Figura 3.4 Sintaxis de una tira

Se puede ver que cada tira contiene dos partes bien diferenciadas, la cabecera y los datos. En la cabecera se definen el tipo de tira del que se trata, el plano al que pertenece, datos referentes al manejo de las referencias para dicha tira, etc. mientras que los datos son los macrobloques codificados (MB en la figura).

3.2.5 Macrobloques

Los macrobloques contienen datos codificados pertenecientes a una región de 16x16 píxeles de un plano del vídeo. En la figura siguiente se muestra un macrobloque con sus campos más importantes

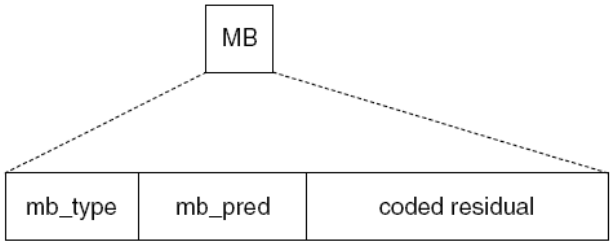


Figura 3.5 Sintaxis de un macrobloque

El campo *mb_type* determina si el macrobloque está codificado en Intra o Inter (P o B). *Mb_pred* indica la predicción Intra utilizada, en el caso de los macrobloques I, o las listas utilizadas como referencia en los macrobloques P y B. El campo *coded residual* contiene los coeficientes de la transformada codificados correspondientes al residuo.

3.2.6 Orden de codificación y visualización

Es importante destacar que el orden de codificación no tiene que ser el mismo que el de visualización de un vídeo, si no que éste dependerá del patrón de tiras con que se vaya a codificar.

Cuando se codifica una tira I, los macrobloques de tipo I que la componen son codificados utilizando predicciones generadas a partir de los macrobloques vecinos superior e izquierdo, por lo que siguiendo un orden de codificación dentro de la tira que vaya de izquierda a derecha y de arriba hacia abajo, estos macrobloques, en caso de que existan, siempre estarán disponibles. No ocurre así con las tiras P, que pueden contener macrobloques de tipo P, y que por tanto necesitarán que existan macrobloques ya codificados pertenecientes a planos anteriores (en el orden de visualización), los cuales son guardados en la lista 0. Es por ello que para poder codificar un macrobloque de tipo P, se necesitará haber codificado algún plano temporalmente anterior. Por último, las tiras de tipo B, pueden hacer uso de referencias tanto anteriores como posteriores (hablando de orden de visualización). Debido a esto, antes de codificar un macrobloque de tipo B será necesario disponer de macrobloques ya codificados pertenecientes a planos anteriores y posteriores al actual.

Estas restricciones condicionarán el orden de codificación y harán que pueda ser distinto del de visualización. Se puede comprobar que en el caso de que sólo se utilicen tiras de tipo I y P, el orden de codificación y visualización será el mismo, pero si se quiere hacer uso de tiras de tipo B, estos serán distintos. A continuación se muestra un ejemplo del orden de codificación cuando intervienen este tipo de tiras:

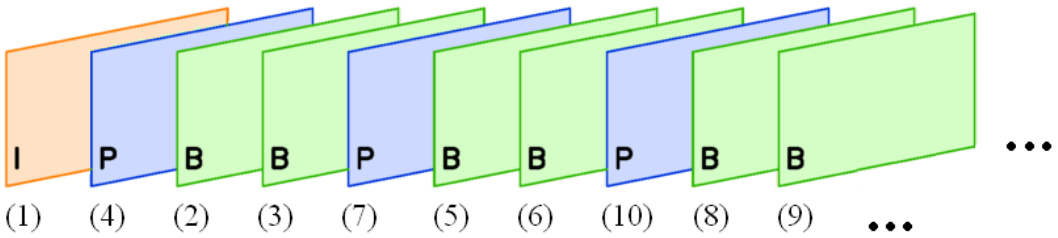


Figura 3.6 Ejemplo de orden de codificación

En la figura, el primer plano a codificar es de tipo I, el único posible en este caso, ya que no se dispone de ningún plano anterior codificado, ni por tanto de referencias. A continuación, se codifica un plano P, que puede contener macrobloques I y macrobloques P, y que podrá utilizar como referencia el plano 1 ya codificado. Aquí se puede ver que ya no coincide el orden de codificación con el de visualización, ya que este plano, el segundo en ser codificado, se corresponde con el cuarto en el orden de visualización. Esto se hace para que los dos planos siguientes, el 2 y el 3, que son de tipo B, dispongan de referencias futuras ya codificadas. A continuación se codificará el plano 2 de tipo B, que tendrá disponibles como referencias el plano 1 y el plano 4, y después el 3 que dispondrá como referencias pasadas del plano 1 y 2, y como referencia futura el plano 4. Se puede ver en la figura, que estas alteraciones en el orden de codificación continúan durante todo el proceso de codificación.

3.3 Predicción Intra

Como se ha explicado en apartados anteriores, la predicción en macrobloques Intra se forma a partir de macrobloques pertenecientes a la misma tira, y que ya han sido codificados. Esta predicción se realizará de forma distinta para las muestras de luminancia que para las de crominancia.

En la predicción Intra, se tienen dos tamaños de bloque posibles sobre los que realizar la predicción, *Intra16x16* (una predicción por macrobloque) o *Intra4x4* (16 bloques por cada macrobloque). Hay un total de nueve modos de predicción para los bloques 4x4 luma, y cuatro modos para los 16x16 luma, mientras que para los bloques de crominancia hay cuatro modos distintos. El codificador elegirá para cada bloque el modo de predicción que minimiza la diferencia entre la predicción generada P, y el bloque que va a ser codificado.

Modos de predicción Intra 4x4 luma

En la predicción 4x4 se divide cada macrobloque en 8 subbloques, realizando la predicción para cada uno de estos subbloques de manera independiente. Para cada uno de ellos el codificador probará los distintos modos de predicción luma, y escogerá aquel que minimice una función de coste, dependiente del residuo y de la tasa necesaria para codificar dicho residuo.

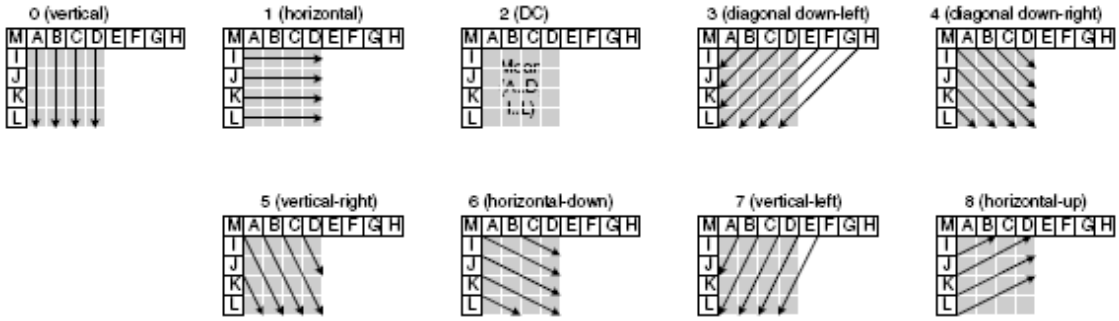


Figura 3.7 Modos de predicción 4x4 luma

En la figura anterior se muestra el funcionamiento de los distintos modos. Las flechas indican la dirección de predicción.

A continuación se detallan en una tabla todos los modos.

0 Vertical	Las muestras adyacentes superiores (A,B,C,D) son copiadas verticalmente, es decir, todas los píxeles por debajo tendrán el mismo valor del píxel adyacente superior.
1 Horizontal	En el modo horizontal, los píxeles extrapolados son los adyacentes por la izquierda (I,J,K,L) y se realiza de manera horizontal
2 DC	Se calcula la media de los valores adyacentes superiores y por la izquierda (media (A,D,I,L), y ese valor es copiado a todos los píxeles del subbloque 4x4.
3 Diagonal abajo-izqda	El valor de los píxeles es interpolado con un ángulo de 45º entre las muestras situadas a la izquierda y arriba.

4 Diagonal abajo-dcha	El valor de los píxeles es extrapolado con un ángulo de 45º en dirección descendente y hacia la derecha
5 Vertical derecha	La extrapolación se realiza con un ángulo de aproximadamente 26.6º a la izquierda de la vertical.
6 Horizontal abajo	El valor de los píxeles es extrapolado con un ángulo de aproximadamente 26.6º hacia abajo de la horizontal
7Vertical izquierda	Se extrapola con un ángulo de aproximadamente 26.6º hacia la derecha de la vertical
8Horizontal arriba	Se interpolan las muestras con un ángulo de aproximadamente 26.6º sobre la horizontal

Tabla 3.1 Modos de predicción Intra4x4 luma

Predicción Intra16x16 luma.

En este caso, el macrobloque es tratado como unidad de predicción, es decir, no se dividirá en bloques de menor tamaño. Como ya se ha dicho, en este tipo de predicción hay cuatro modos distintos:

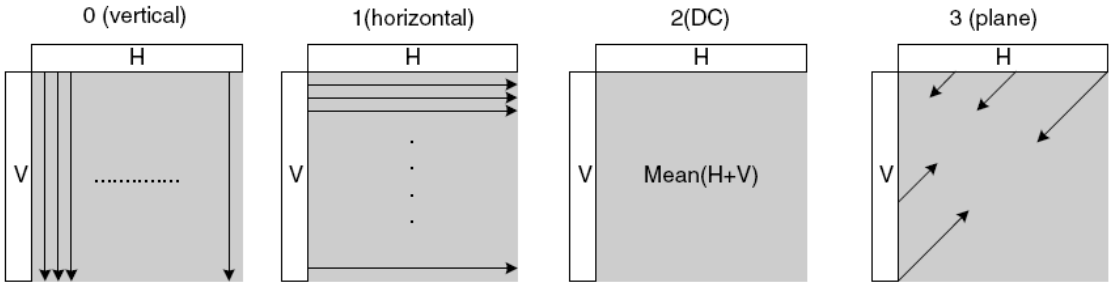


Figura 3.8 Modos de predicción Intra

Los tres primeros modos, *vertical*, *horizontal* y *DC* funcionan de la misma forma que en la predicción 4x4, mientras que el modo *plane* utiliza una función lineal sobre las muestras del bloque adyacente superior y a la izquierda para calcular la predicción de los píxeles

Predicción Intra croma

Para el cálculo de la predicción de la componente de crominancia el macrobloque se divide en bloques de 8x8 muestras. Cada uno de estos bloques es predicho mediante el uso de muestras de crominancia ya codificadas pertenecientes al macrobloque superior y/o al macrobloque situado a la izquierda. Hay cuatro modos de predicción disponibles, los cuales son prácticamente iguales que los usados en la predicción *Intra16x16* luma, salvo por la numeración, que queda así: *DC* (modo 0), *horizontal* (modo 1), *vertical* (modo 2), y *plane* (modo 3). Hay que resaltar que ambas componentes de crominancia utilizarán el mismo modo de predicción.

Se ha visto que la predicción Intra se realiza utilizando los macrobloques ya codificados situados a la izquierda y sobre el macrobloque actual. En el caso de que alguno de estos macrobloques no esté disponible porque se trate de un borde de la imagen, los modos de predicción que los necesiten no serán probados excepto en el caso del modo DC, que siempre será probado, ajustando su funcionamiento al número de muestras disponible. Por ejemplo, en el caso de estar codificando un macrobloque situado en el borde izquierdo de un plano, no se dispone del adyacente por la izquierda,

por lo que el modo *horizontal* nunca se probará, ni tampoco ninguno de los que hagan uso de este macrobloque.

La elección del modo de predicción *Intra4x4* debe ser codificada para que esté disponible en el decodificador, y esto puede requerir un alto número de bits. Para evitar esto, y ya que los modos de predicción Intra de los bloques vecinos están altamente correlados, se utilizará codificación predictiva, indicando mediante un *flag*, si es diferente del esperado, y en caso de que sea diferente, de qué modo se trata. Para la codificación *Intra16x16*, esto no será utilizado, debido a que el número de modos es mucho menor.

3.4 Predicción Inter

Como se comentó en la introducción, la predicción Inter hará uso de la correlación temporal para generar una predicción del macrobloque actual, mediante el uso de macrobloques pertenecientes a planos ya codificados.

3.4.1 Compensación de movimiento basada en una estructura en árbol

Como ya se ha visto en la predicción Intra, un macrobloque puede ser dividido en bloques más pequeños y realizar sobre estos subbloques el cálculo de la predicción. En la predicción Inter el macrobloque también podrá ser dividido, con el fin de obtener predicciones más precisas. A cada una de estas formas de dividir un macrobloque se le llama modo.

La componente de luminancia de cada macrobloque (16x16 muestras) puede ser dividida en cuatro formas distintas. Como un macrobloque 16x16, dos particiones 16x8, dos 8x16 o cuatro particiones 8x8. Además, en el modo 8x8, cada uno de los cuatro submacrobloques puede ser a su vez dividido en otras cuatro formas diferentes, tanto como una partición 8x8, dos 8x4, dos 4x8 o cuatro particiones de tamaño 4x4. Es importante destacar que para cada una de las particiones se realizará la estimación y compensación de movimiento de manera independiente. A continuación se muestra una figura con las distintas posibilidades.

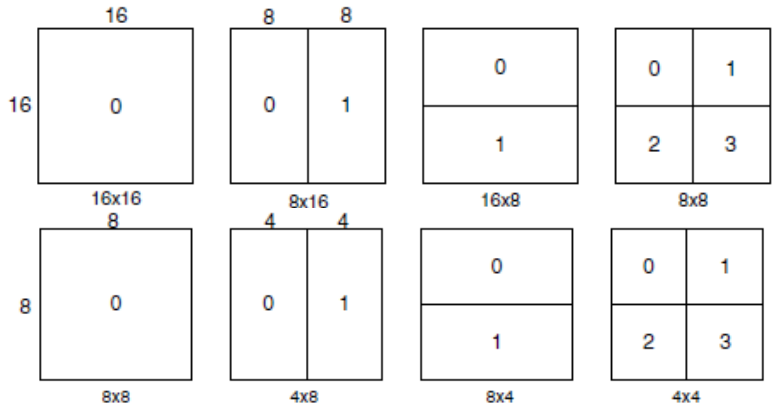


Figura 3.9 Modos Inter

Este método de particionamiento de los macrobloques en diferentes tamaños para la compensación de movimiento se conoce como “compensación de movimiento con estructura en árbol”.

3.4.2 Selección de referencia

Dentro de la predicción Inter, dependiendo del tipo de tira que se vaya a codificar, se podrá hacer uso de una o dos referencias con el fin de generar la predicción; si la tira es de tipo P, el codificador elegirá un bloque ya codificado perteneciente a la lista 0 como predicción, mientras que si la tira es tipo B, se podrán utilizar uno o dos macrobloques ya codificados, pertenecientes a las listas 0 y 1. Tanto el codificador como el decodificador mantendrán estas dos listas que contienen los planos que pueden ser utilizados como referencias. En ellas se almacenan los planos ya codificados y reconstruidos, tanto pasados como futuros.

Para cada macrobloque, el codificador probará un número de planos (este número es seleccionable en la configuración del codificador) como referencias, y realizando la estimación y compensación y movimiento, decidirá cual es el que ofrece un mejor residuo en base a una función de coste, que estimará la tasa y la distorsión a la que dará lugar ese residuo.

Cuando el codificador va a realizar la codificación de un macrobloque P, se dirigirá a la lista 0, con el fin de ver que planos ya codificados y reconstruidos están disponibles. En esta lista, las referencias pasadas más cercanas al macrobloque que queremos codificar tendrán los índices más bajos, mientras que las referencias futuras disponibles más próximas se colocarán a continuación. En la tabla 3.2, se muestra un ejemplo de la lista 0. En ella aparece el estado de la lista cuando el codificador se dispone a buscar la predicción para un macrobloque del plano 70. Se puede ver que en las primeras posiciones de la lista se encuentran los planos pasados, mientras que a continuación aparecen los planos futuros

Índice	Lista 0
0	69
1	68
2	67
3	71
4	72

Tabla 3.2 Ejemplo de lista 0

Los macrobloques de tipo B, además de poder utilizar los planos de la lista 0, tendrán también acceso a los planos de referencia de la lista 1. En ella los planos reconstruidos se colocan de manera inversa que en la lista 0, es decir, los planos futuros y más cercanos al macrobloque actual tendrán índices bajos, mientras que los planos pasados más próximos se irán colocando a continuación. A continuación se muestra el estado de la lista 1 cuando el codificador se dispone a buscar la referencia para un macrobloque perteneciente al plano 70.

Índice	Lista 0
0	71
1	72
2	69
3	68
4	67

Tabla 3.3 Ejemplo de lista 1

En este caso el codificador tiene varias opciones al tener las dos listas disponibles. Si selecciona una sola, realizará el mismo proceso que si de un plano P se tratara, pero pudiendo elegir entre un plano almacenado en lista 0 o en lista 1. Si por el contrario utiliza dos planos dispondrá de varias posibilidades. Al seleccionar un plano de cada lista podrá elegir una referencia pasada y otra futura, dos referencias pasadas, o dos referencias futuras. Este proceso se denomina bipredicción, y a la decisión de cual de todas las opciones disponibles se elige se le llama selección de dirección de predicción. Estas opciones se muestran en la siguiente figura:

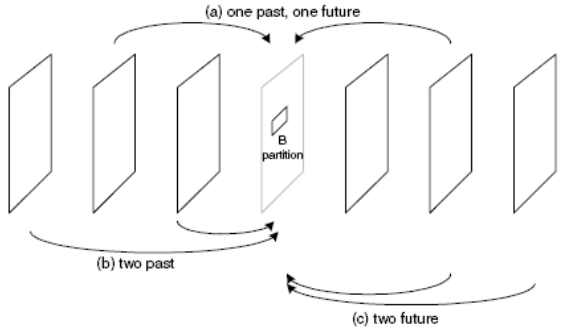


Figura 3.10 Distintas selecciones de referencia

Bipredicción

En la bipredicción, un bloque de referencia (del mismo tamaño que el actual partición o submacrobloque) es creado a partir de las referencias de la lista 0 y la lista 1. Dos zonas de referencia son obtenidas mediante compensación de movimiento de la lista 0 y la lista 1 respectivamente, (que por tanto requerirán dos vectores de movimiento), y cada muestra del bloque de predicción es calculada como la media de las muestras de la lista 0 y la lista1, utilizando la siguiente ecuación:

$$pred(i, j) = pred\,0(i, j) + pred\,1(i, j) + 1) >> 1$$

Ecuación 3.1 Bipredicción

Donde pred0(i,j) y pred1(i,j) son las muestras de predicción obtenidas de la lista 0 y la lista 1 y pred(i,j) es una muestra bipredicha. Una vez calculadas todas las muestras, el residuo se calcula de la misma forma, restando al macrobloque original las muestras del macrobloque bipredicho.

3.4.3 Vector de movimiento

Cada bloque codificado en modo Inter necesitará su propio vector de movimiento independiente, es decir, si se ha seleccionado un modo Inter16x8, se necesitarán dos vectores de movimiento, si se elige un modo *Inter8x8*, se necesitarán cuatro vectores, etc. Como ya se ha comentado, cada uno de estos vectores deberá ser codificado y transmitido.

Hay que señalar que los vectores de movimiento podrán tener una resolución de hasta de un cuarto de píxel cuando hablemos de luminancia, y de un octavo de muestra cuando nos refiramos a crominancia. Con el fin de alcanzar esta resolución, se deberá interpolar la imagen en las posiciones no enteras, ya que en ellas no existen píxeles.

En la siguiente figura se muestran dos ejemplos de predicción: una entera (píxel) y otra no (subpíxel).

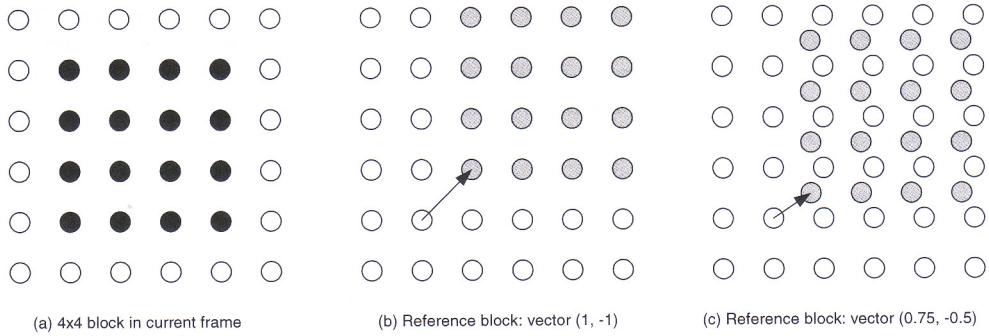


Figura 3.11 Distintos tipos de resolución de los vectores de movimiento

Hay que destacar que este proceso de interpolación ha sido mejorado en el estándar h.264, ya que los métodos utilizados en estándares anteriores no eran muy precisos, mientras que en el actual, este proceso se realiza mediante un filtro FIR de interpolación de 6 coeficientes.

En el caso de la crominancia, se utilizarán otros métodos de interpolación, debido a que el formato de vídeo elegido le asigna una resolución más baja, como se ha comentado anteriormente

Predicción del vector de movimiento

Codificar un vector de movimiento por cada bloque puede suponer un coste alto en lo que a número de bits se refiere, especialmente si se ha elegido un modo que tenga bloques pequeños como el *Inter8x8*. Éste es el motivo por el que se utiliza codificación predictiva de los MV, parecida a la codificación DPCM realizada en los macrobloques.

En una secuencia, las zonas cercanas en un plano suelen seguir el mismo patrón de movimiento, por lo que es lógico que los vectores de movimiento sean bastante parecidos, o lo que es lo mismo, estén altamente correlados. Haciendo uso de esa característica se llevará a cabo la codificación predictiva. La forma de proceder en el estándar es como sigue:

1. Se obtiene una predicción MVp del vector de movimiento con la información de los vecinos (más adelante explicaremos cómo).
2. Se resta la predicción MVp al vector actual para generar el vector diferencia MVD.
3. Codificamos y transmitimos MVD.

Al enviar sólo la diferencia, se consigue que el número de bits a utilizar sea menor, ya que el módulo de MVD será menor.

La forma de generar la predicción MVp dependerá del tamaño de la partición utilizada en los vecinos ya codificados.

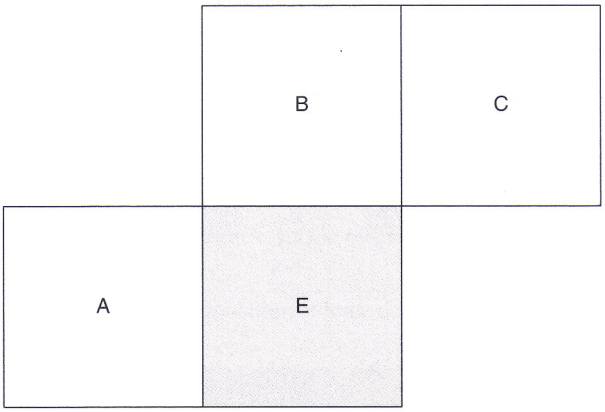


Figura 3.12 Bloque a codificar y vecinos con el mismo tipo de partición

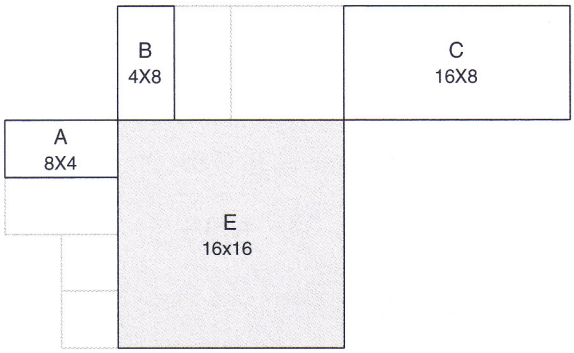


Figura 3.13 Bloque a codificar y vecinos con distinto tipo de partición

Para llevar a cabo la predicción se utilizarán el bloque vecino inmediatamente a la izquierda de del bloque a codificar (A), el que se encuentre encima (B), y el que esté situado arriba a la derecha (C). Siendo E el bloque a codificar, y siguiendo la nomenclatura citada (que se corresponde con la figura 3.13), el procedimiento para generar MVp es el siguiente

1. Para particiones de tamaño distinto a 16x8 y 8x16, MVp será la mediana de los vectores de movimiento de las particiones A, B y C.
2. Para los bloques 16x8, el MVp de la partición 16x8 superior se calcula a partir de B y el de la partición 16x8 inferior a partir de A.
3. Para los bloques 8x16 la predicción de la partición de la izquierda se realiza a partir de A y la de la derecha a partir de C.
4. Para macrobloques sin codificar (*Skipped*), el MVp se calcula como en (1), como si el bloque fuese codificado en modo *Inter16x16*.

Si alguno de los vecinos necesarios para la predicción del vector no está disponible, el proceso de predicción se modifica adecuadamente.

El proceso seguido en el decodificador para generar cada MVp es idéntico. Una vez obtenido MVp bastará con sumárselo al vector diferencia decodificado MVD para obtener el vector de movimiento original.

3.4.4 *Skip*

Este modo sólo estará disponible para tiras tipo P. En él, no se codifica información sobre los vectores de movimiento, ni residuo, siendo por tanto el que resulta en una menor carga de bits. Cuando se decodifica un macrobloque como *Skip*, se calcula el vector de movimiento predicho, y se aplica sobre la imagen de referencia, sin hacer uso de vectores de movimiento diferencial, ni de residuo.

Este modo será elegido principalmente en las zonas homogéneas o estacionarias de la imagen, y cuanto mayor sea el QP, ya que en este caso, los coeficientes del residuo tenderán a valer cero.

3.4.5 Modo *Directo*

Este modo sólo esta disponible para tiras B. En este caso no se enviará vector de movimiento. En vez de eso, el decodificador calcula los vectores de lista 0 y lista 1 en base a vectores anteriormente codificados y los utilizará para llevar a cabo la compensación de movimiento mediante bipredicción en las muestras del residuo.

Suele ocurrir que cuando se utilizan valores de QP elevados, es decir, escalones de cuantificación grandes, los coeficientes del residuo tienden a anularse, por lo que lo más costoso pasa a ser el tener que enviar las coordenadas de los vectores de movimiento. Será en este tipo de casos cuando se optará por este modo.

Existen dos tipos de modo *Directo*, el espacial y el temporal, y mediante el uso de un flag, se indicará al decodificador cual se ha utilizado.

Modo Directo espacial

En este caso se realiza la predicción del vector de movimiento a partir de los vecinos, siendo el procedimiento el que sigue.

Se calculan los vectores de movimiento predichos de las listas 0 y 1, y si el macrobloque cosituado en la primera referencia de la lista 1 tiene un vector de movimiento cuyo módulo es menor que +-1/2 muestras de luminancia (y en algún otro caso) entonces uno o ambos vectores son puestos a 0. En cualquier otro caso se utilizan los dos vectores calculados para realizar la compensación de movimiento por bipredicción.

Modo Directo temporal

En este caso la predicción se realiza de la manera siguiente

1 Se busca la referencia en lista 0 para el macrobloque cosituado en la imagen de la lista 1, la cual se convertirá en la imagen de referencia en lista 0 para la imagen actual.

2.-Se calcula el vector de movimiento para la referencia en lista 0 obtenida.

3.-Se realiza un escalado sobre el vector de movimiento basado en la distancia entre la imagen actual y la referencia en lista 1, y el resultado es el vector de movimiento para la lista 1.

4.-Se vuelve a realizar un escalado basado en al distancia entre el plano actual y la imagen referencia en la lista 0, siendo el resultado el vector de movimiento para la lista 0.

Hay que resaltar que durante el proyecto se utilizará el modo *Directo* espacial.

3.5 Transformación

Como se ha explicado en la introducción, el proceso de transformación permitirá una compresión más eficiente. El objetivo al realizar la transformación será que en el nuevo dominio la correlación sea menor, dejando un número pequeño de coeficientes que sean importantes para la imagen, y un gran número de coeficientes que no afecten tanto a la visualización.

3.5.1Transformada DCT

Como se ha comentado anteriormente, se hará uso de la transformada DCT con el fin de decorrelar el residuo, para mejorar la eficiencia en la codificación.

El proceso de transformación consistirá en pasar de una matriz X de tamaño NxN que contenga los valores de luminancia del residuo para cada píxel, a una matriz Y con los valores de frecuencia, manteniendo las dimensiones NxN. Al realizar esta transformación, gran parte de los valores serán cero, ya que en un solo bloque es bastante poco probable que se encuentren representadas todas las frecuencias, facilitando por tanto la codificación. La acción de la DCT, se puede describir en términos de una matriz de transformación A como sigue:

$$\mathbf{Y = AXA^T}$$

y de su inversa, la IDCT:

$$\mathbf{X = A^T YA}$$

Dónde X es la matriz anteriormente citada, Y la matriz transformada, y A es una matriz de transformación de tamaño NxN. Los elementos de la matriz A se definen como:

$$A_{ij} = C_i \cos\frac{(2j+1)i\pi}{2N} \qquad \text{donde} \qquad C_i = \sqrt{\frac{1}{N}}(i=0), \quad C_i = \sqrt{\frac{2}{N}}(i>0)$$

Se puede comprobar fácilmente que si se escribe la matriz de tamaño 16x16, aparecen varios números racionales, que implicarán gran cantidad de multiplicaciones en el codificador, ya que la operación de transformada se realizará para cada uno de los residuos de tamaño 16x16 en cada uno de los modos posibles. Es por ello que se utilizará una aproximación a la transformada DCT de tamaño 4x4, obteniendo la siguiente matriz de transformación:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

Con la nueva matriz de transformación, bastará con realizar sumas y desplazamientos en los bits para obtener la matriz transformada, ahorrando por tanto gran cantidad de tiempo en el proceso de codificación y decodificación. A cambio, parte del proceso se llevará a cabo durante la cuantificación mediante un factor de escalado como se verá a continuación.

3.5.2 Transformada Hadamard

Hay dos casos en los que se utiliza la transformada Hadamard para el residuo en lugar de la transformada DCT. Para los coeficientes DC de luminancia de los macrobloques *Intra16x16* se utiliza una transformada Hadamard 4x4, mientras que para la crominancia DC de los macrobloques Intra se utiliza un tamaño 2x2.

El motivo de su utilización es que su matriz de transformación sólo contiene valores de +-1, por lo que se ganará en velocidad respecto a la transformada DCT. La expresión general de esta transformada es:

$$\mathbf{Y = HXH^{-1}}$$

Siendo X la matriz de coeficientes, H la matriz de transformación Hadamard, e Y la matriz de coeficientes transformados.

Si el macrobloque se codifica con el modo de predicción *Intra16x16* cada bloque de residuo de tamaño 4x4 es transformado en primer lugar utilizando la transformada DCT simplificada y a continuación se le aplica la transformada Hadamard de tamaño 4x4. La matriz de transformación en este caso es:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

Como se puede observar, se trata simplemente de una serie de sumas y restas.

En el caso de los bloques de crominancia 4x4 Intra, estos son también transformados en primer lugar haciendo uso de la transformada DCT simplificada, y a continuación, se les aplica la transformación Hadamard de tamaño 2x2, siendo la matriz de transformación:

$$H = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

3.6 Cuantificación

El proceso de cuantificación es un método de compresión con perdidas en el cual se pasa de un rango de valores a otro más reducido.

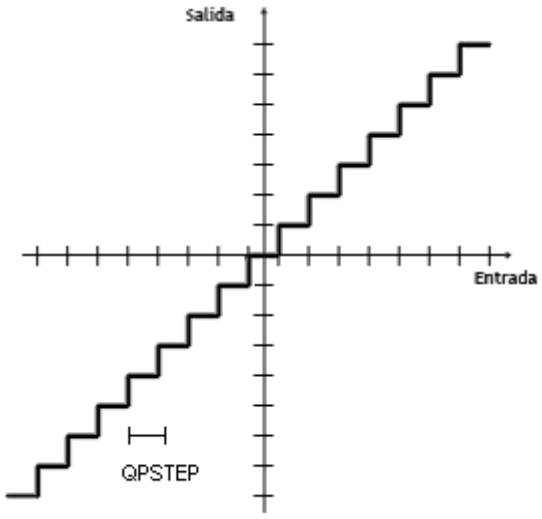


Figura 3.14 Ejemplo de cuantificador

Un cuantificador básico sigue la siguiente fórmula

$$Z_{ij} = \text{round}\left(\frac{Y_{ij}}{Qstep}\right)$$

Ecuación 3.2 Operación básica de cuantificación

Donde Y son los coeficientes transformados, Qstep es el tamaño del escalón de cuantificación, y Z la salida del cuantificador, es decir, los coeficientes transformados cuantificados.

El estándar soporta 52 valores de Qstep, los cuales son indexados por el parámetro QP (*Quantisation Parameter*). La gran cantidad de valores posibles que puede tomar el escalón de cuantificación permite controlar de forma muy precisa y flexible la relación entre tasa y calidad. Esto hará que el parámetro QP sea de vital importancia durante la realización del proyecto, debido a la relación existente entre QP y probabilidad a priori de que se seleccione un modo. Hay que destacar que también permite elegir distintos valores de QP para luminancia y crominancia.

Q_{step}	0.62	1.25	2.5	5	10	20	40	80	160	224
QP	0	6	12	18	24	30	36	42	48	51

Tabla 3.4 Relación entre Q_{step} y *QP*.

Como se explicó anteriormente, debido a la simplificación llevada a cabo en la transformada DCT, se deberá realizar un proceso de post-escalado, que estará implícito en el cuantificador, quedando la fórmula:

$$Z_{ij} = \text{round}\left(\frac{Y_{ij} \cdot PF}{\text{Qstep}}\right)$$

Ecuación 3.3 Operación de cuantificación con factor de post-escalado

donde PF es el factor de post-escalado con el que se evitan las divisiones en el proceso de transformación.

3.7 Reescalado

Mediante el proceso de reescalado obtendremos, a partir de la información que haya llegado al decodificador, los coeficientes transformados. Realiza básicamente la operación inversa a la de la cuantificación, aunque debido al error introducido por el proceso de cuantificación, los coeficientes obtenidos no serán los mismos que los que se tenían a la entrada del cuantificador. La fórmula que utiliza es la siguiente:

$$Y'_{ij} = Zij \cdot Q_{step}$$

Ecuación 3.4 Operación de reescalado

Donde Z_{ij} son los valores de entrada al proceso de reescalado e Y'_{ij} son los coeficientes resultantes.

Como se ha explicado anteriormente, el proceso de cuantificación introduce un factor de postescalado (PF) para simplificar la matriz de transformación. Debido a este factor, la fórmula aplicada en el reescalado queda así:

$$Y'_{ij} = Zij \cdot Q_{step} \cdot PF \cdot 64$$

Ecuación 3.5 Operación de reescalado con factor de post-escalado

Donde se tiene un factor de escalado constante de 64, para evitar errores de redondeo. Los valores a la salida de la transformación inversa se dividirán entre 64 para eliminar este factor de escalado.

3.8 Codificación entrópica

Una vez obtenidos los coeficientes del residuo mediante el uso de la transformada, y realizado ya el proceso de cuantificación se procede a reordenar los coeficientes con el fin de tener los de mayor energía en primer lugar. Para ello se utiliza un orden en zigzag, como se muestra en la siguiente figura:

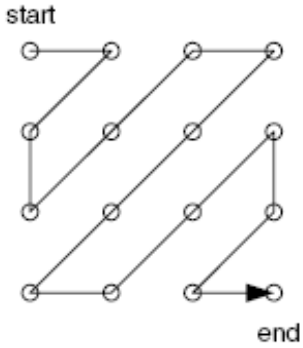


Figura 3.15 Reordenamiento de coeficientes en zigzag

Los elementos se codifican usando códigos de longitud variable (CAVLC) o códigos aritméticos adaptables al contexto (CABAC), siendo esto seleccionable en el codificador. A continuación se explican en detalle ambos modos de codificación.

3.8.1 Codificación adaptativa según el contexto de longitud variable (CAVLC)

Este es el método utilizado para codificar los coeficientes transformados y ordenados del residuo. *CAVLC* está diseñado para aprovechar las características de los bloques 4x4 ya cuantificados.

