

**Universidad Politécnica de Madrid**  
**Escuela Técnica Superior de Ingenieros de**  
**Telecomunicación**



**NUEVO ALGORITMO DE COMPRESIÓN MULTIMEDIA:  
CODIFICACIÓN POR SALTOS LOGARITMICOS**

**Tesis Doctoral**

**Jose Javier García Aranda**  
**Ingeniero de Telecomunicación**

**Año 2015**



Departamento de Ingeniería de Sistemas telemáticos

Escuela Técnica Superior de Ingenieros de  
Telecomunicación

Tesis Doctoral

**NUEVO ALGORITMO DE COMPRESIÓN  
MULTIMEDIA:  
CODIFICACIÓN POR SALTOS LOGARITMICOS**

Autor

**Jose Javier García Aranda**  
Ingeniero de Telecomunicación

Director

**Francisco González Vidal**  
Doctor Ingeniero de Telecomunicación

Año 2015



Autor: Jose Javier García Aranda

Director: Francisco González Vidal



**POLITÉCNICA**

Tribunal nombrado por el Sr. Rector Magnífico de la Universidad Politécnica de Madrid,

el día ..... de ..... de 2015.

Presidente: .....

Vocal: .....

Vocal: .....

Vocal: .....

Secretario: .....

Realizado el acto de defensa y lectura de la Tesis el día ..... de ..... de 2015 en .

Calificación: .....

EL PRESIDENTE

LOS VOCALES

EL SECRETARIO



*“Si buscas resultados distintos no hagas siempre lo mismo”*

*Albert Einstein.*

*“Deben de estar muy cansados de mí... ¡Mirando la misma foto todos estos años!”.*

*Lena Söderberg*



# Resumen

Quizás el Código Morse, inventado en 1838 para su uso en la telegrafía, es uno de los primeros ejemplos de la utilización práctica de la compresión de datos [1], donde las letras más comunes del alfabeto son codificadas con códigos más cortos que las demás. A partir de 1940 y tras el desarrollo de la teoría de la información y la creación de los primeros ordenadores, la compresión de la información ha sido un reto constante y fundamental entre los campos de trabajo de investigadores de todo tipo. Cuanto mayor es nuestra comprensión sobre el significado de la información, mayor es nuestro éxito comprimiéndola.

En el caso de la información multimedia, su naturaleza permite la compresión con pérdidas, alcanzando así cotas de compresión imposibles para los algoritmos sin pérdidas. Estos “recientes” algoritmos con pérdidas han estado mayoritariamente basados en transformación de la información al dominio de la frecuencia y en la eliminación de parte de la información en dicho dominio. Transformar al dominio de la frecuencia posee ventajas pero también involucra unos costes computacionales inevitables.

Esta tesis presenta un nuevo algoritmo de compresión multimedia llamado “LHE” (Logarithmical Hopping Encoding) que no requiere transformación al dominio de la frecuencia, sino que trabaja en el dominio del espacio. Esto lo convierte en un algoritmo lineal de reducida complejidad computacional. Los resultados del algoritmo son prometedores, superando al estándar JPEG en calidad y velocidad.

Para ello el algoritmo utiliza como base la respuesta fisiológica del ojo humano ante el estímulo luminoso. El ojo, al igual que el resto de los sentidos, responde al logaritmo de la señal de acuerdo a la ley de Weber.

El algoritmo se compone de varias etapas. Una de ellas es la medición de la “Relevancia Perceptual”, una nueva métrica que nos va a permitir medir la relevancia que tiene la información en la mente del sujeto y en base a la misma, degradar en mayor o menor medida su contenido, a través de lo que he llamado “sub-muestreado elástico”. La etapa de sub-muestreado elástico constituye una nueva técnica sin precedentes en el tratamiento digital de imágenes. Permite tomar más o menos muestras en diferentes áreas de una imagen en función de su relevancia perceptual.

En esta tesis se dan los primeros pasos para la elaboración de lo que puede llegar a ser un nuevo formato estándar de compresión multimedia (imagen, video y audio) libre de patentes y de alto rendimiento tanto en velocidad como en calidad.



# Abstract

The Morse code, invented in 1838 for use in telegraphy, is one of the first examples of the practical use of data compression [1], where the most common letters of the alphabet are coded shorter than the rest of codes. From 1940 and after the development of the theory of information and the creation of the first computers, compression of information has been a constant and fundamental challenge among any type of researchers. The greater our understanding of the meaning of information, the greater our success at compressing.

In the case of multimedia information, its nature allows lossy compression, reaching impossible compression rates compared with lossless algorithms. These "recent" lossy algorithms have been mainly based on information transformation to frequency domain and elimination of some of the information in that domain. Transforming the frequency domain has advantages but also involves inevitable computational costs.

This thesis introduces a new multimedia compression algorithm called "LHE" (logarithmical Hopping Encoding) that does not require transformation to frequency domain, but works in the space domain. This feature makes LHE a linear algorithm of reduced computational complexity. The results of the algorithm are promising, outperforming the JPEG standard in quality and speed.

The basis of the algorithm is the physiological response of the human eye to the light stimulus. The eye, like other senses, responds to the logarithm of the signal according with Weber law.

The algorithm consists of several stages. One is the measurement of "perceptual relevance," a new metric that will allow us to measure the relevance of information in the subject's mind and based on it; degrade accordingly their contents, through what I have called "elastic downsampling". Elastic downsampling stage is an unprecedented new technique in digital image processing. It lets take more or less samples in different areas of an image based on their perceptual relevance.

This thesis introduces the first steps for the development of what may become a new standard multimedia compression format (image, video and audio) free of patents and high performance in both speed and quality.



# Agradecimientos

En primer lugar deseo expresar mi agradecimiento al director de esta tesis, Dr. Francisco González Vidal, porque fue él quien supo ver que yo debía embarcarme en esta singladura.

Asimismo, agradezco a Joaquin Navarro y Mario Cao su ayuda como compañeros de trabajo y por sus continuas revisiones del artículo que finalmente logramos publicar. Y a Marina, quién fue mi becaria, su trabajo y su aguante para escuchar estoicamente mis teorías cada mañana sin dimitir.

Agradezco a mi familia pues con su apoyo y cariño es más fácil hacer tesis doctorales. A mi madre y a mis hermanos Piluca, Oscar, Maribel y Dani.

Por último agradezco a aquellos que no creen que se pueda mejorar algo porque “ya hay mucha gente trabajando en ello”, o porque “si algo pudiese mejorarse ya estaría inventado” o porque “no es tan fácil”. Ellos son un motivo más para esforzarse en demostrar que es posible vencer cualquier límite existente.



# Índice General

<b>1. Introducción.....</b>	<b>33</b>
1.1    Contexto y motivación .....	33
1.2    Objetivos .....	34
1.3    Metodología de trabajo y estructura de la memoria .....	35
<b>2. Estado del arte .....</b>	<b>39</b>
2.1    Presentación del capítulo .....	39
2.2    Breve historia de la teoría de la información .....	39
2.3    Complejidad computacional .....	41
2.3.1    Complejidad temporal .....	42
2.3.2    Complejidad espacial .....	44
2.3.3    Selección del mejor algoritmo .....	44
2.4    Algoritmos de compresión genéricos .....	46
2.4.1    Algoritmos de pre-procesamiento.....	46
2.4.2    Algoritmos estadísticos.....	48
2.4.3    Algoritmos de diccionario (Lempel-Ziv).....	53
2.5    Algoritmos y formatos de compresión de imágenes.....	55
2.5.1    BMP.....	55
2.5.2    TIFF.....	55
2.5.3    GIF.....	56
2.5.4    PNG .....	56
2.5.5    JPEG .....	58
2.5.6    JPEG2000 .....	65
2.5.7    JPEG XR .....	69
2.5.8    WebP.....	69
2.5.9    H264 intraframe .....	70
2.5.10    Fractales.....	71
2.5.11    Redes neuronales .....	72

2.6	Representación y manipulación de imágenes .....	72
2.6.1	Espacios de color .....	73
2.6.2	Tratamiento de Histogramas .....	76
2.6.3	Filtros .....	79
2.6.4	Operaciones de escalado .....	81
2.7	Medidas de calidad de imágenes.....	85
2.7.1	PSNR.....	85
2.7.2	SSIM .....	86
2.8	Formatos y Algoritmos de compresión de video .....	87
2.8.1	Conceptos .....	87
2.8.2	Estrategia general de algoritmos de compresión de video .....	88
2.8.3	Cloud gaming y video interactivo .....	89
2.9	Conclusiones y cualidades deseables para un nuevo algoritmo .....	91
<b>3.</b>	<b>LHE básico .....</b>	<b>93</b>
3.1	Presentación del capítulo .....	93
3.2	Fundamentos de LHE: modelado del ojo humano .....	93
3.2.1	Respuesta logarítmica al estímulo: Ley de weber .....	93
3.2.2	Modelado en LHE de la ley de Weber.....	95
3.2.3	Umbral de detección del ojo humano .....	101
3.2.4	Modelado en LHE del umbral de detección.....	102
3.2.5	Acomodación del ojo humano.....	103
3.2.6	Modelado en LHE del proceso de acomodación .....	104
3.3	Codificador binario de hops para LHE básico .....	106
3.4	Resumen del algoritmo LHE y diagrama de bloques .....	109
3.5	Paralelización del algoritmo LHE básico .....	109
3.6	Color.....	110
3.7	Resultados.....	111
3.7.1	Calidad vs bpp.....	111
3.7.2	Tiempos de ejecución lineales.....	113
3.8	Resumen del capítulo .....	114
<b>4.</b>	<b>LHE avanzado .....</b>	<b>115</b>
4.1	Presentación del capítulo .....	115

4.2	Importancia del la estrategia de downsampling .....	116
4.2.1	Primera estrategia de downsampling creada y limitaciones.....	116
4.2.2	Introducción a la segunda estrategia de downsampling .....	118
4.2.3	Creación de la malla para el DownSampling Elástico .....	120
4.3	Relevancia Perceptual: concepto y métrica.....	122
4.3.1	Sensación y percepción .....	123
4.3.2	Propósito de la Métrica de relevancia perceptual.....	124
4.3.3	Métrica de relevancia perceptual.....	124
4.3.4	Cálculo de las métricas en los vértices de la malla .....	130
4.3.5	Optimización de las métricas en alta resolución .....	131
4.3.6	Saturación de la métrica de relevancia perceptual .....	132
4.4	Expansión del histograma de la Relevancia Perceptual.....	134
4.5	Cuantización de la métrica de relevancia perceptual.....	136
4.6	Conversión de Métrica de Relevancia Perceptual en PPP .....	139
4.7	Ecuaciones del DownSampling Elástico .....	145
4.7.1	Downsampling Elástico sin restricciones en PPP .....	147
4.7.2	Restricción de máxima elasticidad.....	149
4.7.3	Restricciones de PPP a forma rectangular .....	150
4.8	Downsampling Elástico separable .....	151
4.9	Imágenes en color.....	159
4.10	Segunda cuantización LHE .....	160
4.11	Codificadores binarios .....	165
4.11.1	Codificador binario de hops.....	166
4.11.2	Codificador binario de datos de malla.....	169
4.12	Complejidad y Paralelización .....	172
4.13	Resumen del capítulo .....	174
<b>5.</b>	<b>Decodificador LHE avanzado.....</b>	<b>175</b>
5.1	Presentación del capítulo .....	175
5.2	Decodificador binario de malla.....	176
5.3	Obtención de PPP .....	178
5.4	Adaptación de PPP a forma rectangular .....	179
5.4.1	Restricciones de PPP .....	180
5.4.2	Imágenes en color.....	180

5.5	Decodificador Binario de Hops .....	181
5.6	Recuperación de la señal .....	183
5.7	Interpolación elástica.....	186
5.7.1	Interpolación elástica por vecino cercano.....	186
5.7.2	Interpolación elástica bilineal.....	188
5.7.3	Interpolación elástica bicúbica .....	194
5.7.4	Tiempos de ejecución y paralelización .....	199
5.8	Complejidad y Paralelización .....	201
5.9	Resumen del capítulo .....	203
<b>6.</b>	<b>Video LHE.....</b>	<b>205</b>
6.1	Presentación del capítulo .....	205
6.2	Cálculo de la información diferencial a prueba de LHE .....	207
6.3	Compresión de la información diferencial.....	209
6.4	Regeneración de la información diferencial.....	210
6.5	Cómputo del nuevo fotograma.....	212
6.6	Complejidad y Paralelización .....	215
6.7	Resumen del capítulo .....	216
<b>7.</b>	<b>Resultados .....</b>	<b>217</b>
7.1	Presentación del capítulo .....	217
7.2	Galería Kodak Lossless True Color Image Suite .....	218
7.2.1	Imágenes de la galería .....	218
7.2.2	Gráficas de resultados PSNR de la galería Kodak .....	220
7.2.3	Tablas de resultados PSNR de la galería Kodak .....	222
7.2.4	Gráficas de resultados SSIM de la galería Kodak .....	224
7.2.5	Tablas de resultados SSIM de la galería Kodak.....	226
7.2.6	Algunos detalles comparativos de la galería Kodak .....	228
7.3	Galería miscelánea de la USC.....	229
7.3.1	Imágenes de la galería miscelánea .....	229
7.3.2	Gráficas de resultados PSNR de la galería miscelánea .....	232
7.3.3	Tablas de resultados PSNR de la galería miscelánea .....	235
7.3.4	Gráficas de resultados SSIM de la galería miscelánea .....	238
7.3.5	Tablas de resultados SSIM de la galería miscelánea.....	240

7.3.6	Algunos detalles comparativos de la galería miscelánea .....	243
7.4	Resultados en color.....	244
7.5	Resultados de video.....	247
<b>8.</b>	<b>Conclusiones y líneas de trabajo futuro .....</b>	<b>253</b>
8.1	Análisis de los objetivos.....	253
8.2	Difusión de resultados .....	257
8.3	Líneas de trabajo futuro.....	257
<b>9.</b>	<b>Apéndice I: Artículo publicado .....</b>	<b>261</b>
<b>10.</b>	<b>Bibliografía.....</b>	<b>289</b>
<b>11.</b>	<b>Acrónimos.....</b>	<b>295</b>



# Índice de figuras

Figura 1. Metodología de trabajo .....	36
Figura 2. Historia de la teoría de la información.....	40
Figura 3. Información y energía .....	41
Figura 4. Clasificación de órdenes de complejidad.....	43
Figura 5. Gráficas comparativas de órdenes de complejidad temporal .....	43
Figura 6. Tiempos de ejecución de diferentes órdenes .....	44
Figura 7. Diagrama de distorsión de dos formatos JPEG y JPEG2000 para una misma imagen.....	45
Figura 8. Comparativa computacional de algoritmos/formatos.....	45
Figura 9. Imagen e histograma preprocesada con DPCM.....	47
Figura 10. Codificador ADPCM G.721 .....	48
Figura 11. Construcción del árbol de Huffman .....	49
Figura 12. Codificación aritmética .....	52
Figura 13. Filtro Paeth detectando bordes .....	57
Figura 14. Etapas de JPEG .....	58
Figura 15 DFT y DCT .....	59
Figura 16. Compactación de la energía de una DCT en comparación con una DFT para una misma imagen .....	59
Figura 17. Efecto de bloques de JPEG .....	60
Figura 18. 64 Funciones base de la DCT.....	60
Figura 19. Ejemplo de luminancias y coeficientes tras la transformación 2d-DCT.....	61
Figura 20. JPEG factor de calidad.....	64
Figura 21. Ejemplo de coeficientes cuantizados y recorrido en zig-zag .....	64
Figura 22. Convolución discreta.....	65
Figura 23. Filtros espejo en cuadratura .....	66
Figura 24. Filtrado recursivo de JPEG2000 .....	66
Figura 25. Convolución recursiva wavelet .....	66
Figura 26. Planos de bits .....	67
Figura 27. Stripe dentro de un plano de bit.....	68
Figura 28. Wavelets produciendo efecto de bloques .....	68

Figura 29. Principales Predictores de WebP .....	70
Figura 30. Predicciones de 4x4 en h264 intraframe .....	71
Figura 31. Auto-similitudes .....	71
Figura 32. Compresión usando RNA .....	72
Figura 33. Modelo de color CIE-xyY .....	73
Figura 34. Modelo de color CIELAB .....	74
Figura 35. Gama de colores RGB sobre el modelo CIE-xyY.....	74
Figura 36. Efecto de la Y en el modelo YCbCr .....	76
Figura 37. Modelos YUV.....	76
Figura 38. Histograma de una imagen .....	77
Figura 39. Ecualización de histograma.....	78
Figura 40. Expansión de histograma .....	78
Figura 41. Filtrado de mediana para eliminar ruido .....	80
Figura 42. Ejemplo simple de cálculo de la convolución .....	80
Figura 43. Efectos de convolución .....	81
Figura 44. Interpolaciones más básicas .....	83
Figura 45. Interpolación por vecino, bicúbica y HQX.....	84
Figura 46. Comparativa de dos métodos de interpolación pixel Art .....	84
Figura 47. Aplicación de frames B .....	89
Figura 48. Latencias en cloud-gaming.....	90
Figura 49. Medición de la latencia de codificación.....	90
Figura 50. Reacción lineal de la pupila ante la variación geométrica de la luz.....	94
Figura 51. Sensibilidad al contraste del ojo humano .....	95
Figura 52. Modelado de la respuesta logarítmica del ojo humano .....	95
Figura 53. Diferentes aproximaciones del color de fondo.....	96
Figura 54. Casos posibles y sus referencias disponibles para la predicción .....	96
Figura 55. Hops positivos y negativos distribuidos a lo largo del rango de luminancia disponible.....	97
Figura 56. Ejemplo de asignación de hops y reconstrucción realizada por LHE .....	100
Figura 57. Elección del número óptimo de hops del cuantizador LHE .....	100
Figura 58. Centro de los intervalos cubiertos por cada hop .....	101
Figura 59. Fracción de weber para el ojo humano .....	102
Figura 60. Fracción de weber para $h1=4$ .....	102
Figura 61. Acomodación del ojo al brillo medio .....	103

Figura 62. Acomodación del ojo al brillo medio .....	104
Figura 63. Redundancia horizontal y vertical.....	106
Figura 64. Distribución estadística de hops obtenidos por LHE para la imagen “Lena” .....	106
Figura 65. Distribución estadística de hops obtenidos por LHE para la imagen “baboon” .....	107
Figura 66. Distribución estadística de hops obtenidos por LHE para la imagen “peppers” .....	107
Figura 67. Diagrama de bloques del compresor LHE básico .....	109
Figura 68. Paralelización de operaciones.....	110
Figura 69. Escalados en crominancia .....	111
Figura 70. Imágenes “lena”, “baboon” y “peppers” .....	112
Figura 71. Efecto de la complejidad en el rendimiento .....	113
Figura 72. LHE avanzado .....	115
Figura 73 diagrama de bloques del codificador LHE avanzado .....	116
Figura 74. Cuatro tipos de downsampling, incluyendo el “no downsampled” .....	117
Figura 75 Ejemplo de estrategia de downsampling usando tres niveles de malla .....	117
Figura 76 Una imagen cuyos bloques poseen las mismas métricas .....	118
Figura 77. Downsampling homogéneo vs Downsampling Elástico.....	119
Figura 78. La imagen de Lena, sub-muestreada elásticamente.....	119
Figura 79. Diferentes resultados con Lena usando diferente número de bloques en la malla .....	121
Figura 80. Máximo PPP y dependencia con el tamaño de la imagen .....	121
Figura 81. Diferentes resultados con Kodim21 usando diferente número de bloques en la malla .....	122
Figura 82. Ubicación del contenido que se va a desarrollar .....	123
Figura 83 Información sensorial y perceptual .....	123
Figura 84. Distintas percepciones en fragmentos de una imagen.....	125
Figura 85. Interpolabilidad de la señal.....	125
Figura 86. Relevancia es independiente del número de bordes .....	126
Figura 87. Ejemplo de cálculo de PRx en un bloque que contiene un borde suave .....	127
Figura 88. Ejemplo de cálculo de PRx en un bloque que contiene dos bordes abruptos.....	128
Figura 89. Ejemplo de cálculo de PRx en un bloque que contiene ruido .....	128
Figura 90. Un bloque con PR=0.5 requiere una muestra por cada dos pixeles originales .....	129
Figura 91. Misma imagen a diferentes escalas, poseen la misma métrica de relevancia perceptual.....	130
Figura 92. Métricas de PR en los vértices de la malla.....	131
Figura 93. La misma imagen a distinta resolución .....	131
Figura 94. Las métricas obtenidas realizando un primer LHE sobre un sub-muestreo de la imagen	

original son correctas.....	132
Figura 95. Histograma de valores de la métrica de PRx y PRy para dos imágenes diferentes .....	133
Figura 96. Resultados sin topar y topando la PR en 0.5. Ambas imágenes igualmente reducidas al 10% .....	134
Figura 97. Ubicación del contenido que se va a desarrollar .....	135
Figura 98. Histograma de PR y cuantización a un mismo cuanto (q2).....	135
Figura 99. Expansión del histograma de PR antes de cuantizar .....	136
Figura 100. Ubicación del contenido que se va a desarrollar .....	137
Figura 101. Asignación de PR al punto medio del cuanto (izquierda) y asignación con distribución geométrica (derecha).....	138
Figura 102. Ubicación del contenido que se va a desarrollar .....	139
Figura 103. Máximo downsampling de un bloque a tamaño 2x2.....	140
Figura 104. Rango de valores de CF para la conversión de PR en PPP .....	142
Figura 105. Evolución de PPP en función de PR y el factor de compresión CF.....	142
Figura 106. Dos imágenes con igual nivel de calidad (QL) y distinto PSNR .....	143
Figura 107. Otras dos imágenes con igual nivel de calidad (QL) y distinto PSNR y tasa de bit.....	144
Figura 108. Diferentes resultados con diferentes valores de QL.....	145
Figura 109. Serie aritmética para el cálculo del gradiente de PPP .....	146
Figura 110. Downsampling Elástico de la imagen Lena sin restricciones .....	147
Figura 111. Efecto del estrangulamiento del downsampling sin restricciones .....	148
Figura 112. Efecto de estrangulamiento en imagen Lena .....	148
Figura 113. Un bloque con métricas de PR muy diferentes en sus esquinas .....	149
Figura 114. Límite a la máxima elasticidad .....	149
Figura 115. Ubicación del contenido que se va a desarrollar .....	150
Figura 116. Ubicación del contenido que se va a desarrollar .....	152
Figura 117. Downsampleig Elástico separable.....	153
Figura 118. Cálculo de Downsampleig Elástico promedio .....	153
Figura 119. Comparación de Downsampleig Elástico vs Downsampleig homogéneo .....	155
Figura 120. Diversas imágenes muestreadas al 10% .....	158
Figura 121. Tiempos de ejecución lineales .....	158
Figura 122. Tiempos de ejecución a diferentes ratios .....	159
Figura 123. Escalados en crominancia .....	159
Figura 124. Ubicación del contenido que se va a desarrollar .....	160

Figura 125. Pixels frontera necesarios para estimar el color de fondo .....	162
Figura 126. Dependencias entre bloques .....	162
Figura 127. Secuencia de pasos para codificar por LHE los bloques.....	163
Figura 128. Interpolar es adaptar longitud pero también adaptar los PPP .....	164
Figura 129. Paralelización del segundo LHE sobre los bloques sub-muestreados .....	164
Figura 130. La calidad sin considerar fronteras es menor .....	165
Figura 131. Ubicación del contenido que se va a desarrollar .....	166
Figura 132. Distribución estadística de hops tras LHE sobre los bloques sub-muestreados.....	167
Figura 133. Las dos etapas del codificador binario de hops .....	167
Figura 134. Ejemplo de transformación de hops en símbolos.....	169
Figura 135. Puntos de la malla a codificar .....	170
Figura 136. Las dos etapas del codificador binario de malla .....	170
Figura 137. Distribución de longitud de códigos tras el codificador binario .....	171
Figura 138. Macrobloques de LHE avanzado en función de su complejidad algorítmica .....	172
Figura 139 diagrama de bloques del decodificador LHE avanzado .....	175
Figura 140. Ubicación del contenido que se va a desarrollar .....	176
Figura 141. Número de puntos de la malla.....	178
Figura 142. Ubicación del contenido que se va a desarrollar .....	178
Figura 143. Ubicación del contenido que se va a desarrollar .....	180
Figura 144. Ubicación del contenido que se va a desarrollar .....	181
Figura 145. Estructura del decodificador binario de hops.....	181
Figura 146. Ubicación del contenido que se va a desarrollar .....	183
Figura 147. Casos posibles y sus referencias disponibles para la predicción .....	184
Figura 148. Dependencias entre bloques .....	184
Figura 149. Interpolación de un bloque por vecino cercano .....	186
Figura 150. Segmento vertical cubierto por cada pixel sub-muestreado.....	187
Figura 151. Coordenada donde debemos interpolar el color de un pixel .....	188
Figura 152. Área pintada por una interpolación bilineal .....	189
Figura 153. Pixel parcialmente cubierto interpolable.....	189
Figura 154. Pixel parcialmente cubierto no interpolable .....	190
Figura 155. Estrategia de interpolación con pixeles ya interpolados .....	190
Figura 156. Efecto de cuadrícula.....	191
Figura 157. Mitigación del efecto de cuadrícula.....	191

Figura 158. Resultado de la interpolación .....	192
Figura 159. Problema de la desalineación de muestras en la interpolación elástica .....	192
Figura 160. Áreas que podrían quedarse sin interpolar .....	193
Figura 161. Interpolación horizontal de costuras .....	193
Figura 162. Interpolación de costuras .....	194
Figura 163. Interpolación cúbica necesita 4 muestras .....	196
Figura 164. Uso de la frontera horizontal en la interpolación cúbica vertical.....	197
Figura 165. Uso de la frontera vertical en la interpolación cúbica horizontal.....	198
Figura 166. Interpolación cúbica vertical de costuras .....	199
Figura 167. Linealidad del proceso de interpolación.....	200
Figura 168. Comparativa de tiempos de ejecución .....	200
Figura 169. Comparativa de calidad .....	201
Figura 170. Macrobloques de LHE avanzado en función de su complejidad algorítmica .....	201
Figura 171 Diagrama de bloques del codificador/decodificador de video .....	206
Figura 172. Ubicación del contenido que se va a desarrollar .....	207
Figura 173. Gráfica de dy .....	208
Figura 174. Valor de la función dy .....	209
Figura 175. Ubicación del contenido que se va a desarrollar .....	210
Figura 176. Ubicación del contenido que se va a desarrollar .....	211
Figura 177. La función dy no es interpolable .....	211
Figura 178. La función inversa de dy.....	212
Figura 179. Ubicación del contenido que se va a desarrollar .....	212
Figura 180. Suma de señales de distinto nivel de sub-muestreo .....	213
Figura 181. Macro-bloques del codificador de video LHE en función de su complejidad algorítmica ....	215
Figura 182. Diagrama de distorsión PSNR para galería Kodak.....	220
Figura 183. Diagrama de distorsión PSNR para imagen “kodim23” .....	221
Figura 184. Diagrama de distorsión PSNR para imagen “kodim21” .....	221
Figura 185. Diagrama de distorsión SSIM para imagen “kodim09” .....	222
Figura 186. Diagrama de distorsión SSIM para galería “kodak” (promediada) .....	224
Figura 187. Diagrama de distorsión SSIM para imagen “kodim23” .....	224
Figura 188. Diagrama de distorsión SSIM para imagen “kodim21” .....	225
Figura 189. Diagrama de distorsión SSIM para imagen “kodim09” .....	225
Figura 190. Comparación de kodim20 entre JPEG (arriba) y LHE (abajo) a 0.1bpp.....	228

Figura 191. Comparación de kodim09 entre JPEG (izquierda) y LHE (derecha) a 0.2bpp .....	229
Figura 192. Diagrama de distorsión PSNR para galería “Miscelánea” (promediada) .....	232
Figura 193. Diagrama de distorsión PSNR para la imagen “Lena” .....	232
Figura 194. Diagrama de distorsión PSNR para la imagen “Peppers” .....	233
Figura 195. Diagrama de distorsión PSNR para la imagen “Man” .....	233
Figura 196. Diagrama de distorsión PSNR para la imagen “Baboon” .....	234
Figura 197. Diagrama de distorsión SSIM para galería “miscelanea” (promediada).....	238
Figura 198. Diagrama de SSIM para la imagen “Lena” .....	238
Figura 199. Diagrama de SSIM para la imagen “peppers” .....	239
Figura 200. Diagrama de SSIM para la imagen “peppers” .....	239
Figura 201. Comparación entre JPEG (izquierda) y LHE (derecha) a 0.1bpp de imagen “Lena” .....	243
Figura 202. Comparación entre JPEG2000 (izquierda) y LHE (derecha) a 0.1bpp de imagen “Lena” .....	243
Figura 203. Comparación entre JPEG (izquierda) y LHE (derecha) a 0.3bpp de imagen “Lena” .....	244
Figura 204. Comparación entre JPEG (izquierda), LHE (centro) y JP2K (izquierda) a 0.1bpp de imagen “4.1.01-girl” .....	244
Figura 205. Luminancia, Crominancia U, Crominancia V sub-muestreadas en YUV420 y reconstruidas.	245
Figura 206. Lena en color a 1.6bpp y PSNR 33,22dB .....	245
Figura 207. Luminancia, Crominancia U, Crominancia V sub-muestreadas en YUV420 y reconstruidas.	246
Figura 208. Lena en color a 1.37bpp y PSNR 35,82dB .....	246
Figura 209. Video soccer .....	248
Figura 210. Video soccer LHE a 270 kbps y 930 kbps.....	248
Figura 211. Video “ice” .....	249
Figura 212. Video football.....	249
Figura 213. Video football LHE a 300 kbps y 1000 kbps .....	249
Figura 214. Video Meerkat .....	250
Figura 215. Video Meerkat LHE a 320 kbps y 1300 kbps .....	250
Figura 216. Video Cars .....	251
Figura 217. Video flamingo .....	251
Figura 218. Complejidad comparada de LHE .....	254
Figura 219. Posibles transparencias en LHE.....	258
Figura 220. Interpolación por vecino, bicúbica y HQX.....	259



# Índice de tablas

Tabla 1. Tabla de resultados de ADPCM sobre imágenes del libro “digital Image compression techniques” .....	48
Tabla 2. Resultados de Huffman adaptativo.....	51
Tabla 3. Comparativa entre Huffman y Lempel-Ziv .....	54
Tabla 4. Resultados de compresión de imágenes con distintos algoritmos .....	55
Tabla 5. Filtros predictivos de PNG .....	57
Tabla 6. Experiencia de usuario con diferentes latencias de red y tipos de juego .....	91
Tabla 7. Correspondencia entre hops y símbolos.....	107
Tabla 8. Valores de calidad de LHE básico para imagen “lena” .....	112
Tabla 9 Valores de calidad de LHE básico para imagen “baboon” .....	112
Tabla 10. Valores de calidad de LHE básico para imagen “peppers” .....	113
Tabla 11. Tiempos de ejecución del cuantizador LHE.....	113
Tabla 12. Intervalos para la cuantización y valores de PR asignados a cada cuanto .....	137
Tabla 13. Valores óptimos de la razón geométrica y primer hop en función de la PR promedio .....	161
Tabla 14. Correspondencia entre hops y símbolos.....	167
Tabla 15. Tabla de símbolos para los cuantos .....	171
Tabla 16. Complejidad y paralelización de cada macrobloque funcional de LHE avanzado .....	173
Tabla 17. Tabla de códigos de los cuantos.....	176
Tabla 18. Los cuantos de PR y su significado .....	177
Tabla 19 Códigos binarios del cuantizador LHE .....	182
Tabla 20. Valores óptimos de la razón geométrica y primer hop en función de la PR promedio .....	185
Tabla 21. Complejidad y paralelización de cada macrobloque funcional.....	202
Tabla 22. Complejidad y paralelización de cada macrobloque funcional.....	216
Tabla 23. PSNR para galería Kodak usando LHE.....	222
Tabla 24. PSNR para galería Kodak usando JPEG .....	223
Tabla 25. PSNR para galería Kodak usando JPEG2000.....	223
Tabla 26. SSIM para galería Kodak usando LHE .....	226
Tabla 27. SSIM para galería Kodak usando JPEG .....	226
Tabla 28. SSIM para galería Kodak usando JPEG2000 .....	227

Tabla 29. PSNR para galería Miscelánea usando LHE .....	235
Tabla 30. PSNR para galería Miscelánea usando JPEG .....	236
Tabla 31. PSNR para galería Miscelánea usando JPEG2000 .....	237
Tabla 32. SSIM para galería Miscelánea usando LHE.....	240
Tabla 33. SSIM para galería Miscelánea usando JPEG .....	241
Tabla 34. SSIM para galería Miscelánea usando JPEG2000 .....	242

# Índice de ecuaciones

Ecuación 1. Información de un evento .....	40
Ecuación 2. Entropía termodinámica.....	40
Ecuación 3. Entropía de una fuente.....	40
Ecuación 4. Transformada 1d-DCT.....	59
Ecuación 5. Ecuaciones 2d-FDCT y 2d-IDCT .....	60
Ecuación 6. Cuantización de los coeficientes 2d-DCT.....	63
Ecuación 7. Ecuaciones del modelo YCbCr .....	75
Ecuación 8. Ecualización de histograma .....	77
Ecuación 9. Expansión de histograma.....	78
Ecuación 10. Ley de Weber .....	94
Ecuación 11. Predicción de la componente (YUV) para un pixel .....	96
Ecuación 12 Relaciones entre hops .....	97
Ecuación 13. Cálculo de la razón geométrica que relaciona los hops .....	98
Ecuación 14. Relación entre tiempos de ejecución .....	110
Ecuación 15. Ecuación de la métrica de relevancia perceptual PR <sub>x</sub> .....	126
Ecuación 16. Ecuación de la métrica de relevancia perceptual PR <sub>y</sub> .....	126
Ecuación 17. Expansión del histograma de PR.....	136
Ecuación 18. Longitudes mínima y máxima del segmento sub-muestreado .....	140
Ecuación 19. Fórmula de obtención de PPP a partir de PR.....	140
Ecuación 20. Fórmula de PPP con límites en valor máximo y mínimo .....	141
Ecuación 21. Cálculo del valor máximo del factor de compresión .....	141
Ecuación 22. Cálculo del valor mínimo del factor de compresión.....	141
Ecuación 23. Nivel de calidad (“QL” o “Quality Level”) y reformulación de PPP.....	142
Ecuación 24. Cálculo inicial del gradiente de PPP.....	146
Ecuación 25. Serie aritmética.....	146
Ecuación 26. Cálculo de l' .....	146
Ecuación 27. Ajuste de l' a un número entero.....	147
Ecuación 28. Fórmula final del gradiente de PPP .....	147
Ecuación 29. Condición rectangular.....	150

Ecuación 30. Suma de PPP iniciales y final.....	151
Ecuación 31. Dos ecuaciones de la suma final de los PPP de cada lado .....	151
Ecuación 32. Reajuste de PPP a forma rectangular .....	151
Ecuación 33. Valor inicial de h1 para cada bloque .....	161
Ecuación 34. Fórmula de PPP.....	179
Ecuación 35. Fórmula de la razón de los niveles de calidad .....	179
Ecuación 36 Relaciones entre hops .....	184
Ecuación 37. Gradientes de interpolación.....	187
Ecuación 38. Gradiente de componente de color entre dos muestras .....	188
Ecuación 39. Interpolación lineal.....	195
Ecuación 40. Interpolación cúbica .....	195

# Capítulo 1

## Introducción

### 1.1 Contexto y motivación

Muchos son los motivos para comprimir la información. El espacio que ocupa una información es un recurso siempre limitado. Y el tiempo que podemos invertir en comprimir la información es también un recurso limitado. El espacio-tiempo es el recurso “combinado” que los algoritmos de compresión deben emplear del mejor modo posible para maximizar la calidad obtenida. Ajustar un algoritmo para que utilice razonablemente ambos recursos (espacio y tiempo) es lo que se conoce como el “time-memory” tradeoff [2].

Cada año se producen avances tanto de la capacidad de almacenamiento de datos como en la velocidad de procesamiento en las computadoras. A la vez se produce una reducción de costes de ambos recursos que ponen en cuestión la necesidad de compresión de datos. Sin embargo, cuanto mayor son los recursos disponibles, mayor es la demanda de servicios de información y mayor la demanda de la calidad deseada en dichos servicios, como es el caso de la distribución por Internet de video de alta definición, el “cloud gaming” o algo tan cotidiano como las cámaras fotográficas, cuya resolución y calidad no parece dejarnos nunca satisfechos.

Desde que fue inventado el algoritmo de Huffman en 1952, se han desarrollado muchos algoritmos en una carrera sin fin y hoy en día conviven algunos muy simples con otros muy complejos, en nuestras herramientas más habituales para almacenar y reproducir información.

Es tal la necesidad de comprimir la información que no es raro que surjan de vez en cuando anomalías que dicen violar las leyes de la información, como la que en abril de 1992 anunció la empresa “WEB technologies”, afirmando que disponían de un algoritmo capaz de comprimir recursivamente cualquier información a 1KB. Sin hacer grandes cálculos podemos comprobar que 1KB permite  $2^{1024 \cdot 8} = 10^{2466}$  posibilidades. Un número tan enorme que mucha gente creyó en la idea. Resultó ser falsa. Historias similares sucedieron con las empresas Pixelon, HyperSpace, Pegasus Technologies o ZeoSync. Todas anunciaron haber encontrado el santo grail y finalmente resultó ser mentira, y la verdadera razón que suelen tener es lograr subidas de sus acciones en bolsa.

Según el teorema de Shannon, si la información es completamente aleatoria, no es posible comprimirla. Esto se sabía mucho antes de que todos estos fraudes tuviesen lugar, sin embargo, es sano cuestionar una y otra vez los límites que pensamos que son inviolables. Y por otro lado, es tan clara la necesidad que tenemos de consumir y almacenar información que una mejora notable de la compresión de datos puede suponer un tsunami en el mundo bursátil.

Internet ha sido uno de los impulsores de la mejora en los algoritmos de compresión, sobre todo en la compresión multimedia, porque las limitaciones del ancho de banda iniciales hacían inviable la transmisión rápida de imágenes sin comprimir. JPEG nace en 1986 (su primera versión) para dar respuesta a esta necesidad. Y por el mismo motivo pronto aparecen los primeros compresores de video,

como MPEG-1.

Es la necesidad de superar a JPEG lo que motivó la creación de JPEG2000 en el año 2000. Algo parecido sucedió en la compresión sin pérdidas, con PNG como sucesor natural del formato GIF. Unos formatos suceden a otros pero a mayor compresión también se incrementa la complejidad algorítmica. Mejorar en ratio de compresión es mejorar en el uso del recurso espacio, pero no en el recurso tiempo.

Sin embargo las patentes de JPEG2000 han impedido su amplia difusión. De hecho debido a la publicación incesante en la web de imágenes cada vez más grandes, la Fundación Mozilla se embarcó en un proyecto en 2014 para acelerar la carga de las imágenes JPEG llamado mozjpeg, con reducciones de un 10% respecto del tamaño JPEG original, descartando el formato JPEG2000.

Con la llegada del cloud gaming se hace más evidente la necesidad de mejorar el compromiso espacio-tiempo. El cloud gaming se basa en el principio de que el procesamiento se realiza en servidores. De esta forma los terminales de usuario se limitan a reproducir los gráficos transmitidos por streaming desde los servidores y a realizar la transmisión de los eventos provocados por el usuario. Los requisitos de tiempo de respuesta y calidad de video recibido son elementos cruciales para una calidad de experiencia de juego (QoE) equivalente a la de las consolas tradicionales. Este es un caso de uso donde el recurso tiempo, o lo que es lo mismo, minimizar complejidad computacional del algoritmo de compresión, es fundamental para lograr el objetivo de QoE buscado.

En el servidor, a diferencia de otros servicios de streaming como YouTube, las plataformas de cloud gaming realizan streaming en tiempo real, lo que significa que la fuente de video se está creando a medida que avanza el juego y no es posible aplicar algunas técnicas de compensación de movimiento de compresión de video [3]. Debido a la estricta restricción de la latencia extremo a extremo para la interacción en tiempo real con un videojuego, aun hay retos para el diseño de un sistema de cloud gaming efectivo que satisfaga los requisitos de calidad de experiencia de los usuarios [4]

Existen, pues, retos que justifican continuar trabajando en el desarrollo de algoritmos de compresión en general y en particular en la compresión de imágenes y video como son:

- El “time-memory” trade-off en servicios de estricta latencia de codificación como es el caso del “cloud-gaming”.
- La mejora de la calidad (acerándola lo más posible a la compresión sin pérdidas).
- El problema de las patentes y que afecta a la difusión de los algoritmos (como ocurre con JPEG2000 o con la compresión fractal).

## 1.2 Objetivos

El principal objetivo de esta tesis es la creación de un algoritmo de compresión con pérdidas de imágenes con complejidad computacional lineal, de propósito general, para todo tipo de imágenes (fotografías, dibujos, imagen sintética e iconos) que en un futuro constituir el núcleo de un codificador de video de complejidad lineal.

Este objetivo responde al problema no resuelto de la reducción de la latencia de codificación, cuya reducción hasta alcanzar la codificación instantánea siempre permanecerá como una meta para los

inventores de algoritmos. Su aplicación práctica más inmediata se encuentra en los servicios de video interactivo y en particular en lo que se conoce como “cloud-gaming”, donde la suma de latencias de red y de codificación impiden lograr una experiencia de usuario idéntica a la que tendría con la video consola en sus manos. Otra de sus aplicaciones más inmediatas es la reducción de consumo energético [5], en especial el de los dispositivos portátiles que requieren el uso de baterías de durabilidad limitada. La reducción de complejidad computacional puede lograr ahorros significativos en el consumo energético y alargar el uso de las baterías.

El algoritmo que se va a presentar en esta tesis (codificación por saltos logarítmicos “LHE”) trabaja en el dominio del espacio, rompiendo de este modo con la corriente de “transform compressors” como el conocido JPEG o otros conocidos formatos cuya “semilla inicial” fue sembrada en 1965 con la introducción de la FFT [6]. Desde entonces diversas transformaciones al dominio de la frecuencia han surgido como mejoras, compactando cada vez más la energía al tiempo que se aumentaba la complejidad computacional [7]. La búsqueda de un nuevo algoritmo cuya mayor ventaja sea la eficiencia computacional, ha motivado el procesamiento en el espacio de LHE, sin transformación a frecuencia, explorando otro rumbo respecto del paradigma establecido por décadas.

En base a este objetivo global descrito, se proponen los siguientes objetivos concretos:

- Creación de un algoritmo básico de compresión (sin creación ni uso de patentes) de complejidad lineal, sin transformación al dominio de la frecuencia y fundamentado en la ley de Weber de la percepción y en otras características del comportamiento fisiológico del ojo humano.
- Creación de una métrica de relevancia perceptual que permita medir la relevancia que tiene para el observador cualquier área de la imagen para poder sub-muestrear la imagen original en base a la métrica, asignando más muestras en zonas relevantes y menos en zonas poco relevantes (sub-muestreado “elástico”).
- Creación de los algoritmos de interpolación elástica para visualizar las imágenes comprimidas.
- Creación de un algoritmo completo que usando el sub-muestreado elástico y la ley de weber permita comprimir linealmente y con mayor calidad que JPEG.
- Validación del algoritmo con bases de datos de imágenes de referencia usando diferentes métricas estándares de calidad (PSNR y SSIM).
- Validación del modelo de color YUV (YUV 444, YUV 422 y YUV 420) con el nuevo algoritmo.
- Creación del mecanismo de utilización de LHE en compresión de video, es decir, codificación de información diferencial temporal y primera comparativa con el codificador estándar de mayor difusión (H264)

### 1.3 Metodología de trabajo y estructura de la memoria

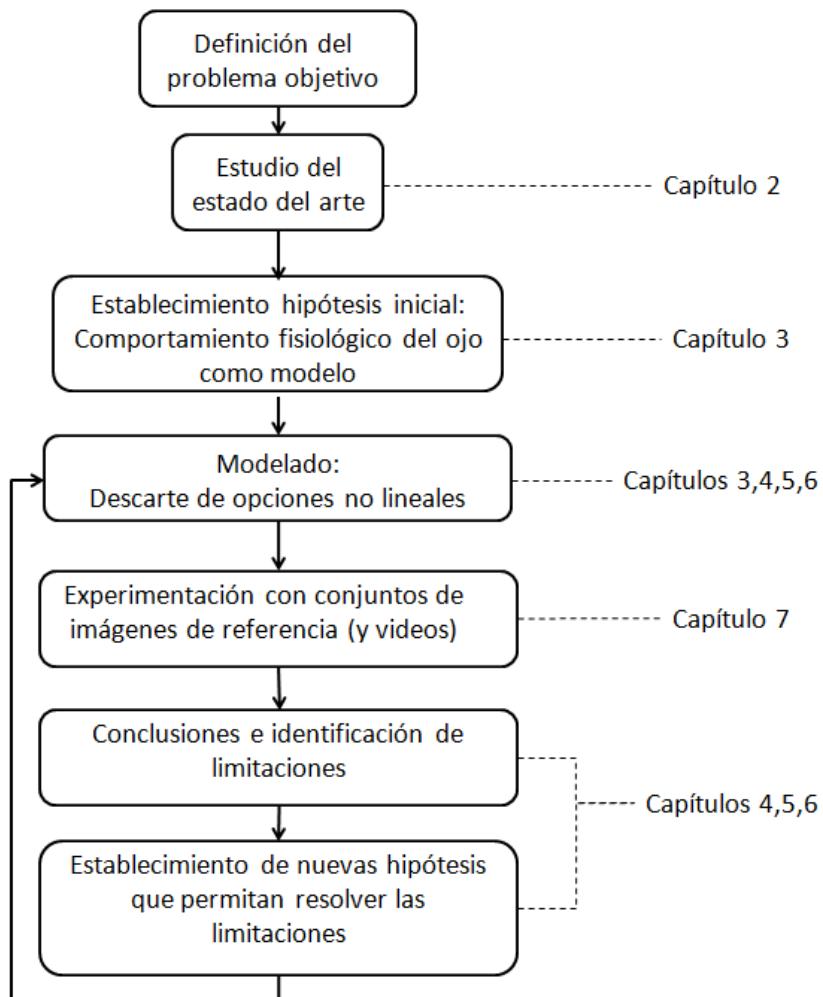
La metodología empleada parte de un estudio de la literatura con el objetivo de identificar y analizar trabajos relacionados. Al tratarse de un algoritmo sin precedentes, el estudio no proporcionará la base del algoritmo pero permitirá conocer necesidades no cubiertas y el conocimiento necesario para diseñar y evaluar el nuevo algoritmo.

Tras este estudio se procede con el establecimiento de hipótesis para la creación de un algoritmo nuevo de complejidad lineal basado en el comportamiento fisiológico del ojo y sobre esta base inicial se va construyendo el algoritmo completo mediante iteraciones sucesivas, descartando siempre cualquier

modelado no lineal que se pueda derivar del planteamiento de las nuevas hipótesis construidas.

La validación de hipótesis y la correcta definición de cada modelado se realizan sobre baterías de imágenes de referencia, utilizadas ampliamente en el contexto de procesado digital de imágenes y validación de codificadores. Para ello se parte de una imagen inicial de referencia (“Lena”) y si el resultado es positivo, se procede a la validación global con el conjunto completo de imágenes de referencia.

La metodología empleada se resume en la figura, en la que se indican además los capítulos relacionados con cada bloque, los cuales se describen seguidamente



**Figura 1. Metodología de trabajo**

Esta tesis se organiza en los siguientes capítulos:

- Capítulo 1: Introduce el contexto, la motivación y los objetivos que persigue esta tesis doctoral.
- Capítulo 2: Estado del arte. En este capítulo se incluyen conceptos generales sobre compresión

de la información, complejidad computacional, algoritmos de compresión de imágenes, tratamiento digital de imágenes, métricas de calidad e introducción a la compresión de video.

- Capítulo 3: descripción del algoritmo LHE “básico”, en el que se exponen sus fundamentos en relación con la fisiología del ojo humano, así como sus diferentes bloques funcionales y un análisis de su complejidad computacional y su capacidad de paralelización.
- Capítulo 4: descripción del algoritmo LHE “avanzado”, en el que introduce la métrica de relevancia perceptual y el mecanismo de downsampling elástico fundamentado en dicha métrica. Asimismo se exponen los diferentes bloques funcionales del algoritmo LHE avanzado y un análisis de su complejidad computacional y su capacidad de paralelización.
- Capítulo 5: descripción del decodificador LHE, en el que se expone el mecanismo para interpretar las imágenes comprimidas con LHE, se describe el mecanismo de interpolación elástica y se exponen los diferentes bloques funcionales del decodificador LHE avanzado, así como un análisis de su complejidad computacional y su capacidad de paralelización.
- Capítulo 6: en este capítulo se presenta el diseño de un codificador de vídeo básico fundamentado en el algoritmo LHE. El codificador de video básico resuelve el primer problema en la construcción de un codificador de video LHE, que consiste en sumar iterativamente información diferencial temporal con pérdidas, que ha pasado por una codificación LHE. Una vez más, se expone un análisis de su complejidad computacional y su capacidad de paralelización.
- Capítulo 7: Resultados de calidad en imagen fija, comparados con los compresores de referencia JPEG y JPEG2000. Se incluyen también resultados de calidad de video comparados con H264
- Capítulo 8: Conclusiones más relevantes del desarrollo de la tesis e introducción de líneas futuras de investigación.
- Apéndice I: Artículo publicado en la revista “IET Image Processing” sobre LHE, aceptado en Diciembre de 2014



# Capítulo 2

## Estado del arte

### 2.1 Presentación del capítulo

En este capítulo se clasifican y describen los algoritmos de compresión existentes más importantes genéricos y específicos de imágenes.

Para ello primeramente introduciré el concepto de entropía de la información y el límite teórico de compresión de cualquier algoritmo.

Hay dos criterios para comparar los algoritmos de compresión:

- Su complejidad computacional
- La calidad de su resultado (en caso de compresión con pérdidas)

Por este motivo procederé a describir los algoritmos con un enfoque orientado a la comparación de coste computacional entre ellos, introduciendo previamente los conceptos de la complejidad computacional.

Se presentarán diferentes métodos de evaluación de la calidad de imágenes obtenida en algoritmos de compresión con pérdidas, que es segundo de los criterios enunciados para comparar algoritmos.

Serán cubiertos también algoritmos de procesamiento de señal relacionados con esta tesis y cuyos conceptos son empleados en el algoritmo LHE, como son las transformaciones de histograma y los modelos de color.

Por último se presentarán los fundamentos de la compresión de video, útiles para comprender el diseño del codificador de video básico de LHE y como podría evolucionar.

### 2.2 Breve historia de la teoría de la información

La Teoría de la Información es la disciplina que se encarga del estudio y cuantificación de la información para poderla representar de forma eficiente. En esta sección introduciré el concepto de cantidad de información o entropía, fundamental en el desarrollo de esta tesis [8].

La cantidad de información que nos proporciona cierto dato es menor cuanto más esperamos ese dato. Ya en el 384 A.C., Aristóteles afirmaba “cuanto menos probable es un evento, más información tiene”. Armar esta frase hasta su formulación actual ha llevado algo más de 2000 años.

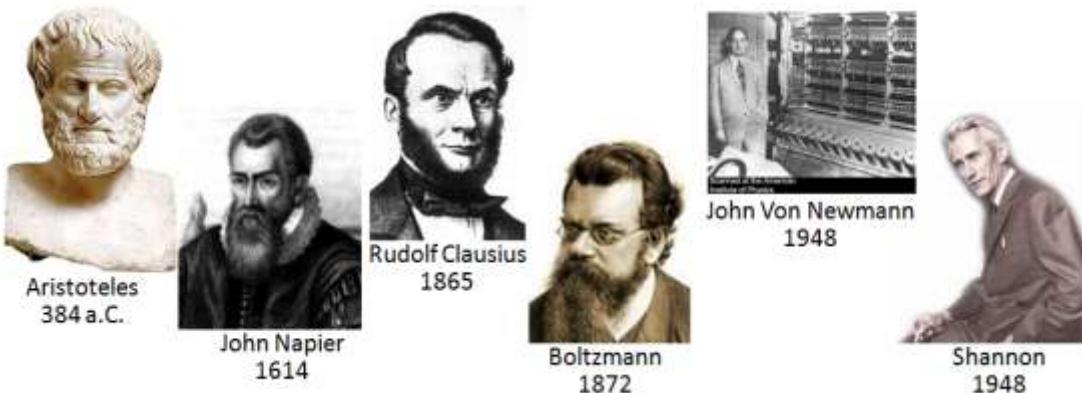


Figura 2. Historia de la teoría de la información

En 1614, John Napier, inventa los logaritmos con lo que la información puede empezar formularse utilizando la probabilidad de un evento:

$$I = \log \left( \frac{1}{p} \right)$$

Ecuación 1. Información de un evento

En 1865 Clausius introduce el concepto de entropía termodinámica “S”, como una función básica de la segunda ley “La cantidad de entropía del universo tiende a incrementarse en el tiempo”. La entropía y la cantidad de información son conceptos equivalentes.

Boltzman la formula analíticamente en 1872: La cantidad de entropía de un sistema es proporcional al logaritmo natural del número de microestados posibles (al parecer dicha fórmula figura en su lápida).

$$S = K \cdot \ln W$$

Ecuación 2. Entropía termodinámica

Shannon 1948 (Bell labs): “La información de dos eventos independientes que suceden a la vez debe ser la suma de la información de ambos”. Shannon no llamó información a este concepto, ni cantidad de información sino que le pidió una sugerencia a Von Newmann y este le dijo que lo llamase entropía [9] porque era una palabra que estaba de moda en la comunidad científica y realmente nadie conocía exactamente su significado.

$$H = E(I) = \sum p \cdot \log_2 \left( \frac{1}{p} \right)$$

Ecuación 3. Entropía de una fuente

El sumatorio es el sumatorio del número total de códigos en un mensaje (o en una fuente). Usando logaritmo en base dos, el resultado lo obtenemos en bits. En un mensaje la fórmula de la entropía sería el sumatorio de  $\log(1/p)$ , pero en una fuente se multiplica por “p” debido a que aunque un símbolo de baja p aporte mucha información, aparece muy poco y la formula hace referencia a una fuente de información, donde en lugar de decir cuántas veces aparece cada símbolo, simplemente multiplicamos por su p para caracterizar la fuente.

Si hay equiprobabilidad de estados entonces  $p=1/n$  para cualquier código y el sumatorio da el máximo valor posible de la entropía. En cualquier caso, dado un mensaje de entropía H expresada en bits, no

podremos representarlo con menos bits que  $H$ , es decir, el teorema de Shannon fija un límite a la máxima compresión de datos posible. El mejor algoritmo de compresión sin pérdidas (lossless) alcanzará  $H$  pero no podrá usar menos bits.

Una reflexión de Roger Penrose [10] en 2010 relaciona la entropía o cantidad de información con la energía/materia.

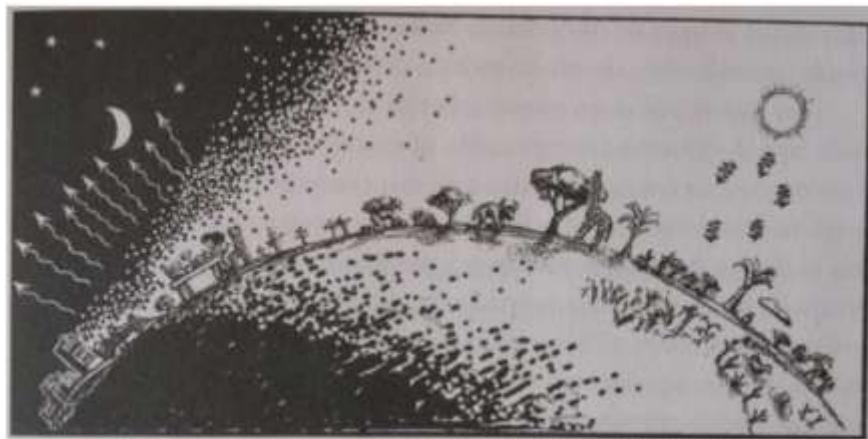


Figura 3. Información y energía

*“La energía que recibe la Tierra cada día del Sol es esencialmente igual a la que devuelve la tierra al espacio. [...] Los fotones que llegan a la tierra procedentes del Sol tienen una energía más alta (longitud de onda más corta  $E=h\nu$ ) que los que la Tierra devuelve al espacio. Dado un balance de energía global (la tierra no se calienta más con el paso del tiempo), [...] debe haber más fotones que se llevan energía de la Tierra que fotones que traen esa misma energía procedentes del Sol. Más fotones implican más grados de libertad y en consecuencia más entropía. Es decir, la energía que llega tiene una entropía menor que la energía que sale.” El Sol es, en realidad, una fuente de baja entropía para nosotros.*

Relacionar la energía (o la materia) con la información tal como lo hace Roger Penrose es trazar un puente entre el concepto de realidad y el concepto de información. Con ello entraríamos en el terreno de la filosofía y en el conocido problema de “lo real”, por donde no pretendo caminar.

## 2.3 Complejidad computacional

Un algoritmo siempre es mejor que otro si haciendo lo mismo su complejidad computacional es inferior. Las potentes máquinas actuales a veces nos hacen olvidar esta fundamental premisa y hacen que la programación se vuelva ineficiente, llevando a autores como David May a formular su “alternativa” a la ley de Moore, afirmando que *“La eficiencia del software se divide a la mitad cada 18 meses compensando la ley de Moore”* [11]. Dicha ley posee una dosis de realidad mayor que la deseable.

El concepto de complejidad computacional se relaciona con dos aspectos de cualquier algoritmo:

- Complejidad temporal : el número de operaciones que requiere
- Complejidad espacial: Los recursos de memoria que consume

El primero de los dos criterios está directamente relacionado con el tiempo que consume su ejecución,

aunque si el algoritmo permite paralelización puede no ser así. No obstante su capacidad de paralelización no reduce su complejidad pues aunque la capacidad de paralelización es una virtud para cualquier algoritmo, lo que realmente importa en la medida de su complejidad es el número de operaciones que requiere.

Determinar la eficiencia de un algoritmo nos permite discernir lo que es factible en la implementación de una solución de lo que es imposible. Así, el objetivo del análisis de complejidad es cuantificar las medidas físicas: tiempo de ejecución y espacio de memoria y comparar distintos algoritmos que resuelven un mismo problema.

A lo largo de este capítulo presentaré distintos algoritmos de compresión y la complejidad que involucra el tipo de operaciones que llevan a cabo. La reducida complejidad tanto espacial como temporal del nuevo algoritmo LHE que presento en esta tesis es una de sus virtudes.

### 2.3.1 Complejidad temporal

El tiempo requerido por un algoritmo expresado como una función del tamaño de la entrada del problema se denomina complejidad en tiempo del algoritmo y se denota  $T(n)$ . El comportamiento límite de la complejidad a medida que crece el tamaño del problema se denomina complejidad en tiempo asintótica. De manera análoga se pueden establecer definiciones para la complejidad en espacio y la complejidad en espacio asintótica.

Para denotar la complejidad de un algoritmo se utiliza el orden “O”, siendo  $O(f(n))$  el conjunto de funciones  $g(n)$  acotadas por  $f(n)$  para valores de  $n$  suficientemente grandes. Entendiendo por “suficientemente grandes” valores tan elevados que hacen que las particularidades (programación, etc.) de un algoritmo sean despreciables frente al volumen teórico del cálculo a realizar.

Por ejemplo, decimos que un algoritmo  $f(n)$  tiene un orden  $O(f(n))=O(n)$  si existen un  $n_0$  y un  $c$ , siendo  $c > 0$ , tal que para todo  $n \geq n_0$ ,  $c \cdot n \geq f(n)$

Ejemplo:

Sea  $f(n) = 3n^3 + 2n^2$  y  $g(n) = n^3$ , con  $n > 0$ . Se dice que  $f$  es dominada por  $g(n) = n^3$ , dado que existe una constante  $c$  ( $c=5$ ) tal que  $f(n) \leq c \cdot g(n)$  y por consiguiente  $O(3n^3 + 2n^2) = O(n^3)$

A continuación se enumeran las propiedades básicas de la notación O:

- $k \cdot O(f(n)) = O(f(n))$
- $O(f(n) + g(n)) = \max\{f(n); g(n)\}$
- $O(f(n)) \cdot O(g(n)) = O(f(n) \cdot g(n))$
- $O(O(f(n))) = O(f(n))$

El tiempo de ejecución de cada operación simple, por lo común puede tomarse como  $O(1)$ . Algunas operaciones matemáticas no deben ser tratadas como operaciones elementales. Por ejemplo, el tiempo necesario para realizar sumas y productos crece con la longitud de los operandos. Sin embargo, en la práctica se consideran elementales siempre que los datos que se usen tengan un tamaño razonable.

**Algunos** de órdenes de complejidad que se presentan con frecuencia son:

- lineal:  $n$

- cuadrático:  $n^2$
- polinómico:  $n^k$ , con  $k \in \mathbb{N}$
- logarítmico:  $\log(n)$
- exponencial:  $c^n$

A modo de conocido ejemplo, el famoso algoritmo “quicksort” fue el primer algoritmo de ordenación con un orden  $O(n) = n\log n$

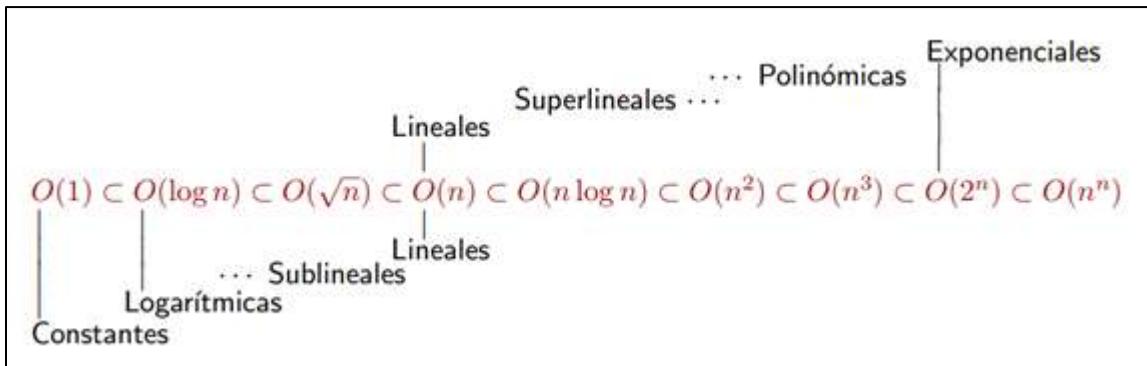


Figura 4. Clasificación de órdenes de complejidad

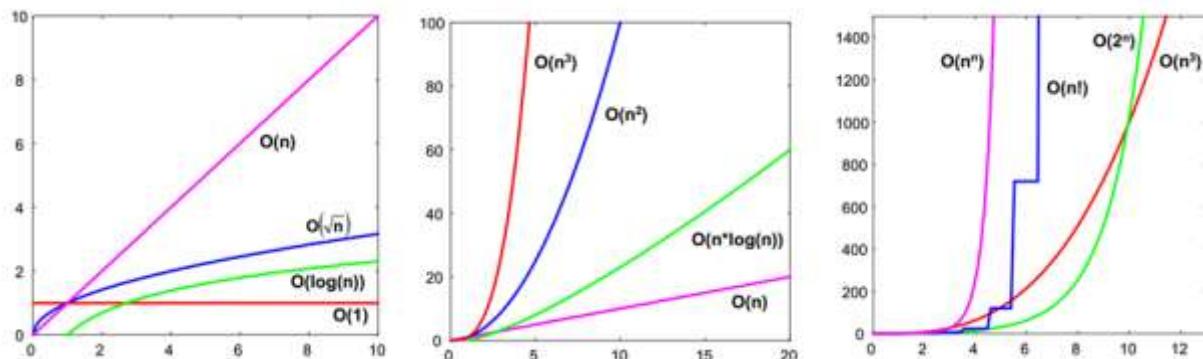


Figura 5. Gráficas comparativas de órdenes de complejidad temporal

A modo ilustrativo en la siguiente tabla se muestran tiempos de ejecución en una máquina que ejecute  $10^9$  operaciones por segundo ( $\approx 1\text{GHz}$ )

Talla	$\log_2 n$	$n$	$n \log_2 n$	$n^2$	$n^3$	$2^n$
10	3.322 ns	10 ns	33 ns	100 ns	1 $\mu$ s	1 $\mu$ s
20	4.322 ns	20 ns	86 ns	400 ns	8 $\mu$ s	1 ms
30	4.907 ns	30 ns	147 ns	900 ns	27 $\mu$ s	1 s
40	5.322 ns	40 ns	213 ns	2 $\mu$ s	64 $\mu$ s	18.3 min
50	5.644 ns	50 ns	282 ns	3 $\mu$ s	125 $\mu$ s	13 días
100	6.644 ns	100 ns	664 ns	10 $\mu$ s	1 ms	$40 \cdot 10^{12}$ años
1000	10 ns	1 $\mu$ s	10 $\mu$ s	1 ms	1 s	
10000	13 ns	10 $\mu$ s	133 $\mu$ s	100 ms	16.7 min	
100000	17 ns	100 $\mu$ s	2 ms	10 s	11.6 días	
1000000	20 ns	1 ms	20 ms	16.7 min	31.7 años	

Figura 6. Tiempos de ejecución de diferentes órdenes

### 2.3.2 Complejidad espacial

La misma idea que se utiliza para medir la complejidad en tiempo de un algoritmo se utiliza para medir su complejidad en el espacio. Decir que un programa es  $O(n)$  en espacio significa que sus requerimientos de memoria aumentan proporcionalmente con el tamaño del problema. Esto es, si el problema se duplica, se necesita el doble de memoria. Del mismo modo, para un programa de complejidad  $O(n^2)$  en espacio, la cantidad de memoria que se necesita para almacenar los datos crece con el cuadrado del tamaño del problema: si el problema se duplica, se requiere cuatro veces más memoria. En general, el cálculo de la complejidad en espacio de un algoritmo es un proceso sencillo que se realiza mediante el estudio de las estructuras de datos y su relación con el tamaño del problema.

### 2.3.3 Selección del mejor algoritmo

La selección de un algoritmo es un proceso es un proceso en el que se deben tener en cuenta muchos factores, entre los cuales podemos destacar los siguientes:

- **El tamaño del problema** que se va a resolver: Un algoritmo para un proceso tamaño pequeño no justifica la realización de un análisis, ya que da lo mismo una implementación que otra
- **La complejidad en tiempo del algoritmo:** Es una primera medida de la calidad de una rutina y establece su comportamiento cuando el número de datos a procesar es grande.
- **La complejidad en espacio del algoritmo:** Si las necesidades de memoria del algoritmo crecen considerablemente con el tamaño del problema el rango de utilidad del algoritmo es baja y se debe descartar. Obviamente este punto depende de los recursos espaciales (memoria) de que dispongamos.
- **Su capacidad de paralelización:** puede que un algoritmo si es paralelizable, permita su ejecución en un tiempo razonable si disponemos de los recursos necesarios para paralelizarlo y en ese caso incluso siendo menos eficiente que otro puede resultar mejor opción.

Normalmente los resultados de los algoritmos de compresión en caso de los compresores de imágenes con pérdidas es un diagrama R-D (Rate-Distortion) donde en el eje de las abscisas se suele representar el bit-rate y en el de las ordenadas la calidad (siendo lo más habitual el PSNR).

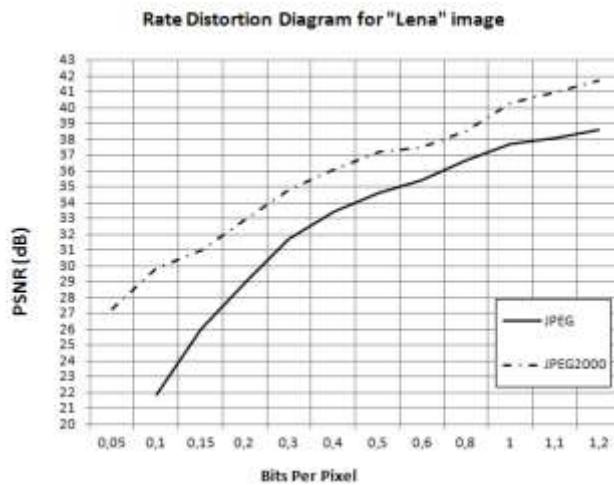


Figura 7. Diagrama de distorsión de dos formatos JPEG y JPEG2000 para una misma imagen

Este tipo de representación nada nos dice del coste computacional de los algoritmos por lo que no podemos concluir que el algoritmo cuya curva sea superior en calidad sea mejor. La adecuación al tipo de aplicación que se pretenda realizar puede depender fuertemente de la complejidad computacional y si se requiere un tiempo de cómputo muy reducido puede que sea preferible un algoritmo de inferior calidad pero de menor complejidad.

Por ese motivo tiene sentido observar los formatos y algoritmos más conocidos con otra perspectiva, donde sea el coste computacional espacial (memoria necesaria para su ejecución) y temporal (operaciones ejecutadas) lo que representemos en abscisas y ordenadas. Esta nueva visión complementaria a la del diagrama R-D puede arrojar luz sobre la idoneidad de un algoritmo para una aplicación concreta. En la Figura 8 se representa este diagrama, en el que se además se incluye LHE, el nuevo algoritmo aportado en esta tesis.

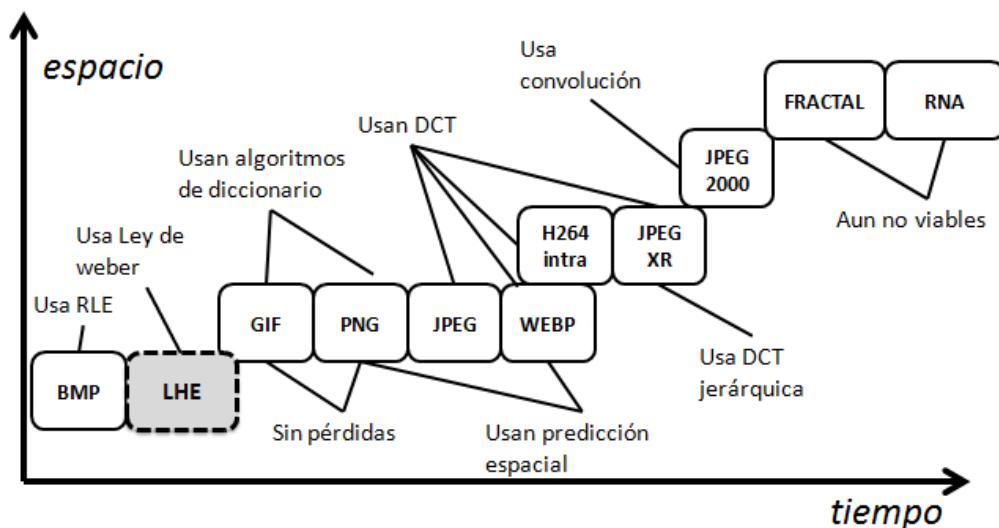


Figura 8. Comparativa computacional de algoritmos/formatos

## 2.4 Algoritmos de compresión genéricos

La compresión máxima a la que puede llegar un algoritmo, sea el que sea, viene fijado por la ley de Shannon, tal y como he enunciado en la introducción a la teoría de la información. Ahora bien, si hablamos de compresión con pérdidas, este límite puede ser claramente superado. Todo investigador de códigos debería tratar de vencer la limitación de Shannon, sin violarla.

Los compresores sin pérdidas se pueden agrupar en

- Algoritmos estadísticos
- Algoritmos basados en diccionario

La diferencia fundamental es que los segundos no requieren realizar un análisis de frecuencias de aparición de los símbolos para escoger el código binario que representará cada símbolo o grupo de símbolos.

Existe un tercer tipo de algoritmos más básicos que también comprimen y que aprovechan un tipo de redundancia muy evidente (repeticiones o similitud entre muestras consecutivas) para transformar los datos originales en una versión reducida de los mismos. Normalmente se utilizan por su simplicidad o bien como pre procesamiento previo de la fuente antes de aplicar un algoritmo de compresión estadístico o basado en diccionario. Los denominaremos “algoritmos de pre-procesamiento”.

### 2.4.1 Algoritmos de pre-procesamiento

#### 2.4.1.1 Run-Length Encoding (RLE)

La compresión RLE [12] (Run-Length Encoding) a veces escrito RLC (Run-Length Coding) es una forma muy simple de compresión de datos en la que secuencias de datos con el mismo valor consecutivos son almacenadas como un único valor más su recuento, es decir, se aprovecha la redundancia de repetición.

El formato BMP es el único tipo de compresión que comprime sólo usando RLE y JPEG lo utiliza de forma efectiva en los coeficientes que quedan después de transformar y cuantificar los bloques en que divide las imágenes.

Si los datos no poseen redundancia de repetición, la compresión RLE puede provocar una expansión de la información que es precisamente lo contrario de lo que persigue, por lo que se debe aplicar con cautela.

#### 2.4.1.2 DPCM (Differential Pulse Code Modulation)

Básicamente consiste en sustituir la fuente  $X_i$  por la fuente  $y_i = (X_i - X_{i-1})$ . Es útil si las muestras vecinas tienden a ser similares como es el caso en audio e imágenes. Las nuevas muestras  $Y_i$  se concentran en torno al cero si hay alta correlación de muestras vecinas, como es el caso de la siguiente imagen, donde se han representado los histogramas de la imagen original (PCM) y de la obtenida por DPCM.

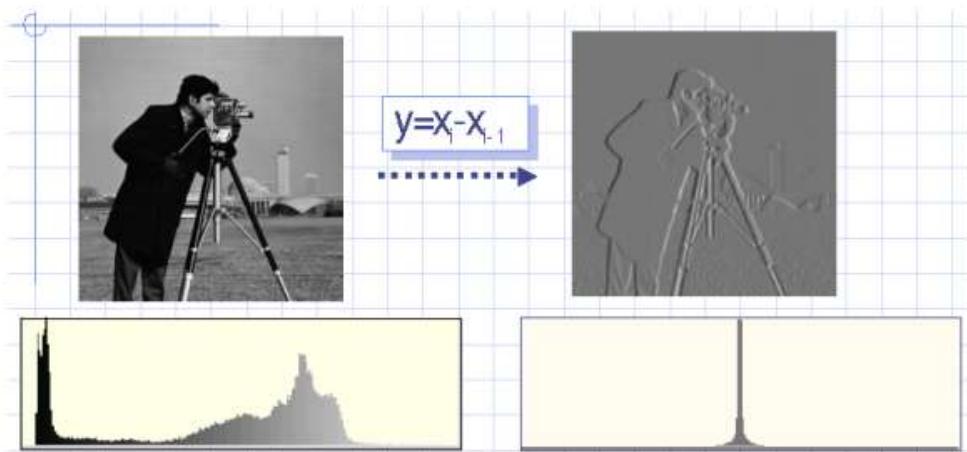


Figura 9. Imagen e histograma preprocesada con DPCM

La clasificación con o sin pérdidas de DPCM en realidad depende del criterio para almacenar  $y_i$  pues si se cuantiza obviamente estaremos ante un algoritmo con pérdidas.

La Modulación Delta (DM) es una variante simplificada de DPCM en el que se usa 1 solo bit para cuantificar las diferencias en aplicaciones de telefonía.

Los sistemas basados en DPCM además suelen incorporar un predictor de la señal, de modo que lo que se codifica es la diferencia entre el valor de la muestra y el valor predicho. A esta nueva señal diferencial se la suele llamar “residuo”. La predicción más simple es considerar que la nueva muestra será igual a la anterior, pero otros mecanismos basados (por ejemplo) en un filtro de mediana son más efectivos pues permite eliminar muestras espúreas o ruido.

#### 2.4.1.3 ADPCM (*Adaptive Differential Pulse Code Modulation*)

El algoritmo ADPCM constituye un compresor completo y no una técnica de pre-procesado, sin embargo he querido incluirlo en esta sección por su estrecha relación con DPCM.

La diferencia fundamental entre DPCM y ADPCM es que el tamaño del escalón considerado en la codificación diferencial es variable y proporcional a la varianza de la señal de entrada, mientras que en DPCM el escalón es constante.

Que sea proporcional a la varianza significa que si la señal de entrada tiene altas fluctuaciones el escalón se hace mayor y si la señal de entrada fluctúa poco, el escalón se hace menor, consiguiendo una mayor precisión en la codificación. Esto además implica que estamos hablando de un codificador con pérdidas, pues los valores digitalizados en PCM se van a codificar con diferente precisión en función del escalón.

Los codificadores adaptativos se denominan “feedforward” si se considera la varianza de la señal de entrada para adaptar el escalón, y en este caso hay que transmitir el tamaño del escalón al receptor pues la señal de entrada no está disponible en recepción. Sin embargo también existe la posibilidad de adaptar el escalón en función de la señal de salida y en ese caso no hay que transmitir ninguna información extra. En ese caso se llaman “feedback”. Los estándares ITU-G721, G726 y G727 son ejemplos de codificadores de voz que utilizan esta técnica.

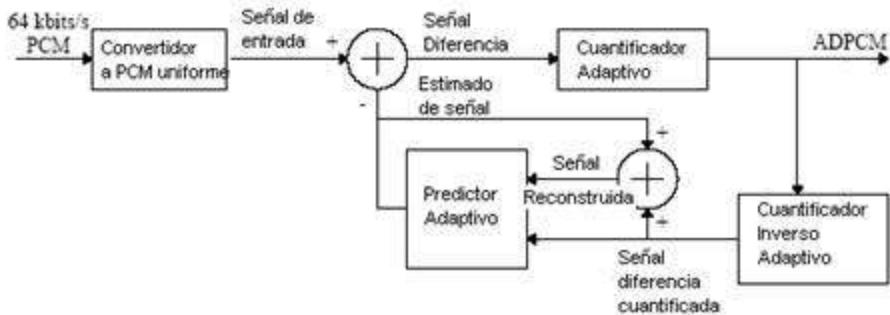


Figura 10. Codificador ADPCM G.721

En codificación de imágenes es posible usar ADPCM y se han realizado evaluaciones al respecto por diversos autores [13] [14], si bien los resultados son inferiores en calidad a los métodos estándar (como JPEG), por lo que no existe ningún formato gráfico estándar que utilice esta técnica.