1.- Una vez hecha la predicción, la transformación y la cuantificación, los bloques contienen casi en su totalidad ceros. Es por ello que *CAVLC* utiliza codificación run-level para representar cadenas de ceros de manera compacta.

2.- Los coeficientes distintos de cero más altos después de ordenarlos son a menudo secuencias de +-1 (llamadas *'Trailing Ones'*) por lo que *CAVLC* señaliza de manera compacta el número de coeficientes +-1.

3.- Ya que el número de coeficientes distinto de cero en bloques vecinos está correlado, este se codifica utilizando una tabla de búsqueda, y la elección de esta tabla dependerá del número de coeficientes distintos de cero de los bloques vecinos.

4.- El valor de los coeficientes distintos de cero tiende a ser mayor al comienzo del array reordenado (cerca del coeficiente *DC*) y se hace más pequeño a medida que se acerca a las altas frecuencias. *CAVLC* se aprovecha de esto adaptando la elección de la tabla de búsqueda para codificar este valor a los valores codificados recientemente.

El procedimiento de codificación *CAVLC* se detalla a continuación:

1.- Codificación del número de coeficientes y *Trailing Ones*.

En primer lugar se codifica el número de coeficientes distintos de cero que hay en el bloque. Para ello se dispone de tres tablas de búsqueda, la primera sesgada hacia valores altos, la segunda hacia valores intermedios y la tercera hacia valores bajos. Como se ha comentado anteriormente, la elección de la tabla dependerá del número de coeficientes no nulos de los vecinos.

Para la codificación del número de +-1 seguidos se utilizan valores de 0 a 3. En caso de que haya más de tres, estos son tratados como casos especiales, y serán codificados como coeficientes normales.

2.- Codificación del signo de cada *Trailing One*.

Para cada *Trailing One* (training +-1) se codifica el signo con un bit (0=+.1=-) en orden inverso, comenzando por las altas frecuencias.

3.- Codificación de los niveles del resto de coeficientes no nulos.

El nivel (signo y magnitud) de cada coeficiente distinto de cero restante es codificado en orden inverso, comenzando con las frecuencias altas. Esta codificación se realiza mediante el uso de un sufijo de longitud variable (mayor cuanto mayor sean los niveles previamente codificados) y un prefijo de longitud fija.

4.- Codificación del número total de ceros antes del último coeficiente.

Esto se debe a que muchos bloques contienen ceros al comienzo del array, y así se evita la codificación de un cero como nivel.

5.- Codificación del número de ceros que precede a cada coeficiente no nulo.

El número de ceros que preceden a cada coeficiente no nulo se codifica en orden inverso.

3.8.2 Codificación Aritmética binaria adaptable al contexto (CABAC)

El uso de codificación *CABAC* permite realizar buenas compresiones mediante (a) la selección de modelos de probabilidad para cada elemento sintáctico de acuerdo con el contexto de dicho elemento, (b) adaptando la estimación de probabilidad basándose en estadísticas locales, y (c) utilizando codificación aritmética en lugar de codificación de longitud variable. La codificación de un símbolo mediante *CABAC* comprende las siguientes etapas:

1.-Binarización: *CABAC* hace uso de codificación aritmética binaria, lo que significa que sólo se codifican decisiones binarias (0 o 1). Un símbolo no binario (por ejemplo un coeficiente de una transformada, un vector de movimiento, o cualquier símbolo con más de dos posibles valores) es convertido en binario antes de realizar la codificación aritmética.

2.- Selección del modelo contextual. Un modelo contextual es un modelo de probabilidades para uno o más grupos de bits de un símbolo binarizado y que es elegido de entre los modelos disponibles dependiendo de las estadísticas de los símbolos codificados recientemente. El modelo contextual guarda la probabilidad de cada grupo de bits siendo 1 o 0.

3.- Codificación aritmética: Un codificador aritmético codifica cada grupo de bits de acuerdo con el modelo de probabilidades seleccionado.

4.- Actualización de las probabilidades: El modelo contextual elegido se actualiza en función del valor actual codificado.

4. La decisión de modo en el estándar H.264

4.1 El problema de la tasa-distorsión

En los apartados anteriores se ha visto parte de la gran cantidad de opciones que tiene el codificador para comprimir una secuencia de vídeo. El problema que se plantea es qué combinación de estos parámetros permitirá una codificación óptima, y cómo definir el término "óptimo" en dicha codificación. Se podría pensar que óptimo en codificación de vídeo es la configuración que resulta en una mayor calidad de imagen, pero eso conllevaría una cantidad de bits inmensa, que dificultaría que se pudiese almacenar o transmitir dicho vídeo. Por "óptima" entenderemos aquella configuración que permita obtener una imagen reconstruida lo más parecida a la original, con una tasa que se mantenga por debajo de un nivel previamente fijado. Matemáticamente hablando se intentará minimizar una función de dos variables sujeta a una determinada restricción:

$$\min D(K) \quad \text{sujeto a} \quad R(K) \le R_c$$

Ecuación 4.1 Condicion de optimización

Donde D es la distorsión, R es la tasa, R_c es la restricción de la que se habla anteriormente, y K son las distintas posibilidades de codificación.

Con el objetivo de resolver cómoda y eficientemente el problema de minimización con restricciones planteado, transformaremos éste en uno sin restricciones mediante el método de los multiplicadores de Lagrange. Así, será equivalente minimizar la ecuación anterior a minimizar la función siguiente:

$$J(k) = D(k) + \lambda R(k)$$

Ecuación 4.2 Función de coste con multiplicadores de Lagrange

Donde D es la distorsión, R es la tasa, λ es el multiplicador de Lagrange, y J será el 'coste' de utilizar una configuración determinada k.

El codificador barrerá las distintas posibilidades de codificación k y se quedará con la que resulte en un menor coste J. Mediante este método obtendremos los parámetros que harán que el proceso de codificación sea óptimo en el sentido definido anteriormente.

Viéndolo gráficamente, si calculamos la tasa y la distorsión para cada una de las configuraciones posibles, obtendremos un conjunto de puntos R-D (tasa-distorsión). Al ponerlos en forma de gráfica, se puede observar que a medida que aumenta la distorsión, disminuye la tasa, pero también que existen puntos que para una misma tasa ofrecen mejores valores de distorsión. A continuación se muestra un ejemplo de una gráfica R-D.

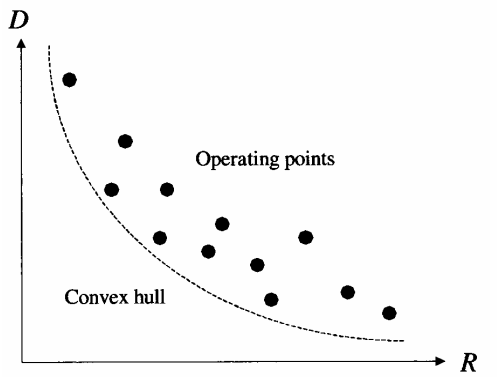


Figura 4.1 Distintos puntos de trabajo R-D

Si igualamos a cero y derivamos la ecuación (II), obtendremos los puntos de mínimo:

$$\frac{\partial J}{\partial R} = \frac{\partial J}{\partial R} + \lambda = 0 \Rightarrow \lambda = -\frac{\partial D}{\partial R}$$

Ecuación 4.3 Minimización de la función de coste

Vemos que para cada valor de λ tendremos una solución única, que será una recta tangente a la curva mostrada en la figura 5.1. A continuación mostramos un ejemplo de un conjunto de soluciones para el problema de la gráfica anterior

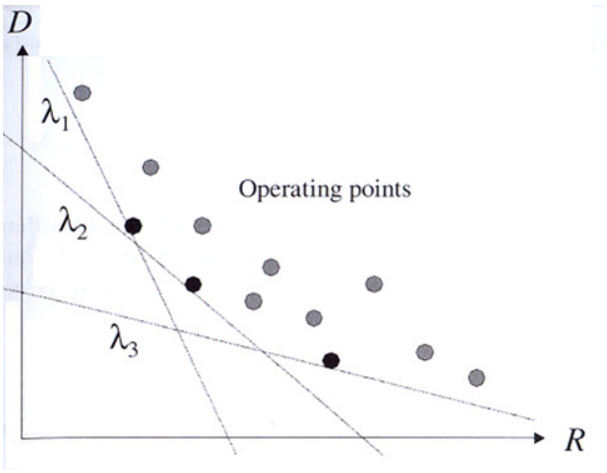


Figura 4.2 Minimización del coste para tres valores de λ

El impedimento que nos encontramos es que debido a la gran cantidad de parámetros que existen en el codificador, es imposible que se comprueben todas las combinaciones, ya que supondría un coste computacional inaceptable en cualquier tipo de aplicación de vídeo.

Para poder abordar este problema, se fijará el parámetro QP (*Quantization Step*) y se establecerá una relación entre este parámetro y λ. En la función de coste definida en la ecuación 4.2, el parámetro λ es el encargado de evaluar la importancia que se la da a la tasa respecto a la distorsión. Se puede ver que el parámetro QP cumple una función parecida en el codificador, ya que a mayor valor de QP más coeficientes serán cuantificados como 0 y por tanto será más importante en la tasa final el tamaño de los vectores de movimiento que la información del residuo. Se concluye por lo tanto que el valor de lambda crecerá con el valor del parámetro QP.

Una vez explicada la teoría general, pasaremos a comentar como actuará el software de referencia. Una de las primeras decisiones que se deberá tomar en las tiras P y B será la de seleccionar la mejor predicción y referencia posible para cada macrobloque. Para ello el codificador deberá realizar una búsqueda sobre las referencias disponibles, con el fin de encontrar la que resulte en una compensación de movimiento óptima. Para llevar a cabo esta decisión, se utilizará un coste que tomará aproximaciones tanto de la tasa como de la distorsión con el fin de realizar el proceso de decisión de una manera más rápida. Como distorsión, tomará la diferencia entre el macrobloque predicho y el original, obteniendo una aproximación de la distorsión real. En cuanto a la tasa, se considerará solamente la tasa necesaria para transmitir los vectores de movimiento, obviando por ejemplo los bits necesarios para transmitir el residuo. La función de coste quedará:

$$J_{motion} = SAD(\mathrm{MV}, \mathrm{Ref}) + \lambda_{\mathrm{motion}} R(\mathrm{MV}, \mathrm{Ref})$$

Ecuación 4.4 Función de coste para la estimación de movimiento

Donde SAD es la suma del módulo de las diferencias, R la tasa aproximada debida a los vectores de movimiento, y $\lambda_{\mathrm{motion}}$ es el multiplicador de Lagrange que pondera ambos factores.

Una vez obtenidas la mejor referencia, y los vectores de movimiento que indican dónde se encontró la mejor predicción, se deberá elegir el modo "óptimo" en el sentido definido al comienzo del apartado. Para ello se calculará la distorsión "real", es decir, la SSD (*Sum of Squared Differences*) entre el macrobloque original y el reconstruido (aquel que se obtiene tras los procesos de DCT, codificación entrópica, IDCT,..). Además, una vez obtenido el residuo, se calculará la cantidad de bits necesarios para codificar dicho residuo, junto con los vectores de movimiento y las cabeceras, y se aplicará la siguiente función de coste:

$$J_{\mathrm{mode},i} = SSD(\{\mathrm{MV}\}_i, \{\mathrm{Ref}\}_i, i) + \lambda_{\mathrm{mode}} R(\{\mathrm{MV}\}_i, \{\mathrm{Ref}\}_i, i)$$

Ecuación 4.5 Función de coste para la decisión de modo

Donde λ_{mode} es el multiplicador de Lagrange para un determinado modo, R es el número de bits necesario para codificar las cabeceras, vectores de movimiento, los índices de referencias, y el residuo transformado. Por lo tanto, como ya se comentó anteriormente, el proceso de decisión de modo consistirá en calcular el coste J_{mode} para cada uno de los tamaños de bloque y quedarnos con aquel que obtiene el mínimo.

Dicho proceso queda resumido, para el estándar H.264, en el siguiente esquema:

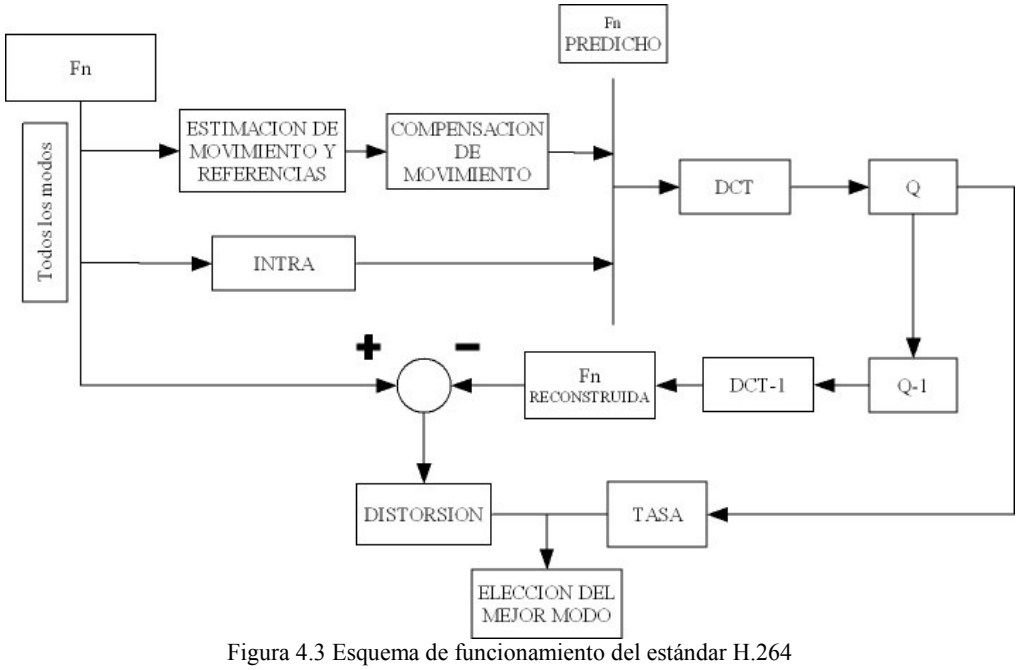


Figura 4.3 Esquema de funcionamiento del estándar H.264

Los distintos modos de codificación que comprueba el codificador se organizan en estructura de árbol. Según el tipo de tira que se vaya a codificar, se dispondrá de un conjunto de modos y opciones diferentes.

En las tiras de tipo I, se comprueban el modo *Intra16x16*, es decir, el macrobloque completo, y el modo *Intra4x4*, en el que el macrobloque original se divide en 16 subparticiones. Para cada uno de estos modos de dividir el macrobloque tendremos además varias formas de predicción, como se ha explicado anteriormente. En el Intra, dispondremos de 4 formas de predicción, mientras que en el modo *Intra16x16* se tendrán 9. Para cada macrobloque perteneciente a una tira Intra se probará cada combinación de modo y forma de predicción, siempre que esto sea posible ya que se necesitará disponer de los macrobloques adyacentes, eligiendo la "mejor" de estas combinaciones.

De forma genérica, se puede decir que tendremos A posibles modos, cada uno con P_A tipos de predicción, quedando un total de AxP_A posibles combinaciones a probar para cada macrobloque de una tira Intra.

En el caso de tiras P o B, además de probar todos los modos Intra, estarán disponibles los modos Inter explicados anteriormente, de tamaños 16x16, 16x8 8x16 y 8x8. Además, este último podrá dividirse en bloques 8x4, 4x8 y 4x4.

Además de todos estos modos, se ha de tener en cuenta que para cada partición elegida en los modos Inter se ha de seleccionar la mejor predicción de todas las disponibles en el codificador mediante el proceso de estimación y compensación de movimiento, lo cual añade mucha complejidad al problema. Expresándolo de manera genérica, tendremos M modos Inter, cada uno barrerá N imágenes de referencia, y un rango de búsqueda para cada referencia de ±W píxeles. Esto hace que se necesiten comprobar $NxMx(2W+1)^2$ posiciones para obtener la mejor predicción para cada modo Inter.

En el caso de tiras B, tendremos que añadir a esto las diversas direcciones de predicción disponibles, con el uso de predicciones futuras, y la posibilidad de utilizar bipredicción.

Una vez obtenida la mejor predicción para cada modo en el caso de las tiras Inter y la mejor forma de predicción en el caso de tiras Intra, el codificador calculará la distorsión para cada modo, y realizará la codificación para obtener la tasa. Con estos dos valores, obtendrá el coste de cada modo, y elegirá el de menor coste.

Como vemos, el problema es complejo, e implica una gran cantidad operaciones. El objetivo del proyecto será el de disminuir el número de operaciones que lleva a cabo el codificador para obtener el mejor modo, intentando que esto afecte lo mínimo posible a la calidad (distorsión) y a la tasa. Para ello, se intentará decidir de manera prematura el mejor modo, evitando el cálculo de todas las posibilidades mencionadas anteriormente.

A continuación mostramos algunos de los algoritmos presentes en la literatura para la resolución del problema al que nos enfrentamos.

4.2 Estado del arte

En este apartado realizaremos un estudio del estado del arte sobre algoritmos propuestos para la decisión rápida de modo. Para ello, se han elegido aquellos que utilizan configuraciones del software de referencia similares a la nuestra, con el fin de poder realizar comparaciones con los resultados finales.

En primer lugar, se analizarán los artículos **[1]** y **[2]** de manera muy detallada, ya que exponen métodos aceptados por el JVT (*Joint video Team*), y que han sido integrados en el software de referencia JM **[11]**, debido a sus buenos resultados. Los resultados obtenidos en estos trabajos nos servirán como referencia a la hora de evaluar la solución aportada en este proyecto. A continuación se verán varios métodos de decisión rápida de modo, y por último un resumen analizando los distintos enfoques empleados en dichos métodos.

4.2.1 Algoritmos estudiados

En *Fast mode decision for h.264* **[1]** se realiza la decisión prematura del modo *Skip*. Para ello, se calcula el vector de movimiento y el cuadro de referencia para el bloque 16x16 dado y a continuación se calcula el coste del modo *Inter16x16*. Teniendo estos datos se comprueba que se cumplan una serie de condiciones:

i) el mejor tamaño de bloque para la compensación de movimiento es 16x16
ii) El cuadro de referencia es el anterior
iii) El vector de movimiento es el (0,0) o el mismo que el PMV (vector de movimiento predicho)
iv) Los coeficientes de la transformada cuantificados son todos cero.

Si se dan todas estas condiciones, el macrobloque es codificado como *Skip*, y el resto de modos no son evaluados. En caso negativo, la segunda parte del algoritmo

decidirá si evaluar los modos Intra o no. Para ello, se continúa con el cálculo del resto de costes para todos los modos Inter. Una vez hallado el mejor modo Inter se calcula el *average rate* (AR), que es el numero medio de bits utilizados para codificar el residuo ya compensado. Este valor es un indicativo del grado de correlación temporal. Además, se calcula el *average boundary error* (ABE) que es el error medio del borde entre los pixeles del macrobloque actual y el adyacente para el mejor de los modos Inter. Este valor indica el grado de correlación espacial. Con estos dos valores, se decide si la correlación espacial es o no más importante que la temporal. Si AR <ABE, la correlación temporal prevalecerá sobre la espacial, por lo que el modo elegido será el mejor de los Inter. En caso contrario, se evaluarán los modos Intra, teniendo por tanto los costes de tasa-distorsión para todos los modos posibles, decidiendo el mejor modo de la forma tradicional.

En *Fast Mode Decision for B slice* **[2]**, que complementa al artículo anterior, se realiza la decisión prematura del modo *Directo* en la tiras B. Para ello se calcula el coste del modo *Directo*, y si el cbp (*coded block pattern*) es igual a cero se elije como mejor modo el *Directo*. En caso negativo, se calculan el resto de modos Inter, y se procede como en el artículo anterior para la decisión de modos Intra.

Ya que el primer algoritmo se centra en la decisión prematura del modo *Skip* en tiras P y en el cálculo de modos Intra, y el segundo se dedica a la decisión prematura del modo *Directo* en tiras B, ambos métodos son compatibles, y por ello son presentados los resultados de manera conjunta en **[2]**.

Al integrar los dos algoritmos, el diagrama de flujo en el codificador en la decisión de modo es como sigue:

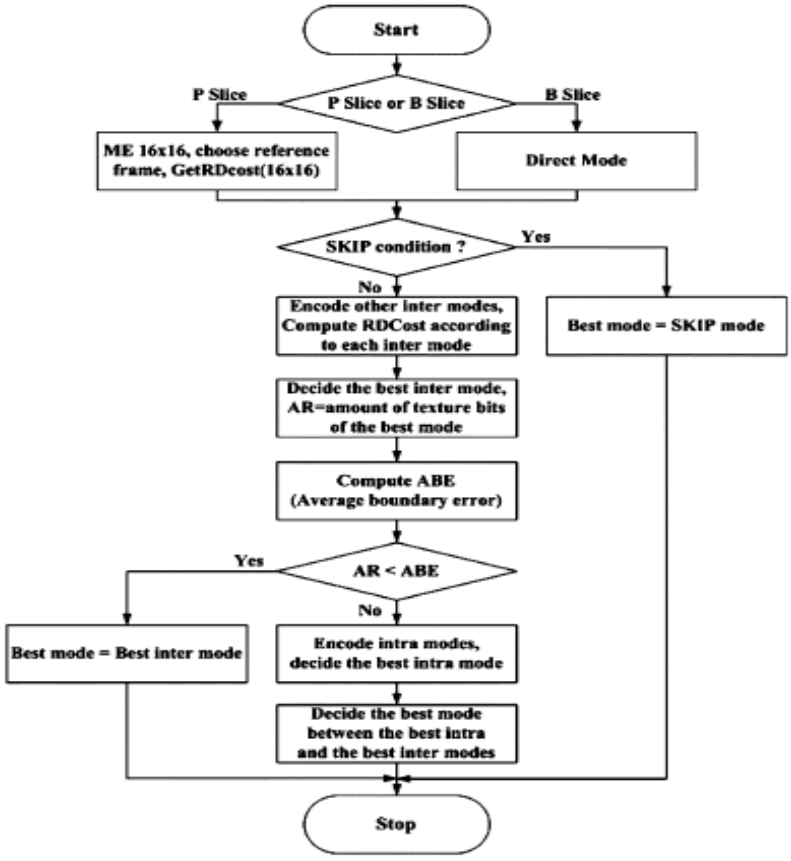


Figura 4.4 Diagrama de flujo de [2]

En *Fast Intermode Decision in H.264/AVC Video Coding* **[3]** se realiza un algoritmo de decisión de modo rápida, basado en medidas que determinan si las regiones analizadas son homogéneas y estacionarias. Una vez tomadas estas medidas, se compararán con unos umbrales, que determinarán el modo a codificar. En este algoritmo no se toma ningún tipo de decisión sobre los modos Intra, por lo que deberán ser evaluados en todos los casos.

En *Fast Mode Prediction for the Baseline and Main Profiles in the H.264 Video Coding Standard* **[4]** también se realizará la decisión prematura de modos. En primer lugar utilizará la detección del modo *Skip* descrita en **[2]**. A continuación, si no se ha realizado parada prematura, los modos *Inter16x16*, *Inter8x8* e *Inter4x4* serán analizados de manera secuencial, comparándolos con un umbral, que decidirá si el modo evaluado es el mejor modo Inter, o se ha de continuar con el proceso. Seguidamente, si no ha decidido aún el mejor modo, partiendo de costes calculados previamente y de la premisa de que existe una relación de monotonicidad entre los costes grandes, el coste *Inter8x8* y los costes de las subparticiones, dividirá los modos a analizar en tres grupos, eligiendo sólo uno de ellos. Finalmente, una vez elegido el mejor modo Inter, se calculará el coste para los modos Intra, y se elegirá el mínimo.

Fast Mode Decisión and Motion Estimation for JVT/H.264 **[5]** plantea algoritmos para una estimación de movimiento rápida, una selección de referencia de entre un conjunto reducido, y una decisión de modo rápido. Para esto último en primer lugar realizará una detección prematura del modo *Skip* mediante la utilización de un umbral basado en la tasa mínima necesaria para codificar un modo Inter. A continuación calculará los costes *Inter16x16*, *Inter8x8* e *Inter4x4* y haciendo uso de la relación de monotonicidad entre los costes de los modos grandes, el coste del modo *Inter8x8* y los costes de las subparticiones, dividirá los modos a analizar en tres grupos, eligiendo sólo uno de ellos. Los modos Intra serán evaluados sólo en el caso de que la energía del residuo del mejor modo Inter sea mayor que cero.

En *Fast Motion Estimation and Mode Decisión With Variable Motion Block Sizes* **[6]** se hará uso de los costes *Inter16x16* e *Inter8x8* para decidir si se evaluarán sólo modos grandes o modos pequeños para elegir el mejor modo Inter. Una vez elegido el mejor modo Inter, se hará uso de un umbral basado en costes para determinar si evaluar también los modos Intra. Finalmente, se elegirá como el óptimo de entre todos los calculados.

En *Fast Multi-block Selection For H.264 Video Coding* **[7]** se va a hacer uso de la información que proporcionan los vectores de movimiento del modo *Inter8x8* para decidir los modos Inter a evaluar. La idea general del algoritmo es que estos vectores nos van a indicar la cantidad de movimiento presente en el macrobloque y, por tanto, los modos más probables a ser seleccionados.

En *Efficient Mode Selection for H.264 Complexity Reduction in a Bayesian Framework* **[8]** se realiza la decisión de modo *Skip* de manera prematura haciendo uso de la diferencia de costes entre el modo *Skip* y el modo ganador del mismo macrobloque en el plano anterior (cosituado). Para ello se modela la densidad de probabilidad del valor de la resta del coste del modo *Skip* y del modo ganador en el macrobloque cosituado, cuando el modo ganador es el *Skip*, y cuando el modo ganador es otro diferente. De esta forma, establece un umbral que dependerá de un factor de actividad

(AF), descrito como la diferencia entre el plano actual y el temporalmente anterior, y del valor de QP utilizado.

En *Scalable Fast Rate-Distortion Optimization for H.264/AVC* **[9]** se evaluarán los modos de manera secuencial, realizando parada prematura en caso de que el coste calculado sea menor que un umbral. Este umbral se obtendrá a partir de las medias y desviaciones típicas de los costes ganadores pertenecientes a los macrobloques previamente codificados.

4.2.2 Resumen

Tras este análisis de diversos métodos propuestos para la decisión de modo vamos a hacer un pequeño resumen de lo visto hasta ahora apuntando similitudes y diferencias entre los algoritmos descritos.

En primer lugar se debe destacar que la mayoría de algoritmos buscarán una decisión prematura de los modos *Skip* y *Directo*. Esto se debe a que son los más sencillos de evaluar por el codificador, y la probabilidad de ser utilizados como modos óptimos en la codificación es muy alta, por lo que, si se detectan prematuramente, se conseguirá evitar el cálculo del resto de modos. Los métodos descritos en **[1]**, **[2]** y **[8]** se centrarán en esta decisión, aunque de manera diferente. Mientras que en los dos primeros se utilizarán el número de coeficientes nulos del residuo, en el tercero se llevará a cabo un modelado de la función densidad de probabilidad de los costes.

Varios algoritmos extienden la decisión prematura al resto de modos, como en **[4]** y **[5]**, que dividirán los modos candidatos en grupos: grandes, pequeños e *Inter8x8*. Teniendo en cuenta la monotonicidad del coste RD para estos tres grupos, y mediante el uso de un coste representativo para cada uno, se decidirá cuál de ellos se evaluará, evitando el cálculo del resto de modos. De manera similar, en **[6]** se dividirán los modos en dos grupos, grandes y pequeños, haciendo uso de costes representativos para elegir el conjunto de modos a evaluar. En **[9]**, la evaluación se realizará de forma secuencial, de manera que se podrá parar en el momento en que se decida el modo óptimo. En cada etapa se calculará un coste, que será comparado con un umbral. Éste será obtenido a partir de la media y de la desviación típica de los modos ganadores pertenecientes a los macrobloques previamente codificados.

El uso de los vectores de movimiento también es una solución común en los algoritmos de decisión de modo, como en **[7]**, donde se utilizan los vectores de movimiento de las particiones 8x8 para decidir de manera prematura el modo *Inter16x16*. Otras características también darán información sobre la estacionariedad del macrobloque, como la definida en **[3]**, que junto con una medida de homogeneidad, servirá para determinar los modos a evaluar.

4.3 Conclusión del estudio del estado del arte

Hemos presentado y estudiado varios de los artículos referentes a decisión de modo publicados hasta hoy día en la comunidad científica, intentando obtener conclusiones específicas del método de trabajo utilizado.

Ahora se intentara encontrar una solución original y que ofrezca buenos resultados al problema que nos compete, buscando mejorar de algún modo lo hasta ahora publicado.

5. Solución propuesta

5.1 Introducción

Nos disponemos a realizar un diseño basado en un clasificador lineal que mejore las prestaciones del algoritmo de decisión de modo utilizado en el software de referencia. El objetivo principal, como ya se ha comentado anteriormente, será reducir el tiempo que emplea el codificador en decidir el modo óptimo de predicción, sin que ello afecte, en la medida de lo posible, a la calidad de la codificación.

5.2 Estructura general del diseño.

Nuestro objetivo será seleccionar el modo a emplear en la codificación de un macrobloque basándonos en decisiones binarias estructuradas jerárquicamente, es decir, mediante un árbol de decisiones. Así, utilizando un conjunto de características del vídeo previamente seleccionadas, se intentará decidir prematuramente si el modo evaluado es el que resulta en un menor coste (según la función de coste ya definida en 4.1), o si, por el contrario, se deberá continuar evaluando el resto de modos.

La ventaja fundamental de proceder de la manera comentada, es decir, tomando decisiones binarias secuencialmente, será que iremos obteniendo las variables necesarias en cada clasificador siempre y cuando lleguemos a ese nivel de decisión. Además, los casos más sencillos y probables ocuparán las primeras etapas de decisión, haciendo que sólo calculemos todas las variables en casos necesarios y complejos.

5.2.1 Clasificador basado en perceptrones monocapa

En este apartado comentaremos brevemente la filosofía y matemática implícita en el diseño de un clasificador basado en perceptrones monocapa.

El perceptrón es un sistema de aprendizaje basado en ejemplos capaz de realizar tareas de clasificación. Cada variable de entrada p_i es ponderada por el peso correspondiente $w_{1,i}$. La suma de todas ellas unida al umbral b será la salida del perceptrón. Por lo tanto, dicha salida de la red viene dada por la siguiente expresión:

$$a = \sum_{i=1}^{R} w_i \cdot p_i + b$$

Ecuación 5.1 Función de salida del perceptrón

A la salida se colocara un decisor duro que decidirá si la hipótesis correcta es una u otra es decir, si efectuamos parada prematura o no.

Resumiendo gráficamente:

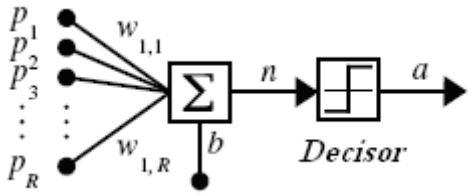


Figura 5.1 Arquitectura del perceptrón monocapa con decisor

Como apunte debemos recordar que el perceptrón únicamente es capaz de clasificar correctamente conjuntos de datos que sean linealmente separables y en caso contrario la probabilidad de error no podrá ser nula. Por otra parte comentar que necesitaremos diseñar y entrenar tantos perceptrones distintos como decisiones queramos tomar.

5.3 Selección de variables

A continuación se describe la etapa de selección de variables (parámetros), crucial para conseguir un buen funcionamiento del diseño. Dicha selección se realizará teniendo en cuenta la información que nos aporta una determinada variable sobre la decisión final a tomar y el coste computacional que supone calcular dicha variable en el codificador. Evidentemente, a más información y menor coste computacional mejor será el parámetro en cuestión para la resolución del problema. Se deduce que estos parámetros deberán describir de alguna forma características de la zona de la imagen a codificar en cuanto a homogeneidad, estacionariedad y movimiento. Para evaluar la información que una (o varias) de las variables a probar nos aporta sobre el problema (y la redundancia entre ellas) recurriremos al clasificador. Las variables serán introducidas en el clasificador, y se comprobarán los resultados que se obtienen con un conjunto de entrenamiento. En base a esto, serán seleccionadas o desechadas.

Se clasificarán en función del tipo de información que a priori aportan sobre el vídeo.

5.3.1 Descripción de las variables de prueba

Información acerca de la complejidad general del Macrobloque (Estimación tasa-distorsión)

Coste Inter16x16 en Referencia 0: Consiste en el calculo del coste *Inter16x16* realizando la estimación y compensación de movimiento solamente sobre la primera referencia presente en la lista 0. Será una buena estimación para ver qué tipo de macrobloque nos disponemos a codificar. Además, tiene en cuenta la QP de trabajo, y es el coste RD. Exploraremos únicamente dicho tamaño en dicha referencia ya que, ahorrando el número de estimaciones de movimiento, se conseguirá reducir el tiempo de codificación.

Coste Inter8x8 en Referencia 0: Consiste en el calculo del coste *Inter8x8* realizando la estimación y compensación de movimiento solamente sobre la primera referencia presente en la lista 0. Aportará información sobre el coste aproximado que supondría codificar un determinado macrobloque con modos pequeños. Además tiene la en cuenta la QP de trabajo, y como en el caso anterior, al utilizar la referencia 0, se ahorrará la búsqueda de la mejor referencia, haciendo más rápido el proceso.

Coste Directo/Skip: El coste *Directo*/*Skip* es el coste que el codificador estima que supondrá codificar con modo *Directo*/*Skip*. Su cálculo implicará bajo "esfuerzo" computacional en el codificador, ya que se utiliza el vector de movimiento predicho (MVp).

Coste Inter16x16: El coste *Inter16x16* nos dará información sobre el coste de codificar con modos grandes (*Inter16x16*, *Inter16x8* y *Inter8x16*). Proporcionará mas información que el coste *Inter16x16* en referencia cero, por el hecho de que en este caso se barrerán todas las referencias disponibles por lo que el coste computacional para obtenerlo será mayor

Información acerca de la correlación temporal. Estacionariedad.

Resta de píxeles: Será el valor absoluto de la resta entre los píxeles del Macrobloque a codificar y su cosituado en el plano anterior. Dará información de correlación temporal entre un plano y el anterior en ese momento y en esa zona de la imagen, o dicho de otra forma, de si el macrobloque a codificar es estacionario en tiempo o no lo es.

Resta de píxeles 2: Resta de píxeles del macrobloque a codificar y el cosituado en la referencia. Tendrá información de la calidad del vídeo en ese momento, ya que la resta será sobre la referencia (ya codificada y reconstruida).

Diferencia de histogramas: Será la diferencia de histogramas entre el macrobloque actual y el cosituado. Da información de correlación temporal entre macrobloques. Es similar a la resta de píxeles, pero independiente del movimiento. Es decir, si existe un objeto que se mueve entre un plano y otro, pero sigue perteneciendo al mismo macrobloque, la diferencia de histogramas tenderá a 0, y no así la resta.

Modo cosituado: Tamaño de bloque con que fue codificado el macrobloque cosituado. Presentará bastante correlación con el modo óptimo para la codificación del macrobloque en el que nos encontramos en ese momento. Como inconveniente comentar que depende de las decisiones que se hayan tomado anteriormente acerca del tamaño de bloque.

Información acerca de la correlación espacial. Homogeneidad local y global.

-Homogeneidad local (INTRAMacrobloque)

Desviación típica de los píxeles de un macrobloque: Será la desviación típica calculada a lo largo de un Macrobloque. Dará información de dispersión en luminancia dentro del Macrobloque, es decir, de la homogeneidad espacial existente en dicho macrobloque.