		<b>LENA</b>	
<b>Technique</b>	<b>Bit rate bits/pixel</b>	<b>RMSE (0-255)</b>	<b>SNR (dB)</b>
1-D DPCM	1.00	18.67	22.71
1-D DPCM	2.00	9.44	28.63
1-D DPCM	3.00	5.11	33.96
2-D DPCM	1.00	14.58	27.74
2-D DPCM	2.00	6.93	31.32
2-D DPCM	3.00	3.71	36.74
1-D ADPCM	1.20	10.91	27.37
1-D ADPCM	2.20	4.37	35.32
1-D ADPCM	3.20	2.16	41.44
2-D ADPCM	1.20	7.84	30.24
2-D ADPCM	2.20	2.87	38.97
2-D ADPCM	3.20	1.37	45.40



Tabla 1. Tabla de resultados de ADPCM sobre imágenes del libro “digital Image compression techniques”

## 2.4.2 Algoritmos estadísticos

Los algoritmos estadísticos requieren conocer la distribución de probabilidades de los símbolos del fichero original o fuente, lo que implica la necesidad de dar dos pasadas, la primera para contar las frecuencias de aparición de cada símbolo y la segunda para realizar la sustitución de los símbolos por los códigos binarios de longitud variable calculados.

### 2.4.2.1 Huffman

Huffman se utiliza como parte de los procedimientos a llevar a cabo en un gran número de formatos de compresión como JPEG, PKZIP o MP3.

Este es un algoritmo de compresión de los llamados “estadísticos”, los cuales calculan el código óptimo

utilizando la probabilidad de los símbolos. El algoritmo Huffman [15] distribuye el numero de bits empleados para representar un símbolo, asignando una menor cantidad de bits a los símbolos de mayor frecuencia de aparición y una mayor cantidad de bits a los símbolos que menos aparecen.

El árbol del algoritmo de Huffman se construye efectuando los siguientes pasos:

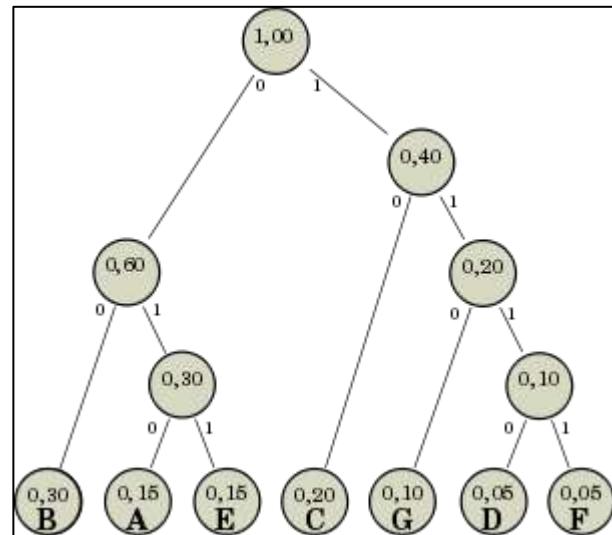
- Se comienza con tantos nodos como símbolos y se asigna a cada nodo un peso igual a la probabilidad del símbolo
- Se ordenan los símbolos de menor a mayor peso
- Se crea un nodo padre para los dos nodos de menor peso y se asigna a dicho nodo un peso igual a la suma de los pesos de los hijos

Se repiten los pasos 1 y 2 hasta que solo quede un nodo con peso 1 (dicho peso es la suma de todas las probabilidades)

Una vez que hemos construido el árbol de nodos y símbolos, es el momento de asignar los códigos binarios. Para ello recorremos el árbol desde el nodo raíz hasta cada símbolo, asignando por el camino un “0” a los enlaces a la izquierda y un “1” a la derecha (la asignación inversa también sirve). El recorrido hasta cada símbolo determina el código en bits asociado a cada símbolo.

Suponiendo el siguiente ejemplo de estadística de símbolos en un mensaje,

Símbolo	probabilidad
A	0,15
B	0,3
C	0,2
D	0,05
E	0,15
F	0,05
G	0,1



El árbol de nodos que se obtendría siguiendo el algoritmo de Huffman sería el que se muestra a continuación

Por ejemplo el código para el símbolo “D” sería definido por el recorrido desde el nodo raíz, es decir 1110

Figura 11. Construcción del árbol de Huffman

La codificación Huffman es óptima cuando la probabilidad de cada símbolo de entrada es una potencia negativa de dos. Los códigos prefijos tienden a ser ligeramente ineficientes en alfabetos pequeños, donde las probabilidades normalmente se encuentran entre esos puntos óptimos. Es decir, la principal limitación de Huffman es que los símbolos deben codificarse con un número entero de bits, y puede haber una diferencia entre la probabilidad de ese símbolo y la longitud en bits escogida en relación con el resto de símbolos.

El peor caso para una codificación Huffman puede darse cuando la probabilidad de un símbolo excede

0.5. Estas situaciones a menudo se pueden comprimir mejor usando RLC (Run Length Code).

Obviamente este algoritmo requiere dos pasadas: la primera para tomar cuenta del número de apariciones de cada símbolo, es decir, para construir al final la tabla de probabilidades, y tras ejecutar el algoritmo, una segunda pasada en la que hacemos la sustitución de cada símbolo por su nuevo código.

La complejidad de Huffman es  $\log(N)$  siendo  $N$  el número de símbolos por lo que si el número de símbolos es reducido, su ejecución no supone apenas coste. Lo único que supone cierto coste es la primera pasada para contar las frecuencias de aparición de los símbolos.

#### 2.4.2.2 *Huffman adaptativo*

Como he mencionado al describir el algoritmo Huffman, éste requiere dos pasadas para llevarse a cabo. La primera es utilizada para calcular las frecuencias de aparición de cada símbolo. Con Huffman adaptativo se elimina la necesidad de la primera pasada, construyendo y modificando el árbol a medida que se va leyendo el mensaje. Además de ser más rápido, esto permite una mejor asignación de códigos ya que aprovecha las diferencias locales de probabilidad de símbolos y por lo tanto permite una mayor compresión de datos. Se ha demostrado que cuanto mayor es el fichero, más adecuada es esta técnica pues aprovecha mejor la localidad [16].

Un árbol Huffman, además de ser un árbol binario completo (cada nodo del árbol tiene cero o dos hijos) cumple dos propiedades. La primera es sencilla y dice que todo nodo no hoja tiene por frecuencia la suma de las frecuencias de sus dos hijos. La segunda es la propiedad del sibling. Esta segunda propiedad nos dice que si A y B son dos nodos del árbol y se cumple que  $\#(A) < \#(B)$  entonces la frecuencia de A es mayor o igual a la de B

Para actualizar el árbol debemos comprobar si cumple con las dos propiedades. En caso de que no cumpla alguna de las propiedades debemos modificarlo para que vuelva a ser válido

Existen ligeras variaciones de implementación de este esquema, siendo las más extendidas las de FGK [17] (Faller-Gallager-Knuth) y la de Vitter [18], conocido como algoritmo "V", algo mejor (aunque más lento) que FGK.

La implementación de Vitter ("V") actualiza el árbol dinámicamente manteniendo su altura al mínimo posible. Ello conlleva menos operaciones que recalcular el árbol de Huffman por cada símbolo pero es más lento que FGK, pues este último no chequea todo el árbol cuando actualiza un peso sino que solo lo modifica parcialmente, mejorando en velocidad aunque no se consiga la misma eficacia que con Vitter

Los resultados dependen mucho del fichero a tratar, pero experimentalmente (y según el mismo Vitter) no se observan grandes diferencias con Huffman estático tal como se aprecia en estos resultados

<i>n = file size in 8-bit bytes,</i>				
<i>k = number of distinct messages.</i>				
simulation results for a small text file				
-----				
n	k	static	Alg. V	Alg. FGK
100	96	83.0	71.1	82.4
500	96	83.0	80.8	83.5
961	97	83.5	82.3	83.7
simulation results for Pascal source code				
-----				
n	k	static	Alg. V	Alg. FGK
100	32	57.4	56.2	58.9
500	49	61.5	62.2	63.0
1000	57	61.3	61.8	62.4
10000	73	59.8	59.9	60.0
12067	78	59.6	59.8	59.9

Tabla 2. Resultados de Huffman adaptativo

Como conclusión podemos decir que la ventaja más importante de Huffman adaptativo frente a Huffman estático no es una mayor compresión (los resultados expuestos son similares) sino la eliminación de la necesidad de la primera pasada sobre el fichero que requiere Huffman estático para construir la tabla de probabilidad de los símbolos. Sin embargo, es más costoso que Huffman estático porque requiere la modificación del árbol mientras se asignan los códigos, algo que Huffman estático no necesita hacer. En definitiva, no supone una gran ventaja.

#### 2.4.2.3 Codificación aritmética

Al igual que Huffman, la codificación aritmética [19] (también llamada “codificación por rango”) requiere el conocimiento a priori de la distribución de probabilidades de los símbolos (aunque existen versiones adaptativas). Pero a diferencia del Huffman adaptativo, la codificación aritmética puede suponer ahorros significativos y también puede ser adaptativa.

La ventaja de la codificación aritmética reside en que a diferencia de Huffman, puede ser óptima aunque las probabilidades de los símbolos no sean una potencia negativa de 2, acercándose más al límite teórico de la compresión máxima (entropía) cuando Huffman no puede. Esto, dicho en otras palabras, significa que la codificación aritmética no tiene la limitación de que cada símbolo sea un número entero de bits y ello permite acercarse más al límite teórico de compresión del teorema de la codificación sin ruido.

En la codificación aritmética no se asigna una palabra de código a cada uno de los símbolos del alfabeto fuente como se hace en las técnicas anteriormente vistas. En esta técnica lo que se hace es codificar una secuencia de entrada de símbolos del alfabeto fuente mediante un número representado en punto flotante, es decir, se asigna una sola palabra código aritmética a una secuencia completa de símbolos fuente. En pocas palabras lo que se hace es multiplicar las probabilidades de un conjunto de símbolos y codificar el número resultante.

Cuando se transmite un valor al receptor es necesario indicar cuantos símbolos se han enviado para que el receptor sepa cuándo debe finalizar el proceso de descompresión. Para indicar esto existen dos técnicas. La primera consiste en enviar antes del valor elegido el número de símbolos que se han comprimido en binario. De esta forma el receptor sabrá cuantos símbolos han sido comprimidos antes

de comenzar el proceso de descompresión. La segunda técnica consiste en utilizar en el alfabeto un símbolo más con una probabilidad asociada. Este símbolo será un símbolo especial que indicará el final de una secuencia de símbolos y comprimidos será comprimido por el emisor después del último símbolo que se quiere comprimir. El receptor sabrá cuando debe finalizar el proceso de descompresión cuando al interpretar el último símbolo descomprimido este símbolo sea un símbolo especial.

La Figura 12 ilustra el proceso básico de la codificación aritmética [20]. En este caso, se codifica una secuencia o mensaje de cinco símbolos,  $a_1a_2a_3a_3a_4$ , generados por una fuente de cuatro símbolos. Al principio del proceso de codificación, se supone que el mensaje ocupa todo el intervalo semi-aberto  $[0,1)$ .

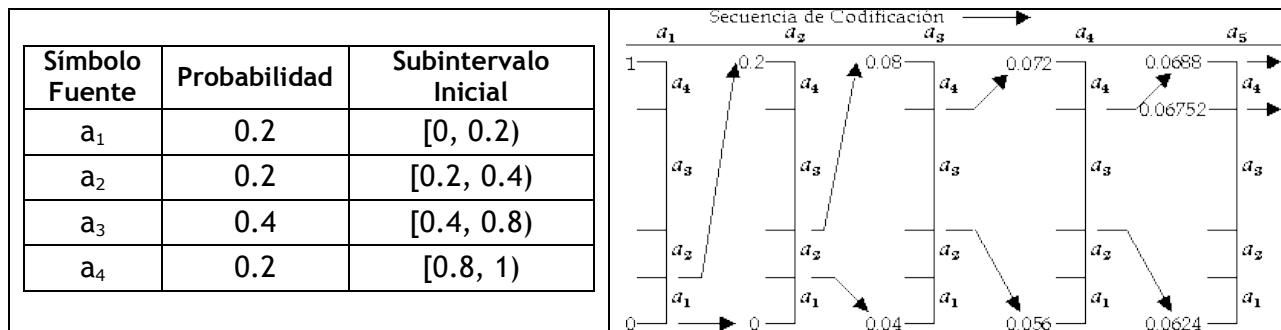


Figura 12. Codificación aritmética

Este intervalo se subdivide inicialmente en cuatro regiones en función de las probabilidades de cada símbolo de la fuente. Por ejemplo, se asocia el sub-intervalo  $[0, 0.2)$  al símbolo  $a_1$ . Puesto que se trata del primer símbolo del mensaje a codificar, el intervalo del mensaje se reduce inicialmente a  $[0, 0.2)$ . Así el intervalo  $[0, 0.2)$  abarca toda la altura de la figura y se marcan los extremos con los valores del rango reducido. Posteriormente, se divide este rango reducido de acuerdo con las probabilidades de los símbolos de la fuente original, y el proceso continúa con el símbolo del mensaje. De esta forma, el símbolo  $a_2$  reduce el sub-intervalo a  $[0.04, 0.08)$ ,  $a_3$  lo reduce aún más, dejándolo en  $[0.056, 0.072)$ , y así sucesivamente. El último símbolo del mensaje, que se debe reservar como indicador especial de fin de mensaje, reduce el intervalo, que pasa a ser  $[0.06752, 0.0688)$ . Por supuesto, se puede utilizar cualquier número que esté dentro del sub-intervalo, como por ejemplo el 0.068, para representar el mensaje.

El cálculo de la reducción de los intervalos se hace de la manera siguiente (tomamos como ejemplo los dos primeros símbolos del ejemplo):

Una vez que se codifica el primer símbolo, el intervalo del mensaje se reduce a  $[0, 0.2)$ . Para codificar el segundo símbolo hay que tener en cuenta que nuestro nuevo rango de trabajo es el establecido por el primer símbolo, y por tanto, el rango del segundo símbolo  $[0.2, 0.4)$  se refiere a porcentajes en el rango actual. Es decir la codificación del segundo símbolo tendrá el rango:

- Límite inferior:  $(\text{amplitud rango} * \text{rango inferior}) + \text{valor inferior}$
- Límite superior:  $(\text{amplitud rango} * \text{rango superior}) + \text{valor inferior}$

Donde: amplitud rango es la amplitud del rango actual (el establecido por el símbolo que acabamos de codificar anteriormente), rango inferior y rango superior se refieren a los límites superior e inferior del rango del símbolo que queremos codificar y valor inferior se refiere al límite inferior del rango actual.

En el proceso de decodificación se utiliza la tabla original en la que se habían asignado unos intervalos

iniciales a cada símbolo.

El receptor recibe a través del canal un número decimal en punto flotante. Lo primero que se hace es comprobar en la tabla a que intervalo pertenece ese valor y el símbolo al que se le ha asociado ese intervalo será el primer símbolo que se ha transmitido. A continuación se le resta al número recibido el extremo inferior del intervalo al que pertenece, y el resultado se divide por la longitud del intervalo. Con el número obtenido repetimos el proceso anterior. Este proceso se repite hasta que todos los símbolos del mensaje hayan sido procesados.

En el mensaje codificado aritméticamente del ejemplo considerado, se utilizan tres dígitos decimales para representar el mensaje de cinco símbolos. Esto se traduce en 3/5, ó 0.6 dígitos decimales por símbolo fuente, lo que se aproxima bastante a la entropía de la fuente, que resulta ser de 0.58 dígitos decimales por símbolo. Conforme aumenta la longitud de la secuencia a codificar, el código aritmético resultante se aproxima a límite establecido por el teorema de la codificación sin ruido.

El siguiente paso es para codificar este número decimal es utilizar un número binario de punto fijo de precisión suficiente para recuperarlo

En la práctica, existen dos factores que hacen que el rendimiento de la codificación se aleje de este límite:

- La inclusión del indicador de fin de mensaje, necesario para separar un mensaje de otro.
- La utilización de aritmética de precisión finita.

Se ha comprobado que la codificación aritmética produce un compresión del orden del 5-10% mejor que la codificación Huffman, pero tiene como inconveniente una mayor complejidad computacional, aun siendo del mismo orden  $N \log(N)$ .

El motivo por el que Huffman está mucho más extendido es porque es muy simple y no sujeto a patentes, pero la mayoría de patentes de la codificación aritmética ya han expirado y ello significa que podrá usarse libremente en muchos algoritmos.

#### 2.4.3 Algoritmos de diccionario (Lempel-Ziv)

Para los códigos estadísticos (e.g., Huffman, aritmético) es necesario conocer la distribución de probabilidades de la fuente o el fichero original, lo que implica la necesidad de dar dos pasadas o utilizar estadísticas medias.

Los algoritmos de diccionario (también llamados “sustitucionales”) se fundamentan en la idea básica de reemplazar una ocurrencia de una frase o secuencia de símbolos por una referencia a una aparición previa de dicha frase.

La compresión por diccionario no usa modelos de predicción estadística para determinar la probabilidad de ocurrencia de un símbolo particular, sino que almacena cadenas de símbolos de entrada en un diccionario. A diferencia de Huffman, la compresión basada en diccionario no requiere dos pasadas.

El algoritmo principal del que han surgido diversas variantes (como LZ77, LZ78, LZW y LZMW) se conoce como Lempel-Ziv (LZ). Ha sido utilizado en el formato “GIF” de compresión de imágenes así como en otras conocidas herramientas como ZIP, compress, o Arj.

Suponiendo que el fichero a comprimir es de texto, la idea básica es evitar repetir trozos de texto y en su lugar indicar la localización de ese trozo en el texto previo y la longitud de ese fragmento.

Ejemplo:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
t	h	e		o	t	h	e	r		o	n	e	
15	16	17	18	19	20	21	22	23	24	25	26	27	
i	s		t	h	e		o	l	d	e	s	t	

La compresión de esta secuencia resulta ser:

"the o{1,3}r[4,2]n{3,2}is{4,1}{1,5}ld{3,1}{16,1}{1,1}"

En este ejemplo el primer {1,3} significa "the" pues indica los 3 primeros caracteres comenzando en la posición 1. Y así se han de interpretar todos los pares, como el {16,1} que significa "s".

Los códigos binarios con los que se codifican los pares {x,y} pueden ser códigos Huffman, ya que lo que hemos hecho ha sido reemplazar símbolos individuales por nuevos símbolos que representan cadenas de símbolos y la transformación a códigos binarios puede ser efectuada con Huffman, de hecho el algoritmo "Deflate" precisamente hace eso y se utiliza en ZIP.

El algoritmo en pasos sería:

1. Inicializar un diccionario con todos los símbolos de longitud unitaria (pej, las letras de un alfabeto o los 255 bytes)
2. Busque en el fichero original la palabra (W) más larga que ya exista en el diccionario
3. Añada W seguido por el siguiente símbolo encontrado al diccionario
4. Codifique W por su índice de entrada en el diccionario
5. Volver a paso 2

Si en algún momento se agotan los índices de N bits en el diccionario, se comienza a codificar todo con códigos de N+1 bits, incluso lo que ya está en el diccionario. Si N llega a ser elevado entonces el proceso se reinicia. En la práctica el tamaño del diccionario se suele limitar a no mas de 12 bits por índice, aunque ello es el motivo por el que se aleja del límite teórico de compresión máxima (entropía)

En resumen, LZ aprovecha la redundancia de repetición de secuencias de símbolos (palabras). Y si dicha redundancia es abundante, puede comprimir mucho.

En general el algoritmo LZ funciona mejor que Huffman adaptativo y tiene un rendimiento similar a la codificación aritmética.

	Adaptive Huffman	Lempel-Ziv
LaTeX file	66%	44%
Speech file	65%	64%
Image file	94%	88%
<i>Size of compressed file as percentage of the original file</i>		

Tabla 3. Comparativa entre Huffman y Lempel-Ziv

En análisis comparativos [21] se ha comprobado que depende del tipo de fuente que el algoritmo idóneo sea uno u otro como evidencia la siguiente tabla de resultados de compresión de imágenes con diferentes métodos. En esta tabla un resultado mayor es mejor pues es  $(\text{tamaño comprimido}) / (\text{tamaño original})$ .

Image	size	Huffman	Run Length	Arithmatic	LZW
dictionary size = 512					
Face	32 x 32	1.1034	0.29889	1.0049	0.9258
Keyboard	48 x 48	1.1791	0.2321	1.0034	0.9446
Airpacific	64 x 64	3.0704	1.1145	2.2667	2.7788
Vegitables	128 x 128	1.0435	0.2283	1.0015	0.9686
Einstein	153 x 153	1.1244	0.2348	1.0013	0.9683
Horses	160 x 160	1.0479	0.2326	1	0.973
Moon	240 x 240	1.389	0.3481	1.1179	1.1635

Tabla 4. Resultados de compresión de imágenes con distintos algoritmos

## 2.5 Algoritmos y formatos de compresión de imágenes

El formato de imágenes sin comprimir RAW, sólo se encuentra disponible en cámaras digitales profesionales. Este formato ofrece la máxima calidad ya que contiene los píxeles en bruto tal y como se han adquirido por el sensor fotográfico. Sin embargo manipular las imágenes en formato RAW tiene muchos inconvenientes, fundamentalmente por el gran tamaño de los ficheros.

La compresión de imágenes, además de beneficiarse de los algoritmos de compresión genéricos, puede aprovechar las características tanto del sistema visual humano como la redundancia espacial presente en cualquier imagen.

En esta sección voy a presentar los formatos y algoritmos de compresión de imágenes más extendidos (con y sin pérdidas), con un mayor énfasis en JPEG por ser el formato más extendido y con más aplicaciones. Conocer en profundidad estos algoritmos nos permitirá comparar y evaluar las ventajas computacionales del nuevo algoritmo LHE que presento en esta tesis, y por ello es imprescindible hacer una revisión de los algoritmos y formatos vigentes.

### 2.5.1 BMP

El formato BMP no presenta compresión alguna, salvo opcionalmente la ineficaz RLE. Esto los hace inadecuados en páginas web debido a su gran tamaño en relación a su resolución.

### 2.5.2 TIFF

El formato TIFF (Tagged Image File Format) fue desarrollado en 1987 por Aldus (ahora pertenece a Adobe). Las últimas especificaciones (Revisión 6.0) se publicaron en 1992. A diferencia de BMP permite compresión entre varios algoritmos (Paquete de bits / CCITT G3y4 / RLE / JPEG / LZW / UIT-T) por lo que TIFF no define un algoritmo de compresión específico sino simplemente un formato de fichero o “contenedor”.

Tiene una ventaja interesante y es que permite usar diferentes espacios de color (RGB, CMYK, CIELAB y YCbCr). Al almacenar un archivo en formato TIFF, este lo guarda con 48 bits de color incluyendo capas y canales alfa.

En resumen, TIFF no es un formato con pérdidas o sin pérdidas, sino que depende del método de

compresión escogido y declarado en la cabecera del archivo.

### 2.5.3     GIF

El formato GIF (“Graphics Interchange Format”) [22] fue desarrollado por Compuserve en 1987 para imágenes de no más de 256 colores, de modo que si tratamos de comprimir una imagen de mayor gama de color lo primero que hará GIF es cuantizarla a sólo 256 posibles colores. En el caso particular en que la imagen sólo posea luminancia ello no va a suponer ninguna degradación.

El núcleo de GIF es una compresión de diccionario LZW, que es una variante de Lempel-Ziv. Se trata pues de una compresión sin pérdidas (en el caso de que la imagen original tenga menos de 256 colores).

Lempel-ziv aprovecha las redundancias de repetición de secuencias de símbolos y ello es muy eficiente en imágenes de pocos colores pues las líneas horizontales poseen una alta redundancia y se pueden reaprovechar muchas cadenas de pixels sin necesidad de volver a codificar todos sus símbolos correspondientes.

Existen dos versiones de este formato de archivos desarrolladas en 1987 y 1989 respectivamente:

El GIF 87a, que es compatible con la compresión LZW, puede entrelazar, (permitir la visualización progresiva) una paleta de 256 colores y tiene la posibilidad de crear imágenes animadas (llamadas GIF animados) almacenando varias imágenes en el mismo archivo.

El GIF 89a, que tiene como agregado la posibilidad de designar un color transparente para la paleta y especificar el tiempo de las animaciones.

Dado que el algoritmo de compresión LZW estaba patentado, todos los editores de software que usaban imágenes GIF debían pagarle derechos de uso a Unisys, la compañía propietaria de los derechos. Esta es una de las razones por las que el formato PNG se está volviendo cada vez más popular, en perjuicio del formato GIF.

### 2.5.4     PNG

Las motivaciones para crear el formato PNG se generaron a partir de 1995, después de que la compañía Unisys anunciara que haría cumplir la patente de software del algoritmo de compresión de datos LZW utilizado por el GIF. Además por entonces los ordenadores ya soportaban millones de colores y GIF está limitado a 256 colores. Tal como había anunciado, en 1999 Unisys puso fin a su política de licencias de patente libres de derechos para los desarrolladores de software libre o no comercial.

El significado de PNG es “PNG no es GIF” (PNG's Not GIF), haciendo alusión a la tradición de siglas recursivas de GNU.

La especificación de la versión 1.0 de PNG fue lanzada el 1 de julio de 1996 y después apareció como RFC 2083. PNG se convirtió en una recomendación W3C el 1 de octubre de 1996 [23] y a partir de 2004 es un estándar internacional (ISO/IEC 15948:2004).

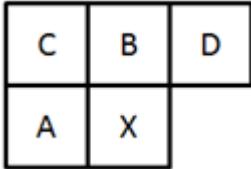
Dos diferencias significativas con GIF (aparte del soporte de 16M colores):

- A diferencia de la transparencia ofrecida por GIF que sólo puede tomar 2 valores (completamente transparente o completamente opaco), el canal alfa de PNG permite utilizar mayor profundidad de bits para lograr efectos de semi-transparencia, propios de objetos translúcidos. Por ejemplo, con una profundidad de 8 bits para transparencias se pueden conseguir 256 grados diferentes de transparencia, como si se tratara de un color.

- Aunque GIF soporta animación, el PNG se desarrolló como un formato de imagen estático y se creó el formato MNG (Multiple Image Network Graphics) como su variante animada.

El algoritmo de PNG se basa en los siguientes 2 pasos para comprimir una imagen:

- Primeramente efectúa un “filtrado predictivo” que consiste en transformar la imagen para prepararla para su posterior compresión. Esta transformación consiste en cambiar los datos originales por el error sobre una predicción. Hay 5 tipos de predicción y el codificador elige un tipo por cada scanline de la imagen en función de cual se adapta mejor. Al codificar cada scanline, ésta es precedida del tipo de filtro predictivo escogido. Para determinar que predicción es la mejor, simplemente se ejecutan todas y se escoge la que se ha adaptado mejor. Como presentaré en el capítulo 8 sobre líneas de trabajo futuro, algo similar se podría hacer en LHE con coste computacional cero.



$X$ = valor original del pixel  
 $X'$ = predicción  
(PNG codifica  $X-X'$ )

Tipo	Predicción de PNG
Ninguno	Ninguna: $X'=0$
Sub	$X'=A$
Up	$X'=B$
Average	$X'=(A+B)/2$
Paeth	Se escoge el valor (A, B o C) más cercano a $(A + B - C)$

Tabla 5. Filtros predictivos de PNG

El más interesante de todos (y más utilizado) es el Paeth [24] pues se adapta bien a bordes verticales y horizontales como muestra la siguiente figura, de un modo sencillo pero ingenioso.

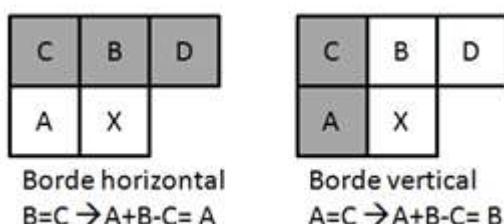


Figura 13. Filtro Paeth detectando bordes

- A continuación se comprime usando el algoritmo “Deflate” (usado en PKZIP), el cual se basa en dos pasos:
- Aplicar un LZ77 por “bloques” de longitud variable y de tamaño máximo 64KB. Estos bloques no son bidimensionales, son simplemente secuencias de símbolos del fichero tras su transformación por el filtrado predictivo.
- Codifica con Huffman estático los bloques comprimidos por LZ77. Por consiguiente, cada bloque irá precedido por una tabla estática de Huffman

## 2.5.5 JPEG

El formato JPEG [25] fue creado en 1994 como resultado de un proceso que se inició en 1986. Aunque normalmente es considerado una única especificación en realidad se compone de 4 partes, la primera de las cuales describe los algoritmos fundamentales (“baseline JPEG”) y el resto son extensiones y otras características especiales [26]. El núcleo de JPEG es la transformada discreta del coseno (DCT), siendo JPEG el ejemplo más característico de los compresores basados en transformación (“transform compression”) [27].

La “baseline JPEG” se compone de 3 etapas: una transformada discreta de coseno (DCT), una etapa de cuantización y finalmente una etapa de compresión Huffman (codificador de entropía).

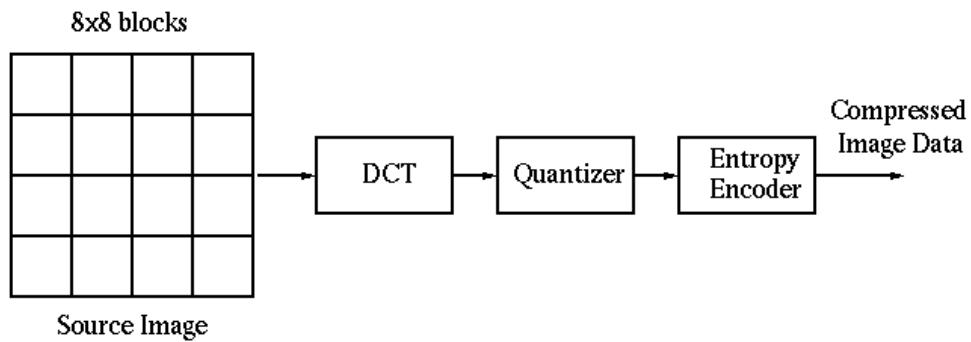


Figura 14. Etapas de JPEG

En el algoritmo JPEG, el 60% del coste computacional lo consume la etapa DCT [28], por ello se han desarrollado muchas ideas (parallelización, computación con números enteros, etc.) para tratar de reducir este coste.

### 2.5.5.1 Primera etapa: DCT

La codificación por transformación por DCT se utiliza en el estándar de compresión de imágenes JPEG y, también, en los estándares de compresión de vídeo H.261, H.263, MPEG, MPEG-2 y MPEG-4 (H264).

El fundamento de la DCT es el mismo que el de la DFT. Multiplicando cada muestra de la señal original por la muestra correspondiente de una señal sinusoidal e integrando (sumando) los resultados se obtiene un coeficiente que nos indica la amplitud del armónico de la frecuencia considerada que está “contenido” en la señal original. Si dicho armónico no está presente, entonces los términos positivos se anulan con los negativos y el coeficiente resulta ser nulo.

La ventaja de la DCT respecto la DFT [29] es que la DCT genera el espectro de una señal simétrica en el tiempo respecto del intervalo considerado, de manera que no se produce una discontinuidad brusca entre el principio y el final del intervalo y por ello no surgen armónicos de alta frecuencia y además las frecuencias resultantes de la transformación son más bajas. En la siguiente figura se muestra el armónico fundamental de una misma señal mediante una DFT y una DCT y se observa como la frecuencia fundamental usando DCT es la mitad que la de DFT.

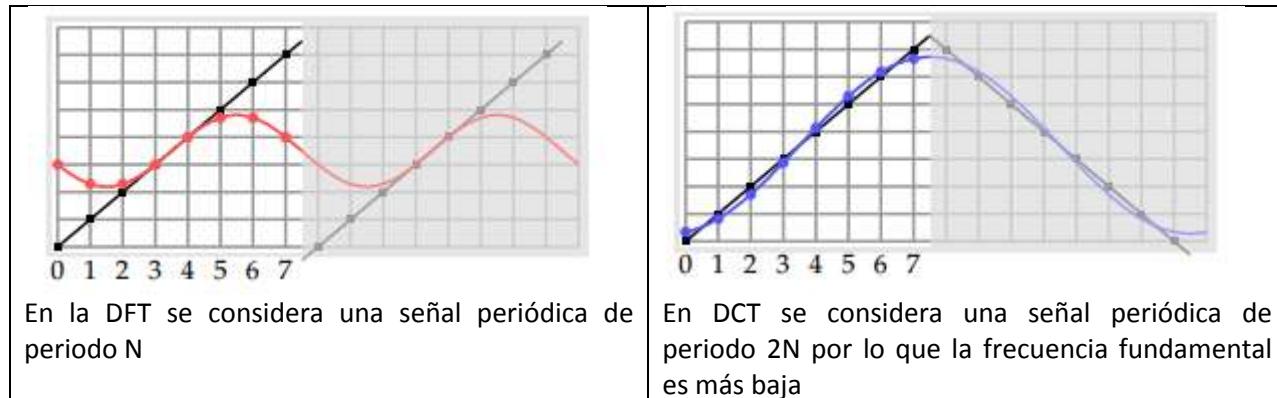


Figura 15 DFT y DCT

El hecho de que los armónicos resultantes sean de menor frecuencia resulta en una mejor compactación de la energía en las frecuencias bajas del espectro como se muestra a continuación.

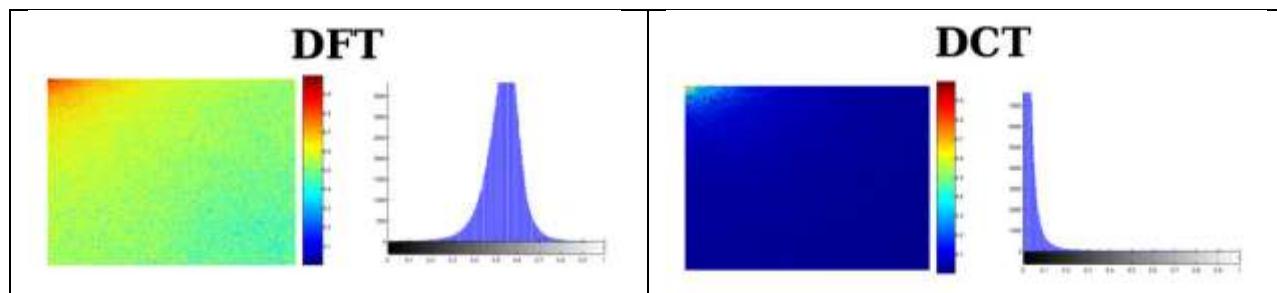


Figura 16. Compactación de la energía de una DCT en comparación con una DFT para una misma imagen

Consideremos el caso unidimensional. La siguiente ecuación permite calcular el coeficiente para un armónico “n” en una señal con N muestras.

$$F(n)=C(n) \sum_{x=0}^{N-1} f(x) \cos\left[(2x+1)n \frac{\pi}{2N}\right] \quad C(n)=\begin{cases} \frac{1}{\sqrt{2}} & n=0 \\ 1 & n \neq 0 \end{cases}$$

Ecuación 4. Transformada 1d-DCT

Esta fórmula indica que para cada armónico, identificado por “n”, tenemos que realizar N operaciones, tantas como muestras de la señal original. Cada operación está compuesta de 3 multiplicaciones, una suma y un coseno. Si contemplamos M posibles armónicos diferentes en la transformación, ello supone  $M \cdot N$  operaciones.

Si consideramos el mismo número de armónicos (valores de n) que de muestras ( $M=N$ , lo cual tiene sentido pues la frecuencia máxima no puede ser superior a un ciclo por cada 2 muestras), el número de operaciones será  $N \cdot N$  y por lo tanto estamos ante un algoritmo de complejidad  $O(N) = N^2$ .

La solución básica utilizada en JPEG para reducir la complejidad de la DCT consiste en limitar a un umbral el número de muestras en el que se aplica el algoritmo, de modo que si la señal original tiene más muestras, se subdivide en bloques. En JPEG se consideran bloques pequeños de 8x8 muestras ( $N=16$ ). El problema de este “truco” es que al limitar los armónicos presentes en los bloques adyacentes (que es lo

que se hace para comprimir la imagen), la señal reconstruida se hace discontinua en las fronteras y se produce un “efecto de bloques”, más apreciable cuanto más intensa es la compresión.



Figura 17. Efecto de bloques de JPEG

En el caso bidimensional (=una imagen), tendremos que considerar 8 armónicos horizontales y 8 verticales, resultando un total de  $M \cdot M = 8 \cdot 8 = 64$  armónicos bidimensionales (llamados “funciones base”). Las ecuaciones teóricas de la FDCT (Forward DCT) y de la IDCT (Inverse DCT) bidimensionales (2d-DCT) considerando bloques de 8x8 son:

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

$$f(x,y) = \sum_{u=0}^7 \frac{C(u)}{2} \sum_{v=0}^7 \frac{C(v)}{2} F(u,v) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

Ecuación 5. Ecuaciones 2d-FDCT y 2d-IDCT

Cada uno de los coeficientes a calcular se corresponde con las frecuencias que en la siguiente figura se representan de forma gráfica. Para propósitos de ilustración, un gris neutro representa cero en esta figura, el blanco representa amplitudes positivas, y el negro representa amplitudes negativas.

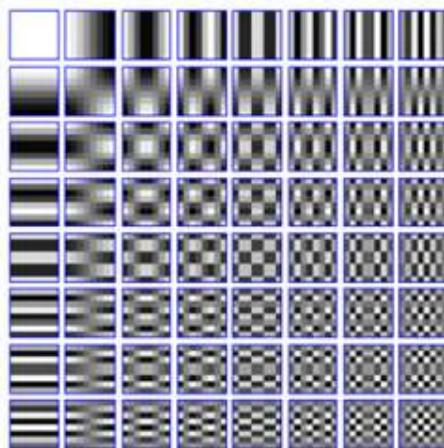


Figura 18. 64 Funciones base de la DCT

Una vez calculados los coeficientes correspondientes a cada uno de los armónicos se habrán transformado los 64 píxeles originales en 64 coeficientes correspondientes a los 64 armónicos considerados. La primera frecuencia (superior izquierda) es realmente una frecuencia nula, por lo que el coeficiente representa el brillo medio y se le denomina coeficiente “DC”, por analogía con la corriente continua, mientras que a los demás 63 coeficientes se les denomina “AC”, por analogía con la corriente alterna.

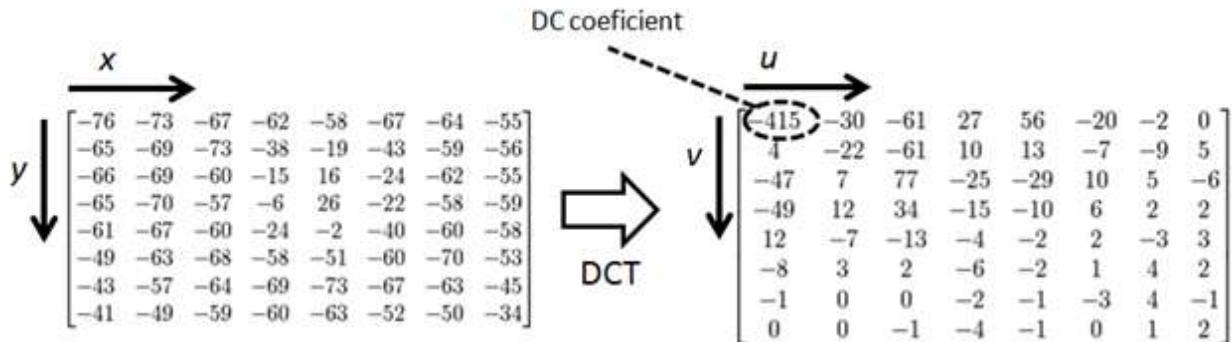


Figura 19. Ejemplo de luminancias y coeficientes tras la transformación 2d-DCT

Calcular en un bloque la transformación para cada armónico involucra 64 operaciones complejas compuestas por dos sumas y 8 multiplicaciones cada una, es decir 640 operaciones simples por armónico, por lo que un bloque de 8x8 pixels involucraría un total de 64 armónicos x 640 = 40.960 operaciones.

Como veremos a continuación es posible ahorrar muchas operaciones tanto en el caso unidimensional como en el bidimensional gracias a las siguientes importantes propiedades matemáticas de la DCT

- Todos los cosenos que dan como resultado cero ahorran muchas operaciones, reduciendo el número de operaciones necesarias.
- La DCT es una función separable (al igual que otras operaciones útiles en imágenes como el downsampling o la interpolación), por lo que para hacer una 2d-DCT podemos primero hacer la DCT considerando solo armónicos horizontales y a continuación considerar los verticales [30]. Es decir, la DCT se puede calcular en dos pasos aplicando una transformación unidimensional en cada uno. En los estándares de compresión MPEG y JPEG se calcula la 2d-DCT de bloques 8x8 calculando primero una 1d-DCT por filas y luego otra 1d-DCT por columnas.

Por estos dos motivos no es una buena idea implementar directamente las fórmulas teóricas presentadas anteriormente ya que no aprovechan las consideraciones planteadas. Voy a presentar a continuación la forma de aprovechar la primera consideración con un ejemplo de la 1d-DCT reducido a 4 muestras pero ilustrativo.

Si observamos la DCT unidimensional se puede ver que la misma puede ser expresada como la multiplicación del vector de datos  $f$  por una matriz de transformación  $T$ .

$$F = T \cdot f$$

Donde la matriz  $T$  tiene la propiedad de poder descomponerse como el producto de dos matrices con menor cantidad de coeficientes no nulos, reduciendo de esta forma la cantidad de operaciones necesarias para su implementación [31].

Si trabajamos con un vector  $f$  de 4 muestras, su DCT será:

$$F(n) = C(n) \sum_{x=0}^3 f(x) \cos\left[(2x+1)n\frac{\pi}{8}\right]$$

Que si se expresa en forma matricial, dejando de lado el factor  $C(n)$  se obtiene que la matriz de transformación es:

$$T = \begin{vmatrix} 1 & 1 & 1 & 1 \\ \cos\left(\frac{\pi}{8}\right) & \cos\left(\frac{3\pi}{8}\right) & \cos\left(\frac{5\pi}{8}\right) & \cos\left(\frac{7\pi}{8}\right) \\ \cos\left(\frac{2\pi}{8}\right) & \cos\left(\frac{6\pi}{8}\right) & \cos\left(\frac{10\pi}{8}\right) & \cos\left(\frac{14\pi}{8}\right) \\ \cos\left(\frac{3\pi}{8}\right) & \cos\left(\frac{9\pi}{8}\right) & \cos\left(\frac{15\pi}{8}\right) & \cos\left(\frac{21\pi}{8}\right) \end{vmatrix}$$

Usando la relación trigonométrica:

$$\cos\left(\pi - \frac{n\pi}{8}\right) = -\cos\left(\frac{n\pi}{8}\right)$$

Se puede simplificar un poco la matriz anterior y evidenciar algunas particularidades

$$T = \begin{vmatrix} 1 & 1 & 1 & 1 \\ \cos\left(\frac{\pi}{8}\right) & \cos\left(\frac{3\pi}{8}\right) & -\cos\left(\frac{3\pi}{8}\right) & -\cos\left(\frac{\pi}{8}\right) \\ \cos\left(\frac{2\pi}{8}\right) & -\cos\left(\frac{2\pi}{8}\right) & -\cos\left(\frac{2\pi}{8}\right) & \cos\left(\frac{2\pi}{8}\right) \\ \cos\left(\frac{3\pi}{8}\right) & -\cos\left(\frac{\pi}{8}\right) & \cos\left(\frac{\pi}{8}\right) & -\cos\left(\frac{3\pi}{8}\right) \end{vmatrix}$$

Como se puede ver solamente hay 3 coeficientes diferentes lo cual va a simplificar las operaciones. Ahora se expresa la matriz  $T$  como el producto de dos matrices que tienen como característica que muchos de sus elementos son nulos.

$$T = \begin{pmatrix} 1 & 1 & 1 & 1 \\ c_1 & c_3 & -c_3 & -c_1 \\ c_2 & -c_2 & -c_2 & c_2 \\ c_3 & -c_1 & c_1 & -c_3 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ c_1 & 0 & c_3 & 0 \\ 0 & c_2 & 0 & -c_2 \\ c_3 & 0 & -c_1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Siendo  $c_k = \cos\left(\frac{k\pi}{8}\right)$

De esta forma se puede hacer la multiplicación  $Tf$  en dos pasos, y como se puede ver fácilmente esto reduce la cantidad de operaciones de 16 productos y 12 sumas a 5 productos y 8 sumas.

Paso 1:

$$\begin{aligned}s_{03} &= f(0) + f(3) \\d_{03} &= f(0) - f(3) \\s_{12} &= f(1) + f(2) \\d_{12} &= f(1) - f(2)\end{aligned}$$

Paso 2:

$$\begin{aligned}F(0) &= s_{03} + s_{12} \\F(1) &= c_1 d_{03} + c_3 d_{12} \\F(2) &= c_2 (s_{03} - s_{12}) \\F(3) &= c_3 d_{03} - c_1 d_{12}\end{aligned}$$

En el caso de tratarse de 8 muestras, y llevando a cabo el mismo procedimiento, se logra calcular la 1d-DCT con 22 multiplicaciones y 28 sumas (60 operaciones).

Ahora pasemos a considerar la propiedad de la separabilidad: La 2d-DCT implica estas operaciones por cada una de las filas del bloque (8 filas x 60 operaciones = 480 operaciones) y posteriormente gracias a la separabilidad se utiliza el resultado obtenido (8 filas transformadas) como si fuesen muestras de la señal de entrada para una segunda DCT aplicada sobre las columnas para las frecuencias verticales, lo cual involucra otras 480 operaciones, resultando alrededor de 1000 operaciones por cada bloque de la imagen, una cifra muy inferior a la que resultaría sin la propiedad de separabilidad (que sería 64 pixels x 60 operaciones = 3840 operaciones por bloque).

#### 2.5.5.2 Segunda etapa: cuantización

Hasta este punto no se ha reducido nada la información, pero la transformación ha generado muchos coeficientes de bajo valor y ahora es cuando se aprovecha esa característica para comprimir (con pérdidas).

A continuación los coeficientes serán cuantizados pero no todos de la misma manera. El ojo humano es muy bueno detectando pequeños cambios de brillo en áreas relativamente grandes pero no cuando el brillo cambia rápidamente en pequeñas áreas (variación de alta frecuencia). Esto permite cuantizar más bruscamente los coeficientes correspondientes a las altas frecuencias sin perder excesiva calidad visual. Para ello se divide cada coeficiente por una constante para ese componente y redondeándolo al número entero más cercano.

$$F^Q(u, v) = \text{Round} \left( \frac{F(u, v)}{\alpha \cdot Q(u, v)} \right)$$

Ecuación 6. Cuantización de los coeficientes 2d-DCT

Siendo  $Q(u, v)$  el valor de cuantización para el coeficiente  $(u, v)$  recomendado por la ISO. Y  $\alpha$  es el factor de calidad, que nos va a permitir comprimir más o menos, ya que este factor hará que se igualen a cero muchos coeficientes, por lo que esas componentes de frecuencia quedarán eliminadas.



Figura 20. JPEG factor de calidad

El factor de calidad de JPEG no tiene un significado físico concreto ni puede usarse como medida de la calidad de una imagen pues el resultado (en PSNR) depende de cada imagen. Este factor suele expresarse en porcentaje (1% - 100%). Aunque usemos un factor de calidad máximo ( $\alpha = 100\%$ ), seguiremos hablando de compresión con pérdidas, pues en el proceso de cuantización siempre se pierde calidad. Tampoco tiene una relación directa con el número de bits por pixel a los que la imagen quedará comprimida.

Finalmente los coeficientes se almacenan recorriendo los en zigzag, sencillamente porque los coeficientes que no se han anulado son mayoritariamente de baja frecuencia y están más cerca de la esquina superior izquierda. Mediante un código de escape (EOB = "End Of Block") se indica que los coeficientes que quedan son ceros, evitando su codificación.

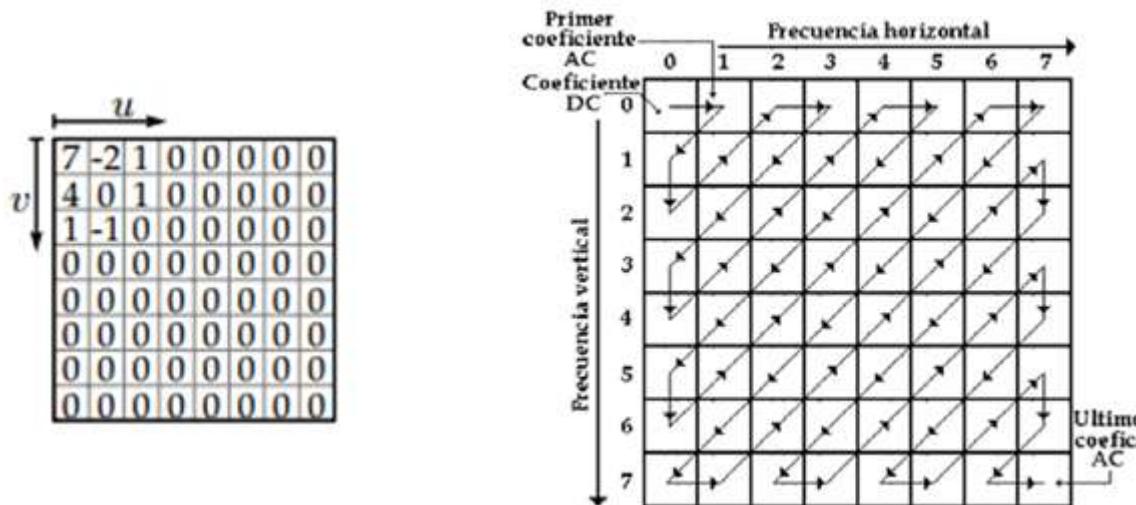


Figura 21. Ejemplo de coeficientes cuantizados y recorrido en zig-zag

### 2.5.5.3 Tercera etapa: codificador de entropía

Por último, a la lista de coeficientes de cada bloque se le aplica un pre-procesamiento RLC y a continuación una compresión Huffman a los coeficientes que han quedado. En el formato JPEG más

avanzado (no el “baseline”) se optimiza aun más esta etapa, aplicando entre otras mejoras, la compresión aritmética en lugar de Huffman, logrando hasta un 10% más de compresión.

La codificación Huffman requiere que la aplicación comprimida defina uno o más conjuntos de tablas de códigos Huffman. Las mismas tablas que se usen para comprimir la imagen se necesitan a la hora de descomprimir, por lo que deben ser almacenadas en el fichero de imagen JPEG.

Hay algunos archivadores como PackJPG [32], sin pérdidas que pueden compactar imágenes JPEG, mostrando hasta 25% de ahorro de tamaño (aunque el tiempo de procesamiento por cada fichero es de varios segundos) [33]. El funcionamiento básico consiste en decodificar la imagen JPEG, extraer los coeficientes, aprovechar redundancias de los coeficientes DC de los bloques adyacentes y volverlos a comprimir usando codificación aritmética [34].

## 2.5.6 JPEG2000

La transformada wavelet DWT [35] (Discrete Wavelet Transform) surge para dar respuesta a las limitaciones de la transformada de Fourier (y por ende también de la DCT) al transformar señales no periódicas cuyo espectro es dependiente del tiempo (o del espacio si se trata de imágenes).

La Transformada de Fourier detecta la presencia de un determinado armónico pero no proporciona información acerca de la evolución en el tiempo de las características espectrales de la señal. Para solucionarlo se creó la “transformada de Fourier con ventana”, donde los armónicos aplican sólo a una ventana temporal (o espacial) y se van desvaneciendo a medida que nos alejamos de un punto central, es decir, es una ventana temporal Gausiana. El problema de este sistema era que la ventana es siempre del mismo tamaño por lo que la localización de la aparición de una singularidad de una señal dependía del ancho elegido para la función ventana.

La transformada wavelet tiene un tamaño de ventana adaptado a la frecuencia del armónico considerado, aplicando una ventana mayor en el caso de bajas frecuencias y menor en el caso de altas frecuencias. Algo parecido a lo que ocurre en cartografía, donde los detalles (altas frecuencias) se corresponden con ventanas de observación pequeñas en espacio.

La implementación de este concepto es sencillo aunque costoso computacionalmente. En lugar de considerar armónicos puros (como se hace en DCT) se consideran sub-bandas de frecuencia y se hace pasar la señal original por una serie de filtros en cascada que van separando las diferentes sub-bandas de frecuencia de la señal original. Filtrar la señal implica convolucionarla con la respuesta al impulso del filtro considerado y el número de muestras del filtro con el que convolucionamos es precisamente el tamaño de la ventana espacial. La convolución es una operación costosa computacionalmente.

$$x[n] \otimes h[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k]$$

Figura 22. Convolución discreta

El procedimiento comienza convolucionando la señal a través de un filtro paso bajo cuya respuesta al impulso es  $h[n]$  y un filtro paso alto cuya respuesta es  $g[n]$ . Ambos filtros deben ser espejos y en cuadratura, donde uno es siempre la imagen espejo del otro en el dominio de la frecuencia. La convolución se realiza en el espacio pero esta relación de los filtros se establece en frecuencia. Los filtros deben mantener esta relación entre sí para no perder información.

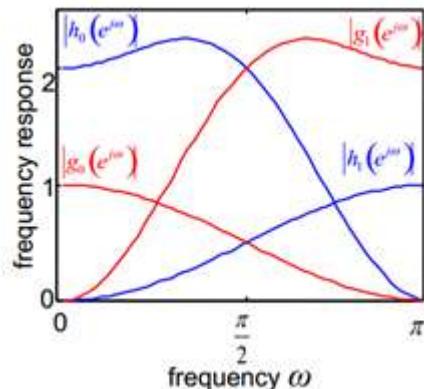


Figura 23. Filtros espejo en cuadratura

Tras el filtrado se sub-muestrea el resultado a la mitad. El sub-muestreo no elimina información de acuerdo al principio de Nyquist ya que puesto que la señal se ha reducido en ancho de banda a la mitad, puede ser sub-muestreada a la mitad. Al sub-muestrear ambos resultados, la suma de muestras sigue siendo igual al de la señal original.

A continuación se realiza de nuevo la operación de convolución con la banda de baja frecuencia obtenida, dividiéndola en dos bandas y así sucesivamente.

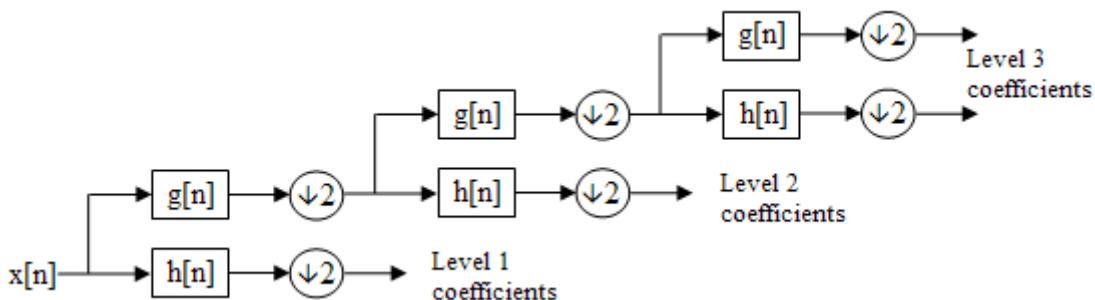


Figura 24. Filtrado recursivo de JPEG2000

El resultado de esta serie de filtrados es una descomposición de la señal en varias bandas de frecuencia. Este proceso de “filtro en cascada” se hace por filas y por columnas (separable) mediante la aplicación de dos convoluciones unidimensionales para conseguir el filtrado bidimensional.



Figura 25. Convolución recursiva wavelet

La imagen final filtrada en diferentes sub-bandas, es cuantizada, de modo que muchos coeficientes poco significativos se hacen cero. JPEG2000 a continuación codifica entrópicamente los resultados agrupando la información en bloques y codificándola con compresión aritmética, mediante un proceso llamado EBCOT (“Embedded Block Coding with Optimized Truncation”) [36]

Los bloques de EBCOT son de 32x32 pixels y el proceso que lleva a cabo EBCOT es computacionalmente costoso. La complejidad de JPEG2000 comparada con JPEG es de un orden de magnitud superior, produciendo imágenes de mejor calidad aunque no substancialmente mejores salvo a altos ratios de compresión [29].

El procedimiento EBCOT organiza los bloques en orden de importancia, de modo que se puedan transmitir más o menos bloques para reproducir la imagen con mayor o menor calidad, lo cual hace de JPEG2000 un algoritmo escalable [37]. Los bloques contienen coeficientes y estos coeficientes se organizan por planos de bit. Un plano de bit son todos aquellos bits de los coeficientes que corresponden a la misma magnitud  $2^p$ , siendo “ $p$ ” el plano considerado.

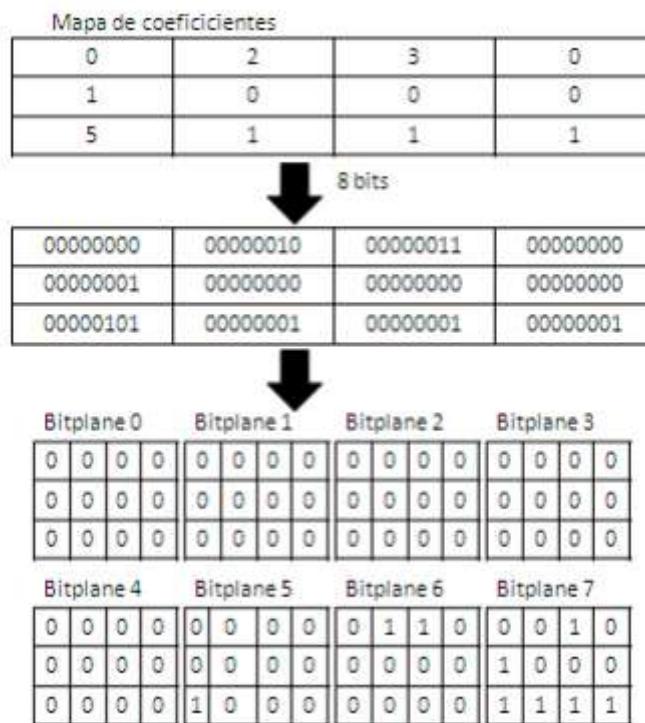


Figura 26. Planos de bits

Cada plano de bits se subdivide en “stripes”, que son segmentos a los que se va a asociar un código aritmético. Los stripes son secuencias unidimensionales de bits, concatenados tal como se presenta en la siguiente figura. El código aritmético que se asocia a cada stripe es calculado mediante un codificador aritmético adaptativo llamado “MQ”.

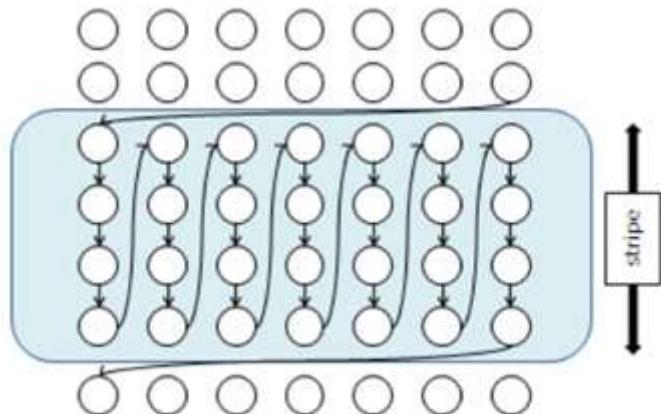


Figura 27. Stripe dentro de un plano de bit

Estos bloques con los que trabaja EBCOT permiten parallelizar el trabajo del codificador entrópico. Los bloques solo aparecen en esta fase, ya que la transformación en coeficientes de las sub-bandas de frecuencia han sido calculados con toda la imagen, es decir, se hace la transformada wavelet considerando la imagen como un único bloque, por ese motivo desaparece el efecto de bloques que suele presentar JPEG. Si JPEG2000 se ejecutase por bloques el efecto reaparecería como se muestra en la siguiente imagen. Esta misma estrategia podría hacerse con JPEG, y si se tratase la imagen como un único bloque desaparecería el efecto de bloques. Lo malo es que perderíamos la capacidad de parallelización del algoritmo y además la DCT se haría costosísima computacionalmente. De hecho una de las soluciones más importantes de JPEG para abordar la DCT con un coste razonable es dividir la imagen en pequeños bloques.



Figura 28. Wavelets produciendo efecto de bloques

En imágenes muy grandes la compresión JPEG2000 puede dividir la imagen en grandes bloques a los que se aplica la transformada DWT de forma independiente pero debido al efecto de bloques no es muy recomendable.

### 2.5.7 JPEG XR

JPEG XR es un estándar de compresión de imágenes fijas y el formato de archivo para imágenes fotográficas basado en la tecnología originalmente desarrollada y patentada por Microsoft bajo el nombre de HD Photo y estandarizada en 2007 con el Joint Photographic Experts Group bajo el nombre de JPEG XR

JPEG XR es conceptualmente muy similar a JPEG y posee las mismas etapas. Sin embargo posee algunas diferencias que lo hacen superior en calidad, aunque también lo hacen más complejo computacionalmente.

Las diferencias fundamentales son:

- Permite transparencias (canal alfa.)
- Permite compresión con y sin pérdidas.
- La transformada a frecuencia JPEG XR es de 4x4 mientras que la transformada de JPEG es de 8x8.
- La transformada a frecuencia de JPEGXR se hace en dos etapas, de modo que un macrobloque de 16x16 pixels se divide en 16 sub-bloques de 4x4 y se transforma en dos etapas, de un modo jerárquico.
- JPEG XR utiliza mecanismos de predicción de coeficientes en el dominio de la frecuencia para ahorrar información
- Los coeficientes en JPEG XR no se recorren en zigzag sino de forma adaptativa en función de los resultados.
- En la etapa de codificador de entropía, los coeficientes de JPEG XR usan Huffman adaptativo mientras que JPEG usa Huffman.

Los resultados en cuanto a calidad obtenidos [38] con JPEG XR son similares a JPEG2000. Recientemente, en 2013 Microsoft ha liberado una librería open source JPEG XR.

### 2.5.8 WebP

WebP [39] es un formato comprimido con pérdidas que se basa en el codificador intraframe del compresor de vídeo VP8 y VP9 de Google. Usa estrategia predictiva en el espacio y DCT.

Es más costoso computacionalmente que JPEG y existe cierta controversia respecto a su calidad pues aunque a bajos bit-rates es superior a JPEG, a medios y altos bit-rates produce un mayor efecto de desenfoque por lo que a veces se pierden detalles [40]

WebP usa una etapa predictiva que tiene en cuenta pixels cercanos ya decodificados para estimar el valor de un pixel. Su estrategia predictiva guarda cierta similitud con PNG. Divide la imagen en bloques (de 16x16 para luminancia y 8x8 para crominancia) y escoge un filtro predictivo entre 13 disponibles en función de lo que sea más adecuado para cada bloque. Y en todos los pixels de un mismo bloque aplica la misma predicción. Las predicciones más usadas están descritas en la siguiente figura. El resto de predicciones son variaciones de estos predictores básicos.

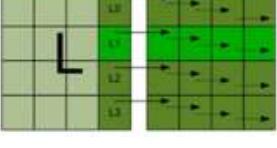
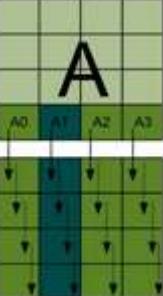
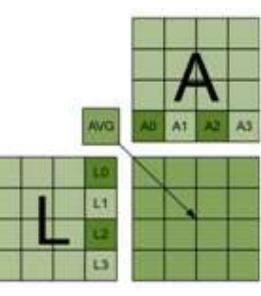
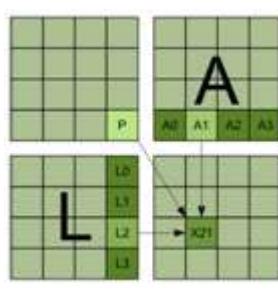
H_PRED	V_PRED	DC_PRED	TM_PRED
			
(Horizontal Prediction) Replica el valor de la columna del bloque izquierdo en todos los pixels del bloque	(Vertical Prediction) Replica el valor de la fila del bloque superior en todos los pixels del bloque	(Direct Current) Rellena todo el bloque con un solo valor para todos los pixels. Calculado como promedio de la fila del bloque superior y la columna del bloque izquierdo	(True Motion Prediction) Calcula la predicción de cada pixel como: $X_{ij} = L_i + A_j - P$

Figura 29. Principales Predictores de WebP

A continuación WebP calcula la diferencia entre la predicción con los valores del bloque original y la diferencia (el “residuo”) se transforma al dominio de la frecuencia usando la DCT en sub-bloques de 4x4.

Los coeficientes resultantes, una vez cuantizados y truncados, son codificados mediante un codificador aritmético [41]. La especificación detallada está publicada en la RFC6386, como parte del estándar VP8

Otra característica de WebP es que maneja paletas de color reducidas que va adaptando en función de la cantidad de colores usados en cada bloque aunque su mayor punto fuerte es su análisis para escoger el mejor filtro predictivo en cada bloque de la imagen.

## 2.5.9 H264 intraframe

Aunque H264 es un estándar de codificación de video, posee su propio algoritmo de compresión intraframe, que guarda similitudes con WebP. Es posible comprimir imágenes usando este formato mediante herramientas como “ffmpeg”, que llevan el códec integrado.

Primeramente lleva a cabo una estrategia predictiva basada en 9 tipos de predicción, para grupos de 4x4 pixeles

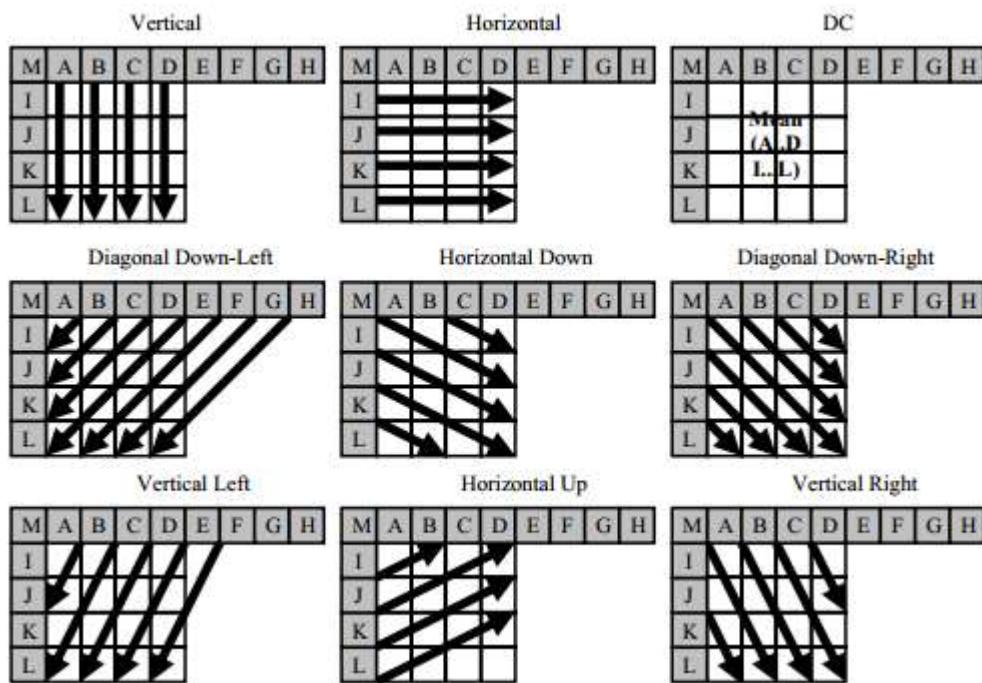


Figura 30. Predicciones de 4x4 en h264 intraframe

Tras la predicción, se calcula el residuo (imagen original-predicción) y se aplica una transformada DCT en bloques de 4x4 que solo usa números enteros.

Por último el resultado es cuantizado y pasado a través de un codificador de entropía que recorre los coeficientes resultantes en zig-zag, igual que JPEG. El codificador de entropía se llama CABAC (Context Adaptive Binary Arithmetic Coding). Se trata de un codificador aritmético avanzado, obviamente superior a Huffman.

### 2.5.10 Fractales

Para poder relatar con precisión este tipo de compresión habría que empezar por definir los fractales y sus ecuaciones, lo cual está fuera del alcance de esta memoria. Los fractales tienen una propiedad llamada auto-similitud, mediante la cual se observan replicas a diferentes escalas o rotaciones. Esta misma propiedad está presente en las imágenes “naturales”

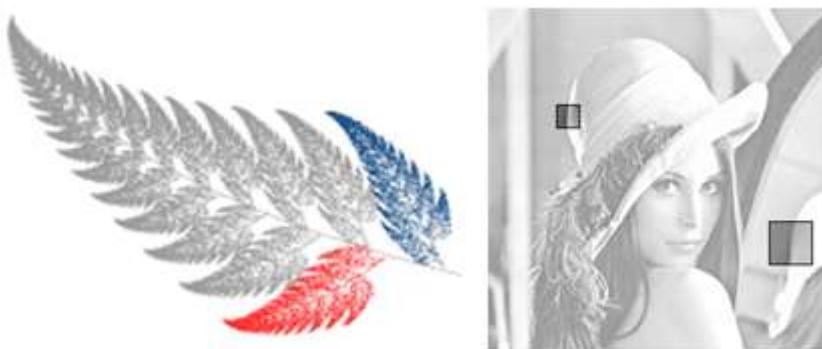


Figura 31. Auto-similitudes

Explorar las similitudes a diferentes escalas de una misma imagen puede resultar en una compresión muy potente, sin embargo el coste computacional para identificarlas es muy elevado. Según [42], El tiempo de compresión para la imagen Lena de 512x512 en una máquina con procesador de 2.83GHz fue de aproximadamente media hora, por lo que es un algoritmo inviable en la práctica actualmente. Existe un formato llamado FIF (Fractal Image Format) pero no es estándar.

### 2.5.11 Redes neuronales

Las redes neuronales artificiales (RNA) pueden entrenarse para aprender o aproximar secuencias de datos y patrones de información.

La aplicación de redes neuronales a la compresión de imágenes [43] es una técnica experimental de alto coste computacional. Consta de dos problemas a resolver:

- Determinación de la mejor topología de la red: no es nada trivial y se utilizan estrategias como el uso de sistemas expertos para su determinación
- Entrenamiento: esta fase consume muchos recursos (horas de cálculo) aunque depende del margen de error que estemos dispuestos a tolerar

La siguiente figura muestra una compresión llevada a cabo con esta técnica y aunque el coste es elevadísimo y el autor [43] no proporciona valores de PSNR obtenidos, al menos refleja que con esta técnica es posible obtener resultados aparentemente muy buenos.



Imagen original 800x600: 1.4MB



Imagen comprimida 32KB (1:45 ≈ 0.7 bpp)

Figura 32. Compresión usando RNA

Para valorar justamente esta técnica habría que conocer sus gráficas de R-D (Rate-Distortion) pero llevar a cabo ese trabajo para una batería de imágenes supone un trabajo inmenso. Cada imagen puede requerir una topología de RNA diferente y su compresión puede suponer horas de cálculo.

## 2.6 Representación y manipulación de imágenes

En esta sección voy a presentar algunos aspectos de la representación de las señales de las imágenes (color y distribución) y como algunas operaciones de manipulación de imágenes relacionadas con el tratamiento de su histograma y la interpolación.

## 2.6.1 Espacios de color

A la representación matemática de un conjunto de colores mediante unas “coordenadas” de color se la denomina “espacio de colores”. Un espacio de color se fundamenta en un modelo de color. Por ejemplo, basarse en tres componentes R, G y B es usar el “modelo” de color RGB pero el detalle de cómo esas tres componentes se ponderan para crear toda la gama sería el “espacio” de color.

Para definir un espacio de color, la referencia estándar habitual es uno de los modelos de color del CIE (comisión Internacional de Iluminación), los cuales están diseñados específicamente para abarcar todos los colores que el ser humano puede ver. Sin embargo los dispositivos de representación pueden no ser capaces de representarlos todos, y el modelo constructivo en el que se basan (como las pantallas de visualización) generan el color a partir de otras componentes (modelo RGB) imponiendo ciertas limitaciones. En caso de tratarse de colores impresos, el modelo adecuado es sustractivo (modelo CYMK) y no aditivo pues al añadir tinta de otro color, el brillo se oscurece en lugar de aumentar.

Los modelos más conocidos se describen a continuación.

### 2.6.1.1 Modelo CIE

La CIE (comisión Internacional de Iluminación) es la autoridad internacional en cuestiones de luz, iluminación, color y espacios de color. El modelo CIE es capaz de representar todos los colores que el ojo humano puede percibir. En realidad la CIE propone tres modelos de color equivalentes CIE-XYZ, CIE-xyY y CIELAB.

El modelo CIE-XYZ representa los colores en un “cubo” tridimensional mientras que el CIE-xyY permite representar los colores en dos dimensiones como se muestra a continuación.

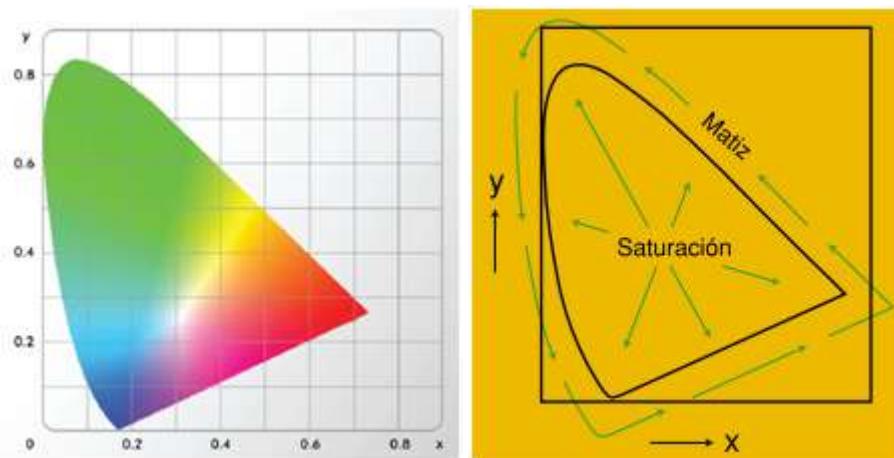


Figura 33. Modelo de color CIE-xyY

En la parte interior de la figura están los colores menos saturados, por eso el blanco se sitúa en el centro.

Para mejorar la representación del color, la CIE desarrolló en 1976 el modelo de color CIELAB. Con este modelo se vuelve a representar en tres dimensiones (al igual que en CIE-XYZ) en lugar de en dos. Las diferencias de color que se perciben como iguales en este espacio de color tridimensional, tienen distancias iguales entre ellas. Para representar los colores utiliza una coordenada de luminancia, otra de

magenta-verde (valores negativos indican magenta y positivos verde) y otra de amarillo-azul (valores negativos indican amarillo y positivos azul).

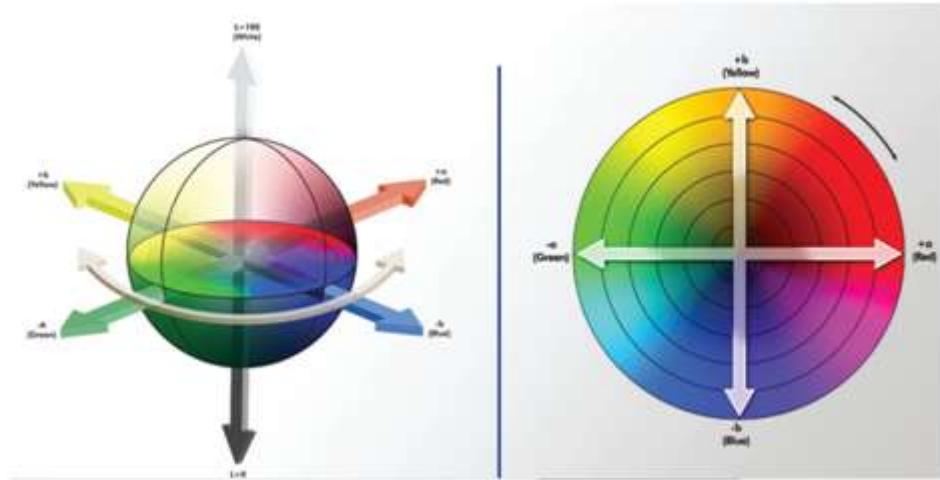


Figura 34. Modelo de color CIELAB

### 2.6.1.2      *Modelo RGB*

Los espacios de color basados en el modelo RGB utilizan una mezcla de colores aditivos. RGB almacena valores individuales para el rojo, el verde y el azul. El espacio de color RGBA es RGB con un canal adicional alfa para indicar transparencia.

Entre los espacios de color basados en RGB se incluye sRGB, Adobe RGB y ProPhoto RGB.

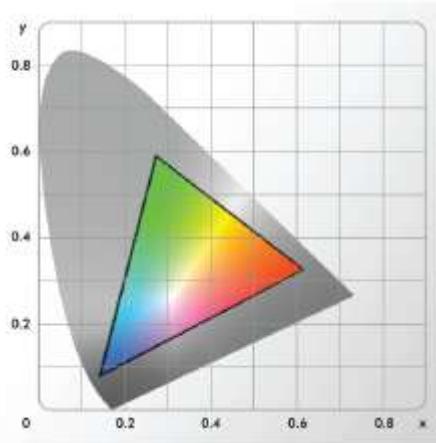


Figura 35. Gama de colores RGB sobre el modelo CIE-xyY

### 2.6.1.3      *Modelos HSV y HSL*

HSV (hue, saturation, value), también conocido como HSB (hue, saturation, brightness) es usado a menudo en aplicaciones artísticas porque es más natural pensar sobre un color en términos de matiz y

saturación que en términos de componentes de color aditivos o sustractivos.

El matiz (hue) es como se percibe el color de un objeto: rojo, anaranjado, verde, azul, etc. La saturación es la intensidad de ese matiz y el brillo es el grado de claridad o intensidad lumínica.

HSV es una transformación de un espacio de color RGB, y sus componentes y colorimetría son relativos al espacio de color RGB del que deriva.

HSL (hue, saturation, lightness/luminance), también conocido como HLS o HSI (hue, saturation, intensity) es bastante similar a HSV, con la "claridad" reemplazando el "brillo". La diferencia es que el "brillo" de un color puro es igual al brillo del blanco, mientras que la claridad de un color puro es igual a la claridad de un gris medio.

#### **2.6.1.4      *Modelo CMYK***

CMYK (Cian, Magenta, Yellow , black) es en realidad la corrección del modelo RYB (Red, Yellow, Blue) según el cual los colores primarios de la pintura son el rojo, amarillo y azul, lo cual es falso en la vida real, aunque algunas aproximaciones hayan conducido a esta teoría, que fue hecha popular por Goethe en su libro “teoría de los colores” en 1810, aunque hasta el 2004 no fue reconocida su falsedad. En CMYK el rojo es sustituido por el magenta y el azul es sustituido por el cian.

CMYK utiliza síntesis sustractiva de color utilizada en el proceso de impresión, porque describe qué clase de tinta necesita aplicarse para que la luz reflejada desde el sustrato y a través de la tinta produzca un color dado. Se empieza con un sustrato blanco (lienzo, página, etc.), y se utiliza la tinta para sustraer el color del blanco para crear una imagen. CMYK almacena valores de tinta para cian, magenta, amarillo y negro. Hay muchos espacios de color CMYK para diferentes conjuntos de tintas, sustratos, etc. (los cuales cambian la ganancia del punto o la función de transferencia para cada tinta y, de esa forma, cambiar la apariencia).

#### **2.6.1.5      *Modelo YUV***

El ojo es mucho más sensible al cambio de luminancia que al de crominancia, por este motivo es más eficiente invertir más bits en la luminancia que en la crominancia. El modelo YUV separa la componente de luminancia de las de crominancia y ello lo hace más adecuado que RGB para comprimir, almacenar o transmitir imágenes, pues en caso de compresión con pérdidas se puede degradar más las componentes de color que la luminancia.

El modelo YUV también se conoce como YIQ, CCIR 601 o YCbCr. Es el modelo utilizado en JPEG. La diferente nomenclatura responde a razones históricas. En los sistemas de televisión analógica en color se hablaba de YUV o YIQ, mientras que en imágenes digitales se habla de YCbCr pero es esencialmente lo mismo. Sin embargo las ecuaciones de YUV y de YCbCr son ligeramente distintas debido a que en YCbCr se ajustan para minimizar el efecto de posibles valores distorsionados

El modelo YCbCr representa los mismos colores que RGB (su gama de color es la misma) y se relaciona con éste mediante las siguientes ecuaciones, utilizadas en JPEG y en MPEG:

$$\begin{aligned} Y &= R \cdot 0.299 + G \cdot 0.587 + B \cdot 0.114 \\ C_b &= 128 + (-0.168 \cdot R - 0.331 \cdot G + 0.5 \cdot B) \\ C_r &= 128 + (-0.5 \cdot R - 0.418 \cdot G + 0.081 \cdot B) \end{aligned}$$

Ecuación 7. Ecuaciones del modelo YCbCr

El motivo de llamar a las componentes de color Cb y Cr es sencillamente porque Cb contiene más de la componente azul que del resto y Cr contiene más de rojo que del resto.

La luminancia contribuye del mismo modo a las componentes de color. La siguiente figura visualiza los resultados de color para dos valores de luminancia [29].

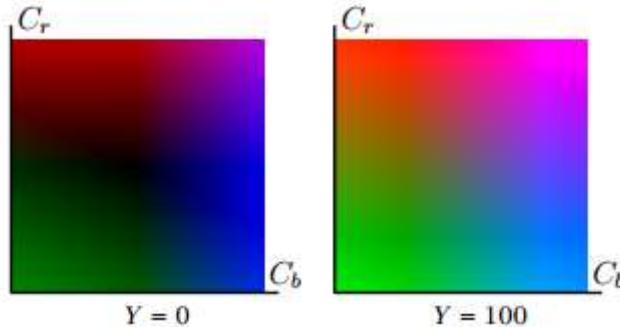


Figura 36. Efecto de la Y en el modelo YCbCr

El modelo YUV (o YCbCr) tiene la particularidad de que al separar las componentes de color podemos almacenar una imagen sub-muestreando las componentes de color sin dañar demasiado la calidad. Esto nos permite ahorrar información y por ello se han creado los modelos YUV422 y YUV420

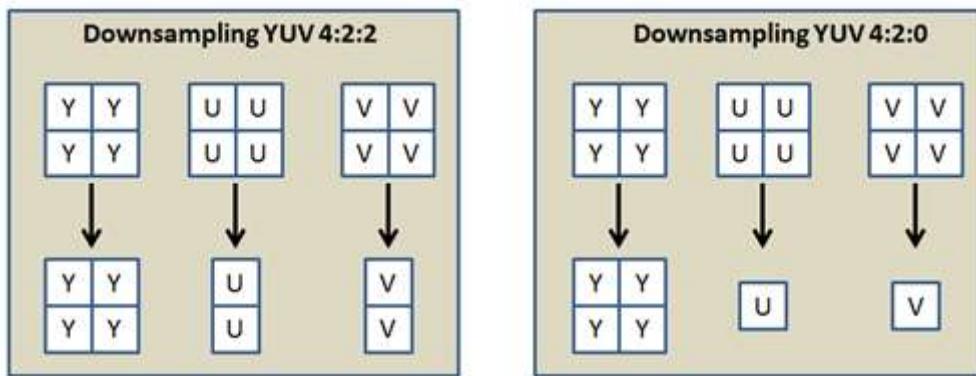


Figura 37. Modelos YUV

**YUV422:** en este caso el número de muestras de crominancia es la mitad que la de la luminancia en la coordenada x, mientras que en la coordenada Y se mantiene el número de muestras.

**YUV420:** en este caso el número de muestras de crominancia es la mitad que la de la luminancia tanto en la coordenada X como en la coordenada Y

## 2.6.2 Tratamiento de Histogramas

El histograma de una imagen es la distribución de los niveles de gris de dicha imagen. La manipulación del histograma puede alterar la imagen para aumentar su contraste, su brillo o para hacer visibles zonas ocultas [44].

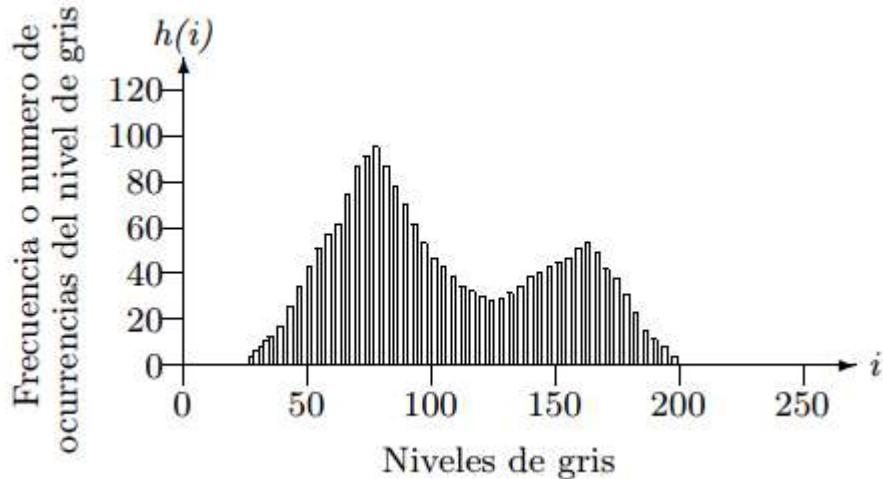


Figura 38. Histograma de una imagen

Para aumentar el brillo de una imagen podemos desplazar el histograma, es decir, corregir los niveles de gris sumándoles el mismo offset a todos ellos.

Cuando la imagen posee poco contraste y deseamos aumentarlo, podemos ecualizar el histograma o bien expandirlo. La ecualización es útil cuando el intervalo cubierto por el histograma original abarca todo el rango disponible, mientras que si el rango de niveles de gris se encuentra concentrado en una zona del intervalo, lo más adecuado es una expansión.

Ecuación de histograma: con esta operación se busca que el histograma se transforme en una línea horizontal. No podemos conseguir una ecualización perfecta porque todos los pixeles de un mismo nivel de gris serán convertidos a un único nuevo nivel de gris pero se puede aproximar a un histograma plano. El histograma acumulado será algo parecido a una recta con pendiente constante  $N/L$  siendo  $N$  el número total de pixels y  $L$  el número de niveles de gris

Llamaremos  $n_k$  al número de pixeles con valor  $k$ . Dividiendo el valor del acumulado de pixeles con tono de gris  $\leq k$ , entre el acumulado total, obtendremos a que nuevo tono de gris deberíamos transformar  $k$  para que la nueva distribución de grises sea homogénea

$$T(k) = \frac{\sum_0^{k-1} n_k}{\sum_0^{L-1} n_k} \cdot (L - 1)$$

Ecuación 8. Ecualización de histograma

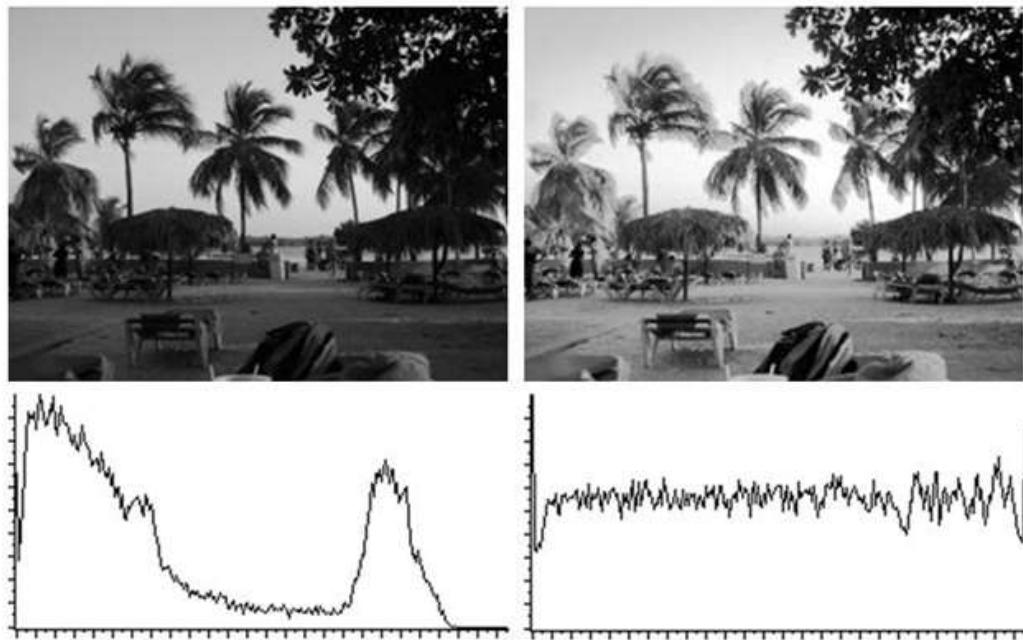


Figura 39. Ecualización de histograma

**Expansión del histograma:** Consiste en aumentar el rango de niveles de gris de la imagen. Se puede conseguir aplicando una transformación de las intensidades mediante una función a trozos o bien aplicando la expansión a todo el histograma. Suponiendo que el rango disponible es  $[0, L-1]$ , podemos llevar un intervalo  $[a,b]$  al  $[0,L-1]$  con la ecuación:

$$l' = \frac{(l - a)}{b - a} \cdot (L - 1)$$

Ecuación 9. Expansión de histograma

Donde  $l$  es el nivel de gris y  $l'$  es el nuevo nivel de gris. En la siguiente figura se muestra el resultado de una expansión de histograma.



Figura 40. Expansión de histograma

A diferencia de la ecualización, con la expansión de histograma, un tono de gris que es mayoritario al principio, es transformado en otro tono que es mayoritario al final. La proporción de los niveles de gris no se altera, lo que se altera es el tono de gris asociado cuya nueva distribución cubre mejor todo el rango disponible. Esta característica hace que si (por ejemplo) el tipo de distribución original era Gausiana, lo seguirá siendo tras la expansión.

Las operaciones que podemos realizar sobre un histograma en realidad las podemos realizar sobre cualquier distribución estadística, aunque si en lugar de tonos de gris hablamos de otra variable, el resultado tendrá otro significado. Por ejemplo el algoritmo LHE presentado en esta tesis realiza una expansión del histograma de relevancia perceptual y no de los niveles de gris. El efecto perseguido es el mismo, aumento de contraste, pero de otra variable.

### 2.6.3 Filtros

Los filtros pueden clasificarse de diversos modos, según se apliquen en el dominio del espacio o de la frecuencia o según su linealidad o no linealidad. Voy a presentar un filtro no lineal y la base de los filtros lineales que es la operación de convolución, utilizada en JPEG2000 y involucrada en el cálculo de vectores de movimiento de los algoritmos de codificación de video. La palabra “lineal” en estos casos hace referencia a sus propiedades matemáticas y no a su complejidad computacional.

#### 2.6.3.1 Filtrado no lineal

El filtro de la mediana, es un procedimiento no-lineal, que se aplica en el dominio del espacio, útil para reducir el ruido impulsivo y del tipo “sal y pimienta” (pixels blancos y negros distribuidos de forma aleatoria), muchas veces presente en las imágenes. El filtro de mediana utiliza los valores de los píxeles contenidos en una ventana de tamaño impar, para determinar el nuevo valor del píxel de interés. El procedimiento para ello, consiste en ordenar todos los píxeles incluidos en la ventana y sustituir el pixel ubicado en el centro de la ventana por el píxel resultado de calcular la mediana.

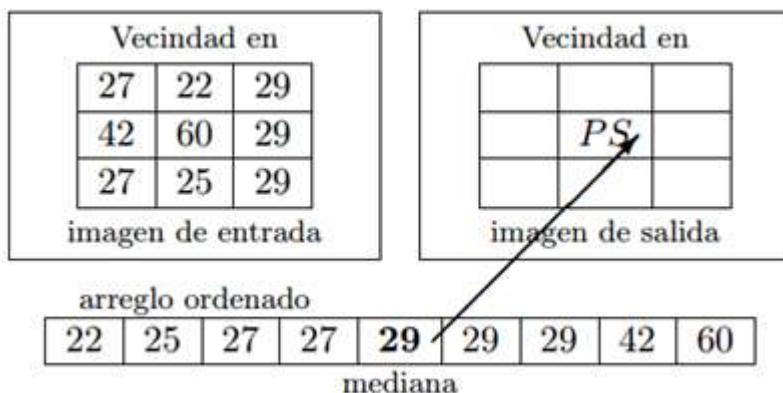




Figura 41. Filtrado de mediana para eliminar ruido

El filtro de media es similar al filtro de mediana solo que en lugar de aplicar la mediana aplicaremos la media, que es una operación menos costosa pues no requiere la ordenación de la lista de coeficientes.

El resultado es equivalente a un filtrado paso bajo, en el que se suaviza más o menos la imagen en función del tamaño de la ventana espacial considerada. Este es un filtro que se puede implementar con una convolución como veremos a continuación.

### 2.6.3.2      *Filtros de convolución (lineales)*

Para todo sistema lineal, la respuesta al mismo de cualquier señal puede calcularse como la convolución de la propia señal con la respuesta al impulso que denominaremos como  $h(x, y)$ . A esta respuesta se la conoce como PSF (“Point Spread Function”). El teorema de la convolución establece que una convolución en el espacio es un producto en el dominio de la frecuencia y viceversa

$$y(m, n) = h(m, n) \otimes x(m, n) = \sum_{m'=-\infty}^{\infty} \sum_{n'=-\infty}^{\infty} h(m - m', n - n') \cdot x(m', n')$$

A continuación un ejemplo muy sencillo de aplicación de la convolución en el que vamos a calcular el valor del pixel del centro. Tenemos la señal (también llamada matriz de imagen), la máscara de convolución y el resultado de la convolución para ese pixel.

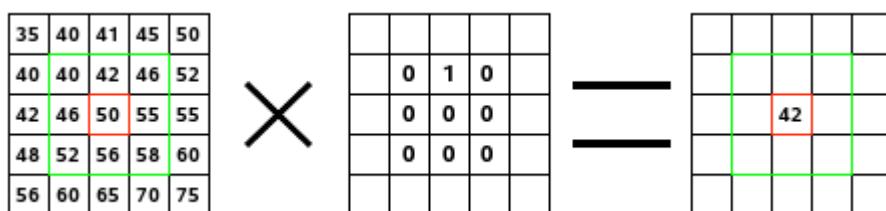


Figura 42. Ejemplo simple de cálculo de la convolución

Cada filtro se caracteriza por su máscara de convolución, que es precisamente su función PSF. Cuanto mayor es el tamaño de la máscara, mayor es el número de multiplicaciones y sumas que hay que efectuar para cada pixel y por lo tanto más costosa computacionalmente es la convolución.

A continuación se muestran algunos ejemplos ilustrativos de filtros de convolución

<table border="1"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>-1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>-1</td><td>5</td><td>-1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>-1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table> <p>Mascara de enfoque</p>	0	0	0	0	0	0	0	-1	0	0	0	-1	5	-1	0	0	0	-1	0	0	0	0	0	0	0		<table border="1"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table> <p>Mascara de desenfoque (paso bajo)</p>	0	0	0	0	0	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0	0	0	0	0	
0	0	0	0	0																																																	
0	0	-1	0	0																																																	
0	-1	5	-1	0																																																	
0	0	-1	0	0																																																	
0	0	0	0	0																																																	
0	0	0	0	0																																																	
0	1	1	1	0																																																	
0	1	1	1	0																																																	
0	1	1	1	0																																																	
0	0	0	0	0																																																	
<table border="1"> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>-4</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> </table> <p>Mascara de detección de bordes</p>	0	1	0	1	-4	1	0	1	0		<table border="1"> <tr><td>-2</td><td>-1</td><td>0</td></tr> <tr><td>-1</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>2</td></tr> </table> <p>Mascara efecto repujado</p>	-2	-1	0	-1	1	1	0	1	2																																	
0	1	0																																																			
1	-4	1																																																			
0	1	0																																																			
-2	-1	0																																																			
-1	1	1																																																			
0	1	2																																																			

Figura 43. Efectos de convolución

## 2.6.4 Operaciones de escalado

### 2.6.4.1 Sub-muestreo

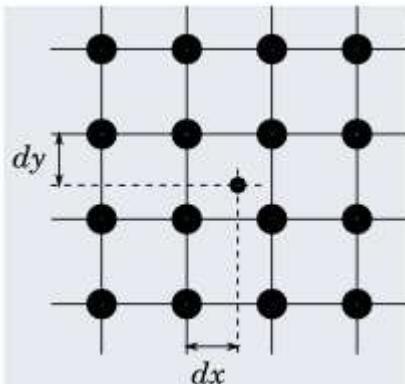
La reducción de tamaño de una imagen (también llamado “Downsampling” o “decimation” ) se puede llevar a cabo mediante dos técnicas:

- “Simple Pixel Selection” (SPS): se escoge un pixel de cada N. Algunas muestras pueden ser ruido, por lo que no es recomendable.
- filtro paso bajo + SPS: la aplicación del filtro puede hacerse de un modo que ya tenga en cuenta el sub-muestreo de modo que para grupo de N pixels calculamos su media o su mediana y pasamos al siguiente grupo de N pixels. Es menos costoso que una convolución.

El downsampling es una operación separable, se puede efectuar por filas y después por columnas.

### 2.6.4.2 Algoritmos de Interpolación

Interolar una imagen (también llamado “Upsampling”) es crear muestras ficticias donde antes no existían, reconstruyendo la información entre las muestras (pixels) existentes.



En esta figura los círculos grandes representan píxeles existentes. Se desea calcular un nuevo valor de píxel en la posición representada por el punto pequeño, definida por las distancias  $dx$  y  $dy$ , medidas desde el píxel más cercano

Existen muchos algoritmos de interpolación, siendo los más conocidos:

**Vecino Cercano (Nearest Neighbour):** Este es el algoritmo de interpolación más simple posible. La interpolación del vecino más próximo selecciona el valor del píxel más cercano redondeando las coordenadas del punto de interpolación deseado.

**Bilineal:** Un método de interpolación ligeramente más sofisticado que el anterior. El algoritmo bilineal interpola a partir de los cuatro píxeles originales que rodean al punto deseado de interpolación. EL proceso es separable, de modo que primero se interpola linealmente el valor de la señal horizontalmente entre las muestras disponibles y después se interpola verticalmente usando las muestras originales y las nuevas.

Interpolación lineal	
$c_1$ : color de la muestra 1 $c_2$ : color de la muestra 2 $c_x$ : color del punto interpolado  	La ecuación general es: $c_x = ax + b$ <p>Tenemos dos incógnitas y dos condiciones:</p> <p>Que pase por el primer punto:  <math>c_1 = a0 + b</math></p> <p>Que pase por el segundo punto:  <math>c_2 = a1 + b</math></p> <p>Quedando:</p> $c_x = (c_2 - c_1)x + c_1$

**Bicúbica:** Los algoritmos de interpolación bicúbica interpolan a partir de los dieciséis píxeles originales más cercanos que engloban el punto de interpolación deseado, mediante un polinomio de grado 3.

Interpolación cúbica	
$c_1$ : color de la muestra 1 $c_2$ : color de la muestra 2 $c_3$ : color de la muestra 3 $c_4$ : color de la muestra 4 $c_x$ : color del punto interpolado	La ecuación general es: $c_x = ax^3 + bx^2 + cx + d$ <p>Y las condiciones son:</p> $c_x = c_1 \text{ cuando } x = -1$ $c_x = c_2 \text{ cuando } x = 0$ $c_x = c_3 \text{ cuando } x = 1$ $c_x = c_4 \text{ cuando } x = 2$

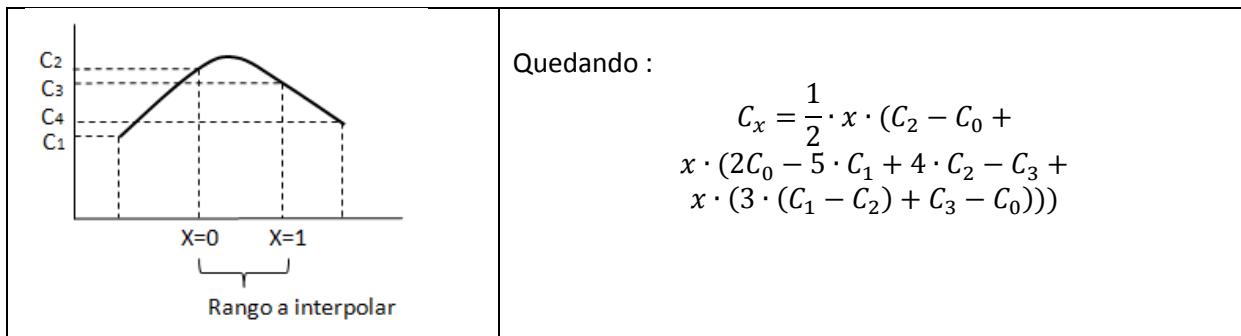


Figura 44. Interpolaciones más básicas

**Spline Bicúbico:** Se trata de un algoritmo de interpolación de píxel de altas prestaciones, que proporciona buenos resultados tanto en lo relativo a velocidad de ejecución como en calidad. Utiliza una función spline cúbica como un filtro separable para llevar a cabo una interpolación por convolución.

**Lanczos:** es un algoritmo iterativo más costoso computacionalmente que el bicúbico y con un resultado de calidad algo superior.

### Algoritmos de escalado para “Pixel Art”

“Pixel art” es el nombre que se le da a las creaciones artísticas dibujadas pixel a pixel, tal como se hacía con los gráficos de ordenador y videoconsola antiguos. Existen algoritmos de interpolación que inicialmente fueron creados para interpolar este tipo de imágenes pero actualmente se aplican en algunas cámaras fotográficas. Los nombres de estos algoritmos son “Efficiency”, “EPX”, “HQX”, “XBR”, “XBRZ”, “Eagle”, “superEagle”, “AdvMAME3x/Scale3x”, “2xsal”, “PhotoZoom”. Muchas de estas técnicas se pueden comparar en [45]

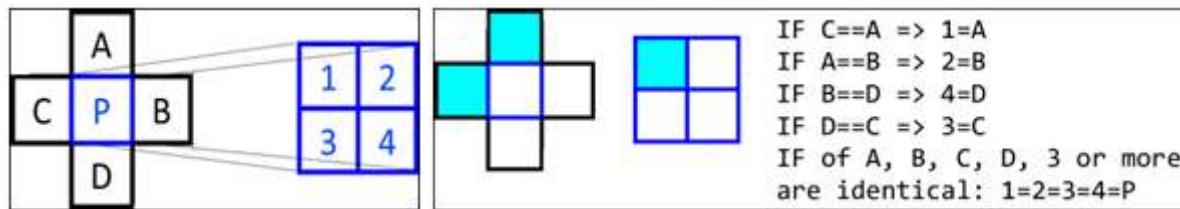
Algunos de ellos son algoritmos muy rápidos, diseñados para ser ejecutados en tiempo real sobre la imagen generada por un emulador de videoconsola. A modo de ejemplo la siguiente figura muestra una interpolación por vecino, bicubica y por último una HQX.



Figura 45. Interpolación por vecino, bicúbica y HQX

Son algoritmos diseñados para imágenes construidas en baja resolución y con poco color por lo que para aplicarlos sobre imágenes fotográficas quizás requerirían ciertas adaptaciones

El algoritmo EPX es uno de los más sencillos y a la vez efectivos. EPX son las siglas de “Eric’s Pixel eXpander” (su creador se llama Eric Johnson). Creado en 1990 cuando Eric trabajaba en Lucas Arts. EPX expande un pixel en 4 nuevos pixels. Inicialmente se colorean con el mismo color que el pixel original y a continuación se hacen una serie de operaciones lógicas que se muestran a continuación:



El resultado no es tan bueno como el de XBR (uno de los más avanzados) pero es muy aceptable y su implementación es muy sencilla [46].

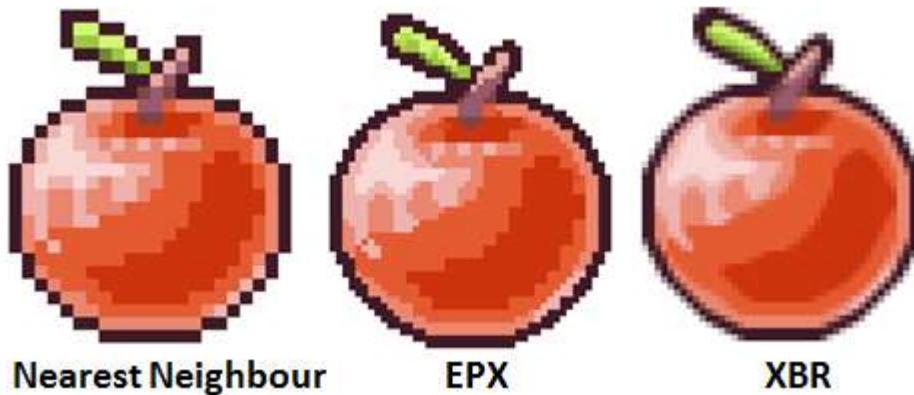


Figura 46. Comparativa de dos métodos de interpolación pixel Art

## 2.7 Medidas de calidad de imágenes

Las medidas de calidad de imágenes pueden ser objetivas y subjetivas

- **Objetivas:** realizadas por una función o algoritmo. Hay dos tipos de medidas objetivas
  1. Con referencia (FR: “Full Reference”): se dispone de la imagen original sin distorsión (imagen de referencia). Las métricas más conocida son el PSNR (Peak Signal to Noise Ratio) y el SSIM (Structured similarity Index Metric)
  2. Sin referencia (NR “No reference”): no se dispone de imagen de referencia: BRISQUE [47]es una conocida métrica de este tipo
- **Subjetivas:** llevada a cabo por personas, a esta métrica se la llama MOS (Mean Opinion Score) y es la base para evaluar las métricas objetivas. Aunque el MOS es la medida más utilizada, también se utiliza el DMOS asociado a la distorsión percibida entre dos imágenes (original y degradada). En este caso en las evaluaciones subjetivas se le muestra a los usuarios la imagen original y la degrada una al lado de la otra y se les pide puntúen en una escala predefinida la distorsión percibida.

En esta sección voy a introducir el cálculo del PSNR y de SSIM. La medida de calidad más conocida y simple de calcular es el PSNR si bien SSIM presenta una mayor similitud con el HSV (Human Vision System) [48].

### 2.7.1 PSNR

PSNR es método más simple y mayormente usado para medir la calidad. Es una buena medida para comparar diferentes resultados con una misma imagen. Sin embargo, las comparaciones entre PSNR de diferentes imágenes carecen de significado. Una imagen con 20dB PSNR puede tener una apariencia mucho mejor que otra con 30dB PSNR. Esto significa que las graficas de PSNR promediadas de PSNR para un conjunto de imágenes no tienen significado “claro” por si mismo aunque si lo tiene la comparación entre graficas realizadas con diferentes compresores.

Para definirla se hace indispensable la formulación del error cuadrático medio, que para dos imágenes monocromas I y K de tamaño M×N se define como:

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \|I(i,j) - K(i,j)\|^2$$

Y la métrica PSNR se calcula como:

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

Donde  $MAX_I$  denota el máximo valor que puede tomar un píxel en la imagen (con 8 bit por muestra será 255). Para una imagen en formato RGB, la definición del PSNR es la misma, pero el MSE se calcula como la media aritmética de los MSEs de los tres colores (R, G y B).

El comité MPEG emplea un valor umbral informal de 0,5 dB en el incremento del PSNR para decidir si se incluye una determinada mejora en un algoritmo de codificación, ya que se considera que este aumento del PSNR es apreciable visualmente.

## 2.7.2 SSIM

Se considera información estructural de una imagen todos aquellos atributos que no son la información de luminancia ni de contraste. Para poder valorar la información estructural, SSIM define dentro de su ecuación, un componente llamado  $S(x,y)$  que va a estimar la similitud de las formas que aparecen en las imágenes  $x$  e  $y$ . Es un componente que como veremos está basado en la varianza de ambas imágenes.

Sean “ $x$ ” e “ $y$ ” dos imágenes, y sean  $\mu_x$ ,  $\mu_y$ ,  $\sigma_x$ ,  $\sigma_y$  y  $\sigma_{xy}$  la media de  $x$ , la media de  $y$ , la varianza de  $x$ , la varianza de  $y$ , y la covarianza de  $x$  e  $y$ , respectivamente.

$$\begin{aligned}\mu(x) &= \bar{x} = \frac{1}{N} \sum_{i=1}^N x_i & \mu(y) &= \bar{y} = \frac{1}{N} \sum_{i=1}^N y_i \\ \sigma_x^2 &= \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2 & \sigma_y^2 &= \frac{1}{N-1} \sum_{i=1}^N (y_i - \bar{y})^2 \\ \sigma_{xy} &= \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x}) \cdot (y_i - \bar{y})\end{aligned}$$

La media y la desviación típica (raíz cuadrada de la varianza) de una señal se consideran estimaciones de la luminancia de la señal. La covarianza (normalizada por la varianza) puede considerarse como una medida de cuánto cambio no lineal posee una señal con respecto a la otra.

Se definen las medidas de luminancia, contraste y comparación estructural de la siguiente forma:

$$l(x,y) = \frac{2\mu_x\mu_y}{\mu_x^2 + \mu_y^2}, \quad c(x,y) = \frac{2\sigma_x\sigma_y}{\sigma_x^2 + \sigma_y^2}, \quad s(x,y) = \frac{\sigma_{xy}}{\sigma_x\sigma_y}$$

Aunque  $s(x,y)$  no usa una descripción representativa directa de las estructuras de las imágenes, refleja la similitud entre las dos estructuras.

Por último, el índice de similitud se define como el producto de todas ellas:

$$S(x,y) = l(x,y) \cdot c(x,y) \cdot s(x,y) = \frac{4\mu_x\mu_y\sigma_{xy}}{(\mu_x^2 + \mu_y^2) \cdot (\sigma_x^2 + \sigma_y^2)}$$

El algoritmo SSIM se aplica en evaluación de calidad de imágenes estáticas usando una aproximación de ventana deslizante. El tamaño de la ventana se fija a 8x8 píxeles. El índice SSIM se calcula para la región contenida en la ventana, que se desplaza píxel a píxel desde la parte superior izquierda a la parte inferior derecha de la imagen. Esto resulta en un mapa de índices SSIM de la imagen, que se considera el mapa de calidad de la imagen en evaluación. El valor global se define como la media del mapa de calidad, es decir, el índice SSIM medio (MSSIM).

El SSIM satisface las siguientes condiciones

- $\text{SSIM}(x,y) = \text{SSIM}(y,x)$
- $\text{SSIM}(x,y) \leq 1$
- $\text{SSIM}(x,y) = 1$  si y sólo si  $x=y$  (en representación discreta,  $x_i = y_i$  para todo  $i=1,2,\dots,N$ )

Existen implementaciones libres de esta métrica, como la implementación del proyecto “kanzi” [49] disponible en GitHub.

## 2.8 Formatos y Algoritmos de compresión de video

Para abordar esta sección voy a introducir conceptos generales y después las estrategias fundamentales de todos los codificadores de video.

### 2.8.1 Conceptos

**Formato contenedor:** es el formato del archivo, el que lo envuelve todo (audio, video, subtítulos, etc.). Formatos contenedores son AVI, MOV, FLV, ASF, MP4, MKV, WebM, VoB, OGG. Normalmente aceptan cualquier códec de video.

**Códec:** especificación concreta de compresión de video (*H.262, H.263, H.264, VP8, divx, xvid, etc.*)

Algunos codecs/estárdares que han tenido/tienen gran difusión:

- MPEG-1 y MPEG-2: son algo más que codecs. Son especificaciones que definen contenedor y codecs de video y audio. Los codecs de video que utilizan son VCD y H262 respectivamente. En los DVDs se utiliza el estándar MPEG-2 y como contenedor se utiliza el VoB.
- MPEG-4: al igual que MPEG-1 y MPEG-2 define mucho más que un códec de video. De hecho en MPEG4 se definen el contenedor MP4 y los codecs de video h263 y H264
- X264: es una implementación de la especificación AVC H264 (advanced video coding) [50]
- VP8: codec desarrollado inicialmente por la empresa On2 en 2008 y adquirido por google. La codificación intraframe de VP8 es precisamente el formato WebP de compresión de imágenes. VP8 está liberado como código abierto bajo licencia similar a BSD.
- VP9: es la nueva versión mejorada del códec VP8 de video de google, presentado en 2013.
- MPEG-H/HEVC (High efficiency video coding) [51] [52]: estas especificaciones definen el contenedor MMT y el estándar H265.

**Formato/extensión de fichero:** un formato o extensión de fichero utiliza un contenedor y un códec. Por ejemplo:

- Un fichero WMV o WMA en realidad es un fichero en cuyo interior hay un contenedor ASF y el contenido es de video o de audio respectivamente.
- Otro ejemplo es la extensión .M4V de Apple que en realidad contiene MPEG-4 con códec h264
- El formato “.3GP” es extensión y a la vez formato contenedor. Es utilizado en dispositivos móviles. El códec inicialmente usado en 3GP fue H263 aunque ha sido reemplazado posteriormente por el códec h264.

- MJPEG (Motion JPEG): en realidad emplea codificación de fotogramas por JPEG, de modo que no puede considerarse un códec de video sino simplemente un formato de fichero, pues solo tiene codificación intra-frame, es decir no relaciona unos fotogramas con otros para ahorrar información.

## 2.8.2 Estrategia general de algoritmos de compresión de video

Los algoritmos de compresión de vídeo como el VP9 y el H.264 utilizan la codificación inter-frame para comprimir una serie de fotogramas.

La codificación inter-frame se basa en dos técnicas fundamentales:

- Codificación diferencial: en la que un fotograma se compara con un fotograma de referencia y sólo se codifican las diferencias
- Estimación de vectores de movimiento: para desplazar bloques del fotograma de referencia y codificar la información diferencial respecto de esos bloques desplazados. Si la estimación es buena se puede ahorrar mucha información, sobre todo en secuencias con desplazamientos horizontales (y/o verticales) de la cámara.

La estimación de vectores de movimiento (también llamada “compensación de movimiento”) tiene en cuenta que gran parte de un fotograma nuevo está ya incluido en el fotograma anterior o posterior, aunque quizás en un lugar diferente del mismo. Esta técnica divide un fotograma en una serie de macrobloques (bloques de píxeles). Se puede componer o “predecir” un nuevo fotograma bloque a bloque, buscando un bloque que coincida en un fotograma de referencia. Si se encuentra una coincidencia, el codificador codifica la posición en la que se debe encontrar el bloque coincidente en el fotograma de referencia.

El cálculo de vectores de movimiento es muy costoso computacionalmente puesto que hay que analizar, uno a uno, cada macrobloque de la imagen y buscar su análogo (un área que se le parezca) en otro frame (esto se conoce como “block matching”). De esta manera, dos tercios del tiempo total de codificación se invierten en la estimación de vectores, lo que hace inviable en ocasiones los juegos online donde la latencia total es crítica para no afectar a la jugabilidad. En el caso de H264 además se pueden usar varios frames como referencia (“multi-picture motion-compensated prediction”), de modo que aunque pueda ser muy óptimo en bit-rate, los cálculos involucrados son muy intensivos.

Con la codificación inter-frame, cada fotograma se clasifica como un tipo “I”, “P” o “B”:

Un fotograma I, o “intra-frame”, es una imagen autónoma que se puede codificar de forma independiente sin hacer referencia a otras imágenes. La primera imagen de una secuencia de vídeo es siempre un fotograma I. Los fotogramas I sirven como puntos de inicio en nuevas visualizaciones o como puntos de resincronización si la transmisión de bits resulta dañada. Los fotogramas I se pueden utilizar para implementar funciones de avance o retroceso rápido o de acceso aleatorio. Un codificador insertará automáticamente fotogramas I a intervalos regulares. Al frame “I” junto con los frames siguientes hasta el siguiente frame “I” es lo que se conoce como GOP (Group of pictures). El GOP comienza siempre con un frame “I”. La desventaja de los frames “I” es que consumen mucha información.

Un fotograma P (Predictivo), hace referencia a partes de fotogramas I o P anteriores para codificar el fotograma. Los fotogramas P suelen requerir menos bits que los fotogramas I, pero con la desventaja de

ser muy sensibles a la transmisión de errores.

Un fotograma B, (Bidireccional-predictivo), es un fotograma que hace referencia tanto a fotogramas anteriores como posteriores. El uso de fotogramas B aumenta la latencia pues no puede ser decodificado hasta que es recibido el frame de referencia que puede ser posterior. Para ilustrarlo, en la secuencia siguiente aparece un árbol completo en el tercer fotograma, que es referenciado en fotogramas intermedios donde aparece parcialmente el árbol. Con fotogramas B pueden especificarse los bloques que constituyen el árbol parcialmente visible en frames intermedios con un vector de movimiento referenciado al frame posterior que contiene el árbol completo.



Figura 47. Aplicación de frames B

Las imágenes B no pueden existir en aplicaciones de cloud gaming ya que no disponemos de un *frame* posterior al actual, se trata de un momento del tiempo que aún no ha ocurrido y no se puede retrasar la transmisión (buffering), tal y como se hace en la emisión en directo de televisión digital (TDT), puesto que esto perjudicaría a la experiencia de juego.

Aunque los codificadores comparten estas dos técnicas básicas (codificación diferencial y vectores de movimiento), existen diferencias entre codificadores en función de cómo codifican cada frame “I”, (por ejemplo VP8 usa WebP y H264 usa H264 intra-frame), en función de cómo estiman los vectores de movimiento (con más o menos cálculos y mayor o menor precisión) o técnicas como el filtro de eliminación de efecto de bloques de H264.

### 2.8.3 Cloud gaming y video interactivo

El “cloud-gaming” es un tipo de juego online en el que tanto la lógica como el motor gráfico del juego se encuentra en la nube, de modo que las imágenes deben ser transmitidas desde el servidor mediante streaming hacia el usuario. El juego en la nube, a diferencia de lo que solemos encontrar en muchos juegos online, como el famoso “World of warcraft” no suele ser multiusuario. En “cloud-gaming” el reto consiste en lograr una jugabilidad fluida a pesar de no tener un motor gráfico ejecutándose localmente como ocurre en los juegos convencionales y en el resto de los juegos online.

Los juegos en la nube ofrecen muchas ventajas a jugadores y desarrolladores de juegos, pero también plantean nuevos desafíos, especialmente en términos de complejidad computacional en lo que respecta a latencias de codificación. Las latencias involucradas en las aplicaciones de video interactivo se reflejan en la siguiente figura.

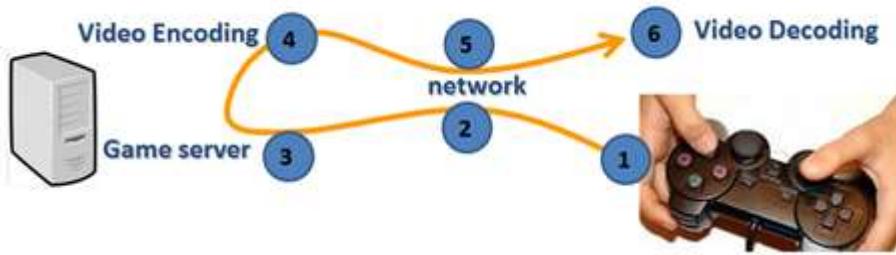


Figura 48. Latencias en cloud-gaming

Entre que un jugador realiza una acción y ve sus efectos en la pantalla pasan al menos:

$$\text{latencia}_{\text{total}} = 2 \cdot \text{latencia}_{\text{red}} + \text{latencia}_{\text{encoder}} + \text{latencia}_{\text{decoder}}$$

Si la latencia de codificación es elevada, el servidor debe estar más cerca de casa del usuario y en consecuencia son necesarios más servidores para llegar a toda la población deseada. En 2012, se informó oficialmente que los problemas financieros de OnLive (empresa que ofrecía un servicio de cloud gaming en USA) eran debido a su incapacidad para sostener los numerosos servidores necesarios para ejecutar el servicio. Esto sugiere que cualquier mejoría en la latencia de codificación del lado del servidor sería importante para hacer viable la rentabilidad del negocio de “cloud-gaming”.

En experimentos llevados a cabo en el contexto del proyecto “SMARTSTREAM” (IPT-430000-2010-071) se midió la latencia de codificación + decodificación de H264 usando la herramienta “streammygame” y un reloj cuya imagen se enviaba del servidor al cliente. Puesto que ambas máquinas estaban en la misma red la latencia de red es despreciable y la diferencia entre los relojes es aproximadamente la latencia de codificación/decodificación (entre 30 y 40 ms). Esta herramienta comprime poco, ya que para ser apta en “cloud-gaming” usa pocos cálculos de compensación de movimiento.



A la derecha se muestra el servidor, marcando el segundo 12.93 y a la izquierda el cliente, con 12.97. La diferencia son 4 centésimas de segundo, es decir, 40 ms.

Figura 49. Medición de la latencia de codificación

Como norma general, y como conclusión del estudio realizado en el proyecto de I+D “SMARTSTREAM” se aprecia que 60 ms de latencia de red (en el caso de 30-40 ms de latencia de codificación/decodificación de video) es el límite para una experiencia de juego satisfactoria independientemente del tipo de juego.

Sujeto1		Latencia en ms							
JUEGO	TIPO	10	20	40	60	80	90	100	150
BJWD3	Puzzle								
NFSHP2	Racing								
SS2	FPS								
RoN	RTS								

Tabla 6. Experiencia de usuario con diferentes latencias de red y tipos de juego

Pasar de decenas de milisegundos de codificación de cada fotograma a unos pocos milisegundos puede proporcionar un cambio relevante en el futuro del “cloud-gaming” tanto técnicamente (mejor experiencia) como económicamente. Hay dos formas de lograrlo: con hardware más rápido y capacidad multiproceso o bien con algoritmos más eficientes. La segunda opción no solo es más elegante sino que permitiría aprovechar aun más las capacidades de un hardware potente y ahorraría energía, esencial en dispositivos móviles.

## 2.9 Conclusiones y cualidades deseables para un nuevo algoritmo

Tras esta revisión de algoritmos y formatos, se puede elaborar una lista de cualidades que debería tener un algoritmo para representar un avance computacional considerando como objetivo último lograr un codificador multimedia de latencia mínima que pueda aplicarse en aplicaciones tipo cloud-gaming:

- Debe evitar las complejas transformaciones al dominio de la frecuencia y lograr una complejidad temporal lineal. Como presentaré en los próximos capítulos, LHE logra esto mediante la codificación logarítmica y el downsampling elástico.
- Debe poseer una reducida complejidad espacial, evitando los costosos procedimientos de JPEG2000 que requieren el uso de mucha memoria. En el caso de LHE, los recursos de memoria requeridos para su ejecución son mínimos.
- Debe poder tener capacidad para soportar mecanismos predictivos pero no probando varios y escogiendo el mejor (como hace PNG) sino decidiendo sin el coste que supone probar cada uno. En LHE gracias a la métrica de relevancia perceptual podemos conocer a priori cual sería la mejor predicción para cada bloque en la que se divide la imagen.
- Debe tener cualidades que permitan usarlo en video evitando el alto coste que suponen los cálculos de compensación de movimiento. En LHE, la evolución de la malla de valores de relevancia perceptual nos va a ofrecer una posibilidad interesante para realizarlo. Este punto está fuera del alcance de la tesis pero se contempla su uso futuro.

Además de estas cualidades, deberá ser paralelizable y por último no debe estar sujeto a patentes para evitar barreras a su difusión. El algoritmo LHE trata de dar respuesta a todas estas necesidades.



# Capítulo 3

## LHE básico

### 3.1 Presentación del capítulo

En este capítulo se presenta el nuevo algoritmo de codificación de imágenes “LHE” con las siguientes características:

- Opera en el dominio del espacio (no necesaria transformación a dominio frecuencia).
- Basado en el modelado del comportamiento del ojo humano: su respuesta logarítmica, su umbral de detección y su proceso de acomodación al brillo medio.
- Complejidad computacional lineal. Es su característica más importante .
- Paralelizable.

El algoritmo presentado en este capítulo es básico (tiene ciertas limitaciones importantes) pero es un paso fundamental en el diseño (y comprensión) del compresor LHE avanzado. Su limitación más importante es que no permite escoger el grado de compresión deseado, tan sólo permite comprimir a un ratio no configurable, que depende de la imagen y que se encuentra entre 1:3 y 1:5. La calidad obtenida se encuentra a medio camino entre JPG y JPEG2000, aunque supera a ambos en menor complejidad computacional, ya que se trata de un algoritmo lineal

El algoritmo LHE básico se complementa con las ampliaciones tratadas en el siguiente capítulo de esta tesis, donde se presentará un LHE avanzado y sin limitaciones.

### 3.2 Fundamentos de LHE: modelado del ojo humano

El modelo de codificación de LHE toma como referencia el comportamiento del ojo humano. En concreto se fundamenta en tres características fundamentales:

- La respuesta logarítmica al estímulo luminoso.
- El umbral mínimo de contraste lumínico para detectar cambios.
- El proceso de acomodación del ojo al brillo medio local.

Estas tres características serán analizadas a continuación y van a ser modeladas dentro del algoritmo LHE.

La idea fundamental de recrear el comportamiento fisiológico del ojo mediante un modelado de complejidad lineal es la base de LHE y supone una de las contribuciones principales de esta tesis

#### 3.2.1 Respuesta logarítmica al estímulo: Ley de weber

El principal fundamento de LHE es la aplicación de la ley psicofísica de Weber-Fechner [53], la cual establece una relación cuantitativa entre la magnitud de un estímulo físico y cómo éste es percibido. Fue propuesta en primer lugar por Ernst Heinrich Weber (1795-1878), y elaborada hasta su

forma actual por Gustav Theodor Fechner (1801-1887). Ernst Heinrich Weber estableció su ley de la sensación (o Ley de Weber) en la que formulaba la relación matemática que existía entre la intensidad de un estímulo y la sensación producida por éste.

Según esta ley, la relación que existe entre la intensidad de un estímulo  $I$  y la sensación producida por éste  $S$  viene dada por la fórmula:

$$S = C \cdot \log \frac{I}{I_0}$$

Ecuación 10. Ley de Weber

Donde  $C$  es una constante y  $I_0$  es la intensidad mínima del estímulo capaz de ser percibida

Básicamente lo que esta ley establece es que **si un estímulo crece como una progresión geométrica, la percepción evolucionará como una progresión aritmética**.

Para valores pequeños de  $I$  el individuo aprecia pequeños cambios, pero cuanto mayor sea  $I$  mayores tienen que ser los cambios para que se aprecien. Esta ley es aplicable a cualquier percepción sensorial: sensación de peso de un objeto, sensación sonora, sensación lumínica, etc.

La siguiente gráfica [54] muestra la relación entre la intensidad lumínica y la respuesta del ojo, concretamente la apertura de la pupila. Se puede apreciar como al aumentar geométricamente la luz (eje horizontal), el área de la pupila disminuye linealmente

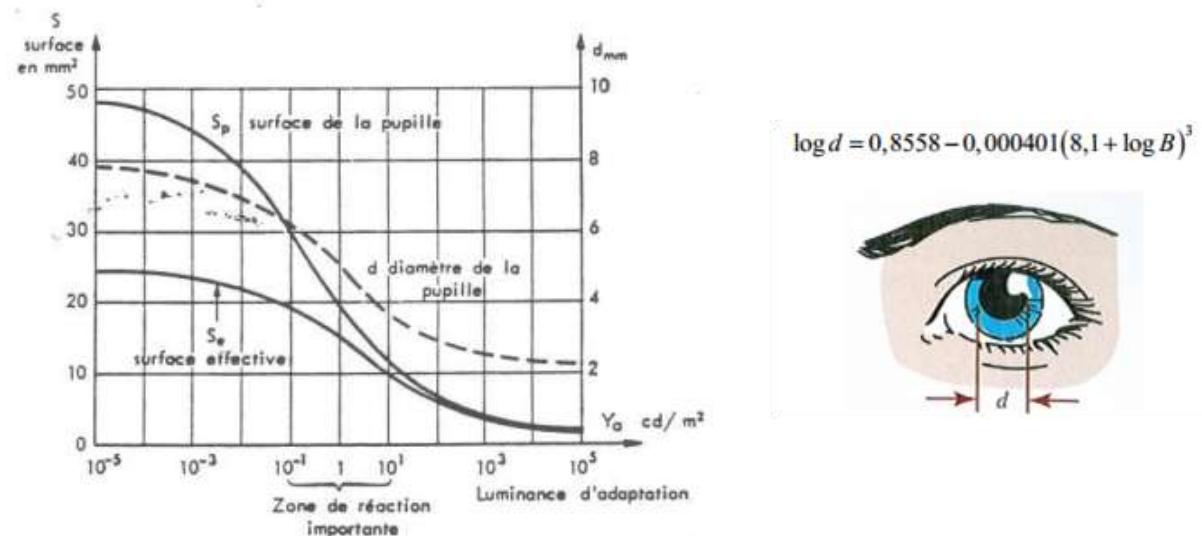


Figura 50. Reacción lineal de la pupila ante la variación geométrica de la luz

Esta respuesta del ojo se traduce en una capacidad de diferenciación de luminancias que dependen del contraste con el fondo. Es decir, **Los humanos vemos objetos si nuestro ojo puede diferenciar su tono del tono del fondo** [55]. En la siguiente figura se muestra la sensibilidad al contraste, cuya parte más significativa debemos considerarla entre las luminancias comprendidas entre  $10^{-1}$  y  $10^1$ . Dicha sección de la curva es aproximadamente lineal, tal como establece la ley de Weber [56].

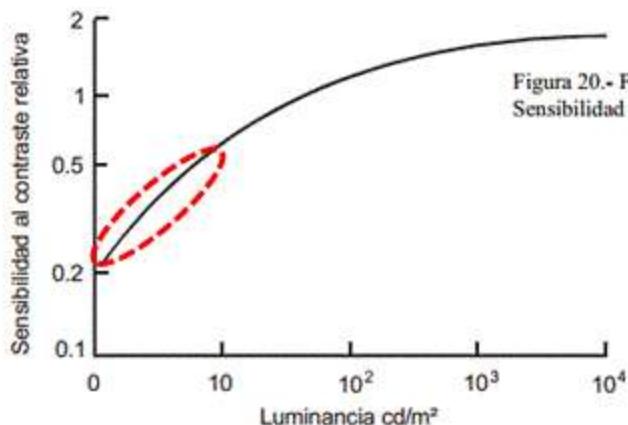


Figura 51. Sensibilidad al contraste del ojo humano

Como conclusión podemos afirmar que el ojo humano codifica información sensorial basándose en la diferencia geométrica entre la luminancia de un objeto y el fondo. Esta información logarítmica es la que se transmite desde el ojo hacia el cerebro para su posterior procesado.

### 3.2.2 Modelado en LHE de la ley de Weber

En LHE no hay transformaciones al dominio de la frecuencia. Por el contrario LHE trabaja escaneando la imagen pixel a pixel y codificando la información de cada pixel. Modelar el comportamiento de la respuesta del ojo según la ley de weber implica definir dos aspectos:

- Estimar el color de fondo para cada pixel
- Codificar la información mediante saltos logarítmicos entre el fondo estimado y la señal

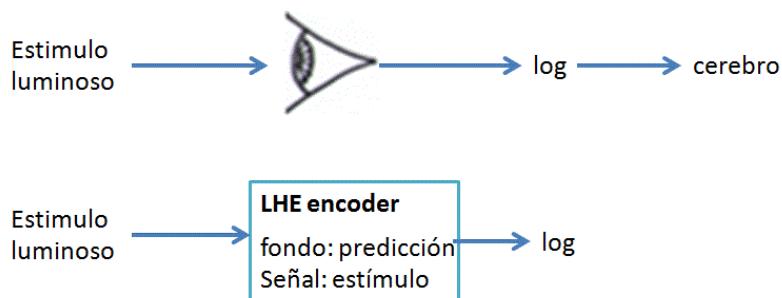
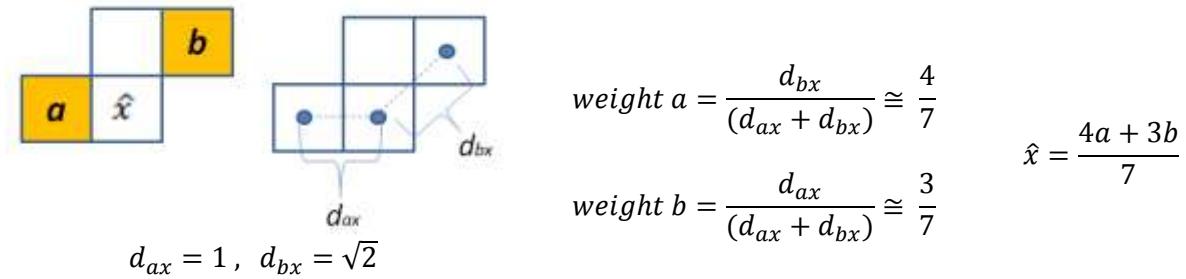


Figura 52. Modelado de la respuesta logarítmica del ojo humano

#### 3.2.2.1 Estimación del color de fondo de un pixel

Estimaremos el color de fondo (o también lo llamaremos “*hop nulo*” o  $hop_0$ ) de la componente ( $Y$ ,  $U$  o  $V$ ) de cada pixel, como el promedio ponderado de las componentes de color del pixel izquierdo ( $a$ ) y superior-derecho ( $b$ ). Este color de fondo es una estimación del valor de la señal, ya que en zonas lisas o gradadas, no hay diferencia entre el fondo y la señal.



Ecuación 11. Predicción de la componente (YUV) para un pixel

Esta predicción permite obtener un color de fondo mejor aproximado que si utilizásemos el pixel izquierdo y el superior (Figura 53 a), ya que al estar más separados a y b, el pixel x queda más centrado, al menos en la coordenada horizontal. Para centrar la coordenada vertical tendríamos que escoger el pixel abajo-izquierdo en lugar del pixel izquierdo pero como LHE opera en scanlines, dicho pixel aun no ha sido calculado y no disponemos de él (Figura 53 c).

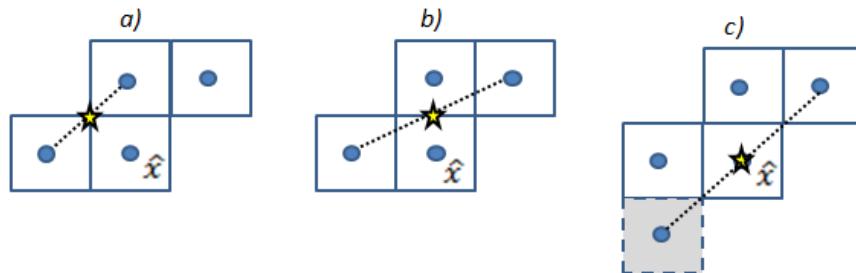


Figura 53. Diferentes aproximaciones del color de fondo

En cuanto a los casos especiales, como son el lado horizontal superior y los lados verticales, la predicción se realiza tomando como única referencia el pixel adyacente, tal y como se ilustra en la Figura 54. Finalmente, el pixel de la esquina superior izquierda, al no tener ninguna referencia (no existen ni "a" ni "b"), su valor no se predice, sino que almacenaremos su valor original

$x_0$				<b>a</b>	<b>x1</b>			$\widehat{x}_0 = \text{original value}$
<b>b</b>							<b>b</b>	
$x_2$			<b>b</b>		<b>a</b>	<b>x4</b>		
		<b>a</b>	<b>x3</b>					$\widehat{x}_1 = a$
								$\widehat{x}_2 = b$
								$\widehat{x}_3 = \frac{4a + 3b}{7}$
								$\widehat{x}_4 = \frac{a + b}{2}$

Figura 54. Casos posibles y sus referencias disponibles para la predicción

La estimación o predicción del color de fondo es parte del modelado del ojo en el que se basa LHE y por consiguiente forma parte del algoritmo LHE y es una contribución de esta tesis

### 3.2.2.2 Codificación por saltos logarítmicos

Utilizando como valor de referencia esta predicción de color de fondo, definiremos un conjunto de hops positivos y negativos, con los que codificaremos el logaritmo de la diferencia entre la señal y la predicción. Al final simplemente almacenaremos el identificador del hop más cercano a la señal y no el valor de su luminancia correspondiente. Es por ello que habremos reducido los grados de libertad originales (256) a tan solo 9 posibilidades (hop nulo + 4 hops positivos + 4 negativos).

Para denominar a los hops usaremos la siguiente notación:

- **$h_i$ :** Valor relativo del hop, es decir, aquel valor de salto respecto al valor de fondo. Por ejemplo, Un  $h_i = 23$  significa que si aplicamos dicho salto y tenemos un color de fondo=100 entonces tendremos un resultado de 123
- **$hop_i$ :** Es el valor absoluto del hop, es decir, el valor de la componente de color (Y, U o V) tras al aplicar el hop  $h_i$  al color de fondo. En el ejemplo anterior  $hop_i = 123$ , mientras que  $h_i = 23$

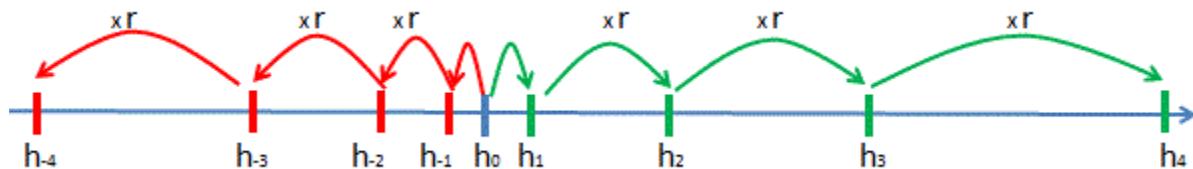


Figura 55. Hops positivos y negativos distribuidos a lo largo del rango de luminancia disponible

*El primer hop ( $h_1$ ) es un caso especial ya que se relaciona con el umbral mínimo de detección de contraste del ojo y será analizado posteriormente. De momento simplemente adelanto que consideraremos un valor variable entre 4-10.*

En cuanto al resto de hops, una vez que tenemos  $h_1$ , el cálculo es inmediato. Simplemente multiplicamos por una razón constante cuyo valor abordaremos a continuación.

$$\begin{aligned} h_2 &= h_1 \cdot r & h_3 &= h_2 \cdot r & h_4 &= h_3 \cdot r \\ h_{-2} &= h_{-1} \cdot r & h_{-3} &= h_{-2} \cdot r & h_{-4} &= h_{-3} \cdot r \end{aligned}$$

Ecuación 12 Relaciones entre hops

La razón debe ser tal que el hop mayor no exceda del límite de luminancia disponible, de modo que la razón debe limitarse a un cierto valor máximo, que va a depender de  $h_0$ . Una razón muy grande tiende a separar los hops mientras que una razón muy pequeña tiende a compactarlos. Compactar los hops es útil cuando estamos codificando una región de cambios de luminancia suaves, mientras que expandirlos

es adecuado en zonas de cambios de luminancia abruptos. Consideraremos estas tres posibilidades:

- Una razón con valor 2 permite cubrir un rango de +/-48 valores de luminancia cuando  $h_1$  es 4. Esta razón es óptima para zonas suaves pues permite discernir entre valores de luminancia muy próximos.
- Si queremos codificar una zona abrupta y cubrir todo el intervalo (+/- 255) de luminancias con  $h_1=8$ , la razón optima sería  $r_{opt}=3$ , ya que permite discernir entre luminancias muy dispares
- A medio camino está  $r_{opt}=2.5$ , que es adecuada tanto para zonas suaves como para abruptas, siendo idónea en zonas intermedias.

Escogeremos un valor de  $r_{opt}=2.5$ , pero debemos cuidar de que ningún hop exceda el valor máximo del rango disponible, ya que podríamos salirnos del intervalo si con la predicción  $h_0$  nos acercamos mucho al valor máximo (255) o al mínimo (0). Para limitar el valor máximo de la razón, vamos a limitar a que  $h_4$  como mucho alcance un 80% del intervalo disponible. Cuando la señal esté por encima de este valor, asociaremos  $h_4$  como el hop más cercano. Cada hop cubre un cierto intervalo por encima y por debajo del valor que representa. La fórmula de la razón será:

$$r_{opt} = 2.5 \quad h_4 = 0.8 \cdot (255 - h_0) \quad \text{siendo} \quad h_4 = h_1 \cdot r^3$$

y por consiguiente,

$$r_{max} = \left( \frac{0.8 \cdot (255 - h_0)}{h_1} \right)^{1/3}$$

$$r = \min(r_{opt}, r_{max})$$

**Ecuación 13. Cálculo de la razón geométrica que relaciona los hops**

Este cálculo también hay que hacerlo para el rango de luminancia negativo, por debajo de  $h_0$ , con lo que obtendremos una razón para los hops positivos y otra para los negativos. La única diferencia en las fórmulas sería el rango disponible, que esta vez sería  $h_0$ , en lugar de  $255 - h_0$ .

La codificación por saltos logarítmicos y el cálculo de su razón geométrica forman parte del modelado del ojo en el que se basa LHE y por consiguiente son parte del algoritmo LHE y son contribuciones de esta tesis

### 3.2.2.3 Algoritmo de selección de hop

Una vez que tengamos los hops pre-calculados, podemos codificar cada pixel de la imagen con un hop. Esto se realiza mediante un proceso lineal, cuyo coste es proporcional al número de píxeles. Este algoritmo va a seleccionar entre los hops disponibles, el más cercano a la señal. En caso de poder escoger entre dos hops debido a que ambos producen el mismo error, escogeremos siempre el más pequeño (el más cercano a  $h_0$ ) ya que como veremos en el apartado correspondiente al codificador binario, los hops se codificarán con códigos de longitud variable y los hops menores ocupan menos bits.

```

//Pixel Prediction
 $\hat{x} = (a + b)/2$ 

//Best Hop Selection
 $error_{min} = |x - \hat{x}|$ 
 $hop_{selected} = 0$ 

//Positive Hops
if ( $x > \hat{x}$ ) then
    foreach ( $h_i$  in  $h_1 \dots h_4$ ) do //hops can be precomputed
         $error = |x - h_i|$ 
        if ( $error < error_{min}$ ) then
             $hop_{selected} = i$ 
             $error_{min} = error$ 
        else
            break
        end
    end

//Negative Hops (same loop for  $h_{-1} \dots h_{-4}$ )
else
    ...
end

```

De cara a un alto rendimiento del algoritmo tendremos pre-calculados los valores de los hops (excepto los de  $h_0$ ,  $h_1$  y  $h_{-1}$ ) para cada valor de  $h_0$ . Estos pre-cálculos son válidos para cualquier imagen, y pueden estar almacenados en una tabla que se cargue inicialmente en el codificador. Puesto que  $h_0$  puede tomar 256 valores y hay 6 hops a pre-calcular, la tabla tendrá  $6 \times 256 = 1536$  entradas, poco mas de 1KB. Sin embargo hay que calcular estas entradas para cada valor de  $h_1$  considerado. Si consideramos 6 valores (desde 4 hasta 10) estamos en torno a los 7KB. La reducida complejidad espacial es una de las virtudes de este algoritmo pues apenas requiere memoria para ser ejecutado.

La siguiente figura representa el resultado de la ejecución de la cuantización que LHE realiza, en este caso con una pequeña imagen de un barco. Además de los hops, el algoritmo requiere almacenar la componente de color del primer pixel de la imagen (esquina arriba-izquierda) para poder ser reconstruida.

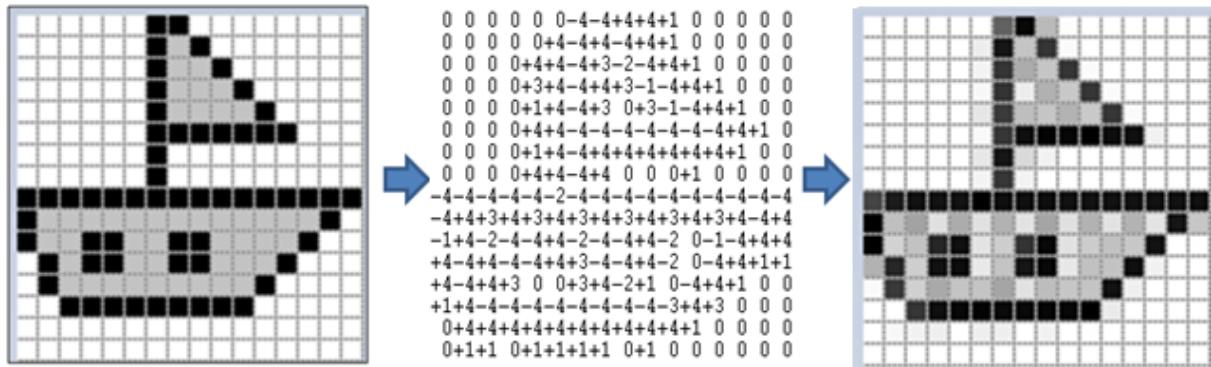


Figura 56. Ejemplo de asignación de hops y reconstrucción realizada por LHE

El algoritmo de selección del mejor hop forma parte del modelado del ojo en el que se basa LHE y por consiguiente es parte del algoritmo LHE y es una contribución de esta tesis

### 3.2.2.4 Número de hops óptimo

El número de hops escogidos (9) responde a un compromiso entre el número de grados de libertad y la calidad obtenida. Cuantos más hops tengamos, obtendremos más calidad al reproducir la señal original, pero también invertiremos más bits en ello. Esto es debido a que al distribuir más hops en el intervalo, el identificador de hop será un número de más bits.

Llega un momento que no compensa aumentar el número de hops invertidos pues la calidad aumenta residualmente y sin embargo a mayor numero de hops, mayor consumo de bits. Experimentalmente se llega a la conclusión de que ese número óptimo de hops es 9 (hop nulo + 4 hops positivos + 4 negativos).



Figura 57. Elección del número óptimo de hops del cuantizador LHE

La elección de 9 hops (4 positivos, 4 negativos y un hop “nulo”) como número óptimo para cuantizar es parte del algoritmo LHE y es una contribución de esta tesis

### 3.2.2.5 Valor de la señal para cada hop

Cada hop cubre un intervalo de la señal asimétrico respecto a la ubicación del hop tal y como se muestra en la siguiente figura:

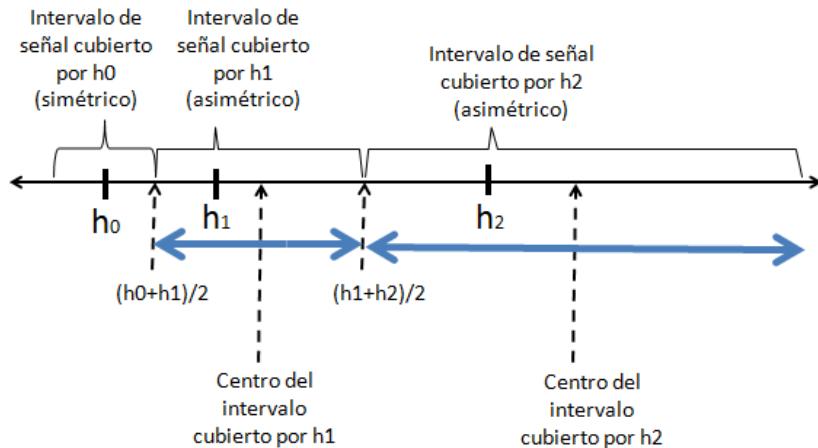


Figura 58. Centro de los intervalos cubiertos por cada hop

Esto significa que el valor de la señal que debemos asignar a un pixel codificado por un determinado hop no será el valor del hop sino el valor del centro del intervalo, pues de ese modo se minimizará el error cometido, ya que el verdadero valor de la señal puede encontrarse en cualquier punto del intervalo. Sin embargo debemos tener en cuenta tres casos especiales:

- En el caso del hop nulo  $h_0$  ambos valores coinciden pues es un intervalo simétrico.
- En los casos  $h_1$  y  $h_{-1}$  no conviene escoger el valor medio pues se encuentra más alejado del hop nulo y por lo tanto se cometerán errores mayores cuando las fluctuaciones de luminancia sean pequeñas.
- En el caso de los hops mayores casos  $h_4$  y  $h_{-4}$  el extremo del intervalo es cero y 255 respectivamente y usar el centro del intervalo puede significar aumentar mucho el salto, perdiendo la relación geométrica con el hop anterior, por lo que en lugar de mejorar, empeora.

La elección del valor medio del intervalo cubierto por cada hop como valor de reconstrucción de la señales una contribución de esta tesis

### 3.2.3 Umbral de detección del ojo humano

Para definir el umbral de detección del ojo humano [57] se va a considerar lo que se denomina fracción de weber [55], que básicamente es la variación porcentual del estímulo de luminancia de fondo capaz de ser detectada [54] [58]. Se expresa como  $\frac{\Delta I}{I}$

Consideraremos una zona iluminada con intensidad  $I + \Delta I$  rodeada de un fondo de intensidad  $I$ , tal como se muestra en la Figura 59. El momento en que la diferencia  $\Delta I$  comienza a ser notada sería el valor de la fracción de weber.

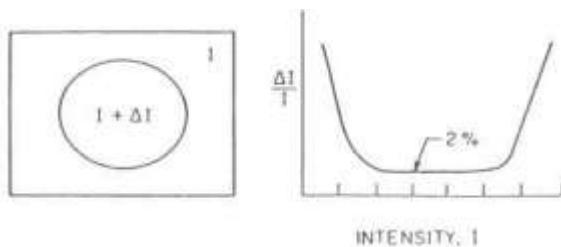


Figura 59. Fracción de weber para el ojo humano

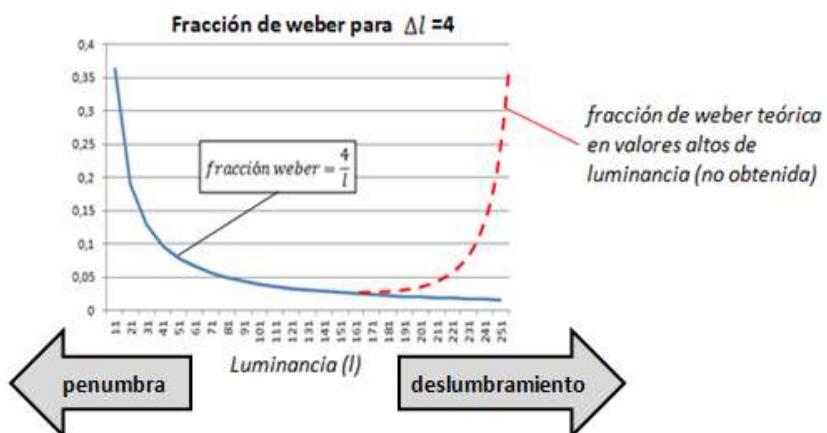
La fracción de Weber del ojo humano no supera 0.02. Es decir, el ojo detecta un 2% de cambio en el valor de la luminancia de fondo. Sin embargo este 2% se ve incrementado en altas y bajas intensidades. Es decir, el ojo se hace menos sensible [56].

Es intuitivamente sencillo comprender esta gráfica ya que en multitud de ocasiones somos capaces de llegar a reconocer o diferenciar objetos pequeños aumentando el nivel de iluminación. Esto explica como al pasar de una intensidad baja a una intensidad media, la fracción de weber baja al 2%. Si seguimos aumentando el nivel de iluminación, llega un momento que la fracción de weber vuelve a aumentar y nuestra capacidad de diferenciar objetos se vuelve a mermar

### 3.2.4 Modelado en LHE del umbral de detección

Lo que se puede concluir de este comportamiento del ojo es que el primer hop como mínimo debe permitir detectar no menos de un cambio de luminancia un 2% superior al valor del fondo (el valor del fondo es el de la de la predicción o hop nulo). O lo que es lo mismo: la separación entre el hop0 y el hop1 debe ser de al menos un 2% de la señal

También se puede concluir que el comportamiento del ojo humano para luminancias bajas y medias es el correspondiente a un valor de  $\Delta l$  constante, es decir, si en lugar de considerar un 2% de la señal, consideramos un valor constante, la gráfica sale muy similar:

Figura 60. Fracción de weber para  $h1=4$ 

Un valor mínimo de  $h1=4$  permite que una variación de luminancia de  $\Delta l = 3$  sea detectada, ya que para variaciones  $\Delta l \leq 2$  se utilizaría el hop nulo o predicción de fondo y para un valor superior a 2 ya consideraríamos el hop1, es decir  $h_0 + 4$ . Obviamente al considerar la luminancia hop1 estaríamos cometiendo un pequeño error, ya que si la señal es  $h_0 + 3$ , el valor de hop1 es una aproximación, no es exacto (es  $h_0+4$ ).

En definitiva, con  $h1=4$  se puede detectar un cambio de luminancia  $\geq 3$ , lo cual es el 2% de 150, más o menos el centro del rango de luminancias. Para valores inferiores de la señal se detectarían cambios por encima del 2% y para valores superiores de la señal se obtendría una detección inferior al 2%

Las luminancias altas dependen del brillo del dispositivo que se utilice para visualizar una imagen, de modo que es “arriesgado” utilizar un  $h1$  muy grande para luminancias altas, ya que al bajar el brillo de la pantalla de visualización, las altas luminancias se moverán hacia la zona intermedia y con un valor alto de  $h1$  no nos estaríamos ajustando a la curva de respuesta al contraste del ojo. En definitiva lo que ocurre es que los dispositivos de visualización se calibran para no producir deslumbramiento en el observador. **Por todo ello en LHE se ha definido un valor mínimo para  $h1$  de 4.**

La definición del valor mínimo del primer hop es parte del modelado del ojo en el que se basa LHE y por consiguiente forma parte del algoritmo LHE y es una contribución de esta tesis

### 3.2.5 Acomodación del ojo humano

El ojo humano una vez que se adapta al brillo medio, percibe no más de 50 niveles cercanos de brillo y todos los brillos lejanos al medio son percibidos como extremos [59]. El brillo medio es una propiedad local del punto en el cual fijamos la vista, no es el brillo de la imagen completa.