Se calcula como:

$$\sqrt{\sigma^2} = \sqrt{\frac{\sum_{i=1}^{N} (X_i - \mu)^2}{N}}$$

Ecuación 5.2 Desviación típica

Gradientes (Roberts y Sobel) verticales y horizontales: Darán información acerca de la cantidad de bordes (y la dirección de estos) presentes en un determinado Macrobloque. Por lo tanto, se puede interpretar como una medida de homogeneidad, de cuánta energía de alta frecuencia existe en una zona de la imagen. Utilizamos para su cálculo Gradientes de Roberts y de Sobel. Resumiendo brevemente, dadas dos coordenadas del vector gradiente devueltas por una de las máscaras propuestas (la de Roberts o la de Sobel):

$$\nabla f(x,y) = \begin{bmatrix} G_x & G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} f(x,y) & \frac{\partial}{\partial y} f(x,y) \end{bmatrix}$$

Ecuación 5.3 Vector gradiente

Obtendremos la dirección de dicho vector como:

$$\phi(\nabla f) = \arctan \frac{G_y}{G_x}$$

Ecuación 5.4 Fase del vector gradiente

Y el módulo como:

$$|\nabla f| = \sqrt{G_x^2 + G_y^2}$$

Ecuación 5.5 Módulo del vector gradiente

En la práctica, debido al coste computacional, el cálculo del módulo del gradiente se aproxima mediante:

$$|G| \approx |G_x| + |G_y|$$

Ecuación 5.6 Aproximación del módulo del gradiente

Finalmente denotar que, en realidad, el valor de la magnitud del gradiente no es tan importante como la relación entre diferentes valores, es decir, se va a decidir si un punto es borde si la magnitud del gradiente supera un determinado umbral, y viceversa. Para el cálculo del gradiente se utilizaron 9 puntos por Macrobloque, haciendo que la precisión de la medida no sea todo lo buena que cabría esperar, pero consiguiendo de este modo un cálculo eficiente desde el punto de vista computacional.

Media de luminancia: Da información de la cantidad de brillo de un determinado macrobloque.

Homogeneidad global (INTERMacrobloque)

Modos vecinos: Tamaño de bloque con que fueron codificados los vecinos. Presentará bastante correlación con el modo óptimo para la codificación del macrobloque en el que nos encontramos en ese momento. Depende de decisiones que hayamos tomado anteriormente acerca del tamaño de bloque.

Información acerca del movimiento

Vectores de movimiento predicho: Da una idea del movimiento existente entorno al bloque a codificar.

5.3.2 Normalización de las variables de entrada

Será de vital importancia normalizar las variables de entrada al clasificador. No sólo por facilitar el entrenamiento del clasificador, sino por incluir dinamismo en el proceso. Vamos a explicar esto con más detenimiento y con un pequeño ejemplo ilustrativo:

Supongamos que tenemos una variable de entrada al clasificador, por ejemplo el coste del modo *Directo* (CMD). Además, pongámonos en la situación de que estamos tomando datos para entrenar nuestra máquina de dos secuencias de vídeo totalmente distintas, esto es: secuencia A: Football, QP=32; secuencia B: Mobile, QP=40. Vamos a analizar tres formas distintas de proceder:

-*No normalizamos los datos*. Si no normalizamos los datos puede ser que un valor de coste (CMD) determinado sea considerado bajo para la secuencia A y ese mismo valor sea alto para la secuencia B. Por lo tanto, la máquina no será capaz de generalizar, ya que el mismo valor de CMD implicaría distintas decisiones en función del ejemplo.

-*Normalizamos respecto a cada caso concreto:* Si procedemos de esta forma, normalizando los valores en función de cada secuencia concreta, tendremos una medida de donde se encuentra el CMD obtenido en un determinado instante respecto a una referencia local (por ejemplo la SAD del modo *Directo*). De esta forma tendremos medidas relativas que no dependerán fuertemente del vídeo ni de la QP de trabajo, logrando que la máquina pueda generalizar.

5.4. Entrenamiento

El entrenamiento es la fase mediante la cuál la máquina ajusta los pesos w y la constante b (ver figura 5.2) haciendo uso de ejemplos etiquetados de forma que se minimice el valor esperado del error de nuestra función de coste, y sea capaz de clasificar muestras futuras que no haya "visto" anteriormente.

Durante este proceso se utilizará un perceptrón con una función hiperbólica (f en la figura 5.2) a la salida, que proporcionará una salida continua, a diferencia del decisor duro empleado en el clasificador, quedando como se muestra en la siguiente figura:

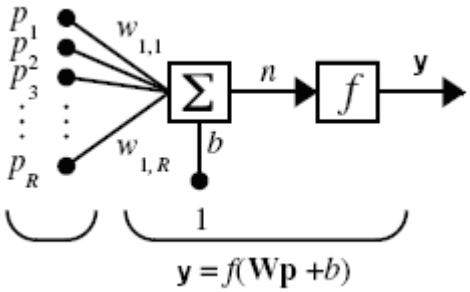


Figura 5.2 Perceptrón con función hiperbólica

El proceso de entrenamiento se ha llevado a cabo con la herramienta de redes neuronales proporcionada por Matlab (Neural Network Toolbox). A continuación se explica la matemática que conlleva el proceso de entrenamiento

5.4.1. Conjunto de entrenamiento

Para entrenar el clasificador se utilizaron datos extraídos de 16 vídeos de muy distintas características, para 2 patrones distintos (IP IBBP) y 4 valores de QP (28 32 36 y 40). Los vídeos se describirán en el apartado de experimentos y resultados.

El hecho de tener un gran número de muestras de entrenamiento de secuencias muy variadas hará que el problema del sobreentrenamiento sea prácticamente inexistente, por lo que no se ajustará a un caso concreto, siendo capaz de generalizar bien. Además, se cogerán muestras de manera aleatoria de cada y se proporcionará el mismo de número de datos para cada secuencia.

5.4.3 Función de coste

El objetivo del diseño de una función de coste específica será penalizar aquellos casos en los cuales un error de clasificación sea crítico en el aumento de tasa en el codificador. Esto es, se tratará de favorecer la correcta clasificación de casos que impliquen costes críticos minimizando la función:

$$C(w) = (t-y)^2 \left(1+\alpha\left(C - C_{\min}\right)\right)$$

Ecuación 5.7 Función de coste para el entrenamiento

Donde "t" es la etiqueta que asignada a dicha muestra, "y" la salida del perceptrón junto con la función hiperbólica, α un factor de ponderación que será determinado de manera experimental, C el coste del modo elegido por el clasificador, y C_{min} el coste del modo óptimo seleccionado por el software de referencia.

Vamos a analizar los distintos tipos de errores, y lo que suponen en el proceso de codificación. Siguiendo la nomenclatura habitual en aplicaciones de radar, tendremos:

Decisión / Hipótesis	No existe blanco	Existe blanco
H_0 no existe blanco	Correcto	Error tipo I
H_1 existe blanco	Error tipo II	Correcto

Decisión / Hipótesis	No existe blanco	Existe blanco
H_0 no existe blanco	$1-P_{FA}$	Probabilidad de falsa alarma (P_{FA})
H_1 existe blanco	$1-P_D$	Probabilidad de detección (P_D)

Tabla 5.1 Tipos de errores

Para analizar los tipos de errores, consideraremos el caso de decisión prematura de modo *Directo*:

A) *Error tipo I. Probabilidad de Falsa alarma*: Se estará decidiendo el modo *Directo* como óptimo cuando no lo es. En este caso el coste de esta decisión errónea será: $(t-y)^2 \cdot (1+\alpha(C_{cir}-C_{\min}))$. Hay que destacar que este será el peor de los casos, ya que conllevará un error en la decisión final, haciendo que se codifique con un modo cuyo coste es superior al óptimo. Es por ello que será el error que se pondrá más énfasis en evitar.

B) *Error tipo II. Probabilidad de no detección*. Se estará desechando la decisión prematura de modo *Directo*, por lo que se continuará con el proceso. Finalmente, elegirá *Directo* como mejor coste entre todos los evaluados, pero en una etapa posterior. Esto no supondrá un coste en cuanto a tasa o calidad, ya que finalmente elegirá el *Directo* como mejor modo, pero si en cuanto a tiempo ya que para llegar a esta conclusión deberá evaluar todos los modos, en vez de realizar parada prematura.

La función de coste propuesta intentará minimizar la probabilidad de falsa alarma, que será el caso que suponga una codificación incorrecta. Hay que tener en cuenta que al entrenar el perceptrón con una función de coste distinta al error cuadrático medio, se conseguirá disminuir la probabilidad de falsa alarma, pero a cambio de un aumento en el error total, es decir, habrá un mayor número de casos de no detección. El factor α será el que determine la importancia que se le atribuye a la falsa alarma. Un valor alto hará que disminuya la falsa alarma, pero a cambio de un aumento considerable en el error total, por lo que muchas muestras no serán detectadas prematuramente, haciendo el algoritmo menos eficiente.

5.4.3.1 Normalización de la función de coste:

Como hemos comentado, la filosofía seguida será la de considerar el coste que supondría clasificar mal un determinado ejemplo. Esta penalización se llevará a cabo en el entrenamiento, de forma que intentará clasificar mejor aquellas muestras más conflictivas en cuanto a incremento de tasa. Para ello, a cada muestra le asignará un coste en función de la clase a la que pertenezca la etiqueta "t".

Supongamos que tenemos un valor de coste de C-Cmin=30. Dicho coste será poco importante proporcionalmente hablando a calidades altas, ya que la tasa de salida a estos valores de QP es grande. Por lo tanto, el incremento proporcional de tasa será bajo. Sin embargo, no ocurrirá lo mismo a calidades bajas, donde la tasa de salida es pequeña, y la proporción será por lo tanto grande. Así mismo existirá mucha variabilidad en función del vídeo que estemos codificando. Para solucionar este problema se normalizará la función de costes. La normalización se hará localmente y respecto a la media de costes, obtenida de datos de entrenamiento de un caso concreto. Esto es, si por ejemplo estamos obteniendo muestras de entrenamiento de la secuencia Football a QP=32, normalizaremos respecto a las datos de Football a QP=32. Matemáticamente hablando, la función de coste quedará de la siguiente manera

$$C(w) = (t-y)^2 \left(1 + \alpha \frac{(C - C\min)}{\mu_{(C-C\min)}} \right)$$

Ecuación 5.8 Función de coste normalizada para el entrenamiento

5.5 Caracterización de los algoritmos

Una vez implementados los algoritmos se deberán evaluar sus prestaciones. Para ello se atenderá fundamentalmente a tres parámetros: tasa, PSNR y tiempo empleado en el proceso de codificación.

Tasa

Es la tasa binaria media resultante de la codificación en $kbits/s$. Nos servirá para evaluar cómo de acertada es nuestra decisión de modo. Debido a que nuestros algoritmos presentarán una solución subóptima, la tasa resultante del proceso de codificación tenderá a aumentar con respecto a la obtenida por el software de referencia. Nuestro objetivo será que se incremente lo mínimo posible. Los resultados se expresarán como el incremento de tasa que supone la utilización del algoritmo analizado:

$$\Delta Tasa_{\text{algoritmo}} = \frac{Tasa_{\text{algoritmo}} - Tasa_{\text{ref}}}{Tasa_{\text{ref}}} \cdot 100$$

Tiempo de codificación

Se trata del tiempo total empleado en el proceso de codificación. Analizaremos este parámetro porque será un reflejo del gasto computacional de cada uno de los algoritmos. Los resultados se expresarán como el porcentaje de tiempo ahorrado respecto de la referencia:

$$\text{Ahorro en tiempo}=\left|\frac{T_{\text{algoritmo}}-T_{\text{ref}}}{T_{\text{ref}}}\right|\cdot 100$$

Hay que tener en cuenta que este parámetro depende fuertemente de la máquina en la que ejecutemos las pruebas. Por este motivo utilizaremos siempre el mismo ordenador e intentaremos que las condiciones de carga del procesador sean similares.

PSNR

Es una medida matemática de la calidad de la codificación. Nos aportará información objetiva sobre el parecido de la secuencia codificada respecto a la secuencia original. Como en el caso de la tasa, el algoritmo implementado estará por debajo en prestaciones que el software de referencia, por lo que nuestro objetivo será que la PSNR disminuya lo mínimo posible. Los resultados se expresarán como la diferencia de la calidad expresada en dB:

$$\Delta\text{PSNR}_{\text{algoritmo}}=\text{PSNR}_{\text{algoritmo}}-\text{PSNR}_{\text{referencia}}$$

Presentación de los resultados. Interpolación

Trabajaremos con tres tipos de gráficas:

-Tasa en función de la PSNR: En este tipo de gráficas representaremos los resultados de tasa y PSNR obtenidos con el codificador de referencia frente al algoritmo desarrollado. Para realizarlo se utilizarán las indicaciones dadas en **[10]**. Cuanto más parecidas sean ambas curvas, mejor será el algoritmo, ya que indicará que la codificación es similar a la original.

-Incremento de tasa en función de la PSNR: Esta gráfica mostrará el incremento de tasa resultante al utilizar el algoritmo respecto a la referencia, para una misma PSNR.

-Decremento en tiempo en función de la PSNR: Representa, para un mismo valor de PSNR, la mejora en tiempo que presenta el algoritmo.

La referencia utilizada será el software de referencia en su versión 10.2 (JM10.2). Obviamente nos interesarán valores elevados en nuestras gráficas de tiempo, y valores reducidos en las de tasa.

Los datos que nos sirven para elaborar las gráficas proceden de las codificaciones realizadas. Dichas codificaciones se realizan para unos determinados valores de QP (28, 32, 36 y 40), lo que se traducirá en una determinada calidad de la

secuencia codificada medida en términos de PSNR. Como no dispondremos de valores de PSNR fijos para cada codificación, deberemos interpolar los datos, con el fin de poder compararlos. Pongamos un ejemplo para ver por qué es necesaria la interpolación en la presentación de nuestros resultados. Supongamos que para un determinado valor de QP, y evaluando dos algoritmos distintos, obtenemos lo siguiente:

Algoritmo 1: PSNR=35dB y tasa: 1600Kbits/seg.

Algoritmo 2: PSNR=34dB y tasa=1300Kbits/seg.

Al intentar comprar las prestaciones de ambos algoritmos, se ve que no es posible, ya que no podremos saber el aumento de tasa para una misma calidad.

Para resolver este problema realizaremos varias codificaciones con cada uno de los algoritmos para obtener una serie de puntos tasa-PSNR. Una vez obtenidos pasaremos a realizar una interpolación para construir una curva tasa-PSNR que describa el funcionamiento del algoritmo. Entonces podremos observar, para una determinada calidad, qué algoritmo es el que presenta un incremento de tasa menor.

5.6 Entorno de trabajo y realización de las pruebas

Para la realización de las pruebas se ha hecho uso de tres archivos:

Lencod.exe: Se trata del ejecutable del codificador. Pasándole como parámetro el fichero de configuración *encoder.cfg* realizará la codificación del vídeo correspondiente.

-*Encoder.cfg:* Es el fichero de configuración que se le pasa al ejecutable como parámetro. En él se define la forma de trabajo del codificador. En el Anexo A se muestra el fichero utilizado.

-*Script en bash:* Se encargará de automatizar el proceso de barrido de parámetros de codificación deseados (en el Anexo B se muestra un ejemplo de dicho *script*).

A continuación se van a comentar los parámetros más importantes para nuestro estudio y los valores que se han utilizado para todas las pruebas.

-*Software de referencia*: Programa que implementa el codificador H.264. Se ha utilizado la versión JMv10.2 **[11]**

-*YUVFormat*: Formato de vídeo utilizado, en nuestro caso se configurará como 4:2:0. Resolución: CIF 352x288 y QCIF 176x144

-*SymbolMode*: Tipo de codificación entrópica utilizada. Para las pruebas realizadas se ha seleccionado CABAC.

-*Frames to be encoded:* Número de planos a codificar. En las pruebas realizadas se utilizan 100 planos.

-*ProfileIDC*: Baseline, Main o Extended. Trabajaremos en el perfil Main.

-*Intra period*: Periodo entre imágenes tipo I. Se ajustará a 30 en el caso de de patrón IP y 10 en el caso de I2B.

-*QPISlice, QPPSlice y QPBSlice*: Parámetros de cuantificación para planos I, P y B. En las pruebas realizadas, los tres serán iguales entre sí.

-*Search range*: Máximo rango de búsqueda para la estimación de movimiento. Nosotros trabajaremos con un valor de 32 píxeles; es decir, una ventana de 65x65 píxeles.

-*Number of references*: Número de imágenes de referencia usadas para la estimación INTER. Nosotros utilizaremos 5.

Todos estos parámetros serán configurados mediante el archivo *encoder.cfg* descrito anteriormente.

Además, las pruebas se realizarán para dos patrones de codificación distintos, IP e I2B. A continuación se muestra el orden de codificación (orden en que aparecen las tiras) y el orden de visualización (subíndice correspondiente a cada tipo de tira) para ambos patrones:

Patrón IP:
$I_0 P_1 P_2 P_3 P_4 ...$

Patrón I2B:
$I_0 P_3 B_1 B_2 P_6 B_4 B_5 P_9 ...$

5.7 Análisis previo de probabilidades a priori

El diseño en cascada de nuestro algoritmo permitirá que cada clasificador decida si el modo evaluado es el óptimo, o se debe pasar a la siguiente etapa de decisión. Con el fin de reducir el tiempo de codificación al máximo, se deberán colocar en las primeras etapas de decisión los modos más probables, y así decidir de manera prematura en el mayor número de ocasiones posible. Para ello, se realiza un estudio de las probabilidades a priori que tendrá cada modo, utilizando los vídeos en resolución CIF Akiyo, Bus, Coastguard, Container, Football, Foreman, Garden, Highway, Mobile, Mother_daughter, News, Paris, Silent, Stefan, Tempete y Waterfall. Los patrones que se han barrido son IPPP, IBBP, e IBBBP. Las QP utilizadas: 28, 32, 36, 40.

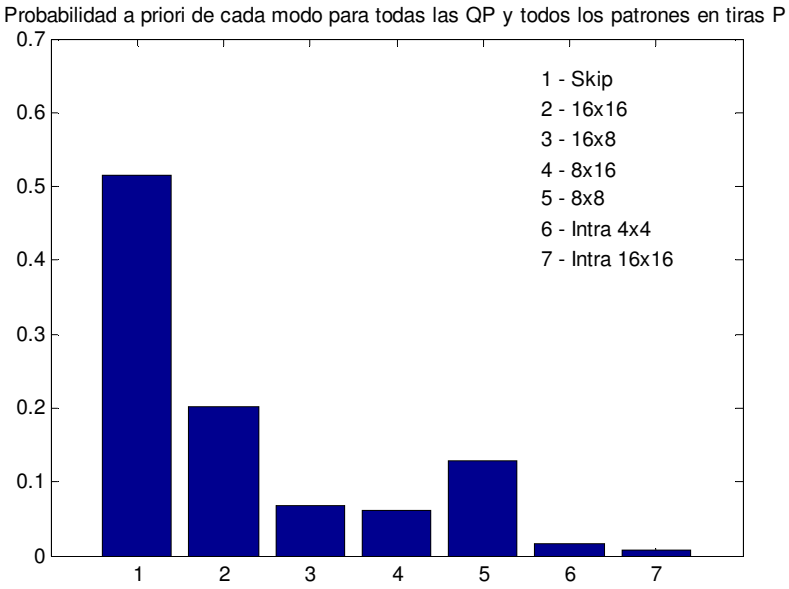


Figura 5.3 Probabilidad a priori de cada modo

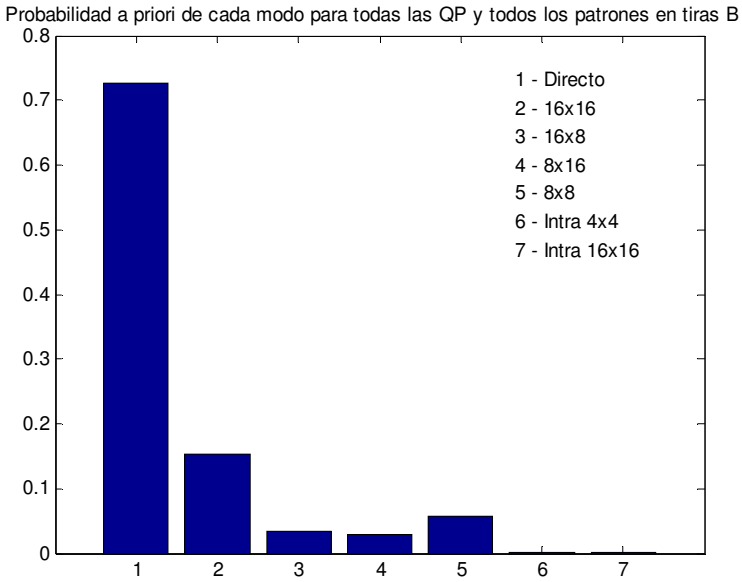


Figura 5.4 Probabilidad a priori de cada modo en tiras B

En las figuras anteriores se puede ver claramente que los modos *Directo* y *Skip* serán los que tengan una probabilidad mayor de ser seleccionados. A continuación, pero con valores mucho más pequeños, tendremos los modos *Inter16x16* e *Inter8x8*. Se puede ver también que los modos Intra rara vez serán elegidos.

Para analizar el efecto del parámetro QP sobre la decisión de modo se ha realizado un análisis en función del valor de QP seleccionado. Los resultados por separado se muestran a continuación

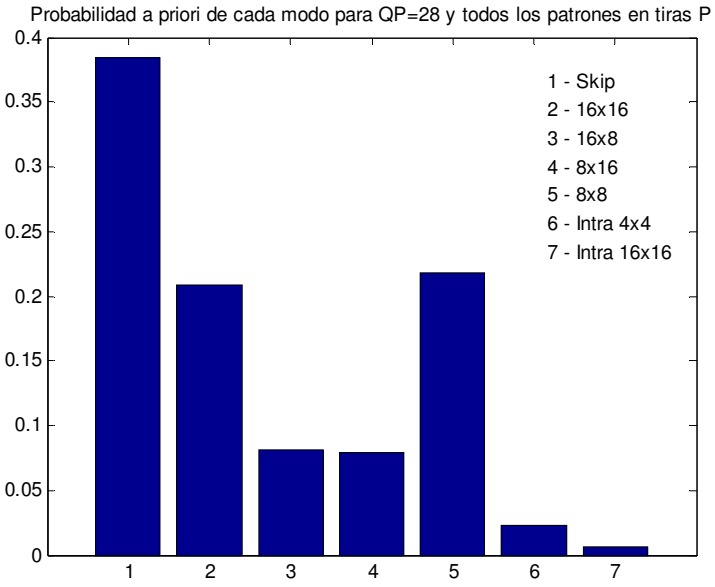


Figura 5.5 Probabilidad a priori de cada modo QP 28 tiras P

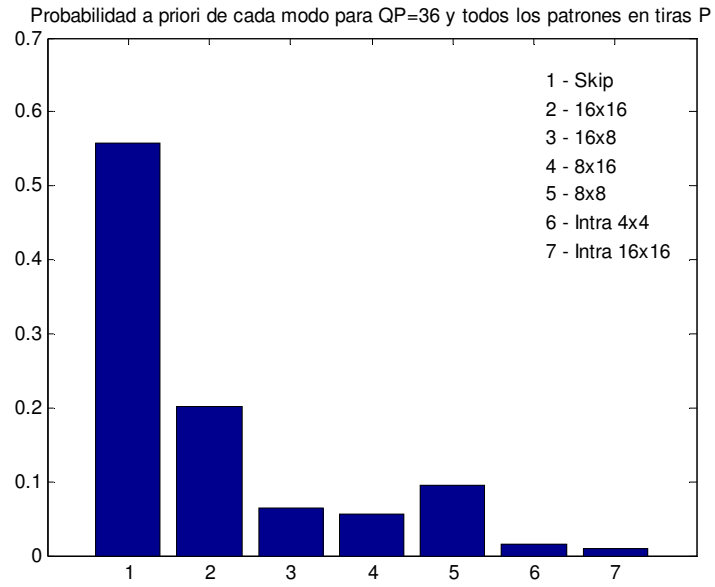


Figura 5.6 Probabilidad a priori de cada modo QP 36 tiras P

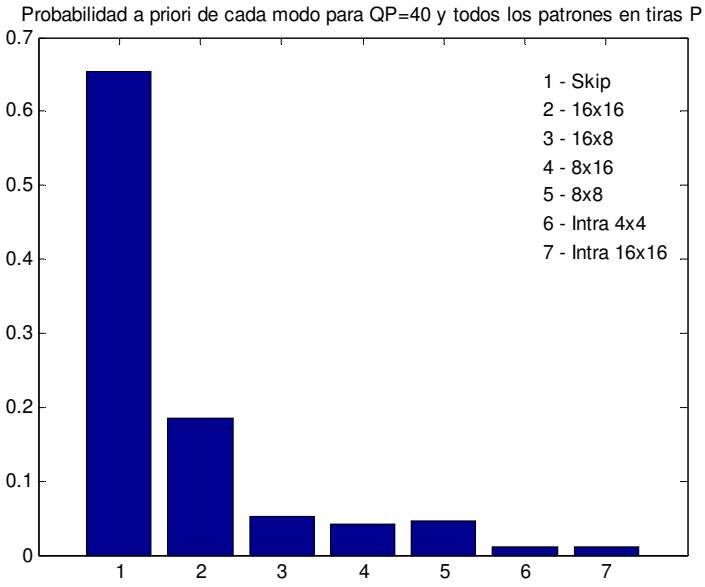


Figura 5.7 Probabilidad a priori de cada modo QP 40 tiras P

Ahora realizaremos el mismo análisis para tiras B

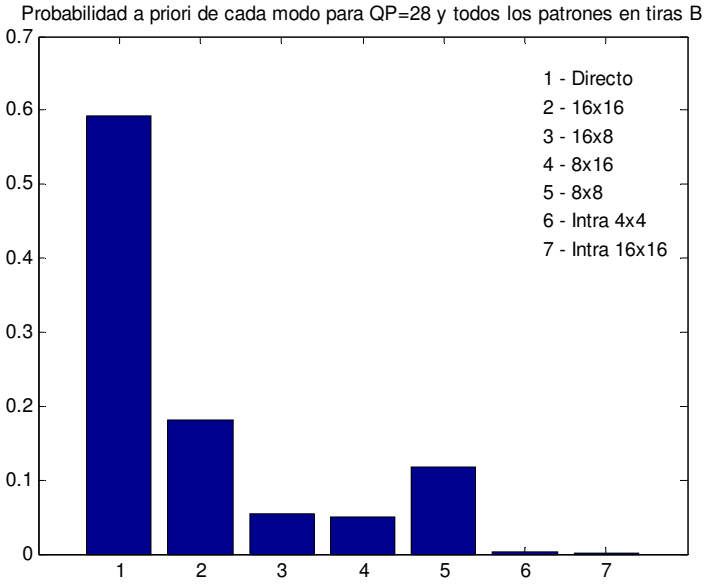


Figura 5.8 Probabilidad a priori de cada modo QP 28 tiras B

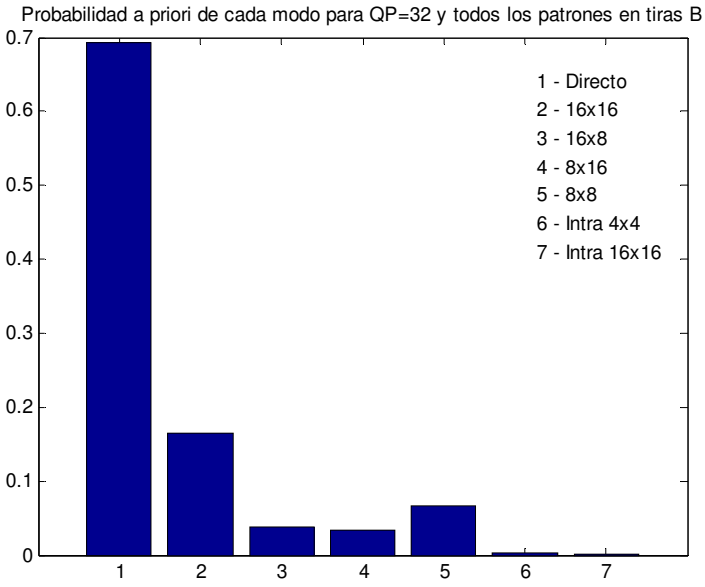


Figura 5.9 Probabilidad a priori de cada modo QP 32 tiras B

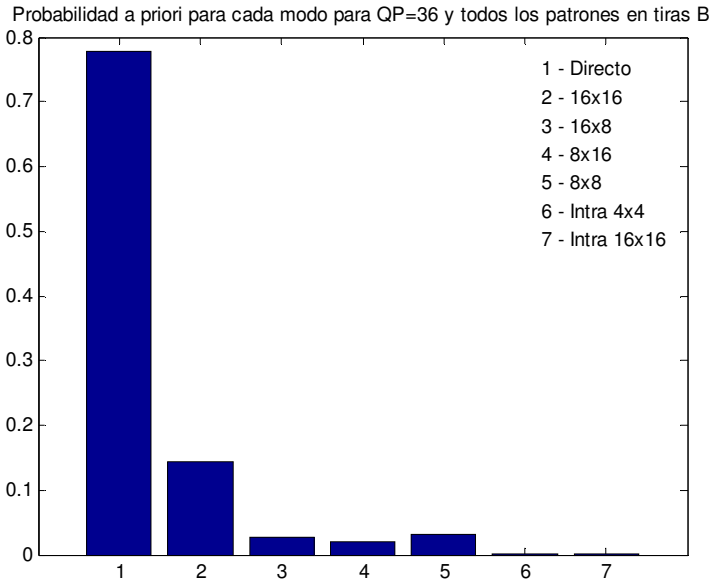


Figura 5.10 Probabilidad a priori de cada modo QP 36 tiras B

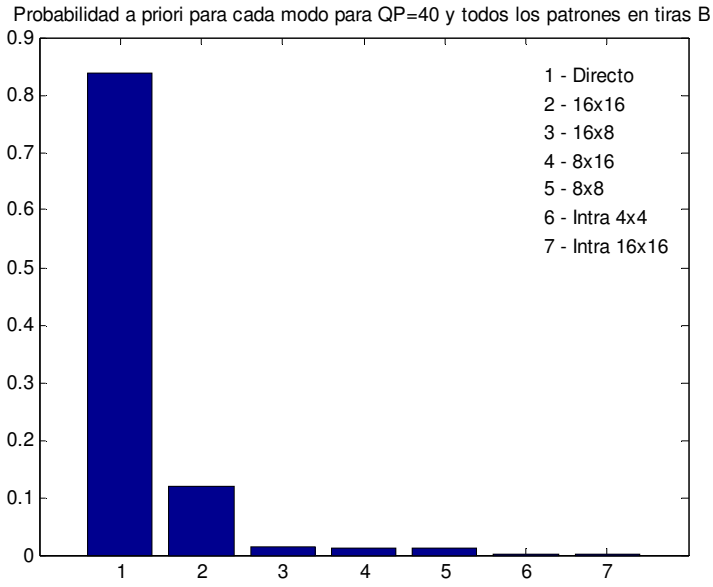


Figura 5.11 Probabilidad a priori de cada modo QP 40 tiras B

Claramente se observa como a medida que aumenta el parámetro QP la probabilidad de elegir el modo *Directo* o el *Skip* aumentará también. Esto se debe al hecho de que al ser el escalón de cuantificación más grande también lo será el valor de λ mas grande dando una mayor importancia a la tasa pasando a ser más importante que la distorsión en la función de coste.

Estos resultados concuerdan con el procedimiento seguido en la mayoría de los artículos analizados en el apartado de estado del arte, que centraban su algoritmo en la decisión prematura de los modo *Directo* y *Skip*. Claramente y a la vista de las gráficas, la primera etapa de nuestro decisor deberá ser también ésta.

6. Experimentos y resultados

6.1 El software de referencia

En este apartado se explicará el funcionamiento del software de referencia JM en su versión 10.2, el cual implementa el estándar H.264. Sobre este software se integrará nuestro algoritmo, y por tanto será importante un estudio detallado del código, para poder modificarlo sin que esto implique en problemas en la codificación. Nos centraremos en la parte del software que se encarga de la decisión de modo, ya que es la que se trata durante este proyecto, comentando en primer lugar y de manera resumida el funcionamiento del codificador, para a continuación detallar los pasos que se llevan a cabo durante el proceso de codificación. No obstante, no entraremos en detalle sobre el código en si ni las operaciones.

En primer lugar, el codificador realizará el cálculo del coste para el modo *Skip* en tiras P, y el modo *Directo* en tiras B. Acto seguido efectuará la estimación de movimiento y la selección de referencia para todos los modos Inter, para a continuación pasar a calcular la distorsión y la tasa de cada modo. Una vez obtenido el coste de codificación para todos los modos Inter, pasará a evaluar los modos Intra, realizando primero el proceso de predicción, y a continuación el del cálculo del coste de codificación mediante la obtención de la distorsión y la tasa. Finalmente, el codificador elegirá como modo óptimo de codificación el que resulte en un menor coste

A continuación mostramos un esquema con los pasos llevados a cabo por el codificador:

Figura 6.1 Esquema de funcionamiento del software de referencia

Pasemos a explicar más en detalle el funcionamiento expuesto en la figura 6.1. En caso de que se esté codificando una tira B, se llevará a cabo el cálculo de los vectores de movimiento para el modo *Directo*. A continuación, para tiras de tipo B y P, se llevará a cabo la estimación y compensación de movimiento de todos los modos Inter. De estos, primero se evaluarán los modos Inter grandes (*Inter16x16 Inter16x8*, e *Inter8x16*) y después los Inter pequeños (*Inter8x8*, *Inter8x4* e *Inter4x8*). Para estos últimos, una vez obtenida la mejor predicción, el codificador pasará a realizar la DCT, la cuantificación y finalmente el cálculo de la reconstrucción (mediante la IDCT y el proceso de reescalado). Con estos datos calculará la distorsión, (sin tener en cuenta la crominancia), y la tasa, eligiendo como mejor modo pequeño el que resulte en un menor coste RD. Hay que tener en cuenta que en el caso de los modos pequeños habrá realizado el proceso completo de codificación por lo que guardará todos los valores obtenidos para el modo ganador, de forma que al calcular posteriormente el coste teniendo en cuenta tanto crominancia como luminancia, no tenga que repetir estas operaciones.

A continuación, si se está codificando un macrobloque de una tira P, se obtendrá el vector de movimiento para el modo *Skip*. Una vez calculado, se evaluarán las distintas formas disponibles de predicción de la crominancia para modos Intra.

En el siguiente paso será en donde se calculará el coste RD para cada modo, mediante un bucle que irá recorriendo los distintos modos disponibles según el tipo de tira que se esté codificando. El orden será el siguiente: *Directo*/*Skip*, *Inter16x16*, *Inter16x8*, *Inter8x16*, Inter pequeño (el mejor de los que se ha calculado anteriormente), *Intra4x4* e *Intra16x16*. Hay que destacar en este punto que cuando evalúe el modo *Inter16x16* en tiras B, calculará el coste para todas las direcciones de predicción, en lista 0, lista 1 o ambas mediante bipredicción.

Como se ha comentado anteriormente, para modos Inter grandes calculará la DCT, la predicción (con las referencias ya calculadas) la reconstrucción, y la predicción de la crominancia, y con todos estos datos, obtendrá la distorsión y la tasa, teniendo el coste RD.

Para los modos Inter pequeños ya dispondrá de los coeficientes del mejor modo pequeño, por lo que simplemente los copiará, obtendrá la predicción de la crominancia y calculará la distorsión y la tasa, esta vez teniendo en cuenta la crominancia.

Cómo se ha comentado anteriormente, dependiendo del tipo de tira que se este codificando, se evaluarán distintos modos. En el caso de tiras de tipo I el codificador no tendrá disponibles los modos Inter, por lo que pasará directamente a evaluar los modos Intra. En el modo *Intra4x4* decidirá que modo de predicción Intra recorriendo el macrobloque en subbloques 4x4, realizando para cada uno de ellos los distintos modos de predicción Intra (9 modos diferentes), siempre que se disponga de los vecinos necesarios para tal fin. Una vez obtenida la predicción, calculará el residuo, la DCT, la cuantificación, y la reconstrucción y se quedará con la forma de predicción que resulte en un coste RD menor.

Para el modo *Intra16x16* obtendrá la predicción de la lumimancia para todos los modos (4 diferentes) siempre que disponga de los bloques vecinos necesarios. Esta función se quedará con la predicción que suponga una menor SAD entre el bloque

original y la predicción. A continuación, para la predicción elegida, calculará los coeficientes, realizará la cuantificación y obtendrá la reconstrucción. Con esto, calculará el coste RD.