Para ilustrarlo podemos considerar la siguiente figura:

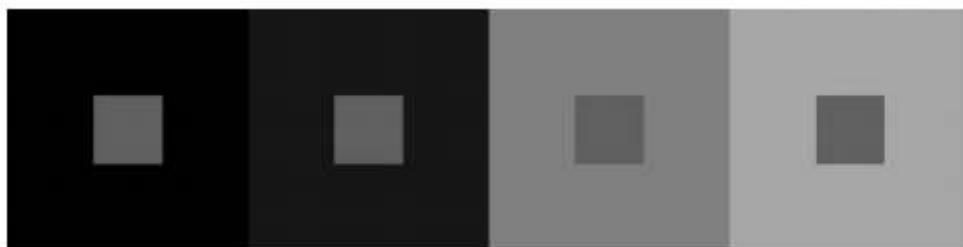
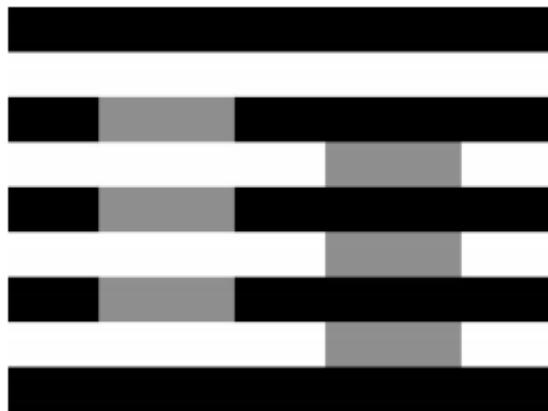


Figura 61. Acomodación del ojo al brillo medio

La imagen está formada por cuadros pequeños del mismo tono de gris rodeados de otros cuadros de mayor tamaño y de luminosidades distintas. A pesar de que los cuadros centrales son todos de la misma tonalidad, se puede observar que el cuadro pequeño situado en la parte derecha se percibe con una luminosidad menor que sus cuadros homólogos situados a su izquierda. La explicación a este fenómeno se debe a la adaptación al brillo medio que realiza el sistema de visión humano. Ante una  $\Delta l$  elevada, el ojo percibe un brillo poco preciso, mientras que cuando las fluctuaciones son pequeñas, el ojo gana precisión.

El siguiente ejemplo [60] creado por Edward H. Adelson, profesor de Ciencias de la Visión de MIT es similar y sirve para demostrar lo mismo.



**Figura 62. Acomodación del ojo al brillo medio**

Aunque los dos tonos de gris son idénticos percibimos como más claro el izquierdo. Es debido a la acomodación al brillo medio. El brillo medio de la izquierda es mayor por encontrarse delimitado por franjas blancas. Es un gris medio pero el ojo le asigna un valor erróneo más brillante del que le corresponde. En el caso derecho ocurre lo contrario, solo que el error que comete el ojo es hacia un brillo más oscuro. Siempre que un tono se aleja del brillo medio se pierde precisión en su interpretación por el ojo, ya sea por exceso o por defecto.

Si en un fondo de color blanco colocásemos un objeto muy oscuro lo percibiríamos como negro por el mismo motivo. En otras palabras, a mayor contraste, menor sensibilidad pues el ojo solo tiene precisión en los tonos cercanos al brillo medio.

### 3.2.6 Modelado en LHE del proceso de acomodación

En las regiones con altas fluctuaciones de luminancia ( $\Delta l$  elevadas) se deberá asignar hops grandes para ajustarnos lo más posible al valor de la señal. En dichas regiones lo deseable es disponer de unos hops distribuidos de modo que cubran bien el rango de luminancia disponible, es decir que estén poco compactados.

Para lograrlo, en dichas regiones se aumentará el valor de  $h_1$  hasta un cierto límite superior y con ello se expandirán todos los hops pues todos los demás se obtienen multiplicando por una razón al  $h_1$ . Si la señal deja de fluctuar de forma abrupta y se hace más suave, se disminuirá paulatinamente el valor de  $h_1$  hasta un límite inferior (el cual ya se ha argumentado que será 4 debido al umbral mínimo de detección). Cuando  $h_1$  alcance el valor mínimo, los hops estarán más próximos unos de otros, más compactados y el rango de luminancias cubierto será inferior al rango disponible, de modo que LHE ganará en precisión cerca del brillo medio, igual que el ojo humano.

El valor máximo que escogido es 10. Sin embargo, dependiendo de la imagen, el valor máximo óptimo puede ser diferente. Si la imagen tiene cambios de luminancia muy suaves se puede escoger un valor máximo inferior y el error cometido al aproximar la señal en cada hop será menor, pues los hops estarán más compactados. Si por el contrario la imagen es muy abrupta, un valor máximo de  $h_1=16$  puede ser el que mejor calidad de resultado ofrezca. No obstante existe un compromiso entre este valor y los bpp, ya que un valor bajo genera unos hops mas compactos y por lo tanto el hop  $h_i$  escogido en cada pixel tendrá un índice “ $i$ ” mayor y ocupará más bits. El valor 10 ha resultado ser adecuado en este sentido.

La regla (en pseudocódigo) que usaremos para hacer crecer y decrecer a  $h_1$  es la siguiente:

```
//tunning hop1 for the next hop
min_h1=4
max_h1=10
small_hop=false
for each pixel do
    if (selected_hop<=h1 && selected_hop>=h-1) then
        small_hop=true
    else
        small_hop=false
    endif
    if (small_hop && previous_small_hop) then
        h1=h1-1
        if (h1<min_h1) then
            h1=min_h1
        endif
    else
        h1=max_h1
    endif
    previous_small_hop=small_hop
endfor
```

Básicamente esta estrategia consiste en hacer decrecer el valor de  $h_1$  hasta un valor mínimo (4) cuando los hops que estamos seleccionando son pequeños ( $h_1$  o  $h_0$ ). Esto “compacta” los hops en un rango menor de luminancias. Por el contrario si se presenta un hop grande, haremos crecer a  $h_1$  de forma inmediata al valor máximo, con lo que los hops se expanden cubriendo así un mayor rango de luminancias, permitiendo precisar mejor el valor de la señal. **Esta estrategia simula el comportamiento de “acomodación” del ojo al nivel de brillo medio local.**

La estrategia de cambio del primer hop ( $h_1$ ) es parte del modelado del comportamiento del ojo humano en el que se basa LHE y la elección del valor máximo de  $h_1$  forman parte del algoritmo LHE y son contribuciones de esta tesis

### 3.3 Codificador binario de hops para LHE básico

En el proceso de asignación de códigos binarios a los hops se ha definido en dos pasos:

- Primeramente se transforman los hops en símbolos, teniendo en cuenta las redundancias espaciales. Un símbolo a veces significará un hop y a veces otro diferente, según ciertas reglas de redundancia.
- A continuación, se asignan códigos Huffman a los símbolos.

Si se aplicasen códigos Huffman directamente a los hops, desaprovecharía la redundancia espacial que permite ahorrar mucha información (según la imagen). Por eso primero se deben transformar los hops en símbolos que aprovechan la redundancia espacial

Se pueden tener en cuenta dos importantes características de las imágenes naturales (las imágenes típicas o naturales con las que cualquier procesamiento de imágenes debería enfrentarse como las contenidas en la galería de imágenes de test de “kodak Lossless True Color Image Suite” [61])

- **Su redundancia espacial:** un pixel se va a parecer al izquierdo (redundancia horizontal) y/o al superior (redundancia vertical). En LHE esto se traduce en que el hop más frecuente será el  $h_0$ , y/o el hop cuyo valor se corresponda con el hop situado en la posición superior.

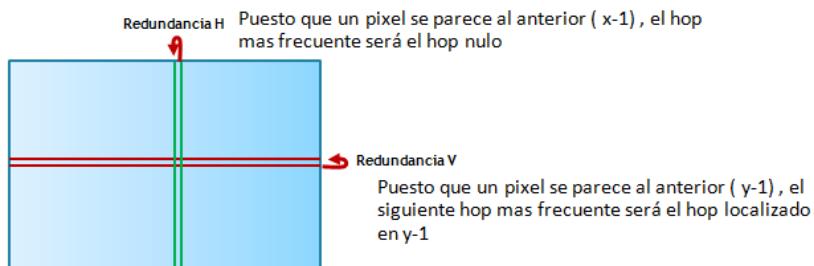


Figura 63. Redundancia horizontal y vertical

- **Su distribución estadística:** las imágenes suelen contener un porcentaje elevado de zonas suaves ogradadas y un porcentaje menor de pixels que contienen bordes delimitando dichas zonas suaves. En LHE esto se traduce en que los hops más pequeños son los más frecuentes. En la siguiente figura se muestra la distribución de hops de la imagen “Lena”. Como se aprecia, casi el 90% de los hops son el  $h_0$ ,  $h_1$  y  $h_{-1}$ , por lo que lo más adecuado es un código binario de longitud variable, siendo los códigos más cortos los de los hops más pequeños

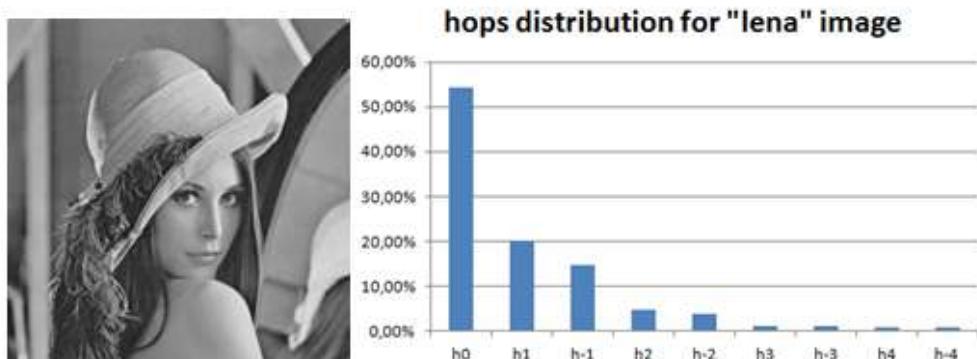
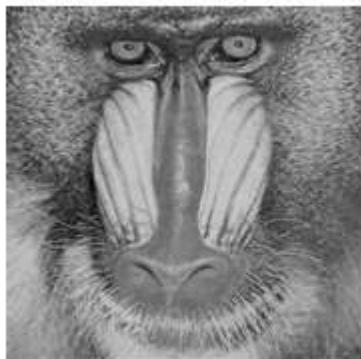


Figura 64. Distribución estadística de hops obtenidos por LHE para la imagen “Lena”



hops distribution for "baboon" image

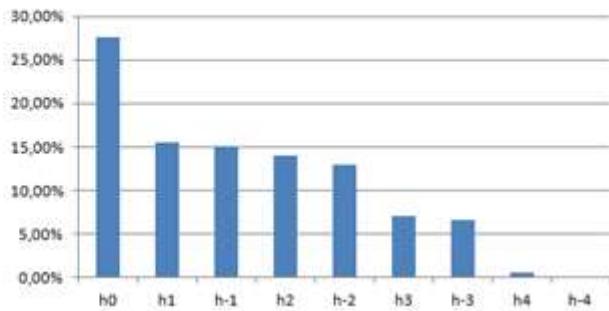


Figura 65. Distribución estadística de hops obtenidos por LHE para la imagen “baboon”



hops distribution for "peppers" image

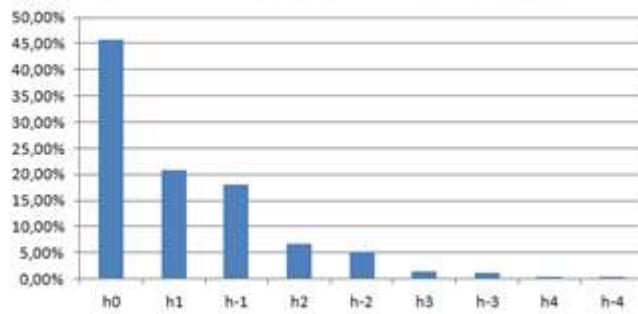


Figura 66. Distribución estadística de hops obtenidos por LHE para la imagen “peppers”

Con estas dos características se ha definido la siguiente estrategia de asignación de símbolos: Examinaremos el hop a considerar y lo compararemos con cada uno de los hops posibles siguiendo el siguiente orden:

Hop	Orden (símbolo a asignar)
$h_0$	1
up	2
$h_1$	3
$h_{-1}$	4
$h_2$	5
$h_{-2}$	6
$h_3$	7
$h_{-3}$	8
$h_4$	9
$h_{-4}$	10

Cada símbolo representa la “negación” de todos los hops correspondientes a símbolos de orden inferior. De este modo, el símbolo 3 niega al  $h_0$ , al up y al  $h_1$ .

Otra forma de verlo es identificar el símbolo con el número de comprobaciones realizadas siguiendo el orden de la tabla.

Tabla 7. Correspondencia entre hops y símbolos

El hop “up” es el hop que se encuentre en la posición superior de nuestro hop a considerar y puede ser

cualquiera de los 9 posibles hops. En la estrategia que se presenta a continuación será el segundo valor a considerar, de modo que si el hop a codificar coincide con el hop “up”, se le asignará el símbolo “2” con independencia de cuál sea. Por otro lado, si up coincide con  $h_0$ , entonces el significado del código “2” ya no será el valor de up, pues up coincide con  $h_0$ , por lo tanto “2” significará el hop  $h_1$ .

**For each pixel  $i$  do**

```

If (hop[i] ==  $h_0$ ) then
    Symbol[i] = 1
    continue

else
    discard_hop( $h_0$ )
    Symbol[i] = Symbol[i] + 1 //penalty

endif
up= hop(i-image_width) //“up” can be any hop
if (hop(i)== up) then
    Symbol[i] = Symbol[i] + 1
    continue //symbol[i] has been computed

else
    discard_hop(up)
    Symbol[i] = Symbol[i] + 1 //penalty

endif
//if reach this point, then prediction based on redundancy has failed.
symbol = symbol + symbol_table(hop(i)) // use the table of hop to symbols
if (hop(i) > up) then
    Symbol[i] = Symbol[i] - 1
endif
if (hop(i) >  $h_0$ ) then
    Symbol[i] = Symbol[i] - 1
endif
//symbol[i] has been computed
endfor

```

Existe una optimización adicional que se puede realizar, basada en la siguiente característica: Tras un borde, los tonos tienen a suavizarse, pues la textura de los objetos suele ser suave.

Esto en LHE se traduce en que tras un  $h_i > h_1$ , aumenta la probabilidad de encontrar un hop más pequeño de igual signo. En esos casos se prioriza (tras haber comprobado  $h_0$  y “up”) el hop inmediatamente inferior (o superior en caso de ser un hop negativo). Si falla esta predicción, se

descartará el hop y se penaliza sumando una unidad al símbolo.

Esta última optimización produce mejoras muy pequeñas, del orden de 0.02bpp, de modo que la complejidad que introduce es cuestionable frente al beneficio que proporciona.

Tras esta asignación de códigos, lo que resulta es una transformación de hops en símbolos, con prominencia de los símbolos más pequeños.

A continuación puesto que el número de apariciones de cada uno de los 10 símbolos es conocido, se realiza una asignación de códigos binarios mediante un algoritmo de Huffman, que al tratarse de tan solo 9 símbolos, será un proceso inmediato de complejidad  $O(1)$  y se obtienen los códigos binarios asignados de un modo inmediato. Otra posibilidad sería usar Huffman adaptativo pero puesto que Huffman estático ya comprime bastante y es muy inferior en coste, se ha optado por esta opción.

El algoritmo del codificador binario de hops aprovecha la distribución estadística de los hops de una imagen así como tres tipos de redundancias espaciales de la misma. El algoritmo del codificador binario es una aportación de esta tesis.

### 3.4 Resumen del algoritmo LHE y diagrama de bloques

A continuación se resume mediante pseudocódigo todo el algoritmo LHE básico:

*Pre-compute table of hop values using a geometrical ratio  $r=2.5$*

**For each pixel  $i$  do**

    Compute  $h_0$

    Choose the closest hop from the list of pre-computed hops. Each pair  $(h_0, h_1)$  defines a different list

    Compute  $h_1$  for the next iteration, in the interval  $[h_{1\min}, h_{1\max}]$

**Endfor**

**For each hop Assign a symbol**

    Compute Huffman binary codes

**For each Symbol assign its binary code**

El algoritmo se representa mediante dos bloques funcionales en la Figura 67 , siendo estos dos bloques el cuantizador logarítmico y el codificador binario.

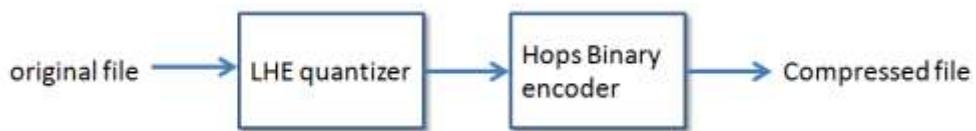


Figura 67. Diagrama de bloques del compresor LHE básico

### 3.5 Paralelización del algoritmo LHE básico

En LHE, la asignación de un hop a un pixel (es decir, la cuantización en hops) requiere el conocimiento previo de las componentes de color resultantes de haber cuantizado el pixel izquierdo y superior-derecho. Esto es debido a que para calcular la predicción de color de fondo son requeridos estos dos

pixeles.

En la siguiente figura, que representa una matriz de píxeles, aparecen etiquetados con el mismo número aquellos que pueden ser cuantizados al mismo tiempo.

1	2	3	4		1	2	3	4	5	6	7	8
3	4	5	6		2	4	5	6	7	8	9	10
3	6	7	8		3	6	7	8	9	10	11	12
4	8	9	10		4	8	9	10	11	12	13	14
5	10	11	12		5	10	11	12	13	14	15	16
6	12	13	14		6	12	13	14	15	16	17	18
7	14	15	16		7	14	15	16	17	18	19	20
8	16	17	18		8	16	17	18	19	20	21	22

Figura 68. Paralelización de operaciones

El número de píxeles originales era 16 y se puede cuantizar en 10 pasos. En la segunda matriz el número de píxeles es 64 y se puede cuantizar en 22 pasos. En cualquier imagen de lado  $N$ , con  $N^2$  pixels, se puede paralelizar en  $3N - 2$  pasos

El lado vertical izquierdo y el horizontal derecho son casos especiales. Al no disponerse de las dos referencias para calcular la predicción, se puede usar el pixel superior o anterior como alternativa.

Según la capacidad multiproceso del sistema donde se ejecute el algoritmo, se podrá reducir en mayor o menor medida el tiempo de cálculo (aunque el número de operaciones sea el mismo). El límite inferior del tiempo necesario viene dado por la formula:

$$t_p = \frac{3N - 2}{N^2} \cdot t_s \cong \frac{3}{N} \cdot t_s$$

Ecuación 14. Relación entre tiempos de ejecución

Donde  $t_p$  es el tiempo invertido con la máxima paralelización posible y  $t_s$  es el tiempo invertido mediante un procesamiento secuencial. Para valores grandes de  $N$ , la capacidad de paralelización de LHE permite reducir los tiempos de ejecución secuenciales en una proporción N:3, siendo N la longitud en pixeles del lado de la imagen.

### 3.6 Color

LHE utiliza el espacio de color YUV para representar la información de la imagen. El modelo YUV define diferentes modelos de *downsampling* para las señales de crominancia. Los más utilizados por estándares como JPEG son el modelo YUV 4:2:0 y el modelo YUV 4:2:2. Ambos modelos son aplicables en LHE.

La justificación de elección de este modelo de color es precisamente su orientación a la compresión de

la información, es decir, en los modelos YUV422 y YUV420 se ahorra información de crominancia partiendo del hecho de que el ojo humano es menos sensible a los cambios de color que a los cambios de brillo. Como resultado, una imagen puede representarse con menos información de crominancia que de luminancia sin sacrificar la calidad percibida [62].

Para aplicar el modelo YUV 4:2:2, las componentes de crominancia se escalan a la mitad únicamente en la dirección horizontal mientras que la dirección vertical mantiene su tamaño. El hecho de que los bloques de crominancia en este modelo se escalen en la dirección horizontal en lugar de en la vertical se debe al hecho de que el ojo humano es menos sensible a los cambios que se producen en la dirección horizontal que en la vertical.

Para aplicar el modelo YUV 4:2:0, cada una de las componentes de crominancia se escalan a la mitad, tanto en la dirección horizontal como en la dirección vertical.

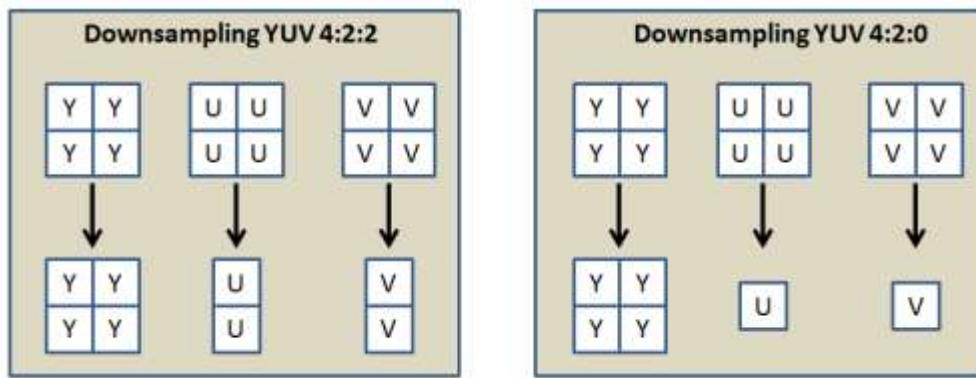


Figura 69. Escalados en crominancia

Las imágenes constituidas por las señales de crominancia U, V son menos abruptas, más suavizadas que las de luminancia, por lo que la proporción de hops pequeños es aún mayor y en consecuencia la compresión de estas componentes se ve beneficiada.

## 3.7 Resultados

### 3.7.1 Calidad vs bpp

En las siguientes tablas se muestran ejemplos de las conocidas imágenes de test “Lena”, “baboon” y “peppers” de la galería “misclánea” de la Universidad de California [63]. La limitación más importante de LHE básico es la imposibilidad de escoger los bpp a los que deseamos comprimir. El ratio de compresión obtenido depende de la imagen y normalmente va a encontrarse en el intervalo 1:3 a 1:5

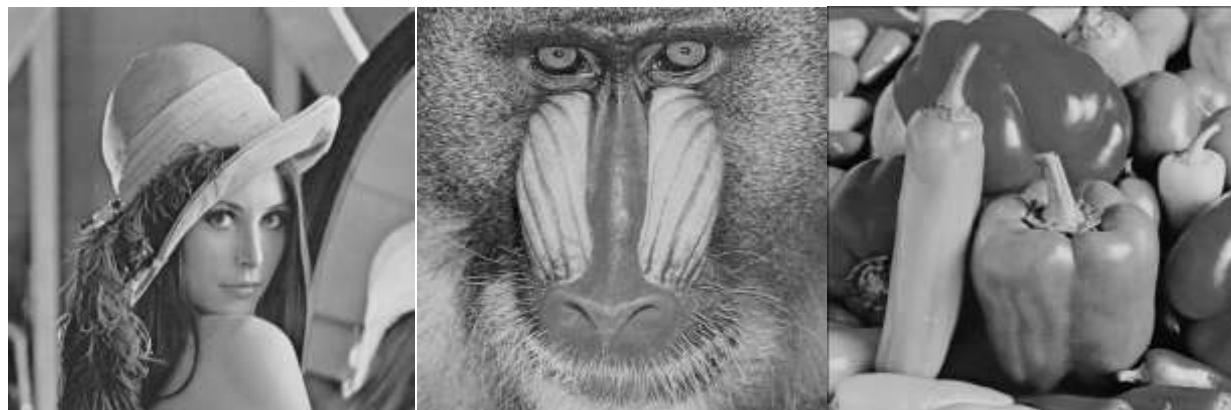


Figura 70. Imágenes "lena", "baboon" y "peppers"

En el caso de "Lena" se muestran dos cálculos de LHE. Aunque generalmente el mejor resultado se obtiene con  $h1max=10$ , en "Lena" lo conseguimos con  $h1max=8$ .

El PSNR es tan elevado que ilustrar el resultado con una imagen no aportaría nada ya que la imagen original y la codificada con LHE son indistinguibles. En su lugar se muestra la comparativa con JPEG y JPEG2000 tomando como referencia los mismos bpp obtenidos.

Compressor	bpp	PSNR
basicLHE ( $h1max=10$ )	1.9	42.11
basicLHE ( $h1max=8$ )	1.97	43.35
JPG	1.9	40.3
JPEG2000	1.9	44.3

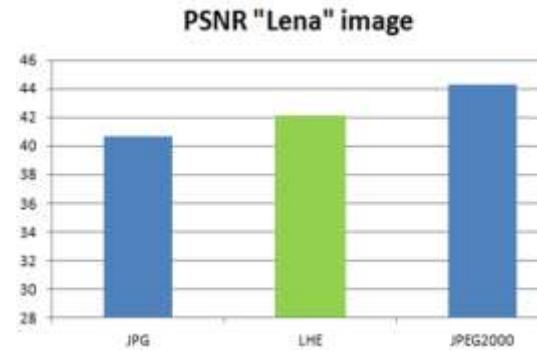


Tabla 8. Valores de calidad de LHE básico para imagen "lena"

Compressor	bpp	PSNR
basicLHE ( $h1max=10$ )	2.8	34
JPG	2.8	32,3
JPEG2000	2.8	37,2

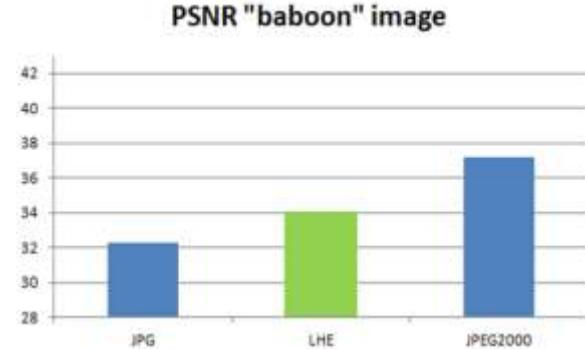


Tabla 9 Valores de calidad de LHE básico para imagen "baboon"

Compressor	bpp	PSNR
basicLHE (h1max=10)	2.1	39
JPG	2.1	37,4
JPEG2000	2.1	41,6

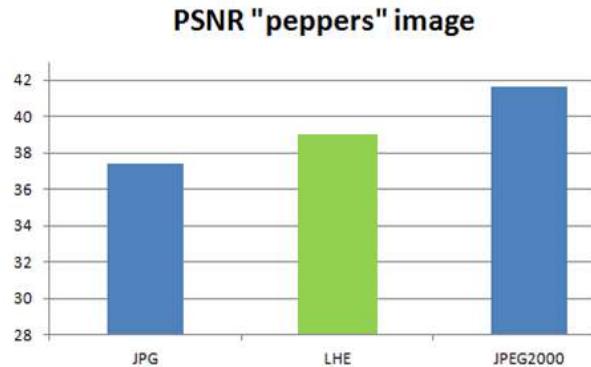


Tabla 10. Valores de calidad de LHE básico para imagen “peppers”

LHE básico funciona con un valor fijo de  $h1max$ , pero LHE avanzado es capaz de ajustar mejor el valor de  $h1max$  en función de la zona de la imagen en la que se encuentra cuantificando. En el próximo capítulo de esta tesis se tratará este punto.

### 3.7.2 Tiempos de ejecución lineales

A modo ilustrativo, el efecto de la complejidad en el rendimiento de un algoritmo se ilustra en la siguiente gráfica, para la que se ha considerado de forma genérica que una operación consume un milisegundo. Como se aprecia, un algoritmo de  $O(n)$  es mucho más rápido que  $O(n\log n)$  para valores de  $n$  altos.

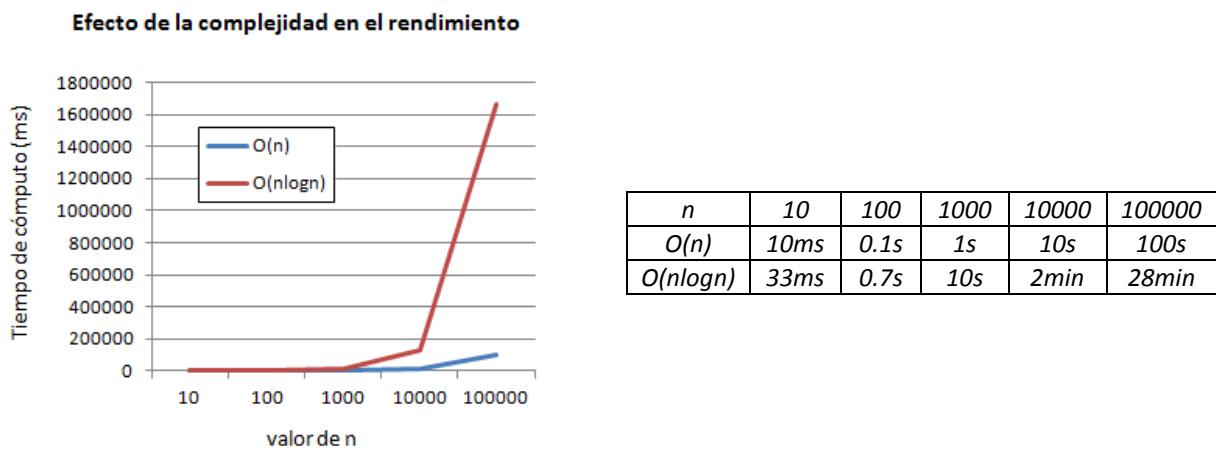


Figura 71. Efecto de la complejidad en el rendimiento

El prototipo Java desarrollado (sin paralelización) testado en Intel i5-3320@2.6Ghz ejecuta la cuantización LHE en los tiempos que se muestran en la siguiente tabla, confirmando la complejidad lineal  $O(n)$  de LHE. Los algoritmos en los que se basan JPEG y JPEG2000 poseen una complejidad superior  $O(n \cdot \log n)$ , por lo que LHE supone una ventaja computacional frente al estado del arte.

Image resolution (px)	184k(429x429)	414k (720x576)	921k(1280x720)	2073k(1920x1080)
Quantization time (ms)	2.8	6.7	15	29.49

Tabla 11. Tiempos de ejecución del cuantizador LHE

### 3.8 Resumen del capítulo

En este capítulo se ha presentado el codificador LHE básico, que sirve de base para el codificador LHE avanzado.

El fundamento de LHE básico es el modelado del ojo humano, en tres aspectos: la respuesta logarítmica al estímulo, la sensibilidad mínima y el proceso de acomodación al brillo medio local.

La idea fundamental de recrear el comportamiento fisiológico del ojo mediante un modelado de complejidad lineal es la base de LHE y supone una de las contribuciones principales de esta tesis

Para modelar el ojo se han creado nuevos algoritmos y estrategias que son contribuciones de esta tesis, todas ellas englobables en lo que he denominado “algoritmo LHE básico”

- La estimación o predicción del color de fondo
- La codificación por saltos logarítmicos y el cálculo de su razón geométrica
- El algoritmo de selección del mejor hop
- La elección de 9 hops para cuantizar
- La asignación del valor medio del intervalo cubierto por cada hop como valor de reconstrucción de la señal
- La definición del valor mínimo del primer hop basada en la fracción de Weber
- El algoritmo de adaptación del primer hop con el que se modela la acomodación al brillo medio
- La elección del valor máximo del primer hop
- El algoritmo del codificador binario de hops, que aprovecha redundancias espaciales y distribución estadística de los hops

El algoritmo LHE básico ofrece resultados de calidad intermedios entre JPEG y JPEG2000, aunque tiene una limitación importante: la imposibilidad de escoger los bpp a los que deseamos comprimir. El ratio de compresión obtenido depende de la imagen y normalmente va a encontrarse en el intervalo 1:3 a 1:5

La complejidad de LHE es lineal. Los algoritmos en los que se basan JPEG y JPEG2000 poseen una complejidad superior  $n \cdot \log(n)$ , por lo que LHE supone una ventaja computacional frente al estado del arte. Una de las razones de su baja complejidad es que opera en el dominio del espacio, a diferencia de otros algoritmos que operan en el dominio de la frecuencia. Esto evita la conversión de dominios y por lo tanto simplifica el proceso de codificación.

La capacidad de paralelización de LHE permite reducir los tiempos de ejecución secuenciales en una proporción N:3, siendo N la longitud en píxeles del lado de la imagen.

Puesto que LHE utiliza el espacio de color YUV para representar la información, la aplicación de estrategias de downsampling de las señales de crominancia YUV 4:2:0 y YUV 4:2:2 son válidas en LHE para generar imágenes en color.

# Capítulo 4

## LHE avanzado

### 4.1 Presentación del capítulo

En el capítulo anterior se ha presentado el algoritmo LHE básico, cuya limitación fundamental es la imposibilidad de escoger el bit-rate (los bpp) del resultado deseado, es decir, la imposibilidad de escoger entre una mayor o menor compresión.

En este capítulo se presenta el algoritmo del codificador LHE avanzado, que permite escoger el grado de compresión deseado. En muchas de las figuras se adelantarán resultados para los que es necesario decodificar la imagen comprimida, lo cual requiere de algoritmos que se expondrán en el capítulo dedicado al decodificador. Sin embargo son necesarios para ilustrar y argumentar muchas de las decisiones tomadas en el diseño del codificador.

En LHE avanzado se permite escoger el grado de compresión deseado. Para ello primeramente se divide la imagen en una malla de bloques, se ejecuta el algoritmo de cuantización LHE básico en cada bloque y se mide la importancia de la información contenida en cada bloque analizando en los hops resultantes. A continuación se sub-muestrea en mayor o menor medida cada bloque, en función de la importancia de la información que contiene y por último se cuantiza los pixels de cada bloque sub-muestreado mediante un segundo LHE. Esto va a permitir reducir la cantidad de hops a almacenar, ya que los bloques menos sub-muestreados contienen menos píxeles y por lo tanto menos hops.

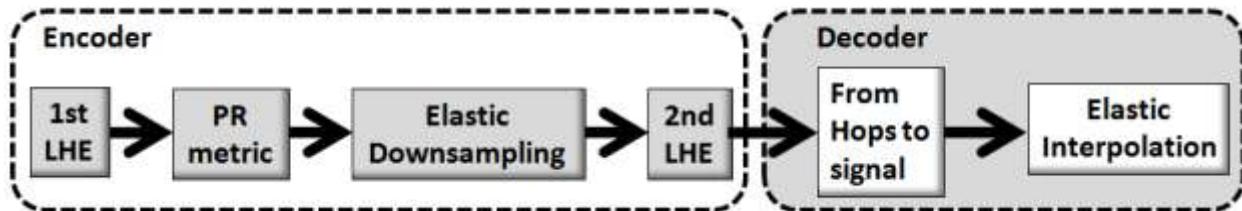


Figura 72. LHE avanzado

La forma de acometer el algoritmo LHE avanzado implica la creación de nuevos conceptos y algoritmos que permitan evaluar la información representada en los hops y escalar de forma “elástica” cada bloque, permitiendo que cada esquina posea un nivel de sub-muestreo diferente y por lo tanto preservando más información donde más se necesite.

El siguiente diagrama de bloques funcional resume todo el proceso en detalle y es el objeto del presente capítulo. A medida que se profundice en cada bloque, este diagrama será mostrado de nuevo señalando en qué bloque se centra la explicación.

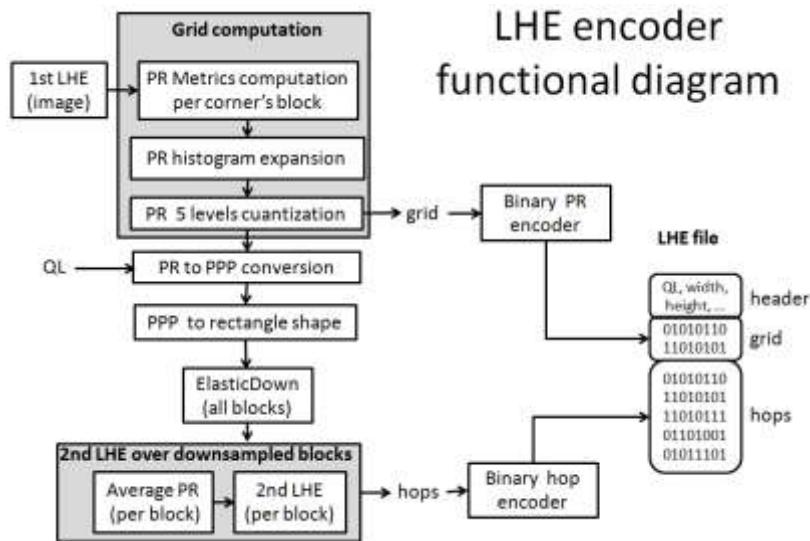


Figura 73 diagrama de bloques del codificador LHE avanzado

## 4.2 Importancia de la estrategia de downsampling

En el algoritmo LHE, la estrategia de downsampling tiene una relevancia de primer orden. La calidad final de la imagen va a depender directamente de la capacidad de conseguir sub-muestrear la imagen dañando lo menos posible la calidad, para interpolar con la máxima fidelidad posible.

La calidad que pierda la imagen cuando es reducida, es una calidad irrecuperable. Cada decibelio que la estrategia de downsampling consiga preservar es de vital importancia en el resultado final.

En este capítulo se mostrará la comparación entre la calidad perdida con un downsampling homogéneo y el Downsampling Elástico, el cual constituye una contribución de esta tesis que permite mejorar notablemente la calidad con el mismo número de pixeles preservados.

### 4.2.1 Primera estrategia de downsampling creada y limitaciones

Como parte de esta tesis, se han creado dos estrategias de downsampling, la primera de las cuales ha sido reemplazada por una segunda más avanzada y a la que me referiré como “downsampling elástico”.

El mecanismo inicial definido y publicado en [64] definido como downsampling para LHE se basaba en escalar un bloque en una o varias direcciones dependiendo de la cantidad de información presente en el bloque, el cual se calculaba mediante unas métricas a efectuar sobre los hops constituyentes de dicho bloque. No detallaré dichas métricas por estar incluidas en el artículo [64] y por haber sido mejoradas para la segunda estrategia, aun siendo una contribución de esta tesis. El resultado de las métricas define lo suave o abrupto que es un bloque y con ello se puede definir un umbral que dicha métrica debe vencer para que un bloque sea escalado

Para comprimir más o menos una imagen basta con modificar el umbral para las métricas anteriormente mencionadas. El resultado del downsampling de un bloque debe resultar en un cuadrado de 4x4 pixeles, un rectángulo de 8x4, un rectángulo de 4x8, o bien ningún downsampling si no se vence el umbral de la métrica de los hops contenidos en el bloque.

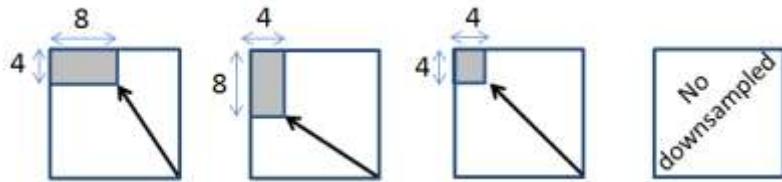


Figura 74. Cuatro tipos de downsampling, incluyendo el “no downsampled”

Este tipo de compresión se efectúa en todos los bloques en los que se divide la imagen inicialmente, que son bloques de 32x32 píxeles (Figura 75, etiqueta “1”). En aquellos bloques donde no se ha efectuado ningún downsampling por no vencer el umbral, se subdivide en 4 sub-bloques de 16x16 píxeles y se vuelve a intentar el proceso teniendo en cuenta esta vez los resultados de las métricas sobre hops de estos bloques más pequeños (Figura 75, etiqueta 2). El proceso se repite una vez más sobre los bloques que queden sin escalar, usando una última medida de sub-bloque de 8x8 (Figura 75, etiqueta 4).

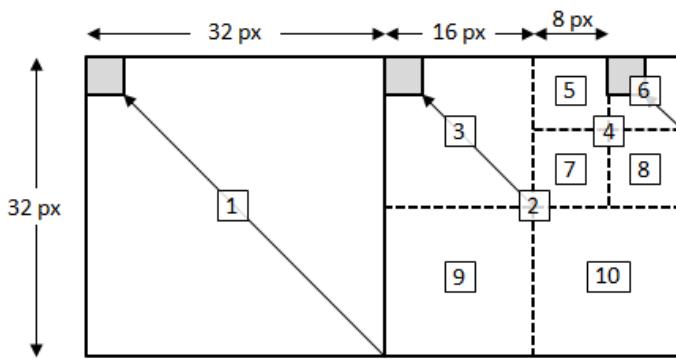


Figura 75 Ejemplo de estrategia de downsampling usando tres niveles de malla

Finalmente, además de almacenar los hops de los bloques escalados o no escalados, se debe almacenar la información de malla, donde se han definido 4 tipos de escalado por bloque (2 bits por bloque). La estrategia así definida permite muchos niveles de escalado modificando el umbral que escogemos para las métricas de hops.

Una buena estrategia de downsampling debe permitir bajar los bpp de forma continua, y así lograr también una curva “PSNR vs bpp” continua, para de este modo poder ofrecer la posibilidad de elegir cualquier intensidad de la compresión de forma continua.

El problema o limitación de esta técnica es que cuando se vence el umbral escogido en la métrica definida sobre los hops, se produce un “salto” en la cantidad de píxeles que desaparecen y por lo tanto se produce un salto en el PSNR por el mismo motivo, perdiendo la continuidad de la función de PSNR vs bpp.

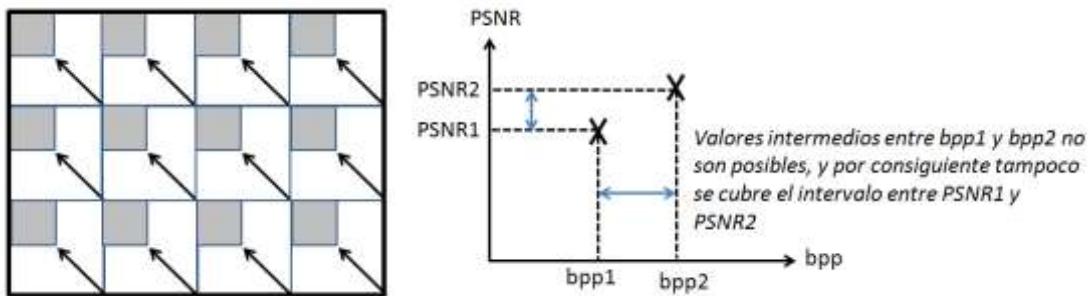


Figura 76 Una imagen cuyos bloques poseen las mismas métricas

El problema descrito se agrava cuando la imagen se parece mucho a sí misma. Si todos los bloques de los que está compuesta una imagen son idénticos, los valores de las métricas efectuadas con sus hops serán idénticos, y por consiguiente vencerán el umbral todos los bloques a la vez. En ese caso la imagen comprimida pasará de  $x$  bpp a  $(x - \Delta x)$  bpp, siendo  $\Delta x$  un gran salto en bpp que condiciona a su vez un gran salto en PSNR. En el ejemplo (Figura 76), suponiendo que todos los bloques sean idénticos, al vencer el umbral de escalado lo hacen todos a la vez, creándose una imagen sub-muestreada de  $\frac{1}{4}$  del tamaño original, por lo que se produce una gran discontinuidad en el downsampling: se pasa de no sub-muestrear nada a sub-muestrear a  $\frac{1}{4}$  de la imagen original. Ello conlleva un salto en PSNR y en bpp.

Puesto que la estrategia anterior se basaba en un conjunto de 3 mallas, en una imagen idéntica a sí misma, primeramente se vencerá el umbral de la malla 0, luego el de la malla 1 y finalmente el de la malla 2, teniendo por lo tanto solo 3 niveles de compresión. En este caso la función bpp se vuelve totalmente discontinua y por lo tanto también la función PSNR, que podrá tomar tan sólo 3 valores posibles.

Las métricas sobre hops para determinar la cantidad de información presente en un bloque, así como el algoritmo de downsampling basado en 4 tipos de escalado y 3 mallas son contribuciones de esta tesis y han sido publicadas en [64]. No obstante, gracias a esa investigación y a los resultados obtenidos se han observado limitaciones que han permitido sentar las bases para la creación de una nueva técnica de downsampling “elástica” que se detallará a continuación.

#### 4.2.2 Introducción a la segunda estrategia de downsampling

La segunda estrategia de downsampling creada en esta tesis se ha denominado “Downsampling Elástico”. Se fundamenta en una medida de la “relevancia perceptual” (la cual será definida más adelante) que permite saber, para cada bloque de una rejilla simple en la que se divide la imagen, la cantidad de información que se debe preservar en cada una de sus esquinas. De este modo, para un mismo bloque se tomarán más muestras en la zona donde haya más información relevante para el ojo humano.

El “Downsampling Elástico”, por lo tanto, básicamente consiste en sub-muestrear de un modo no homogéneo. Es decir, se conservan más pixels de unas áreas y menos de otras.

El número de pixels originales que representa un pixel sub-muestreado lo llamaremos **“PPP” (Pixels Per Pixel)**. Este número no puede ser menor de 1. En un sub-muestreado homogéneo todos los pixels sub-muestreados representan el mismo número de pixels originales, pero en el Downsample Elástico no es así. Se asignan diferentes PPP en cada esquina en ambas coordenadas (X e Y) y se evoluciona el número de PPP linealmente entre esquinas, quedando una imagen resultante deformada respecto de la original.

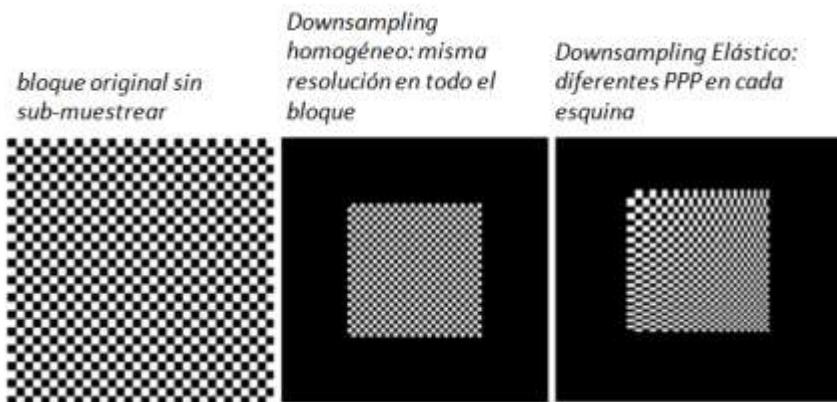


Figura 77. Downsampling homogéneo vs Downsampling Elástico

En la figura representada, se ha optado por asignar más resolución (menos PPP) a la esquina superior izquierda del “tablero de ajedrez”. Se puede apreciar que el primer cuadrado conserva el tamaño original (PPP=1), mientras que el cuadrado inferior derecho es notablemente más pequeño (PPP mayor).

Para llevar a cabo esta técnica, la imagen original se divide en bloques y en función de la medida de Relevancia Perceptual calculada en cada vértice del bloque, se realiza un downsampling no homogéneo. Es decir, se conservan más pixels en las regiones con mayor información relevante. Ello permite superar la calidad que tendría una imagen sub-muestreada al mismo número de pixeles de forma homogénea, ya que el Downsampling Elástico asigna más pixels en las zonas donde hay más información relevante y al interpolar la imagen, el PSNR de la imagen resultante será más alto.

En la siguiente imagen aparece un ejemplo de Downsampling Elástico, en el que la imagen completa de Lena se ha escalado a diferentes PPP en cada esquina. La zona derecha de la imagen ha quedado con mayor resolución que la zona izquierda, más “apretada”. Al reconstruir por interpolación esta imagen, previsiblemente la zona derecha se podrá reconstruir con mayor calidad, en perjuicio de la parte izquierda que se reconstruirá con peor calidad.



(Aclaración: En este caso se ha asignado más resolución a la parte derecha tan solo como ejemplo, no porque esa zona posea más información relevante).

Figura 78. La imagen de Lena, sub-muestreada elásticamente

El concepto “Downsampling Elástico” es una contribución de la presente tesis. No existe nada parecido en el estado del arte, y los artículos sobre downsampling e interpolación jamás consideran esta posibilidad, lo cual es lógico, pues no disponen de un mecanismo de medida de la relevancia perceptual para asignar diferentes PPP, el cual es precisamente parte del algoritmo LHE avanzado.

La creación del “Downsampling Elástico” conlleva, como se explicará más adelante, la creación de la “interpolación elástica”. Los actuales métodos de interpolación en el estado del arte [65] consideran un sub-muestreado homogéneo, pero no elástico.

#### 4.2.3 Creación de la malla para el Downsample Elástico

El procedimiento de Sub-muestreado Elástico (o “Downsampling Elástico”) va a realizarse sobre una malla de bloques fija. Es decir, tendremos el mismo número de bloques independientemente de la resolución de la imagen. Si la imagen posee más resolución, entonces los bloques tendrán más pixels, pero no habrá más bloques. La única excepción es que en imágenes muy pequeñas fijaremos un tamaño mínimo para los bloques (que al menos tengan un lado de 8 pixels).

Hay otros algoritmos que fijan el tamaño del bloque en pixels, como JPEG, de modo que en función del tamaño de la imagen necesitan más o menos bloques. En Downsample Elástico, gracias al muestreo selectivo (muestreamos más donde hay más información), podemos hacer uso de una malla constituida por un número de bloques fijo y comprimir en muy pocas muestras algunos bloques y usar muchas muestras en otros.

Un bloque será sub-muestreado en al menos 4 muestras, debido a que vamos a poder tener valores de PPP diferentes en cada esquina. Quedaría definir cuántos bloques va a tener la malla.

Los criterios para elegir el número de bloques de la malla son:

- Cada bloque debe tener suficiente variación de información en su interior para que tenga sentido muestrear a diferentes PPP en cada una de sus esquinas, por lo que no pueden ser muy pequeños.
- Un bloque muestreado elásticamente posee diferentes PPP en sus esquinas pero no puede tener un máximo o mínimo a mitad de camino, es decir, el numero de muestras crece o decrece a lo largo de una arista pero no puede crecer y luego decrecer. Por eso si el bloque es grande, y contiene algo detallado en el centro, tendremos que asignar muchas muestras en todo el bloque ya que no podemos asignar muchas muestras solo en el centro. Este es un motivo para que los bloques no sean demasiado grandes. Es decir, no pueden ser bloques enormes pues entonces no habrá bloques sin apenas información que nos permitan ahorrar muchas muestras
- Interesa que haya pocos bloques para que la información de los PPP de las esquinas no ocupe demasiada información. Es decir, en este sentido, cuanto más grandes sean mejor, pues habrá menos bloques y por tanto habrá menos información que almacenar.

Con estos criterios se ha escogido el número 32 como el número de bloques que va a tener la malla en el lado mayor de la imagen (normalmente el horizontal). El lado menor tendrá los bloques necesarios para que tengan forma cuadrada. Este número es un parámetro configurable del algoritmo, aunque 32

ha dado buenos resultados en general. En la siguiente figura se muestra el resultado de ensayar diferentes números de bloques para la imagen Lena usando el algoritmo LHE avanzado que se desarrollará a lo largo de este capítulo.

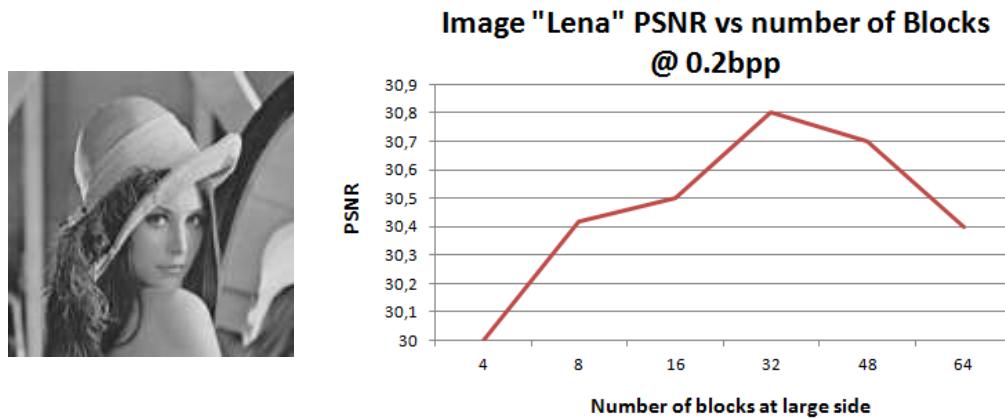


Figura 79. Diferentes resultados con Lena usando diferente número de bloques en la malla

El efecto de disminución de PSNR muchos bloques más pequeños ( 64 bloques en Figura 79) responde al hecho de que cada bloque es sub-muestreado por separado en al menos 4 muestras (una por cada esquina para poder tener diferente valor de PPP en cada una), de modo que cuando hay pocos bloques muy grandes, un gran área lisa (sin apenas información) ocupa pocos bloques y por lo tanto pocas muestras (4 por bloque), mientras que con muchos bloques pequeños, las áreas grandes sin apenas información ocupan muchos bloques y por lo tanto más muestras. Si dedicamos muchas muestras a zonas sin apenas información, el ratio de compresión para una misma calidad es peor. En un caso extremo en el que los bloques fuesen tan pequeños que solo contuviesen 4 pixeles en su interior, la posible compresión sería nula, ya que cada bloque al menos va a sub-muestrearse a 4 muestras.

El número máximo de PPP (al que llamaré  $PPP_{max}$ ) es dependiente de la resolución de la imagen, ya que si la compresión máxima ocurre cuando un bloque es sub-muestreado a 4 muestras, entonces la ecuación de  $PPP_{max}$  queda como se muestra a continuación:

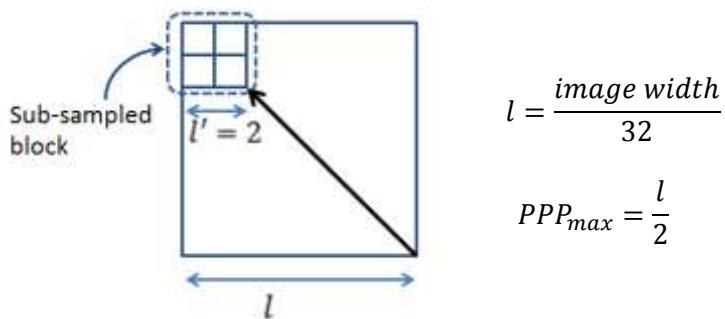


Figura 80. MÁximo PPP y dependencia con el tamaño de la imagen

En la siguiente imagen (Figura 81) se aprecia como muchos bloques contienen bordes, rocas, elementos de las casas que hay que muestrear más intensamente y otros bloques más “lisos” que contienen cielo o paredes donde se pueden ahorrar muestras. Existen bloques que contienen parte de cielo y parte del borde de un objeto. En esos casos el Downsampling Elástico asignará más muestras a la zona del bloque

que contiene el borde. En este caso, y en general, 32 bloques de ancho resultan adecuados. No obstante este número no es el número óptimo para cualquier imagen, sino que es un número que experimentalmente resulta adecuado para las bases de datos de imágenes con las que se ha trabajado y queda abierto como un parámetro de configuración del algoritmo.

Como se aprecia en la gráfica, en este caso 48 bloques producen el mismo resultado que 32, por lo que es más interesante 32, ya que son menos bloques y por tanto menos información de malla. Este mismo tipo de resultado se ha obtenido con todas las imágenes de la galería Kodak [61].

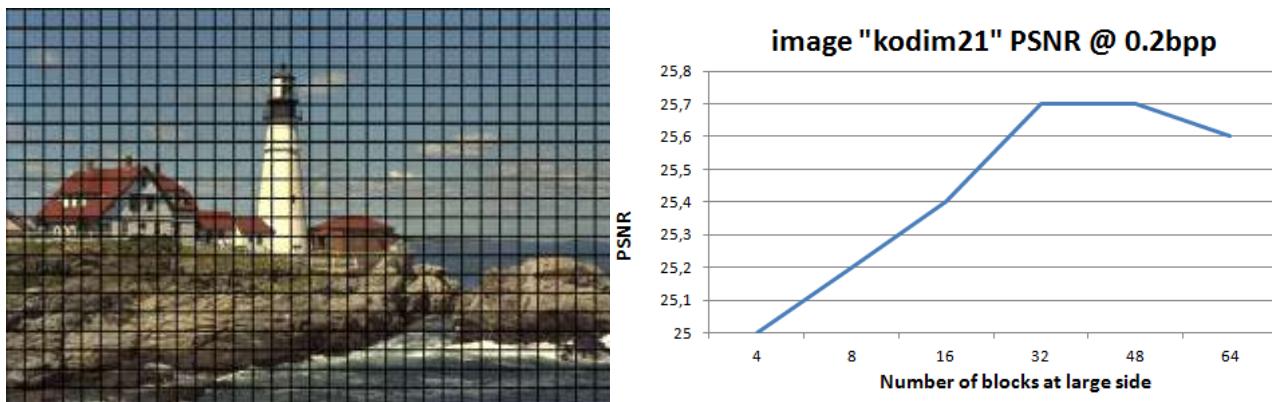


Figura 81. Diferentes resultados con Kodim21 usando diferente número de bloques en la malla

Una vez más, cuando escogemos muchos bloques (64) empieza a disminuir el PSNR, debido a que gastamos muestras en zonas lisas y quedan por lo tanto menos muestras disponibles para las zonas que necesitan más detalle.

La malla con número de bloques independiente de la resolución de la imagen es una contribución de la presente tesis. Cada bloque debe tener suficiente variación de información en su interior para que tenga sentido muestrear a diferentes PPP en cada una de sus esquinas, por lo que no pueden ser muy pequeños. Pero no pueden ser bloques enormes pues entonces no habrá bloques sin apenas información que nos permitan ahorrar muestras. Por otro lado, cuanto menor sea el número de bloques, menos costoso en almacenamiento será la información de los valores de PPP de las esquinas de esta malla. El valor experimental que ha resultado óptimo en la mayoría de los casos, tanto por calidad como por eficiencia en gasto de información ha sido 32 bloques en el lado mayor de la imagen.

### 4.3 Relevancia Perceptual: concepto y métrica

Para ubicar el contenido de este apartado, a continuación se muestra el diagrama de bloques de LHE avanzado, destacando el módulo en el que se ubica el contenido que se va a desarrollar.

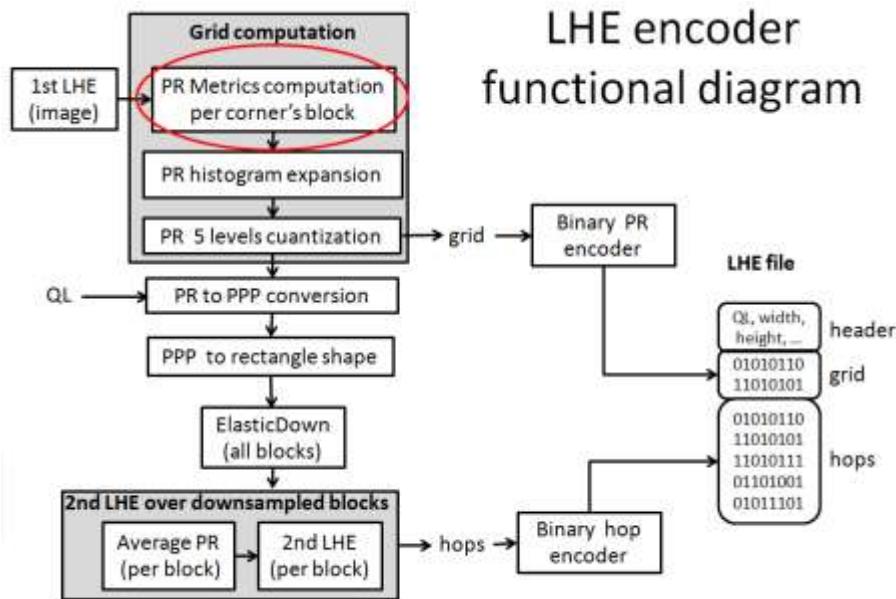


Figura 82. Ubicación del contenido que se va a desarrollar

Previo a la definición de la métrica de Relevancia Perceptual (me referiré a ella como “PR” *Perceptual Relevance*) se debe considerar la introducción de otros conceptos sobre los que se apoya. Una vez presentados estos conceptos y definida la métrica de PR, se mostrará el modo de usarla en la nueva técnica de downsampling, llamada “Downsampling Elástico”

#### 4.3.1 Sensación y percepción

La sensación o información sensorial [66] es la respuesta autónoma y aislada de cada uno de nuestros sentidos a su estimulación. Esta respuesta es logarítmica tal como establece la ley de weber y carece de interpretación, es la mente quien construye las percepciones a partir de dicha información sensorial [67]

Por ejemplo en el caso de la visión, la sensación o información sensorial sería la respuesta logarítmica al estímulo luminoso. Y la percepción sería una figura, o una textura. La percepción introduce la interpretación de la sensación por parte del cerebro [68]. Es pues, algo subjetivo, un fenómeno psíquico, donde el valor de la percepción no depende solo del estímulo, sino de la interpretación que hace el sujeto de la sensación provocada por dicho estímulo [69].

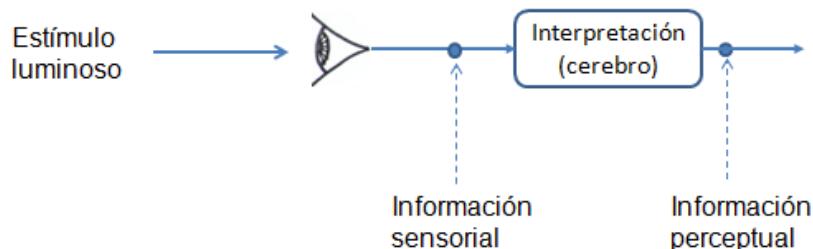


Figura 83 Información sensorial y perceptual

Lo que para el individuo tiene sentido no es la información sensorial (la respuesta logarítmica del ojo) sino la interpretación que hace su mente de dicha información, es decir, la información perceptual, la cual es la traducción a figuras y/o texturas de la información sensorial.

### 4.3.2 Propósito de la Métrica de relevancia perceptual

La medida de la relevancia perceptual va a permitir tener un criterio para sub-muestrear los bloques (en los que se dividirá una imagen) en función de la cantidad de información perceptual que la mente extraiga de ellos. Esto significa que la métrica hace posible una herramienta para reducir la información original de una imagen minimizando la degradación de la percepción por parte del sujeto.

Resumiendo lo que se va a ver en los siguientes apartados (realmente hay mas pasos pero estos son los más relevantes):

- La imagen es dividida en bloques, mediante una malla única, sin niveles jerárquicos, con todos los bloques de igual tamaño (siendo 32 el número fijo de bloques en el lado mayor).
- Se Calcula la métrica de PR en cada esquina de cada bloque en sus dos dimensiones espaciales y después se asigna más o menos pixels originales por cada pixel sub-muestreado. Es decir, más o menos “**PPP**” (**Pixels Per Pixel**). Este número no puede ser menor de 1, lógicamente. El PPP que se asigna en cada esquina dependerá del valor de PR en esa esquina y de un factor que indicará la intensidad de compresión que deseemos.
- Posteriormente se realiza un sub-muestreo no homogéneo, al que llamaré “elástico”, interpolando linealmente los PPP entre esquinas, asignando más pixels sub-muestreados (menor PPP) en las esquinas de mayor PR. Con este proceso cada bloque será transformado en una versión reducida del mismo, degradando lo menos posible aquellas zonas donde nuestra métrica de relevancia perceptual sea mayor.
- Por último cuantizaremos con LHE los bloques sub-muestreados.

En las próximas secciones de este capítulo, se deducirá la fórmula de la relevancia perceptual, así como la fórmula que permite hacer la conversión entre PR y PPP y posteriormente se abordará paso a paso la forma de llevar a cabo el proceso de downsampling e interpolación.

El enfoque en el que se basa la estrategia de downsampling, considerando una métrica de relevancia perceptual, así como una malla única y el DownSampling Elástico son contribuciones de esta tesis y forman parte del algoritmo LHE avanzado.

### 4.3.3 Métrica de relevancia perceptual

La información sensorial es la respuesta logarítmica al estímulo y por lo tanto se puede equiparar a la transformación a saltos logarítmicos (Hops) que realiza LHE partiendo de una imagen en bruto. Los hops tal como se han visto en el capítulo de LHE básico, son un modelado de la respuesta del ojo humano.

La mente debe analizar esta información sensorial para extraer la información perceptual. El significado que la mente otorga a una escena puede ser muy complejo, ya que la información perceptual está jerarquizada. El campo de la psicología de la percepción es muy amplio y existen muchos estudios al respecto, como son las “leyes de la Gestalt”. Son un conjunto de leyes que nos revelan como la mente interpreta las formas presentes en una imagen [70] [69] [71].

A pesar de que el proceso de interpretación que hace la mente puede ser muy complejo, podemos medir el “primer nivel” de interpretación que puede hacer la mente a partir de la información sensorial, es decir, en lugar de tratar de averiguar la interpretación final que hace la mente de una figura (es un coche, es una casa,...), se tratará de medir si lo que hay en una imagen o un trozo de una imagen carece

de información perceptual, es percibido como figuras o partes de figuras o incluso puede llegar a ser percibido como ruido.

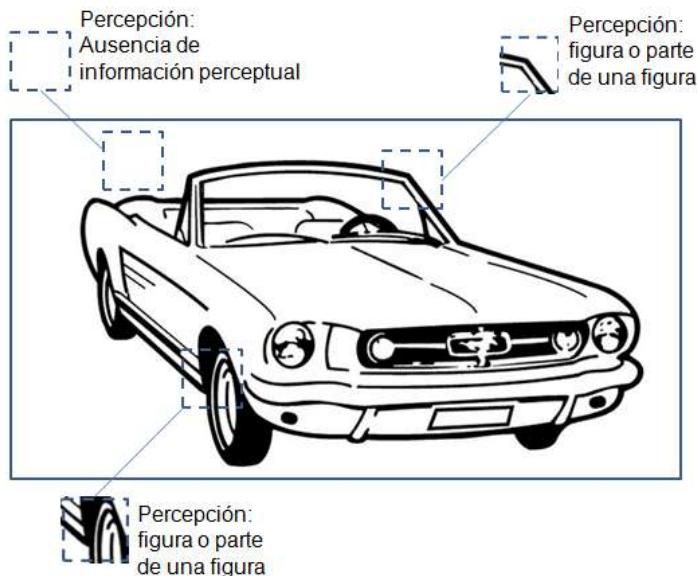
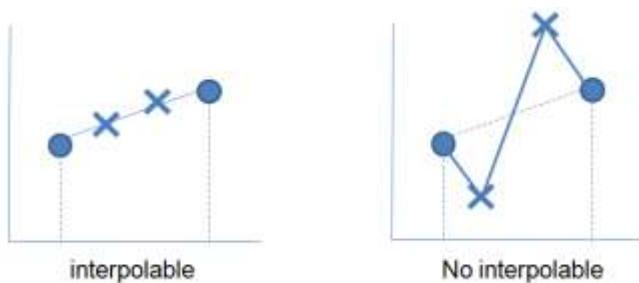


Figura 84. Distintas percepciones en fragmentos de una imagen

Una imagen va a contener mayor información perceptual **cuanto menos interpolable sea**. Un cuadrado blanco podríamos regenerarlo a partir de sus cuatro esquinas, pero no es posible hacerlo con la imagen de un coche. Para regenerar el coche por interpolación necesitamos más muestras. Cuantas más figuras detalladas haya en la escena, mas muestras necesitaremos para regenerarla por interpolación, y la mente tendrá mucha más información sensorial que interpretar.

En definitiva, la relevancia perceptual (PR) de una cierta información sensorial es inversamente proporcional a su interpolabilidad.



En estos dos ejemplos se trata de interpolar dos valores intermedios de la señal a partir de dos muestras ubicadas en los extremos. Los cambios de signos impiden la correcta interpolación, así como la falta de linealidad.

Figura 85. Interpolabilidad de la señal

Para medir la interpolabilidad de una información sensorial, o lo que es lo mismo, de los hops logarítmicos derivados de una imagen estímulo original, se debe analizar las dos coordenadas por separado ( $PR_x$  y  $PR_y$ ) y en cada coordenada se debe analizar cuantos hops son imposibles de interpolar. Esto nos lo va a revelar los cambios de signo de los hops. El tamaño del hop nos dará una medida del error cometido si dicha información fuese interpolada en lugar de ser preservada. Por lo tanto debemos sumar aquellos hops que rompen la tendencia ascendente o descendente de la secuencia de hops y con ello tendremos una medida de la no interpolabilidad. Cuanto menor sea la interpolabilidad, mayor resultará el valor de esta métrica.

Tan poco interpolable es un bloque que contiene un solo borde como otro bloque que contenga tres bordes. Lo que define su interpolabilidad es lo abrupto que sea el borde y no cuantos bordes haya en una porción de la imagen, ya que para reproducir la misma percepción mediante muestreo e interpolación necesitaremos una cadencia de muestreo equivalente en ambos casos.

Un bloque puede contener más información sensorial (3 bordes) pero no por ello es menos interpolable que otro que contiene un solo borde. Lo que importa de cara a la percepción es si es un borde suave o abrupto, pero no si hay tres bordes o solo uno.

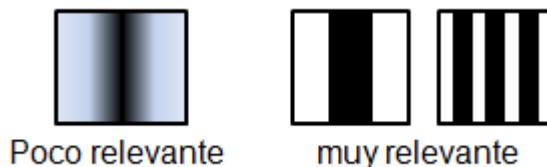
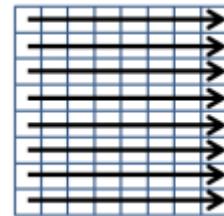


Figura 86. Relevancia es independiente del número de bordes

Por lo tanto para definir completamente la métrica de relevancia perceptual se divide por el numero de hops que cambian de signo ( $C_x$ ) multiplicado por el tamaño máximo del hop ( $H_{max}$ ). En caso de ser  $C_x = 0$ , simplemente asignaremos  $PR_x = 0$

$$PR_x = \frac{\sum_{i=1}^{C_x} |h_i|}{C_x \cdot |H_{max}|}$$

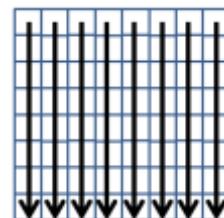
Se contabilizan en el numerador la magnitud de los hops que producen cambios de signo o que tomen el valor  $H_{max}$ , recorriendo todo el bloque mediante scanlines horizontales. En el denominador se multiplica el numero de hops considerados en el numerador multiplicado por el máximo orden posible de hop ( $H_{max}=4$ ).



Ecuación 15. Ecuación de la métrica de relevancia perceptual PRx

$$PR_y = \frac{\sum_{i=1}^{C_y} |h_i|}{C_y \cdot |H_{max}|}$$

Se contabilizan en el numerador la magnitud de los hops que producen cambios de signo o que tomen el valor  $H_{max}$ , recorriendo todo el bloque mediante scanlines verticales. En el denominador se multiplica el numero de hops considerados en el numerador multiplicado por el máximo orden posible de hop ( $H_{max}=4$ ).



Ecuación 16. Ecuación de la métrica de relevancia perceptual PRy

El sumatorio que aparece en el numerador de ambas ecuaciones contempla los hops que cambian de signo respecto del hop anterior en cada scanline. Esta suma de hops se hace en valor absoluto, siendo el valor  $h_i$  comprendido entre 1 y 4 (donde 1 es el salto más pequeño y 4 es el salto mayor  $H_{max} = 4$ ).

$C_x$ ,  $C_y$  son el número de hops que han cambiado de signo evaluados en cada dirección

$H_{max}$  es el hop de mayor tamaño.  $H_{max} = 4$

Pasemos a considerar algunos ejemplos para ilustrar los resultados de la métrica:

- Un borde muy suave con  $PR=0.25$
- Un borde abrupto con  $PR=1$

- Ruido, con PR elevado pero menor que 1
- Un borde con PR=0.5.

En todos los ejemplos vamos a considerar el hop máximo  $H_{max} = 4$

#### Un borde muy suave: PR = 1/4 =0.25

Esta imagen ha sido generada con saltos de luminancia de 20 en 20. Son saltos para los que normalmente va a hacer falta un hop no nulo. La métrica resultante es PRx=0.25 y PRy=0

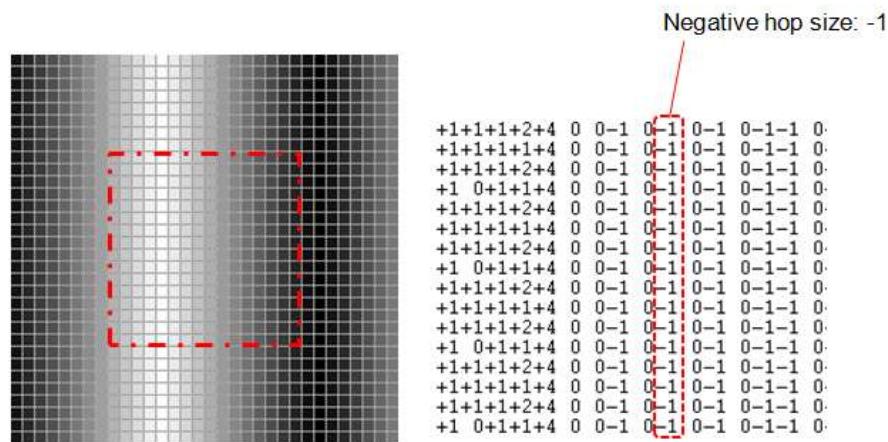


Figura 87. Ejemplo de cálculo de PRx en un bloque que contiene un borde suave

El bloque cuya PR ha sido calculada es el bloque interior. Sus dimensiones son 16x16

Se observa como en vertical no hay cambios de signo. En horizontal solo hay un cambio de signo por scanline. Los saltos a hop nulo (nulo=0) no cuentan, de modo que por cada scanline solo hay un cambio de signo, el cual se corresponde con un salto pequeño de peso 1

El numerador será num\_scanlines x 1 = 16 x 1 = 16

El denominador será: num\_scanlines x 4 =16x4 =64

Un bloque puede tener una métrica aun menor, si llega a ser una gradación o un color liso tan suave que la PR puede llegar a ser cero o cercana al cero

#### Un solo borde abrupto: PR=1

En este ejemplo se muestran dos bordes en el mismo bloque. Una vez mas PRy=0 debido a que no presenta cambios de signo, pero PRx es 1

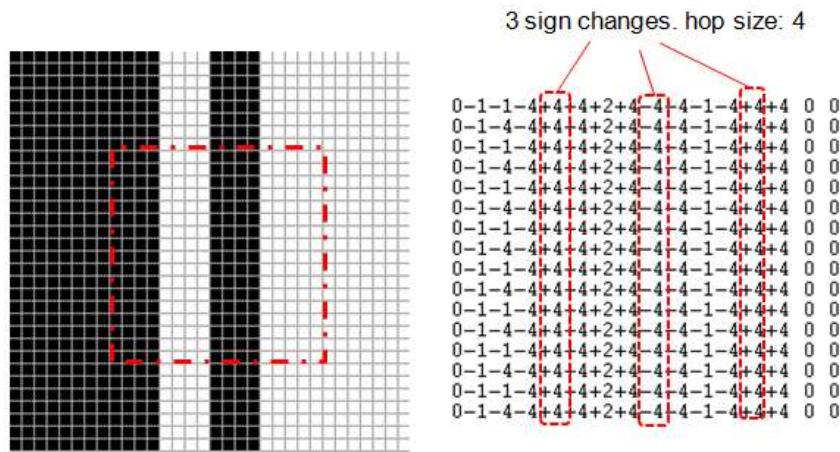


Figura 88. Ejemplo de cálculo de PRx en un bloque que contiene dos bordes abruptos

El numerador será num\_scanlines x 3cambios x 4 = 16 x 12 = 192

El denominador será: num\_scanlines x 3 cambios x 4 = 192

En caso de haber tenido un único borde abrupto la métrica habría salido exactamente igual, pues no depende de cuantos bordes hay sino de lo abruptos que son

Un borde abrupto exige una muestra por cada pixel para reproducir exactamente la percepción. Si usamos menos e interpolamos, estaremos suavizando el borde. Entre un borde suave como se ha mostrado en el ejemplo anterior y un borde abrupto como el de este ejemplo, se encuentran bordes de muy diversa intensidad con valores de PR intermedios, los cuales reflejan la “no interpolabilidad” de la señal.

#### Ruido: PR >0.5

En este ejemplo hemos generado luminancia aleatoria y el PR ha subido mucho aunque no ha llegado a 1. PRx= 0.88, PRy=0.89

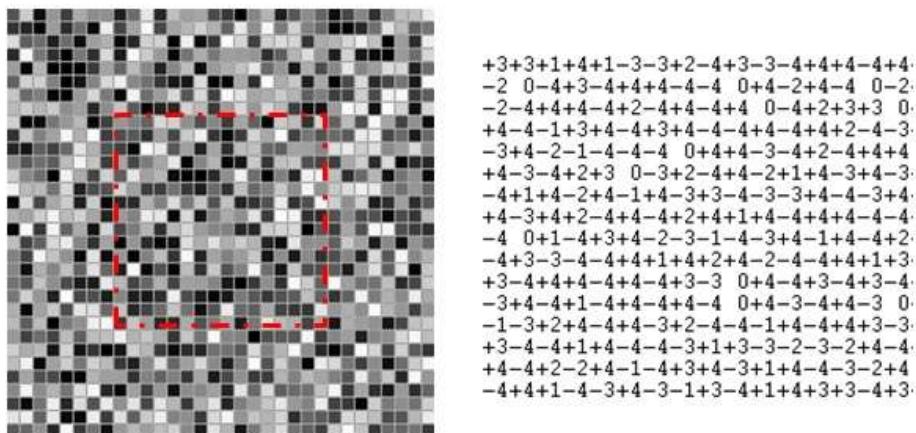


Figura 89. Ejemplo de cálculo de PRx en un bloque que contiene ruido

Como se observa el ruido es bastante abrupto pero no tanto como un borde bien definido

### Un borde no muy abrupto, con PR=0.5

La métrica de PR toma valores desde cero (ningún cambio de signo) hasta 1 (todos los cambios de signo abruptos, de tamaño  $H_{max}$ ). Vamos a analizar el significado del valor intermedio de PR, es decir, 0.5.