Dentro de este bucle, al obtener el coste RD de cada modo, se irá quedando con el mínimo. Finalmente, los datos del mejor modo serán guardados.

Se puede ver que el proceso es diferente según el modo de que se trate. Mientras que el mejor de los modos grandes se decidirá teniendo en cuenta tanto crominancia como luminancia, para los modos pequeños sólo se tendrá en cuenta la luminancia. Algo parecido ocurre con los modos Intra; mientras que en el *Intra4x4* calcula el coste RD para todos los modos de predicción, en el *Intra16x16* sólo realizará la SAD entre el macrobloque original y la reconstrucción para decidir el mejor modo de predicción, siendo éste un proceso más sencillo. Además, hay que destacar que el cálculo de la DCT se realizará siempre en bloques de tamaño 4x4.

Así es, a grandes rasgos, el proceso que sigue el codificador para evaluar los modos y decidir con cuál codificar. Dicho proceso será de vital importancia en nuestro trabajo ya que para lograr una correcta implementación del nuevo algoritmo debemos saber cómo funciona el software de referencia, los pasos que sigue y en qué punto de la codificación nos encontramos en cada línea de código.

6.2 Diseño general

En este apartado se explicará el proceso realizado en el diseño del decisor, que se encuentra dividido en tres partes. En una primera fase, se realizará el entrenamiento del perceptrón utilizando las características extraídas del conjunto de vídeos *entrenamiento*, descrito más adelante. Para este proceso se han utilizado el mismo número de muestras de cada secuencia, escogidas manera aleatoria. El número de vídeos utilizados y su diversidad intentará garantizar una generalización en el decisor implementado.

Antes de evaluar todas las características, se deberá tener en cuenta el coste que puede suponer su cálculo. Es evidente que la característica coste *Inter16x16* aporta más información sobre la decisión de parada prematura en modo *Directo* que el coste *Inter16x16* en referencia cero, pero éste último supondrá una carga computacional mucho menor que el anterior. Es por ello que se buscará un compromiso entre información aportada y coste computacional. Para ello, durante el desarrollo del proyecto se tendrá en mente el tiempo utilizado por el codificador para la evaluación de los distintos modos, que se muestra en el Anexo D. Finalmente, el factor determinante para incluir la característica será su funcionamiento en el clasificador, es decir, se buscarán aquellas que produzcan una probabilidad de falsa alarma menor, manteniendo el error total lo más bajo posible.

Una vez obtenidas las características, se deberán calcular los valores de los pesos y bias del perceptrón. Para ello se realizará un barrido del parámetro α de la función de coste en el entrenamiento del perceptrón, obteniendo así un conjunto de pares probabilidad de falsa alarma, probabilidad de no detección. Como se ha comentado antes, la utilización de la función de coste reducirá la probabilidad de falsa alarma, pero

a costa de un incremento mayor en el error total que el obtenido en un entrenamiento con el error cuadrático medio, ya que no se minimizará éste, si no lo función de coste proporcionada. Es por ello que habrá que seleccionar un α que resulte en un punto de trabajo con una probabilidad de falsa alarma baja, sin que eso suponga un aumento demasiado alto de la no detección.

A continuación, se llevará a cabo la fase de validación de parámetros, en la cual, además de comprobar que el perceptrón diseñado en la etapa anterior funciona para un grupo distinto de secuencias de vídeo de las usadas en la etapa de entrenamiento, se establecerán unas restricciones en cuanto al incremento de tasa y distorsión respecto a la referencia, que se intentarán no sobrepasar. Una vez establecidas, mediante el uso de un umbral (μ) a la salida del decisor, se podrá ajustar el funcionamiento del algoritmo, con el fin de obtener el mayor ahorro computacional sujeto a las restricciones anteriores. El umbral, permitirá que más o menos muestras sean clasificadas como *Skip*/*Directo*, haciendo que varíe el ahorro en tiempo, y por tanto el incremento de tasa y distorsión. Para esta etapa se utilizará el grupo de secuencias *validación* descrito más adelante. Hay que destacar que la validación será distinta para el patrón IP que para el patrón I2B, ya que en el patrón I2B se deberá ser más conservador en las tiras P. Al no ser codificadas en el orden de visualización, las tiras P se encontrarán más alejadas de la tira anterior, habiendo una menor correlación entre la imagen y la referencia que se utilizará, aumentando la probabilidad de una mala elección. Esto afectará por tanto a las tiras B que le siguen a continuación (recordemos que el orden de codificación en el patrón IP es $I_0P_1P_2P_3P_4$, mientras que el Patrón I2B es $I_0P_3B_1B_2P_6B_4B_5P_9$) produciendo un empeoramiento notable en calidad, y un aumento en tasa.

Una vez realizado el entrenamiento y validación para los decisores implementados, se presentarán los resultados finales con el conjunto de secuencias *test*, y se realizará una comparativa con el algoritmo **[2]** descrito en el apartado 4.2.1.

Vídeos utilizados:

Conjunto de vídeos *entrenamiento*: Akiyo, Bus, Coastguard, Container, Football, Foreman, Garden, Highway, Mobile, Mother_daughter, News, Paris, Silent, Stefan, Tempete, Waterfall, todos ellos de tamaño CIF.

Conjunto de vídeos *validación*: Akiyo, Carphone, Container, Foreman, Mobile, News, Salesman, Silent, Suzie, todos ellos de tamaño QCIF.

Conjunto de vídeos *test*: Bridge-far, Claire, Coastguard, Grandma, Miss-America, Mother_daughter, en tamaño QCIF y Akiyo, Container, Foreman, Mobile, Paris, Silent, Stefan, Tempete en tamaño CIF.

Todos estos conjuntos se encuentran formados por vídeos utilizados frecuentemente en la literatura de codificación de vídeo y cuya descripción se encuentra en el Anexo C. A primera vista podría parecer que el uso de parte de los vídeos de entrenamiento para mostrar los resultados finales no sería adecuado, pero se debe tener en cuenta que en el apartado entrenamiento se han utilizado unas 1000 muestras de un total de 38000 que forman cada secuencia utilizada, escogidas de manera aleatoria, haciendo improbable que el decisor se encuentre ajustado para dichos vídeos. Además se ha de destacar que las pruebas serán realizadas con valores de QP=28,32,36 y 40.

6.3 Diseño del decisor 1

Dados los resultados obtenidos en el apartado 5.7, el primer decisor intentará detectar de manera prematura el modo *Skip* en la tiras P, y el *Directo* en las tiras B. A pesar de mostrar ambos resultados dentro del mismo decisor, se ha de tener en cuenta que se trata de modos diferentes, pero que al aparecer cada uno de ellos en un tipo de tiras diferente, y ser los modos más probables a priori, se presentarán sus resultados de manera conjunta en la codificación.

6.3.1 Entrenamiento

Con el fin de describir el macrobloque, varias características como gradientes o vectores de movimiento fueron probadas en el proceso de selección. Finalmente, se utilizaron sólo características relacionadas con el coste de los modos, al resultar ser las más informativas para tomar la decisión.

Las características utilizadas para la decisión prematura del modo *Skip* han sido:

-Coste *Skip*/ SAD modo *Skip*:

$$\frac{SAD_{Skip} + \lambda R_{Skip}}{SAD_{Skip}} = 1 + \lambda \frac{R_{Skip}}{SAD_{Skip}}$$

-Coste *Inter16x16* en referencia cero / SAD modo *Skip*:

$$\frac{SAD_{16x16r0} + \lambda R_{16x16r0}}{SAD_{Skip}} = \frac{SAD_{16x16r0}}{SAD_{Skip}} + \lambda \frac{R_{16x16r0}}{SAD_{Skip}}$$

El coste *Skip* no supone un cálculo de estimación de movimiento (evitando así la parte de mayor coste computacional), mientras que para la obtención del coste *Inter16x16* en referencia cero se realizará el proceso de estimación y compensación de movimiento, pero acotado a la primera referencia de la lista correspondiente, siendo su cálculo liviano.

En el caso de la decisión prematura del modo *Directo* se utilizará:

-Coste *Directo*/ SAD modo *Directo*.

$$\frac{SAD_{Directo} + \lambda R_{Directo}}{SAD_{Directo}} = 1 + \lambda \frac{R_{Directo}}{SAD_{Directo}}$$

-Coste *Inter16x16* en referencia cero/ SAD modo *Directo:*

$$\frac{SAD_{Inter16x16r0} + \lambda R_{Inter16x16r0}}{SAD_{Directo}} = \frac{SAD_{Inter16x16r0}}{SAD_{Directo}} + \lambda \frac{R_{Inter16x16r0}}{SAD_{Directo}}$$

Al igual que en el caso anterior, el modo *Directo* no necesita cálculos de estimación de movimiento, por lo que la información necesaria para tomar la decisión no supondrá un coste computacional alto.

En ambos casos, la primera característica aportará información sobre la relación entre la tasa y la distorsión del modo *Directo* (o *Skip*). La segunda característica aportará información sobre la distorsión que ofrece el modo *Inter16x16* y la que ofrece el directo, además de la tasa necesaria para codificar con el modo *Inter16x16*. Como se ha comentado en apartados anteriores, las características se encuentran normalizadas, mejorando así la generalización.

Otra ventaja de utilizar estas características será que podremos evitar casos de probabilidad de falsa alarma, ya que en el caso de que el coste en referencia cero del modo *Inter16x16* sea menor que el modo *Directo* (o *Skip*), significará que existe al menos un modo Inter en referencia cero cuyo coste es menor, por lo que en ese caso, *Directo* (o *Skip*) no será el modo óptimo de codificación. Es por ello, que se pondrá como condición para evaluar una muestra que el coste *Directo* (o *Skip*) sea menor que el *Inter16x16* en referencia cero. En caso contrario, se realizará el proceso completo de decisión de modo.

El esquema de funcionamiento se representa en la figura 6.2 para el primer decisor en tiras P, y en la figura 6.3 para tiras B

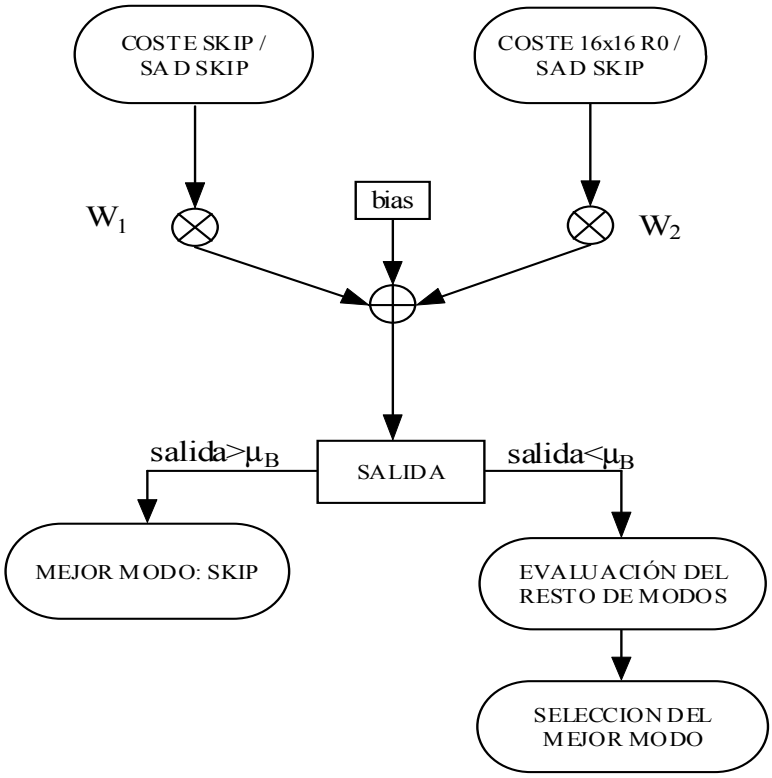


Figura 6.2 Esquema del decisor 1 en tiras P

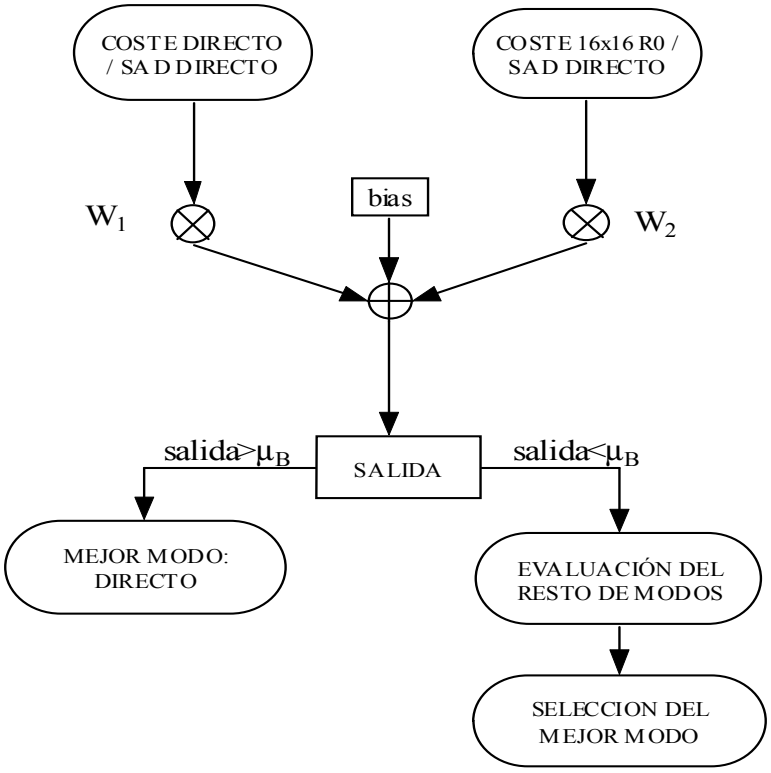


Figura 6.3 Esquema del decisor 1 en tiras B

Tras el proceso de entrenamiento los parámetros obtenidos fueron

Para un valor de α =3.5 Tiras B $\begin{cases} w_1 = -0.0424 \\ w_2 = -0.0154 \\ bias = 2.1664 \end{cases}$

Para un valor de α =5 Tiras P $\begin{cases} w_1 = -1.9562 \\ w_2 = 1.9044 \\ bias = 0.4402 \end{cases}$

6.3.2 Validación de parámetros

Las condiciones impuestas para la validación de parámetros en el primer decisor serán las siguientes:

IP $\begin{cases} \Delta\text{Tasa media} < 0.7\% \\ \Delta\text{Tasa para cada secuencia} < 1\% \\ \Delta\text{PSNR media} < 0.05\text{dB} \end{cases}$ I2B $\begin{cases} \Delta\text{Tasa media} < 1\% \\ \Delta\text{Tasa para cada secuencia} < 2\% \\ \Delta\text{PSNR media} < 0.1\text{dB} \end{cases}$

Al imponer restricciones tanto para el conjunto de secuencias como para cada una de ellas, garantizamos el funcionamiento general del decisor, y además de su estabilidad haciendo que no haya grandes desviaciones en función del vídeo, indicando esto una buena generalización. Por otra parte, estas restricciones dejarán margen para la

implementación de un segundo decisor, al partir de unas condiciones de codificación mínimas tanto de calidad (PSNR) como de incremento de tasa.

Como se ha comentado anteriormente, el valor del umbral (μ) en el decisor, se calculará de manera independiente para cada patrón analizado. Los resultados obtenidos se muestran a continuación, siguiendo las formulas indicadas en el apartado 5.5:

Resultados en IP para μ_P=0.6

Secuencia	Tiempo (%)	ΔTasa %	PSNR (dB)
Akiyo (QCIF)	78,58	0,48	-0,028
Carphone (QCIF)	39,07	0,69	-0,036
Container (QCIF)	76,35	0	0
Foreman (QCIF)	27,05	1,04	-0,046
Mobile (QCIF)	9,34	0,25	-0,005
News (QCIF)	68	0,70	-0,042
Salesman (QCIF)	74,65	0,30	-0,017
Silent (QCIF)	60,31	0,91	-0,052
Suzie(QCIF)	38,07	0,30	-0,020
Media	52.3854	0.52	-0.027

Tabla 6.1 Resultados de validación decisor 1 IP

Resultados en I2B para μ_P=0.6 y μ_B=1

Secuencia	Tiempo (%)	ΔTasa (%)	PSNR (dB)
Akiyo (QCIF)	77,31	0,24	-0,022
Carphone (QCIF)	42,74	1,62	-0,076
Container (QCIF)	74,48	0,60	-0,027
Foreman (QCIF)	35,53	1,13	-0,062
Mobile (QCIF)	25,37	0,72	-0,040
News (QCIF)	64,65	1,35	-0,086
Salesman (QCIF)	70,57	0,82	-0,041
Silent (QCIF)	59,40	1,42	-0,073
Suzie(QCIF)	46,17	0,56	-0,022
Media	55.14	0.94	-0.0507

Tabla 6.2 Resultados de validación decisor 1 I2B

Observando el resultado de las tablas se puede comprobar que se cumplen las condiciones impuestas, manteniéndose el incremento de tasa y la perdida de calidad por debajo de los límites establecidos tanto en el patrón IP como en el I2B. Es evidente que la mejora en tiempo no podrá ser estable para todos los vídeos, ya que dependerá de cuantas veces el modo *Directo* o *Skip* sea el mejor para determinada secuencia. Cabe destacar el funcionamiento del primer decisor en secuencias como Akiyo y Salesman, en las que se consigue una mejora en tiempo del 70% con incrementos de tasa despreciables y pérdidas de calidad muy bajas.

El hecho de que este conjunto de secuencias cumplan las restricciones impuestas no nos garantizará que las cumpla cualquier vídeo. A pesar de la buena generalización del algoritmo, es posible que haya secuencias de vídeo, que por sus características (movimientos muy bruscos, o muchos cambios de escena), no sean fácilmente

clasificables, al encontrarse las salidas del decisor muy próximas a la frontera de clasificación. A pesar de esto, y vistos los resultados de validación se puede concluir que el decisor generaliza bastante bien, teniendo unos resultados muy regulares, permitiendo el diseño de un segundo decisor.

6.4 Diseño del decisor 2

Este decisor se encargará de realizar una parada prematura en la decisión de modo para aquellas muestras que no hayan sido seleccionadas en la primera etapa. Para ello, en un primer lugar se intentó realizar parada prematura en el modo *Inter16x16*, pero debido a los resultados obtenidos se optó por una implementación más conservadora. Para evitar errores importantes en el proceso de codificación, el segundo decisor elegirá si el mejor modo pertenece a los modos grandes o a los modos pequeños. Esto reducirá el error, pero supondrá que el ahorro en tiempo obtenido no será tan importante como el obtenido por el primer decisor. Además, esta forma de decidir supondrá diferencias importantes tanto en la función de coste como en el umbral del decisor.

Trataremos en primer lugar el umbral. En este caso, si se plantea como en el primer decisor, variarlo no conseguirá reducir la falsa alarma, si no que hará que se escojan un mayor número de veces los modos grandes o los modos pequeños. Es por ello que se ha decidido establecer una región de indeterminación, en la cual se evaluarán todos los modos de predicción Inter, tanto grandes como pequeños, con el fin de poder regular el error, aun a riesgo de bajar por ello las prestaciones en tiempo. Para ello se han establecido dos umbrales de manera simétrica alrededor de cero.

Como se ha dicho, también será necesario cambiar la función de coste, ya que no tenemos un solo modo, sino que se seleccionará entre dos conjuntos. Esto hace que quede como:

$$C(w) = (t-y)^2 \left(1 + \alpha \left(\frac{\left| C_{\min\, grandes} - C_{\min\, pequeños} \right|}{\mu_{\left(C_{\min\, grandes} - C_{\min\, pequeños} \right)}} \right) \right)$$

Ecuación 6.1 Función de coste para el decisor 2

Al redefinir la función de coste de esta forma, conseguiremos que al entrenar se preste más atención a los casos en los que una mala clasificación suponga un gran incremento en la tasa o una perdida importante de calidad, al ser la diferencia de costes grande.

Una vez aclarados los cambios introducidos para el diseño del segundo decisor, pasamos a la fase de entrenamiento.

6.4.1 Entrenamiento

Para esta fase se han utilizado las mismas secuencias que en el decisor anterior, pero descartando aquellas muestras en las que el mejor modo era el *Directo* (o *Skip*), ya que estas son seleccionadas en la etapa previa.

Tras el proceso de selección de características, se han escogido las siguientes

Coste *Inter16x16* / SAD *Inter16x16*:

$$\frac{SAD_{Inter16x16} + \lambda R_{Inter16x16}}{SAD_{Inter16x16}} = 1 + \lambda \frac{R_{Inter16x16}}{SAD_{Inter16x16}}$$

Coste *Inter8x8* en referencia cero/ SAD *Inter16x16*:

$$\frac{SAD_{Inter8x8r0} + \lambda R_{Inter8x8r0}}{SAD_{Inter16x16}} = \frac{SAD_{Inter8x8r0}}{SAD_{Inter16x16}} + \lambda \frac{R_{Inter8x8r0}}{SAD_{Inter16x16}}$$

En el caso del modo *Directo* se utilizará:

Coste *Directo*/ SAD *Directo:*

$$\frac{SAD_{Directo} + \lambda R_{Directo}}{SAD_{Directo}} = 1 + \lambda \frac{R_{Directo}}{SAD_{Directo}}$$

Coste *Inter16x16*/ SAD *Directo:*

$$\frac{SAD_{Inter16x16} + \lambda R_{Inter16x16}}{SAD_{Directo}} = \frac{SAD_{Inter16x16}}{SAD_{Directo}} + \lambda \frac{R_{Inter16x16}}{SAD_{Directo}}$$

Se puede ver que en este caso se ha hecho uso del coste *Inter16x16* así como del coste *Inter8x8* en referencia cero en el caso de tiras P, lo que supondrá un aumento en el coste computacional para la toma de esta decisión. Esto se debe a que con las características disponibles hasta ese momento no se podía realizar la decisión con una tasa de acierto suficiente.

En el caso de las tiras P el coste *Inter16x16*, que es el más elegido entre los modos grandes, supondrá una muy buena aproximación al coste del mejor modo grande, mientras que el coste *Inter8x8* en referencia cero nos proporcionará una estimación del coste de los modos pequeños, no teniendo que hacer uso de varias referencias, evitando así un buen número de estimaciones de movimiento.

En el caso de tiras B, ha sido suficiente con la información aportada por el coste *Inter16x16* y el coste del modo *Directo* para poder tomar decisiones sin que estas supongan un decremento importante en la eficiencia de codificación.

Se ha de resaltar también que al haberse calculado el modo directo y el 16x16 durante el proceso, el codificador elegirá siempre el mejor modo de entre los calculados, a pesar de que se haya tomado la decisión de parada prematura en modos grandes o

pequeños. Es decir, si se decide por ejemplo que el mejor modo pertenece a los modos grandes, se calcularán estos, y el modo elegido como mejor será el mínimo de todos los calculados (*Directo*/*Skip*, *Inter16x16* modos pequeños y modos Intra).

El esquema de funcionamiento se representa en la figura 6.4 para el segundo decisor en tiras P, y en la figura 6.5 para tiras B

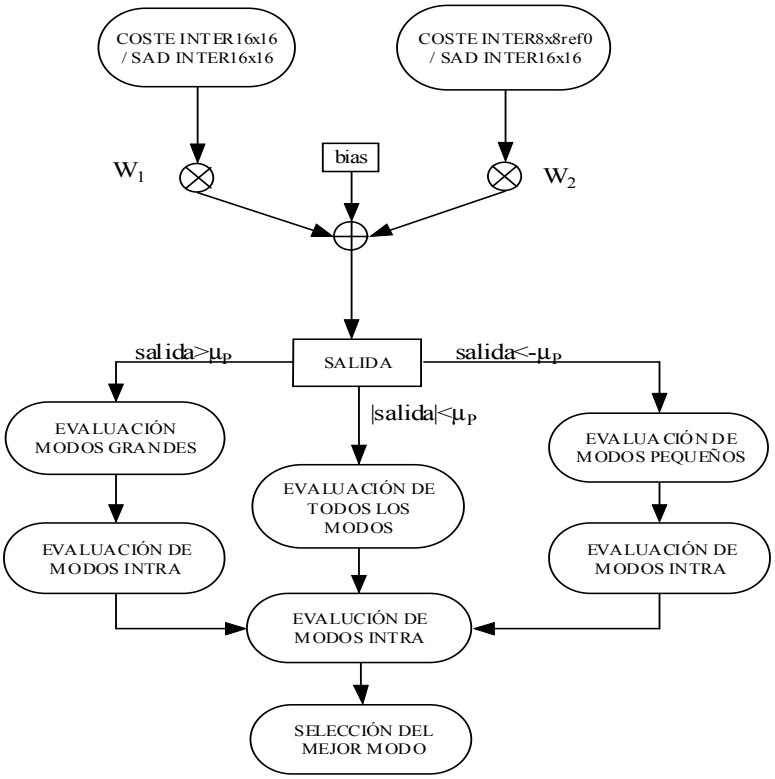


Figura 6.4 Esquema del decisor 2 en tiras P

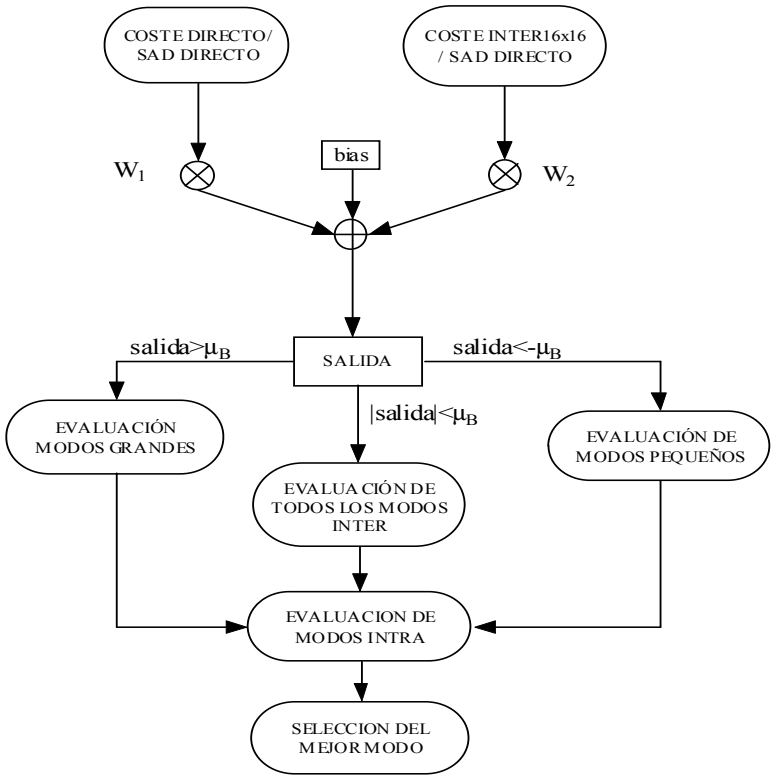


Figura 6.5 Esquema del decisor 2 en tiras B

Tras el proceso de entrenamiento los parámetros obtenidos fueron

Para un valor de α =4 Tiras P $\begin{cases} w_1 = -0.2607 \\ w_2 = 0.2388 \\ bias = 0.9583 \end{cases}$

Para un valor de α =6 Tiras B $\begin{cases} w_1 = 0.0222 \\ w_2 = -0.0723 \\ bias = 1.4034 \end{cases}$

6.4.2 Validación de parámetros

Las condiciones impuestas para la validación de parámetros deberán ser más flexibles que en el caso anterior, ya que se partirá de la perdida en PSNR y el aumento de tasa producido por el primer decisor. En este caso se pondrán las siguientes restricciones de validación:

IP $\begin{cases} \Delta\text{Tasa media} < 1\% \\ \Delta\text{Tasa para cada secuencia} < 1{,}5\% \\ \Delta\text{PSNR media} < 0{,}1\text{dB} \end{cases}$ I2B $\begin{cases} \Delta\text{Tasa media} < 1{,}3\% \\ \Delta\text{Tasa para cada secuencia} < 2{,}5\% \\ \Delta\text{PSNR media} < 0.13\text{dB} \end{cases}$

Como en el caso anterior, los patrones IP e I2B se validarán de manera independiente

Resultados en IP para μ_P=0,7

Secuencia	Tiempo (%)	ΔTasa (%)	PSNR(dB)
Akiyo (QCIF)	80,45	0,54	-0,035
Carphone (QCIF)	51,31	1,40	-0,070
Container (QCIF)	77,11	0,13	0,003
Foreman (QCIF)	41,25	1,14	-0,052
Mobile (QCIF)	18,39	0,90	-0,037
News (QCIF)	69,88	1,36	-0,072
Salesman (QCIF)	76,30	0,27	-0,018
Silent (QCIF)	64,44	1,50	-0,071
Suzie(QCIF)	52,77	0,62	-0,022
Media	59.10	0.87	-0.042

Tabla 6.3 Resultados de validación decisor 2 IP

Resultados en I2B para μ_P=1,9 y μ_B=0,5

Secuencia	Tiempo (%)	ΔTasa (%)	PSNR (dB)
Akiyo (QCIF)	78,66	0,27	-0,026
Carphone (QCIF)	53,52	2,14	-0,110
Container (QCIF)	74,28	0,60	-0,027
Foreman (QCIF)	47,25	1,29	-0,067
Mobile (QCIF)	35,21	1,12	-0,059
News (QCIF)	67,14	1,48	-0,092
Salesman (QCIF)	72,04	0,86	-0,040
Silent (QCIF)	63,14	1,83	-0,096
Suzie(QCIF)	53,30	0,66	-0,023
Media	60.50	1.14	-0.060

Tabla 6.4 Resultados de validación decisor 2 I2B

De los resultados mostrados en las tablas se puede concluir que el segundo decisor supone una mejora importante en tiempo respecto del anterior, aunque a costa de un incremento en tasa y una perdida de calidad. Para tiras P se consigue una mejora media de en torno al 7%, mientras que para tiras B la mejora se sitúa en torno al 5%. Estas mejoras, que en principio podrían parecer bajas, son importantes, ya que hay que tener en cuenta la capacidad de acción que queda sobre el resto de modos una vez se ha actuado sobre el *Directo* y el *Skip*, que tienen una probabilidad a priori de ser seleccionados como mejor modo de en torno al 0,5 y 0,7 de media en tiras P y B respectivamente. Además se ha de tener en cuenta que para este segundo decisor se han utilizado características que suponen un coste computacional más alto que las utilizadas

en el apartado anterior, y los modos Intra han sido evaluados en todos los casos debido a la dificultad de la clasificación.

6.5 Test

Para evaluar los resultados del algoritmo implementado, se compararán con **[1]** y **[2]**. Como se ha comentado en el apartado 4.2.1, ambos algoritmos son complementarios, siendo el algoritmo presente en **[1]** el que se encargue de la parada prematura en tiras P del modo *Skip*, y **[2]** de la parada prematura en tiras B del modo *Directo*. Ambos han sido aceptados por el Joint Video Team (JVT) del ISO/IEC MPEG & ITU-T VCEG, por lo que se encuentran implementados en el software de referencia, lo que permitirá que los datos expuestos en la comparación sean obtenidos en las mismas condiciones que los algoritmos evaluados. Se ha de resaltar además que cuando se muestren los resultados obtenidos por **[2]** en el patrón I2B, ambos algoritmos estarán activados, detectando de esta manera tanto modo *Directo* como modo *Skip*.

IP

Secuencia	Algoritmo	Tiempo (%)	ΔTasa (%)	PSNR (dB)
Bridge-far (QCIF)	[1]	61,78	0,07	-0,003
	Decisor 1	93,81	0,81	-0,050
	Decisor 2	93,83	0,97	-0,058
Claire (QCIF)	[1]	44,95	0,47	-0,022
	Decisor 1	75,02	0,61	-0,039
	Decisor 2	75,93	0,42	-0,028
Coastguard (QCIF)	[1]	14,65	-0,01	0,001
	Decisor 1	25,99	-0,03	0,005
	Decisor 2	45,33	0,42	-0,017
Grandma (QCIF)	[1]	46,34	0,04	0,003
	Decisor 1	76,35	0,40	-0,023
	Decisor 2	77,93	0,43	-0,022
Miss-america (QCIF)	[1]	34,89	-0,51	0,020
	Decisor 1	66,15	-0,07	0,003
	Decisor 2	68,40	0,14	-0,004
Mother_daughter (QCIF)	[1]	34,61	-0,01	0,014
	Decisor 1	62,90	0,42	-0,004
	Decisor 2	68,99	0,10	0,001
Akiyo (CIF)	[1]	44,39	-0,14	0
	Decisor 1	76,79	0,35	-0,023
	Decisor 2	80,51	0,30	-0,013
Container (CIF)	[1]	45,26	-0,60	0,019
	Decisor 1	74,24	0,27	-0,014
	Decisor 2	77,42	0,52	-0,027
Foreman (CIF)	[1]	16,64	-0,04	0
	Decisor 1	37,42	1,29	-0,066
	Decisor 2	51,70	1,73	-0,074
Mobile (CIF)	[1]	7,495	-0,022	-0,002
	Decisor 1	11,61	0,50	-0,016
	Decisor 2	26,04	1,20	-0,053
Paris (CIF)	[1]	31,57	0,16	-0,012
	Decisor 1	50,86	0,91	-0,052
	Decisor 2	55,12	1,34	-0,064
Silent (CIF)	[1]	38,15	-0,10	0,002
	Decisor 1	65,57	1,07	-0,046
	Decisor 2	71,43	1,19	-0,048
Stefan (CIF)	[1]	12,49	-0,15	0,001
	Decisor 1	23,15	0,53	-0,018
	Decisor 2	30,57	2,04	-0,080
Tempete	[1]	9,708	0,18	-0,008
	Decisor 1	20,85	1,04	-0,039
	Decisor 2	31,18	1,09	-0,043
Media	**[1]**	**31.64**	**-0.04**	**0.001**
	Decisor 1	**54.34**	**0.58**	**-0.027**
	Decisor 2	**61.03**	**0.85**	**-0.037**

Tabla 6.5 Resultados de test para IP

Del funcionamiento al seleccionar el patrón IP, se ha de resaltar en primer lugar la gran diferencia que suponen los algoritmos implementados respecto a **[1]**, pasando de un 31% de ahorro en tiempo a un 54% y 60% en media para los decisores 1 y 2 respectivamente. Esto conlleva un aumento no despreciable en la tasa respecto a la conseguida por **[1]**, en el cual el incremento de tasa se mantiene próximo a cero, pero sin llegar a superar en la mayoría de las secuencias el 1,5% de incremento, excepto en el caso de la secuencia Stefan, en el que la tasa aumenta hasta un 2%, cifra que aunque alta comparándola con la obtenida por **[1]**, no puede considerarse fuera de los límites para evaluar como bueno el funcionamiento del algoritmo.