Tomemos como ejemplo el siguiente bloque de 4x4, en el que hay presente un borde. Es un borde donde el cambio de luminancia se ha efectuado en dos píxeles, de modo que no es tan abrupto como el ejemplo de PR=1. De hecho el cálculo de PRx da como resultado 0.5

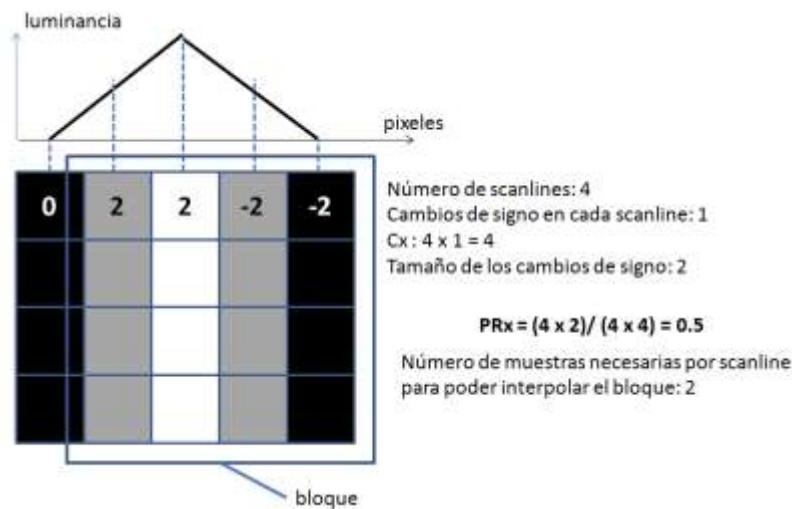
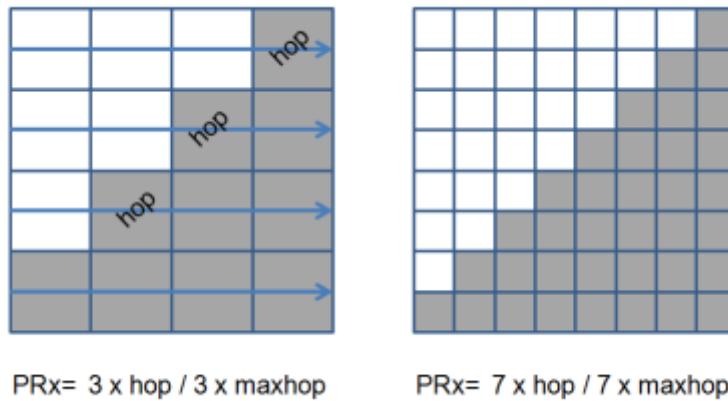


Figura 90. Un bloque con PR=0.5 requiere una muestra por cada dos píxeles originales

Como se puede apreciar, con PR=0.5 se requiere dos muestras por cada scanline para hacer una interpolación aproximada del bloque original, es decir, 1 muestra por cada dos píxeles originales (PPP=2) y para valores de PR superiores hará falta 1 muestra por cada algo menos de dos píxeles originales (PPP<2).

A partir de PR=0.5 es necesaria una muestra por cada dos píxeles o menos (hasta una) para poder reproducir fielmente la percepción creada por la imagen original. En el siguiente apartado se profundizará en estos casos de **cercanía a la saturación** de la información perceptual.

Tal como se ha definido la métrica, es independiente de la resolución, debido a que si un bloque que contiene un borde es muy grande, el numerador contendrá muchos sumandos, pero el denominador también, dando como resultado la misma métrica, tal como se aprecia en este ejemplo de bloques de tamaño 4x4 y 8x8 respectivamente. Ambos poseen el mismo valor de métrica, siendo versiones de la misma imagen a diferente resolución.



**Figura 91. Misma imagen a diferentes escalas, poseen la misma métrica de relevancia perceptual**

La métrica de relevancia perceptual (PR) es una contribución de la presente tesis. Tanto el concepto como su fórmula. No existe nada parecido en el estado del arte.

La métrica así definida mide la interpolabilidad de una imagen o fragmento de una imagen a partir de la información sensorial (los hops) generada por el cuantizador LHE. Una imagen va a contener mayor información perceptual cuanto menos interpolable sea, por lo que la métrica tal como está definida, nos da un valor que podemos interpretar como “relevancia perceptual”.

Esta métrica es independiente de la resolución y su valor depende de lo abruptos o suaves que sean las transiciones de brillo y no de cuantas transiciones se encuentren en el bloque.

Un valor crítico de PR es 0.5, una vez alcanzado, es necesario al menos una muestra por cada dos píxeles originales para poder interpolar la imagen original.

#### 4.3.4 Cálculo de las métricas en los vértices de la malla

Para calcular la relevancia perceptual en las esquinas de cada bloque de nuestra malla, consideraremos una malla imaginaria cuyos bloques se sitúan centrados en cada esquina de nuestra malla “real”. Se calcula la relevancia perceptual en ambas coordenadas ( $PR_x$ ,  $PR_y$ ) en cada uno de estos bloques imaginarios y ese será el valor de PR de la esquina compartida por los 4 bloques que confluyen en ella.

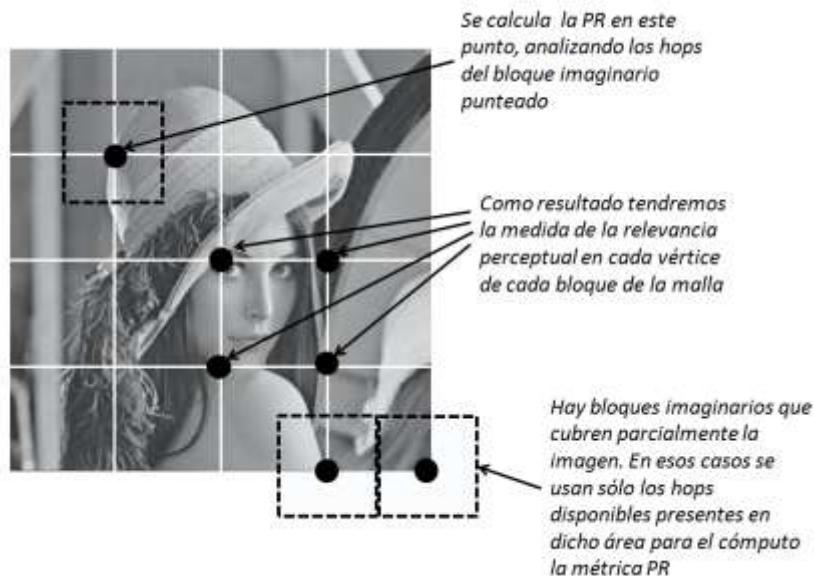


Figura 92. Métricas de PR en los vértices de la malla

#### 4.3.5 Optimización de las métricas en alta resolución

De cara a ejecutar este proceso a mayor velocidad, y teniendo en cuenta que las métricas de relevancia perceptual son independientes de la resolución, se puede acelerar el primer paso de la compresión (la cuantificación por hops logarítmicos) considerando un subconjunto de los pixeles de la imagen original

En el siguiente ejemplo, se ha calculado PR<sub>x</sub> y PR<sub>y</sub> en dos versiones de la imagen:

- Original: 1024x1024 pixels
- Sub-muestreada por Simple Pixel Selection ("SPS"): 512x512

En ambos casos se ha escogido una malla de 32x32 bloques



Figura 93. La misma imagen a distinta resolución

El promedio de diferencia de valor tanto en PR<sub>x</sub> como en PR<sub>y</sub> para ambas imágenes es del 3%, por lo que no es una diferencia significativa. Por lo tanto podemos calcular las métricas de relevancia perceptual sobre una imagen sub-muestreada por SPS para acelerar el cálculo. De hecho de cara a optimizar el proceso no es necesaria sub-muestrearla, basta con ejecutar la cuantificación LHE inicial sobre 512 pixels equiespaciados de la imagen original (suponiendo que es mayor). Puesto que los valores de PR son casi idénticos, también lo son los valores finales de PPP y en consecuencia el escalado elástico es prácticamente el mismo tal y como se muestra en la siguiente figura

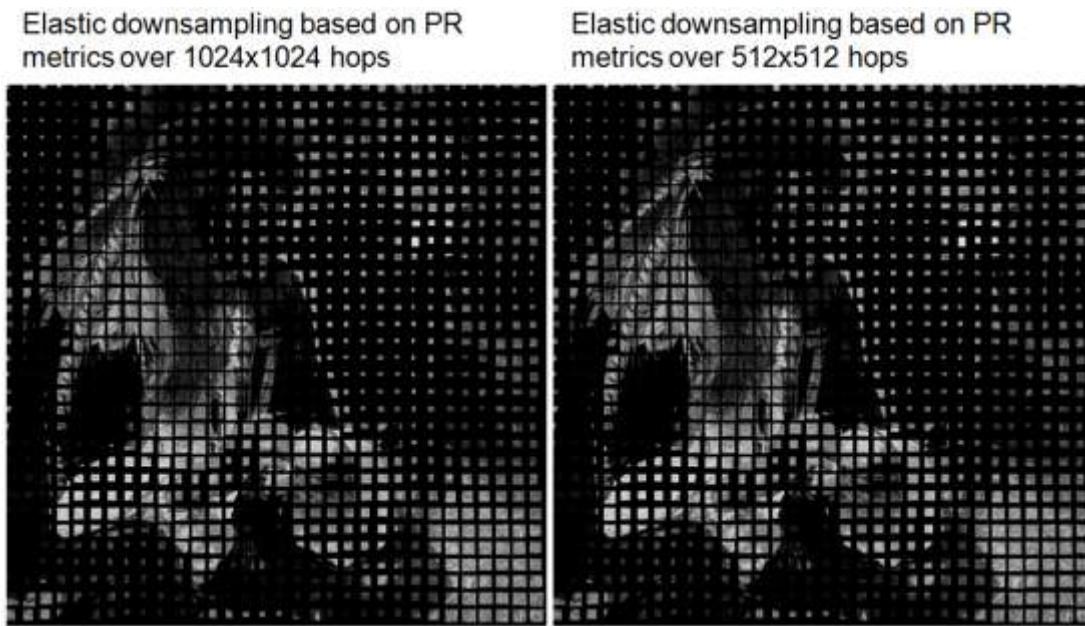


Figura 94. Las métricas obtenidas realizando un primer LHE sobre un sub-muestreo de la imagen original son correctas

Esta optimización es importante incluso en el caso de disponer de paralelización pues el trabajo a repartir entre cada hilo que calcule la primera cuantificación LHE será menor

La optimización del proceso de cálculo de las métricas mediante la ejecución del primer LHE sobre una imagen sub-muestreada por “Simple Pixel Selection” es una contribución de la presente tesis, apoyada en la definición de la métrica de relevancia perceptual independiente de la resolución que he creado.

#### 4.3.6 Saturación de la métrica de relevancia perceptual

Si la información sensorial es muy elevada, la percepción queda saturada. Es lo que ocurre con los bordes abruptos y con el ruido. Llega un momento que la percepción es la misma tanto si hay un poco más de información sensorial como si hay un poco menos. Existe un umbral a partir del cual se puede considerar que la información perceptual apenas aumenta, aunque haya margen para más información sensorial. En la siguiente figura aparece la distribución estadística de la métrica de relevancia perceptual de la imagen “Lena” y del ruido blanco, usando las fórmulas presentadas en el apartado anterior y considerando tanto PR<sub>x</sub> como PR<sub>y</sub> en la misma curva de distribución.

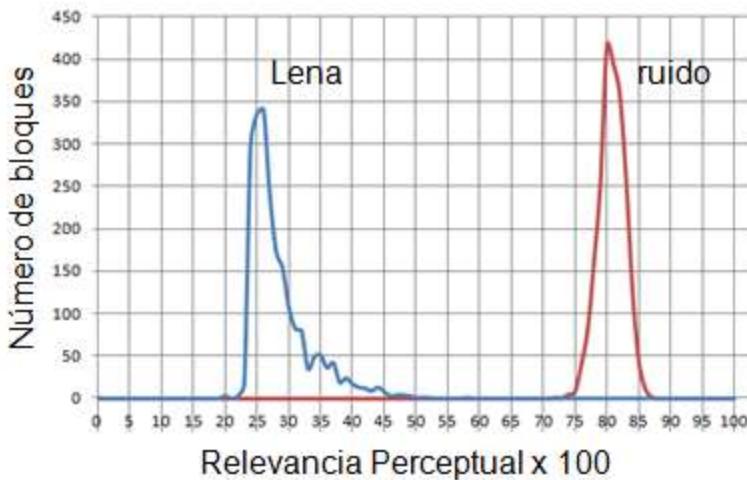


Figura 95. Histograma de valores de la métrica de PRx y PRy para dos imágenes diferentes

Teniendo en cuenta los casos analizados en el apartado anterior, se puede afirmar que “Lena” es una imagen esencialmente “suave”, con un pico en PR=0.25, aunque posee un cierto porcentaje (muy bajo) de bloques que superan el valor PR=0.5 (necesitando una muestra por cada dos pixeles o menos para su correcta interpolación, tal como hemos visto en uno de los ejemplos anteriores.)

Cuando incrementos adicionales de información sensorial no generan un incremento significativo en la información perceptual, diremos que hemos entrado en “saturación perceptual”. Este umbral de comienzo de saturación perceptual se puede situar en PR=0.5, que es cuando se comienza a necesitar menos de 1 muestra por cada dos pixeles originales para poder reproducir la misma percepción (y por consiguiente  $PPP \leq 2$ ).

A partir de PR=0.5 la información perceptual es muy elevada, por lo que interesa preservar la información de igual modo (mismo PPP) para todos los valores superiores a 0.5 ya que de lo contrario se priorizaría una percepción ruidosa frente a una información perceptual elevada pero no ruidosa. En los ejemplos que se muestran a continuación se puede ver gráficamente la necesidad de establecer un umbral de saturación de información perceptual.

#### Observación

*Con las siguientes imágenes que a continuación se muestran, se adelantan resultados para los que es necesario efectuar el sub-muestreo e interpolación basado en la métrica de relevancia perceptual, los cuales se explicarán en secciones posteriores de este capítulo. Sin embargo conviene presentar este resultado para expresar las ventajas de topar la relevancia perceptual en 0.5, justo en el comienzo de la saturación, antes de transformar el valor de PR en un valor de PPP*

En estas dos imágenes se ha conservado un 10% de la información original, basándose en la métrica de relevancia perceptual. Con un 10% se indica que al escalar espacialmente los bloques constituyentes, se ha reducido a un 10% del número de pixeles originales. Unos bloques se han reducido más y otros menos, en función del valor de la métrica de (PRx, PRy) de cada bloque. La diferencia entre ambas imágenes es que en la segunda se ha conservado la misma cantidad de información perceptual (mismo PPP) en todas aquellos bloques donde  $PR \geq 0.5$ .

Como se puede apreciar, en la primera imagen vemos dos problemas

- El coche ha salido borroso, por que se han usado menos PPP en la textura ruidosa que posee un PR mayor, y más en el coche pues su PR es menor (mayor PPP implica menos muestras)
- La textura ruidosa no se ha protegido de forma homogénea, apareciendo “borrones” dentro de la textura debido a fluctuaciones de PR

Ambos problemas desaparecen en la segunda imagen, donde se ha topado el valor de PR a 0.5. El resultado es obviamente mejor y el número de muestras es el mismo (un 10%)

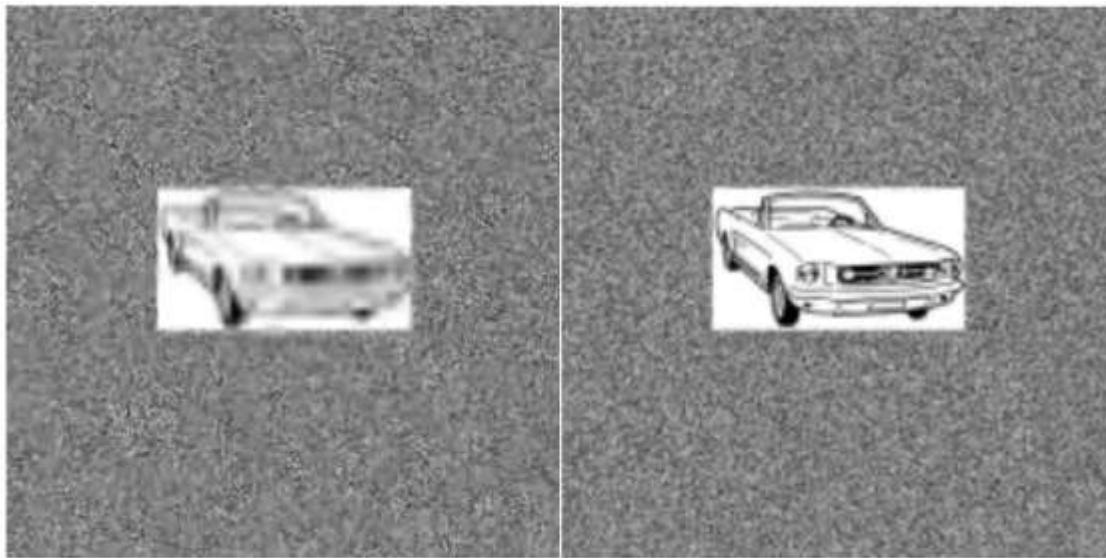


Figura 96. Resultados sin topar y topando la PR en 0.5. Ambas imágenes igualmente reducidas al 10%

El establecimiento de un umbral de saturación de Relevancia Perceptual es una contribución de esta tesis. Este umbral lo establecemos justo cuando empiezan a ser necesarios menos de una muestra por cada dos pixels ( $PPP \leq 2$ ) para regenerar una imagen por interpolación produciendo la misma percepción que la original, es decir, para un valor de  $PR=0.5$

#### 4.4 Expansión del histograma de la Relevancia Perceptual

Para ubicar el contenido de este apartado, a continuación se muestra el diagrama de bloques de LHE avanzado, destacando el módulo en el que se ubica el contenido que se va a desarrollar.

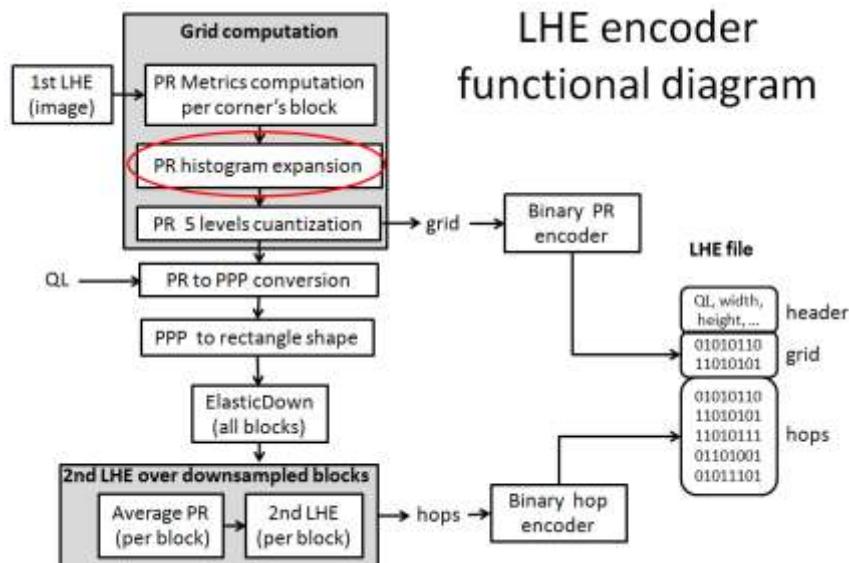


Figura 97. Ubicación del contenido que se va a desarrollar

La métrica de relevancia perceptual es un número decimal en el intervalo [0,1] o si se encuentra topada, en el intervalo [0, tope], siendo “tope” < 1. Si tenemos en cuenta que pretendemos dividir una imagen en bloques y calcular su relevancia perceptual en cada esquina de cada bloque, el almacenaje de estos números puede suponer una gran cantidad de información.

Lo más adecuado para ahorrar información es cuantizar la métrica de PR en unos pocos niveles. El problema es que si el histograma de PR de la imagen se encuentra muy compactado, corremos el riesgo de que todos los valores de PR se cuanticen al mismo valor y no podremos posteriormente asignar más o menos PPP en base a esta métrica cuantizada. Es lo que ocurre en la siguiente figura, donde todos los valores de PR pertenecen al intervalo cubierto por el cuarto “q2” y por consiguiente al convertir su valor a PPP no habrá diferencias entre ellos.

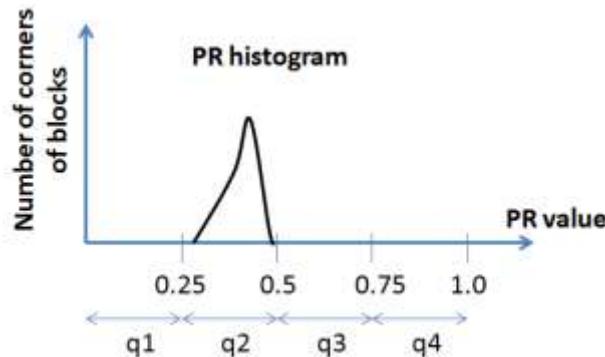


Figura 98. Histograma de PR y cuantización a un mismo cuarto (q2)

Es por ello que resulta interesante corregir la distribución de PR, expandiéndola en el intervalo [0,1]. Para ello tomaremos un intervalo de PR a expandir, concretamente el intervalo [0.2, 0.5]. Los límites de este intervalo tienen la siguiente justificación:

- **Límite inferior ( $PR_{min} = 0.2$ ):** los valores de PR por debajo de 0.2 son el resultado de unos hops con pocas fluctuaciones y muy suaves, casi lisos. Tras la expansión tomarán el valor cero.

- **Límite superior ( $PR_{max} = 0.5$ ):** al topar la métrica de PR en 0.5 como umbral de saturación, no se obtendrán valores superiores en el histograma, de modo que tras la expansión, los valores de  $PR=0.5$  alcanzarán el valor  $PR=1.0$

Veamos como la misma imagen del ejemplo anterior es cuantizada después de la corrección de intervalo en diferentes cuantos a los que se asignan diferentes PPP, de modo que podremos tener más información en las zonas con mayor medida de relevancia perceptual.

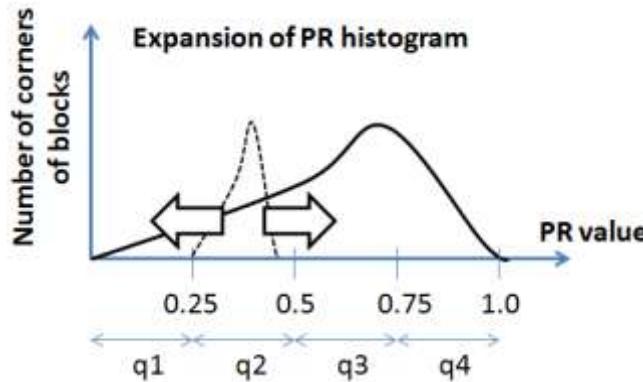


Figura 99. Expansión del histograma de PR antes de cuantizar

La ecuación necesaria para realizar el proceso de expansión del histograma se presenta a continuación. Es una simplificación de la formula general de expansión de cualquier histograma, donde los valores finales mínimo y máximo tras la expansión son 0 y 1 respectivamente. Como el valor mínimo del intervalo a expandir es  $PR_{min} = 0.2$ , cualquier valor de  $PR_{inicial}$  que se encuentre por debajo se debe convertir en  $PR_{final} = 0$  tras la expansión.

$$PR_{final} = \frac{PR_{inicial} - PR_{min}}{PR_{max} - PR_{min}}, \quad \text{donde } PR_{min} = 0.2 \text{ y } PR_{max} = 0.5$$

Ecuación 17. Expansión del histograma de PR

#### Contribución:

La expansión del histograma de una variable es una estrategia ampliamente utilizada en tratamiento de señales, no pudiendo considerarla en sí misma como una contribución de esta tesis. Sin embargo tanto la exclusión de los valores saturados para la ecualización como la propia aplicación de esta estrategia sobre una métrica completamente nueva supone una contribución en cuanto a lo que es la definición del procedimiento a seguir para poder transformar la métrica de PR en un valor óptimo de PPP, que diferencie con claridad donde hay más PR y donde hay menos, con independencia de la poca diferencia inicial que pudiese haber

## 4.5 Cuantización de la métrica de relevancia perceptual

Para ubicar el contenido de este apartado, a continuación se muestra el diagrama de bloques de LHE avanzado, destacando el módulo en el que se ubica el contenido que se va a desarrollar.

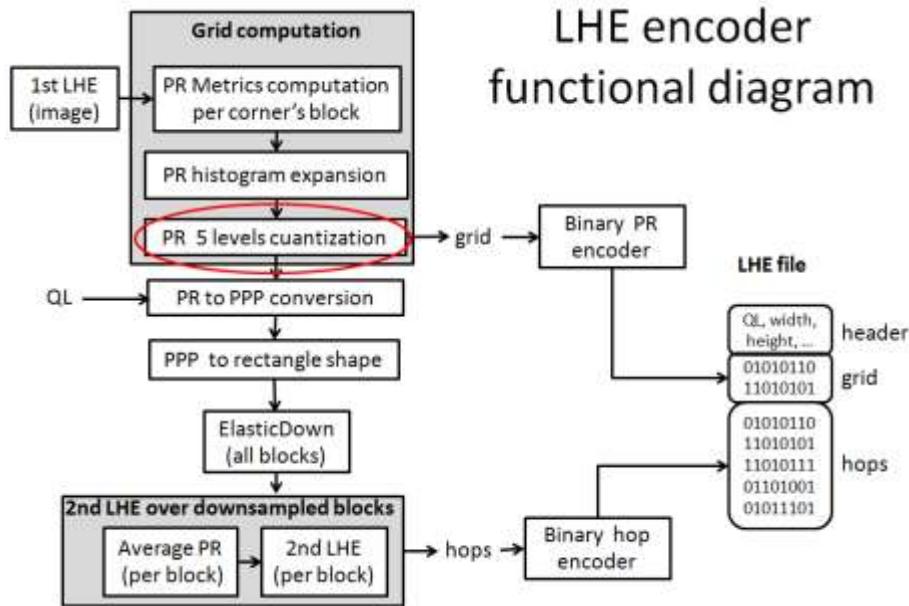


Figura 100. Ubicación del contenido que se va a desarrollar

A continuación procedemos a la cuantización de PR en cinco posibles valores. Para ello se divide el intervalo [0,1] en 5 sub-intervalos. Cada sub-intervalo va a representar aquellas áreas de la imagen con valor de PR similar.

Intervalo del cuanto	Valor PR' final asignado al cuanto	Comentario/aclaración
[0.75, 1.0]	PR'=1.0	Todos los bloques cuya PR fue topada en 0.5 caen en este intervalo tras la expansión del histograma y por tanto poseen el mismo tratamiento, es decir, todos se transforman en PR'=1.
[0.5, 0.75)	PR'=0.5	Valor asignado de PR' mitad del valor asignado al intervalo superior
[0.25, 0.5)	PR'=0.25	Valor asignado de PR' mitad del valor asignado al intervalo superior
[0.125, 0.25)	PR'=0.125	Valor asignado de PR' mitad del valor asignado al intervalo superior
[0.0, 0.125)	PR'=0.0	Para bloques sin apenas relevancia

Tabla 12. Intervalos para la cuantización y valores de PR asignados a cada cuanto

Al valor de PR asignado a cada intervalo de PR se le va a denotar como PR'.

En realidad estos 5 intervalos responden a una división equitativa de todo el rango de PR [0,1] en 4 intervalos, a los que se añade un intervalo especial, el más pequeño (de 0 a 0.125). Este intervalo menor está pensado para aquellos bloques de la imagen con una muy baja relevancia perceptual, tan baja que se transformarán en PR'=0.

El valor de PR que se asigna a las muestras que pertenecen a cada intervalo, no será el valor medio de cada intervalo sino un valor que permite una mayor diferenciación entre las zonas de mayor y menor

relevancia perceptual, de modo que tras comparar resultados con diferentes estrategias, la que produce las imágenes más naturales es la que resulta de asignar a PR valores en sucesión geométrica de razón=2 (Tabla 12).

Esta estrategia acentúa las diferencias con el intervalo anterior, pero sin llegar a causar discontinuidades de resolución visibles en la imagen resultante tras aplicar la conversión a PPP (Pixels per pixel), o lo que es lo mismo, número de pixels que representa cada muestra (la fórmula de conversión de PR a PPP será objeto de un apartado posterior).

Otras distribuciones de valores de PR' equidistantes o distribuidos de forma logarítmica han dado peores resultados tanto de PSNR (resultado objetivo) como subjetivamente. Una distribución donde asignemos el punto medio del intervalo al valor del cuanto produce valores de PSNR ligeramente peores en el resultado final. Además, la distribución geométrica parece mejor subjetivamente debido a que la diferencia de calidad entre bloques pertenecientes a diferentes cuantos de PR es mejor percibida.

La explicación matemática se encuentra en la relación entre PPP y PR', la cual es inversamente proporcional (como se presentará más adelante). Ello implica que incrementos lineales de PR no se traducen en incrementos lineales de PPP, sino que si una de ellas aumenta el doble, la otra disminuye a la mitad. En otras palabras, para poder representar el doble de detalles (pixels intermedios) hace falta muestrear el doble de pixels (PPP mitad) y para ello hay que doblar el valor de PR'.

En la siguiente imagen se aprecia una notable mejoría de calidad en los bordes de la hélice, en las letras impresas en el fuselaje del avión, en el tren de aterrizaje de la rueda delantera, etc



*Valor del cuanto: centro del intervalo  
Compresión final: 0.15bpp  
PSNR: 26.2dB*

*Valor del cuanto: asignación geométrica  
Compresión final: 0.15bpp  
PSNR: 28.0dB*

**Figura 101. Asignación de PR al punto medio del cuanto (izquierda) y asignación con distribución geométrica (derecha)**

Cuando lo que pretendemos es gastar el menor número de pixels posible, cada pixel que invertimos en un bloque de poca relevancia es un pixel que no invertimos en una zona de alta relevancia. Por ese motivo es mejor asignar geométricamente los valores de PR a los cuantos (Figura 101). Sin embargo una diferencia demasiado acentuada puede provocar cambios visibles de resolución antiestéticos. Con una cuantización con distribución geométrica de razón 2, los resultados han sido óptimos objetiva y subjetivamente.

La cuantización de la relevancia perceptual a valores geométricamente distribuidos es una contribución de esta tesis. Su elección responde a un criterio de calidad pero considerando la homogeneidad en las transiciones de resolución en la imagen resultante. Otras distribuciones de valores de PR' equidistantes

o distribuidos de forma logarítmica han dado peores resultados tanto estéticos como en la evaluación de la relación señal ruido (PSNR) de la imagen resultante. La cuantización se ha de llevar a cabo tras la expansión del histograma.

## 4.6 Conversión de Métrica de Relevancia Perceptual en PPP

Para ubicar el contenido de este apartado, a continuación se muestra el diagrama de bloques de LHE avanzado, destacando el módulo en el que se ubica el contenido que se va a desarrollar.

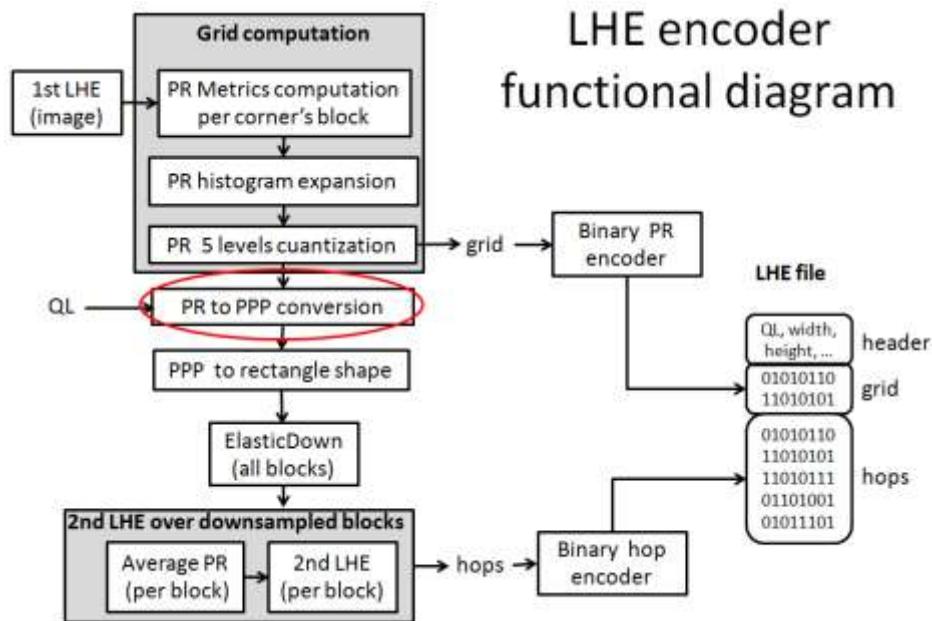


Figura 102. Ubicación del contenido que se va a desarrollar

El número de pixels originales que representa un pixel sub-muestreado se ha denominado “**PPP**” (**Pixels Per Pixel**). Este número no puede ser menor de 1. En un sub-muestreado homogéneo todos los pixels sub-muestreados representan el mismo número de pixels originales, pero gracias a la métrica de relevancia perceptual se puede asignar diferentes PPP en cada esquina de los bloques constituyentes en los que dividiremos la imagen y evolucionar el valor de PPP linealmente entre esquinas, quedando una imagen resultante sub-muestreada de forma elástica.

Lo primero que se ha de definir es la compresión máxima o el valor máximo de PPP al que llamaremos  $PPP_{max}$ . El valor de  $PPP_{max}$  está directamente relacionado con el tamaño de la imagen original y el número de bloques en la que se va a dividir. Esto es debido a que cada bloque será sub-muestreado por separado y el menor número de pixels en que se va a reducir un bloque será de 2x2. No se va a convertir un bloque en un solo pixel porque se requiere que cada esquina del bloque pueda tener diferente valor de PPP, de modo que se conservará siempre al menos un pixel por cada una de las 4 esquinas, es decir, el bloque sub-muestreado al menos medirá 2x2. A modo de ejemplo, si la imagen mide 512 pixeles de ancho y se divide en 32 bloques (de ancho), el valor de  $PPP_{max} = 0.5 \cdot \frac{512}{32} = 8$  permite compresiones de hasta 1:64 ya que se están considerando dos coordenadas (x e y).

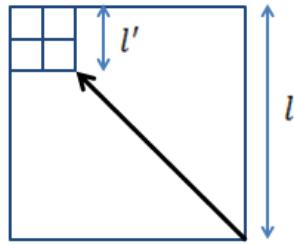


Figura 103. Máximo downsampling de un bloque a tamaño 2x2

Para deducir la fórmula que permite convertir la PR' en PPP consideraré una sola dimensión, no diferenciaré si es PR'x o PR'y lo que estamos calculando. Las fórmulas son válidas tanto para PR como para PR', pero se mostrará PR' en ellas para clarificar que este proceso se debe llevar a cabo tras haber asignado el valor del cuanto PR' al intervalo de PR considerado.

La relación entre la longitud original (en pixels) de un segmento de señal y la longitud del mismo segmento sub-muestreado es el valor de los PPP utilizados, llegando a 1 en el caso de que el segmento sub-muestreado sea igual al original.

En las siguientes ecuaciones se denota por  $l$  a la longitud del segmento original y  $l'$  a la longitud del segmento sub-muestreado. El cociente entre ellas es precisamente  $PPP = l/l'$

Las longitudes mínima (máxima compresión) y máxima (compresión nula) del segmento sub-muestreado considerado serán:

$$l'_{min} = \frac{l}{PPP_{max}}$$

$$l'_{max} = l$$

Ecuación 18. Longitudes mínima y máxima del segmento sub-muestreado

Dependiendo del valor de la métrica PR, se ha de asignar a  $l'$  un valor comprendido entre  $l'_{min}$  y  $l'_{max}$ . Considerando en el cálculo que la métrica PR tiene un valor comprendido en el intervalo [0,1]

$$l' = l'_{min} + (l - l'_{min}) \cdot PR'$$

$$l' = \frac{l}{PPP_{max}} + (l - \frac{l}{PPP_{max}}) \cdot PR'$$

$$l' = l \left( \frac{1 + (PPP_{max} - 1) \cdot PR'}{PPP_{max}} \right)$$

Y teniendo en cuenta que:

$$PPP = \frac{l}{l'}$$

Entonces, tenemos:

$$PPP = \frac{PPP_{max}}{1 + (PPP_{max} - 1) \cdot PR'}$$

Ecuación 19. Fórmula de obtención de PPP a partir de PR

Esta fórmula permite calcular exactamente los PPP a asignar en las esquinas de un bloque a partir de su métrica de relevancia perceptual PR.

A esta ecuación le queda un factor a añadir, que es precisamente el factor de compresión deseado (“CF”). Este factor será el mismo para todos los bloques de la imagen y multiplicará a los PPP resultantes ofreciendo un resultado que hará aumentar los *PPP* desde el valor mínimo obtenido hasta el valor máximo  $PPP_{max}$  para una máxima compresión. Una vez alcanzado este valor  $PPP_{max}$ , no se superará, ya que si se supera estaríamos comprimiendo más de lo máximo que hemos definido. Igualmente un factor multiplicativo que reduzca por debajo de 1 el resultado tampoco podrá considerarse. Los topes de PPP serán 1 y  $PPP_{max}$

$$PPP = \frac{PPP_{max}}{1 + (PPP_{max} - 1) \cdot PR'} \cdot CF$$

*if*  $PPP > PPP_{max}$  *then*  $PPP = PPP_{max}$

*else if*  $PPP < 1$  *then*  $PPP = 1$

#### Ecuación 20. Fórmula de PPP con límites en valor máximo y mínimo

Esta es la ecuación con la que traducir PR a PPP que es ajustable mediante el parámetro “CF” para obtener versiones comprimidas de una imagen, siempre otorgando menos PPP a aquellas zonas de mayor PR.

Los PPP asignados con esta última ecuación podrán recorrer el intervalo  $[1, PPP_{max}]$ . En cuanto al valor de CF, puede tomar valores desde  $1/PPP_{max}$  hasta  $PPP_{max}$ , ya que:

Cuando CF toma su valor máximo, el valor de PPP toma el valor  $PPP_{max}$  para  $PR' = PR_{max}$

$$PPP_{max} = \frac{PPP_{max} \cdot CF_{max}}{1 + (PPP_{max} - 1) \cdot PR_{max}}$$

Por tanto:

$$CF_{max} = 1 + (PPP_{max} - 1) \cdot PR_{max}$$

#### Ecuación 21. Cálculo del valor máximo del factor de compresión

En cuanto al valor mínimo, CF toma su valor mínimo cuando PPP vale 1 para  $PR = PR_{min}$

$$1 = \frac{PPP_{max} \cdot CF_{min}}{1 + (PPP_{max} - 1) \cdot PR_{min}}$$

$$CF_{min} = \frac{1 + (PPP_{max} - 1) \cdot PR_{min}}{PPP_{max}}$$

#### Ecuación 22. Cálculo del valor mínimo del factor de compresión

Teniendo en cuenta que  $PR_{max} = 1$  y que  $PR_{min} = 0$  tenemos que  $CF \in [1/PPP_{max}, PPP_{max}]$ , con los significados que se describen en la siguiente figura:

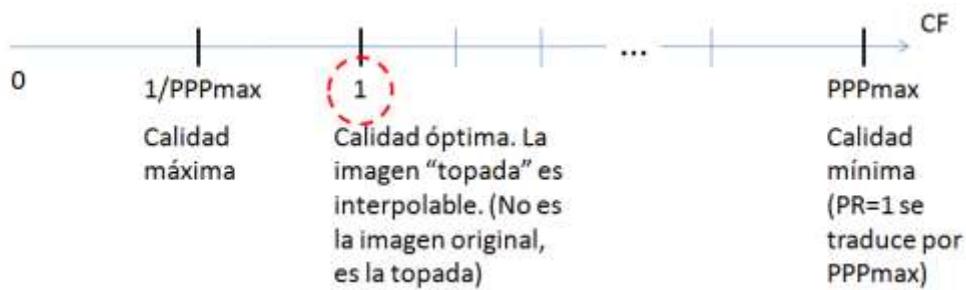


Figura 104. Rango de valores de CF para la conversión de PR en PPP

La siguiente figura muestra la evolución de los PPP en los que se transforman diferentes valores de PR en función del factor CF, para una imagen donde  $\text{PPP}_{\text{max}} = 8$ .

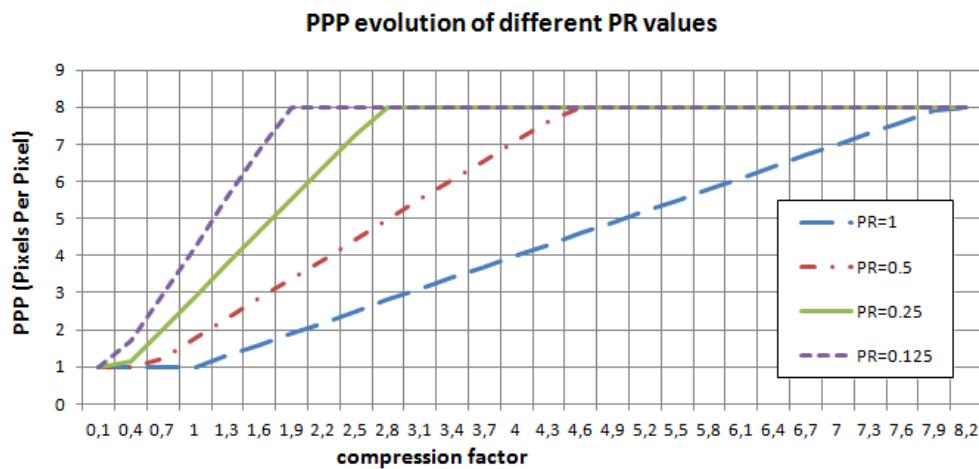


Figura 105. Evolución de PPP en función de PR y el factor de compresión CF

Manejar el factor CF para lograr la compresión deseada es difícil pues los límites dependen del valor de  $\text{PPP}_{\text{max}}$ , el cual va a depender, como veremos más adelante, del tamaño de la imagen. Por otro lado cada vez que CF se multiplica por dos, los PPP se multiplican por 4 ya que se trabaja con dos coordenadas y los PPP resultantes finales serán la multiplicación de los PPP de la coordenada x por los PPP de la coordenada y. Por todo ello se hace más natural el uso de un "nivel de calidad" (QL), que pueda tomar valores entre 0 y 99. Para lograrlo, se divide el rango de variación de CF en 100 partes en escala geométrica, definiendo una razón "r" que al elevarla al máximo número de niveles transforme  $CF_{\text{min}}$  en  $CF_{\text{max}}$

$$CF_{\text{min}} \cdot r^{99} = CF_{\text{max}} \rightarrow r = \left( \frac{CF_{\text{max}}}{CF_{\text{min}}} \right)^{\frac{1}{99}}$$

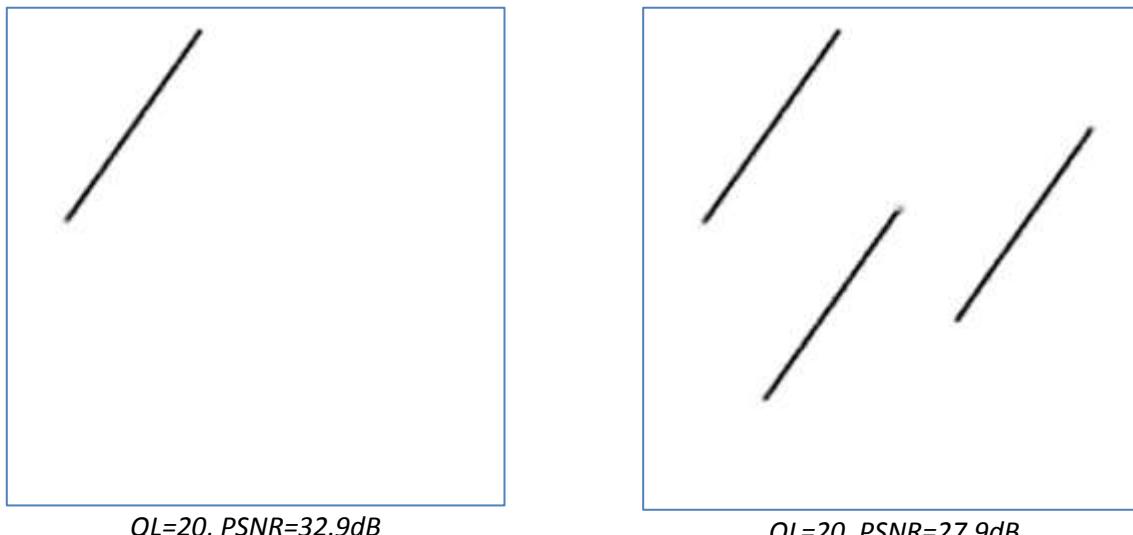
$$CF = CF_{\text{min}} \cdot r^{99-QL} \rightarrow PPP = \frac{\text{PPP}_{\text{max}} \cdot CF_{\text{min}} \cdot r^{99-QL}}{1 + (\text{PPP}_{\text{max}} - 1) \cdot PR'} \quad \text{donde } QL \in [0,99]$$

Ecuación 23. Nivel de calidad ("QL" o "Quality Level") y reformulación de PPP

El motivo para escoger una razón geométrica es el siguiente: cada vez que subimos de nivel, estaremos multiplicando por  $r$  al factor de compresión CF y por lo tanto estaremos añadiendo un cierto porcentaje de muestras sobre las ya existentes. Si no usásemos una razón, y dividiésemos el intervalo en partes iguales, cada nivel no implicaría una mejora porcentual, por lo que los niveles más altos apenas producirían mejoras significativas, mientras que los niveles bajos producirían cambios enormes de calidad.

Con este nuevo parámetro “Quality Level” (QL) se ha dividido en 100 calidades distintas las posibilidades de compresión y dadas dos imágenes de igual tamaño (y por tanto mismo valor de  $PPP_{max}$ ) obtendremos la misma “calidad equivalente”, ya que a igualdad de PR tendremos los mismos PPP en ambas imágenes. Esto no significa que ambas vayan a tener el mismo PSNR ni los mismos bpp finales, ya que dependerá de cuantos bloques suaves y abruptos haya en cada una de las imágenes pero al menos la calidad (el valor de PPP) con la que se comprime un área con el mismo PR será la misma. Este nuevo concepto de “calidad equivalente” es el significado físico del parámetro QL.

En el siguiente ejemplo han sido comprimidas dos imágenes con el mismo QL. Ambas presentan diferente PSNR porque una contiene un borde y la otra 3. La que contiene 3 bordes lógicamente acumula más errores y su PSNR es inferior. Sin embargo ambas imágenes codifican con igual calidad cada borde, es decir, poseen la misma “calidad equivalente”.



**Figura 106. Dos imágenes con igual nivel de calidad (QL) y distinto PSNR**

En la Figura 107, se muestran dos imágenes donde nuestro personaje tiene sus bordes igualmente degradados. En una de las imágenes aparece rodeado de flores, y son los errores en las flores los que contribuyen a disminuir el PSNR, ya que como se puede apreciar, el personaje aparece igualmente degradado. Es decir, a diferencia del PSNR, el QL permite cuantificar el nivel de degradación de una imagen debido a LHE con independencia de qué imagen se trata.

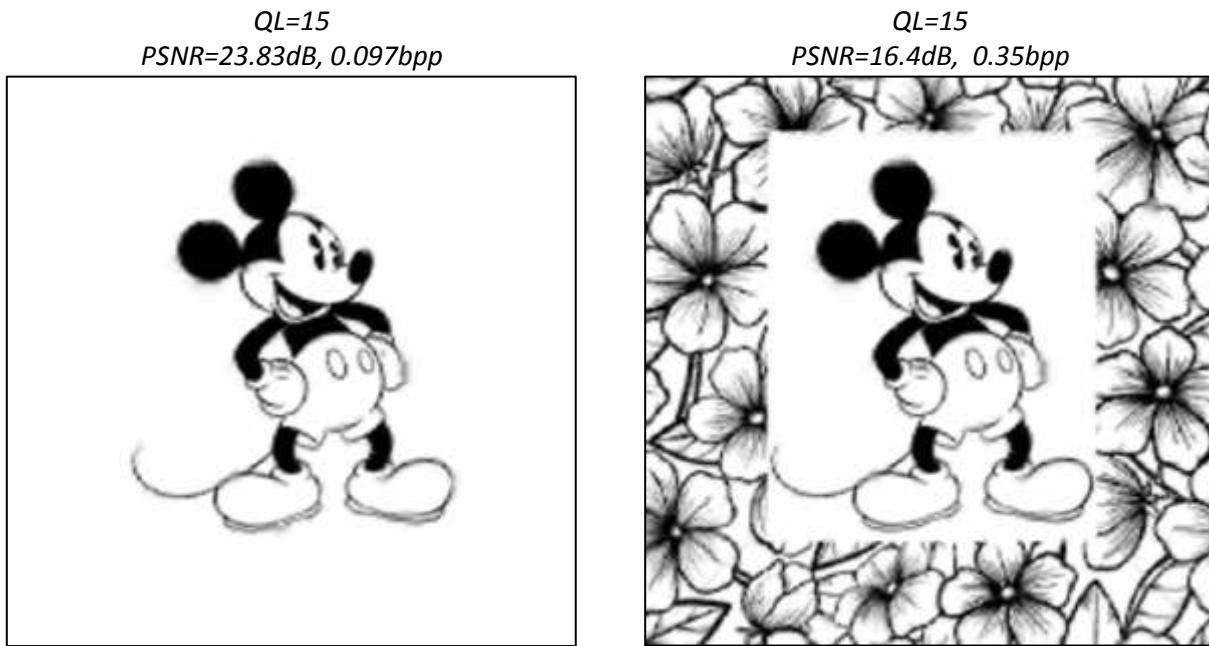


Figura 107. Otras dos imágenes con igual nivel de calidad (QL) y distinto PSNR y tasa de bit

Un ajuste adicional que se realiza es asignar siempre  $PPP = PPP_{max}$  cuando  $PR' = 0$ , en lugar de aplicar la Ecuación 22, ya que de lo contrario, se asignaría menos PPP (es decir, mas muestras) a medida que aumenta QL y no merece la pena gastar muestras en zonas de extremadamente baja relevancia perceptual. Con este ajuste, el valor de  $PR_{min}$  en la Ecuación 22 será el del cuanto inmediatamente superior, es decir  $PR_{min} = 0.125$ , tal como se define en la Tabla 12, por lo que el  $CF_{min}$  será algo mayor que  $1/PPP_{max}$ .

Aunque es necesario comprender el downsampling y la interpolación elástica para generar las siguientes imágenes, adelanto estos resultados para ilustrar el concepto de Quality Level (QL).





Figura 108. Diferentes resultados con diferentes valores de QL

La fórmula de conversión desde la métrica de PR al valor de PPP con el que se va a realizar el procedimiento de downsampling es una contribución de esta tesis. Esta conversión permite asignar de forma óptima PPP a las esquinas de cada bloque en que se divide una imagen, teniendo en cuenta tanto la relevancia perceptual de cada esquina como el grado de compresión espacial que se quiere alcanzar.

La fórmula de la conversión requiere de un parámetro que define el grado de compresión al que queremos someter a la imagen y al que he denominado “Quality Level” (QL), el cual puede tomar un valor entre 0 y 99. Dos imágenes con el mismo valor de QL tendrán la misma “calidad equivalente”, es decir, los mismos PPP para la misma relevancia perceptual. La “calidad equivalente” es una aportación de esta tesis.

Un nivel (QL) al estar relacionados mediante una razón geométrica con el nivel anterior, producirá siempre una mejora del mismo valor porcentual respecto del nivel anterior, tanto si es un nivel bajo como si es un nivel alto.

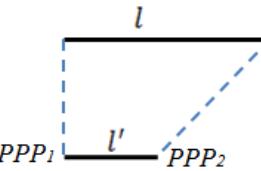
## 4.7 Ecuaciones del Downsampling Elástico

En este apartado se presentan los cálculos que permiten obtener tanto el gradiente de PPP como la longitud de un segmento sub-muestreado elásticamente a partir de los PPP iniciales y finales.

El desarrollo matemático se hará para una sola dimensión. El desarrollo para dos dimensiones no tiene ninguna variación, es el mismo, ya que se aplica a cada dimensión espacial por separado.

El cálculo del gradiente de PPP en la imagen sub-muestreada, va a permitir construir esta imagen sub-muestreada pixel a pixel, sabiendo exactamente cuántos píxeles originales corresponden con cada pixel de la imagen sub-muestreada.

Dado un segmento de longitud  $l$ , al sub-muestrearlo elásticamente entre  $PPP_1$  y  $PPP_2$  se quedará con una cierta longitud  $l'$  y el gradiente de PPP se puede calcular como:



$$\alpha = \frac{PPP_2 - PPP_1}{l' - 1}$$

Ecuación 24. Cálculo inicial del gradiente de PPP

Este gradiente no es el definitivo, como veremos a continuación. Si aplicamos este gradiente se debe cumplir que al sumar  $\alpha$  un número de veces  $l' - 1$ , se debe alcanzar el valor de  $PPP_2$ :

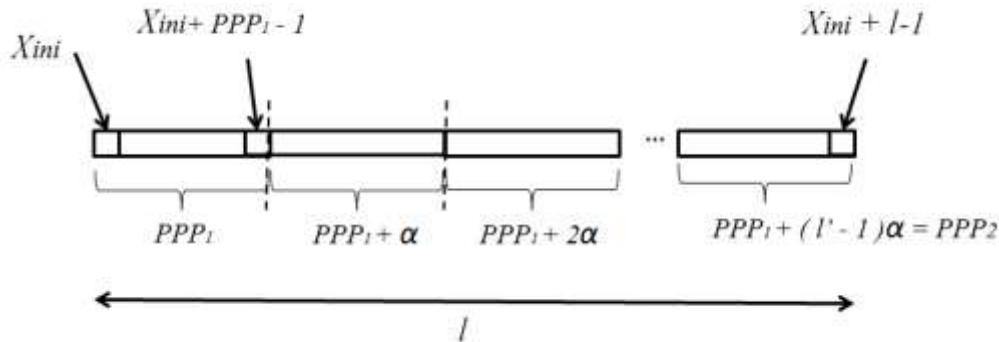


Figura 109. Serie aritmética para el cálculo del gradiente de PPP

Siendo:

$PPP_1$ : Pixels per pixel del extremo izquierdo del segmento

$PPP_2$ : Pixels per pixel del extremo derecho del segmento

$l$ : longitud del lado del bloque

$l'$ : longitud del lado submuestreado

$\alpha$ : gradiente de PPP del segmento

Puesto que estamos ante una serie aritmética se debe cumplir:

$$l = PPP_1 + (PPP_1 + \alpha) + (PPP_1 + 2\alpha) + \dots + (PPP_1 + (l' - 1) \cdot \alpha)$$

Ecuación 25. Serie aritmética

$$l = PPP_1 + (PPP_1 + \alpha + PPP_2) \cdot \frac{l' - 1}{2}$$

$$2l = 2PPP_1 + PPP_1 l' - PPP_1 + (l' - 1)\alpha + PPP_2 l' - PPP_2$$

$$\text{y puesto que } PPP_2 - PPP_1 = (l' - 1)\alpha$$

$$l' = \frac{2l}{PPP_1 + PPP_2}$$

Ecuación 26. Cálculo de  $l'$ 

Esta ecuación permite calcular la longitud del lado sub-muestreado. Sin embargo se trata de un número decimal y se trabaja con píxeles, de modo que se ha de redondear al valor entero superior más cercano.

$$l'' = \text{INT}(0.5 + l')$$

Ecuación 27. Ajuste de  $l'$  a un número entero

Al hacerlo, la Ecuación 26 deja de cumplirse. Para que se siga cumpliendo se debe reajustar el gradiente. Para calcular el nuevo gradiente, se sustituye el nuevo valor de  $l'$  calculado en la Ecuación 26 en la Ecuación 25 y se despeja el nuevo valor del gradiente:

$$\alpha' = \frac{2 \cdot l - 2 \cdot \text{PPP}_1 \cdot l''}{(l'' - 1) \cdot l''}, \quad \text{o bien,} \quad \alpha' = \frac{\text{PPP}_2 - \text{PPP}_1}{(l'' - 1)}$$

Ecuación 28. Fórmula final del gradiente de PPP

Aunque este desarrollo matemático es correcto, se ha asumido que cada vértice puede adoptar cualquier valor de PPP. A continuación se demostrará que eso no es viable, por lo que se ha optado por restringir los PPP de las esquinas para que el bloque sub-muestreado presente una forma rectangular. La justificación a forma rectangular responde a dos necesidades:

- La necesidad de poder codificar con LHE los bloques sub-muestreados resultantes
- La necesidad de hacer el downsampling e interpolación mediante un procedimiento separable, mucho más rápido y de sencilla implementación

A continuación se describen estas justificaciones.

#### 4.7.1 Downsampling Elástico sin restricciones en PPP

Un downsampling sin restricciones donde cada vértice puede tomar cualquier valor de PPP genera algo que no es un cuadrilátero, ya que aunque la transición de PPP entre vértices se realiza linealmente, eso no implica que se generen lados rectos



Figura 110. Downsampling Elástico de la imagen Lena sin restricciones

El problema de estas figuras es que si a continuación se han de codificar mediante un algoritmo de tipo scanline (LHE es así) lo cual representa un problema ya que no tenemos scanlines iguales. Cada scanline empieza y acaba en un punto diferente. Además, LHE necesita el pixel superior para codificar un pixel y a veces no se va a disponer del mismo, dependiendo del ángulo de cada lado.

Por otro lado, si la diferencia de PPP es muy grande entre vértices, la figura resultante puede “estrangularse”, lo cual se ha comprobado, y en esos casos es imposible (o al menos nada trivial) codificar la información sub-muestreada. En la Figura 111, el punto de estrangulamiento se corresponde con un punto de lado vertical derecho original y a la vez con otro punto lejano situado en el lado horizontal inferior. Este “choque” de pixels impide su correcta reconstrucción ya que se estaría

mezclando pixeles espacialmente muy lejanos. Además como fruto de la deformación aparece la imagen invertida por debajo de la línea de suelo, correspondiente al área que se ha deformado en exceso

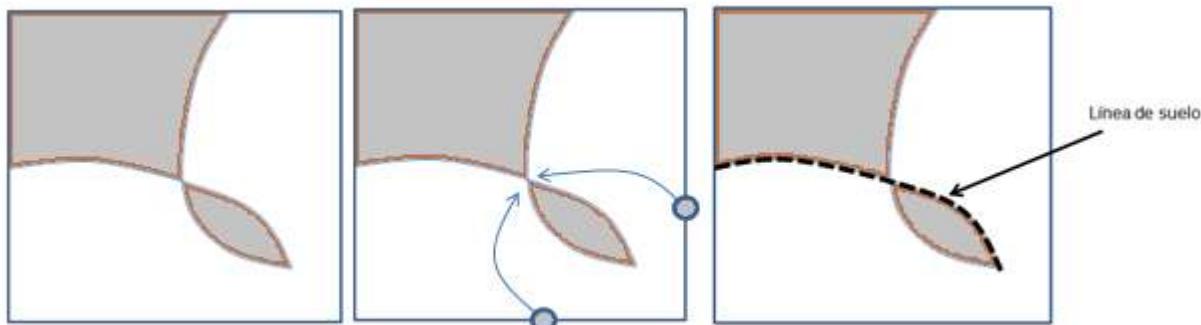


Figura 111. Efecto del estrangulamiento del downsampling sin restricciones

Centrando la figura sub-muestreada la situación sigue pudiéndose producir, aunque tolera mayor margen de diferencia en los PPP de las diferentes esquinas antes de empezar el “estrangulamiento”

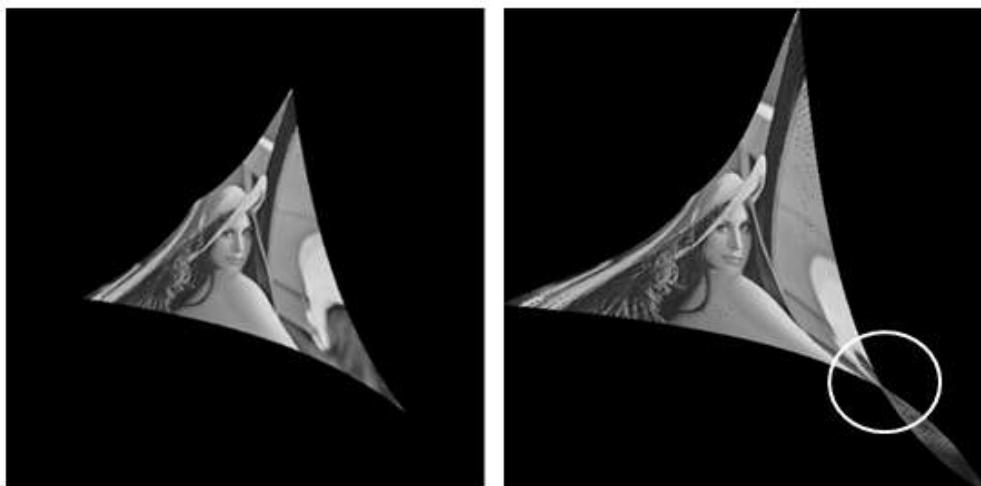


Figura 112. Efecto de estrangulamiento en imagen Lena

Por todo ello, y por la simplicidad y rendimiento del sub-muestreo a bloques rectangulares, se ha optado por ajustar los PPP de las esquinas de un bloque para que presente una forma rectangular.

Al optar por restringir los PPP de modo que presenten una forma rectangular, no solo se elimina el problema de los scanlines y el efecto de estrangulamiento, sino que además, el rendimiento del downsampling a forma rectangular es muy elevado porque **permite hacerlo de forma separable**, es decir, primero en dirección horizontal y después en vertical, ahorrando muchas operaciones.

Este criterio (el rendimiento) unido al problema de no disponer de scanlines, son los argumentos que han llevado a restringir los PPP a forma rectangular.

La forma rectangular va a provocar que los PPP de las esquinas de bloques adyacentes ya no van a ser exactamente los mismos pero si la transformación de PPP a nuevos valores de PPP que generen una forma rectangular se hace con suficiente sutileza (tal como se explicará a continuación), los bloques interpolados encajarán con suavidad, sin notarse saltos abruptos de resolución.

#### 4.7.2 Restricción de máxima elasticidad

Antes de aplicar la restricción a forma rectangular, se puede limitar la máxima elasticidad que puede tener un bloque, es decir, la máxima diferencia entre los PPP de lados opuestos.

Tiene sentido limitarlo porque si se analiza cómo se calculan las métricas, puede ocurrir que un borde localizado en la diagonal de un bloque sea degradado en la zona central, aunque en sus extremos posea un PR elevado y por tanto un PPP bajo.

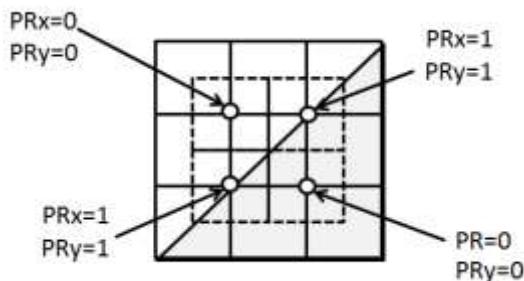


Figura 113. Un bloque con métricas de PR muy diferentes en sus esquinas

En este caso en la zona central del borde nos encontramos a la misma distancia de las esquinas de  $PR=0$  que da las esquinas con  $PR=1$ , con lo que es previsible que el PPP asignado se coloque entre medias de los valores de PPP asignados a las esquinas. Pero ese valor medio es demasiado alto para un borde abrupto y justo en la zona central se va a producir el emborronamiento.

Esto se ilustra en la siguiente imagen, un simple borde diagonal que en algunos casos produce dicho emborronamiento. Limitando la máxima diferencia de PPP al triple, este efecto se minimiza.

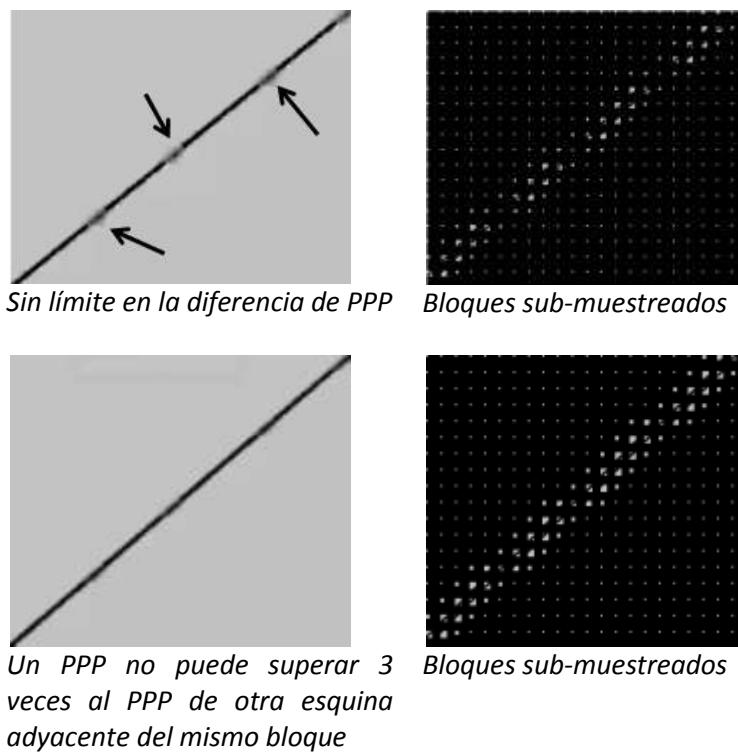


Figura 114. Límite a la máxima elasticidad

#### 4.7.3 Restricciones de PPP a forma rectangular

Para ubicar el contenido de este apartado, a continuación muestro el diagrama de bloques de LHE avanzado, destacando el módulo en el que se ubica el contenido que voy a desarrollar.

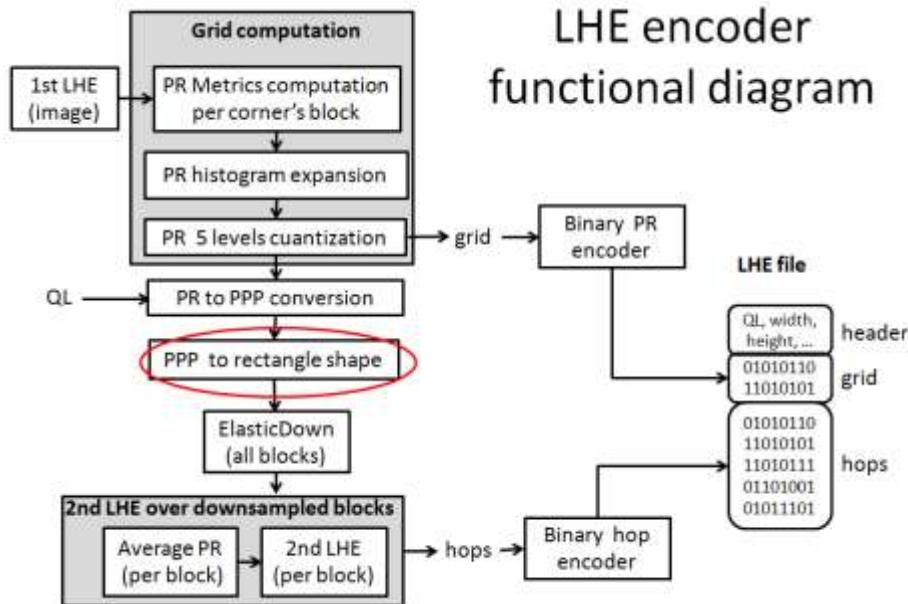
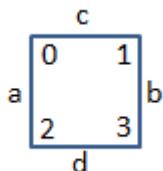


Figura 115. Ubicación del contenido que se va a desarrollar

Para que los PPP asignados a las esquinas de un bloque produzcan un bloque sub-muestreado de forma rectangular, se debe cumplir que la longitud de los lados verticales “a” y “b” debe ser la misma, y la longitud de los lados horizontales “c” y “d” debe ser la misma.



teniendo en cuenta que  $l' = \frac{2l}{PPP_1 + PPP_2}$   
(Siendo  $PPP_1$  y  $PPP_2$  los valores PPP en los extremos de un segmento), entonces si estamos ante un rectángulo:

$$\begin{aligned} PPP_{0x} + PPP_{1x} &= PPP_{2x} + PPP_{3x} \\ PPP_{0y} + PPP_{2y} &= PPP_{1y} + PPP_{3y} \end{aligned}$$

Ecuación 29. Condición rectangular

Para que esto suceda debemos modificar los PPP que hemos asignado a las esquinas de un modo proporcional, de modo que los ajustemos sin alterar demasiado la deformación que queríamos producir, pero produciendo la forma final rectangular

Llamaremos  $S_a$  a la suma de PPP de los extremos del lado “a”,  $S_b$  es la suma correspondiente al lado “b” y  $S$  será la suma de los PPP tras la transformación de los  $PPP_{ij}$  en nuevos valores a los que llamaremos  $PPP'_{ij}$  con los que convertiremos la forma sub-muestreada en un rectángulo.

$$\begin{aligned} S_a &= PPP_{0y} + PPP_{2y} \\ S_b &= PPP_{1y} + PPP_{3y} \\ S &= PPP'_{0y} + PPP'_{2y} = PPP'_{1y} + PPP'_{3y} \end{aligned}$$

Ecuación 30. Suma de PPP iniciales y final

Para el cálculo de “S” se realiza una suma ponderada en la que se otorga mayor peso a la suma correspondiente al lado de menor PPP (de más muestras). Suponiendo que el lado de menor PPP es el lado “b”, entonces:

$$S = \frac{S_a + weight \cdot S_b}{1 + weight} \quad \text{y como } l'' = INT\left(0.5 + \frac{2 \cdot l}{S}\right)$$

Entonces también:  $S = \frac{2 \cdot l}{l''}$

Ecuación 31. Dos ecuaciones de la suma final de los PPP de cada lado

Con las dos ecuaciones de restricción rectangular y el nuevo valor de “S” calculado vamos a reajustar los PPP. Como valor de “weight” para el lado de menor PPP se ha escogido weight=2, por que le da mayor importancia al lado de mayor número de muestras, aunque sigue el lado de mayor PPP y da buenos resultados. Si no se otorgase mayor peso al lado de menor PPP, entonces un bloque que sólo contiene algo de información cerca de uno de los lados quedaría demasiado afectado por el gran PPP del lado contrario, generando un cierto emborronamiento. Es una forma de proteger un poco más las zonas de mayor detalle. De este modo si un bloque tiene un lado con más PR que otro, va a pesar más el lado de mayor PR en su geometría final.

A continuación muestro el ajuste de PPP para el lado “a” (el lado que contiene las esquinas 0 y 2). En las siguientes ecuaciones se muestra cómo hacerlo

$$\begin{aligned} \text{if } PPP_{0y} < PPP_{2y} \text{ then } & \begin{cases} PPP'_{0y} = \frac{S \cdot PPP_{0y}}{S_a} \\ PPP'_{2y} = S - PPP'_{0y} \end{cases} \\ \text{if } PPP'_{2y} > PPP_{max} \text{ then } & \begin{cases} residuo = PPP'_{2y} - PPP_{max} \\ PPP'_{2y} = PPP_{max} \\ PPP'_{0y} = PPP'_{0y} + residuo \end{cases} \end{aligned}$$

Ecuación 32. Reajuste de PPP a forma rectangular

Las ecuaciones del cálculo del gradiente del Downampling Elástico, la limitación de máxima elasticidad, así como el procedimiento de reajuste para generar formas sub-muestreadas rectangulares que permiten la cuantización con LHE son contribuciones de esta tesis

## 4.8 Downsampling Elástico separable

Para ubicar el contenido de este apartado, a continuación se muestra el diagrama de bloques de LHE avanzado, destacando el módulo en el que se ubica el contenido que se va a desarrollar.

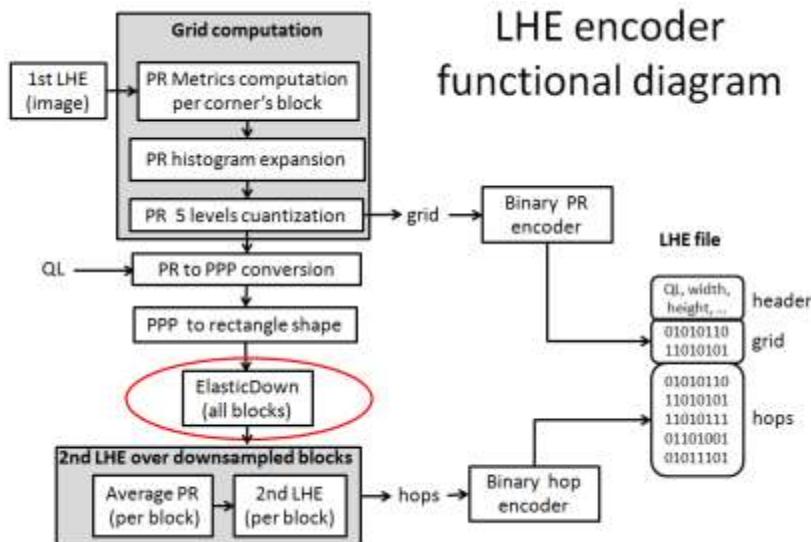


Figura 116. Ubicación del contenido que se va a desarrollar

En el diagrama de bloques mostrado, los pasos que se han dado desde la ejecución de las métricas de PR hasta el downsampling apenas consumen operaciones. Son cálculos que se aplican sobre los valores de PR de las esquinas de la malla, y no sobre los píxeles. Es decir una lista de menos de 1000 elementos ( $32 \times 32 = 1024$ ). El número de operaciones es, por lo tanto, constante con independencia del número de píxeles de la imagen por lo que la complejidad es  $O(1)$ . Si se compara esto con operaciones que se lleven a cabo con los  $N$  píxeles de la imagen podemos estar hablando de más de mil órdenes de magnitud, ya que una imagen puede fácilmente tener 1M de píxeles o más, mientras que tan solo hay 1K esquinas. Por ello todos los pasos desde la ejecución de las métricas hasta el downsampling son despreciables en cuanto a tiempo de cálculo.

Incluso el primer LHE y el cálculo de métricas como se ha visto se pueden llevar a cabo sobre una imagen reducida por “Single Pixel Selection” que reduzca al menos 4 veces los cálculos de las métricas. Ahora vamos a acometer el Downsampling Elástico, el cual involucra tantas operaciones como píxeles tenga la imagen original.

Gracias a la restricción rectangular, el Downsampling Elástico es “separable”. Una función es separable si puede expresarse como la suma de funciones de una única variable. El downsampling convencional es una función separable y conmutativa [72]. Gracias a la restricción rectangular, el downsampling elástico también es separable y conmutativo

$$\text{down}(x, y) = \text{down}(x) + \text{down}(y) = \text{down}(y) + \text{down}(x)$$

Primeramente se realiza el downsampling en la coordenada X. Esto generará una imagen comprimida únicamente en el eje x. A continuación se hace lo mismo con la coordenada Y, tomando como punto de partida la imagen escalada en horizontal y recorriendo tan solo su nueva longitud de lado horizontal, que es inferior al original. El segundo downsampling es más rápido que el primero pues aunque realiza el mismo tipo de operaciones, lo hace sobre una imagen menor a la original, ya sub-muestreada en una dimensión.

En la figura se muestran estos dos pasos, con un bloque que contiene una figura humana. Cada bloque tendrá un Downsampling Elástico diferente en función de los PPP asignados en cada una de sus esquinas

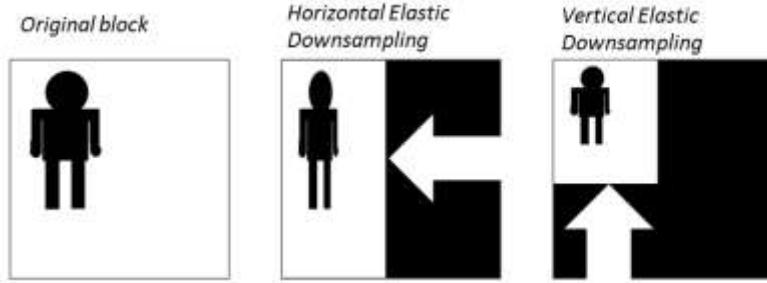


Figura 117. Downsampleig Elástico separable

La estrategia separable que se ha descrito es imposible llevarla a cabo con un Downsampleig Elástico sin restricción a forma rectangular. Imaginemos que ocurriría si tratásemos de hacerlo sin restricciones: el primer paso (el escalado H) generaría scanlines de tamaño variable y el escalado V no podría trabajar con ellas pues cada scanline vertical tendría una trayectoria completamente curvada.

Para ejecutar el downsampleig separable, se sub-muestrea primero en la coordenada X y luego en la Y (aunque daría igual hacer primero la coordenada Y, y después la X). En cada una de estas dos coordenadas se interpola el valor de PPP entre los extremos del segmento considerado, sumando en cada iteración el gradiente de PPP tal como se define en la Ecuación 28.

Los valores que toma PPP son números decimales. Esto significa que durante el proceso de downsampleig se debe considerar pixels parciales en la suma promedio de un pixel que represente a varios pixels.