I2B

Secuencia	Algoritmo	% Tiempo	%Tasa	PSNR
Bridge-far (QCIF)	[2]	84,11	-0,07	-0,003
	Decisor 1	89,54	0,34	-0,027
	Decisor 2	89,49	0,34	-0,028
Claire (QCIF)	[2]	72,27	-0,30	0,026
	Decisor 1	76,60	0,19	-0,012
	Decisor 2	79,18	0,24	-0,016
Coastguard (QCIF)	[2]	43,87	2,76	-0,091
	Decisor 1	41,42	0,20	-0,015
	Decisor 2	48,28	0,79	-0,043
Grandma (QCIF)	[2]	74,18	-0,19	0,006
	Decisor 1	73,37	0,37	-0,013
	Decisor 2	73,99	0,42	-0,015
Miss-america (QCIF)	[2]	63,52	0,61	-0,033
	Decisor 1	69,81	0,43	-0,013
	Decisor 2	72,15	0,56	-0,021
Mother_daughter (QCIF)	[2]	65,43	-0,36	0,006
	Decisor 1	66,18	0,42	-0,028
	Decisor 2	68,16	0,49	-0,037
Akiyo (CIF)	[2]	73,08	0,41	-0,021
	Decisor 1	78,25	0,82	-0,041
	Decisor 2	79,81	0,87	-0,050
Container (CIF)	[2]	74,94	0,37	-0,020
	Decisor 1	73,32	1,18	-0,029
	Decisor 2	74,10	1,41	-0,044
Foreman (CIF)	[2]	40,74	5,75	-0,237
	Decisor 1	41,87	1,40	-0,066
	Decisor 2	52,28	1,61	-0,070
Mobile (CIF)	[2]	24,19	1,47	-0,049
	Decisor 1	21,77	0,93	-0,061
	Decisor 2	28,35	1,30	-0,082
Paris (CIF)	[2]	49,74	1,33	-0,067
	Decisor 1	48,21	1,11	-0,057
	Decisor 2	54,87	1,58	-0,080
Stefan (CIF)	[2]	62,65	0,78	-0,030
	Decisor 1	63,69	1,06	-0,045
	Decisor 2	67,97	1,50	-0,070
Silent (CIF)	[2]	27,16	2,70	-0,107
	Decisor 1	30,19	0,98	-0,049
	Decisor 2	41,59	1,53	-0,073
Tempete (CIF)	[2]	33,21	3,04	-0,109
	Decisor 1	33,19	0,81	-0,040
	Decisor 2	42,69	1,17	-0,052
Media	**[2]**	**56.36**	**1.31**	**-0.052**
	Decisor 1	**57.67**	**0.73**	**-0.036**
	Decisor 2	**62.35**	**0.99**	**-0.049**

Tabla 6.6 Resultados de test para I2B

De la comparación anterior se puede comprobar como el funcionamiento en el patrón I2B es mejor, tanto en velocidad como en tasa y PSNR para la mayoría de los casos. Además se puede observar que sus resultados son más estables. Utilizando el algoritmo **[2]** la tasa se dispara en varios casos de los analizados (Coastguard Foreman Silent y Tempete) en los que se sobrepasa el 2% mientras que en otros obtiene incluso mejoras en tasa (Akiyo Claire Grandma y Mother_Daughter), obteniendo por tanto resultados muy irregulares.

Después de analizar los resultados para ambos patrones, se puede concluir que el algoritmo implementado supera en rendimiento al expuesto en [1] y [2] por varias razones. En primer lugar, demuestra un mejor funcionamiento cuando se utilizan tiras B, al obtenerse una reducción del coste computacional mayor con un incremento en tasa muy por debajo del presentado por [1] y [2]. En cuanto a las tiras P, a pesar de tener un mayor incremento de tasa respecto a la referencia, esta no puede considerarse como alta, ya que se mantiene en el 0.85% de media, siendo las mejoras en coste computacional considerables (se pasa de un 31% en media ofrecido por [1] a un 60%).En segundo lugar el algoritmo propuesto es más flexible, ya que mediante la utilización de umbrales se podrá calibrar la relación entre incremento de tasa y ahorro en tiempo. Esto permitiría reducir el incremento de tasa en el caso de que se utilice en aplicaciones en las que la tasa sea crucial, o reducir el coste computacional en aquellas aplicaciones en las que prime la velocidad de codificación.

A continuación se muestran los resultados de manera gráfica, siguiendo las indicaciones mostradas en el apartado 5.6, para las secuencias Paris CIF con un patrón IP y Akiyp CIF con un patrón I2B, con el objeto de ilustrar los resultados obtenidos.

Paris CIF IP

A la vista de la figura 6.6 se ha de destacar la similitud entre la referencia y el algoritmo propuesto, ya que ambas figuras prácticamente se superponen. Se puede también la tendencia de la tasa respecto a la PSNR, que como se esperaba aumenta a medida que aumenta la calidad, ya que serán necesarios más bits para la transmisión de secuencias más complejas al generar un mayor residuo y necesitarse vectores de movimiento más grandes.

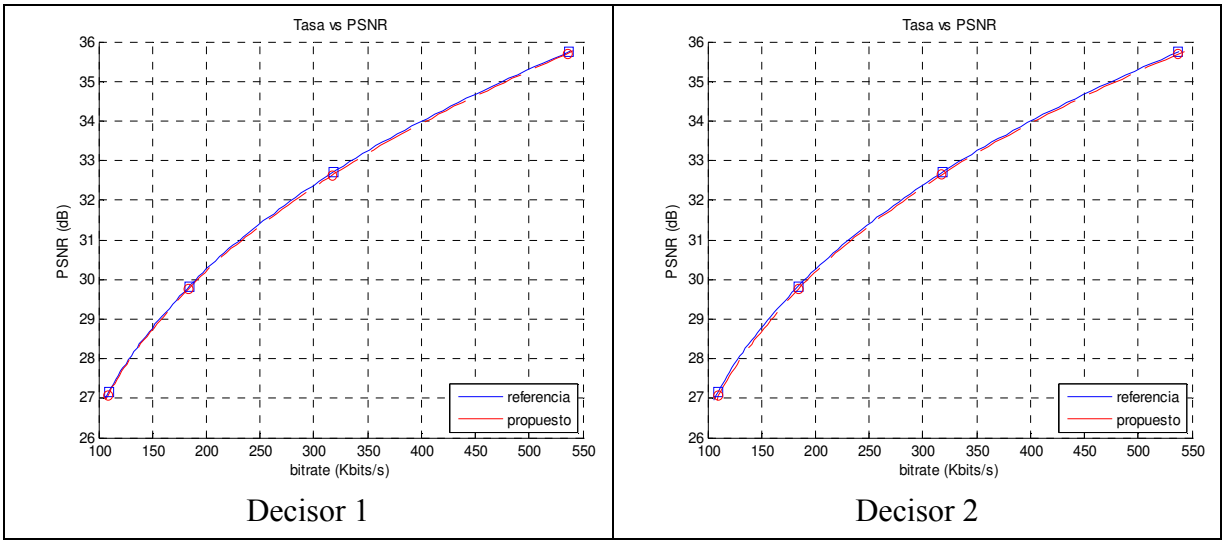


Figura 6.6 Gráfica de PSNR vs Tasa

En la figura 6.7 se muestra la mejora en tiempo obtenida por el algoritmo implementado respecto de la PSNR para la secuencia Paris para un patrón IP. Como se esperaba, a calidades bajas se obtiene un mayor ahorro computacional, ya que para QP altas (recordemos que un escalón de cuantificación alto supone una merma en la calidad) habrá un mayor número de muestras seleccionadas como *Skip* y *Directo* por el primer decisor. Además se puede observar la mejora itroducida por el seugndo decisor que produce una mejora de aproximadamente un 5%, como se mostraba en la tabla 6.5

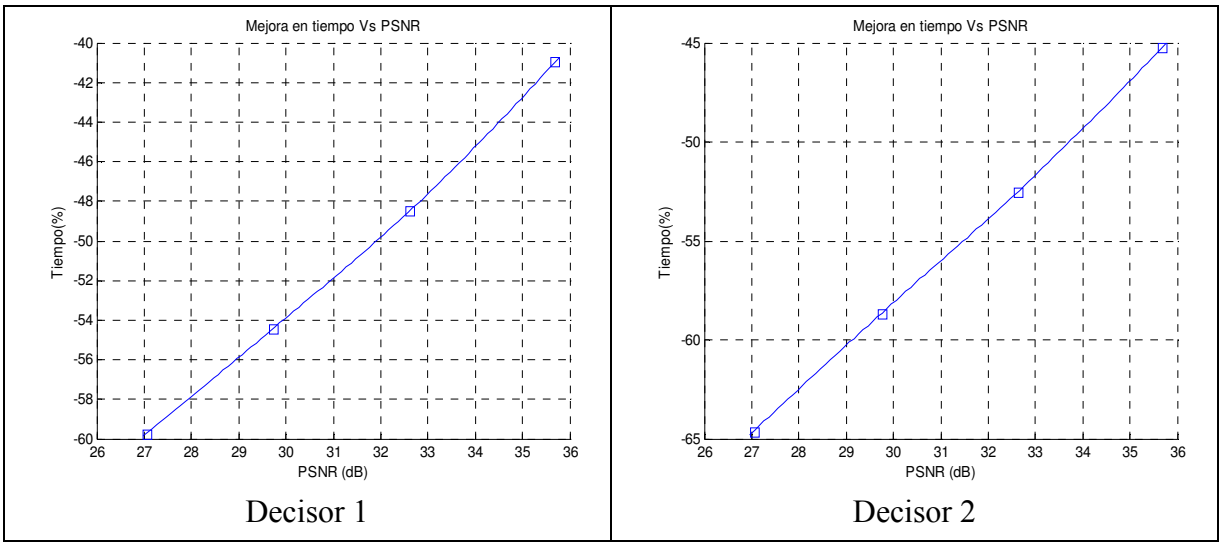


Figura 6.7 Gráfica de mejora en tiempo vs PSNR

La siguiente figura muestra el incremento de tasa que se produce al utilizar el algoritmo diseñado, respecto a la referencia, para los distintos valores de PSNR. Esta gráfica cambiará mucho según el vídeo analizado, ya que dependiendo de las características propias de cada secuencia, los decisores diseñados funcionarán mejor o peor para las distintas calidades. Se puede ver que al introducir el segundo decisor, el incremento de tasa aumenta de manera considerable, y esto a pesar de que para la decisión en tiras P se han utilizado variables nuevas, que caracterizan los modos grandes (coste *Inter16x16*) y los modos pequeños (coste *Inter4x4*).

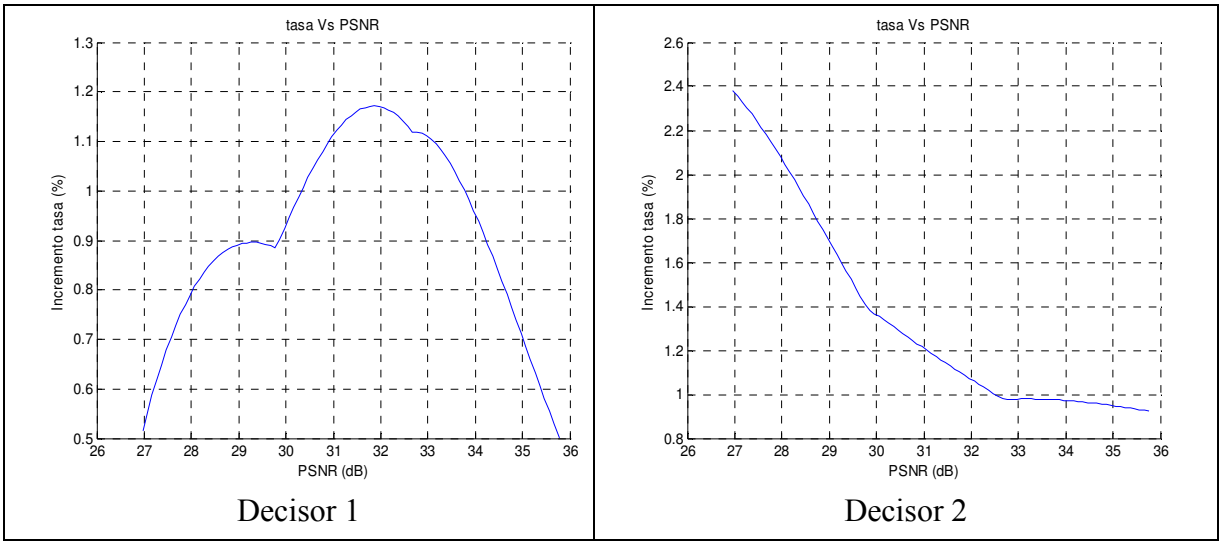


Figura 6.8 Gráfica de incremento de tasa vs PSNR

Akiyo CIF I2B

Como en el caso anterior, se mantiene la similitud entre la referencia y el algoritmo propuesto, incluso al introducir el segundo decisor, siendo prácticamente inapreciable la perdida de calidad para un mismo valor de tasa.

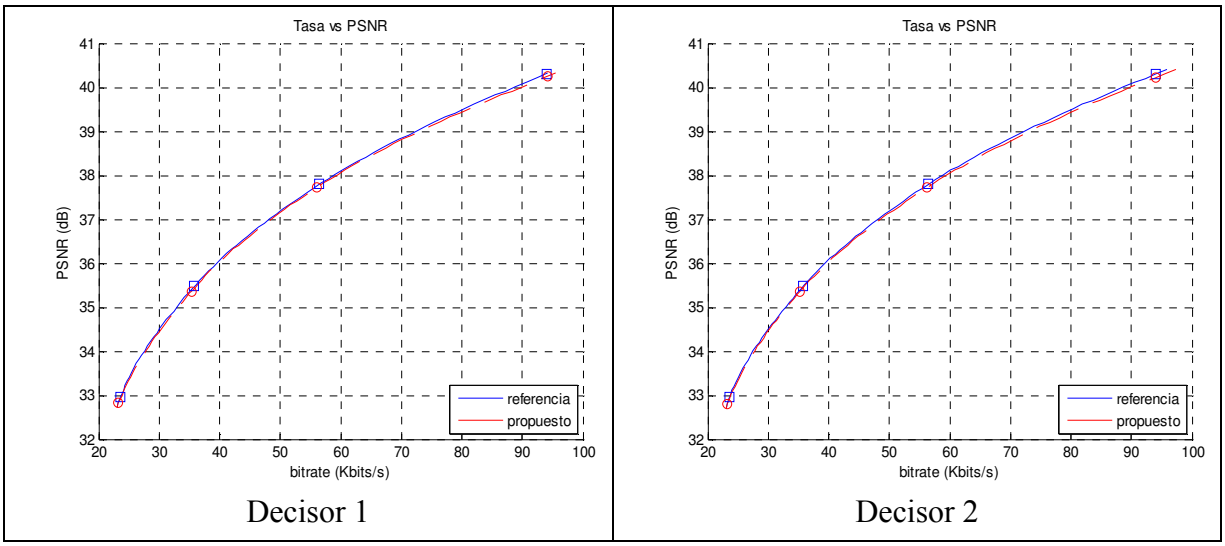


Figura 6.9 Gráfica de PSNR vs Tasa

En la figura 6.10 se muestra la mejora en tiempo obtenida por el algoritmo implementado respecto de la PSNR para la secuencia Akiyo en resolución CIF. Como se esperaba la tendencia es la misma que en el caso del patrón IP, siendo mucho mayor la mejora para calidades bajas, debido a la alta probabilidad a priori de los modos *Directo* y *Skip*. Además se puede ver que el decisor 2 actuará sobre todo a calidades altas, donde es más probable que aparezcan modos Inter (ver apartado 5.7). Esto reducirá la pendiente de la curva de mejora en tiempo haciendo que sea más estable para todas las calidades.

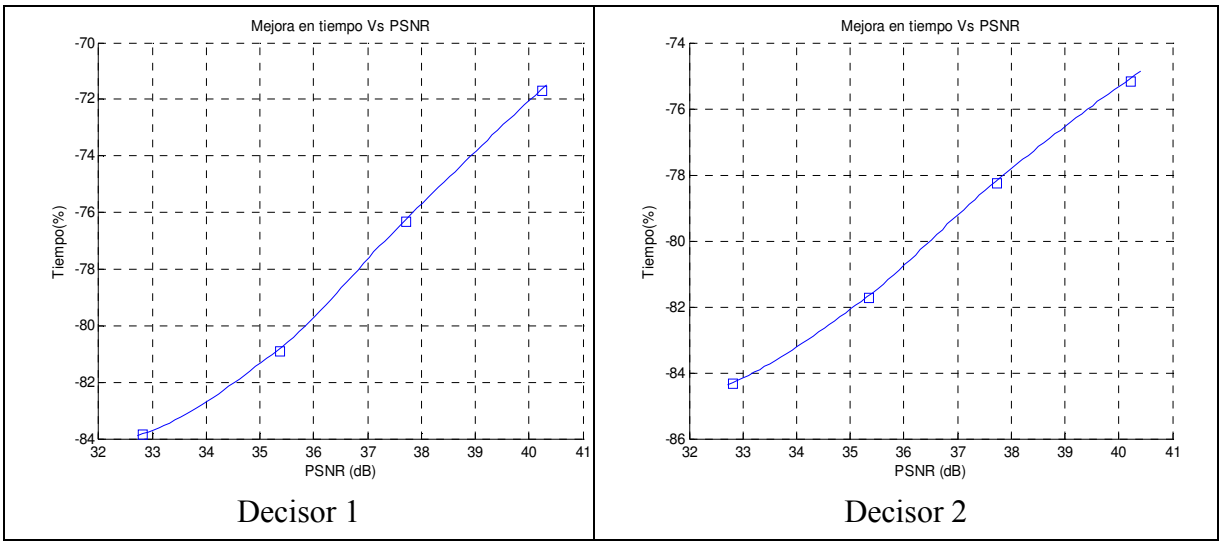


Figura 6.10 Gráfica de mejora en tiempo vs PSNR

A continuación se muestra el incremento de tasa para distintos valores de PSNR. En este caso, el incremento es mucho menor al introducir el segundo decisor. Esto es debido a que el segundo decisor no actuará en demasiadas ocasiones, ya que con el

primer decisor se ahorrará el 78% del tiempo, por lo que no habrá demasiados macrobloques que pasen por la segunda etapa de decisión. Esto supondrá que el margen de mejora para el segundo decisor será muy bajo (1% en media como se muestra en la tabla 6.6).

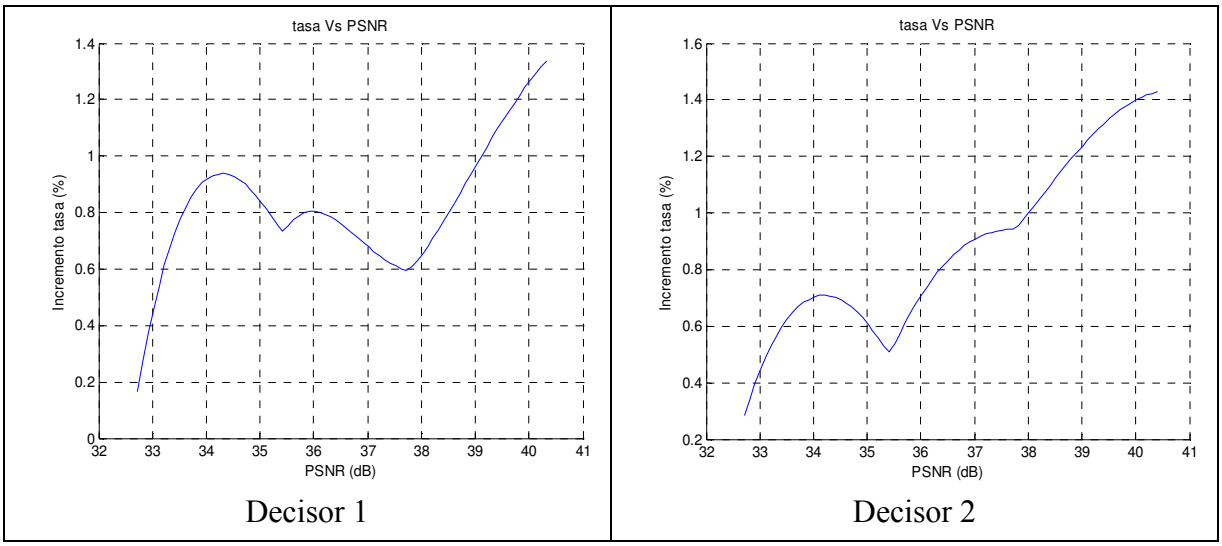


Figura 6.11 Gráfica de incremento de tasa vs PSNR

7. Conclusiones y líneas futuras

7.1 Conclusiones

En el presente proyecto se ha presentado el trabajo realizado en el entorno de la codificación de vídeo, en la etapa de decisión de modo, con el objetivo de conseguir una reducción en el coste computacional del codificador H.264.

En primer lugar se ha diseñado un clasificador encargado de la decisión prematura de los modos *Skip* en tiras P y *Directo* en tiras B. El funcionamiento de dicho clasificador supone una gran mejora en lo que a estimaciones de movimiento se refiere, consiguiendo ahorros de tiempo considerables en la mayoría de secuencias probadas, manteniendo el incremento de tasa bajo, con una perdida de calidad muy baja.

El segundo clasificador diseñado decidirá si el modo óptimo pertenece al grupo de los modos grandes (*Inter16x16*, *Inter16x8* o *Inter8x16*) o de los modos pequeños (*Inter8x8* y sus subdivisiones). Presenta una mejora en coste computacional mucho más reducida que el anterior. Esto se debe a varios motivos: Los modos *Skip* y *Directo* son los más utilizados (tienen una probabilidad a priori de ser seleccionados como mejor modo del 0,5 y 0,7 en media respectivamente) por lo que las mejoras obtenidas sobre el primer decisor serán siempre menos importantes. También se ha de tener en cuenta ya no se seleccionará un modo, si no un grupo de modos, por lo que se deberán realizar un número importante de estimaciones de movimiento. Además, se ha hecho necesaria la utilización de dos características nuevas para la decisión, lo que se partirá de un incremento del coste computacional respecto de los resultados del primer decisor.

Como es lógico, dependiendo del incremento de tasa que se tolere (y por tanto del porcentaje de falsa alarma que obtenga en los clasificadores) se podrá obtener un mayor o menor ahorro temporal. Esto será regulable mediante la variación de los umbrales situados a la salida de los perceptrones. La posibilidad de variar el comportamiento del algoritmo de manera sencilla supone una gran ventaja respecto a otras soluciones vistas, ya que permitirá adaptar el algoritmo a las aplicaciones según la calidad necesaria, y la capacidad de cálculo disponible.

Comparándonos con los algoritmos presentados en el estudio del estado del arte podemos decir que superamos al algoritmo propuesto en **[1]** y **[2]** para la detección de *Skip*/*Directo* en prestaciones al codificar patrones con tiras, obteniendo una reducción mayor en coste computacional, y consiguiendo un incremento de tasa menor, con una perdida de calidad muy baja en el patrón I2B. Es evidente que las prestaciones en el patrón IP del método propuesto por **[1]** son muy buenas, ya que la tasa y la PSNR no se ven afectadas, pero el hecho de que la mejora en tiempo no sea flexible supone un gran inconveniente. Por el contrario, el algoritmo propuesto, a pesar de que suponga un aumento en tasa y una perdida de calidad, estás son muy bajas, y se consigue una reducción mucho mayor en el tiempo de codificación (una media del 30%) que el presentado en **[1]**.

7.2 Líneas futuras

Tras la realización del proyecto, y comprobado el potencial del uso de clasificadores para la decisión prematura de modo, surgen varias líneas de investigación futuras:

-Profundizar en las etapas de decisión Inter, realizando clasificadores para llevar a cabo la decisión a nivel de submacrobloque. Para ello es posible que se necesite el cálculo de nuevas variables, y muy posiblemente el cálculo de algún coste, por lo que habría que evaluar la posible mejora en tiempo, respecto de la perdida en PSNR y el aumento de tasa que se podría producir.

-Extensión del problema a la decisión Inter/Intra mediante el uso de características como las propuestas, o combinación del algoritmo realizado con alguno de los existentes en esta área. Se ha de destacar que en el caso de parada prematura en el primer clasificador (*Skip*/*Directo*) no se comprobarán los modos Intra, pero en caso contrario, los modos Intra serán evaluados, lo cual es un coste importante teniendo en cuenta que pocas veces son elegidos estos modos.

-Todo el trabajo se ha realizado utilizando perceptrones monocapa. Es posible que el uso de otro tipo de redes neuronales más complejas (pero cuya implementación sea asumible dentro de un codificador) resultase en mejores prestaciones, debido a la limitación inherente del método utilizado, ya que como se ha explicado, el perceptrón sólo será capaz de discriminar conjuntos linealmente separables.

Anexo A Fichero de configuración utilizado en el software de referencia JM10.2

```
# New Input File Format is as follows
# <ParameterName> = <ParameterValue> # Comment
#
# See configfile.h for a list of supported ParameterNames


###############################################################################
# Files
###############################################################################
InputFile           = "C:\videos\CIF\paris.cif"       # Input sequence
InputHeaderLength    = 0     # If the inputfile has a header, state it's length in byte here
StartFrame           = 0     # Start frame for encoding. (0-N)
FramesToBeEncoded    = 7     # Number of frames to be coded
FrameRate            = 25.0  # Frame Rate per second (0.1-100.0)
SourceWidth          = 352   # Frame width
SourceHeight         = 288   # Frame height
TraceFile           = "trace_enc.txt"
ReconFile           = "test_rec.yuv"
OutputFile          = "test.264"

###############################################################################
# Encoder Control
###############################################################################
ProfileIDC          = 77  # Profile IDC (66=baseline, 77=main, 88=extended; FREXT
Profiles: 100=High, 110=High 10, 122=High 4:2:2, 144=High 4:4:4, for params see
below)
LevelIDC            = 51  # Level IDC   (e.g. 20 = level 2.0)

IntraPeriod         =  10  # Period of I-Frames (0=only first)
EnableOpenGOP        =  0  # Support for open GOPs (0: disabled, 1: enabled)
IDRIntraEnable       =  0  # Force IDR Intra  (0=disable 1=enable)
QPISlice            =  40  # Quant. param for I Slices (0-51)
QPPSlice            =  40  # Quant. param for P Slices (0-51)
FrameSkip           =  2  # Number of frames to be skipped in input (e.g 2 will code
every third frame)
ChromaQPOffset       =  0  # Chroma QP offset (-51..51)
UseHadamard          =  1  # Hadamard transform (0=not used, 1=used)
DisableSubpelME      =  0  # Disable Subpixel Motion Estimation (0=off/default, 1=on)
SearchRange         = 32  # Max search range
NumberReferenceFrames =  5  # Number of previous frames used for inter motion
search (1-16)
PList0References     =  0  # P slice List 0 reference override (0 disable, N <=
NumberReferenceFrames)
Log2MaxFNumMinus4    =  0  # Sets log2_max_frame_num_minus4 (-1 : based on
FramesToBeEncoded/Auto, >=0 : Log2MaxFNumMinus4)
```

```
Log2MaxPOCLsbMinus4   = -1  # Sets log2_max_pic_order_cnt_lsb_minus4 (-1 :
Auto, >=0 : Log2MaxPOCLsbMinus4)

GenerateMultiplePPS   =  0  # Transmit multiple parameter sets. Currently parameters
basically enable all WP modes (0: diabled, 1: enabled)
ResendPPS             =  0  # Resend PPS (with pic_parameter_set_id 0) for every coded
Frame/Field pair (0: disabled, 1: enabled)

MbLineIntraUpdate     =  0  # Error robustness(extra intra macro block updates)(0=off,
N: One GOB every N frames are intra coded)
RandomIntraMBRefresh  =  0  # Forced intra MBs per picture
InterSearch16x16      =  1  # Inter block search 16x16 (0=disable, 1=enable)
InterSearch16x8       =  1  # Inter block search 16x8  (0=disable, 1=enable)
InterSearch8x16       =  1  # Inter block search  8x16 (0=disable, 1=enable)
InterSearch8x8        =  1  # Inter block search  8x8  (0=disable, 1=enable)
InterSearch8x4        =  1  # Inter block search  8x4  (0=disable, 1=enable)
InterSearch4x8        =  1  # Inter block search  4x8  (0=disable, 1=enable)
InterSearch4x4        =  1  # Inter block search  4x4  (0=disable, 1=enable)

IntraDisableInterOnly  = 0  # Apply Disabling Intra conditions only to Inter Slices
(0:disable/default,1: enable)
Intra4x4ParDisable     = 0  # Disable Vertical & Horizontal 4x4
Intra4x4DiagDisable    = 0  # Disable Diagonal 45degree 4x4
Intra4x4DirDisable     = 0  # Disable Other Diagonal 4x4
Intra16x16ParDisable   = 0  # Disable Vertical & Horizontal 16x16
Intra16x16PlaneDisable = 0  # Disable Planar 16x16
ChromaIntraDisable     = 0  # Disable Intra Chroma modes other than DC
EnableIPCM             = 0  # Enable IPCM macroblock mode

DisposableP            = 0  # Enable Disposable P slices in the primary layer (0:
disable/default, 1: enable)
DispPQPOffset          = 0  # Quantizer offset for disposable P slices (0: default)

###########################################################################
# B Slices
###########################################################################

NumberBFrames         =  2  # Number of B coded frames inserted (0=not used)
QPBSlice              =  40  # Quant. param for B slices (0-51)
BRefPicQPOffset       =  0  # Quantization offset for reference B coded pictures (-
51..51)
DirectModeType        =  1  # Direct Mode Type (0:Temporal 1:Spatial)
DirectInferenceFlag   =  1  # Direct Inference Flag (0: Disable 1: Enable)
BList0References      =  0  # B slice List 0 reference override (0 disable, N <=
NumberReferenceFrames)
BList1References      =  1  # B slice List 1 reference override (0 disable, N <=
NumberReferenceFrames)
                      # 1 List1 reference is usually recommended for normal GOP
Structures.
                      # A larger value is usually more appropriate if a more flexible
```

```
                    # structure is used (i.e. using PyramidCoding)

BReferencePictures  =  1  # Referenced B coded pictures (0=off, 1=on)

PyramidCoding       =  0  # B pyramid (0= off, 1= 2 layers, 2= 2 full pyramid, 3 =
explicit)
PyramidLevelQPEnable   =  0  # Adjust QP based on Pyramid Level (in increments of
1). Overrides BRefPicQPOffset behavior.(0=off, 1=on)
ExplicitPyramidFormat  = "b2r28b0e30b1e30b3e30b4e30" # Explicit Enhancement
GOP. Format is {FrameDisplay_orderReferenceQP}.
                                    # Valid values for reference type is r:reference, e:non
reference.
PyramidRefReorder     =  1  # Reorder References according to Poc distance for
PyramidCoding (0=off, 1=enable)
PocMemoryManagement   =  1  # Memory management based on Poc Distances for
PyramidCoding (0=off, 1=on)

BiPredMotionEstimation = 0   # Enable Bipredictive based Motion Estimation
(0:disabled, 1:enabled)
BiPredMERefinements    = 0   # Bipredictive ME extra refinements (0: single, N: N
extra refinements (1 default)
BiPredMESearchRange    = 16  # Bipredictive ME Search range (8 default). Note that
range is halved for every extra refinement.
BiPredMESubPel        = 0   # Bipredictive ME Subpixel Consideration (0: disabled, 1:
single level, 2: dual level)


###############################################################################
# SP Frames
###############################################################################

SPPicturePeriodicity  =  0  # SP-Picture Periodicity (0=not used)
QPSPSlice           = 28  # Quant. param of SP-Slices for Prediction Error (0-51)
QPSP2Slice          = 27  # Quant. param of SP-Slices for Predicted Blocks (0-51)
SI_FRAMES             =  0  # SI frame encoding flag (0=not used, 1=used)
SP_output           =  0  # Controls whether coefficients will be output to encode
switching SP frames (0=no, 1=yes)
SP_output_name        =  "low_quality.dat" # Filename for SP output coefficients
SP2_FRAMES            =  0  # switching SP frame encoding flag (0=not used, 1=used)
SP2_input_name1       = "high_quality.dat" # Filename for the first swithed bitstream
coefficients
SP2_input_name2       = "low_quality.dat"  # Filename for the second switched
bitstream coefficients
###############################################################################
# Output Control, NALs
###############################################################################

SymbolMode            =  1  # Symbol mode (Entropy coding method: 0=UVLC,
1=CABAC)
OutFileMode           =  0  # Output file mode, 0:Annex B, 1:RTP
```

```
PartitionMode        =  0  # Partition Mode, 0: no DP, 1: 3 Partitions per Slice

##############################################################################
# CABAC context initialization
##############################################################################

ContextInitMethod      =  1     # Context init (0: fixed, 1: adaptive)
FixedModelNumber       =  0     # model number for fixed decision for inter slices ( 0, 1, or 2 )

##############################################################################
# Interlace Handling
##############################################################################

PicInterlace           =  0     # Picture AFF    (0: frame coding, 1: field coding, 2:adaptive frame/field coding)
MbInterlace            =  0     # Macroblock AFF (0: frame coding, 1: field coding, 2:adaptive frame/field coding)
IntraBottom            =  0     # Force Intra Bottom at GOP Period

##############################################################################
# Weighted Prediction
##############################################################################
WeightedPrediction     =  0     # P picture Weighted Prediction (0=off, 1=explicit mode)
WeightedBiprediction   =  0     # B picture Weighted Prediciton (0=off, 1=explicit mode,  2=implicit mode)
UseWeightedReferenceME  =  0     # Use weighted reference for ME (0=off, 1=on)

##############################################################################
# Picture based Multi-pass encoding
##############################################################################

RDPictureDecision      =  0     # Perform RD optimal decision between different coded picture versions.
                                # If GenerateMultiplePPS is enabled then this will test different WP methods.
                                # Otherwise it will test QP +-1 (0: disabled, 1: enabled)
RDPictureIntra         =  0     # Perform RD optimal decision also for intra coded pictures (0: disabled (default), 1: enabled).
RDPSliceWeightOnly     =  0     # Only consider Weighted Prediction for P slices in Picture RD decision. (0: disabled, 1: enabled (default))
RDBSliceWeightOnly     =  0     # Only consider Weighted Prediction for B slices in Picture RD decision. (0: disabled (default), 1: enabled )

##############################################################################
# Loop filter parameters
##############################################################################
```

```
LoopFilterParametersFlag = 0      # Configure loop filter (0=parameter below ingored,
1=parameters sent)
LoopFilterDisable        = 0      # Disable loop filter in slice header (0=Filter, 1=No
Filter)
LoopFilterAlphaC0Offset  = 0      # Alpha & C0 offset div. 2, {-6, -5, ... 0, +1, .. +6}
LoopFilterBetaOffset     = 0      # Beta offset div. 2, {-6, -5, ... 0, +1, .. +6}

##############################################################################
# Error Resilience / Slices
##############################################################################

SliceMode            =  0   # Slice mode (0=off 1=fixed #mb in slice 2=fixed #bytes in
slice 3=use callback)
SliceArgument        = 50   # Slice argument (Arguments to modes 1 and 2 above)

num_slice_groups_minus1 = 0  # Number of Slice Groups Minus 1, 0 == no FMO, 1 ==
two slice groups, etc.
slice_group_map_type          = 0  # 0:  Interleave, 1: Dispersed,    2: Foreground with
left-over,
                      # 3:  Box-out,    4: Raster Scan   5: Wipe
                      # 6:  Explicit, slice_group_id read from SliceGroupConfigFileName
slice_group_change_direction_flag = 0    # 0: box-out clockwise, raster scan or wipe
right,
                            # 1: box-out counter clockwise, reverse raster scan or wipe left
slice_group_change_rate_minus1    = 85   #
SliceGroupConfigFileName          = "sg0conf.cfg"   # Used for slice_group_map_type 0,
2, 6

UseRedundantSlice     = 0    # 0: not used, 1: one redundant slice used for each slice
(other modes not supported yet)

##############################################################################
# Search Range Restriction / RD Optimization
##############################################################################

RestrictSearchRange  =  2  # restriction for (0: blocks and ref, 1: ref, 2: no restrictions)
RDOptimization       =  1  # rd-optimized mode decision (0:off, 1:on, 2: with losses)
                    # 0: RD-off (Low complexity mode)
                    # 1: RD-on (High complexity mode)
                    # 2: RD-on (Fast high complexity mode - not work in FREX Profiles)
                    # 3: with losses
DisableThresholding  =  0  # Disable Thresholding of Transform Coefficients (0:off,
1:on)
DisableBSkipRDO      =  0  # Disable B Skip Mode consideration from RDO Mode
decision (0:off, 1:on)
SkipIntraInInterSlices   =  1 # Skips Intra mode checking in inter slices if certain mode
decisions are satisfied (0: off, 1: on)

# Explicit Lambda Usage
```

```
UseExplicitLambdaParams  =  0  # Use explicit lambda scaling parameters (0:disabled, 1:enabled)
LambdaWeightIslice       =  0.65 # scaling param for I slices. This will be used as a multiplier i.e. lambda=LambdaWeightISlice * 2^((QP-12)/3)
LambdaWeightPslice       =  0.68 # scaling param for P slices. This will be used as a multiplier i.e. lambda=LambdaWeightPSlice * 2^((QP-12)/3)
LambdaWeightBslice       =  2.00 # scaling param for B slices. This will be used as a multiplier i.e. lambda=LambdaWeightBSlice * 2^((QP-12)/3)
LambdaWeightRefBslice    =  1.50 # scaling param for Referenced B slices. This will be used as a multiplier i.e. lambda=LambdaWeightRefBSlice * 2^((QP-12)/3)
LambdaWeightSPslice      =  1.50 # scaling param for SP slices. This will be used as a multiplier i.e. lambda=LambdaWeightSPSlice * 2^((QP-12)/3)
LambdaWeightSIslice      =  0.65 # scaling param for SI slices. This will be used as a multiplier i.e. lambda=LambdaWeightSISlice * 2^((QP-12)/3)

LossRateA           = 10  # expected packet loss rate of the channel for the first partition, only valid if RDOptimization = 2
LossRateB           =  0  # expected packet loss rate of the channel for the second partition, only valid if RDOptimization = 2
LossRateC           =  0  # expected packet loss rate of the channel for the third partition, only valid if RDOptimization = 2
NumberOfDecoders    = 30  # Numbers of decoders used to simulate the channel, only valid if RDOptimization = 2
RestrictRefFrames   =  0  # Doesnt allow reference to areas that have been intra updated in a later frame.

###############################################################################
# Additional Stuff
###############################################################################

UseConstrainedIntraPred  =  0  # If 1, Inter pixels are not used for Intra macroblock prediction.
LastFrameNumber          =  0  # Last frame number that have to be coded (0: no effect)
ChangeQPI                = 16  # QP (I-slices)  for second part of sequence (0-51)
ChangeQPP                = 16  # QP (P-slices)  for second part of sequence (0-51)
ChangeQPB                = 18  # QP (B-slices)  for second part of sequence (0-51)
ChangeQPBSRefOffset      =  2  # QP offset (stored B-slices)  for second part of sequence (-51..51)
ChangeQPStart            =  0  # Frame no. for second part of sequence (0: no second part)

NumberofLeakyBuckets     =  8                  # Number of Leaky Bucket values
LeakyBucketRateFile      =  "leakybucketrate.cfg"  # File from which encoder derives rate values
LeakyBucketParamFile     =  "leakybucketparam.cfg" # File where encoder stores leakybucketparams

NumberFramesInEnhancementLayerSubSequence  = 0  # number of frames in the Enhanced Scalability Layer(0: no Enhanced Layer)
NumberOfFrameInSecondIGOP                  = 0  # Number of frames to be coded in the second IGOP
```

```
SparePictureOption       =  0   # (0: no spare picture info, 1: spare picture available)
SparePictureDetectionThr  =  6   # Threshold for spare reference pictures detection
SparePicturePercentageThr = 92   # Threshold for the spare macroblock percentage

PicOrderCntType          = 0    # (0: POC mode 0, 1: POC mode 1, 2: POC mode 2)

###################################################################################
#Rate control
###################################################################################

RateControlEnable    =     0   # 0 Disable, 1 Enable
Bitrate              =  45020   # Bitrate(bps)
InitialQP            =     24   # Initial Quantization Parameter for the first I frame
                           # InitialQp depends on two values: Bits Per Picture,
                           # and the GOP length
BasicUnit            =     11   # Number of MBs in the basic unit
                           # should be a fractor of the total number
                           # of MBs in a frame
ChannelType          =      0   # type of channel( 1=time varying channel; 0=Constant
channel)

###################################################################################
#Fast Mode Decision
###################################################################################
EarlySkipEnable      =      0   # Early skip detection (0: Disable 1: Enable)
SelectiveIntraEnable =      0   # Selective Intra mode decision (0: Disable 1: Enable)

###################################################################################
#FREXT stuff
###################################################################################

YUVFormat            = 1      # YUV format (0=4:0:0, 1=4:2:0, 2=4:2:2, 3=4:4:4)
RGBInput             = 0      # 1=RGB input, 0=GBR or YUV input
BitDepthLuma         = 8      # Bit Depth for Luminance (8...12 bits)
BitDepthChroma       = 8      # Bit Depth for Chrominance (8...12 bits)
CbQPOffset           = 0      # Chroma QP offset for Cb-part (-51..51)
CrQPOffset           = 0      # Chroma QP offset for Cr-part (-51..51)
Transform8x8Mode     = 0      # (0 : only 4x4 transform, 1: allow using 8x8 transform
additionally, 2: only 8x8 transform)
ResidueTransformFlag  = 0     # (0: no residue color transform 1: apply residue color
transform)
ReportFrameStats     = 0      # (0:Disable Frame Statistics 1: Enable)
DisplayEncParams     = 0      # (0:Disable Display of Encoder Params 1: Enable)
Verbose              = 1      # level of display verboseness (0:short, 1:normal, 2:detailed)

###################################################################################
#Q-Matrix (FREXT)
###################################################################################
QmatrixFile          = "q_matrix.cfg"
```

```
ScalingMatrixPresentFlag = 0    # Enable Q_Matrix  (0 Not present, 1 Present in SPS, 2
Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag0  = 3    # Intra4x4_Luma    (0 Not present, 1 Present in SPS, 2
Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag1  = 3    # Intra4x4_ChromaU (0 Not present, 1 Present in SPS, 2
Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag2  = 3    # Intra4x4_chromaV (0 Not present, 1 Present in SPS, 2
Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag3  = 3    # Inter4x4_Luma    (0 Not present, 1 Present in SPS, 2
Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag4  = 3    # Inter4x4_ChromaU (0 Not present, 1 Present in SPS, 2
Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag5  = 3    # Inter4x4_ChromaV (0 Not present, 1 Present in SPS, 2
Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag6  = 3    # Intra8x8_Luma    (0 Not present, 1 Present in SPS, 2
Present in PPS, 3 Present in both SPS & PPS)
ScalingListPresentFlag7  = 3    # Inter8x8_Luma    (0 Not present, 1 Present in SPS, 2
Present in PPS, 3 Present in both SPS & PPS)

###########################################################################
#Rounding Offset control
###########################################################################

OffsetMatrixPresentFlag  = 0    # Enable Explicit Offset Quantization Matrices  (0:
disable 1: enable)
QOffsetMatrixFile        = "q_offset.cfg" # Explicit Quantization Matrices file

AdaptiveRounding         = 0    # Enable Adaptive Rounding based on JVT-N011 (0:
disable, 1: enable)
AdaptRndPeriod           = 1    # Period in terms of MBs for updating rounding offsets.
                        # 0 performs update at the picture level. Default is 16. 1 is as in
JVT-N011.
AdaptRndChroma           = 0    # Enables coefficient rounding adaptation for chroma

AdaptRndWFactorIRef      = 4    # Adaptive Rounding Weight for I/SI slices in
reference pictures /4096
AdaptRndWFactorPRef      = 4    # Adaptive Rounding Weight for P/SP slices in
reference pictures /4096
AdaptRndWFactorBRef      = 4    # Adaptive Rounding Weight for B slices in reference
pictures /4096
AdaptRndWFactorINRef     = 4    # Adaptive Rounding Weight for I/SI slices in non
reference pictures /4096
AdaptRndWFactorPNRef     = 4    # Adaptive Rounding Weight for P/SP slices in non
reference pictures /4096
AdaptRndWFactorBNRef     = 4    # Adaptive Rounding Weight for B slices in non
reference pictures /4096

###########################################################################
#Lossless Coding (FREXT)
###########################################################################
```

```
QPPrimeYZeroTransformBypassFlag = 0    # Enable lossless coding when qpprime_y
is zero (0 Disabled, 1 Enabled)

###########################################################################
#Fast Motion Estimation Control Parameters
###########################################################################

UseFME                = 0    # Use fast motion estimation (0=disable/default,
1=UMHexagonS,
                      # 2=Simplified UMHexagonS, 3=EPZS patterns)

EPZSPattern           = 2    # Select EPZS primary refinement pattern.
                      # (0: small diamond, 1: square, 2: extended diamond/default,
                                                          # 3: large diamond)
EPZSDualRefinement    = 3    # Enables secondary refinement pattern.
                      # (0:disabled, 1: small diamond, 2: square,
                      # 3: extended diamond/default, 4: large diamond)
EPZSFixedPredictors   = 2    # Enables Window based predictors
                      # (0:disabled, 1: P only, 2: P and B/default)
EPZSTemporal          = 0    # Enables temporal predictors
                                  # (0: disabled, 1: enabled/default)
EPZSSpatialMem        = 0    # Enables spatial memory predictors
                                  # (0: disabled, 1: enabled/default)
EPZSMinThresScale     = 0    # Scaler for EPZS minimum threshold (0 default).
                      # Increasing value can speed up encoding.
EPZSMedThresScale     = 0    # Scaler for EPZS median threshold (1 default).
                      # Increasing value can speed up encoding.
EPZSMaxThresScale     = 0    # Scaler for EPZS maximum threshold (1 default).
                      # Increasing value can speed up encoding.
```

Anexo B Script utilizado en las pruebas

```
# ------------------------------------------------------------------------------
# Descripción:
# Genera una batería de bitstreams haciendo un barrido de parámetros
# ------------------------------------------------------------------------------


DirectorioFicheros='c://videos//'


# --------------------------------
# Ficheros que se van a codificar -> "InputFile"

BarridoFicheros='Football Mobile Coastguard Paris Container Tempete Akiyo News'


# --------------------------------
# QP que se van a probar. Estos valores se pondrán igual para
# la QP de I, P y B
# QPFirstFrame, QPRemainingFrame, QPBPicture

BarridoQP='5 10 15 20 25 30 35 40 45 50'



# IP  IntraPeriod=0 FrameSkip=0 NumberBFrames=0# --------------------------------
# Especificamos I,P,B: -> intraperiod, FrameSkip, NumberBFrames
# Especifica qué tipo de tira hay en el bitstream
# I   IntraPeriod=1 FrameSkip=0 NumberBFrames=0
# I1B IntraPeriod=0 FrameSkip=1 NumberBFrames=1
# I2B IntraPeriod=0 FrameSkip=2 NumberBFrames=2
# I3B IntraPeriod=0 FrameSkip=3 NumberBFrames=3

BarridoTipoCuadro='IP I2B I3B'


# --------------------------------
# Especificamos restricciones de tamaños de bloque
# Grupo1: solo 16x16
# Grupo2: solo 16x16 y 8x8
# Grupo3: solo 16x16, 8x8 y 4x4
# Grupo4: solo 16x16, 16x8, 8x16, 8x8
# Grupo5: todos los modos

BarridoPrediccion='Grupo5'

# --------------------------------
#         CODIFICACION
```

```
# ------------------------------
for strFichero in $BarridoFicheros; do
for QP in $BarridoQP; do
for bTipoCuadro in $BarridoTipoCuadro; do
for Restriccion in $BarridoPrediccion; do

        strFicheroRuta=$DirectorioFicheros$strFichero'.cif'
        strParam1=' -p InputFile='$strFicheroRuta
        strParam2=' -p QPISlice='$QP' -p QPPSlice='$QP' -p QPBSlice='$QP
        if [ "$bTipoCuadro" = 'I' ]; then
                strParam3=' -p IntraPeriod=1 -p FrameSkip=0 -p    NumberBFrames=0 -
p FramesToBeEncoded=100'
        else if [ "$bTipoCuadro" = 'IP' ]; then
                strParam3=' -p IntraPeriod=30 -p FrameSkip=0 -p  NumberBFrames=0 -
p FramesToBeEncoded=100'
        else if [ "$bTipoCuadro" = 'I1B' ]; then
                strParam3=' -p IntraPeriod=15 -p FrameSkip=1 -p  NumberBFrames=1 -
p FramesToBeEncoded=50'
        else if [ "$bTipoCuadro" = 'I2B' ]; then
                strParam3=' -p IntraPeriod=10 -p FrameSkip=2 -p  NumberBFrames=2 -
p FramesToBeEncoded=33'
        else if [ "$bTipoCuadro" = 'I3B' ]; then
                strParam3=' -p IntraPeriod=8 -p FrameSkip=3 -p    NumberBFrames=3 -
p FramesToBeEncoded=25'
        else
                echo "Tipo cuadro desconocido"
                continue
        fi
        fi
        fi
        fi
        fi

        if [ "$Restriccion" = 'Grupo1' ]; then
                strParam4=' -p InterSearch16x16=1'
        else if [ "$Restriccion" = 'Grupo2' ]; then
                strParam4=' -p InterSearch16x16=1 -p InterSearch8x8=1'
        else if [ "$Restriccion" = 'Grupo3' ]; then
                strParam4=' -p InterSearch16x16=1 -p InterSearch16x8=1 -p
InterSearch8x16=1'
        else if [ "$Restriccion" = 'Grupo4' ]; then
                strParam4=' -p InterSearch16x16=1 -p InterSearch16x8=1 -p
InterSearch8x16=1 -p InterSearch8x8=1'
        else if [ "$Restriccion" = 'Grupo5' ]; then
                strParam4=' -p InterSearch16x16=1 -p InterSearch16x8=1 -p
InterSearch8x16=1 -p InterSearch8x8=1 -p InterSearch8x4=1 -p InterSearch4x8=1 -p
InterSearch4x4=1'
        else
                echo "Tipo restriccion desconocida"
                continue
```

```
        fi
        fi
        fi
        fi
        fi

        strOUTPUT=$strFichero'_QP'$QP'_'$bTipoCuadro'_'$Restriccion

        strParamGlobal=$strParam1$strParam2$strParam3$strParam4

        # Volcado de parámetros de codificador en fichero

        echo $strOUTPUT

        rm -f data.out
        ./lencod.exe -f encoder.cfg $strParamGlobal > $strOUTPUT'.1.OUT'
        cat data.out >> 'resultados\'$strFichero'.out'

done
done
done
done
```

Anexo C Descripción de los vídeos utilizados

Trabajaremos con los vídeos de prueba típicos en el ámbito de la codificación de vídeo. De este modo podremos compararnos con otros autores siempre en igualdad de condiciones.

A continuación realizaremos una breve descripción de las secuencias de vídeo utilizadas:

- **Akiyo**

Figura C.1. Secuencia Akiyo.

Se trata de una secuencia en la que una mujer está presentando el telediario. El único movimiento presente es el de su cara. Se trata de uno de los vídeos más fáciles de codificar.

- **Bridge-far**

Figura C.2. Secuencia Bridge-far.

Se trata de un puente con la cámara fija a bastante distancia. En este caso el movimiento será principalmente el del agua.

- **Bus**

Figura C.3. Secuencia Bus.

En este vídeo la cámara se centra en un autobús que va circulando por la ciudad. Durante la secuencia se ven varios coches aparcados y cómo una furgoneta intenta adelantar al autobús. Por otro lado, en la parte superior de la imagen se ven las copas de numerosos árboles, lo que supone una zona con gran cantidad de detalles difícil de codificar.

- **Carphone**

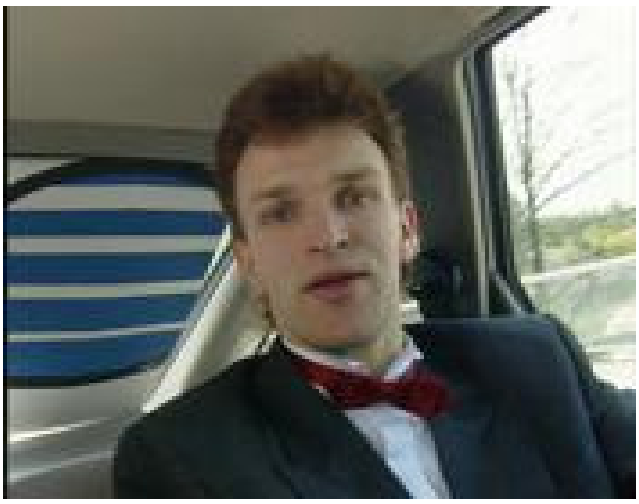


Figura C.4. Secuencia Carphone.

En esta secuencia aparece un hombre hablando, con gran expresividad en la cara, y algún movimiento brusco. Sus manos aparecen y desaparecen la imagen, complicando la codificación. Además tendremos el movimiento del paisaje en la ventanilla.

- **Claire**

Figura C.5. Secuencia Claire.

Se trata de una mujer hablando sobre un fondo fijo. El único movimiento es el de su cara y su cabeza, por lo que el resto de la imagen será codificada principalmente con modos *Skip* y *Directo*.

- **Coastguard**

Figura C.6. Secuencia Coastguard.

En esta secuencia la cámara enfoca en primer lugar a una lancha que avanza por el agua de derecha a izquierda. A continuación se centra en una embarcación mayor que se mueve en sentido contrario al de la lancha. La trayectoria de ambas es prácticamente constante a lo largo de todo el vídeo.

- **Container**

Figura C.7. Secuencia Container.

En esta secuencia la cámara se encuentra fija grabando cómo abandona el puerto un barco. Tras él aparece una lancha que lleva un movimiento similar. Por otro lado, hay un mástil con una bandera hondeando al viento, que será una zona difícil de codificar. Al final del vídeo aparecen unos pájaros que vuelan rápidamente y que también complicarán la codificación.

- **Football**

Figura C.8. Secuencia Football.

Se trata de una secuencia de un partido de fútbol americano. Se caracteriza por una gran cantidad de movimientos muy rápidos, por lo que será uno de los vídeos más difíciles de codificar. Por último, destacar que la parte inferior de la imagen es una banda negra, por lo que será una zona en la que predomine el uso del modo *Directo*.

- **Foreman**

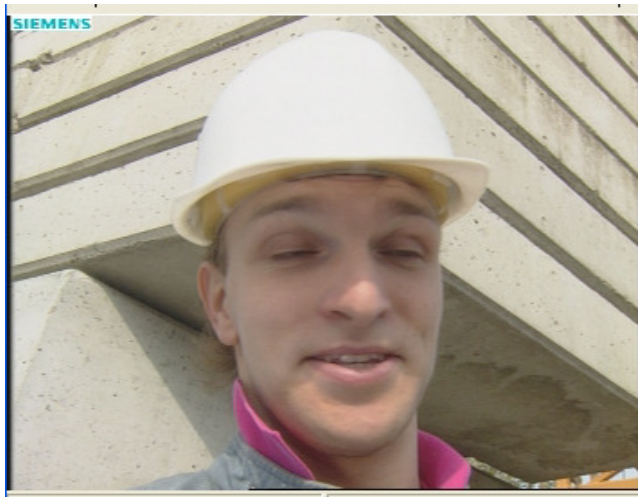


Figura C.9. Secuencia Foreman

Se trata de un vídeo en el que un hombre sostiene la cámara con una mano y se graba a sí mismo. Hacia la mitad de la secuencia realiza un movimiento rápido y pasa a enfocar una zona de obras.

- **Garden**

Figura C.10. Secuencia Garden

Se trata de una secuencia de vídeo en la que la cámara se encuentra en movimiento. En la parte superior de la imagen se observan una serie de viviendas, mientras que en la parte central aparece un gran número de flores. Ésta última zona presenta un gran nivel de detalle, por lo que será difícil de codificar. Por último destacar que, como ocurría en otros vídeos, la parte inferior de la imagen es una banda negra que favorecerá el uso del modo *Directo*.

- **Grandma**

Figura C.11. Secuencia Grandma.

En esta secuencia aparece una señora hablando sobre un fondo fijo. Además no realiza más movimientos que el de su cara y cabeza, por lo que el resto de la imagen permanece prácticamente estático.

- **Highway**

Figura C.12. Secuencia Highway.

Es una secuencia en la que la cámara viaja a bordo de un vehículo y graba la autopista por la que éste circula según va avanzando. El cielo se encuentra repleto de nubes y de vez en cuando se ve algún coche en sentido contrario o incluso adelantándonos.

- **Miss America**

Figura C.13. Secuencia Miss-America.

Se trata de una secuencia en la que aparece una chica hablando. El fondo es estático, y la chica realiza movimientos de cabeza al hablar, y mueve un poco el cuerpo, pero sin movimientos bruscos.

- **Mobile**

Figura C.14. Secuencia Mobile.

Es una secuencia en la que un tren de juguete avanza empujando una pequeña pelota. Además aparece otro objeto que presenta un movimiento oscilante y un calendario que va ascendiendo conforme avanza el vídeo. Tiene, por tanto, varios elementos que pueden suponer un problema en la codificación. Por último señalar, que al igual que ocurría en "Football", la parte inferior de la imagen es una banda negra, por lo que predominará el uso del modo *Directo*.

- **Mother_daughter**

Figura C.15. Secuencia Mother_daughter

Se trata de un vídeo en el que aparecen sentadas una madre y su hija. Apenas existe movimiento salvo el de la cabeza de la madre. Por su parte, la niña permanece la mayor parte del tiempo quieta. Por último destacar que el fondo de escena es bastante sencillo. Se trata de una pared lisa con un cuadro colgado.

- **News**

Figura C.16. Secuencia News.

Se trata de una secuencia en la que un hombre y una mujer están presentando el telediario. Apenas existe movimiento en el primer plano, pero en el fondo de escena se observa una grabación de un par de personas practicando ballet. Será esta parte la que presente una mayor dificultad en el proceso de codificación.

- **Paris**

Figura C.17. Secuencia Paris.

Se trata de un vídeo sin apenas movimiento en el que hay dos personas charlando. El hombre se dedica a hacer girar un bolígrafo entre sus dedos y la mujer no para de tirarse una pelota de una mano a otra. Por último hay que destacar que el fondo tiene gran cantidad de detalles.

- **Salesman**

Figura C.18. Secuencia Salesman

Es una secuencia en la que aparece un hombre describiendo un objeto. Para ello realiza movimientos continuos con sus brazos y el objeto. El fondo es estático, pero con mucho detalle.

- **Silent**

Figura C.19. Secuencia Silent.

Es un vídeo en el que aparece una mujer sentada utilizando el lenguaje de signos. El fondo es estático y el único movimiento presente es el de los brazos y el de la cabeza de dicha mujer.

- **Stefan**

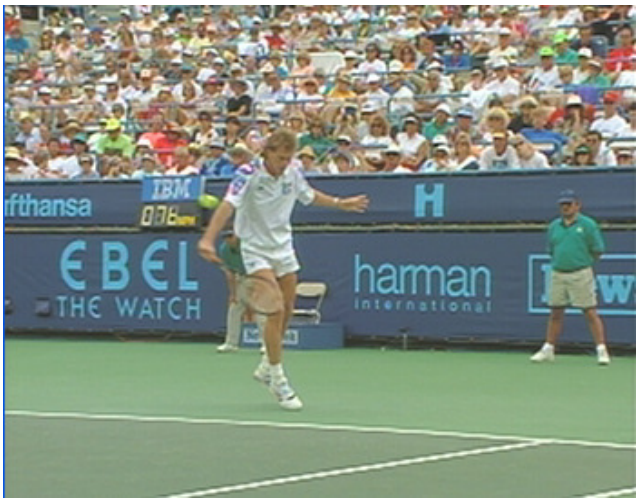


Figura C.20. Secuencia Stefan.

Es una secuencia en la que se ve a un jugador de tenis golpeando a la pelota en el fondo de la pista. Cabe señalar que el fondo de escena tiene un elevado nivel de detalle y que se encuentra en movimiento constante, por lo que será difícil de codificar.

- **Suzie**

Figura C.21. Secuencia Suzie.

En esta secuencia aparece una mujer hablando por teléfono en primer plano, realizando movimientos de cabeza bastante bruscos. Además el teléfono sale y vuelve a entrar en el vídeo.

- **Tempete**

Figura C.22. Secuencia Tempete

En esta secuencia se muestran unas flores a medida que la cámara se va alejando. Hay que destacar que aparece la caída de un gran número de hojas secas con un movimiento bastante caótico e impredecible.

- **Waterfall**

Figura C.23. Secuencia Waterfall

Se trata de un vídeo en el que aparece un paisaje muy colorido, con una catarata en medio. La cámara se aleja lentamente. Hay poco movimiento, salvo en la cascada, y el introducido por el movimiento de la cámara.

Anexo D Consideración sobre resultados.

A la hora de evaluar los resultados, se ha de tener en cuenta que los modos Intra son calculados tantas veces como formas de predicción Intra de la crominancia estén disponibles, y siempre obteniendo el mismo resultado en la predicción de la luminancia. Es decir, que aunque siendo independientes la predicción de la luminancia y la de la crominancia, en el código aparecen como ligadas. Este es un error en el codificador, que se ha corregido en versiones posteriores, y que hará que el cálculo de modos Intra cobre más importancia.

En la siguiente gráfica se muestra el porcentaje en tiempo empleado en el cálculo del coste RD para cada uno de los modos:

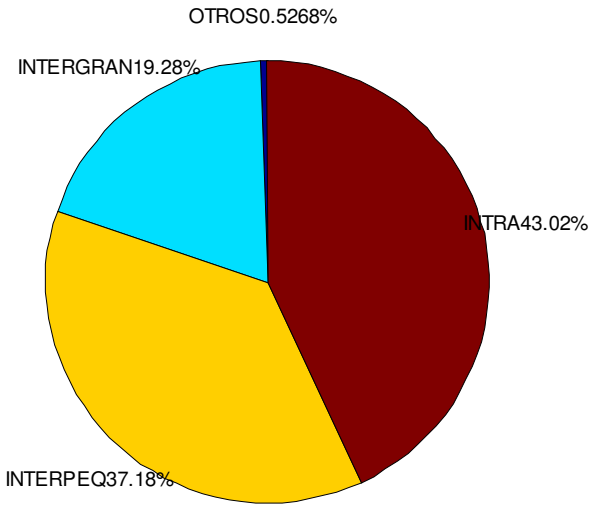


Figura D.1 Porcentaje de tiempo utilizado para el cálculo del coste RD en cada modo

Los resultados de esta gráfica se encuentran desvirtuados por el hecho comentado anteriormente. El cálculo de los modos Intra se repetirá hasta 4 veces, una por cada modo de predicción de la luminancia, siendo esto innecesario. Para presentar datos más realistas, se ha realizado un cálculo aproximado del tiempo empleado en estas repeticiones, eliminándolo del análisis, y el resultado se muestra en la siguiente gráfica:

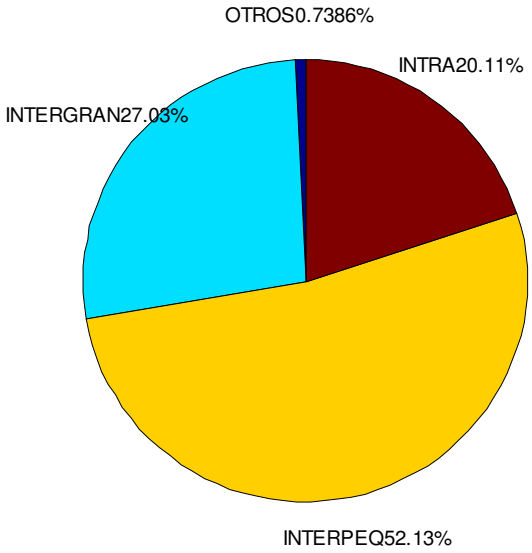


Figura D.2 Porcentaje corregido de tiempo utilizado para el cálculo del coste RD en cada modo

Estas figuras servirán además para mostrar que el tiempo utilizado para el cálculo de los modos *Skip* y *Directo* (OTROS en la figura D.2) será muy pequeño comparado con el resto de los modos. Además los Inter pequeños, al tener que realizar un gran número de estimaciones de movimiento, serán los más costosos computacionalmente, por lo que se ha intentado evitar su cálculo en todo momento durante el diseño del algoritmo.

Anexo E Resultados para todas las secuencias analizadas

CIF IP

Secuencia	Algoritmo	% Tiempo	%Tasa	PSNR
Akiyo (QCIF)	[1]	44,39	-0,14	0
	Decisor 1	76,79	0,35	-0,023
	Decisor 2	80,51	0,30	-0,013
Bus (CIF)	[1]	12,76	0,21	-0,010
	Decisor 1	20,81	0,58	-0,030
	Decisor 2	38,59	1,97	-0,088
Coastguard (CIF)	[1]	11,80	0,07	0
	Decisor 1	22,09	0,45	-0,013
	Decisor 2	43,10	0,45	-0,017
Container (CIF)	[1]	45,26	-0,56	0,018
	Decisor 1	74,24	0,27	-0,014
	Decisor 2	77,42	0,52	-0,026
Football (CIF)	[1]	10,60	0,04	-0,003
	Decisor 1	18,11	0,46	-0,023
	Decisor 2	28,49	0,81	-0,036
Foreman (CIF)	[1]	16,64	-0,04	0
	Decisor 1	37,42	1,29	-0,066
	Decisor 2	51,70	1,73	-0,074
Garden (CIF)	[1]	7,222	0,24	-0,007
	Decisor 1	9,291	0,16	-0,009
	Decisor 2	18,03	0,63	-0,024
Highway (CIF)	[1]	27,38	0,16	-0,009
	Decisor 1	57,46	2,18	-0,071
	Decisor 2	64,16	2,31	-0,079
Mobile (CIF)	[1]	7,49	-0,02	-0,002
	Decisor 1	11,61	0,50	-0,016
	Decisor 2	26,04	1,20	-0,052
Mother_daughter (CIF)	[1]	34,30	-0,19	0,016
	Decisor 1	65,24	0,23	-0,005
	Decisor 2	73,72	0,04	-0,003
News (CIF)	[1]	42,80	-0,01	0,002
	Decisor 1	69,78	0,84	-0,053
	Decisor 2	73,07	1,89	-0,097
Paris (CIF)	[1]	31,57	0,16	-0,012
	Decisor 1	50,86	0,91	-0,052
	Decisor 2	55,12	1,34	-0,063
Silent (CIF)	[1]	38,15	-0,10	0,002
	Decisor 1	65,57	1,07	-0,045
	Decisor 2	71,43	1,19	-0,048
Stefan (CIF)	[1]	12,49	-0,15	0,009
	Decisor 1	23,15	0,53	-0,018
	Decisor 2	30,57	2,04	-0,080

Tempete (CIF)	[1]	9,70	0,18	-0,008
	Decisor 1	20,85	1,04	-0,039
	Decisor 2	31,18	1,09	-0,043
Waterfall (CIF)	[1]	21,87	-0,14	0,011
	Decisor 1	52,42	0,47	-0,014
	Decisor 2	52,99	0,40	-0,011
Media	**[1]**	**23.40**	**-0.02**	**0**
	Decisor 1	**42.23**	**0.71**	**-0.03**
	Decisor 2	**51.01**	**1.12**	**-0.04**

Tabla E.1 Resultados totales para resolución CIF y patrón IP

QCIF IP

Secuencia	Algoritmo	% Tiempo	%Tasa	PSNR
Akiyo (QCIF)	[1]	44,79	-0,033	-0,001
	Decisor 1	78,57	0,486	-0,028
	Decisor 2	80,45	0,545	-0,035
Bridge-far (QCIF)	[1]	61,78	0,072	-0,003
	Decisor 1	93,81	0,817	-0,050
	Decisor 2	93,83	0,970	-0,058
Carphone (QCIF)	[1]	17,84	0,132	-0,002
	Decisor 1	39,07	0,696	-0,036
	Decisor 2	51,31	1,402	-0,070
Claire (QCIF)	[1]	44,95	0,471	-0,022
	Decisor 1	75,02	0,611	-0,039
	Decisor 2	75,93	0,425	-0,028
Coastguard (QCIF)	[1]	14,65	-0,009	0,001
	Decisor 1	25,99	-0,032	0,005
	Decisor 2	45,33	0,417	-0,016
Container (QCIF)	[1]	52,21	-0,352	0,015
	Decisor 1	76,35	0,008	0,003
	Decisor 2	77,11	0,130	0,003
Foreman (QCIF)	[1]	12,77	-0,030	0,009
	Decisor 1	27,11	1,047	-0,046
	Decisor 2	41,25	1,148	-0,052
Grandma (QCIF)	[1]	46,34	0,037	0,003
	Decisor 1	76,35	0,398	-0,023
	Decisor 2	77,93	0,438	-0,021
Miss-America (QCIF)	[1]	34,89	-0,516	0,020
	Decisor 1	66,15	-0,068	0,003
	Decisor 2	68,40	0,144	-0,003
Mobile (QCIF)	[1]	5,95	-0,220	0,012
	Decisor 1	9,29	0,253	-0,005
	Decisor 2	18,39	0,907	-0,037

Mother-daughter (QCIF)	[1]	34,61	-0,003	0,014
	Decisor 1	62,90	0,423	-0,004
	Decisor 2	68,99	0,100	0,001
News (QCIF)	[1]	44,61	0,071	-0,010
	Decisor 1	68,00	0,705	-0,042
	Decisor 2	69,88	1,364	-0,072
Salesman (QCIF)	[1]	47,15	-0,097	0,001
	Decisor 1	74,65	0,307	-0,017
	Decisor 2	76,30	0,275	-0,018
Silent (QCIF)	[1]	35,20	0,122	-0,018
	Decisor 1	60,31	0,918	-0,052
	Decisor 2	64,44	1,505	-0,071
Suzie (QCIF)	[1]	18,01	-0,011	-0,002
	Decisor 1	38,07	0,305	-0,020
	Decisor 2	52,77	0,620	-0,023
Media	**[1]**	**34.38**	**-0.024**	**0.001**
	Decisor 1	**58.11**	**0.458**	**-0.023**
	Decisor 2	**64.16**	**0.693**	**-0.033**

Tabla E.2 Resultados totales para resolución QCIF y patrón IP

CIF I2B

Secuencia	Algoritmo	% Tiempo	%Tasa	PSNR
Akiyo (QCIF)	[2]	73,08	0,41	-0,021
	Decisor 1	78,25	0,82	-0,041
	Decisor 2	79,81	0,87	-0,050
Bus (CIF)	[2]	29,04	3,25	-0,137
	Decisor 1	26,10	0,42	-0,029
	Decisor 2	40,02	1,51	-0,080
Coastguard (CIF)	[2]	35,31	2,96	-0,078
	Decisor 1	41,97	0,65	-0,017
	Decisor 2	49,28	0,42	-0,019
Container (CIF)	[2]	74,94	0,37	-0,020
	Decisor 1	73,32	1,18	-0,029
	Decisor 2	74,10	1,41	-0,044
Football (CIF)	[2]	21,66	1,58	-0,070
	Decisor 1	18,37	0,46	-0,028
	Decisor 2	29,63	1,16	-0,065
Foreman (CIF)	[2]	40,74	5,75	-0,237
	Decisor 1	41,87	1,40	-0,066
	Decisor 2	52,28	1,61	-0,070
Garden (CIF)	[2]	18,82	2,62	-0,111
	Decisor 1	15,17	0,32	-0,023
	Decisor 2	26,70	0,73	-0,053

Highway (CIF)	[2]	53,27	3,85	-0,111
	Decisor 1	56,16	1,21	-0,039
	Decisor 2	63,08	1,34	-0,042
Mobile (CIF)	[2]	24,19	1,47	-0,049
	Decisor 1	21,77	0,93	-0,061
	Decisor 2	28,35	1,30	-0,082
Mother_daughter (CIF)	[2]	64,63	0,51	-0,036
	Decisor 1	67,22	0,98	-0,046
	Decisor 2	70,42	1,02	-0,046
News (CIF)	[2]	66,25	1,27	-0,065
	Decisor 1	67,62	2,44	-0,104
	Decisor 2	71,01	2,80	-0,128
Paris (CIF)	[2]	49,74	1,33	-0,067
	Decisor 1	48,21	1,11	-0,056
	Decisor 2	54,87	1,58	-0,081
Silent (CIF)	[2]	62,65	0,78	-0,030
	Decisor 1	63,69	1,06	-0,050
	Decisor 2	67,97	1,50	-0,069
Stefan (CIF)	[2]	27,16	2,70	-0,107
	Decisor 1	30,19	0,98	-0,049
	Decisor 2	41,59	1,53	-0,073
Tempete (CIF)	[2]	33,22	3,04	-0,109
	Decisor 1	33,19	0,81	-0,040
	Decisor 2	42,69	1,17	-0,052
Waterfall (CIF)	[2]	57,29	1,29	-0,080
	Decisor 1	53,01	0,11	0,001
	Decisor 2	56,32	0,41	-0,008
Media	**[2]**	**45.75**	**2.0791**	**-0.084**
	Decisor 1	**46.01**	**0.9352**	**-0.042**
	Decisor 2	**53.01**	**1.2760**	**-0.060**