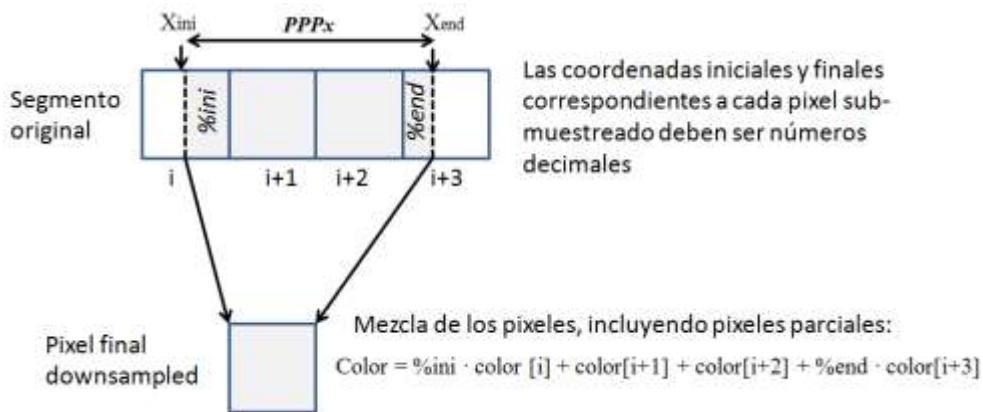


Figura 118. Cálculo de Downsampleig Elástico promedio

Puesto que tenemos los valores de PPP de las esquinas de todos los bloques de la imagen, se puede paralelizar completamente esta operación, si disponemos de suficientes procesadores, no hay ninguna limitación en cuanto a la paralelización del Downsampleig Elástico.

El siguiente pseudocódigo resume el proceso del downsampleig en coordenada X. El downsampleig en coordenada Y es idéntico salvo porque la imagen original es el resultado del primer downsampleig y por lo tanto cambia la longitud  $l$  del bloque de partida, que se ha transformado en  $l''$ . Esto se debe tener en cuenta en el cálculo de los gradientes que se efectúan antes del primer bucle “for”.

```

 $PPPxa=PPPx[0]$ 
 $PPPxb=PPPx[1]$ 
 $grad\_PPPxa= (PPPx[2]-PPPx[0])/l \text{ // } l \text{ is the length of the source block to be downsampled}$ 
 $grad\_PPPxa= (PPPx[3]-PPPx[1])/l \text{ // } l \text{ is the length of the source block to be downsampled}$ 
for  $y=block\_yini$  to  $block\_yfin$ 
     $grad\_ppp=pppxb-pppxa/(l''-1)$ 
    for  $x'=block\_xini$  to  $block\_xini+l''$ 
        //addition of initial %
         $porcent= xini-(int)xini;$ 
         $color+=image[x,y] \times porcent$ 
        for  $x=x\_ini+1$  to  $INT(x\_xtart+pppx)$ 
             $color+=image[x,y]$ 
        next  $x$ 
        //addition of last %
         $porcent= x\_xtart+pppx- (int)( x\_xtart+pppx)$ 
         $color+=image[x,y] \times porcent$ 
         $color=color /pppx$ 
         $pppx=pppx+grad\_ppp$ 
         $xini=xini+pppx$ 
    next  $x'$ 
     $PPPxa=pppxa+grad\_PPPxa$ 
     $PPPxb=pppxa+grad\_PPPxb$ 
next  $y$ 

```

La restricción rectangular es una aportación de esta tesis que permite la ejecución separable del algoritmo.

Antes de pasar al último paso (cuantización LHE de los bloques sub-muestreados) se va a analizar el resultado obtenido comparado con un downsampling homogéneo con igual número de muestras. Este test ha sido realizado con la base de datos de imágenes de Kodak [61] (24 imágenes), usando interpolación bicúbica de las muestras en ambos tipos de downsampling.

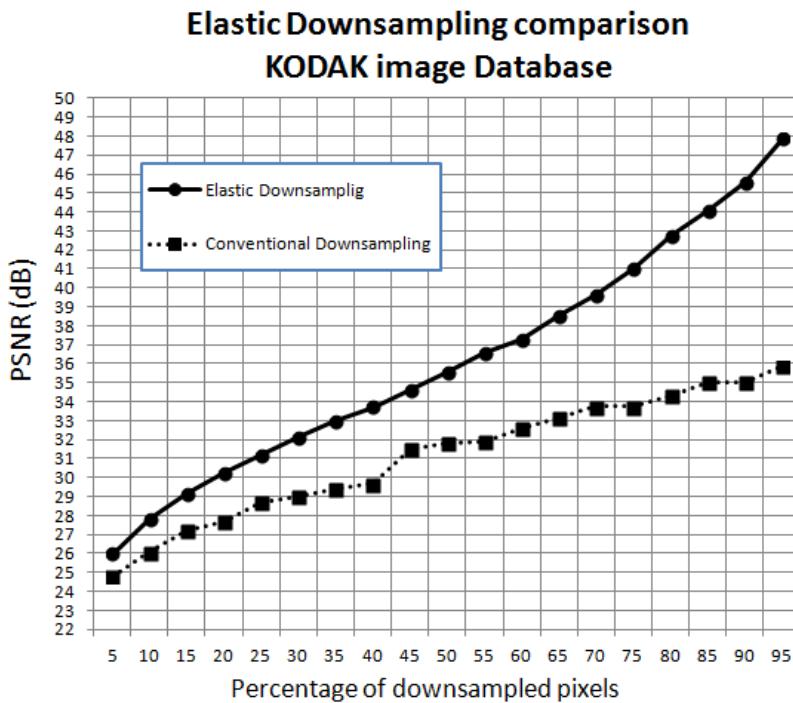


Figura 119. Comparación de DownSampling Elástico vs DownSampling homogéneo

La gráfica demuestra que el DownSampling Elástico representa una mejora notable en comparación con el downampling homogéneo convencional.

Lo que importa en esta gráfica es la comparativa, no tanto el valor de PSNR alcanzado. Las dos gráficas poseen dos puntos en común: el punto de máxima compresión (4 muestras por bloque, algo más del 1%) y el punto de compresión nula (el cual no ha sido representado, sería el 100% de porcentaje de pixeles sub-muestreados). Al 95% de muestreado, el downsampling homogéneo tiene una caída dramática de PSNR respecto del DownSampling Elástico. Esto es debido a que el Downsampling Elástico protege todo aquello que tenga mayor relevancia perceptual.

A bajos porcentajes de muestras, el PSNR de ambos tipos de sub-muestreado se hace más similar. Sin embargo, subjetivamente las diferencias son muy relevantes, tal como se muestra en detalles de los siguientes ejemplos calculados al 10% de porcentaje de muestras.



Original image: Kodim05



Conventional  
(detail)



Elastic downsampling



Elastic  
downsampling  
(detail)



Original image: Kodim09



Elastic downsampling



Conventional (detail)



Elastic downsampling (detail)



*Original image: Kodim20*



*Conventional (detail)*



*Elastic downsampling*



*Elastic downsampling (detail)*

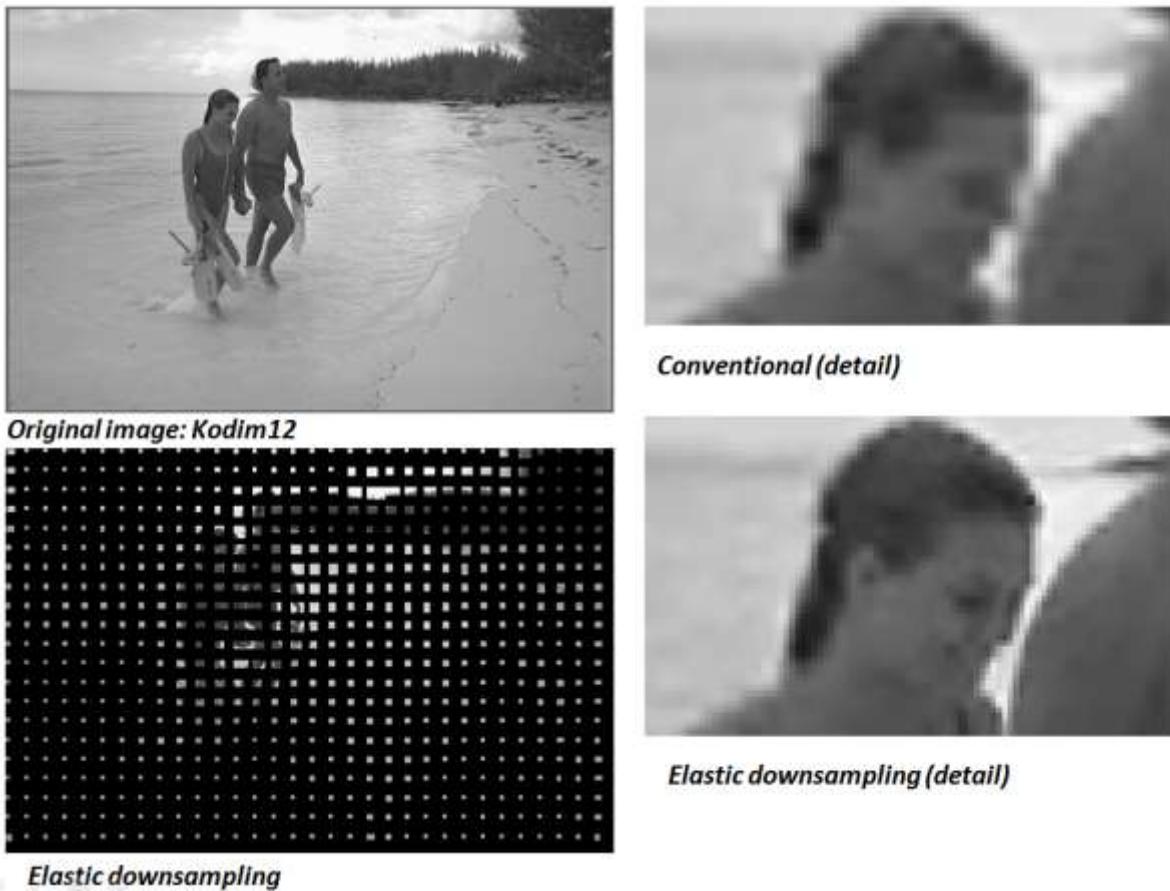


Figura 120. Diversas imágenes muestreadas al 10%

En cuanto a los tiempos de ejecución, el downsampling elástico (al igual que lo es el downsampling convencional) ha resultado ser estrictamente lineal tal como era esperable y como refleja la siguiente gráfica:

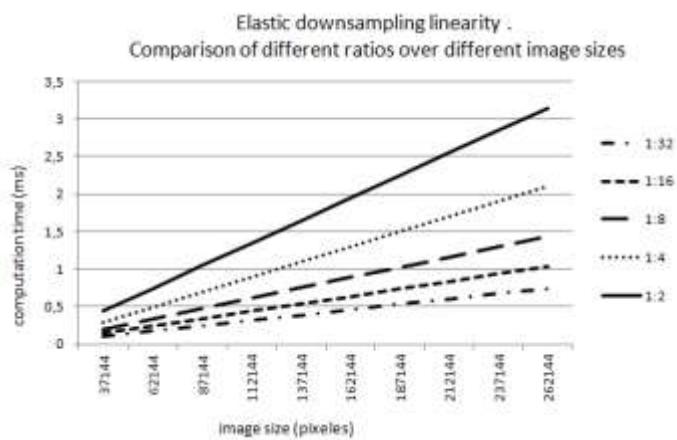


Figura 121. Tiempos de ejecución lineales

Para una imagen de tamaño  $N$ , los tiempos de sub-muestreado disminuyen a medida que sub-

muestreamos menos. Si se realiza el doble de muestras, se consume un solo un 50% más de tiempo y no el doble de tiempo. Esto es debido a la separabilidad del downsampling. El downsampling en la coordenada X deja una imagen reducida para ser muestreada en la coordenada Y. Si se sub-muestrea la mitad en X y la mitad en Y, entonces haremos la mitad de operaciones en cada dimensión, pero habremos reducido 4 veces el tamaño de la imagen resultante. La gráfica revela esta circunstancia

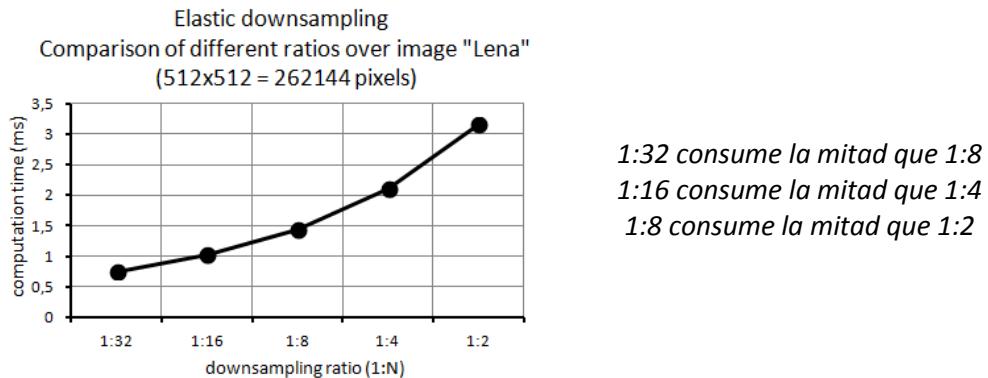


Figura 122. Tiempos de ejecución a diferentes ratios

## 4.9 Imágenes en color

Cuando trabajemos con imágenes en color, no será necesario ejecutar todo el proceso de nuevo (primer LHE, métricas, cálculo de PPP) con las señales de crominancia. Los modelos de color YUV444, YUV422 y YUV420 establecen una relación entre la cantidad de muestras de luminancia y las de crominancia, por lo que simplemente podemos aplicar estos modelos para los bloques ya sub-muestreados en luminancia.

La elección del modelo de color YUV ya se ha justificado en el capítulo 3, pero en LHE avanzado se añade la justificación de eficiencia computacional pues no se requiere ejecutar ningún paso con las señales de crominancia hasta llegar al downsampling. Las métricas de luminancia (más importantes para el ojo humano) son las que definirán los PPP de luminancia y por consiguiente, las de crominancia se derivan de forma inmediata en los modelos YUV 422 y YUV420.

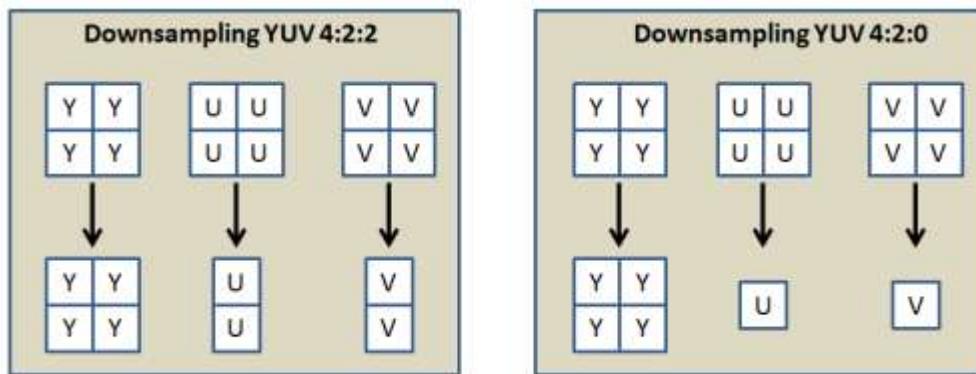


Figura 123. Escalados en crominancia

El modelo de color será uno de los parámetros que almacenaremos en la cabecera del fichero LHE y podrá tomar 4 valores: B/N, YUV444, YUV422 y YUV420

El tratamiento que daremos a cada uno de los modelos de color es el siguiente:

**YUV444:** puesto que se utilizan el mismo número de muestras para las señales de luminancia y crominancia, los bloques sub-muestreados de las señales U y V tendrán las mismas dimensiones que los bloques de luminancia. El proceso de downsampling para estas señales será, por lo tanto, idéntico al de la luminancia.

**YUV422:** en este caso el número de muestras de crominancia es la mitad que la de la luminancia en la coordenada x, mientras que en la coordenada Y se mantiene el número de muestras.

Lo que se hace por tanto es asignar a los bloques sub-muestreados de las señales de crominancia la misma longitud en Y que a los de luminancia y la mitad en la coordenada X, no pudiendo en ningún caso dar lugar a un lado con  $l''_x < 2$ , ya que en ese caso estaríamos asignando un  $PPP$  superior a  $PPP_{max}$

**YUV420:** en este caso el número de muestras de crominancia es la mitad que la de la luminancia tanto en la coordenada X como en la coordenada Y.

Lo que se hace por tanto es asignar a los bloques sub-muestreados de las señales de crominancia la mitad de la longitud en ambas coordenadas que a los bloques de luminancia, no pudiendo en ningún caso dar lugar a un lado con  $l''_x < 2$ , ya que en ese caso estaríamos asignando un  $PPP$  superior a  $PPP_{max}$

La validación de los modelos de color con LHE ha sido llevada a cabo con la primera estrategia de downsampling de tres niveles de malla, por lo que en principio son perfectamente válidos para el downsampling elástico, ya que aunque es más flexible y proporciona más calidad, se basa igualmente en diferentes intensidades de escalado en función de la zona de la imagen y la interpolación y unión de todas ellas.

## 4.10 Segunda cuantización LHE

Para ubicar el contenido de este apartado, a continuación se muestra el diagrama de bloques de LHE avanzado, destacando el módulo en el que se ubica el contenido que se va a desarrollar.

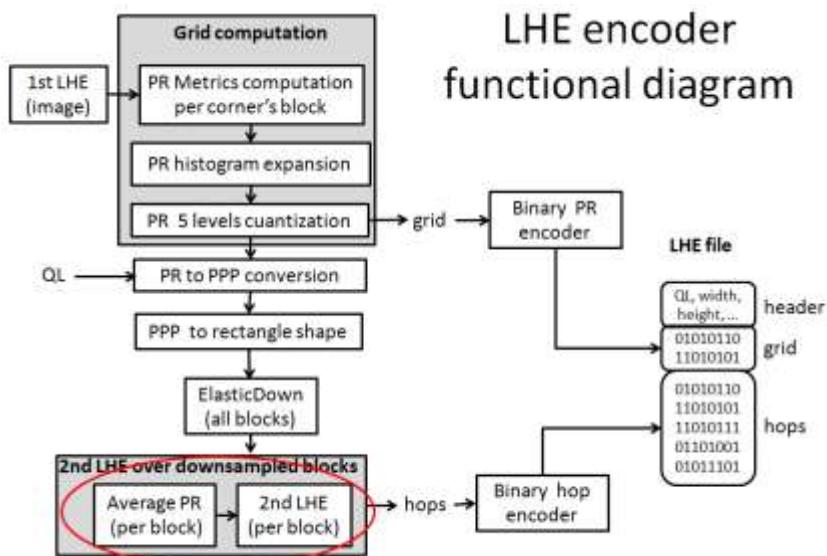


Figura 124. Ubicación del contenido que se va a desarrollar

Lo primero que se va a llevar a cabo es el cálculo de la razón geométrica óptima y el hop  $h1_{max}$  óptimo

para el cálculo de hops en cada bloque. Esto es posible hacerlo porque a diferencia del LHE básico, en LHE avanzado se llega a esta etapa con las métricas de PR y se conoce si un bloque es suave o abrupto.

Para ello se calcula la media de los valores cuantizados  $PR'_x$  y  $PR'_y$  de todas las esquinas. Esto se ha de hacer con los valores cuantizados porque el decodificador no dispone de los valores originales, sino de los cuantizados y el decodificador deberá poder calcular esta razón geométrica sin necesidad de que deba ser almacenada en el fichero comprimido.

La media de la métrica  $PR'_{avg}$  permite saber si estamos ante un bloque abrupto o uno suave y en función de su valor se puede adaptar los valores óptimos de la razón geométrica que relaciona los hops  $r_{opt}$  y el hop  $h1_{max}$ . Estos valores están ajustados de forma heurística, probando con la base de datos de imágenes de kodak. Lógicamente, a mayor  $PR'_{avg}$ , menor  $r_{opt}$  y mayor  $h1_{max}$

$$PR'_{avg} = \frac{\sum_0^3 PR'_x(i) + \sum_0^3 PR'_y(i)}{8}$$

$PR'_{avg}$	$r_{opt}$	$h1_{max}$
1.0	30	16
[0.75,1.0)	25	14
[0.5,0.75)	25	12
[0.25,0.5)	22	10
[0,0.25)	20	8

Tabla 13. Valores óptimos de la razón geométrica y primer hop en función de la PR promedio

Para el primer hop del bloque a  $h_1$  se le asigna un valor intermedio entre su valor mínimo (su valor mínimo siempre es 4) y su valor máximo:

$$h_1 = \frac{4 + h1_{max}}{2}$$

Ecuación 33. Valor inicial de  $h1$  para cada bloque

El valor de  $h_1$  fluctuará en el rango  $[0, h1_{max}]$  siguiendo el algoritmo presentado en el capítulo sobre LHE básico, y que modela el proceso de acomodación del ojo humano.

Para la cuantización, se podría considerar cada bloque sub-muestreado como una imagen y aplicar un LHE básico con la razón y  $h1$  calculados, sin embargo proporciona más calidad considerar las fronteras de cada bloque en la estimación del color de fondo del algoritmo LHE. Otro motivo para considerar las fronteras es que si se considera cada bloque como una imagen entonces se debería guardar el valor del primer pixel (esquina izquierda superior) ya que carece de referencias. En el caso de tener una imagen cuadrada de 32x 32 bloques, esto supone  $32 \times 32 = 1024$  bytes adicionales.

Para estimar el color de fondo (el  $hop_0$ ) de los píxeles situados en el borde de un bloque, se requiere la referencia de otros píxeles que no se encuentran en el bloque, y cuyo valor deberemos calcular.

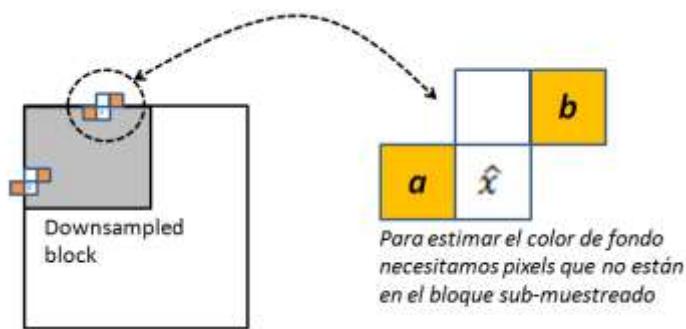
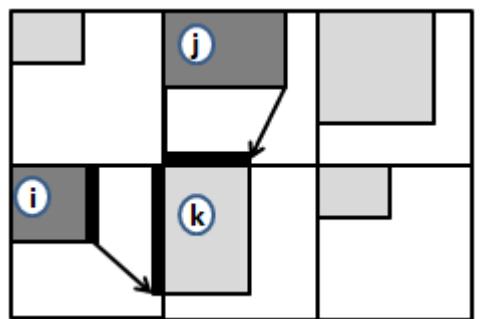


Figura 125. Pixels frontera necesarios para estimar el color de fondo

Para codificar los pixels frontera de un bloque es necesario hacer una interpolación de la última scanline del bloque frontera (izquierdo o superior) al tamaño del lado del nuevo bloque sub-muestreado  $l''$ , ya que todos los bloques se encuentran sub-muestreados. Para realizar esa interpolación se parte de los bloques frontera sub-muestreados y cuantizados con LHE, y no se usa la imagen original en ningún caso, pues el proceso debe ser reproducible en el decodificador.



*Para codificar el bloque "k" primero se debe interpolar las luminancias de la frontera vertical del bloque "i" y las de la frontera horizontal del bloque "j"*

Figura 126. Dependencias entre bloques

En la siguiente figura se muestran los pasos para tener las fronteras necesarias en cada bloque antes de codificarlo. El ejemplo es el de una serie de bloques situados en el lado superior de la imagen. En caso de tratarse de un bloque cualquiera, antes de codificarlo se debe tener la frontera izquierda y la frontera superior (el siguiente ejemplo muestra el caso de frontera izquierda)

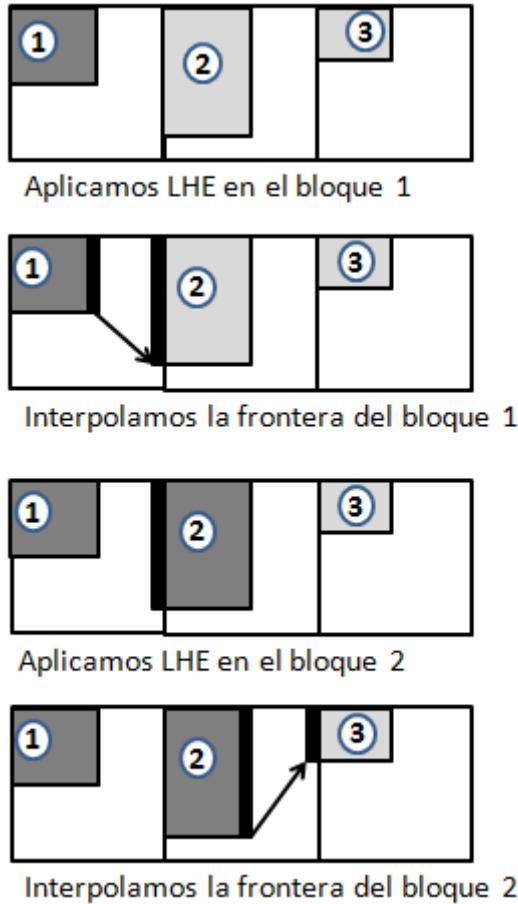


Figura 127. Secuencia de pasos para codificar por LHE los bloques

La interpolación solo es necesaria realizarla en la última scanline del bloque (su frontera) y no en todo el bloque, por lo que es un procedimiento muy rápido. Además, se puede realizar mediante la técnica de interpolación de vecino cercano, la más simple y rápida, sin perjuicio de la calidad. **Se ha comprobado experimentalmente que realizar una interpolación lineal de la frontera no mejora la calidad respecto de hacerlo por vecino cercano.** Se utiliza como parte de la referencia para el color de fondo del siguiente hop y no como valor de la componente de color de ningún pixel, por eso una interpolación por vecino es suficiente.

Esta es una interpolación elástica, es decir, debe tener en cuenta que los PPP evolucionan y por lo tanto no es válido simplemente “comprimir” o “estirar” la frontera del bloque. Hay que adaptar los PPP mediante una interpolación elástica. La forma de realizar interpolaciones elásticas se explica en el capítulo dedicado al decodificador. La siguiente figura ilustra el problema. La solución no consiste en estirar la frontera de un bloque hasta la nueva longitud, sino adaptar su contenido, los valores de los pixels que contiene, a los PPP del bloque adyacente.

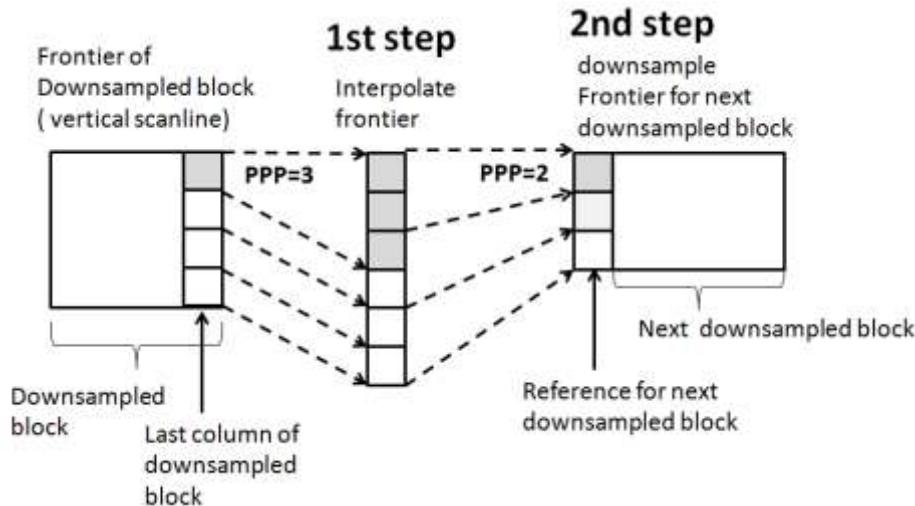


Figura 128. Interpolación es adaptar longitud pero también adaptar los PPP

Una forma de implementar la interpolación de fronteras (con el cambio de PPP que implica) es dividirlo en dos pasos: Tras codificar un bloque, interpolaremos sus fronteras hasta la longitud del lado del bloque. Y a continuación, sub-muestrearemos sus fronteras interpoladas hasta el tamaño del bloque adyacente sub-muestreado, teniendo en cuenta los nuevos PPP.

La interpolación de fronteras impide la posibilidad de codificar con LHE todos los bloques simultáneamente pero tenemos la posibilidad de paralelizar muchos bloques. En la siguiente figura aparecen representados los bloques que se pueden codificar simultáneamente etiquetados con el mismo número

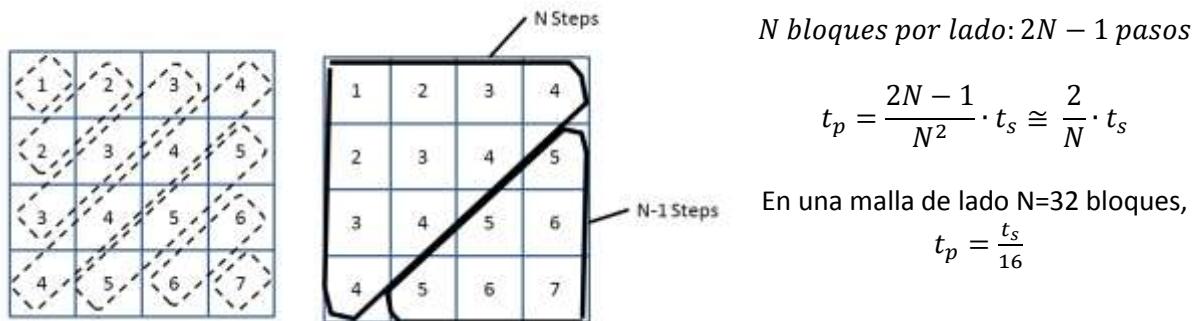


Figura 129. Paralelización del segundo LHE sobre los bloques sub-muestreados

Donde  $t_p$  es el tiempo invertido con la máxima paralelización posible y  $t_s$  es el tiempo invertido mediante un procesamiento secuencial. Para una malla de N=32 bloques, la capacidad de paralelización de LHE permite reducir los tiempos de ejecución secuenciales 16 veces. Por ejemplo, si un bloque se codifica en 1ms (en la práctica se tarda mucho menos) y hay 32x32=1024 bloques, secuencialmente tardaríamos 1024 ms, pero en paralelo se puede realizar en 1024/16 =64ms.

A pesar de estos cálculos, el algoritmo permite paralelización total de todos los bloques si no consideramos interpolación de fronteras. De no hacer esta interpolación de fronteras de bloques, la calidad resultante disminuye algo. Se pueden procesar todos los bloques simultáneamente si usamos las referencias del propio bloque, como al estimar el color de fondo en los límites de la imagen. Considerar

fronteras tiene un coste adicional en operaciones muy bajo y su única penalización es la imposibilidad de paralelizar todos los bloques a la vez, aunque aun permite un gran nivel de paralelización. A cambio conseguimos más calidad.

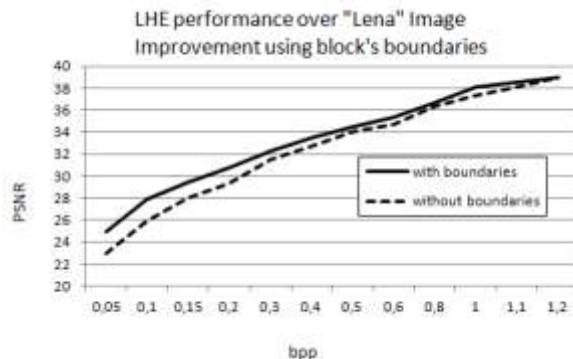


Figura 130. La calidad sin considerar fronteras es menor

El ajuste de la razón geométrica y del máximo valor de  $h_1$  a partir de las métricas cuantizadas de relevancia perceptual de cada bloque es una contribución de esta tesis y permite reproducir con más fidelidad los valores originales de las componentes de color de cada bloque de la imagen de forma individualizada.

El procedimiento de interpolación de fronteras de bloques y paralelización de la codificación LHE en cada bloque es una aportación de esta tesis y forma parte del algoritmo LHE avanzado.

## 4.11 Codificadores binarios

LHE avanzado requiere dos codificadores binarios. Uno para codificar los hops resultantes de la cuantización logarítmica de los bloques sub-muestreados elásticamente y otro para codificar los valores de la métrica PR cuantizados.

Para ubicar el contenido de este apartado, a continuación se muestra el diagrama de bloques de LHE avanzado, destacando los módulos en el que se ubica el contenido que se va a desarrollar.

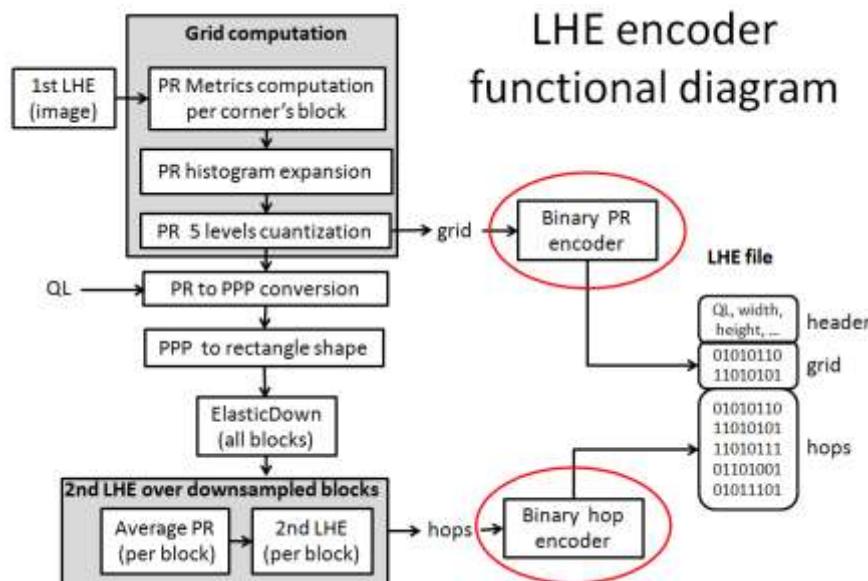


Figura 131. Ubicación del contenido que se va a desarrollar

Aunque son diferentes, ambos codificadores binarios se fundamentan en las mismas dos técnicas:

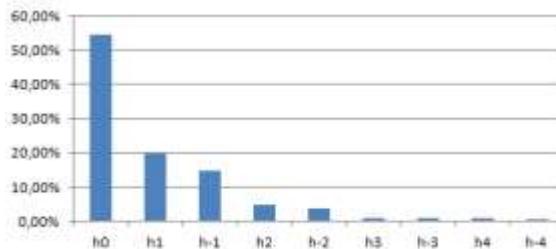
- Aprovechamiento de las redundancias espaciales
- Codificación de símbolos con longitud variable

#### 4.11.1 Codificador binario de hops

Al igual que ocurre con LHE básico, tras el downsampling Elástico se ha constatado que la cuantización logarítmica de los bloques sub-muestreados resultantes genera una distribución estadística de hops en la que los hops más pequeños son mayoritarios. Cuando el downsampling es muy intenso, la prominencia de los hops pequeños no es tan acusada, pero sigue siendo muy significativa.



Hops distribution for "lena" image  
(100% of samples, QL=99)



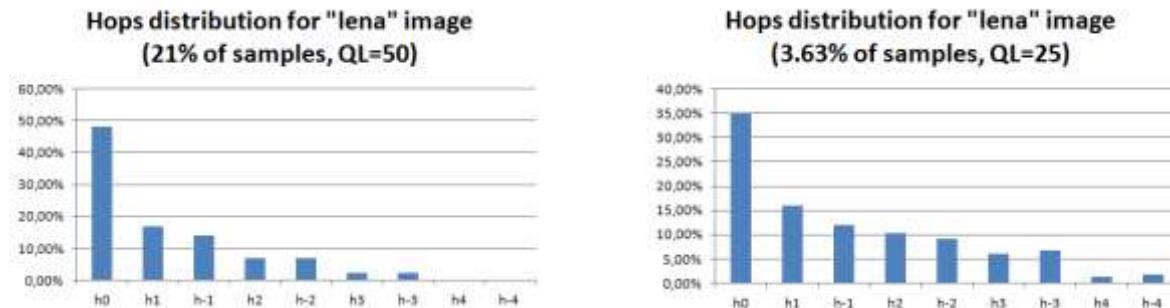


Figura 132. Distribución estadística de hops tras LHE sobre los bloques sub-muestreados

Esta distribución con tan alta permite basar la codificación en símbolos de longitud variable, al igual que hicimos con LHE básico. En la siguiente figura se representan los componentes del codificador binario de hops.

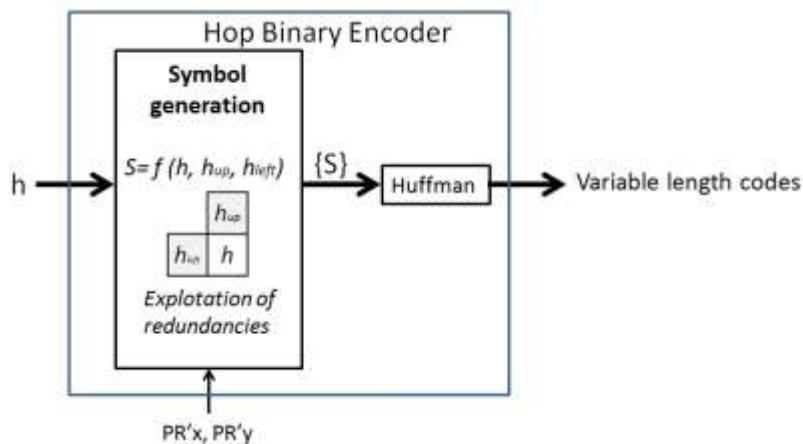


Figura 133. Las dos etapas del codificador binario de hops

Primeramente aprovechando las redundancias espaciales se transforman los hops en símbolos y posteriormente se signa un código binario a cada símbolo, una vez que se calcula la tabla óptima de códigos de longitud variable usando Huffman estático (se podría usar Huffman adaptativo pero por simplicidad y eficiencia se opta por el estático). Dicha tabla es global a toda la imagen y se debe almacenar en el fichero comprimido, aunque al tratarse de sólo 9 símbolos se trata de unos pocos bytes. Para transformar un hop ( $h$ ) en un símbolo ( $S$ ) vamos a usar la siguiente tabla:

$h$	$S$
$h_0/up$	1
$h_1$	2
$h_{-1}$	3
$h_2$	4
$h_{-2}$	5
$h_3$	6
$h_{-3}$	7
$h_4$	8
$h_{-4}$	9

Tabla 14. Correspondencia entre hops y símbolos

La estrategia consiste en definir el significado de un símbolo “n” como la negación de todos los símbolos inferiores desde “1” hasta “n-1”

El orden en el que se descartan los hops es lo que determinará lo grande o pequeño que será nuestro símbolo. Cada hop descartado significará sumar 1 al símbolo, por lo que si se acierta en la primera predicción, se obtendrá un símbolo “1”. Si acertamos en la segunda predicción el símbolo será “2” y así sucesivamente. Es decir, el símbolo es el intento en el que se ha predicho correctamente el hop.

En cuanto a las redundancias espaciales encontradas, a diferencia de LHE básico ahora se dispone de las métricas PR del bloque y se pueden usar para optimizar un poco más la predicción del hop

En bloques con mucha información horizontal, va a ser más frecuente encontrarnos con el hop “up” (el localizado justo encima) que con el hop h0, por ello se debe seguir esta estrategia:

Al empezar a codificar un bloque, compararemos  $PR'x_{avg}$  con  $PR'y_{avg}$

$$PR'x_{avg} = \frac{\sum_0^3 PR'_x(i)}{4}, \quad PR'y_{avg} = \frac{\sum_0^3 PR'_y(i)}{4}$$

Empezamos asignando S=1.

- si  $PR'x_{avg} \leq PR'y_{avg}$  se comprueba primero si el hop a codificar es h0 (redundancia horizontal) y si no coincide entonces se suma 1 a S y se comprueba con “Up”. Si tampoco coincide sumaremos otro 1
- si  $PR'x_{avg} > PR'y_{avg}$  se comprueba primero si el hop a codificar es “up” (redundancia vertical) y si no coincide entonces sumamos 1 a S y a continuación se comprueba si es h0. Si tampoco coincide entonces se suma otro 1

Si no se tiene éxito en la predicción del hop, entonces se deberá codificar sumando el número correspondiente de la tabla de correspondencia entre hops y símbolos. Si los hops ya negados son inferiores al símbolo en la tabla, entonces no hace falta repetir esas negaciones, de modo que se resta el número de hops inferiores ya negados a dicho número.

Tras codificar el primer hop del bloque pasamos al segundo hop y el nuevo orden de predicción es el siguiente:

- Si en el hop anterior se ha encontrado una redundancia horizontal (h0) entonces se comprobará h0 en primer lugar.
- Si en el hop anterior se ha encontrado una redundancia vertical (“up”) entonces se comprobará “up” en primer lugar
- Si en el hop anterior no se ha encontrado ninguna redundancia, entonces se comprobará en el orden determinado por  $PR'x_{avg}$  y  $PR'y_{avg}$ , al igual que se hizo con el primer hop

Al igual que en LHE básico, existe una optimización adicional que se puede realizar, basada en la siguiente característica: Tras un borde, los tonos tienden a suavizarse, pues la textura de los objetos suele ser suave.

Esto en LHE se traduce en que tras un  $h_i > h_1$ , aumenta la probabilidad de encontrarnos con un hop más pequeño de igual signo. Lo que se hará será simplemente priorizar en esos casos (tras haber

comprobado  $h_0$  y “up”) el hop inmediatamente inferior (o superior en caso de ser un hop negativo). Si falla esta predicción, se descarta el hop y para ello se penaliza sumando un 1 a S.

Esta última optimización produce mejoras muy pequeñas, del orden de 0.02bpp, de modo que la complejidad que introduce es cuestionable frente al beneficio que proporciona.

Tras esta conversión de hops a símbolos que aprovecha las redundancias espaciales, la distribución estadística de símbolos es más acusada en símbolos de pequeños de lo que lo eran los hops pequeños.

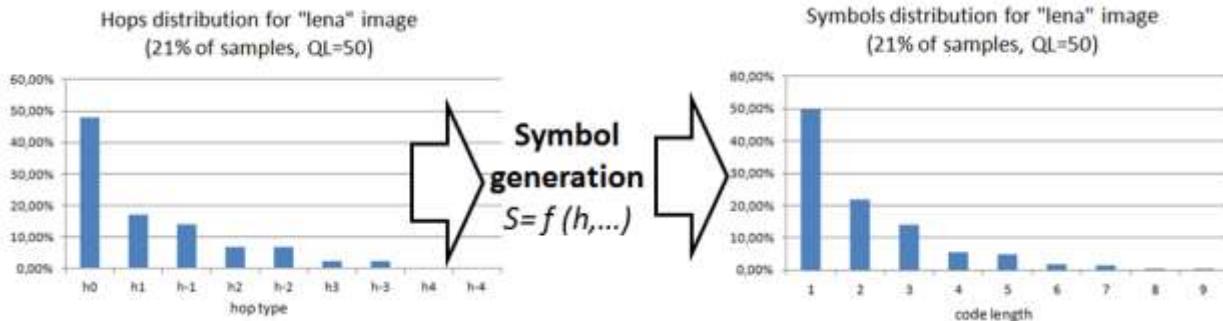


Figura 134. Ejemplo de transformación de hops en símbolos

Por último se determinarán los códigos Huffman basados en el análisis del número de apariciones de cada símbolo (Huffman estático).

La estrategia del codificador binario de hops, que transforma hops en símbolos aprovechando las redundancias espaciales y posteriormente aplica una tabla de códigos Huffman es una contribución de esta tesis y forma parte del algoritmo LHE avanzado

#### 4.11.2 Codificador binario de datos de malla

El codificador binario de datos de malla tiene el propósito de codificar en binario los valores cuantizados de la métrica PR correspondientes a las esquinas de los bloques. Se han definido 5 niveles de cuantización de modo que solo hay 5 valores posibles.

La malla consta de tantos puntos como vértices tienen los bloques.

El número de puntos es  $(N + 1) \cdot (M + 1)$ , siendo N el número de bloques de ancho (N=32) y M el número de bloques de alto. Por cada punto se tiene que codificar el valor del cuadro PRx y el de PRy. En la siguiente figura se ha representado una malla de 4 bloques de ancho destacando sus puntos.

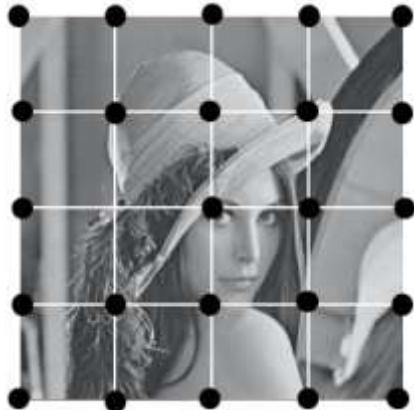


Figura 135. Puntos de la malla a codificar

La métrica de PR posee redundancias espaciales. Esto significa que bloques adyacentes van a tener un valor de la métrica similar. En las imágenes sub-muestreadas de la Figura 120 se puede visualizar como los bloques adyacentes poseen tamaño de bloque sub-muestreado similar. Esto es debido a la redundancia espacial de los valores de la métrica PR de una imagen que se transforman en valores de PPP similares.

Por este motivo, el codificador binario de datos de malla tiene una estructura idéntica al codificador binario de hops. En una primera etapa se transforman los cuantos de PR en símbolos, aprovechando las redundancias espaciales. En una segunda etapa se codifican usando códigos Huffman de longitud variable.

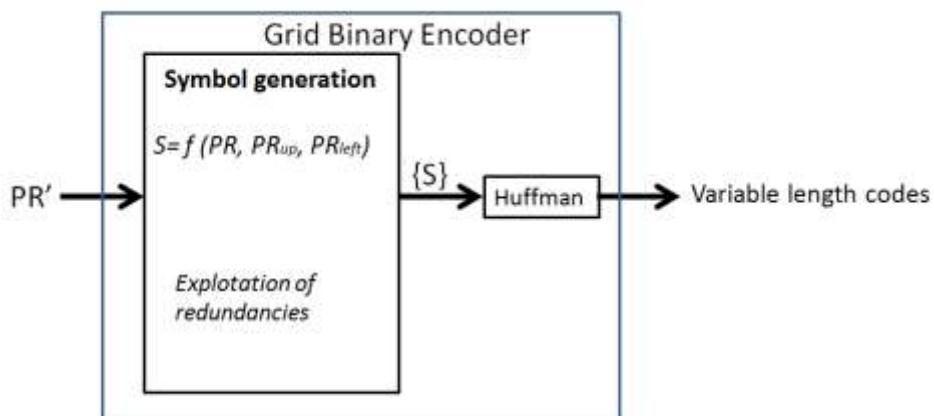


Figura 136. Las dos etapas del codificador binario de malla

Se ordenan los valores de los cuantos desde el más frecuente al menos frecuente y se asigna un número (del 0 al 4). A continuación se crea una tabla de símbolos que van a aprovechar las redundancias espaciales. Cada símbolo tendrá asociado un código Huffman en función de su frecuencia de aparición.

Esta tabla de traducción de símbolos a códigos se debe almacenar en la cabecera del fichero comprimido. En el caso de los cuantos de la malla los cuantos más frecuentes suelen también ser los más pequeños pero puede haber algo de variación, sobre todo en imágenes con fondos lisos donde el cuanto más frecuente es el de PR=0, muy infrecuente en imágenes sin fondos lisos.

*Ejemplo de tabla de cuantos ordenada según frecuencia de aparición*

Cuento de PR (5 posibles cuantos)	orden
PR=0 (cuanto=0)	1
PR=0.125 (cuanto=1)	0
PR=0.25 (cuanto=2)	2
PR=0.5 (cuanto=3)	4
PR=1 (cuanto=4)	3

*Ejemplo de tabla de símbolos que almacenaremos en la cabecera*

Símbolo (5 posibles símbolos)	Código Huffman (ejemplo)
left/up/Más frecuente(=1)	1
0	01
2	001
4	0001
Menos frecuente(=3)	0000

**Tabla 15. Tabla de símbolos para los cuantos**

La estrategia será muy similar a la asignación de símbolos a hops. Cada símbolo va a significar el descarte de los cuantos asociados a los símbolos inferiores. Cuanto mejor sea nuestra predicción, más pequeños serán los símbolos.

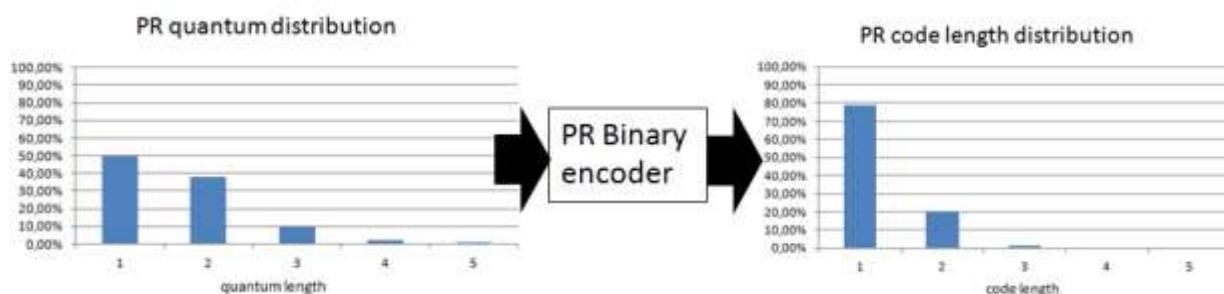
A continuación se recorre la malla de valores PRx y se realiza la predicción del cuanto considerando el siguiente orden:

1. Cuanto situado a la izquierda (si existe)
2. Cuanto situado arriba (si existe y no es igual al cuanto situado a la izquierda)
3. Cuanto más frecuente (si no es ninguno de los dos anteriores)
4. Resto de cuantos ordenados por frecuencia de aparición

En caso de no acertar en ninguno de los tres primeros casos, habremos acumulado un 1 por cada símbolo fallido, es decir, en total un “3”. Despues usaremos la

Tabla 15 para asignar el símbolo, al cual restaremos tantos unos como símbolos ya descartados y considerados en el “3”. De este modo no negaremos dos veces el mismo símbolo. Para los valores de PRy haremos el mismo proceso, considerando el mismo orden de cuantos en la predicción

El valor de los cuantos es independiente del nivel de calidad (QL) o del bit-rate (bpp) empleado en la compresión, de modo que la malla posee exactamente la misma información para compresiones intensas que para compresiones leves. En la siguiente figura se muestra el efecto del cuantizador binario de PR para la imagen “Lena”, cuya malla de 1024 bloques (1089 puntos) se codifica en 2663 bits



**Figura 137. Distribución de longitud de códigos tras el codificador binario**

La estrategia del codificador binario de PR, fundamentada en sus redundancias espaciales es una contribución de esta tesis y forma parte del algoritmo LHE avanzado

## 4.12 Complejidad y Paralelización

En la siguiente figura se ha dividido el diagrama de bloques de LHE avanzado en varios macro-bloques funcionales en función de su complejidad algorítmica.

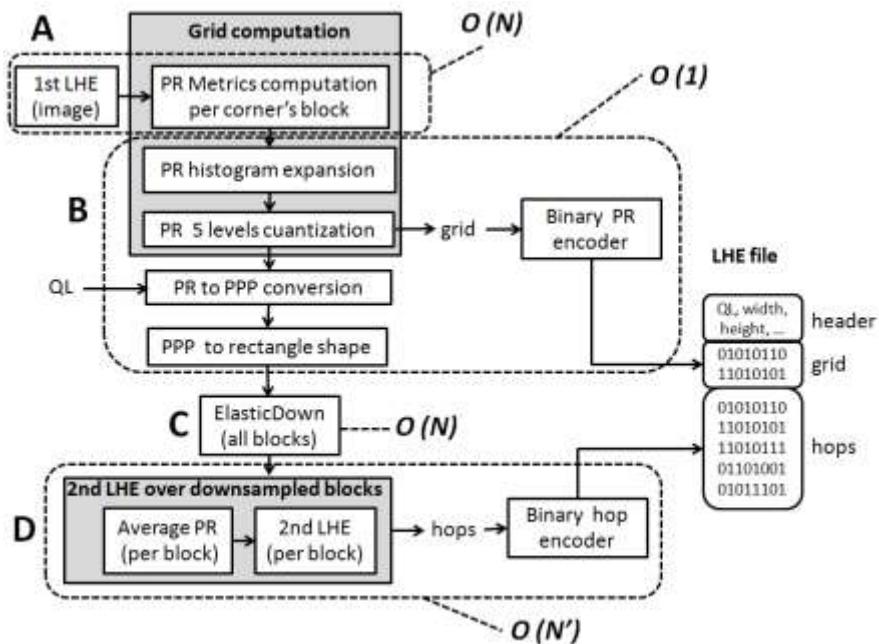


Figura 138. Macrobloques de LHE avanzado en función de su complejidad algorítmica

- **Primer LHE básico y cálculo de métricas:** para una imagen de  $N$  pixeles implica una complejidad  $O(N)$  ya que cada pixel requiere ser procesado para ser transformado en un hop. Este macro-bloque funcional se puede ejecutar en paralelo si dividimos la imagen en los bloques que posteriormente van a ser sub-muestreados por separado. Cada bloque puede ser codificado en LHE por separado como si se tratase de una imagen individual y el resultado en métricas es idéntico. La máxima paralelización implica 2 pasos para cada bloque: LHE y cálculo de métricas. Una optimización factible es llevar a cabo este proceso con una imagen sub-muestreada por “Simple Pixel Selection”, de un cuarto del tamaño original. Los resultados son idénticos.
- **Las operaciones sobre las métricas:** para una malla de 32 bloques de ancho lleva un número de operaciones constante con independencia de la resolución de la imagen original. En los pasos de este macro-bloque funcional tendremos por lo tanto una complejidad  $O(1)$ . La paralelización de este macro-bloque es plena, pudiéndose calcular todos los bloques en un solo paso y codificando en binario en un segundo paso
- **El downsampling elástico** involucra tantas operaciones como píxeles tenga la imagen original. La complejidad de este bloque es  $O(N)$ . En cuanto a su posible paralelización, es completa: todos los bloques se pueden sub-muestrear en paralelo en un solo paso

- **El segundo LHE sobre los bloques sub-muestreados** es una operación de coste computacional  $O(N')$ , siendo  $N'$  el numero de muestras de los bloques resultantes del proceso de downsampling. Ello significa que la complejidad es inferior que si se tratase de  $N$ , pudiendo reducirse en un orden de magnitud para compresiones al 10% de muestras. En cuanto a la paralelización de este macro-bloque funcional, tenemos dos opciones:
  - Una de peor calidad, en la que podemos paralelizar todos los bloques a la vez (1 solo paso)
  - y otra que mejora la calidad considerando fronteras pero involucrando un numero de pasos  $2K+1$ , siendo  $K$  el ancho en bloques de la imagen ( que hemos establecido en  $K=32$ ).

En ambos casos habría que dar un último paso adicional para transformar los hops en códigos binarios.

Macrobloque	complejidad	Pasos con máxima paralelización
A	$O(N)$	2
B	$O(1)$	2
C	$O(N)$	1
D	$O(N')$	Dos opciones: $1+1=2$ $(2K+1)+1=2k+2$

Tabla 16. Complejidad y paralelización de cada macrobloque funcional de LHE avanzado

A partir de esta tabla tenemos:

- La posibilidad más rápida de peor calidad: 7 pasos
- La posibilidad más lenta de mejor calidad:  $6+2K = 6+64 = 70$  pasos

Es decir, un orden de magnitud de diferencia en tiempo de cálculo si se dispone de los suficientes recursos para paralelizar al máximo.

### 4.13 Resumen del capítulo

En este capítulo se ha presentado el codificador LHE avanzado. El objetivo de LHE avanzado es superar las limitaciones de LHE básico, en concreto la de ofrecer la capacidad de escoger el grado de compresión deseado.

Para conseguir este objetivo he creado la técnica del downsampling elástico, la cual es una contribución de esta tesis. Esta técnica requiere de unas métricas de Relevancia perceptual, cuya definición y fundamentos teóricos son contribuciones de esta tesis.

La definición completa del algoritmo LHE avanzado es una contribución de esta tesis e incluye las siguientes contribuciones:

- Definición de métrica de relevancia perceptual independiente de la resolución
- Concepto del Downsampling elástico
- Malla de un solo nivel con número de bloques fijos independiente de la resolución
- Conversión de métricas de relevancia perceptual en PPP
- Saturación y expansión de la métrica PR
- Cuantización geométrica de la relevancia perceptual
- Nivel de calidad (QL) y calidad equivalente
- Ecuaciones de downsampling elástico y restricción rectangular
- Downsampling elástico separable
- Segundo LHE con configuración de razón geométrica adaptada a las métricas y tratamiento de fronteras
- Codificador binario de hops con explotación de las redundancias espaciales de los bloques sub-muestreados y asignación de tabla de códigos binarios de longitud variable
- Codificador binario de métricas de PR cuantizadas que aprovecha las redundancias espaciales de la métrica y asignación de tabla de códigos binarios de longitud variable

# Capítulo 5

## Decodificador LHE avanzado

### 5.1 Presentación del capítulo

En el capítulo anterior se ha descrito el algoritmo del codificador LHE avanzado, y este capítulo se va a describir el complemento al anterior, el decodificador.

Se describirán los algoritmos necesarios en el decodificador para interpretar la malla y los hops y realizar la interpolación elástica de las muestras que se reconstruirán a partir de los hops

El siguiente diagrama de bloques resume todo el proceso y es el objeto del presente capítulo. A medida que se vaya explicando cada bloque, se mostrará este diagrama de nuevo señalando en qué bloque se va a centrar la explicación. En total hay seis bloques que se deben ejecutar secuencialmente en el orden que aparecen etiquetados en la Figura 139.

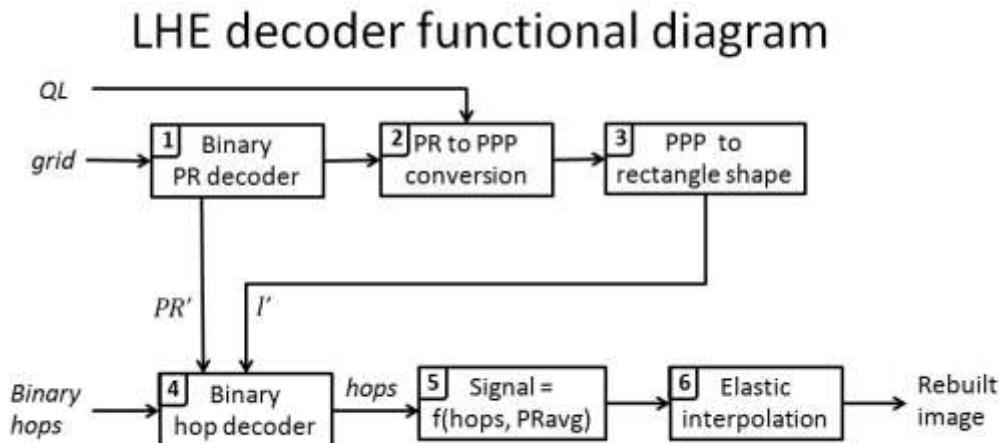


Figura 139 diagrama de bloques del decodificador LHE avanzado

Los pasos en líneas generales son:

1. Primero se interpreta la malla, que contiene la lista de valores cuantizados de PR.
2. Teniendo en cuenta el “Quality Level” (QL), se convierte cada valor de PR en un PPP.
3. Ajuste de los PPP para obtener forma rectangular, tal como se hizo en el codificador. Con ello se podrá calcular la longitud de los lados de cada bloque sub-muestreado en coordenadas X e Y.
4. Interpretación de los símbolos binarios y transformación a hops, para lo cual es necesario conocer que tabla de códigos binarios se ha usado (la deduciremos del valor de  $PR_{avg}$ ) y que

tamaño tiene cada bloque sub-muestreado ( $l'_x, l'_y$ ) ya que los símbolos se han apoyado en las redundancias espaciales y es necesario conocer que hop está en la posición superior para interpretar cada símbolo. Sin conocer las dimensiones exactas de cada bloque sub-muestreado no se puede “colocar” la secuencia de símbolos que llega y por lo tanto no se podría interpretar.

5. Transformación de los hops en la señal (luminancia o crominancia).
6. Interpolación elástica de los valores de la señal al tamaño original, bloque a bloque pues cada bloque tiene un downsampling elástico diferente.

El diseño del decodificador LHE avanzado es una contribución de esta tesis y forma parte del algoritmo LHE avanzado.

## 5.2 Decodificador binario de malla

Para ubicar el contenido de este apartado, a continuación se muestra el diagrama de bloques del decodificador LHE avanzado, destacando el módulo en el que se ubica el contenido que se va a desarrollar.

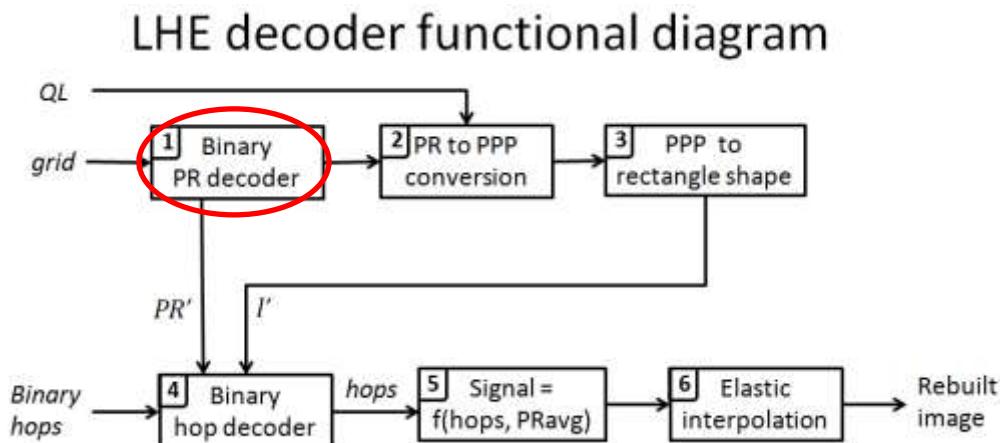


Figura 140. Ubicación del contenido que se va a desarrollar

Para entender cómo funciona este módulo hay que remitirse al codificador binario de datos de malla correspondiente en el lado del codificador. Primeramente se consulta la tabla de traducción de códigos binarios de longitud variable a símbolos. Esta tabla debe encontrarse en la cabecera del fichero LHE.

Código	símbolo
1	Cuanto más frecuente/left/up
01	
001	
0001	
0000	Cuanto menos frecuente

Tabla 17. Tabla de códigos de los cuantos

Cada vez que se lea un símbolo, a continuación habrá que traducirlo a uno de los 5 posibles cuantos. Los cuantos varían de 0 a 4, significando cada uno de ellos los diferentes valores de la métrica de PR separados geométricamente, es decir:

Cuanto de PR (5 posibles cuantos)
PR'=0 (cuanto=0)
PR'=0.125 (cuanto=1)
PR'=0.25 (cuanto=2)
PR'=0.5 (cuanto=3)
PR'=1 (cuanto=4)

Tabla 18. Los cuantos de PR y su significado

Los símbolos tienen en cuenta las redundancias espaciales y un mismo símbolo puede significar un cuanto u otro dependiendo de los cuantos adyacentes.

Para realizar la traducción se tendrá que llevar a cabo un procedimiento similar al del codificador. Un símbolo  $S=N$  se interpreta como la negación de los cuantos a los que son traducidos los símbolos de menor longitud  $[N-1, N-2, \dots, 1]$ . El orden en que se codifican es:

1. Cuanto situado a la izquierda (si existe)
2. Cuanto situado arriba (si existe y no es igual al cuanto situado a la izquierda)
3. Cuanto más frecuente (si no es ninguno de los dos anteriores)
4. Resto de cuantos ordenados por frecuencia de aparición

Por ejemplo:

- Si tenemos  $S=1$  la traducción será el cuanto que tengamos a la izquierda. Si dicho cuanto no existe, entonces será el cuanto situado arriba, y si no existe, entonces será el cuanto más frecuente.
- Si tenemos  $S=2$  la traducción será el cuanto siguiente a los cuantos descartados por  $S=1$ , siguiendo el orden establecido
- Si tenemos  $S=3$  la traducción será el cuanto siguiente a los cuantos descartados por  $S=2$ , siguiendo el orden establecido

En Consecuencia para interpretar un símbolo  $S=N$  se deben de interpretar todos los anteriores para poder conocer el cuanto al que se traduce el símbolo  $N$ .

Una vez leído cada símbolo y transformado en el cuanto correspondiente (tanto para  $PR_x$  como para  $PR_y$ ) se obtendrá una malla de puntos con los valores de la métrica  $PR$  cuantizada (a la que se denota como  $PR'$ ). Si la malla consta de 32 bloques de ancho, tendremos 33 puntos de ancho, ya que la malla de valores de  $PR'$  incluye siempre un punto más que el número de bloques. En la siguiente figura hay 4 bloques de ancho y la malla de  $PR'$  tiene 5 puntos de ancho:



Figura 141. Número de puntos de la malla

### 5.3 Obtención de PPP

Para ubicar el contenido de este apartado, a continuación se muestra el diagrama de bloques del decodificador LHE avanzado, destacando el módulo en el que se ubica el contenido que se va a desarrollar.

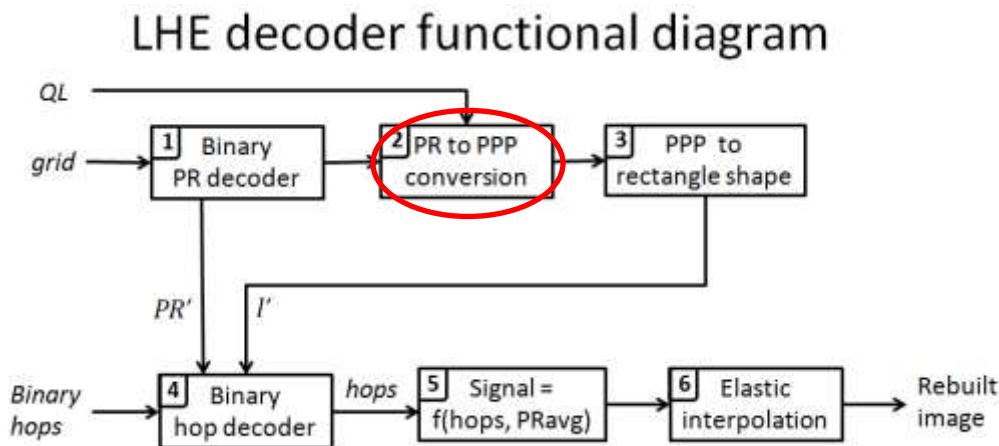
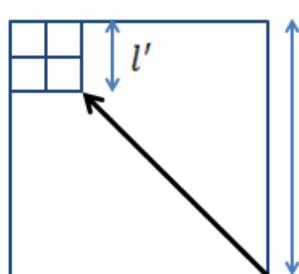


Figura 142. Ubicación del contenido que se va a desarrollar

Primeramente se debe calcular el valor de  $PPP_{max}$  para lo cual se deben leer las dimensiones de la imagen presentes en la cabecera del archivo, así como el ancho en bloques de la imagen, también presente en la cabecera del archivo. El valor de  $PPP_{max}$  se alcanza cuando se aplica la máxima compresión a un bloque, transformándolo en un bloque sub-muestreado de 2x2 píxeles



$$PPP_{max} = \frac{\text{image\_width}}{2 \cdot \text{blocksH}}$$

Donde:

- “image\_width” es el ancho en píxeles de la imagen, presente en la cabecera
- “blocksH” es el ancho en bloques de la imagen, presente en la cabecera

Una vez que tenemos  $PPP_{max}$  se calculará el valor de  $PPP$  de cada punto de la malla usando la ecuación ya presentada en capítulo dedicado al codificador:

$$\text{si } PR' > 0 \text{ entonces } PPP = \frac{PPP_{max} \cdot CF_{min} \cdot r^{99-QL}}{1 + (PPP_{max} - 1) \cdot PR'}$$

$$\text{Si } PR' = 0 \text{ entonces } PPP = PPP_{max}$$

**Ecuación 34. Fórmula de PPP**

Esta ecuación incluye el parámetro QL que debe estar presente en la cabecera del archivo LHE y cuyo valor está comprendido entre 0 y 99. Además es necesario calcular la razón “r” de la serie geométrica de los 100 valores que puede tomar QL (los 100 niveles de calidad). El valor de “r” se calculará del siguiente modo:

$$PR_{min} = 0.125 \text{ (el cuarto mínimo, no se usa el valor cero)}$$

$$CF_{min} = \frac{1 + (PPP_{max} - 1) \cdot PR_{min}}{PPP_{max}}$$

$$CF_{max} = 1 + (PPP_{max} - 1) \cdot PR_{max} \text{ (siendo } PR_{max} = 1)$$

$$r = \left( \frac{CF_{max}}{CF_{min}} \right)^{\frac{1}{99}}$$

**Ecuación 35. Fórmula de la razón de los niveles de calidad**

Como valor de  $PR_{min}$  se debe usar 0.125 ya que cuando  $PR'=0$  entonces en la transformación a PPP siempre se convertirá a  $PPP_{max}$ , tal como se explica en el capítulo del codificador. Esto permite que con independencia de QL, si un bloque no tiene información, su PPP siempre será  $PPP_{max}$ .

A partir de la Ecuación 34 se puede calcular los valores de PPP de cada punto de la malla usando los cuantos de PR correspondientes. Los valores PPP calculados pueden no corresponderse con una forma rectangular, por lo que a continuación deben ser adaptados, tal como se hizo en el codificador.

## 5.4 Adaptación de PPP a forma rectangular

Para ubicar el contenido de este apartado, a continuación se muestra el diagrama de bloques del decodificador LHE avanzado, destacando el módulo en el que se ubica el contenido que se va a desarrollar.

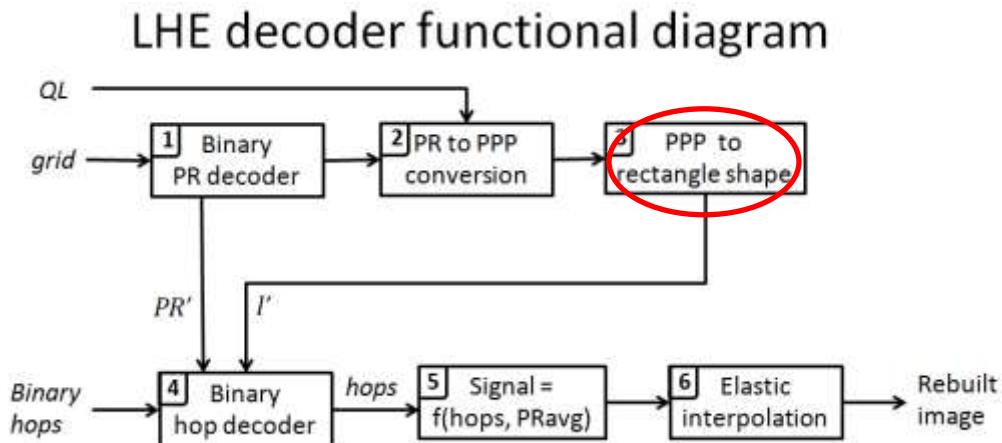


Figura 143. Ubicación del contenido que se va a desarrollar

#### 5.4.1 Restricciones de PPP

Se van a aplicar dos tipos de restricciones a los valores de PPP obtenidos, tal y como se hizo en el codificador. Estas restricciones son:

- Máxima elasticidad: Si en el codificador se restringe a que un PPP no puede superar 3 veces al PPP de otra esquina adyacente del mismo bloque, aquí también se debe hacer. Esto supone un primer ajuste de PPP.
- Forma rectangular: se aplican las ecuaciones tal y como han sido presentadas en el capítulo del codificador. Este es el segundo y más importante ajuste de PPP.

Tras aplicar la restricción a forma rectangular, se obtendrán unos nuevos valores de PPP ajustados para cada punto de la malla, tanto en coordenada x como en coordenada y. Asimismo, obtendremos el ancho y alto de cada bloque sub-muestreado, a las que hemos llamado  $l''_x, l''_y$

En este punto ya es posible interpretar los símbolos con los que se han representado a los hops, ya que conociendo el ancho y alto de cada bloque sub-muestreado se deduce cuántos símbolos de la lista de símbolos se corresponden con cada bloque y también se pueden colocar adecuadamente en su posición para poder interpretarlos teniendo en cuenta las redundancias espaciales.

#### 5.4.2 Imágenes en color

En caso de tratarse de una imagen en color, en la cabecera deberá estar presente dicha circunstancia mediante un indicador del modelo YUV utilizado, el cual podrá tomar 4 valores: YUV444, YUV422 o YUV420 o por supuesto, el modo sin color (B/N).

La interpretación es la que ya se ha expuesto en el capítulo del codificador LHE avanzado. Tras los símbolos correspondientes a los hops de la señal de luminancia, en el fichero LHE se deben encontrar a continuación los símbolos de las señales U y V. Puesto que las longitudes de los bloques sub-muestreados de luminancia son conocidas, los de las señales de crominancia se pueden calcularlos a partir de éstos, teniendo en cuenta el modelo de color.

Se ha de considerar también que nunca una longitud puede ser inferior a dos píxeles pues un bloque como mucho va a ser reducido a 4 muestras de la señal original. Esto significa que aunque se trabaje en YUV420 o YUV422, si un bloque sub-muestreado en luminancia mide 2x2, los bloques de U, V también medirán 2x2 pues ni su ancho ni su alto pueden medir menos de 2 píxeles.

## 5.5 Decodificador Binario de Hops

Para ubicar el contenido de este apartado, a continuación se muestra el diagrama de bloques del decodificador LHE avanzado, destacando el módulo en el que se ubica el contenido que se va a desarrollar.

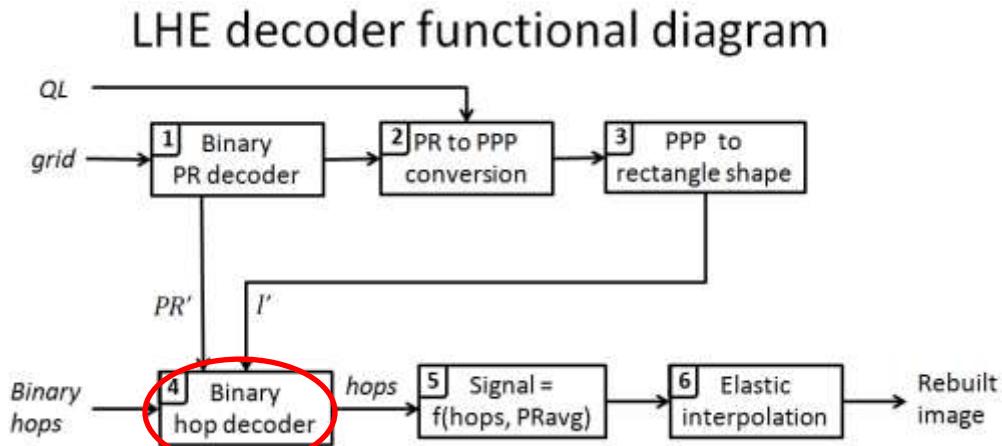


Figura 144. Ubicación del contenido que se va a desarrollar

En este módulo se van a extraer uno a uno los símbolos almacenados en el fichero y se van a convertir en hops. La estructura interna del decodificador binario de hops es exactamente la inversa a la del codificador binario de hops, tal como muestra la siguiente figura, en la que se han representado ambos.

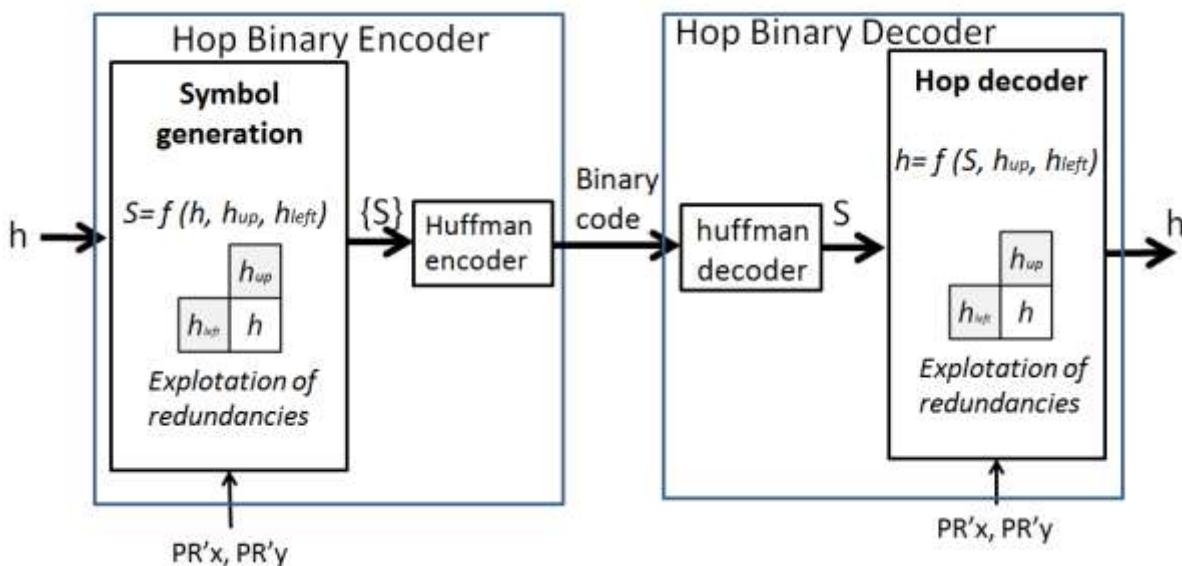


Figura 145. Estructura del decodificador binario de hops

Los pasos que se deben dar son:

- Primeramente se decodifican los símbolos usando la tabla Huffman que debe encontrarse en la cabecera del fichero. Uno a uno se trasladan cada código binario a un símbolo.
- Una vez extraído el símbolo, se procede a interpretar a qué hop corresponde. Para ello se debe

tener en cuenta los hops ya decodificados situados a la izquierda y en la posición superior. Puesto que se conocen las longitudes exactas de cada bloque sub-muestreado que estamos decodificando, se pueden ordenar los símbolos en matrices del tamaño ancho y alto de cada bloque sub-muestreado y así conocer qué símbolos son el izquierdo y el superior.

Para transformar un símbolo ( $S$ ) en un hop ( $h$ ) se usa la misma tabla del codificador

$S$	$h$
1	$h_0/\text{up}$
2	$h_1$
3	$h_{-1}$
4	$h_2$
5	$h_{-2}$
6	$h_3$
7	$h_{-3}$
8	$h_4$
9	$h_{-4}$

Tabla 19 Códigos binarios del cuantizador LHE

La interpretación que se hace de cada símbolo se basa en que un símbolo niega a todos los hops que representen los símbolos menores.

Para el primer símbolo del bloque:

- si  $PR'x_{avg} \leq PR'y_{avg}$ . Si  $S = 1$  se trata de  $h_0$  (redundancia horizontal) y si  $S = 2$  entonces se transforma en el hop “up” (el hop situado en la posición superior, si existe)
- si  $PR'x_{avg} > PR'y_{avg}$  Si  $S = 1$  se trata de “up” (redundancia vertical, si existe) y si  $S = 2$  entonces se transforma en el hop  $h_0$ . En caso de no existir “up”, entonces la interpretación de  $S=1$  es el hop  $h_0$ .
- En caso de no haber traducido el símbolo con el paso anterior, se consulta la Tabla 19 y se transforma en el hop correspondiente, pero teniendo en cuenta que no se debe descartar dos veces a ningún hop.

Ejemplo1:

Tenemos  $S=5$ ,  $\text{up}=h_{-1}$ ,  $PR'x_{avg} \leq PR'y_{avg}$

Entonces el valor  $S=1$  descarta a  $h_0$ ,  $S=2$  descarta a  $\text{up}$ , es decir a  $h_{-1}$ ,  $S=3$  descarta a  $h_1$  y  $S=4$  descarta a  $h_2$  pues  $h_{-1}$  ya ha sido descartado. Por lo tanto  $S=5$  es  $h_2$

Ejemplo2:

Tenemos  $S=5$ ,  $\text{up}=h_4$ ,  $PR'x_{avg} \leq PR'y_{avg}$

Entonces el valor  $S=1$  descarta a  $h_0$ ,  $S=2$  descarta a  $\text{up}$ , es decir a  $h_4$ ,  $S=3$  descarta a  $h_1$  y  $S=4$  descarta a  $h_{-1}$ . Por lo tanto  $S=5$  es  $h_2$

- Para el resto de símbolos seguiremos el mismo procedimiento, con la excepción de que si en un símbolo  $S_i$  se produce una redundancia ( $h_0$  o up), en el siguiente símbolo  $S_{i+1}$  se

interpreta en primer lugar el hop correspondiente la redundancia que se ha producido en  $S_i$ . En caso de no haber ninguna redundancia en  $S_i$ , entonces se usará el orden establecido para el símbolo inicial del bloque.

Se realizará esta transformación para cada hop de cada bloque, y al final se obtendrán todos los hops recuperados. Esta operación es paralelizable totalmente, pudiendo efectuarse la conversión de códigos en hops en todos los bloques a la vez, ya que no hay dependencias entre ellos.

## 5.6 Recuperación de la señal

Para ubicar el contenido de este apartado, a continuación se muestra el diagrama de bloques del decodificador LHE avanzado, destacando el módulo en el que se ubica el contenido que se va a desarrollar.

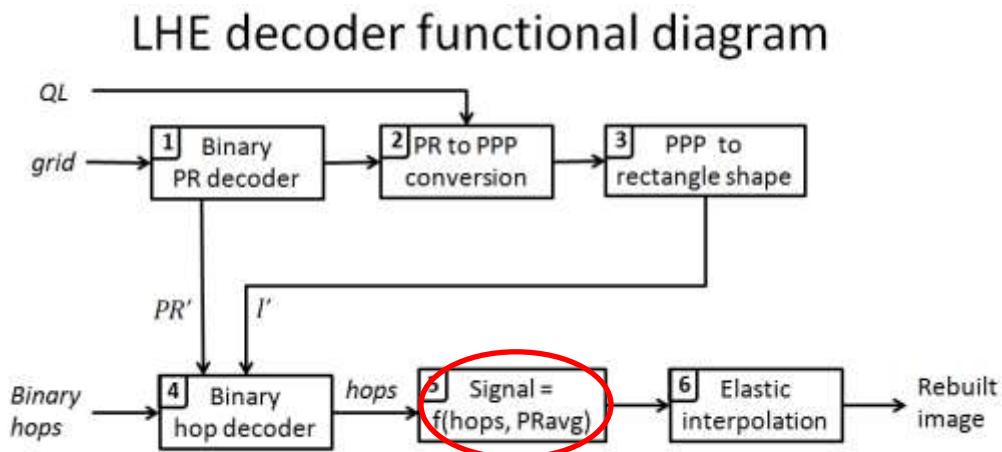


Figura 146. Ubicación del contenido que se va a desarrollar

Al referirse a la “señal” se va a considerar que se pretende recuperar la luminancia. Para recuperar las señales de crominancia el procedimiento a emplear es exactamente el mismo.

Se parte de una lista de identificadores de hop “ $i$ ”, que se han obtenido gracias al decodificador binario y ahora deben ser interpretados. Interpretar un hop “ $i$ ” es recuperar su luminancia asociada. Para ello se debe de conocer el valor de la luminancia de los pixels que se usan como referencias para el hop, es decir el situado a la izquierda (denotado como “a” en la siguiente figura) y el situado en la posición superior derecha (denotado como “b”). Estas dos luminancias de referencia permiten calcular el “color de fondo”, tal y como se expuso en el capítulo sobre LHE básico

$x_0$				$a$	$x_1$	
$b$						$b$
$x_2$			$b$		$a$	$x_4$
		$a$	$x_3$			

$$\widehat{x}_0 = \text{original value}$$

$$\widehat{x}_1 = a$$

$$\widehat{x}_2 = b$$

$$\widehat{x}_3 = \frac{4a + 3b}{7}$$

$$\widehat{x}_4 = \frac{a + b}{2}$$

Figura 147. Casos posibles y sus referencias disponibles para la predicción

Conocer las luminancias “a” y “b” no es inmediato. Estos valores hacen referencia a luminancias ya calculadas. En el interior del bloque no es un problema, pero en las fronteras del bloque que no sean fronteras de la imagen, es necesario hacer una interpolación de las fronteras de los bloques adyacentes a las longitudes del bloque que está siendo interpretado, tanto para la frontera vertical izquierda como para la horizontal superior. Esta interpolación se realizará mediante la técnica de vecino cercano.

Esta es una interpolación elástica, es decir, debe tener en cuenta que los PPP evolucionan y por lo tanto no es válido simplemente “comprimir” o “estirar” la frontera del bloque. Hay que adaptar los PPP mediante una interpolación elástica. La forma de hacerlo se explica en un apartado posterior.

El proceso es idéntico al que lleva a cabo el codificador, antes de codificar o decodificar un bloque, es necesario interpolar las fronteras con las que tiene dependencia.



Para decodificar el bloque “k” primero se deben interpolar las luminancias de la frontera vertical del bloque “i” y las de la frontera horizontal del bloque “j”

Figura 148. Dependencias entre bloques

En el primer bloque de la imagen (superior izquierdo) no hay fronteras de ningún otro bloque previo, pero en los siguientes bloques hay fronteras previas verticales, horizontales o ambas.

Una vez realizada dicha interpolación, se dispondrá de las luminancias de referencia necesarias para calcular la predicción del color de fondo ( $hop_0$ ) aplicando las ecuaciones de la Figura 147.

A continuación se deben sumar a la luminancia  $hop_0$  el valor del salto en luminancia que supone el hop “i”, usando las ecuaciones presentadas en el capítulo sobre LHE básico.

$$h_2 = h_1 \cdot r \quad h_3 = h_2 \cdot r \quad h_4 = h_3 \cdot r$$

$$h_{-2} = h_{-1} \cdot r \quad h_{-3} = h_{-2} \cdot r \quad h_{-4} = h_{-3} \cdot r$$

Ecuación 36 Relaciones entre hops

En estas ecuaciones aparece la razón geométrica y el hop  $h_1$  que es diferente al resto. La razón es la

misma para todos los hops del bloque y la deducimos de la siguiente tabla presentada en el capítulo del codificador LHE avanzado. La razón geométrica se puede deducir a partir de la media del valor cuantizado de PR.

$$PR'_{avg} = \frac{\sum_0^3 PR'_x(i) + \sum_0^3 PR'_y(i)}{8}$$

$PR'_{avg}$	$r_{opt}$	$h1_{max}$
1.0	30	16
[0.75,1.0)	25	14
[0.5,0.75)	25	12
[0.25,0.5)	22	10
[0,0.25)	20	8

Tabla 20. Valores óptimos de la razón geométrica y primer hop en función de la PR promedio

Sólo queda conocer el valor de  $h_1$ , pues a partir de la Tabla 18 se sabe cuál es su valor máximo y su valor mínimo (el valor mínimo siempre es 4). Para ello simplemente antes de interpretar el primer hop del bloque, se establece como valor inicial de  $h_1$  el mismo valor que establece el codificador LHE, es decir:

$$h_1 = \frac{4 + h1_{max}}{2}$$

Una vez que se calcula la luminancia del primer hop se debe ajustar  $h_1$  siguiendo el algoritmo que regula su evolución y que se muestra a continuación en pseudocódigo.

```
//hop is "small hop" if it belongs to {h_{-1}, h_0, h_1}
if (small_hop AND previous_small_hop) then
    h1=h1-1
    if (h1<min_h1) then
        h1=min_h1
    endif
    else
        h1=max_h1
    endif
    previous_small_hop=small_hop
```

Este proceso se llevará a cabo para todos los hops de un bloque, y para todos los bloques de la imagen. Puesto que son necesarias las fronteras de los bloques anteriores adyacentes (izquierdo y superior), la paralelización de esta tarea requiere  $2N-1$  pasos siendo  $N$  el número de bloques de la horizontal de la imagen. Y como  $N=32$ , en total son 63 pasos. Esto ya ha sido explicado en el capítulo del codificador pues ocurre exactamente lo mismo.

Existe la posibilidad de paralelizar completamente esta tarea si se prescinde de las fronteras de los bloques vecinos y se considera cada bloque como una imagen individual. En ese caso todos los bloques se pueden procesar a la vez pero la calidad resultante es algo inferior, tal como se expuso en el capítulo del codificador LHE avanzado.

## 5.7 Interpolación elástica

El concepto “interpolación elástica” se refiere a la interpolación que se realiza sobre una imagen sub-muestreada elásticamente.

La interpolación elástica se puede realizar aplicando las mismas técnicas conocidas en el estado del arte que en se aplican para interpolar imágenes, aunque con ciertos matices que se van a detallar en este apartado.

Las técnicas que se han implementado en esta tesis y con las que se han generado resultados son “vecino cercano”, “bilineal” y “bicúbica”. En función del tipo de interpolación los resultados son más o menos buenos, siendo la interpolación bicúbica la más recomendable en calidad de estas tres y la de vecino cercano la más rápida aunque la peor en calidad.

La interpolación es un proceso “separable” [72], igual que lo ha sido el downsampling elástico. Puesto que en downsampling primero se sub-muestra la coordenada X y después la Y, se va a realizar la interpolación en orden inverso, “deshaciendo” el downsampling. En realidad al ser un proceso separable también es conmutativo y por lo tanto da igual el orden en que se efectúen las operaciones. Esto se ha comprobado experimentalmente y se han obtenido idénticos resultados.

### 5.7.1 Interpolación elástica por vecino cercano

Es la interpolación más sencilla y rápida, ya que se basa simplemente en repetir el valor del brillo del pixel escalado en todos los pixels que representa.

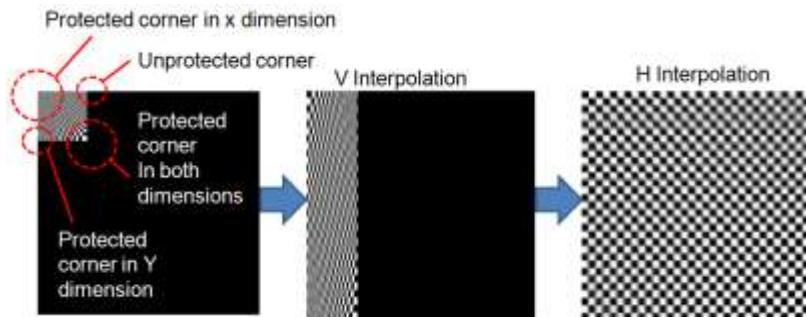


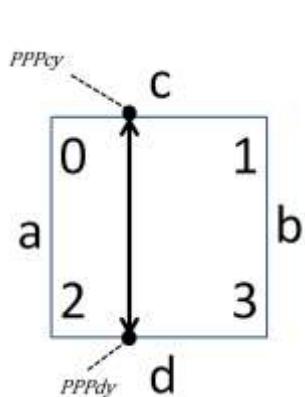
Figura 149. Interpolación de un bloque por vecino cercano

Tras interpolar un bloque, se recupera una versión degradada del bloque original, que rellena por completo el área ocupada por dicho bloque.

Para poder realizar esta operación se debe calcular cuántos píxeles representa cada pixel sub-muestreado y puesto que se dispone de los valores de PPP en cada esquina del bloque, se puede realizar una interpolación lineal de la variable PPP entre esquinas, sumando el gradiente de PPP que calcularemos como se describe a continuación en cada iteración.

Se van a presentar las ecuaciones para la interpolación vertical ya que la horizontal es idéntica.

El objetivo es calcular el gradiente de  $PPP_y$  en cada pixel en cualquier scanline vertical entre el lado “c” y el lado “d”, al que se va a denotar como  $Grad_{PPP_y}$ . Para ello primeramente se calcula el valor de  $PPP_y$  en el punto inicial del lado c donde empieza la scanline, al que llamo  $PPP_{cy}$ .



$$Grad_{PPP_y} = \frac{PPP_{dy} - PPP_{cy}}{l_a'' - 1}$$

siendo:

$$\begin{aligned} PPP_{cy} &= PPP_{0y} + x \cdot Grad_{PPP_{cy}} \\ PPP_{dy} &= PPP_{2y} + x \cdot Grad_{PPP_{dy}} \end{aligned}$$

$$Grad_{PPP_{cy}} = \frac{PPP_{1y} - PPP_{0y}}{l_c'' - 1}$$

$$Grad_{PPP_{dy}} = \frac{PPP_{3y} - PPP_{2y}}{l_c'' - 1}$$

Ecuación 37. Gradienes de interpolación

Cada pixel sub-muestreado representa un segmento de longitud  $PPP_y$  en la imagen original, de modo que a lo largo del scanline se suma el gradiente  $Grad_{PPP_y}$  por cada pixel sub-muestreado, y se rellena con la luminancia de este pixel todos los píxeles que representa.

Aclaración: El denominador de la fórmula de  $Grad_{PPP_y}$  lleva un “-1”, esto es debido a que se está tratando con variables discretas y se va a alcanzar  $PPP_{dy}$  desde  $PPP_{cy}$  en  $l_a'' - 1$  pasos puesto que se quiere alcanzar  $PPP_{dy}$  al principio del último pixel y no al final. Si no se resta el “-1” entonces el último pixel no alcanzará  $PPP_{dy}$ , sino algo menos.

Hay que tener en cuenta que un pixel sub-muestreado no representa un número entero de píxeles, sino que el segmento de longitud  $PPP_y$  puede empezar en medio de un píxel y terminar en medio de otro. Esto significa que el pixel inicial y final son casos especiales que se deberán mezclar adecuadamente con el brillo del pixel anterior o posterior, proporcionalmente al porcentaje de pixel cubierto en cada caso

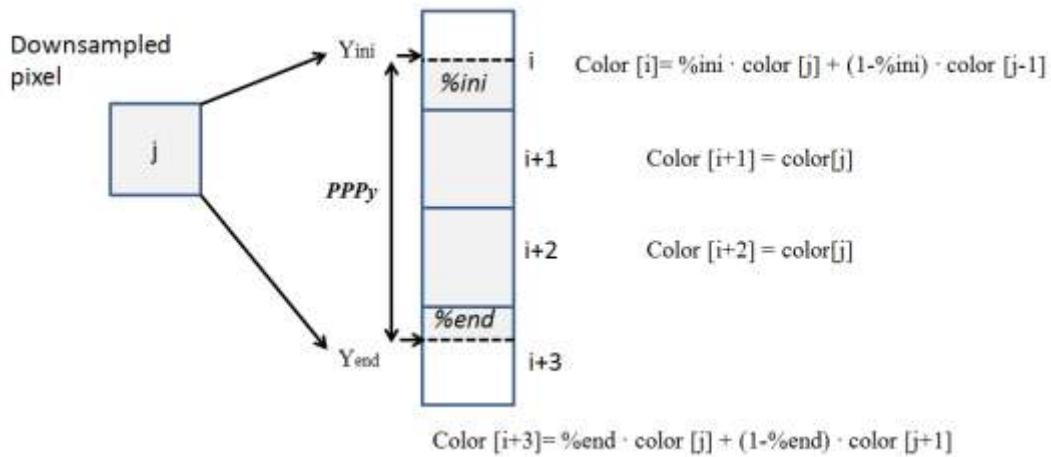


Figura 150. Segmento vertical cubierto por cada pixel sub-muestreado

En la interpolación por vecino cercano, y en el caso particular de los límites del bloque, el valor de  $\%ini$  o el valor de  $\%end$  siempre serán 100%, con independencia de los PPP de dicho pixel ya que la restricción

rectangular garantiza que la suma de todos los PPP que representan los pixeles sub-muestreados de una scanline es igual a la longitud del lado. Esto significa que en las fronteras de los bloques no es necesario hacer mezclas con pixeles de otros bloques. Las mezclas entre pixeles solo ocurren en el interior del bloque y siguiendo el planteamiento de la Figura 150

Una vez realizada la interpolación en la coordenada Y, se procederá con la interpolación en la coordenada X siguiendo el mismo proceso descrito y obteniendo finalmente un bloque completamente lleno.

La estrategia de interpolación elástica, complementaria al downsampling elástico, es una contribución de esta tesis.

### 5.7.2 Interpolación elástica bilineal

AL igual que con la interpolación por vecino cercano se trata de un proceso separable: primero se interpola la coordenada Y y después la X. Exactamente en orden inverso a como se hizo al sub-muestrear (aunque es arbitrario). Las ecuaciones para el cálculo de PPP son exactamente las mismas que las que aplicamos en vecino cercano.

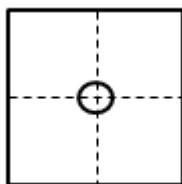
A diferencia de la interpolación por vecino, la interpolación bilineal no replica el color del pixel sub-muestreado en el área representada por dicho pixel, sino que interpola linealmente las luminancias comprendidas entre dos muestras consecutivas.

Se calcula el gradiente de luminancia (o la componente de color) entre dos muestras consecutivas mediante la siguiente ecuación:

$$grad_{color} = \frac{color[j + 1] - color[j]}{PPP}$$

Ecuación 38. Gradiente de componente de color entre dos muestras

Este gradiente se utiliza para colorear los pixels intermedios entre dos muestras consecutivas, pintando cada pixel con el valor de color interpolado en su centro.



El color de cada pixel se calcula interpolando el valor de la luminancia en su centro

Figura 151. Coordenada donde debemos interpolar el color de un pixel

Al necesitar dos muestras para interpolar entre ellas, y puesto que cada pixel sub-muestreado representa a PPP pixels, el espacio comprendido entre el los laterales del bloque y  $\frac{PPP}{2}$  quedará sin pintar, y por lo tanto pendiente de interpolar con los bloques adyacentes:

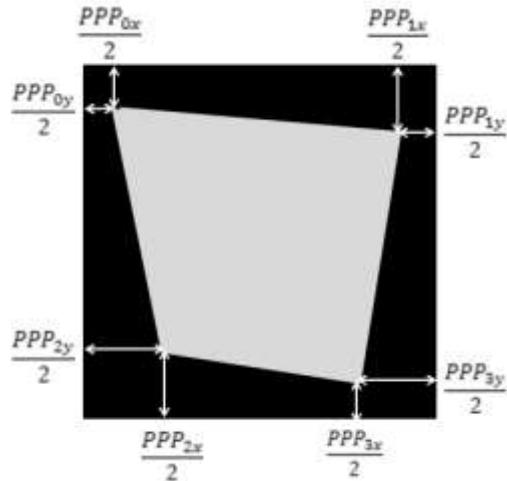


Figura 152. Área pintada por una interpolación bilineal

Debido a que cada esquina puede poseer un valor de PPP diferente tanto en X como en Y, el resultado es un cuadrilátero cuyos laterales no son rectos y con píxeles parcialmente llenos por donde pasan los laterales. Si la interpolación no fuese elástica los valores de PPP serían siempre los mismos y el cuadrilátero tendría los laterales perfectamente rectos y paralelos, permitiendo la interpolación entre bloques fácilmente.

La coordenada interpolada de una muestra se calcula sumando el valor de PPP actualizado (sumándole el gradiente de PPP) a la coordenada interpolada de la muestra anterior. En la siguiente figura se muestra un pixel parcialmente coloreado pero como el centro de dicho pixel se encuentra entre dos muestras, no hay ningún problema en interpolar su color.

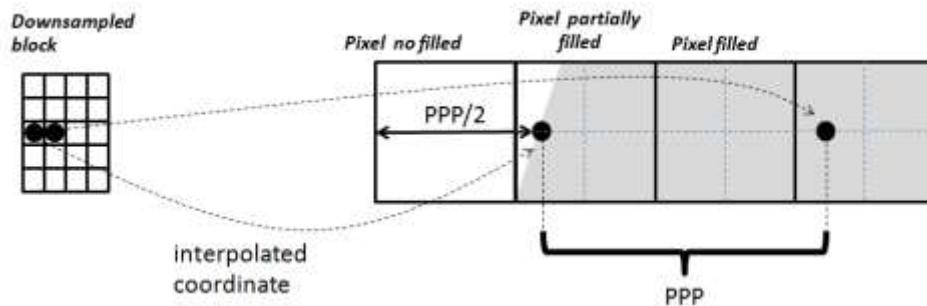


Figura 153. Pixel parcialmente cubierto interpolable

En esta otra figura no ocurre lo mismo. El centro del pixel parcialmente coloreado queda fuera del intervalo entre las dos muestras y por lo tanto su color no puede ser interpolado usando únicamente las muestras del bloque sub-muestreado.

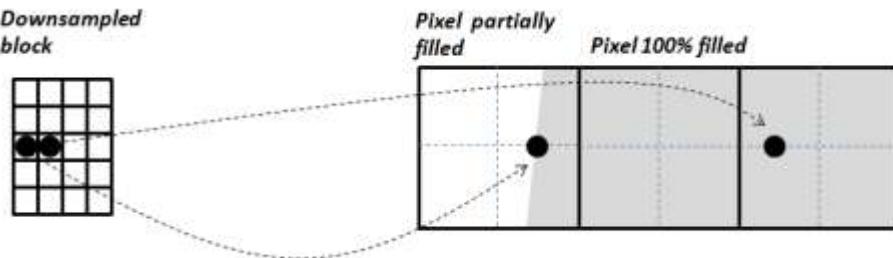


Figura 154. Pixel parcialmente cubierto no interpolable

Si se trata de interpolar con la última muestra el bloque anterior se cometería un error pues el salto de  $PPP_y$  entre bloques puede ser notable y se estaría interpolando con una muestra desalineada en altura. Este problema no ocurre en una interpolación bilineal convencional, pero si ocurre en elástica.

En este tipo de píxeles parcialmente llenos se puede emplear dos estrategias:

1. Pintarlo con el valor de la muestra.
2. No pintarlos e interpolarlos posteriormente con el bloque adyacente ya interpolado. Esto se puede hacer de dos modos:
  - Para hacer correctamente esta estrategia habría que almacenar el valor de la muestra y su coordenada interpolada (que puede no coincidir con el centro de un pixel) para cada pixel de los laterales del bloque.
  - Otra posibilidad es no almacenar nada e interpolar con los valores de color de los pixels ya pintados, cuyo color ya ha sido interpolado como muestro a continuación.

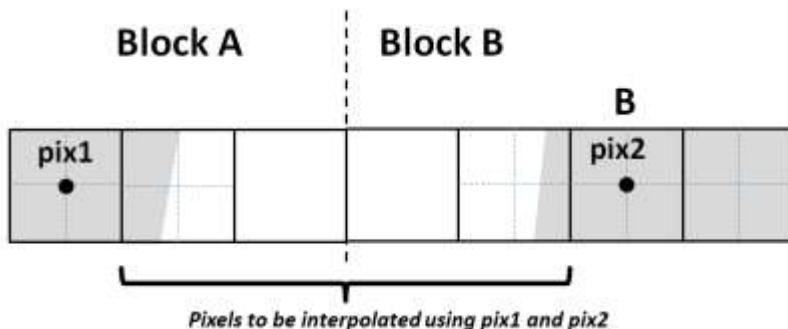


Figura 155. Estrategia de interpolación con píxeles ya interpolados

Se han evaluado la opción (1) y la (2b) midiendo el PSNR resultante en todos los ratios de compresión, y siempre ha resultado ligeramente mejor pintarlo con el valor de la muestra, es decir, la opción (1). La opción (2a) ha sido descartada porque añade complejidad adicional y gasto de memoria para ganar muy poca calidad extra. De hecho la diferencia de PSNR entre (1) y (2b) es normalmente muy baja.

El inconveniente del método (1) es que si el porcentaje de relleno del pixel es muy bajo, el error cometido es máximo y se puede producir un efecto de “cuadrícula” cuando el valor de PPP es cercano a 2, es decir, cuando la compresión no es muy intensa y los bloques adyacentes se encuentran muy próximos. En este caso rellenarlo con el color de la muestra se perciben claramente las fronteras de los bloques, al no encontrarse interpoladas.

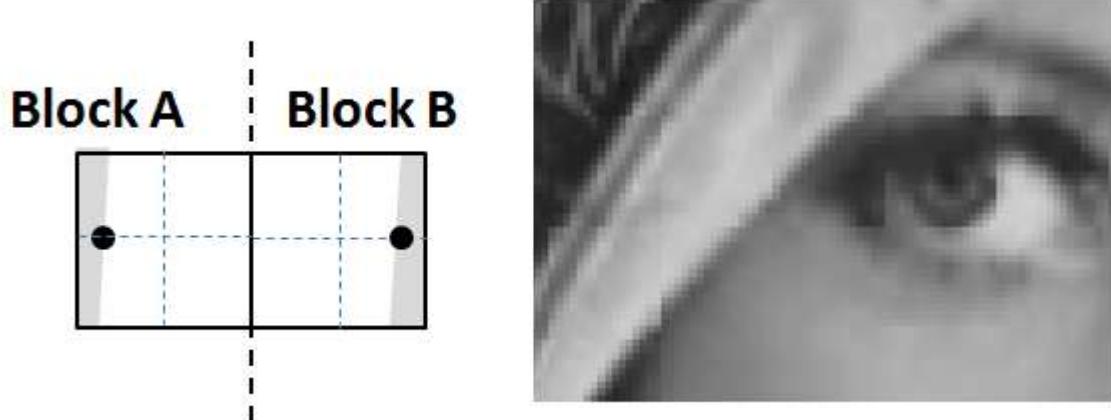


Figura 156. Efecto de cuadrícula

Para mitigar este problema, se ha optado por una tercera estrategia, consistente en pintar el pixel con el valor de la muestra siempre que al menos haya un cierto área rellena, concretamente que se cubra un mínimo del 25% del valor de la coordenada que está siendo interpolada. Esta tercera opción de compromiso no mejora los resultados de la simple asignación (opción 1) pero el efecto de cuadrícula se mitiga bastante, tal y como se muestra en la Figura 157

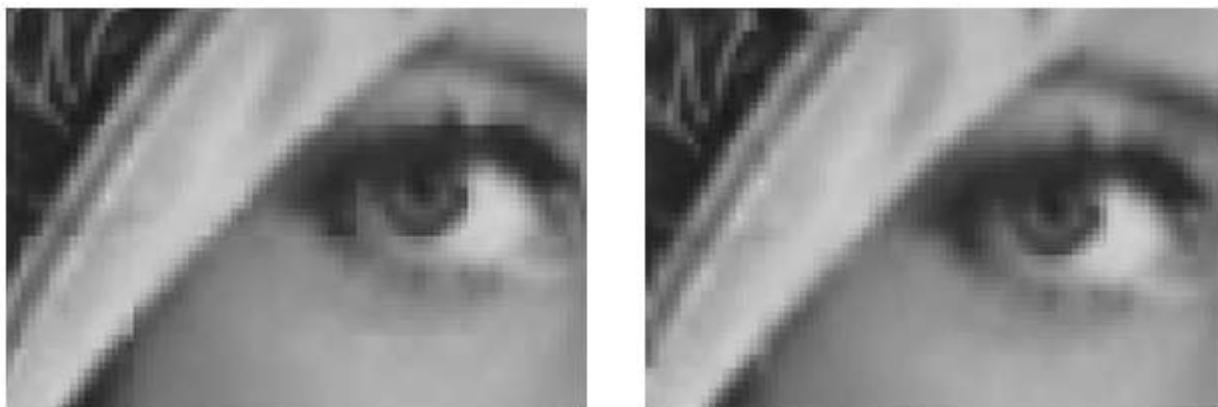


Figura 157. Mitigación del efecto de cuadrícula

Una vez interpolados todos los bloques de la imagen tendremos un resultado como el que se muestra a continuación.

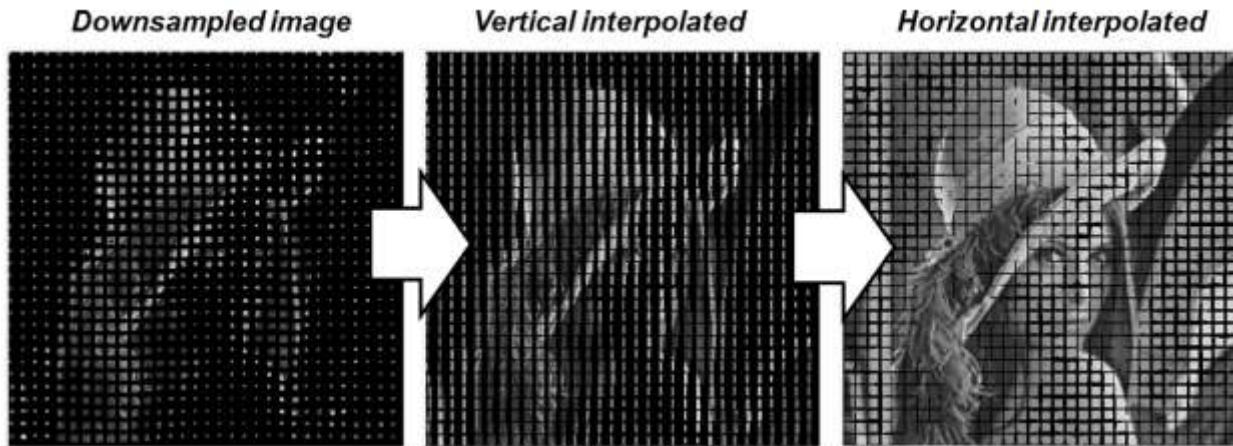


Figura 158. Resultado de la interpolación

Ahora se debe interpolar las “costuras” entre bloques, primero las horizontales y después las verticales o viceversa. Esta parte del proceso no existe en la interpolación por vecino, porque los bloques en ese tipo de interpolación quedan completos y sin costuras al ser interpolados.

Tampoco existe en la interpolación bilineal convencional, donde al tener todas las muestras el mismo valor de PPP, se puede interpolar entre bloques sin dificultad. Para visualizar el problema de esta desalineación, observe la siguiente figura donde se muestran cuatro bloques adyacentes

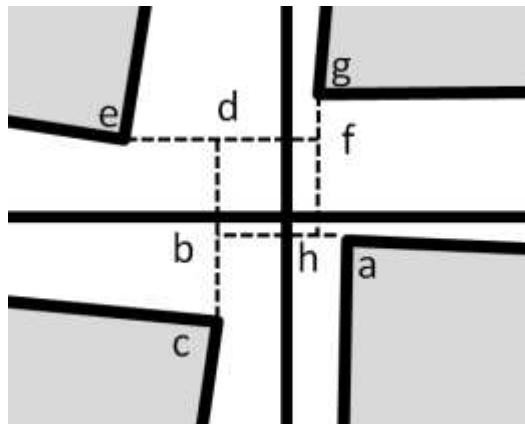


Figura 159. Problema de la desalineación de muestras en la interpolación elástica

Para interpolar los pixels situados a la izquierda del pixel “a”, (por ejemplo el “h”) se necesitaría conocer “b” el cual puede ser interpolado entre “c” y “d”. Como para ello se necesita a “d”, éste debe ser interpolado entre “e” y “f” pero “f” a su vez debe ser calculado interpolando entre “g” y “h” y esto no es posible pues “h” es parte de los puntos que se pretendía interpolar.

En una primera aproximación se realizó una interpolación horizontal entre bloques y después vertical. Esto habría funcionado si el downsampling fuese convencional y no elástico ya que los bloques interpolados serían rectángulos pero al interpolar elásticamente, los laterales no son paralelos y esa estrategia deja áreas sin interpolar como se muestra a continuación

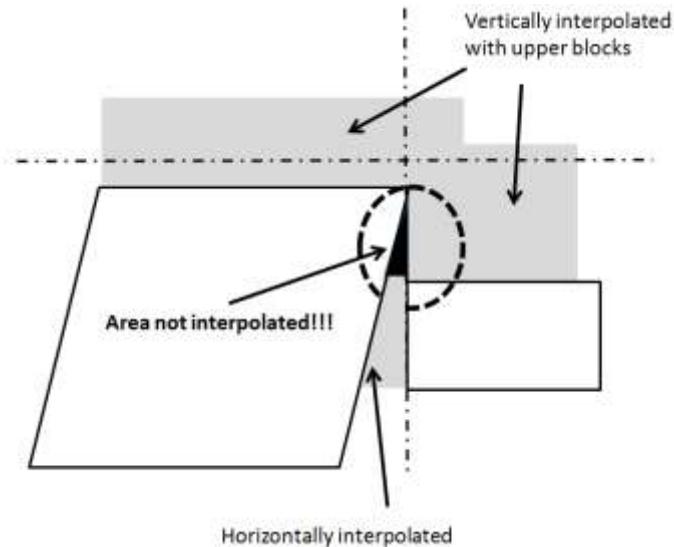


Figura 160. Áreas que podrían quedarse sin interpolar

La solución adoptada consiste en:

- Se Interpolan horizontalmente todos los bloques, cada bloque con su bloque anterior situado a la izquierda. El rango vertical será aquel en el que coincidan los laterales de ambos bloques y para cada scanline horizontal simplemente recorreremos desde  $x_{ini} - \frac{PPP_{max}}{2}$  hasta  $x_{ini} + \frac{PPP_{max}}{2}$ . Se empieza a pintar sólo cuando se encuentre el primer pixel negro y se acaba cuando se encuentre un pixel de color, sin necesidad de llegar a  $x_{ini} + \frac{PPP_{max}}{2}$ . Los pixels negros serán así interpolados linealmente entre los pixels laterales de cada uno de los dos bloques interpolados.

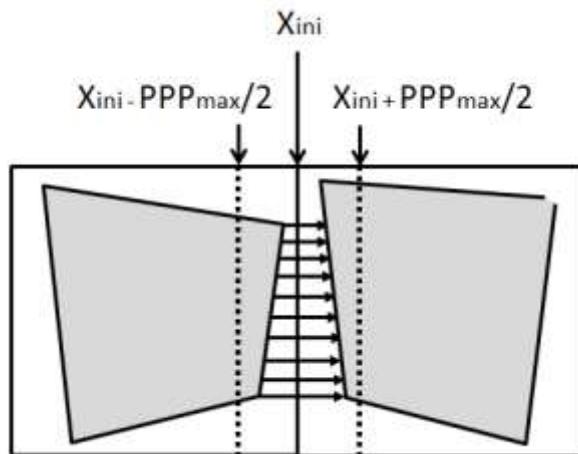


Figura 161. Interpolación horizontal de costuras

Un modo muy sencillo de optimizar esta interpolación es calcular el mayor  $PPP_x$  del lateral derecho del bloque izquierdo y usarlo en lugar de  $PPP_{max}$  para calcular la coordenada inicial del recorrido de la interpolación de la costura.

Los casos particulares de los bloques situados a la izquierda de la imagen no se pueden

interpolar con ningún bloque izquierdo, de modo que simplemente se replica el color del primer pixel disponible y lo mismo con los bloques situados a la derecha de la imagen.

- A continuación se interpola verticalmente. Cada scanline recorrerá desde  $Y_{ini} - \frac{PPP_{max}}{2}$  hasta  $Y_{ini} + \frac{PPP_{max}}{2}$ . Se empieza a pintar cuando encontramos un pixel negro pero no se acaba cuando se encuentre un pixel de color, sino que se continúa el scanline por si se encuentra otro tramo de pixels negros que deban ser interpolados linealmente entre sus extremos, llegando hasta el extremo  $Y_{ini} + \frac{PPP_{max}}{2}$ . Esta estrategia garantiza que no va a quedar ningún área sin interpolar como el mostrado en la Figura 160

Los casos particulares de los bloques situados en la parte superior de la imagen no se pueden interpolar con ningún bloque superior, de modo que simplemente se replica el color del primer pixel disponible y lo mismo con los bloques situados en el lateral inferior de la imagen.

- El resultado tras interpolar horizontalmente y verticalmente las costuras es una imagen completa como la que se muestra a continuación, partiendo de la imagen con los bloques ya interpolados y llegando a la reconstrucción de la imagen original.



Figura 162. Interpolación de costuras

#### Contribución

La interpolación elástica bilineal requiere una estrategia de interpolación de pixels laterales parcialmente cubiertos así como una estrategia de interpolación entre “costuras”. Estos problemas son inexistentes en una interpolación convencional y son producidas por el desalineamiento de muestras. La estrategia para resolver ambos problemas constituye una contribución de la presente tesis, que forma parte del proceso de interpolación elástica.

### 5.7.3 Interpolación elástica bicúbica

La interpolación elástica bicúbica, al igual que la bilineal, y la de vecino es separable, por lo que primeramente se realizará una interpolación en coordenada Y y después en coordenada X. Al igual que la bilineal, la bicúbica produce bloques interpolados con laterales no paralelos cuyas “costuras” deben ser interpoladas.

Sin embargo hay una diferencia fundamental entre ambas que está relacionada con el número de muestras que involucran los cálculos de una interpolación bicúbica. La interpolación bilineal toma las

cuatro muestras más cercanas para interpolar linealmente el resultado. La interpolación bicúbica, en cambio, requiere un total de 16 pixeles cercanos.

Puesto que se realiza la interpolación de forma separable se puede hablar de dos muestras en el caso lineal y de cuatro en la cúbica. La interpolación lineal aproxima el valor de un punto entre dos muestras con una recta que une ambos valores, por lo que la ecuación es de tipo  $y = ax + b$  donde los coeficientes se calculan simplemente imponiendo que la recta pase por las dos muestras disponibles.

Interpolación lineal	
$c_1$ : color de la muestra 1 $c_2$ : color de la muestra 2 $c_x$ : color del punto interpolado 	La ecuación general es: $c_x = ax + b$ Tenemos dos incógnitas y dos condiciones: Que pase por el primer punto: $c_1 = a0 + b$ Que pase por el segundo punto: $c_2 = a1 + b$ Quedando: $c_x = (c_2 - c_1)x + c_1$

Ecuación 39. Interpolación lineal

En la interpolación cúbica se aproxima con una curva de tipo  $y = ax^3 + bx^2 + cx + d$ . Este polinomio al ser una curva de tercer grado, permite valores interpolados por encima y por debajo de los valores inicial y final de cada segmento. Los máximos y mínimos son aquellos puntos donde la derivada es cero de modo que puesto que la derivada es de grado 2, se tiene un máximo y un mínimo, dentro o fuera del intervalo a interpolar. El intervalo a interpolar es el comprendido entre las dos muestras centrales ( $c_2$  y  $c_3$ )

Interpolación cúbica	
$c_1$ : color de la muestra 1 $c_2$ : color de la muestra 2 $c_3$ : color de la muestra 3 $c_4$ : color de la muestra 4 $c_x$ : color del punto interpolado 	La ecuación general es: $c_x = ax^3 + bx^2 + cx + d$ Y las condiciones son: $c_x = c_1$ cuando $x = -1$ $c_x = c_2$ cuando $x = 0$ $c_x = c_3$ cuando $x = 1$ $c_x = c_4$ cuando $x = 2$ Quedando : $C_x = \frac{1}{2} \cdot x \cdot (C_2 - C_0 + x \cdot (2C_0 - 5 \cdot C_1 + 4 \cdot C_2 - C_3 + x \cdot (3 \cdot (C_1 - C_2) + C_3 - C_0)))$

Ecuación 40. Interpolación cúbica

Uno de los motivos por el que este tipo de interpolación es mejor que el bilineal está relacionado con una degradación inevitable en el sub-muestreado. Al realizar el sub-muestreado, los puntos más brillantes se mezclan con otros menos brillantes y se oscurecen, mientras que a los puntos más oscuros les ocurre lo contrario. Con la interpolación bilineal ningún punto interpolado se sale del rango de valor de brillo de las muestras mientras que con la bicúbica, al poder tener máximos y mínimos en coordenadas diferentes a las muestras, puede superar el rango de brillos de las mismas y así aproximarse más a la imagen original (en la mayoría de los casos).

Se ha mostrado que una interpolación cúbica involucra 4 muestras para cualquier pixel que se desee interpolar ubicado entre las 4 muestras centrales. Sin embargo una interpolación lineal involucra tan solo dos muestras. Se ha de tener en cuenta en todo momento que aunque el objetivo es realizar una interpolación bicúbica, al hacerlo mediante un proceso separable el problema se transforma en una interpolación cúbica.

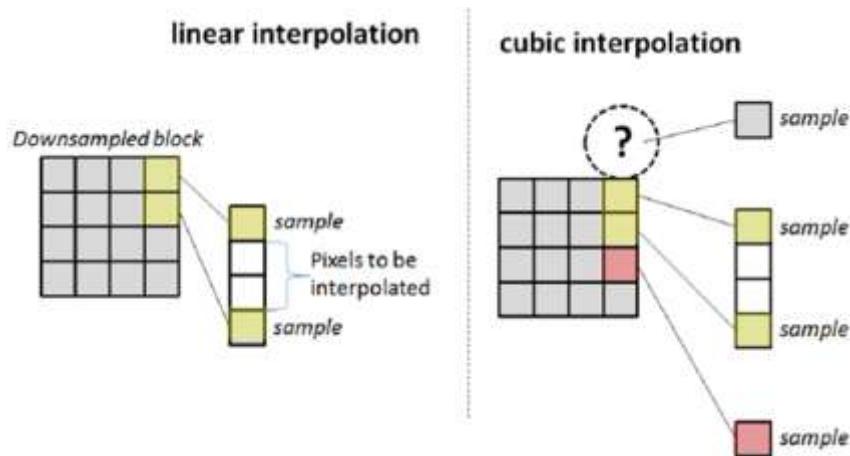


Figura 163. Interpolación cónica necesita 4 muestras

La necesidad de cuatro muestras implica que para empezar a interpolar el bloque verticalmente necesitaríamos una fila adicional de muestras ubicada en una fila superior al bloque. Y lo mismo sucede al final de la scanline vertical, se necesitaría una muestra más abajo que no se encuentra en el bloque sub-muestreado.

La solución propuesta consiste en utilizar las muestras de los bloques superior e inferior pero puesto que dichos bloques pueden tener un ancho diferente y unos PPP diferentes, es necesario realizar un proceso de interpolación y downsampling elástico para alcanzar los nuevos valores de PPP del bloque y al ancho del bloque sub-muestreado que estamos interpolando.

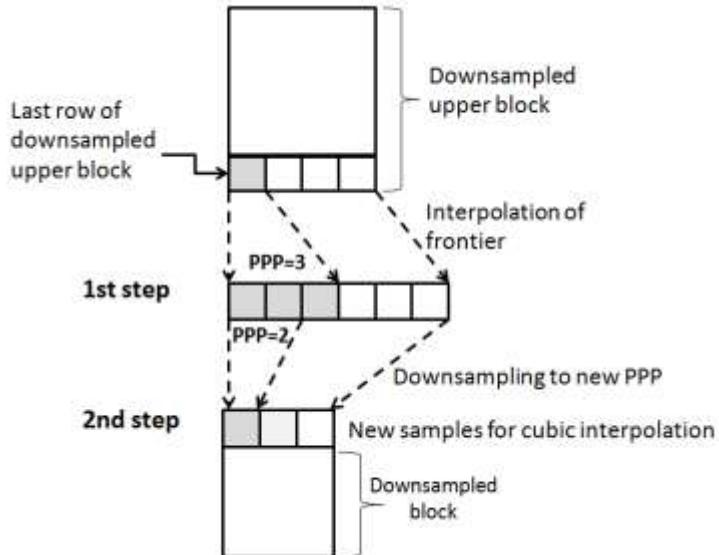


Figura 164. Uso de la frontera horizontal en la interpolación cúbica vertical

La solución es equivalente tanto para la fila superior al bloque como para la fila posterior al bloque, involucrando en un caso el bloque superior y en el otro caso el bloque inferior. De este modo se obtienen las muestras que necesitamos para interpolar.

- La fila superior, es algo que ya se ha calculado en un paso anterior del decodificador, pues es necesario calcular las fronteras para interpretar los hops, por lo tanto es algo que ya se tiene hecho.
- En cuanto a la fila posterior es lo único nuevo que se tiene que hacer. Este proceso de interpolación se debe hacer por vecino cercano, al igual que se hizo con la frontera para la interpretación de hops.

Con esta estrategia se comete cierto error pues aunque se están realineando las muestras de dos bloques adyacentes, se puede estar usando una muestra errónea en los extremos del scanline horizontal, debido a que generamos la muestra por vecino cercano. Es imposible escoger una muestra correcta en los extremos pues para ello primero tendríamos que interpolar entre bloques y ya hemos visto que es imposible (Figura 159). Por ello se usa la aproximación de vecino cercano para estas muestras, cuyo valor en general no va a distar mucho del valor correcto.

Otra posibilidad es usar los valores de las muestras de la frontera del propio bloque como valores de las muestras no disponibles pero se ha verificado que en ese caso el error cometido es mayor.

Una vez interpolado verticalmente, se realiza la interpolación horizontal del bloque. En esta ocasión no es necesario interpolar las fronteras verticales y sub-muestreárlas. Basta con interpolarlas las fronteras hasta la longitud vertical del bloque, pues ahora se parte de un bloque interpolado verticalmente.

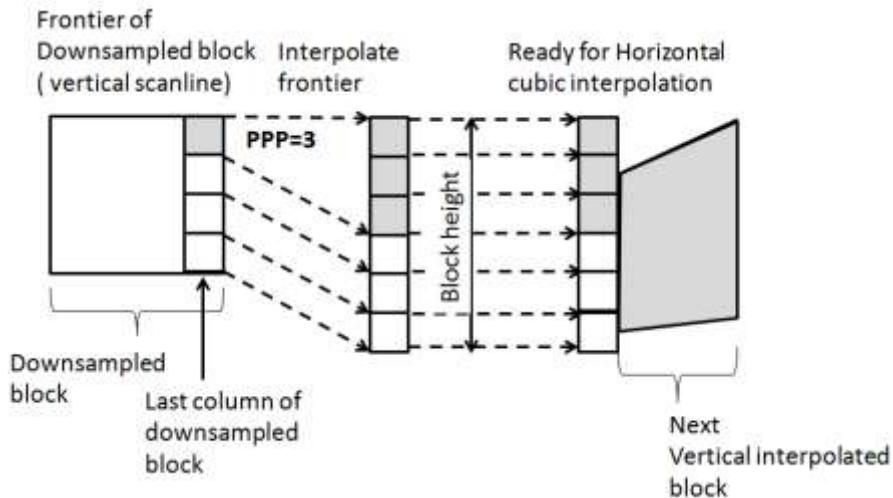


Figura 165. Uso de la frontera vertical en la interpolación cúbica horizontal

Análogamente a lo que sucede con la interpolación vertical, en esta interpolación horizontal:

- La columna anterior (la frontera vertical del bloque izquierdo) es algo que ya se ha calculado en un paso anterior del decodificador, pues es necesario calcular las fronteras para interpretar los hops, por lo tanto es algo que ya se encuentra hecho.
- En cuanto a la columna posterior es lo único nuevo que se tiene que hacer. Este proceso de interpolación se debe hacer por vecino cercano, al igual que se hizo con la frontera para la interpretación de hops.

Tras interpolar los bloques, el resultado es una imagen con costuras entre bloques pendientes de ser interpoladas. Para interpolar las costuras se usa la misma estrategia descrita que para la interpolación bilineal. Primero lo se hace en horizontal y luego en vertical (en realidad da igual el orden). La diferencia esta vez es que como para realizar una interpolación cúbica se necesitan 4 muestras equiespaciadas, no sirve con escoger los dos últimos pixels coloreados del primer bloque y usarlos junto con los dos primeros pixels coloreados del siguiente, ya que entre ambos grupos hay una gran distancia mientras que entre cada par la distancia es de un pixel. Se debe emplear la siguiente estrategia para la interpolación vertical:

- C1 toma el color del pixel en la coordenada x de la penúltima fila del bloque sub-muestreado interpolada. Su distancia con C2 es  $\frac{PPP_y}{2}$
- C2 y c3 son los pixels ya coloreados en la coordenada x. Están espaciados entre sí la suma de  $\frac{PPP_y}{2}$  del bloque superior y  $\frac{PPP_y}{2}$  del interior. No son iguales pero son distancias similares.
- C4 debe ser el color en la coordenada x de la segunda fila del bloque sub-muestreado interpolada. Está separada  $\frac{PPP_y}{2}$  del bloque superior

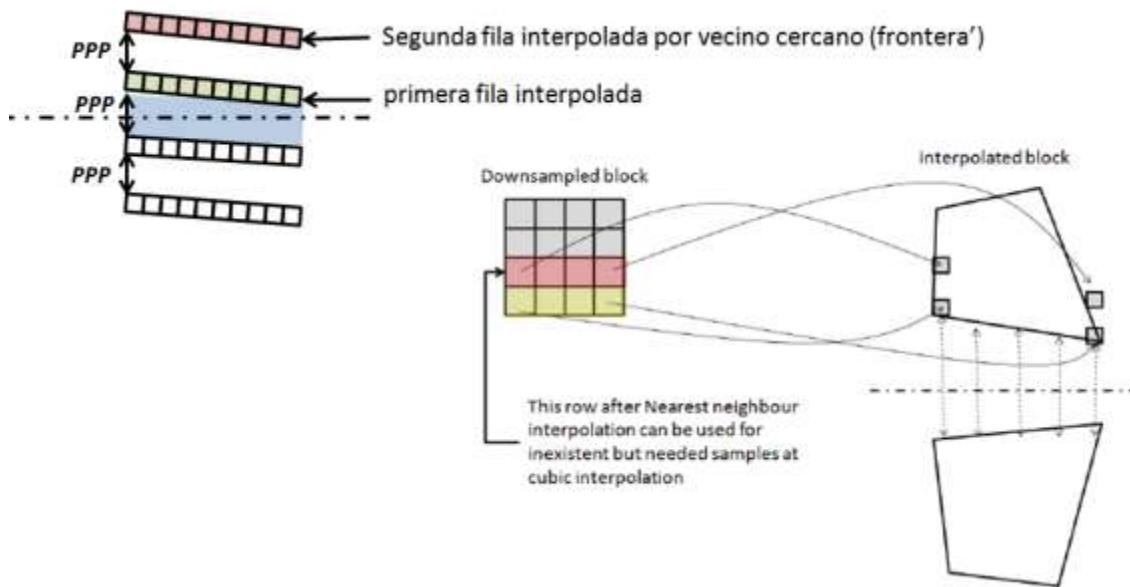


Figura 166. Interpolación cúbica vertical de costuras

En la figura se puede apreciar como gracias a la interpolación por vecino de la penúltima fila se puede usar una muestra para la interpolación cúbica entre los dos bloques que en principio no estaba disponible.

Como ya se tienen las fronteras inferiores horizontales interpoladas del paso de interpretación de hops, y también se tienen las superiores, necesarias para la interpolación cúbica del bloque, lo único nuevo que se necesita es interpolar por vecino cercano una fila más, tanto en la parte superior del bloque como en la inferior.

Para la interpolación de costuras horizontal aplica el mismo razonamiento. El ejemplo que se ha mostrado es el de la interpolación vertical de costuras pero es exactamente igual.

Como último detalle, si se realiza la interpolación horizontal y después la vertical (así se ha implementado en esta tesis), puede quedar algún hueco sin llenar (ver Figura 160). La estrategia a aplicar es exactamente la misma explicada en la interpolación bilineal. Se realiza una interpolación vertical “a tramos” y cuando se encuentre un tramo sin llenar donde el principio y el final pertenezcan al mismo bloque, se realiza una interpolación usando las muestras adyacentes disponibles. Solo cuando nos encontramos en el área entre ambos bloques se usará la estrategia presentada en la Figura 160.

La interpolación elástica bicúbica requiere una solución a la ausencia de muestras necesarias en los límites del bloque, la cual he realizado mediante interpolaciones por vecino. También es necesario un tratamiento especial de la interpolación de costuras que requiere la interpolación por vecino de una fila adicional tanto en la parte superior del bloque como en la inferior. Estas soluciones aportadas son parte del proceso de interpolación elástica, el cual es una contribución de esta tesis

#### 5.7.4 Tiempos de ejecución y paralelización

EL número de operaciones a realizar es lineal con el número de píxeles en los tres tipos de interpolación en los tres tipos de interpolación considerados. Las siguientes gráficas están obtenidas sin ningún tipo

de paralelización para reflejar el tiempo consumido por las operaciones involucradas.

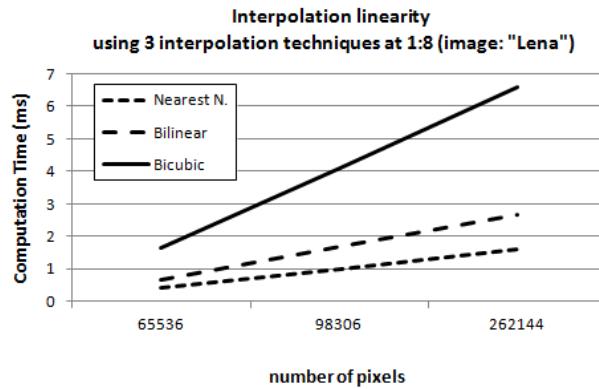


Figura 167. Linealidad del proceso de interpolación

Sin embargo, para cualquier ratio de compresión, la interpolación bicúbica es significativamente más costosa que las de vecino cercano y bilineal tal como se muestra en la siguiente grafica comparativa

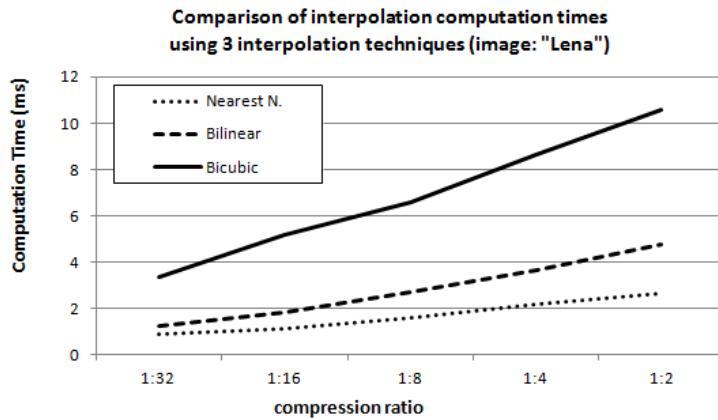


Figura 168. Comparativa de tiempos de ejecución

En cuanto a la calidad, los mejores resultados se consiguen con la interpolación bicúbica. La interpolación bilineal resulta ser un compromiso entre velocidad y calidad, pues posee una calidad cercana a la interpolación bicúbica y una velocidad similar a la de vecino cercano. No obstante, si es posible permitir cierta inversión en algo más de cálculo la mejor opción siempre es la interpolación bicúbica.

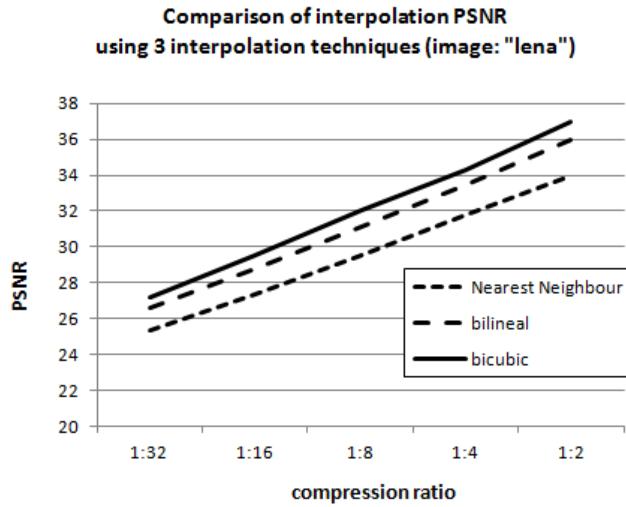


Figura 169. Comparativa de calidad

Respecto la capacidad de paralelización, Cada bloque es completamente independiente y en consecuencia podemos interpolar todos los bloques en paralelo, en un solo paso.

En el caso de las interpolaciones bilineal y bicúbica, tras la interpolación de bloques hay que dar un paso más de interpolación de costuras. Sin embargo este es un paso de bajo coste que además cuesta menos cuanto más ha costado la interpolación. Es decir, las costuras serán más estrechas cuanto menos comprimida esté la imagen y en ese caso cuesta más interpolarla, pero cuesta menos llenar sus finas costuras.

## 5.8 Complejidad y Paralelización

En la siguiente figura se ha dividido el diagrama de bloques de LHE avanzado en varios macro-bloques funcionales en función de su complejidad algorítmica.

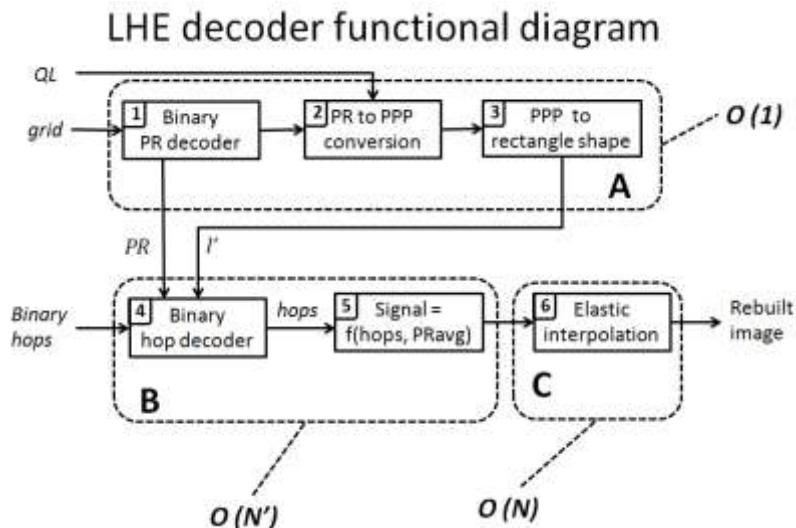


Figura 170. Macrobloques de LHE avanzado en función de su complejidad algorítmica

- **Decodificación de información de malla:** para una imagen de N pixeles implica una complejidad  $O(1)$  ya que la malla posee el mismo número de datos independientemente del tamaño de la imagen. En concreto se ha definido una malla de 32 bloques en el lateral mayor (ancho o alto). Esto significa que en el peor de los casos hay que procesar  $32 \times 32 = 1024$  datos. El número de pasos para cada código binario de la malla es 3, primero decodificar los cuantos de PR, luego transformar a PPP y por último ajustar a forma rectangular.

Se puede paralelizar la transformación de cuantos en PPP de todos a la vez y el ajuste a forma rectangular se puede realizar a la vez en todos los bloques, por lo que son 3 pasos que se ejecutan secuencialmente pero cada paso se puede paralelizar al máximo.

- **Decodificación de hops y transformación a valor de señal:** el número de operaciones que involucra tanto el decodificador de hops como la interpretación de los mismos para recuperación de la señal es  $O(N')$  siendo  $N'$  el número de muestras de la imagen sub-muestreada, el cual puede ser muy inferior al número de muestras originales  $N$ . Esta fase es paralelizable con un mínimo de  $2K-1$  pasos siendo  $K$  el número horizontal de bloques. Esto es debido a la necesidad de interpolar las fronteras para seguir interpretando hops.
- **La interpolación elástica** conlleva un número de operaciones que depende directamente del número de píxeles originales de la imagen. Implica tres pasos: interpolar los bloques, interpolar las costuras verticales y por último interpolar las costuras horizontales.

Los tres pasos de esta fase se deben dar secuencialmente, pero cada uno es paralelizable al 100%

Macrobloque	complejidad	Pasos con máxima paralelización
A	$O(1)$	3
B	$O(N')$	Usando fronteras: $1 + 2K - 1$ Sin usarlas: $2$ (Símbolos->hops + hops->señal)
C	$O(N)$	3 (vertical + horizontal + costuras)

Tabla 21. Complejidad y paralelización de cada macrobloque funcional

A partir de esta tabla tenemos:

- La posibilidad más rápida de peor calidad: 8 pasos ( $3 + 2 + 3 = 8$ )
- La posibilidad más lenta de mejor calidad:  $3 + 2K + 3 = 6 + 2K = 6 + 2 \cdot 32 = 70$  pasos

Es decir, un orden de magnitud de diferencia en tiempo de cálculo si se dispone de los suficientes recursos para paralelizar al máximo.

## 5.9 Resumen del capítulo

En este capítulo he presentado el decodificador LHE avanzado. El objetivo de LHE avanzado es superar las limitaciones de LHE básico, en concreto la de ofrecer la capacidad de escoger el grado de compresión deseado.

Para conseguir este objetivo se ha creado la técnica del downsampling elástico, la cual es una contribución de esta tesis y como complemento necesario he creado la interpolación elástica, la cual es también una contribución de esta tesis.

La definición completa del algoritmo LHE avanzado es una contribución de esta tesis e incluye las siguientes contribuciones adicionales en el decodificador:

- concepto de interpolación elástica separable
- Ecuaciones de interpolación elástica
- Solución a la interpolación de laterales parcialmente cubiertos
- Solución a la interpolación de costuras horizontales y verticales. Estas últimas algo diferentes
- solución para la interpolación cúbica en los límites del bloque basada en generación de muestras usando interpolaciones por vecino
- solución para la interpolación cúbica de costuras



# Capítulo 6

## Video LHE

### 6.1 Presentación del capítulo

En este capítulo se van a describir los algoritmos necesarios para codificar y decodificar video basado en el algoritmo LHE. He decidido presentar ambas partes (codificador y decodificador) en un mismo capítulo porque los componentes del decodificador son parte del codificador, de modo que se puede definir el decodificador como un subconjunto del codificador.

El codificador de video LHE aprovecha la redundancia temporal de la imagen, es decir, codifica los cambios de un fotograma al siguiente. Esta información diferencial temporal va a ser codificada logarítmicamente (usando LHE). Este codificador tiene la ventaja de la velocidad pues LHE es un algoritmo de complejidad lineal.

Una mejora futura (que no es objeto de esta tesis) consistiría en estimar vectores de movimiento basados en las variaciones de relevancia perceptual de las esquinas de la malla. En este caso no se trata de la malla de un fotograma sino de la malla de la información diferencial. Es por ello que la información de relevancia perceptual de las esquinas de la malla describe el movimiento que experimentan los objetos, siendo nula cuando el objeto queda inmóvil. Esta estrategia permitiría estimar vectores de movimiento casi de forma instantánea, sin el alto coste que suele suponer su cálculo pues en los algoritmos convencionales involucra operaciones de convolución. El primer paso para llegar a este “codificador de video LHE avanzado” es diseñar y poner a prueba el codificador de video “básico” que al menos codifique logarítmicamente la información diferencial.

La información diferencial el LHE requiere un cálculo y un tratamiento especial, para que no afecten negativamente:

- Ni los errores de cuantización de LHE
- Ni el downsampling elástico y variable de un frame a otro.

Es por ello que abordar estas dos cuestiones son los primeros pasos para llegar a construir un codificador de video LHE avanzado. La resolución de estos problemas constituye el contenido del presente capítulo.

El siguiente diagrama de bloques resume todo el proceso. A medida que se vaya explicando cada bloque, se presentará este diagrama de nuevo señalando en qué bloque se va a centrar la explicación.

## VideoEncoder/decoder functional block

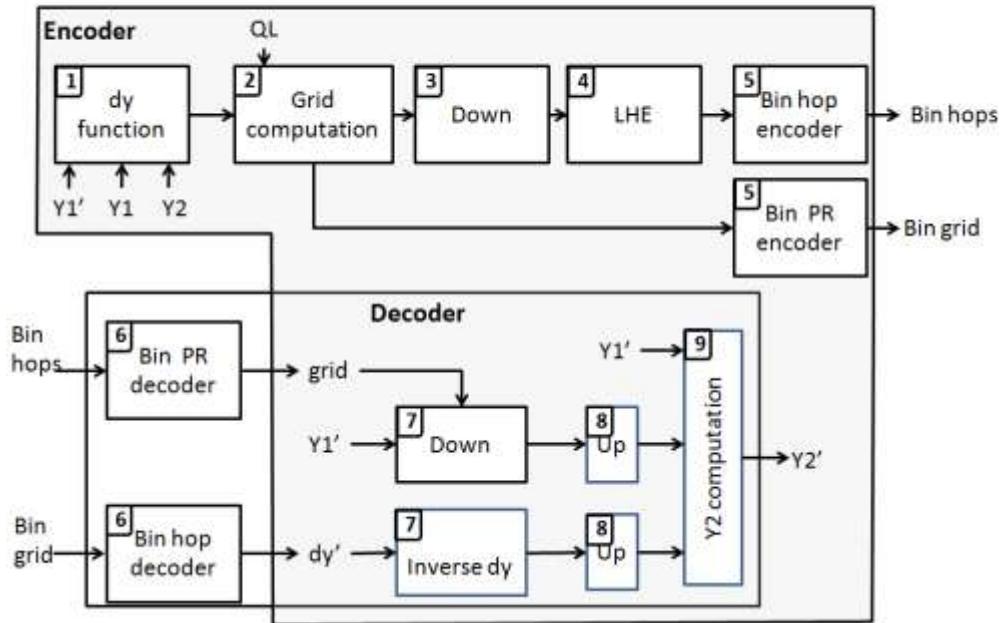


Figura 171 Diagrama de bloques del codificador/decodificador de video

Donde

Y2: nuevo fotograma

Y1: fotograma anterior

Y1': fotograma anterior reconstruido en el decodificador

Y2': nuevo fotograma reconstruido en el decodificador

dy: información diferencial “especial” para LHE

dy': información diferencial cuantizada por LHE y submuestreada

Los pasos en líneas generales son (los números se corresponden con la figura):

- Primero se calcula la información diferencial entre el último fotograma reconstruido y el nuevo. Para calcular el último fotograma reconstruido se necesita ejecutar los módulos del decodificador, ya que no se trata de calcular la diferencia entre el nuevo fotograma Y2 y el anterior Y1 sino la diferencia entre el nuevo Y2 y el ultimo reconstruido Y1'
- Se calcula las métricas de relevancia perceptual de este fotograma diferencial, y se cuantizan en los niveles que define LHE (son 5 cuantos). A continuación se convierten estas métricas cuantizadas (PR') en PPP según el nivel de calidad deseado (QL)
- Se efectúa un downsampling elástico del fotograma diferencial
- Se codifica con hops logarítmicos los bloques sub-muestreados
- Se codifica en binario los hops resultantes así como la información cuantizada de la malla de PR

- Decodificación de los hops y la información de malla. Este paso no es necesario en el codificador pero sí en el decodificador.
- Con la información de la malla, se realiza un downsampling elástico del último fotograma reconstruido. Esto es necesario porque luego se va a sumar información diferencial y no se puede sumar información de diferentes grados de downsampling como se explicará más adelante.
- Interpolación elástica de los valores de la señal al tamaño original, bloque a bloque pues cada bloque tiene un downsampling elástico diferente.
- Calculo del el nuevo fotograma reconstruido Y2'

El diseño del codificador de video LHE básico es una contribución de esta tesis.

La propuesta de cálculo de vectores de movimiento a partir de los datos de malla es otra contribución que deberá ser puesta a prueba como trabajo futuro.

## 6.2 Cálculo de la información diferencial a prueba de LHE

Para ubicar el contenido de este apartado, a continuación se muestra el diagrama de bloques del codificador de vídeo, destacando el módulo en el que se ubica el contenido que se va a desarrollar.

**VideoEncoder/decoder functional block**

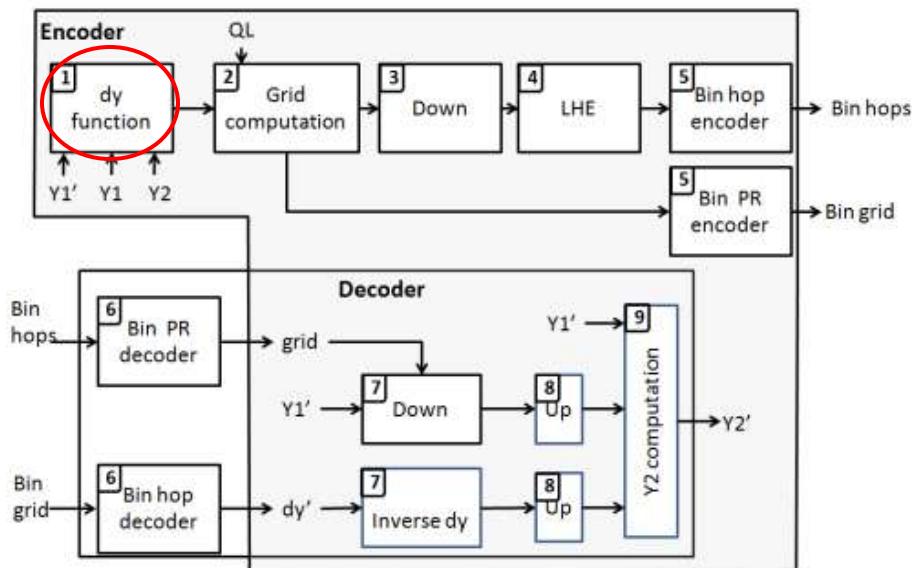


Figura 172. Ubicación del contenido que se va a desarrollar

Dados dos fotogramas consecutivos a los que llamaremos Y1 e Y2, se va a calcular la información diferencial "Dy" como la diferencia de Y2 respecto Y1', siendo Y1' el fotograma correspondiente a la reconstrucción de Y1 por parte del decodificador.

Dy se puede calcular para cada pixel como  $D_y = Y_2 - Y'_1$ , por lo tanto  $D_y \in [-255, 255]$

$Dy$  es una función continua y por lo tanto interpolable. Es lo que en principio necesita el decodificador para tratar de reproducir  $Y_2$  a partir de  $Y_1'$ . Sin embargo no es adecuada para LHE pues las imprecisiones de LHE al codificarla serían transmitidas y suponiendo una secuencia de fotogramas idénticos en los que  $Y_2=Y_1'$ , transmitir un error pequeño en el hop correspondiente a un mismo pixel en cada fotograma (por ejemplo, +2) supondría aumentar la luminancia indefinidamente. Por lo tanto se necesita una función de información diferencial que “resista” las imprecisiones de LHE.

En lugar de “ $Dy$ ”, se va a definir una función llamada “ $dy$ ” con dos características:

- Que permita ser codificada con sólo 8 bits: esto lo haré así para utilizar el cuantizador LHE que actualmente trabaja con señales de rango [0 , 255]
- Que tenga en cuenta las limitaciones de LHE para no codificar errores pequeños de luminancia.

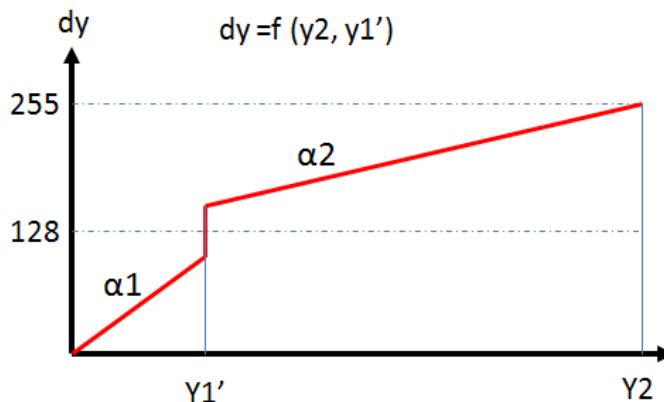


Figura 173. Gráfica de  $dy$

La función  $dy$  es una función a 3 tramos. A continuación se explica el significado de cada uno:

**El primer tramo** permite codificar valores de  $Y_2$  inferiores a  $Y_1'$ . La pendiente de dicho tramo ( $\alpha_1$ ) se ajusta para poder codificar valores entre 0 e  $Y_1'$ . En consecuencia, dicha pendiente siempre estará comprendida en el intervalo [0.5 , 1]. El peor caso es el de mayor imprecisión y es cuando  $Y_1'$  toma el valor 255. En ese caso la pendiente de dicha recta es 0.5. Para valores de  $Y_1'$  por debajo de 128 la precisión es máxima pues la pendiente es 1. No se usarán pendientes superiores a 1 pues ello no aportaría mayor precisión

**El segundo tramo** es vertical. Es lo que hace que  $dy$  sea resistente a los errores de cuantización de LHE. Es una discontinuidad de tamaño  $h_{1\min}$ , es decir de tamaño 4 (+2 hacia arriba y -2 hacia abajo). Esta discontinuidad permite que LHE pueda cometer un error de  $\pm 2$  al codificar  $dy$  sin que tenga consecuencias en el brillo de  $Y_2'$ . Suponiendo que  $Y_2 = Y_1'$  y que  $dy$  se codifica usando LHE y se produce un error de +2. La gráfica de  $dy$  nos indica que mientras  $dy$  no sea superior a  $128+2=130$ , se puede asumir que  $Y_2$  es igual a  $Y_1'$ , lo cual es correcto. Si no se considerase este tramo veríamos parpadeos y fluctuaciones de brillo que se corresponden con los errores de LHE, por eso este tramo vertical es necesario.

**El tercer tramo** permite codificar valores de  $Y_2$  superiores a  $Y_1'$ . La pendiente de dicho tramo ( $\alpha_2$ ) se ajusta para poder codificar valores entre  $Y_1'$  y 255. En consecuencia, dicha pendiente siempre estará comprendida en el intervalo [0.5 , 1]. El peor caso es el de mayor imprecisión y es cuando  $Y_1'$  toma el valor 0. En ese caso la pendiente de dicha recta es 0.5. Para valores de  $Y_1'$  por encima de 128 la precisión es máxima pues la pendiente es 1. No se usarán pendientes superiores a 1 pues eso no aportaría mayor

precisión.

Una condición adicional que se tendrá en cuenta en el cómputo de la función dy es que tras 3 fotogramas en los que  $Y2=Y1$  (es decir,  $Dy=0$ ) se asignará  $dy=128$  aunque  $Y2-Y1'$  no sea nulo. Esto es debido a que el downsampling y la reconstrucción de la información pueden hacer que no solo  $Y2$  sea inalcanzable, sino que lo que se consiga diste mucho de la imagen  $Y2$ . Sin embargo ello ocurre por comprimir intensamente y en esos casos tras 3 refinamientos se debe parar pues debido al intenso downsampling no se consigue mejorar más la calidad de la imagen.

Cada fotograma “dy” se compone de una imagen cuantizada con LHE. Los objetos que permanecen inmóviles “desaparecen” pues la función dy tomará el valor 128 en todos los píxeles de esos objetos y por consiguiente casi todos los hops serán nulos y las métricas de PR darán cero, asignándose el cuarto nulo que forzará a un downsampling extremo y generando tan solo 4 hops por bloque. Cuando el movimiento se detiene, LHE comprime al máximo.

Cuando objetos de muchos detalles se muevan por la pantalla, la información diferencial de las zonas del fotograma donde esos objetos se desplazan será elevada y LHE comprimirá menos dichas zonas que otras donde haya menos movimiento.

La siguiente imagen ilustra el valor que toma la función dy ante el movimiento de los objetos. Los marcadores han desaparecido porque permanecen inmóviles.



Figura 174. Valor de la función dy

La definición de la función dy resistente a los errores de cuantización de LHE y definida en tres tramos es necesaria para poder codificar la información diferencial usando LHE con la mayor precisión en 8 bits por muestra y con tolerancia a los errores de LHE. Representa una contribución de esta tesis

### 6.3 Compresión de la información diferencial

Para ubicar el contenido de este apartado, a continuación se muestra el diagrama de bloques del codificador de vídeo, destacando el módulo en el que se ubica el contenido que se va a desarrollar.