Tabla E.3 Resultados totales para resolución CIF y patrón I2B

QCIF I2B

Secuencia	Algoritmo	% Tiempo	%Tasa	PSNR
Akiyo (QCIF)	[2]	74,65	-0,53	0,039
	Decisor 1	77,31	0,24	-0,022
	Decisor 2	78,63	0,27	-0,026
Bridge-far (QCIF)	[2]	84,11	-0,07	-0,002
	Decisor 1	89,54	0,34	-0,027
	Decisor 2	89,49	0,34	-0,027
Carphone (QCIF)	[2]	44,62	4,02	-0,220
	Decisor 1	42,74	1,62	-0,076
	Decisor 2	53,52	2,14	-0,110

Claire (QCIF)	[2]	72,27	-0,30	0,026
	Decisor 1	76,60	0,19	-0,012
	Decisor 2	79,18	0,24	-0,016
Coastguard (QCIF)	[2]	43,87	2,76	-0,091
	Decisor 1	41,42	0,20	-0,015
	Decisor 2	48,28	0,79	-0,042
Container (QCIF)	[2]	76,56	-0,17	0,009
	Decisor 1	74,48	0,60	-0,027
	Decisor 2	74,25	0,60	-0,027
Foreman (QCIF)	[2]	37,69	3,50	-0,182
	Decisor 1	35,53	1,13	-0,062
	Decisor 2	47,25	1,29	-0,067
Grandma (QCIF)	[2]	74,18	-0,19	0,006
	Decisor 1	73,37	0,37	-0,013
	Decisor 2	73,99	0,42	-0,015
Miss-America (QCIF)	[2]	63,52	0,61	-0,033
	Decisor 1	69,81	0,43	-0,013
	Decisor 2	72,15	0,56	-0,020
Mobile (QCIF)	[2]	29,19	3,26	-0,138
	Decisor 1	25,37	0,72	-0,040
	Decisor 2	35,21	1,12	-0,059
Mother-daughter (QCIF)	[2]	65,43	-0,36	0,006
	Decisor 1	66,18	0,42	-0,027
	Decisor 2	68,16	0,49	-0,037
News (QCIF)	[2]	67,15	0,30	-0,025
	Decisor 1	64,65	1,35	-0,086
	Decisor 2	67,17	1,47	-0,092
Salesman (QCIF)	[2]	72,90	-0,22	0,008
	Decisor 1	70,57	0,82	-0,042
	Decisor 2	72,02	0,86	-0,040
Silent (QCIF)	[2]	60,20	0,07	0,003
	Decisor 1	59,43	1,42	-0,074
	Decisor 2	63,11	1,84	-0,096
Suzie (QCIF)	[2]	46,41	2,96	-0,123
	Decisor 1	46,17	0,56	-0,022
	Decisor 2	53,30	0,66	-0,023
Media	**[2]**	**60.85**	**1.04**	**-0.048**
	Decisor 1	**60.88**	**0.69**	**-0.037**
	Decisor 2	**65.05**	**0.87**	**-0.047**

Tabla E.4 Resultados totales para resolución QCIF y patrón I2B

Bridge-far QCIF IP

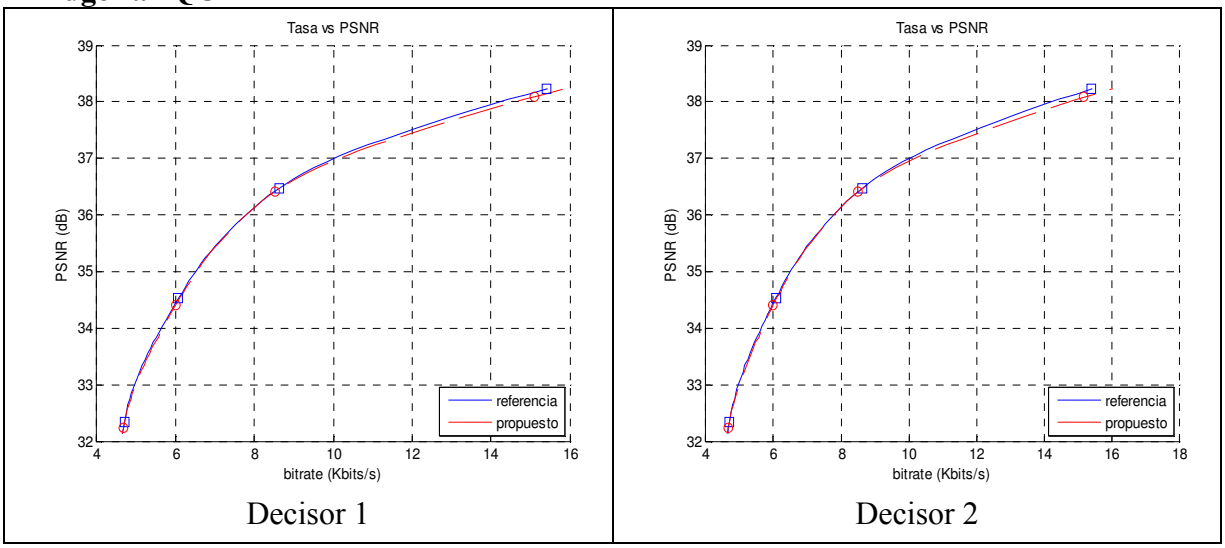


Figura E.1 Gráfica de PSNR vs Tasa Bridge-far QCIF IP

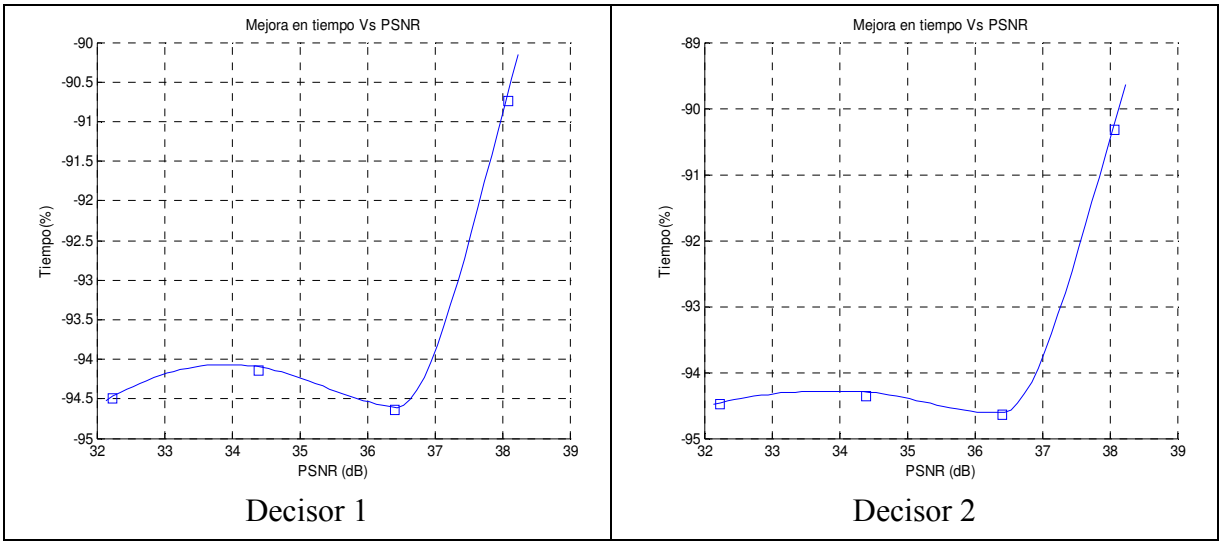


Figura E.2 Gráfica de mejora en tiempo vs PSNR Bridge-far QCIF IP

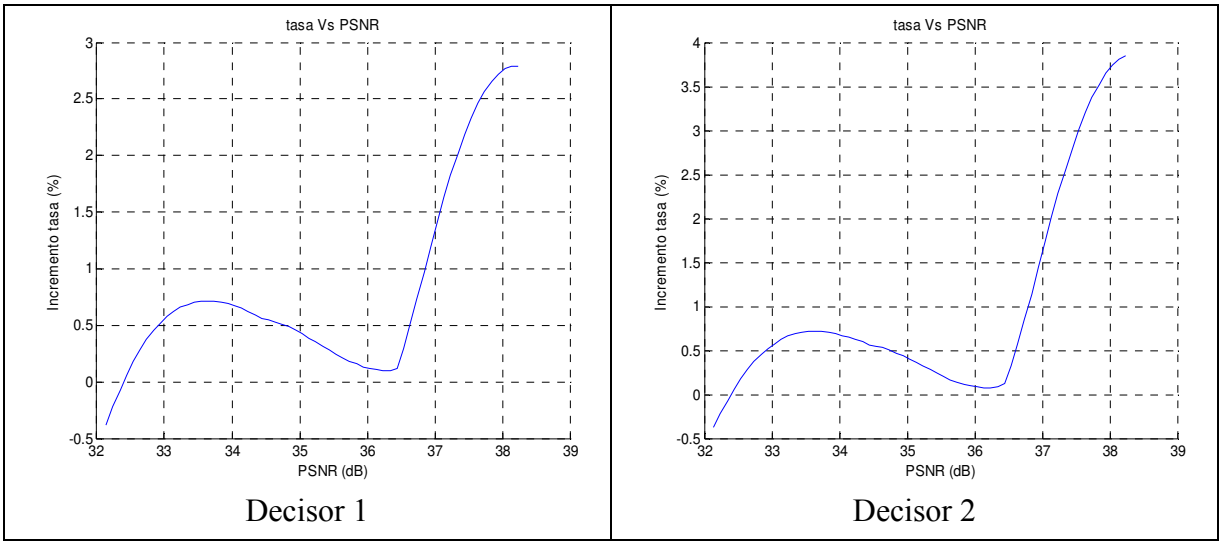


Figura E.3 Gráfica de incremento de tasa vs PSNR Bridge-far QCIF IP

Claire QCIF IP

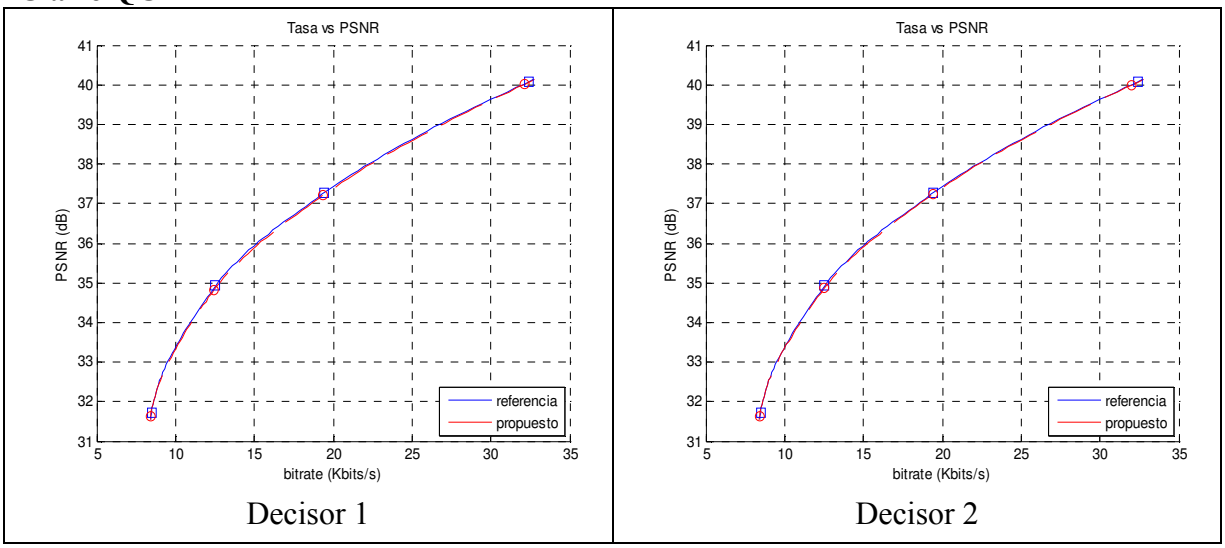


Figura E.4 Gráfica de PSNR vs Tasa Claire QCIF IP

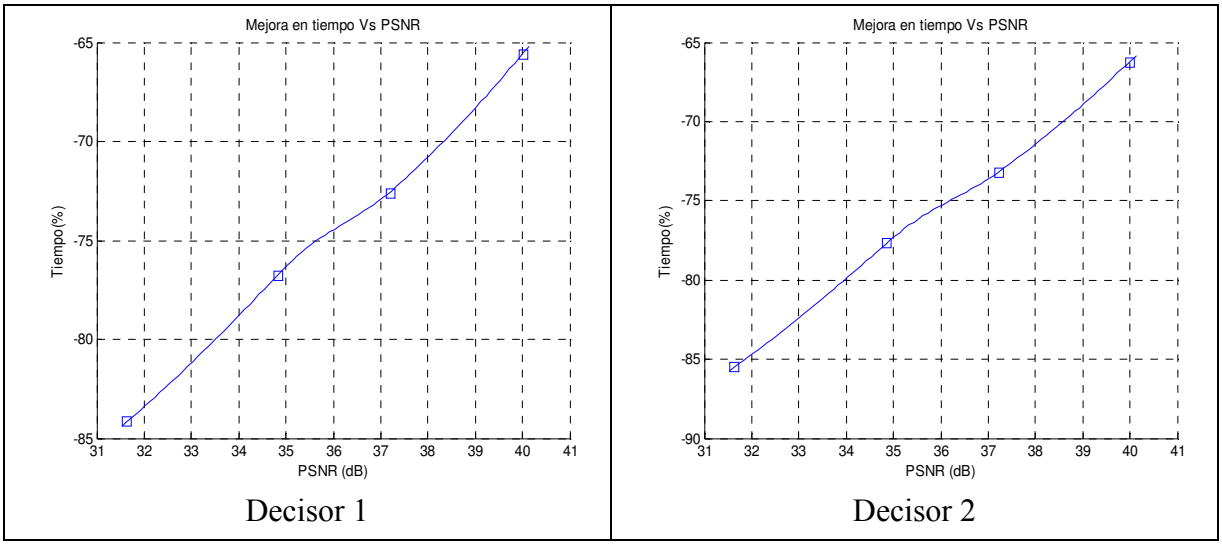


Figura E.5 Gráfica de mejora en tiempo vs PSNR Claire QCIF IP

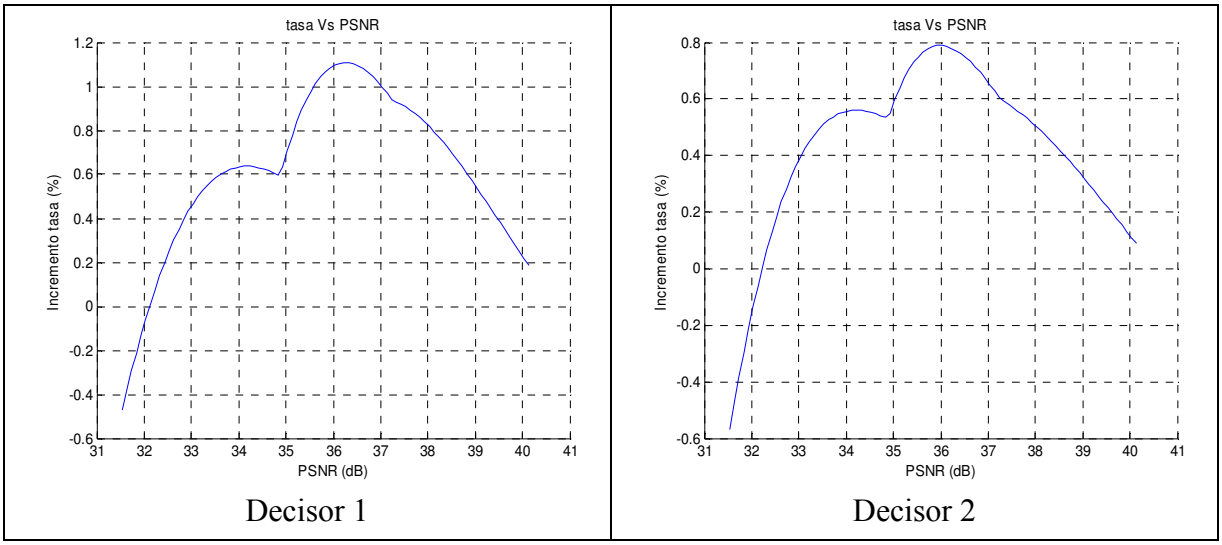


Figura E.6 Gráfica de incremento de tasa vs PSNR Claire QCIF IP

Coastguard QCIF IP

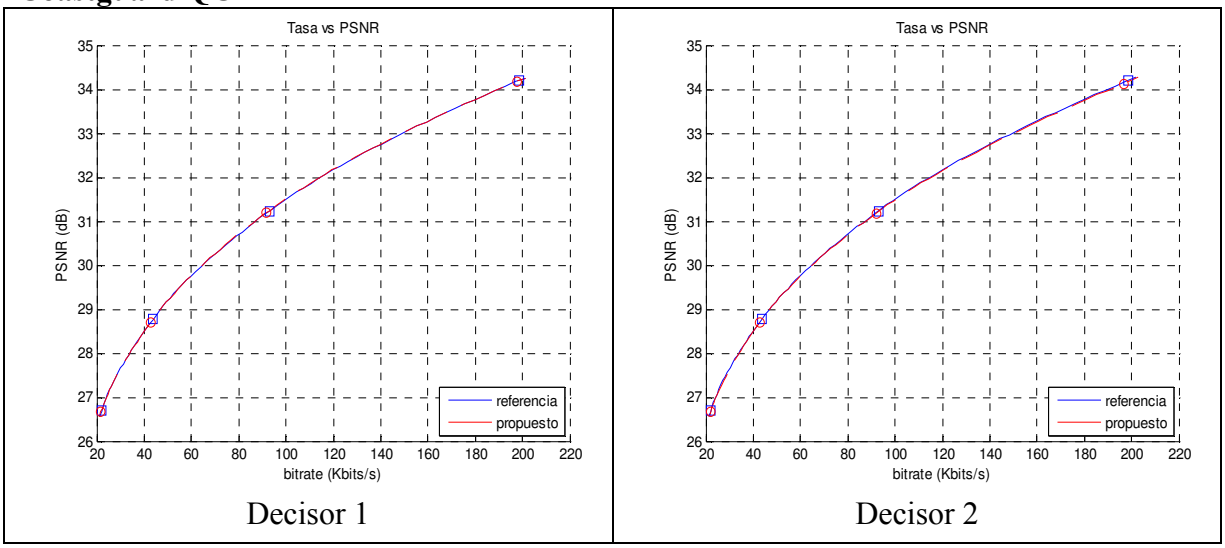


Figura E.7 Gráfica de PSNR vs Tasa Coastguard QCIF IP

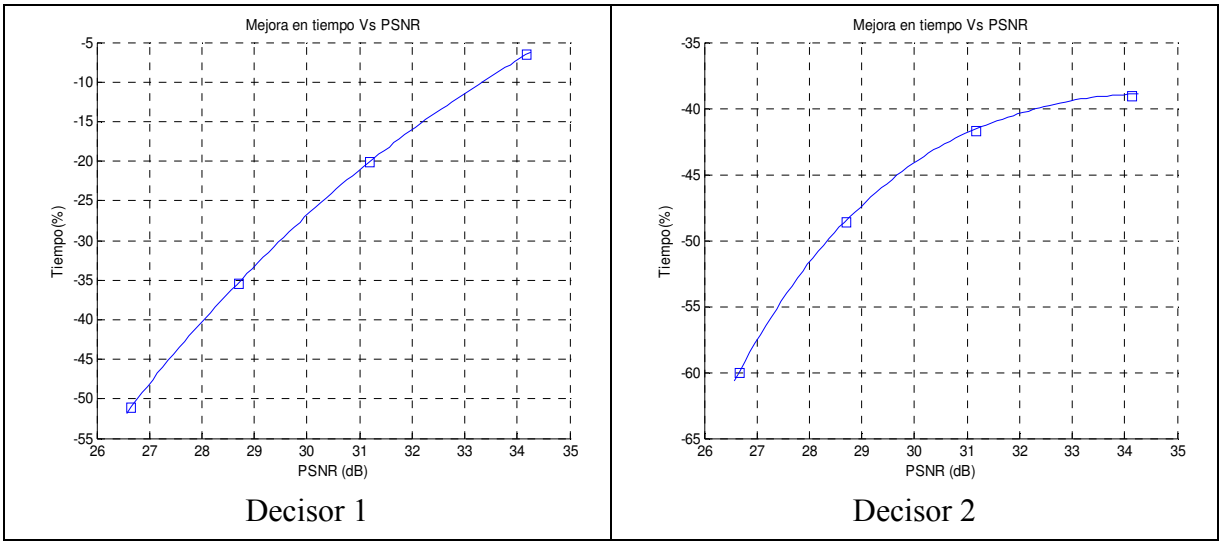


Figura E.8 Gráfica de mejora en tiempo vs PSNR Coastguard QCIF IP

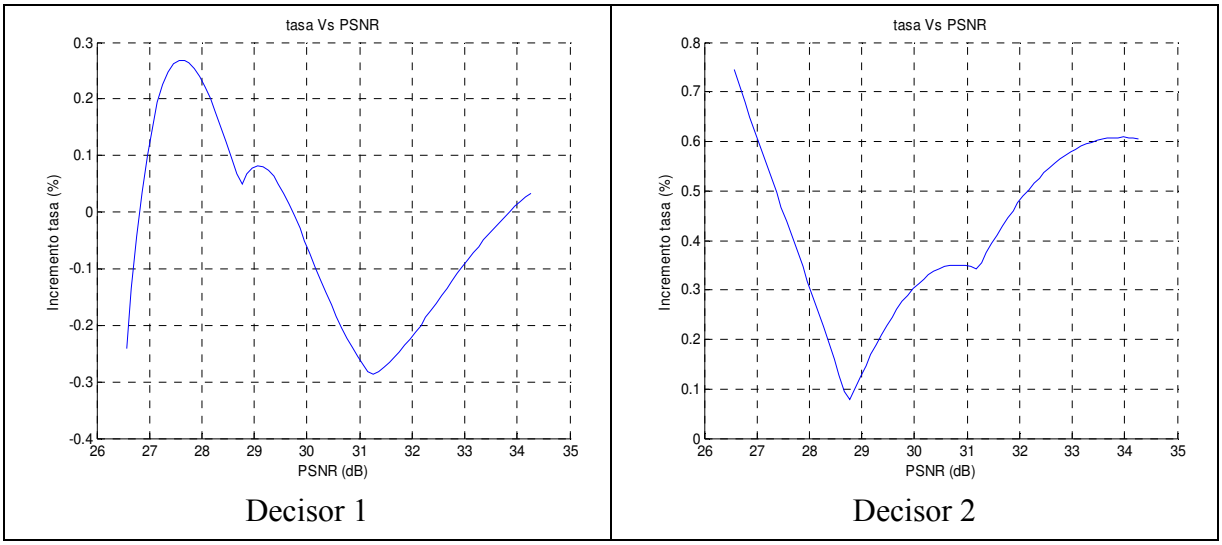


Figura E.9 Gráfica de incremento de tasa vs PSNR Coastguard QCIF IP

Grandma QCIF IP

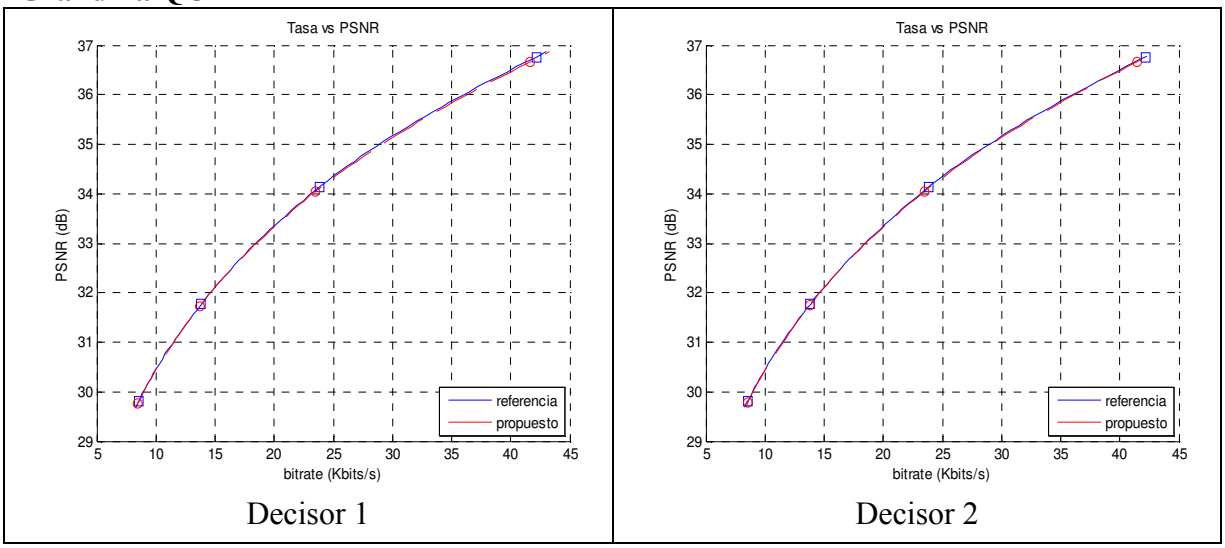


Figura E.10 Gráfica de PSNR vs Tasa Grandma QCIF IP

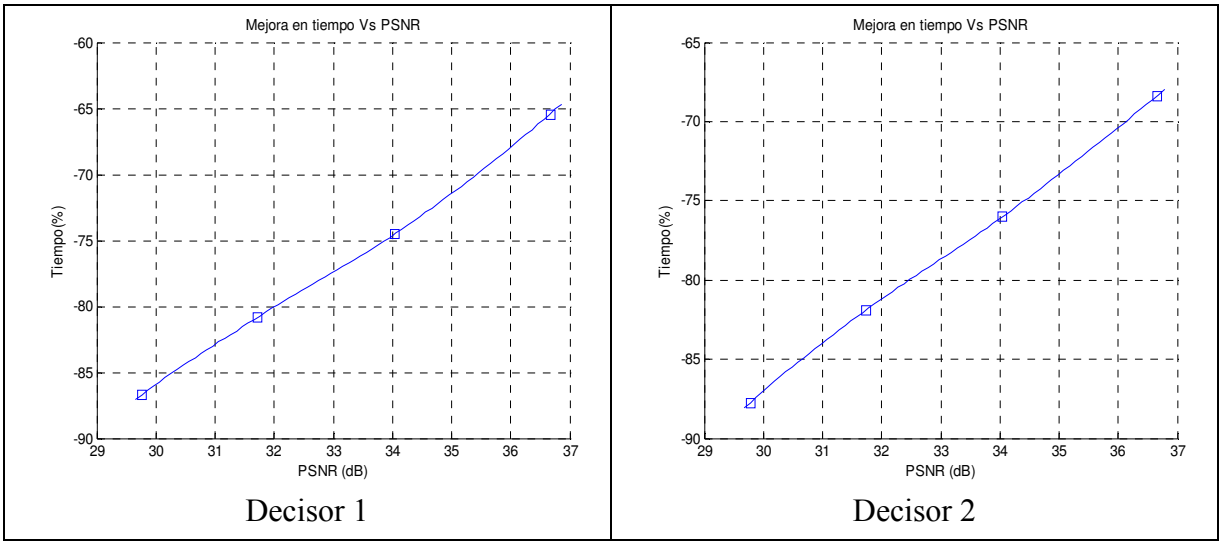


Figura E.11 Gráfica de mejora en tiempo vs PSNR Grandma QCIF IP

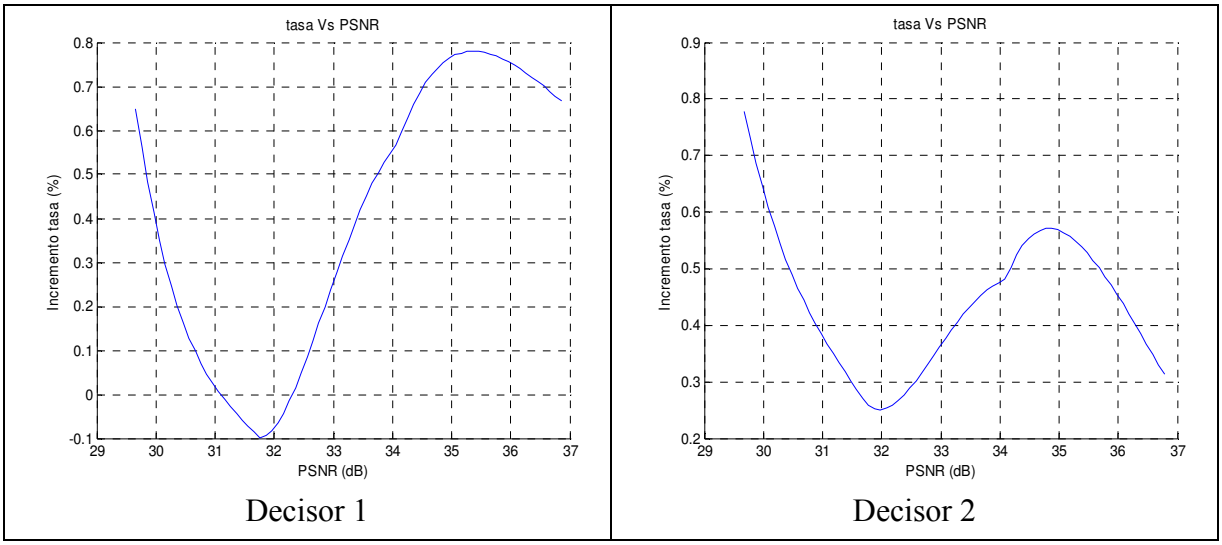


Figura E.12 Gráfica de incremento de tasa vs PSNR Grandma QCIF IP

Miss-america QCIF IP

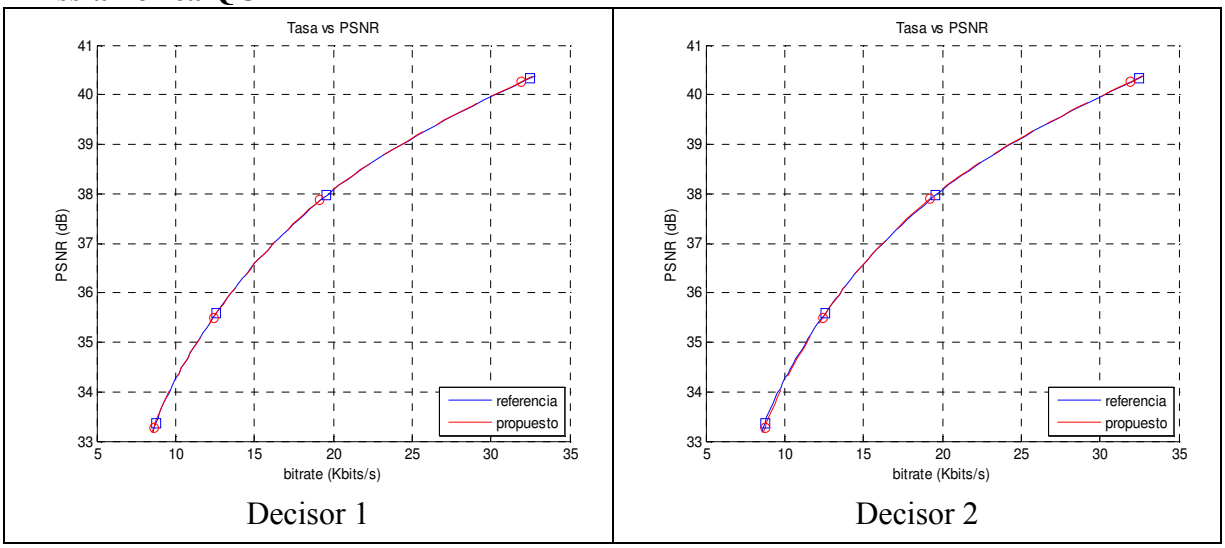


Figura E.13 Gráfica de PSNR vs Tasa Miss-america QCIF IP

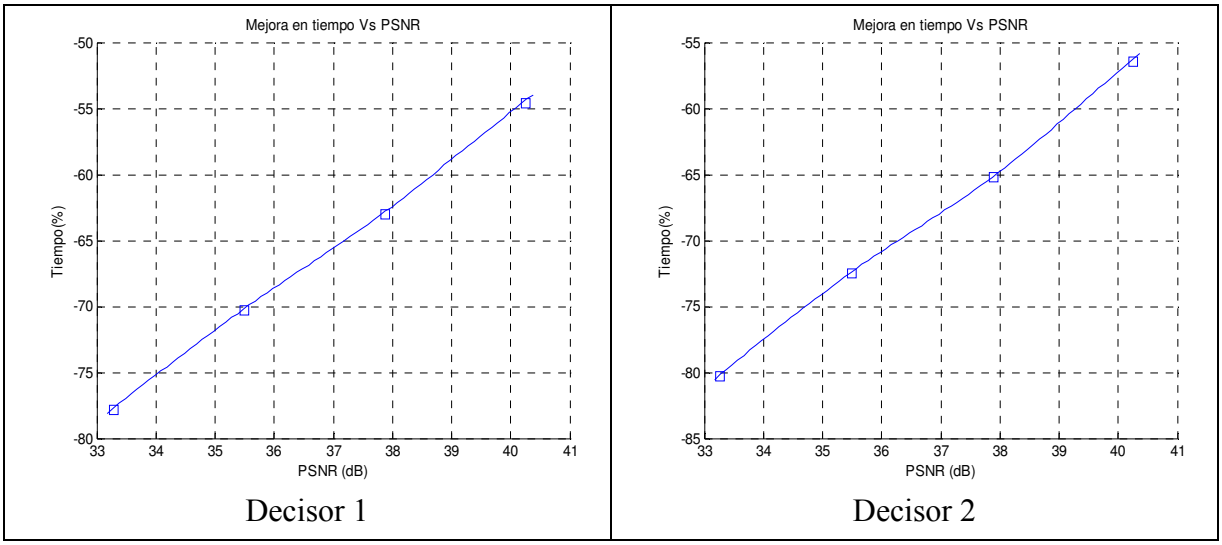


Figura E.14 Gráfica de mejora en tiempo vs PSNR Miss-america QCIF IP

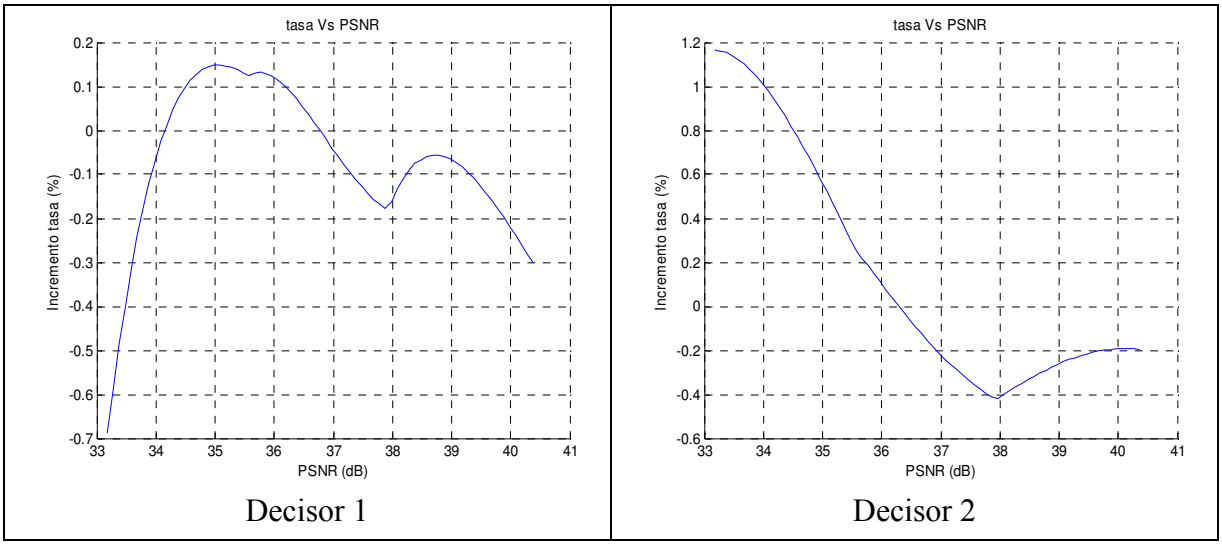


Figura E.15 Gráfica de incremento de tasa vs PSNR Miss-america QCIF IP

Bridge-far QCIF I2B

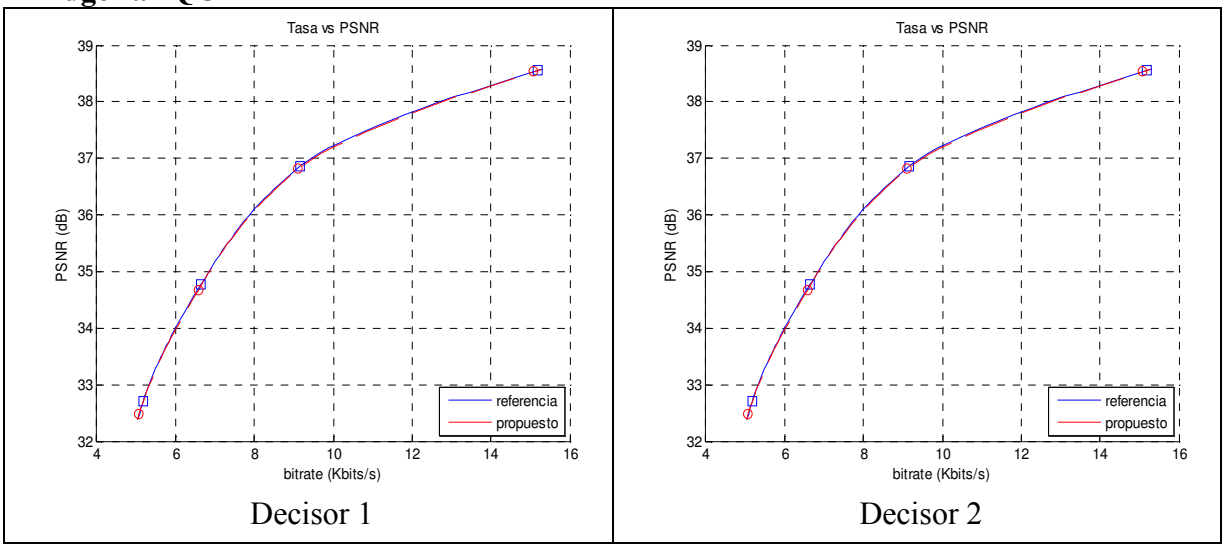


Figura E.16 Gráfica de PSNR vs Tasa Bridge-far QCIF I2B

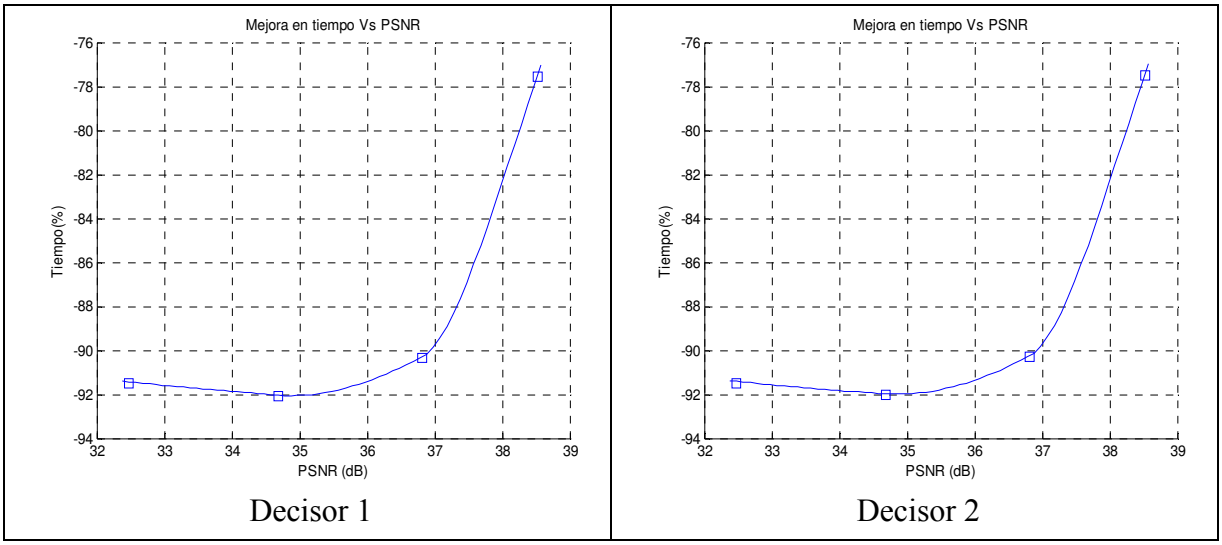


Figura E.17 Gráfica de mejora en tiempo vs PSNR Bridge-far QCIF I2B

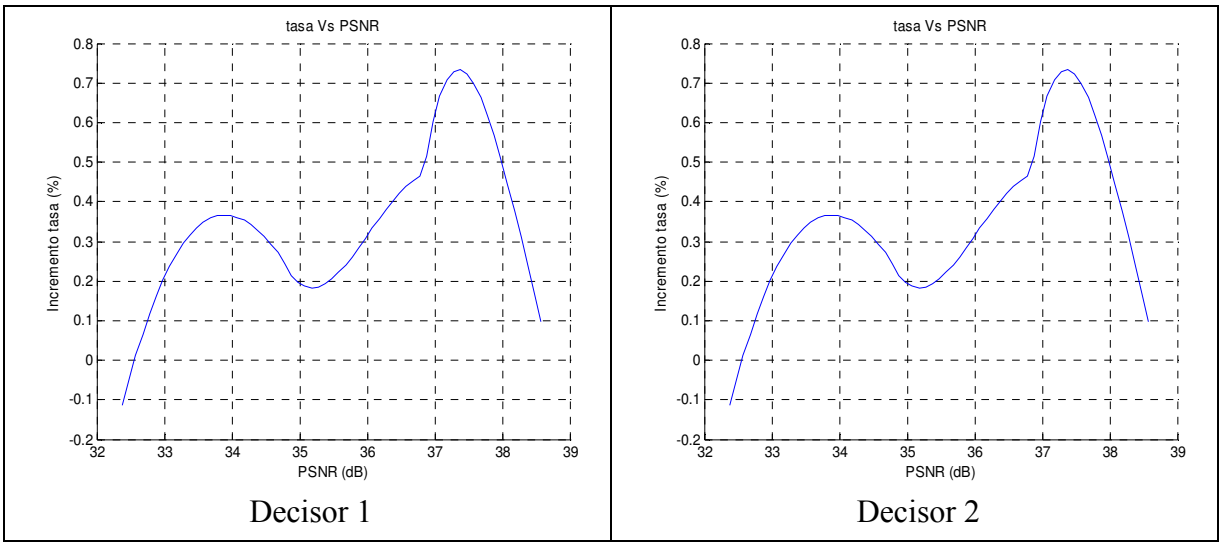


Figura E.18 Gráfica de incremento de tasa vs PSNR Bridge-far QCIF I2B

Claire QCIF I2B

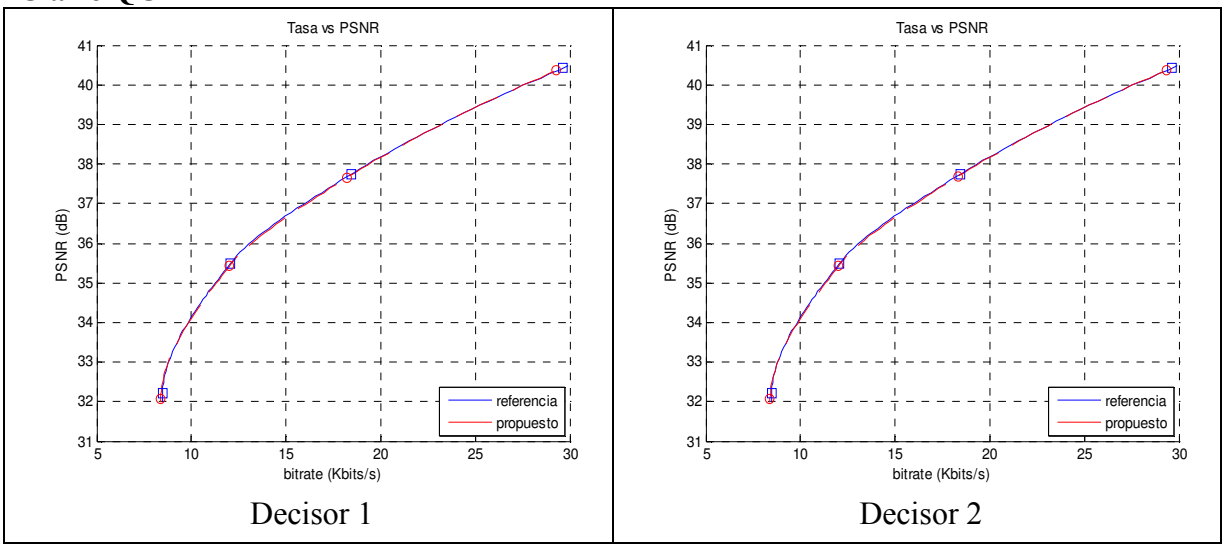


Figura E.19 Gráfica de PSNR vs Tasa Claire QCIF I2B

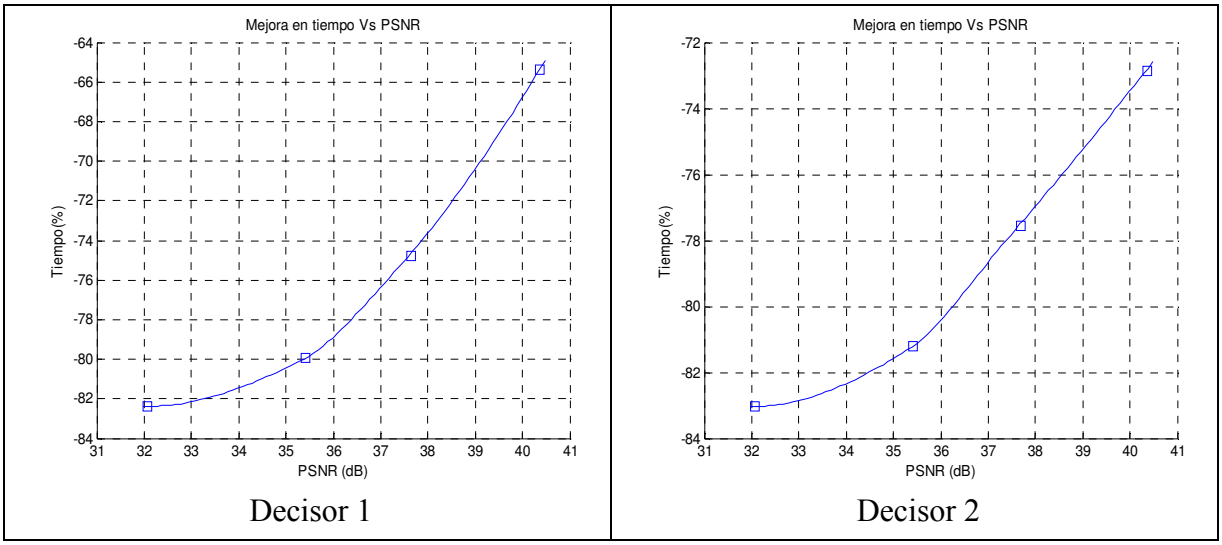


Figura E.20 Gráfica de mejora en tiempo vs PSNR Claire QCIF I2B

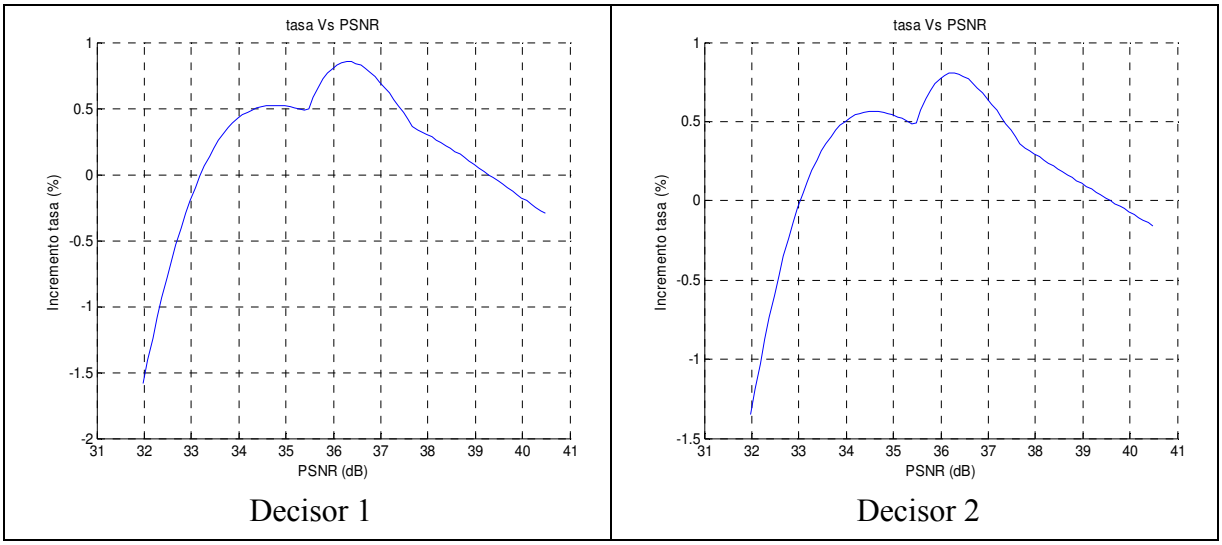


Figura E.21 Gráfica de incremento de tasa vs Claire QCIF I2B

Coastguard QCIF I2B

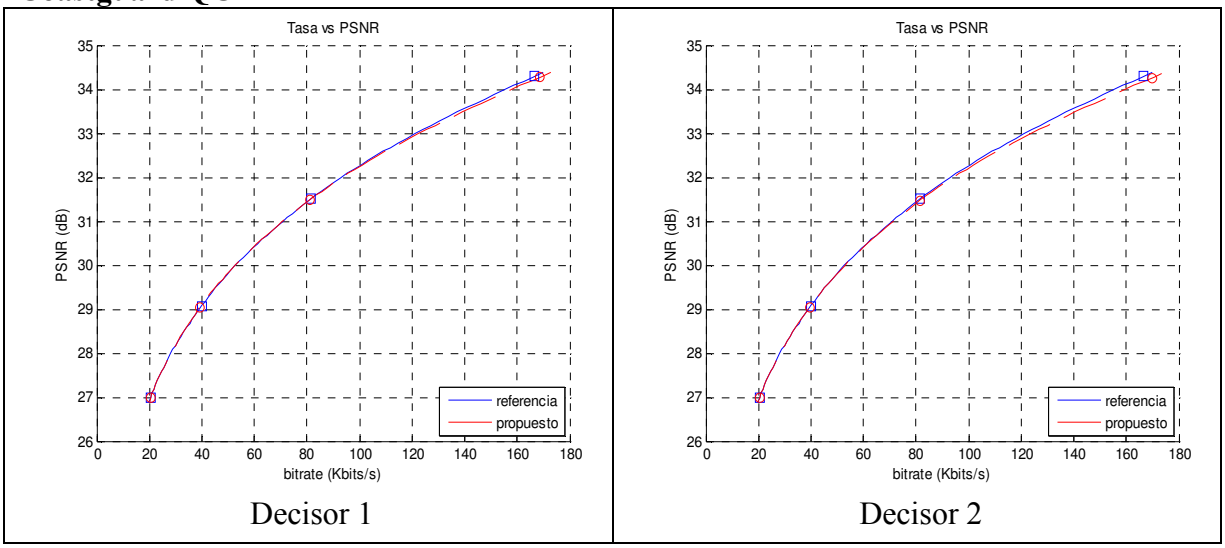


Figura E.22 Gráfica de PSNR vs Tasa Coastguard QCIF I2B

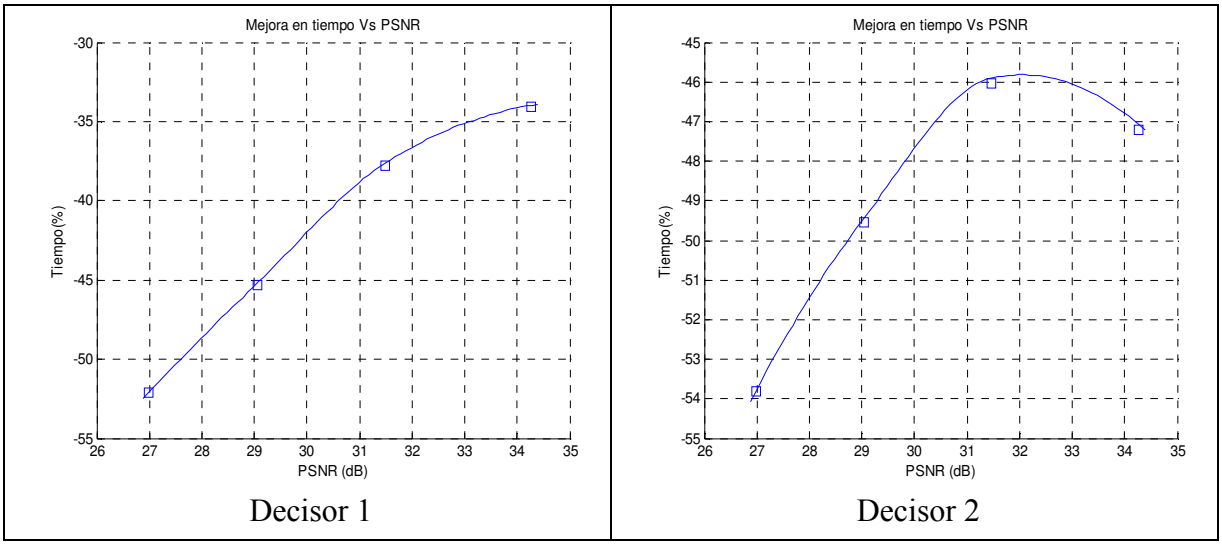


Figura E.23 Gráfica de mejora en tiempo vs PSNR Coastguard QCIF I2B

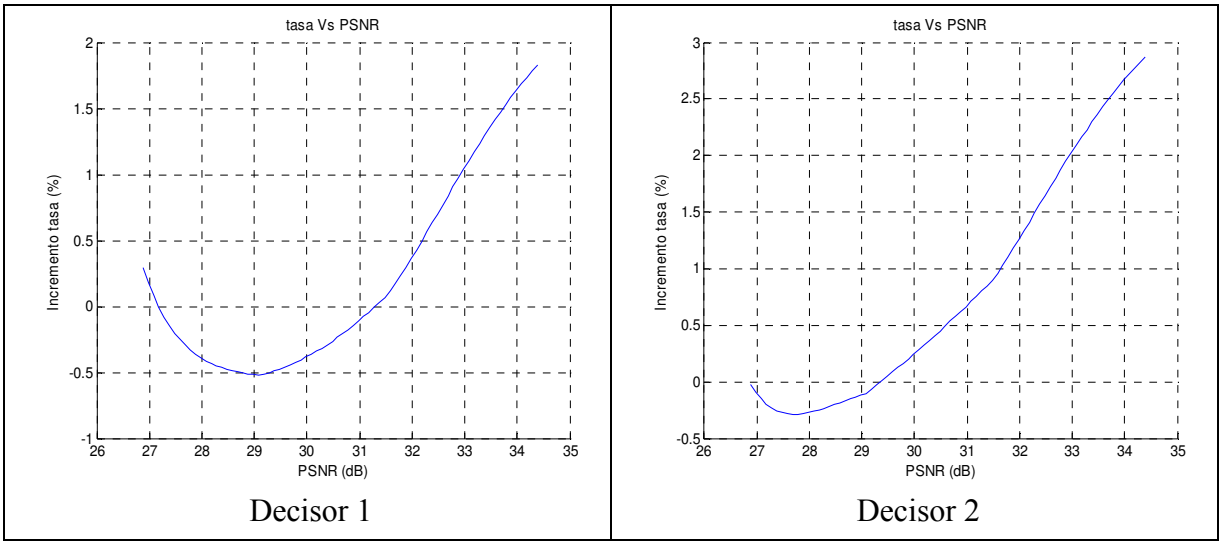


Figura E.24 Gráfica de incremento de tasa vs PSNR Coastguard QCIF I2B

Grandma QCIF I2B

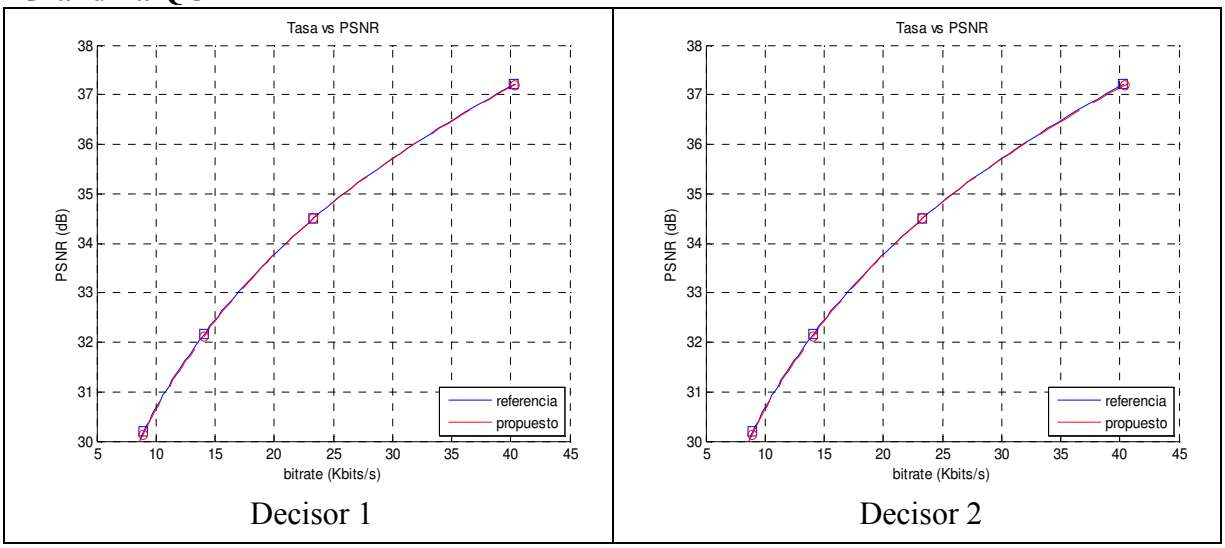


Figura E.25 Gráfica de PSNR vs Tasa Grandma QCIF I2B

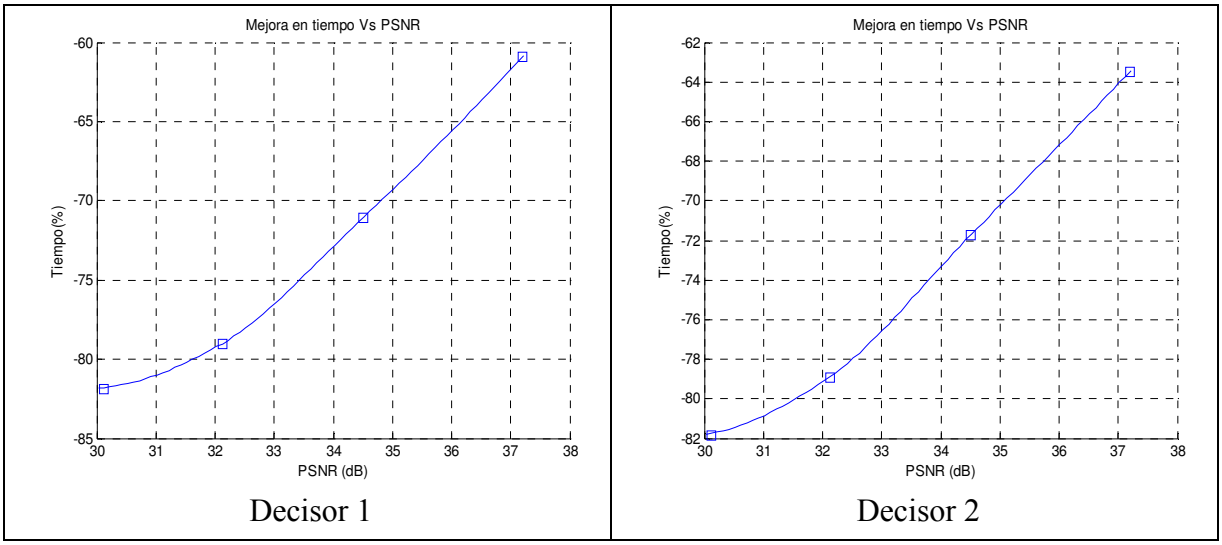


Figura E.26 Gráfica de mejora en tiempo vs PSNR Grandma QCIF I2B

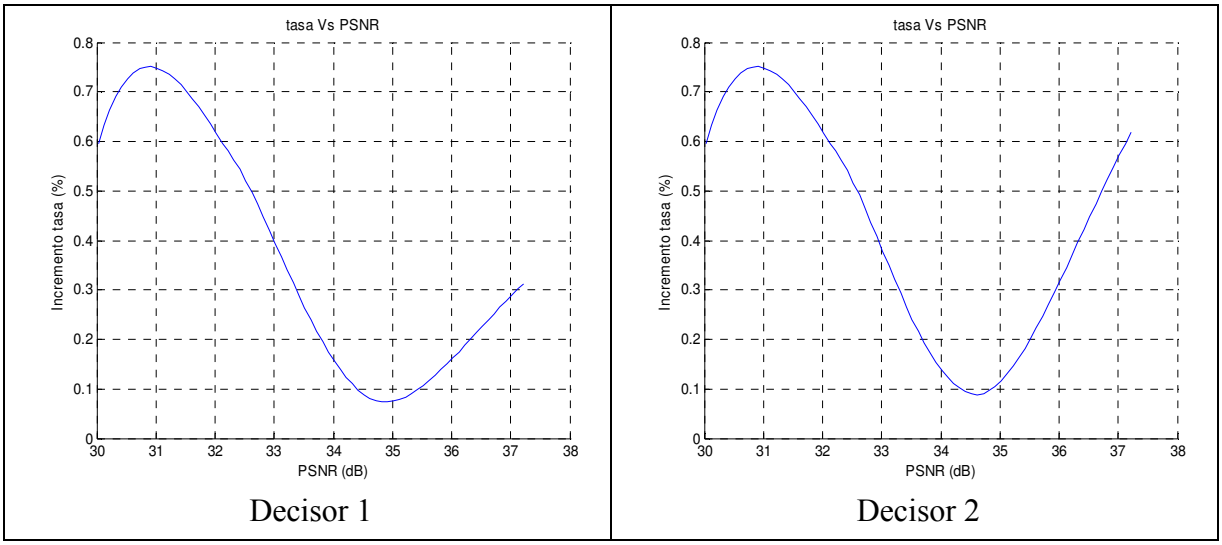


Figura E.27 Gráfica de incremento de tasa vs PSNR Grandma QCIF I2B

Miss-america QCIF I2B

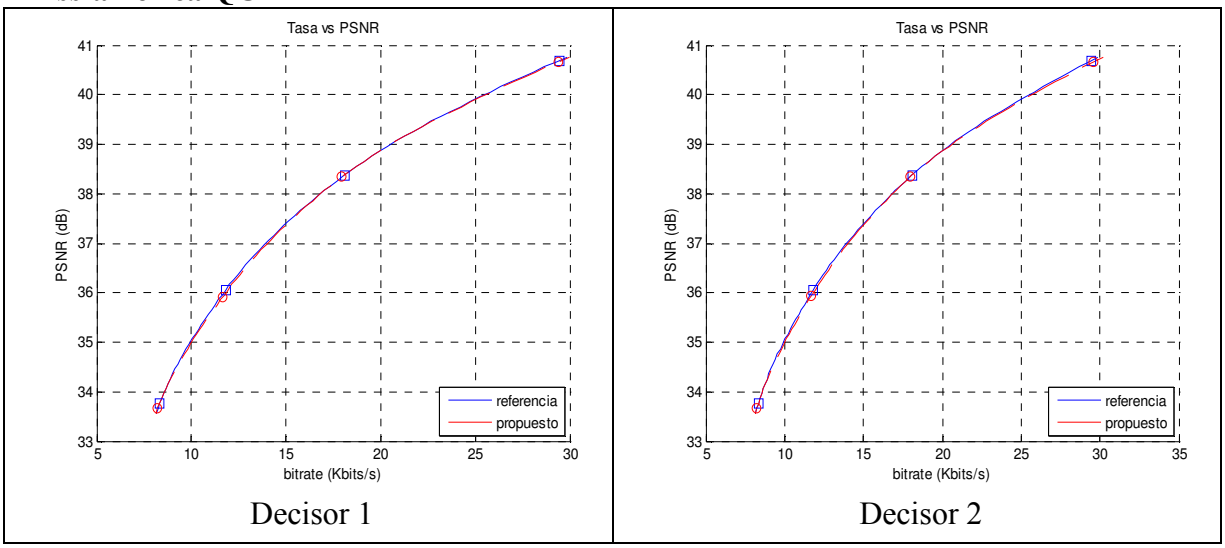


Figura E.28 Gráfica de PSNR vs Tasa Miss-america QCIF I2B

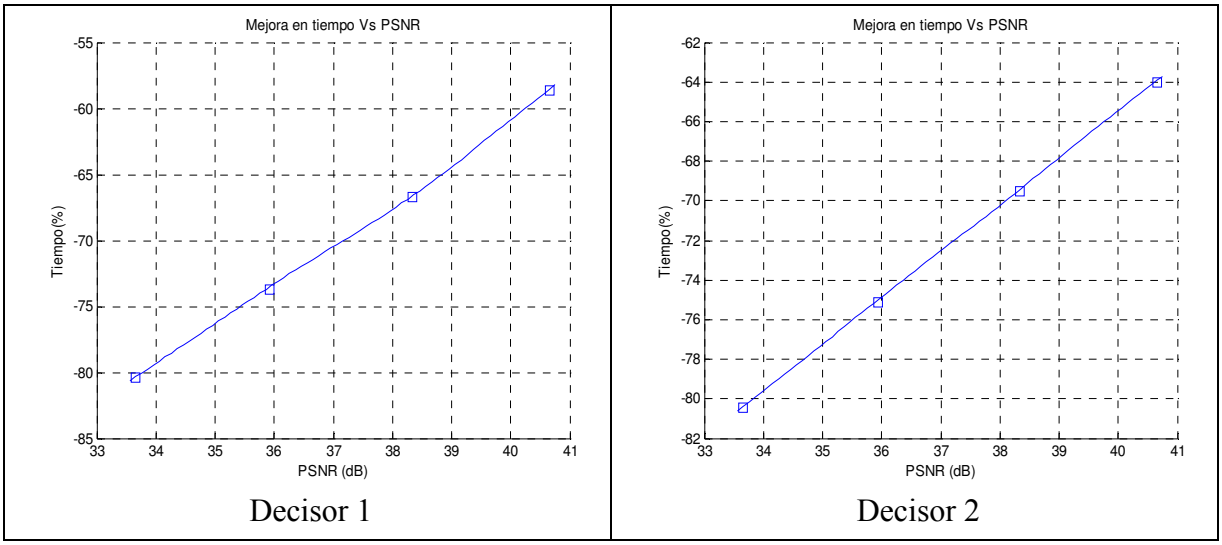


Figura E.29 Gráfica de mejora en tiempo vs PSNR Miss-america QCIF I2B

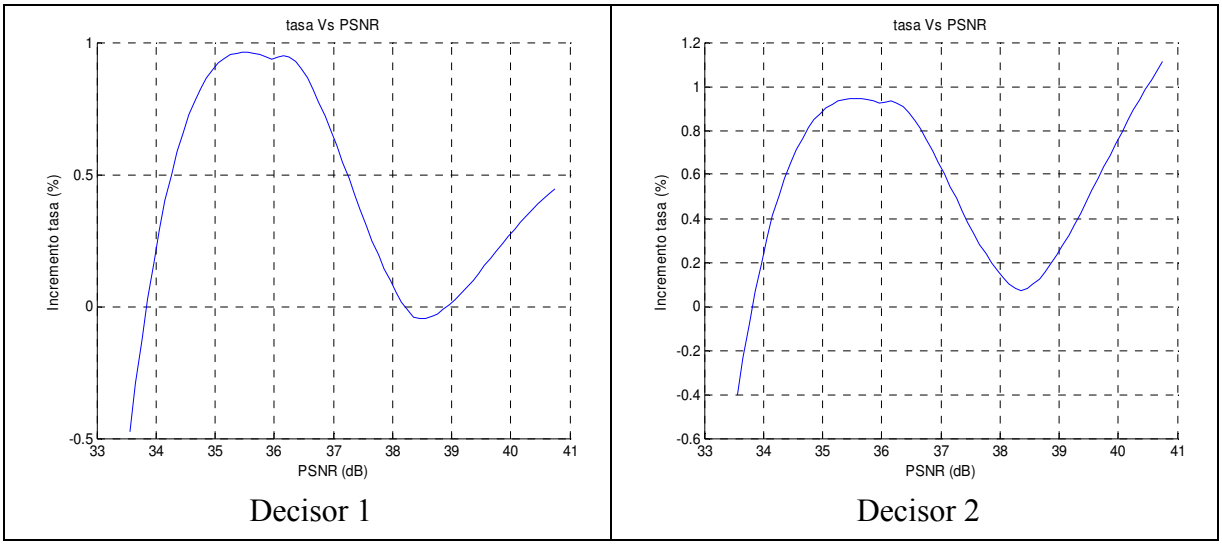


Figura E.30 Gráfica de incremento de tasa vs PSNR Miss-america QCIF I2B

Referencias

[1] J.Lee and B. Jeon "Fast Mode Decision for H.264". Joint Video Team of ISO/IEC MPEG&ITU-T VCEG. 10th Meeting: Waikoloa, Hawaii, USA December 2003. Document JVT-J033

[2] J. Lee, I.Choi, W. Choi and B. Jeon "Fast Mode Decision for B slice" Joint Video Team of ISO/IEC MPEG&ITU-T VCEG. 11th Meeting: Munich, Dei, March 2004. Document JVT-K021

[3] D. Wu, F. Pan, K. P. Lim, S. Wu, Z. G. Li, X. Lin, S. Rahardja and C. C. Ko. "Fast Intermode Decision in H.264/AVC Video Coding". TCSVT VOL. 16, NO. 6, July 2005

[4] C. Grecos and M. Yuan "Fast Mode Prediction for the Baseline and Main Profiles in the H.264 Video Coding Standard" IEEE Transaction on Multimedia, Vol. 8, NO. 6, December 2006.

[5] P. Yin, H. C. Tourapis y otros. "Fast Mode Decision and Motion Estimation for JVT/H.264". ICIP VOL. 3, Issue 15-17, September 2003

[6] J. Lee y B. Jeon. "Fast Mode Decision for H.264 with Variable Motion Block Sizes". ISCIS 2003.

[7] Andy Chang, Peter H.W.Wong y otros. "Fast Multi-block Selection for H.264 Video Coding" ISCAS 2004

[8] M. Bystrom, I. Richardson y Y. Zhao "Efficient Mode Selection for H.264 Complexity Reduction in a Bayesian Framework" Signal Processing: Image Communication, vol. 23, issue 2, pp. 71-86, February 2008.

[9] F. Pan, H. Yu y Z. Lin "Scalable Fast Rate-Distortion Optimization for H.264/AVC" EURASIP Journal on Applied Signal Processing Volume 2006, NO. 1, pp. 71-86

[10] Gary J. Sullivan and Thomas Wiegand "Rate Distortion Optimization for Video Compression". Nov 1998.

[11] Software de referencia JM: http://iphome.hhi.de/suehring/tml/

[12] Iain E.G Richardson. “H.264 and MPEG-4 Video Compression”. Wiley. 2003

[13] Iain E.G Richardson. “Video Codec Design” Wiley. 2002.