## VideoEncoder/decoder functional block

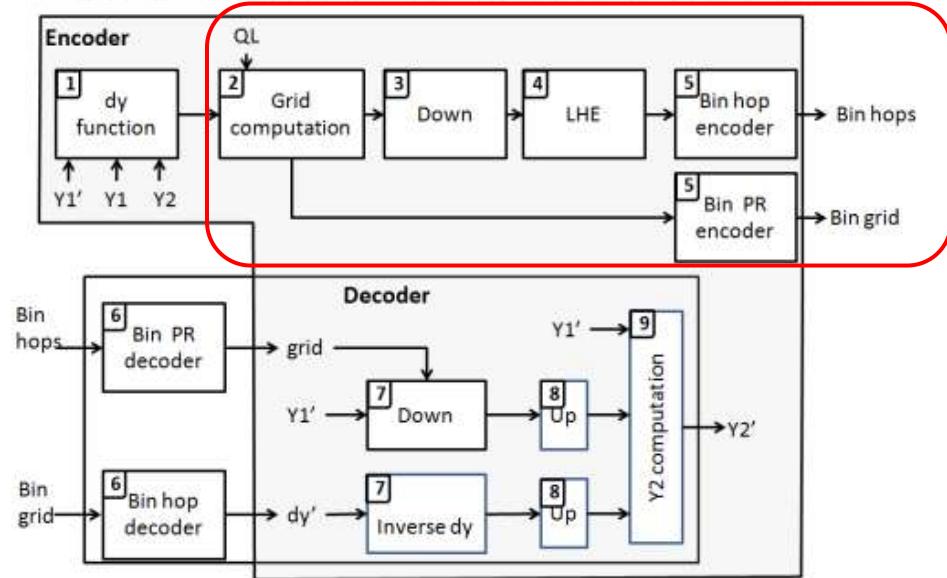


Figura 175. Ubicación del contenido que se va a desarrollar

Lo que se hace en los pasos 2, 3, 4, 5 ya lo hemos visto en el capítulo dedicado a LHE avanzado. Básicamente se trata de codificar la información diferencial “dy” usando LHE. Se ha querido representar en el diagrama todos los pasos (del 2 al 5) para una mejor comprensión del proceso, aunque se podría simplemente resumir todos esos pasos con un bloque llamado “LHE image encoder”.

### 6.4 Regeneración de la información diferencial

Para ubicar el contenido de este apartado, a continuación se muestra el diagrama de bloques del codificador de vídeo, destacando el módulo en el que se ubica el contenido que se va a desarrollar.

## VideoEncoder/decoder functional block

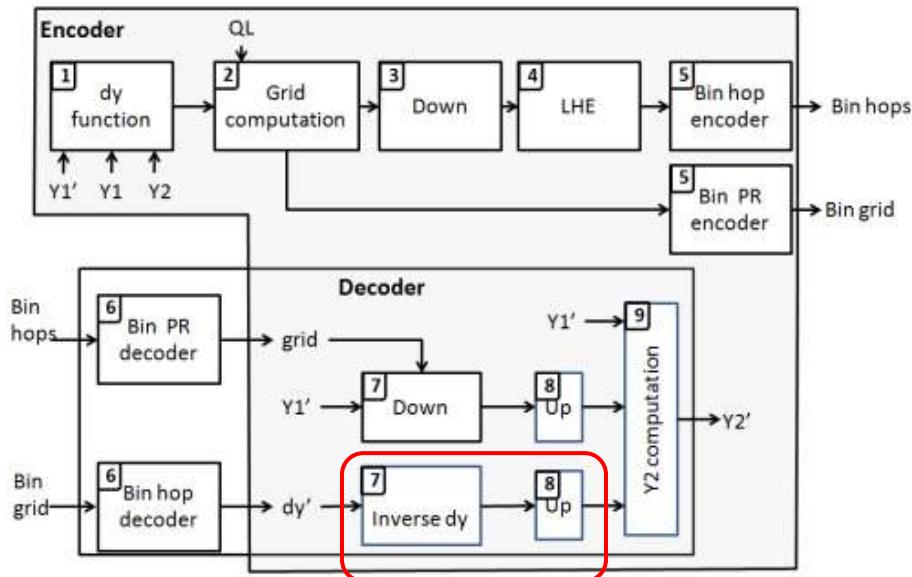


Figura 176. Ubicación del contenido que se va a desarrollar

En el decodificador lo que se tiene como punto de partida es  $Y_1'$  y se debe reconstruir  $Y_2$  a partir de la información diferencial que nos llega. Obviamente no se va a poder regenerar perfectamente  $Y_2$ , y a lo que se obtenga lo denotaremos  $Y_2'$ .

El problema con el que debemos enfrentarnos es que lo que llega no es la función  $dy$ , sino una versión de la función  $dy$  degradada debido al sub-muestreo y a la cuantización LHE, a la que se ha llamado  $dy'$ .

No se puede interpolar la función  $dy'$  porque se trata de una función discontinua. La siguiente grafica de ejemplo muestra como se interpola erróneamente entre dos muestras debido a la discontinuidad

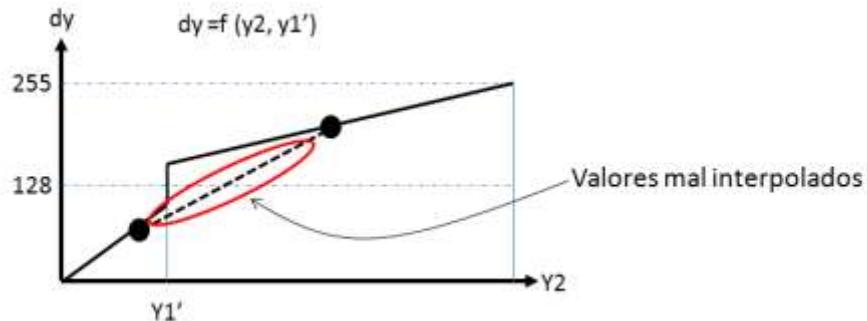


Figura 177. La función  $dy$  no es interpolable

Lo que se ha de hacer es sencillamente transformar la señal sub-muestreada  $dy'$  en una señal continua, es decir, en una versión sub-muestreada de lo que equivaldría a la función  $Y_2$ . A todo par de valores ( $dy$ ,  $Y_1'$ ) le corresponde un valor  $Y_2$  y podemos calcular dicha traducción.

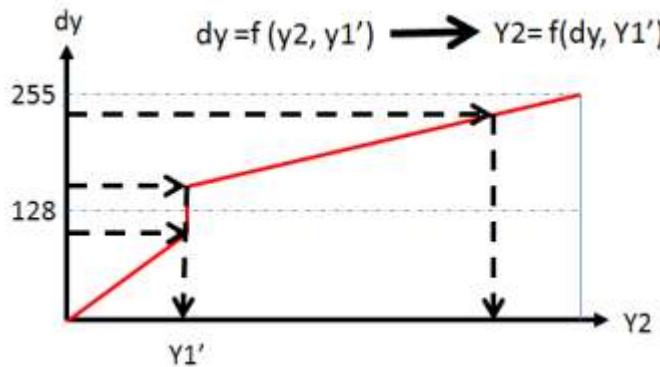


Figura 178. La función inversa de dy

El resultado de esta función inversa no va a ser  $Y_2$  por que se ha partido de la versión sub-muestreada y cuantizada de  $dy$ , es decir, se ha partido de  $dy'$ . Por lo tanto a lo que se ha calculado lo vamos a llamar  $Y_2''_{down}$  y tras interpolarla, se obtendrá  $Y_2''$

Cuando no hay movimiento  $Y_2=Y_1$  y la función  $dy$  no contiene los objetos inmóviles. Por ello se debe calcular el fotograma  $Y_2'$  teniendo en cuenta tanto los objetos que se han movido y que aparecen en  $Y_2''$  como los objetos que han permanecido inmóviles y que se encuentran en  $Y_1'$ .

La forma de combinar estas variables para obtener  $Y_2'$  se explica a continuación

## 6.5 Cómputo del nuevo fotograma

Para ubicar el contenido de este apartado, a continuación se muestra el diagrama de bloques del codificador de vídeo, destacando el módulo en el que se ubica el contenido que se va a desarrollar.

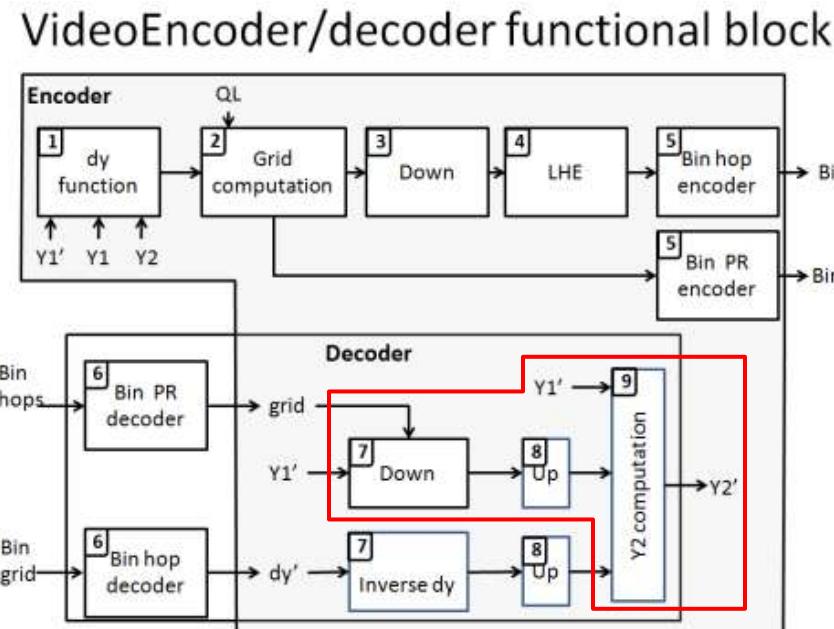


Figura 179. Ubicación del contenido que se va a desarrollar

Antes de exponer el procedimiento de cálculo de  $Y2'$  se va a exponer el problema al que nos enfrentamos.

La función  $dy'$  no tiene la resolución de  $Y1'$  ni está referida a los valores de luminancia de  $Y1'$  sino a los valores de luminancia de una  $Y1''$  con un nivel de downsampling en cada bloque correspondiente al nuevo fotograma. Si se quiere obtener esta  $Y1''$  debemos hacer downsampling de  $Y1'$  con la información de malla del fotograma  $dy'$  y a continuación se interpola.

Ahora se va a ilustrar con un ejemplo lo que ocurre cuando se suma una función interpolada  $dy$  a un fotograma  $Y1'$  que no tiene los mismos escalados que  $dy$ . Se va visualizar con dos bloques que han sido sub-muestreados a diferente intensidad y posteriormente interpolados por la técnica de vecino cercano. Imaginemos un objeto con un borde definido. El objeto se desplaza y en consecuencia la función  $dy$  contiene cambios pero no tienen porqué tener exactamente la misma resolución. Lo mismo ocurre si empieza a aumentar el brillo pero el objeto no se mueve. La función  $dy$  no posee bordes y se va a sub-muestrear intensamente. Supongamos que a partir de  $dy'$  calculamos  $Dy'$  como  $Dy'=Y2''-Y1'$ . Veamos el resultado:

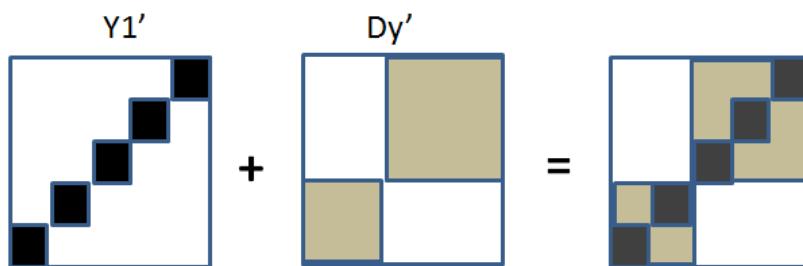


Figura 180. Suma de señales de distinto nivel de sub-muestreo

El resultado tiene mezcla de resoluciones. Y ello genera manchas y “estelas” en los objetos animados en la imagen resultante. La conclusión es que no se pueden sumar funciones de diferente resolución. Con esto definimos la primera regla para el cálculo:

- No sumar informaciones de diferente resolución

Para sumar  $Dy'$  primero debemos ajustar  $Y1'$  a la resolución de  $dy'$ , y después se podrán sumar  $Y1''=UP(DOWN(Y1'))$ , y podemos calcular  $Dy'$  como  $Dy'=Y2''-Y1'$

Sin embargo si siempre calculamos  $Y1''$  y sumamos  $dy'$  entonces los elementos inmóviles se degradarán pues no poseen detalle en la función  $dy$ . Esto lleva a la segunda regla del cálculo:

- Preservación de la información inmóvil

Debemos preservar  $Y1'$  cuando  $dy$  sea 128, es decir en los casos en los que  $Y2''$  sea igual a  $Y1''$  debemos igualar  $Y2'=Y1'$

Hay dos tipos de cambios que podemos observar: movimientos y cambios de brillo. En el caso de cambios de brillo,  $Y2''$  no va a ser igual a  $Y1'$  pero  $Y2''$  no contiene los detalles de  $Y1'$ , tan solo sus incrementos/disminuciones de brillo. Sería un error igualar  $Y1'=Y2''$  cuando  $dy$  sea pequeña. Esto nos lleva a la tercera regla:

- Estrategia ante pequeños cambios de brillo

En los casos en los que  $dy'$  sea cercana a 128, emplearemos una combinación lineal de  $Y1'$  y de  $Y1''+dy'$  para calcular el valor de  $Y2'$ . De este modo no se perderán los detalles de  $Y1'$  pero tendremos en cuenta los pequeños cambios presentes en  $dy'$

La determinación de los coeficientes de la combinación lineal ha sido experimental. Este pseudocódigo resume el proceso y en él figuran los valores de los coeficientes

*//la grid tiene la misma resolución que Y1' o al menos no es destructiva en este pixel*

$Dy'=Y2''-y1''$

IF  $Y1'=Y1''$  THEN  $Y2=Y1''+DY'$  (o lo que es lo mismo,  $Y2'=Y2''$ )

*//la resolución de la grid es distinta a la de Y1', mayor o menor. 3 casos*

*//cambio minúsculo o nulo (umbral u1=1)*

ELSE IF  $Dy' \leq u1$  THEN  $Y2'=Y1'$

*//cambio pequeño. Preservamos el detalle de Y1' (u2=4) pero aplicamos los cambios de brillo*

ELSE IF  $Dy' \leq u2$  THEN  $Y2'=f * (Y1''+Dy') + (1-f) * Y1'$  siendo  $f=(Dy'-u1)/(u2-u1)$

*//cambio grande. Hay movimiento o intenso cambio de brillo intenso. Aplicamos Dy' sobre Y1''*

ELSE  $Y2'=Y1''+Dy'$  (o lo que es lo mismo  $Y2'=Y2''$ )

Aparentemente el primer caso y el último son iguales pero no es así. En el primer caso  $Y1'$  es igual a  $Y1''$  pero  $Dy'$  puede ser pequeño, sin embargo en ese caso no nos interesa entrar por la rama de la combinación lineal y por eso lo tenemos en cuenta al principio.

El downsampling elástico en un vídeo genera el problema de la suma diferencial de información de diferente resolución. Dicho problema ha sido resuelto con la estrategia expuesta en esta sección, basada en tres reglas:

- No sumar informaciones de diferente resolución,
- Preservar la información inmóvil, y
- Estrategia ante pequeños cambios de brillo basada en una combinación lineal de la información diferencial y la información presente en el fotograma anterior.

La forma de solucionar el problema de la suma diferencial de información con downsampling variable es una contribución de esta tesis y forma parte del algoritmo de codificación de video LHE básico

## 6.6 Complejidad y Paralelización

En la siguiente figura se muestra la estructura del codificador de video LHE agrupando los bloques que constituyen el codificador de imagen fija LHE y eliminando los decodificadores binarios que solo debe ejecutar el decodificador de vídeo, pues el codificador ya tiene los hops decodificados.

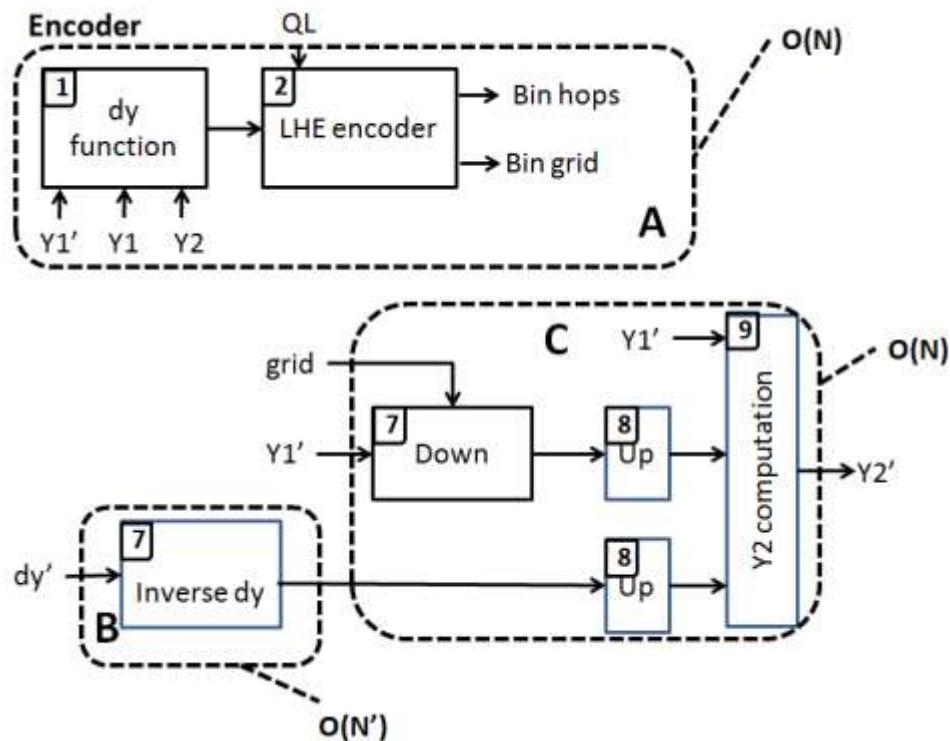


Figura 181. Macro-bloques del codificador de video LHE en función de su complejidad algorítmica

- Bloque A: La función dy y el codificador LHE son componentes de complejidad  $O(N)$ . puesto que el codificador (ya visto en anterior capítulo) requiere 70 pasos, se obtienen 71 pasos, pues la función dy es paralelizable y se requiere un único paso. En video interesa un LHE sin fronteras.
- Bloque B: la función inversa se ejecuta sobre la señal  $dy'$  que posee  $N'$  pixels, siempre menos que  $N$  pues es una señal sub-muestreada. En el caso del video normalmente será  $N' \ll N$ . Esta función es paralelizable y se ejecuta en un solo paso.
- Bloque C: comprende los bloques 7, 8 y 9, siendo cada uno paralelizable y de complejidad  $O(N)$ . Las interpolaciones tienen 3 pasos cada una porque requieren la interpolación de costuras tras la interpolación de bloques, de modo que en total se contabilizan 5 pasos.

En total se obtienen 75 pasos porque la función inversa y el downsampling de  $Y1'$  se pueden ejecutar a la vez.

Macrobloque	complejidad	Pasos con máxima paralelización
A	$O(N)$	$1+70 = 71$
B	$O(N')$	1
C	$O(N)$	5 (uno de ellos en paralelo con B)

Tabla 22. Complejidad y paralelización de cada macrobloque funcional

Si se consideran los pasos que involucra el codificador de imagen fija (70) y los del codificador de video (75) podemos afirmar que se puede codificar un fotograma prácticamente a la misma velocidad que se codifica una imagen fija.

## 6.7 Resumen del capítulo

En este capítulo he presentado el codificador/decodificador de vídeo LHE básico. El objetivo de este codificador es solucionar los problemas inherentes a la cuantización LHE y al downsampling elástico para poder utilizar información diferenciar temporal.

Para ello, la solución comprende dos partes fundamentales:

- Se ha definido una función dy resistente a los errores de cuantización de LHE
- Se han definido 3 reglas de cómputo del nuevo fotograma a partir de la información diferencial:
  - No sumar informaciones de diferente resolución,
  - Preservar la información inmóvil, y
  - Estrategia ante pequeños cambios de brillo basada en una combinación lineal de la información diferencial y la información presente en el fotograma anterior

El resultado ha sido un codificador de video capaz de codificar fotogramas a la misma velocidad que el codificador de imagen fija (casi el mismo número de pasos).

El codificador construido no pretende ser la solución definitiva, sino un primer paso en la construcción de un codificador de video LHE avanzado, cuya ventaja fundamental sería la inclusión del cálculo de vectores de movimiento a partir de la información de malla, evitando operaciones de convolución y por lo tanto evitando el freno que suelen suponer dichos cálculos en el estado del arte. Este diseño se contempla como trabajo futuro.

# Capítulo 7

## Resultados

### 7.1 Presentación del capítulo

En este capítulo se muestran resultados de calidad de codificación de imagen fija, llevados a cabo con imágenes estándar de referencia.

Las imágenes estándar de referencia son imágenes usadas por muchas instituciones para poner a prueba algoritmos de procesamiento y compresión de imágenes. Las imágenes en algunos casos están escogidas por representar imágenes típicas o naturales con las que cualquier procesamiento de imágenes debería enfrentarse. En otros casos las imágenes representan diferentes retos para los algoritmos de compresión por contener texturas con detalles muy finos o transiciones abruptas, bordes, regiones uniformes, etc.

Dos de las galerías de imágenes más reconocidas para esta labor son:

- **Kodak Lossless True Color Image Suite** [61]: 24 imágenes fotográficas con diferentes motivos de tamaños 768x512 y 512x768
- **USC-SIPI Image Database** [63]: consiste en 4 galerías de imágenes entre las que se encuentra la galería “mísceleña”, que contiene las famosas imágenes de “lena”, “baboon”, “peppers”, etc. Las resoluciones de las imágenes son variadas, 256x256, 512x512 y 1024x1024

Como método de medida de la calidad se ha escogido dos: el PSNR (Peak Signal to Noise Ratio) y el SSIM. De estas dos medidas la más conocida es PSNR si bien SSIM presenta una mayor similitud con el HSV (Human Vision System) [48]

- PSNR es método más simple y mayormente usado para medir la calidad. Es una buena medida para comparar diferentes resultados con una misma imagen pero comparaciones entre PSNR de diferentes imágenes es algo que no tiene significado. Una imagen con 20dB PSNR puede tener una apariencia mucho mejor que otra con 30dB PSNR. Esto significa que las gráficas de PSNR promediadas para cada conjunto de imágenes no tienen significado “claro” por si mismo aunque **si lo tiene la comparación entre gráficas realizadas con diferentes compresores**.
- En cuanto a la métrica SSIM se ha utilizado la implementación del proyecto “kanzi” [49](Compression and graphic processing utility classes in Java ) disponible en GitHub.

Para cada uno de los dos conjuntos de imágenes, primeramente se presentarán las imágenes que componen la galería y a continuación gráficas de resultados y las tablas de resultados con las que se han construido las gráficas

Todos los resultados han sido realizados sobre versiones en blanco y negro de las imágenes, ya que el color no lo he implementado en esta versión mejorada del codificador. En la primera versión del codificador (sin downsampling elástico) se implementó el color para validar el modelo YUV sobre LHE pero en esta segunda versión no está disponible por el momento.

Como conclusiones se puede decir que:

- LHE proporciona una calidad intermedia entre JPEG y JPEG2000 mientras que a medios y altos bit-rates es similar en calidad a JPEG.
- Añadida a esta ventaja se encuentra su velocidad, superior a JPEG y JPEG2000 debido a su menor complejidad.
- Otra interesante característica de LHE es la posibilidad de alcanzar tasas de bit de 0.05bpp, alcanzables por JPEG2000, pero no por JPEG.
- Por último, tras analizar visualmente las diferencias con JPEG2000, se aprecia como los bordes en LHE están más difuminados que en JPEG2000, debido a los efectos del downsampling y la interpolación. Es esperable que LHE pueda mejorar en este aspecto mediante mecanismos de downsampling usando la mediana en lugar de la media y mecanismos avanzados de interpolación, mediante los que se puedan reconstruir bordes abruptos incluso con fuertes compresiones, con estrategias similares a los algoritmos “HQX” o “XBR” [65] aunque deberían ser adaptados o reformulados para poder actuar de forma “elástica”.

## 7.2 Galería Kodak Lossless True Color Image Suite

### 7.2.1 Imágenes de la galería





kodim07.bmp



kodim08.bmp



kodim09.bmp



kodim10.bmp



kodim11.bmp



kodim12.bmp



kodim13.bmp



kodim14.bmp



kodim15.bmp



kodim16.bmp



kodim17.bmp



kodim18.bmp



## 7.2.2 Gráficas de resultados PSNR de la galería Kodak

Rate Distortion for Kodak Lossless Truecolor Image Suite

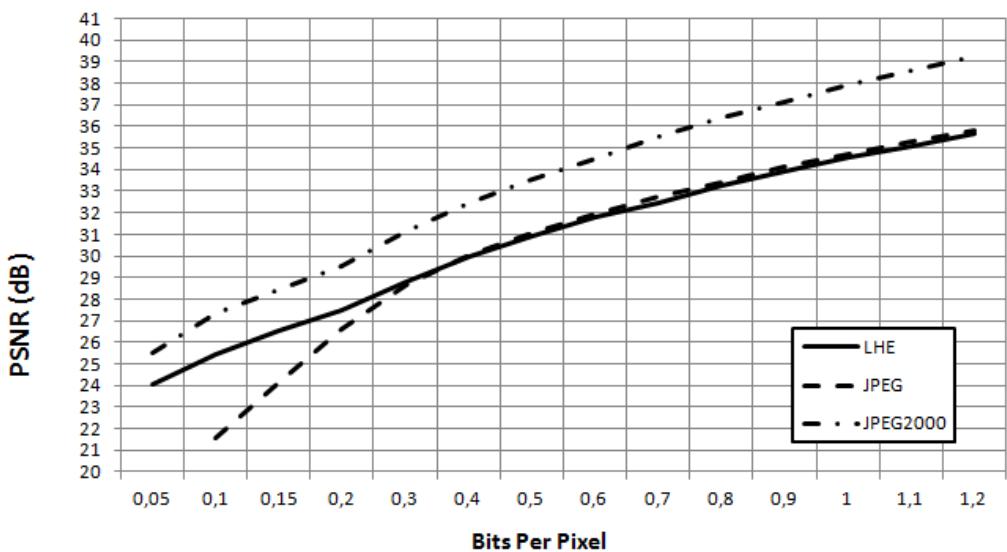


Figura 182. Diagrama de distorsión PSNR para galería Kodak

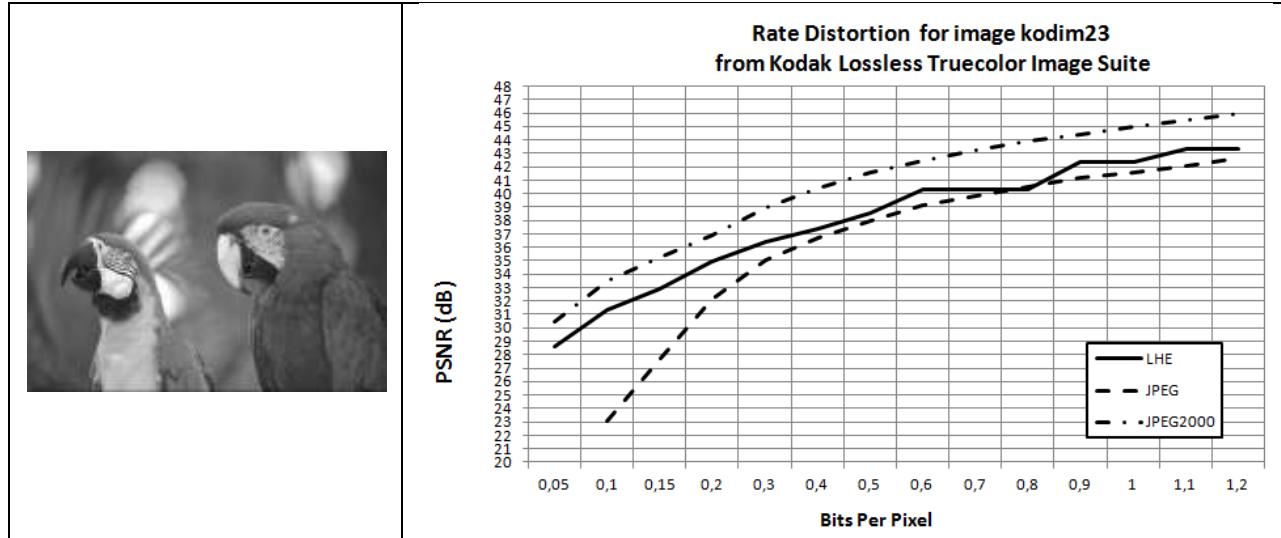


Figura 183. Diagrama de distorsión PSNR para imagen “kodim23”

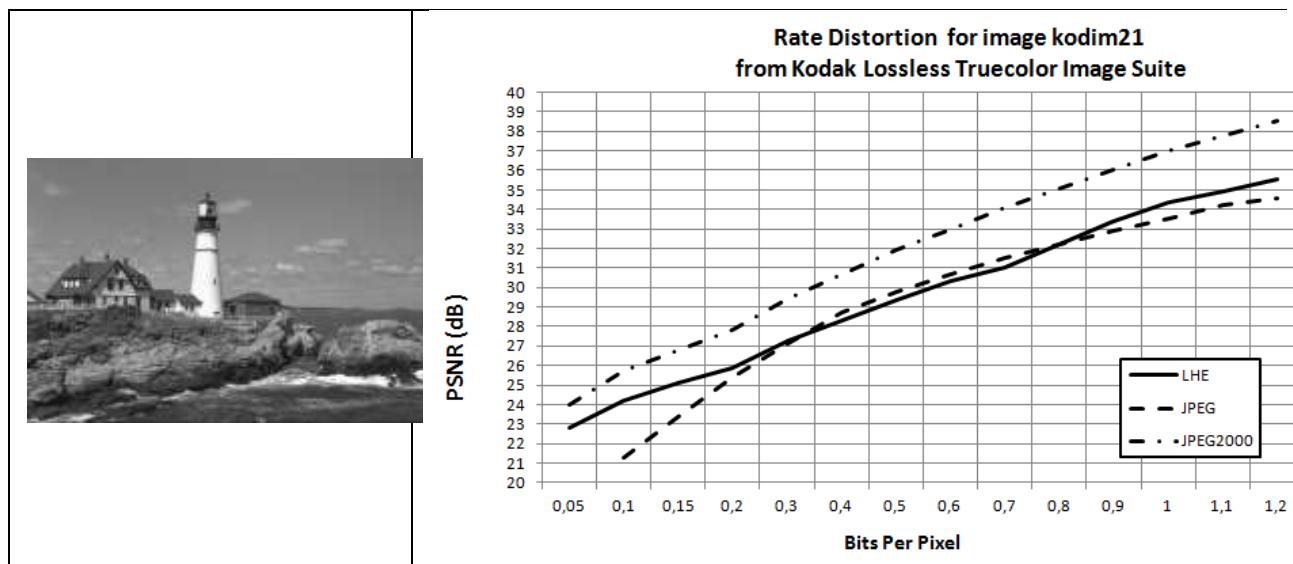


Figura 184. Diagrama de distorsión PSNR para imagen “kodim21”

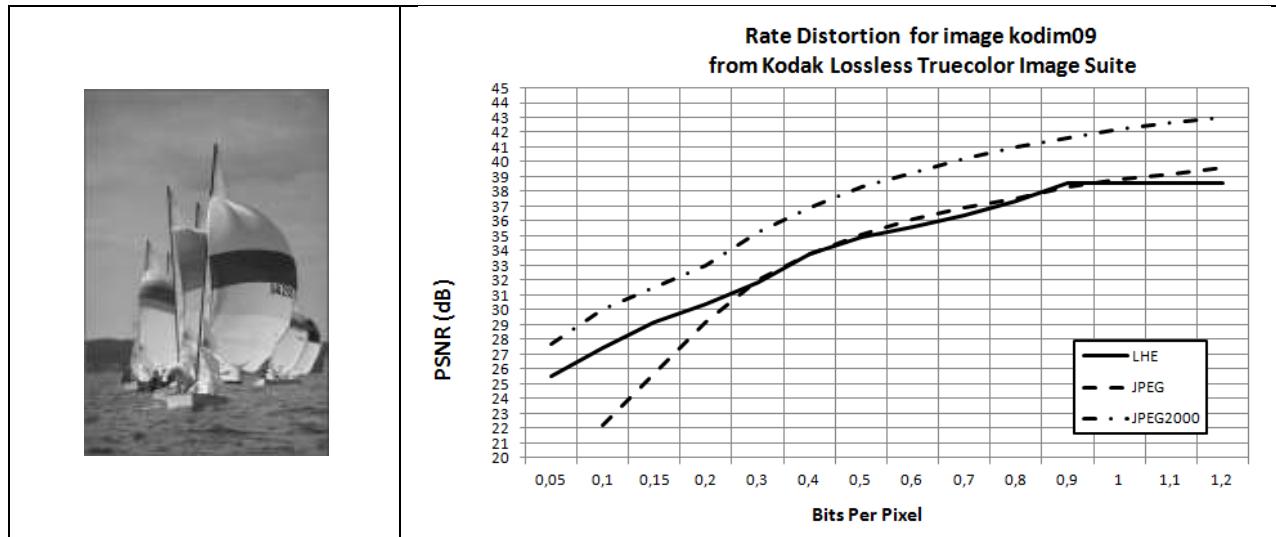


Figura 185. Diagrama de distorsión SSIM para imagen “kodim09”

### 7.2.3 Tablas de resultados PSNR de la galería Kodak

Image	size	bits per pixel (bpp)													
		0,05	0,1	0,15	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1	1,1	1,2
kodim01	768x512	20,31	21,16	21,96	22,53	23,36	24,27	24,97	25,73	26,28	26,83	27,45	28,04	28,74	29,21
kodim02	768x512	28,45	29,56	30,57	31,60	32,57	33,36	33,82	34,91	35,58	36,37	36,37	37,34	37,34	38,59
kodim03	768x512	28,04	29,74	30,81	32,22	33,82	34,91	36,37	37,34	37,34	38,59	38,59	40,35	40,35	40,35
kodim04	512x768	26,83	28,45	30,14	31,06	32,57	33,36	34,33	35,58	35,58	36,37	37,34	37,34	38,59	38,59
kodim05	512x768	19,90	20,55	21,54	22,39	23,23	24,12	24,67	25,65	26,46	27,02	27,68	28,31	28,89	29,74
kodim06	768x512	22,43	23,67	24,33	25,03	26,12	27,13	28,04	28,89	29,56	30,35	30,81	31,60	32,22	32,95
kodim07	768x512	23,96	25,37	26,64	28,17	30,35	32,22	33,82	34,91	36,37	37,34	37,34	38,59	38,59	40,35
kodim08	768x512	18,19	18,74	19,40	20,37	21,21	22,12	22,60	23,72	24,33	24,85	25,37	26,12	26,55	27,02
kodim09	512x768	25,51	27,45	29,21	30,35	31,90	33,82	34,91	35,58	36,37	37,34	38,59	38,59	38,59	38,59
kodim10	512x768	25,65	27,23	28,89	29,94	31,60	32,95	34,33	34,91	35,58	36,37	37,34	37,34	38,59	38,59
kodim11	768x512	23,49	24,91	25,73	26,55	27,80	28,89	29,74	30,57	31,32	32,22	32,95	33,82	34,33	34,91
kodim12	768x512	26,73	28,89	30,14	31,06	32,22	33,36	34,33	34,91	35,58	36,37	37,34	37,34	38,59	38,59
kodim13	768x512	19,45	19,84	20,50	21,11	21,87	22,53	22,99	23,40	24,44	24,97	25,30	25,80	26,46	26,64
kodim14	768x512	22,46	23,91	24,79	25,51	26,83	27,56	28,59	29,38	30,14	30,81	31,32	31,90	32,95	33,36
kodim15	768x512	25,73	27,13	28,45	29,21	31,06	31,90	32,95	33,36	34,33	34,33	34,91	35,58	36,37	37,34
kodim16	768x512	26,64	27,56	28,45	29,38	30,35	31,60	32,57	33,36	34,33	34,91	35,58	36,37	37,34	
kodim17	512x768	25,37	27,13	28,59	29,56	31,06	32,57	33,36	34,91	34,91	36,37	36,37	37,34	37,34	38,59
kodim18	512x768	22,39	23,49	24,27	25,03	26,20	27,13	28,04	29,21	29,74	30,57	31,06	31,90	32,57	32,95
kodim19	512x768	22,79	24,33	25,58	26,37	27,92	29,05	29,94	31,06	31,90	32,95	33,82	34,33	34,91	35,58
kodim20	768x512	24,73	26,73	27,92	29,56	31,32	32,95	34,33	34,91	35,58	36,37	37,34	38,59	38,59	
kodim21	768x512	22,83	24,22	25,10	25,88	27,23	28,31	29,38	30,35	31,06	32,22	33,36	34,33	34,91	35,58
kodim22	768x512	25,16	26,37	27,13	28,04	29,38	30,57	31,32	32,22	32,95	33,82	34,33	34,91	35,58	36,37
kodim23	768x512	28,59	31,32	32,95	34,91	36,37	37,34	38,59	40,35	40,35	40,35	42,35	42,35	43,36	43,36
kodim24	768x512	21,66	22,79	23,49	24,27	25,37	26,46	27,23	28,04	29,05	29,74	30,57	31,32	31,90	32,57
<b>promedio</b>		<b>24,05</b>	<b>25,44</b>	<b>26,52</b>	<b>27,50</b>	<b>28,82</b>	<b>29,94</b>	<b>30,88</b>	<b>31,80</b>	<b>32,46</b>	<b>33,23</b>	<b>33,89</b>	<b>34,56</b>	<b>35,11</b>	<b>35,66</b>

Tabla 23. PSNR para galería Kodak usando LHE

Image	size	bits per pixel (bpp)												
		0,05	0,1	0,15	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1	1,1
kodim01	768x512	19,8	21,1	22,4	24,3	25,3	26,4	26,9	27,7	28,2	28,9	29,4	29,9	30,4
kodim02	768x512	25,4	28,1	30,8	32,3	33,3	34,1	34,8	35,4	36	36,6	37,2	37,5	38,1
kodim03	768x512	22,7	26,65	30,6	33,1	34,6	35,7	36,7	37,7	38,5	39,2	40	40,7	41,3
kodim04	512x768	22,7	26,1	29,5	31,4	32,6	33,6	34,4	35,2	35,8	36,4	36,9	37,3	37,8
kodim05	512x768	19,3	20	20,7	23,2	24,2	25,3	26,3	27	27,6	28,3	29	29,6	30,2
kodim06	768x512	20,9	22,9	24,9	26,2	27,6	28,6	29,3	30,1	30,9	31,4	32,2	32,8	33,4
kodim07	768x512	22	24,8	27,6	30,5	32,5	33,8	34,9	36	36,9	37,7	38,4	39,2	39,9
kodim08	768x512	18,7	19,45	20,2	22	23,6	24,6	25,5	26,4	27,1	27,7	28,4	29	29,6
kodim09	512x768	22,2	25,7	29,2	32	33,8	35,1	36,1	36,9	37,5	38,3	38,8	39,2	39,6
kodim10	512x768	22,1	25,45	28,8	31,2	32,9	34,1	35,2	36	36,8	37,5	38,2	38,7	39,1
kodim11	768x512	21,6	23,7	25,8	27,7	29,1	30	30,7	31,5	32,2	32,9	33,5	34,2	34,7
kodim12	768x512	22,8	26,6	30,4	32,5	33,8	34,9	35,8	36,6	37,3	37,9	38,5	39	39,6
kodim13	768x512	18,7	19,7	20,7	21,8	22,6	23,5	24,1	24,6	25,1	25,6	26,1	26,5	27
kodim14	768x512	20,8	22,6	24,4	26,4	27,6	28,5	29,3	29,9	30,6	31,1	31,7	32,2	32,7
kodim15	768x512	22,9	26,1	29,3	31,4	32,8	33,8	34,7	35,5	36,3	36,9	37,7	38,2	38,6
kodim16	768x512	22,3	25,45	28,6	30,2	31,3	32,3	33,2	34	34,6	35,4	36	36,6	37,2
kodim17	512x768	22,3	25,1	27,9	30	31,5	32,6	33,7	34,4	35,2	35,9	36,6	37,3	37,7
kodim18	512x768	20,7	22,35	24	25,8	27	27,8	28,6	29,4	30	30,7	31,2	31,9	32,4
kodim19	512x768	21,5	23,8	26,1	28,4	29,6	30,7	31,6	32,4	33,1	33,8	34,4	35	35,5
kodim20	768x512	22	25,6	29,2	31,3	32,9	34,1	35,2	36,2	37	37,7	38,5	39,3	39,9
kodim21	768x512	21,3	23,35	25,4	27,1	28,7	29,8	30,7	31,5	32,2	32,9	33,5	34,2	34,6
kodim22	768x512	21,7	24,35	27	28,8	30,1	31,1	31,8	32,5	33,2	33,7	34,3	34,8	35,4
kodim23	768x512	23,1	27,6	32,1	35	36,7	38	39,1	39,8	40,5	41,2	41,6	42,1	42,6
kodim24	768x512	20,3	21,95	23,6	25,3	26,4	27,5	28,3	29,2	29,8	30,6	31,2	31,9	32,5
promedio		21,58	24,10	26,63	28,67	30,02	31,08	31,95	32,75	33,43	34,10	34,72	35,30	35,83

Tabla 24. PSNR para galería Kodak usando JPEG

Image	size	bits per pixel (bpp)													
		0,05	0,1	0,15	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1	1,1	1,2
kodim01	768x512	21,8	23,1	23,9	24,7	25,9	26,8	27,8	28,52	29,24	29,96	30,68	31,4	32	32,6
kodim02	768x512	29,8	31,3	32,15	33	34,2	35,3	36,2	37,04	37,88	38,62	39,26	39,9	40,52	41,14
kodim03	768x512	29,6	31,6	32,85	34,1	36,1	37,6	39,2	40,32	41,44	42,48	43,44	44,4	45,04	45,68
kodim04	512x768	28,6	30,4	31,4	32,4	33,8	35	35,9	36,82	37,74	38,54	39,22	39,9	40,54	41,18
kodim05	512x768	20,3	21,8	22,75	23,7	25	26,3	27,4	28,28	29,16	30,04	30,92	31,8	32,54	33,28
kodim06	768x512	23,9	25,3	26,2	27,1	28,6	29,8	30,9	31,94	32,98	33,88	34,64	35,4	36,14	36,88
kodim07	768x512	25,8	28,4	29,95	31,5	33,6	35,6	37,1	38,46	39,82	41	42	43	43,68	44,36
kodim08	768x512	18,8	20,5	21,6	22,7	24,2	25,5	26,7	27,7	28,7	29,64	30,52	31,4	32,08	32,76
kodim09	512x768	27,7	30	31,5	33	35,3	36,9	38,3	39,26	40,22	41	41,6	42,2	42,62	43,04
kodim10	512x768	27,3	29,6	31	32,4	34,4	36	37,2	38,2	39,2	40,06	40,78	41,5	41,98	42,46
kodim11	768x512	24,9	26,6	27,55	28,5	30,1	31,2	32,4	33,32	34,24	35,12	35,96	36,8	37,48	38,16
kodim12	768x512	29,1	31,2	32,35	33,5	35,1	36,4	37,6	38,52	39,44	40,26	40,98	41,7	42,28	42,86
kodim13	768x512	19,8	20,9	21,6	22,3	23,3	24,2	25	25,72	26,44	27,08	27,64	28,2	28,82	29,44
kodim14	768x512	23,6	25,2	26,1	27	28,4	29,4	30,3	31,18	32,06	32,84	33,52	34,2	34,88	35,56
kodim15	768x512	28,1	30,3	31,4	32,5	34,2	35,5	36,6	37,64	38,68	39,58	40,34	41,1	41,8	42,5
kodim16	768x512	27,6	29	29,95	30,9	32,4	33,8	35	35,96	36,92	37,8	38,6	39,4	40,14	40,88
kodim17	512x768	26,6	28,7	30	31,3	33,1	34,6	35,8	36,84	37,88	38,78	39,54	40,3	40,94	41,58
kodim18	512x768	23,2	24,5	25,35	26,2	27,6	28,9	29,8	30,76	31,72	32,56	33,28	34	34,7	35,4
kodim19	512x768	25,1	27	28,2	29,4	30,9	32,2	33,3	34,14	34,98	35,86	36,78	37,7	38,4	39,1
kodim20	768x512	27,4	29,7	31,05	32,4	34,3	35,8	37,1	38,42	39,74	40,9	41,9	42,9	43,74	44,58
kodim21	768x512	24	25,7	26,75	27,8	29,4	30,7	31,9	32,98	34,06	35,08	36,04	37	37,78	38,56
kodim22	768x512	26,2	27,7	28,6	29,5	30,8	31,9	32,9	33,7	34,5	35,32	36,16	37	37,64	38,28
kodim23	768x512	30,5	33,5	35,2	36,9	38,9	40,4	41,6	42,4	43,2	43,88	44,44	45	45,46	45,92
kodim24	768x512	22,6	23,9	24,85	25,8	27,3	28,6	29,6	30,6	31,6	32,52	33,36	34,2	35	35,8
promedio		25,51	27,33	28,43	29,53	31,12	32,43	33,57	34,53	35,49	36,37	37,15	37,93	38,59	39,25

Tabla 25. PSNR para galería Kodak usando JPEG2000

### 7.2.4 Gráficas de resultados SSIM de la galería Kodak

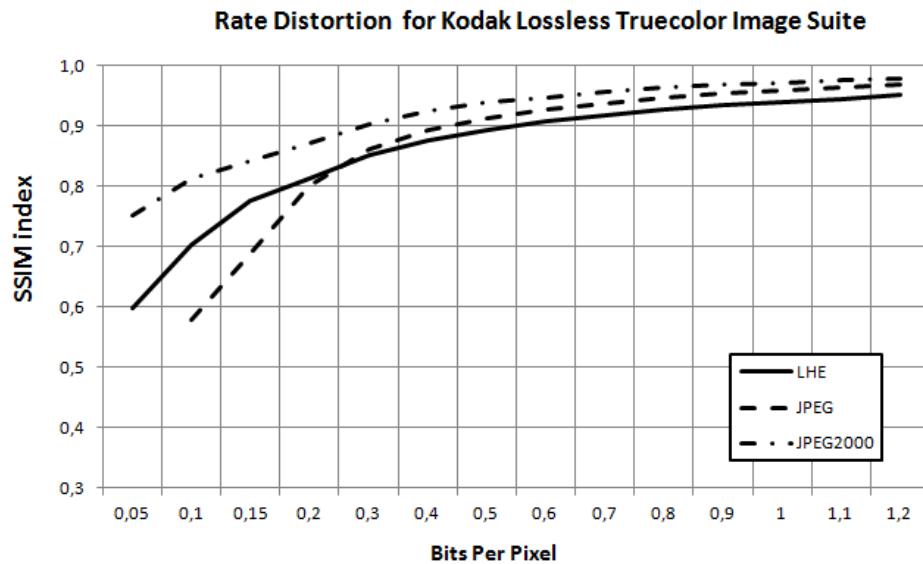


Figura 186. Diagrama de distorsión SSIM para galería “kodak” (promediada)

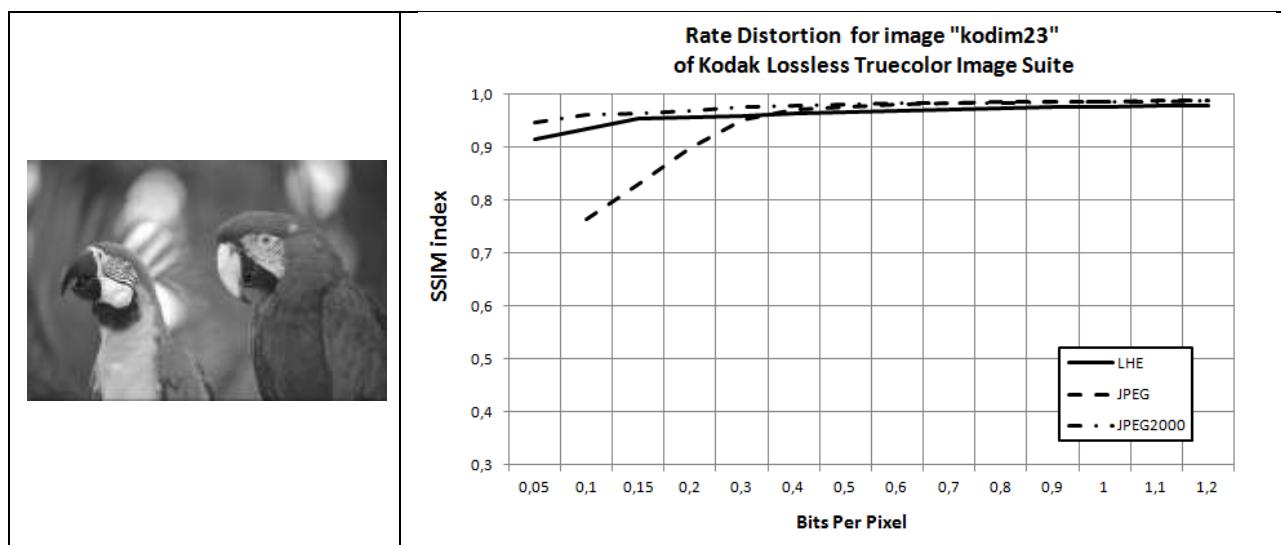


Figura 187. Diagrama de distorsión SSIM para imagen “kodim23”

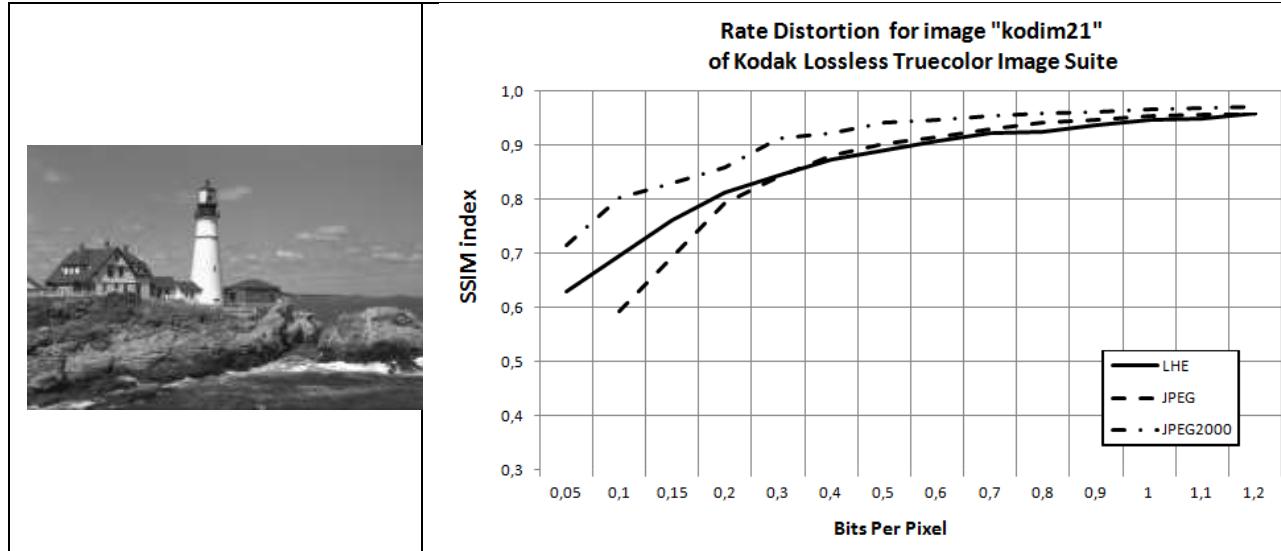


Figura 188. Diagrama de distorsión SSIM para imagen “kodim21”

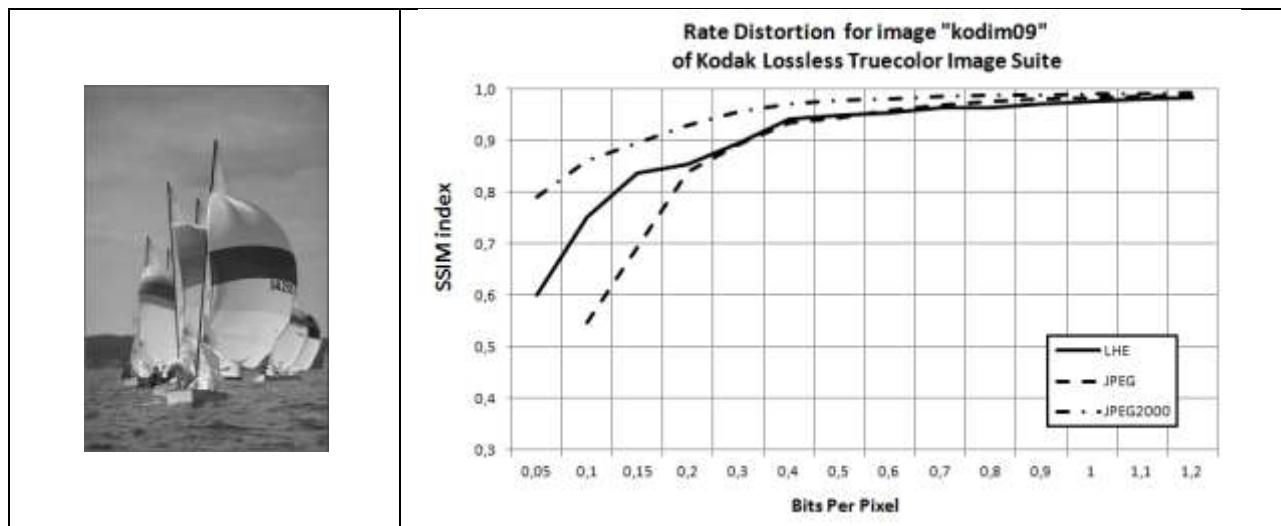


Figura 189. Diagrama de distorsión SSIM para imagen “kodim09”

### 7.2.5 Tablas de resultados SSIM de la galería Kodak

image	size	bits per pixel (bpp)													
		0,0500	0,1000	0,1500	0,2000	0,3000	0,4000	0,5000	0,6000	0,7000	0,8000	0,9000	1,0000	1,1000	1,2000
kodim01	768x512	0,3867	0,5254	0,6074	0,6592	0,7305	0,7705	0,7949	0,8125	0,8320	0,8418	0,8613	0,8809	0,8984	0,9043
kodim02	768x512	0,6973	0,7969	0,8574	0,8887	0,9004	0,9004	0,9160	0,9297	0,9404	0,9434	0,9521	0,9580	0,9590	0,9609
kodim03	768x512	0,6699	0,7568	0,8096	0,8516	0,9082	0,9375	0,9551	0,9600	0,9688	0,9717	0,9746	0,9775	0,9785	
kodim04	512x768	0,7598	0,8633	0,8838	0,9131	0,9424	0,9482	0,9531	0,9551	0,9639	0,9678	0,9727	0,9727	0,9756	0,9756
kodim05	512x768	0,5205	0,6299	0,6924	0,7363	0,7881	0,8320	0,8389	0,8613	0,8750	0,8965	0,9043	0,9160	0,9268	0,9287
kodim06	768x512	0,5615	0,6816	0,7764	0,8184	0,8818	0,8984	0,9326	0,9424	0,9541	0,9561	0,9658	0,9678	0,9707	0,9697
kodim07	768x512	0,5400	0,7773	0,9121	0,9629	0,9385	0,9424	0,9668	0,9785	0,9453	0,9824	0,9775	0,9824	0,9805	0,9795
kodim08	768x512	0,4844	0,4824	0,5352	0,6074	0,6475	0,6963	0,7568	0,7842	0,8057	0,8359	0,8369	0,8379	0,8359	0,8584
kodim09	512x768	0,6006	0,7510	0,8369	0,8535	0,8945	0,9414	0,9482	0,9531	0,9639	0,9639	0,9717	0,9756	0,9805	0,9824
kodim10	512x768	0,6836	0,8027	0,8613	0,8818	0,9209	0,9355	0,9482	0,9561	0,9629	0,9639	0,9697	0,9727	0,9756	0,9766
kodim11	768x512	0,5293	0,6377	0,7305	0,8086	0,8311	0,8418	0,8486	0,8770	0,8828	0,9141	0,9209	0,9199	0,9365	0,9463
kodim12	768x512	0,5117	0,7793	0,8770	0,9395	0,9482	0,9482	0,9531	0,9521	0,9678	0,9736	0,9785	0,9775	0,9795	0,9863
kodim13	768x512	0,4297	0,5303	0,6045	0,6777	0,7246	0,7686	0,7900	0,8213	0,8379	0,8535	0,8672	0,8730	0,8809	0,8848
kodim14	768x512	0,5654	0,6855	0,7368	0,8447	0,8633	0,8848	0,8896	0,9014	0,9150	0,9297	0,9346	0,9395	0,9375	0,9414
kodim15	768x512	0,5996	0,5186	0,7930	0,6504	0,8291	0,8242	0,8311	0,8262	0,8506	0,7920	0,8271	0,8184	0,8223	0,8799
kodim16	768x512	0,6611	0,8496	0,8809	0,8945	0,9307	0,9365	0,9482	0,9463	0,9502	0,9609	0,9658	0,9688	0,9736	
kodim17	512x768	0,6758	0,7764	0,8594	0,8809	0,9141	0,9355	0,9473	0,9541	0,9609	0,9707	0,9717	0,9746	0,9766	0,9785
kodim18	512x768	0,4785	0,5635	0,6221	0,6865	0,7559	0,7930	0,8232	0,8477	0,8691	0,8936	0,9102	0,9229	0,9365	0,9463
kodim19	512x768	0,6816	0,7725	0,8223	0,8555	0,8867	0,9131	0,9189	0,9385	0,9424	0,9512	0,9551	0,9590	0,9629	0,9668
kodim20	768x512	0,7803	0,8320	0,8535	0,8740	0,8896	0,9053	0,9229	0,9316	0,9404	0,9502	0,9541	0,9658	0,9678	0,9717
kodim21	768x512	0,6309	0,6953	0,7607	0,8125	0,8438	0,8730	0,8906	0,9072	0,9229	0,9248	0,9375	0,9473	0,9492	0,9580
kodim22	768x512	0,5029	0,5957	0,6719	0,7275	0,7734	0,8242	0,8555	0,8760	0,8896	0,9141	0,9209	0,9375	0,9424	0,9473
kodim23	768x512	0,9150	0,9346	0,9531	0,9561	0,9580	0,9629	0,9668	0,9688	0,9717	0,9736	0,9756	0,9785	0,9775	
kodim24	768x512	0,5039	0,6230	0,6650	0,7256	0,7813	0,8271	0,8486	0,8730	0,9102	0,9209	0,9326	0,9482	0,9502	0,9580
promedio		0,5988	0,7026	0,7760	0,8128	0,8519	0,8765	0,8931	0,9065	0,9176	0,9265	0,9347	0,9403	0,9446	0,9513

Tabla 26. SSIM para galería Kodak usando LHE

image	size	bits per pixel (bpp)													
		0,05	0,1	0,15	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1	1,1	1,2
kodim01	768x512	0,4248	0,5313	0,6377	0,7451	0,8145	0,8447	0,8691	0,8750	0,8877	0,9063	0,9248	0,9326	0,9355	
kodim02	768x512	0,5811	0,7290	0,8770	0,9004	0,9189	0,9277	0,9268	0,9346	0,9453	0,9541	0,9609	0,9639	0,9707	
kodim03	768x512	0,3906	0,6006	0,8105	0,8838	0,9150	0,9346	0,9502	0,9580	0,9648	0,9697	0,9746	0,9775	0,9824	
kodim04	512x768	0,6533	0,7671	0,8809	0,9160	0,9336	0,9453	0,9551	0,9609	0,9678	0,9736	0,9775	0,9795	0,9805	
kodim05	512x768	0,5576	0,6079	0,6582	0,7598	0,8330	0,8564	0,8799	0,8955	0,9180	0,9336	0,9453	0,9551	0,9570	
kodim06	768x512	0,5820	0,7075	0,8330	0,8848	0,9160	0,9365	0,9463	0,9492	0,9570	0,9639	0,9707	0,9746	0,9795	
kodim07	768x512	0,4785	0,7070	0,9355	0,9326	0,9990	0,9551	0,9492	0,9766	0,9863	0,9873	0,9883	0,9902	0,9912	
kodim08	768x512	0,4502	0,5425	0,6348	0,7422	0,7852	0,8262	0,8516	0,8809	0,9004	0,9063	0,9189	0,9229	0,9307	
kodim09	512x768	0,5469	0,6929	0,8389	0,8936	0,9346	0,9434	0,9580	0,9688	0,9756	0,9805	0,9824	0,9844	0,9873	
kodim10	512x768	0,6211	0,7358	0,8506	0,8975	0,9189	0,9336	0,9473	0,9570	0,9668	0,9697	0,9746	0,9785	0,9805	
kodim11	768x512	0,4805	0,6187	0,7568	0,8486	0,8750	0,9004	0,9102	0,9199	0,9355	0,9385	0,9463	0,9570	0,9609	
kodim12	768x512	0,7979	0,8633	0,9287	0,9424	0,9551	0,9805	0,9824	0,9863	0,9902	0,9922	0,9932	0,9941		
kodim13	768x512	0,5342	0,5972	0,6602	0,7451	0,7793	0,8252	0,8525	0,8574	0,8672	0,8828	0,8936	0,9063	0,9150	
kodim14	768x512	0,6670	0,7500	0,8330	0,8818	0,9121	0,9072	0,9277	0,9355	0,9443	0,9502	0,9590	0,9658	0,9678	
kodim15	768x512	0,8643	0,9185	0,9727	0,9863	0,9883	0,9941	0,9971	0,9980	0,9980	0,9990	0,9990	0,9990	0,9990	
kodim16	768x512	0,6104	0,7368	0,8633	0,9072	0,9297	0,9463	0,9521	0,9600	0,9668	0,9727	0,9756	0,9775	0,9805	
kodim17	512x768	0,6104	0,7256	0,8408	0,8984	0,9258	0,9355	0,9561	0,9648	0,9707	0,9746	0,9775	0,9805	0,9824	
kodim18	512x768	0,4141	0,5454	0,6768	0,7744	0,8281	0,8613	0,8779	0,8984	0,9131	0,9229	0,9307	0,9385	0,9463	
kodim19	512x768	0,6777	0,7456	0,8135	0,8730	0,9063	0,9316	0,9424	0,9473	0,9561	0,9639	0,9678	0,9707	0,9727	
kodim20	768x512	0,7129	0,7817	0,8506	0,8965	0,9199	0,9365	0,9434	0,9492	0,9531	0,9590	0,9629	0,9668	0,9688	
kodim21	768x512	0,5928	0,6924	0,7920	0,8428	0,8799	0,9023	0,9150	0,9297	0,9414	0,9473	0,9541	0,9570	0,9580	
kodim22	768x512	0,3955	0,5234	0,6514	0,7637	0,8242	0,8496	0,8779	0,9033	0,9219	0,9287	0,9277	0,9365	0,9434	
kodim23	768x512	0,7637	0,8306	0,8975	0,9521	0,9697	0,9766	0,9805	0,9824	0,9844	0,9844	0,9854	0,9873	0,9883	
kodim24	768x512	0,4717	0,5850	0,6982	0,7842	0,8154	0,8496	0,8770	0,8955	0,9121	0,9229	0,9336	0,9414	0,9443	
promedio		0,58	0,69	0,80	0,86	0,89	0,91	0,93	0,94	0,95	0,95	0,96	0,96	0,97	

Tabla 27. SSIM para galería Kodak usando JPEG

Image	size	bits per pixel (bpp)														
		0,05	0,1	0,15	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1	1,1	1,2	
kodim01	768x512	0,6055	0,6934	0,7241	0,7549	0,8047	0,8438	0,8760	0,8893	0,9025	0,9166	0,9314	0,9463	0,9523	0,9584	
kodim02	768x512	0,8516	0,8691	0,8789	0,8887	0,9170	0,9346	0,9482	0,9564	0,9646	0,9701	0,9729	0,9756	0,9779	0,9803	
kodim03	768x512	0,6992	0,8262	0,8691	0,9121	0,9424	0,9619	0,9766	0,9824	0,9883	0,9920	0,9936	0,9951	0,9957	0,9963	
kodim04	512x768	0,8936	0,9170	0,9277	0,9385	0,9531	0,9639	0,9678	0,9717	0,9756	0,9787	0,9811	0,9834	0,9844	0,9854	
kodim05	512x768	0,6182	0,6953	0,7310	0,7666	0,8252	0,8523	0,8955	0,9092	0,9229	0,9324	0,9379	0,9434	0,9496	0,9559	
kodim06	768x512	0,7637	0,8438	0,8799	0,9160	0,9443	0,9551	0,9658	0,9717	0,9775	0,9822	0,9857	0,9893	0,9906	0,9920	
kodim07	768x512	0,8799	0,9053	0,9189	0,9326	0,9668	0,9814	0,9863	0,9879	0,9895	0,9906	0,9914	0,9922	0,9928	0,9934	
kodim08	768x512	0,5244	0,6230	0,6699	0,7168	0,7754	0,8232	0,8350	0,8564	0,8779	0,8961	0,9109	0,9258	0,9320	0,9383	
kodim09	512x768	0,7910	0,8623	0,8960	0,9297	0,9570	0,9697	0,9775	0,9814	0,9854	0,9879	0,9891	0,9902	0,9910	0,9918	
kodim10	512x768	0,7803	0,8633	0,8975	0,9316	0,9434	0,9570	0,9639	0,9693	0,9748	0,9789	0,9816	0,9844	0,9855	0,9867	
kodim11	768x512	0,7793	0,8057	0,8159	0,8262	0,8662	0,8867	0,8975	0,9158	0,9342	0,9469	0,9539	0,9609	0,9652	0,9695	
kodim12	768x512	0,8994	0,9600	0,9688	0,9775	0,9824	0,9854	0,9863	0,9891	0,9918	0,9936	0,9943	0,9951	0,9957	0,9963	
kodim13	768x512	0,6006	0,6563	0,6982	0,7402	0,7930	0,8076	0,8213	0,8494	0,8775	0,8955	0,9033	0,9111	0,9205	0,9299	
kodim14	768x512	0,7559	0,8281	0,8491	0,8701	0,9043	0,9287	0,9434	0,9500	0,9566	0,9625	0,9676	0,9727	0,9752	0,9777	
kodim15	768x512	0,9883	0,9932	0,9951	0,9971	0,9971	0,9980	0,9984	0,9988	0,9990	0,9990	0,9990	0,9992	0,9994		
kodim16	768x512	0,8389	0,8721	0,8887	0,9053	0,9336	0,9482	0,9580	0,9666	0,9752	0,9805	0,9824	0,9844	0,9859	0,9875	
kodim17	512x768	0,7998	0,8691	0,8970	0,9248	0,9385	0,9580	0,9658	0,9717	0,9775	0,9818	0,9846	0,9873	0,9887	0,9900	
kodim18	512x768	0,5361	0,6689	0,7178	0,7666	0,8350	0,8701	0,8926	0,9090	0,9254	0,9383	0,9477	0,9570	0,9605	0,9641	
kodim19	512x768	0,8203	0,8623	0,8789	0,8955	0,9238	0,9385	0,9570	0,9609	0,9648	0,9682	0,9709	0,9730	0,9752	0,9768	
kodim20	768x512	0,8105	0,8545	0,8784	0,9023	0,9297	0,9404	0,9551	0,9586	0,9621	0,9650	0,9674	0,9697	0,9730	0,9764	
kodim21	768x512	0,7158	0,8018	0,8301	0,8584	0,9111	0,9219	0,9404	0,9467	0,9529	0,9578	0,9613	0,9648	0,9676	0,9703	
kodim22	768x512	0,5586	0,6396	0,7202	0,8008	0,8428	0,8896	0,9180	0,9316	0,9453	0,9549	0,9604	0,9658	0,9699	0,9740	
kodim23	768x512	0,9453	0,9600	0,9644	0,9688	0,9766	0,9775	0,9814	0,9822	0,9830	0,9838	0,9846	0,9854	0,9865	0,9877	
kodim24	768x512	0,5557	0,6445	0,7046	0,7646	0,8232	0,8945	0,9209	0,9354	0,9498	0,9602	0,9664	0,9727	0,9760	0,9793	
promedio		0,75	0,81	0,84	0,87	0,90	0,92	0,94	0,95	0,96	0,96	0,97	0,97	0,97	0,98	

Tabla 28. SSIM para galería Kodak usando JPEG2000

### 7.2.6 Algunos detalles comparativos de la galería Kodak



Figura 190. Comparación de kodim20 entre JPEG (arriba) y LHE (abajo) a 0.1bpp



Figura 191. Comparación de kodim09 entre JPEG (izquierda) y LHE (derecha) a 0.2bpp

## 7.3 Galería miscelánea de la USC

### 7.3.1 Imágenes de la galería miscelánea



4-1-01.bmp



4-1-02.bmp



4-1-03.bmp



4-1-04.bmp



4-1-05.bmp



4-1-06.bmp



4-1-07.bmp



4-1-08.bmp



4-2-01.bmp



4-2-02.bmp



4-2-03.bmp



4-2-04.bmp



4-2-05.bmp



4-2-06.bmp



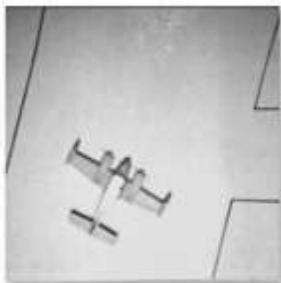
4-2-07.bmp



5-1-09.bmp



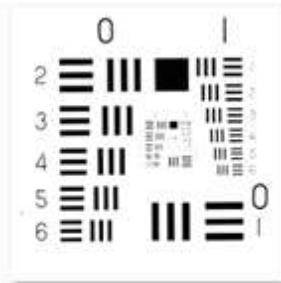
5-1-10.bmp



5-1-11.bmp



5-1-12.bmp



5-1-13.bmp



5-1-14.bmp



5-2-08.bmp



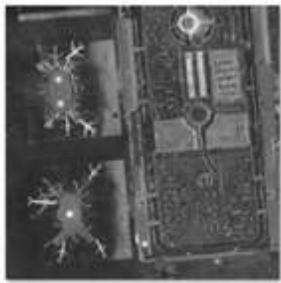
5-2-09.bmp



5-2-10.bmp



5-3-01.bmp



5-3-02.bmp



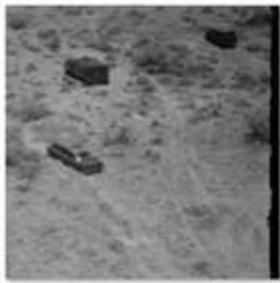
7-1-01.bmp



7-1-02.bmp



7-1-03.bmp



7-1-04.bmp



7-1-05.bmp



7-1-06.bmp



7-1-07.bmp



7-1-08.bmp



7-1-09.bmp



7-1-10.bmp



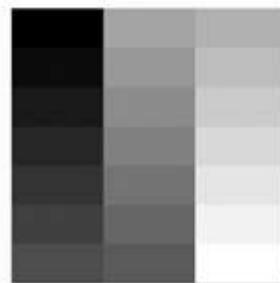
7-2-01.bmp



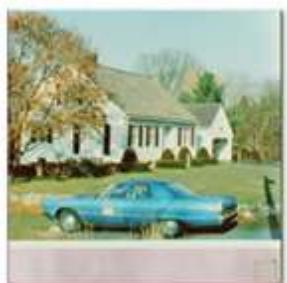
boat-512.bmp



elaine-512.bmp



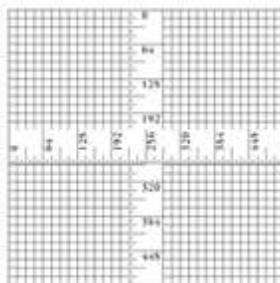
gray21-512.bmp



house.bmp



numbers-512.bmp



ruler-512.bmp



testpat-1k.bmp

### 7.3.2 Gráficas de resultados PSNR de la galería miscelánea

**Rate Distortion Diagram for "Miscelanea" image database**

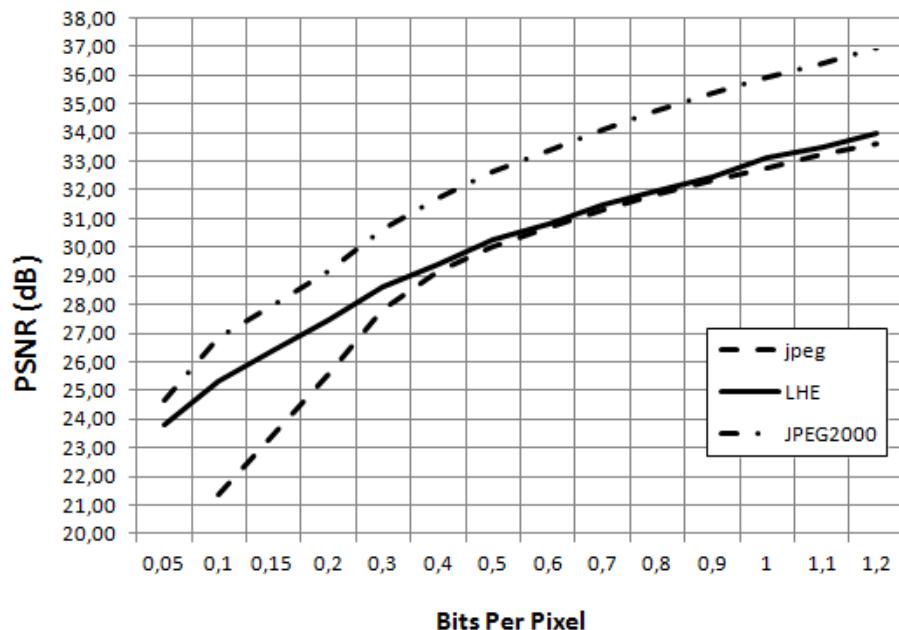


Figura 192. Diagrama de distorsión PSNR para galería “Miscelánea” (promediada)

**Rate Distortion Diagram for "Lena" image**

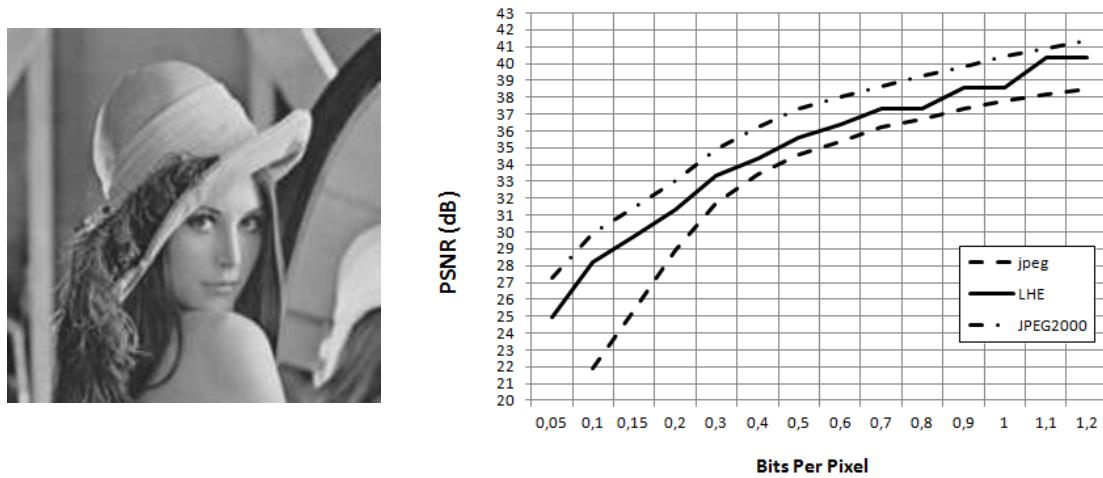


Figura 193. Diagrama de distorsión PSNR para la imagen “Lena”

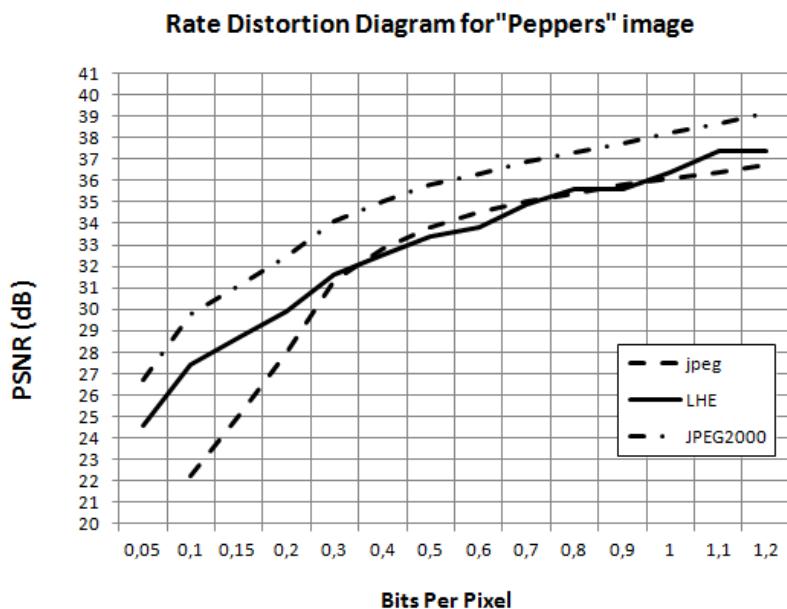


Figura 194. Diagrama de distorsión PSNR para la imagen “Peppers”

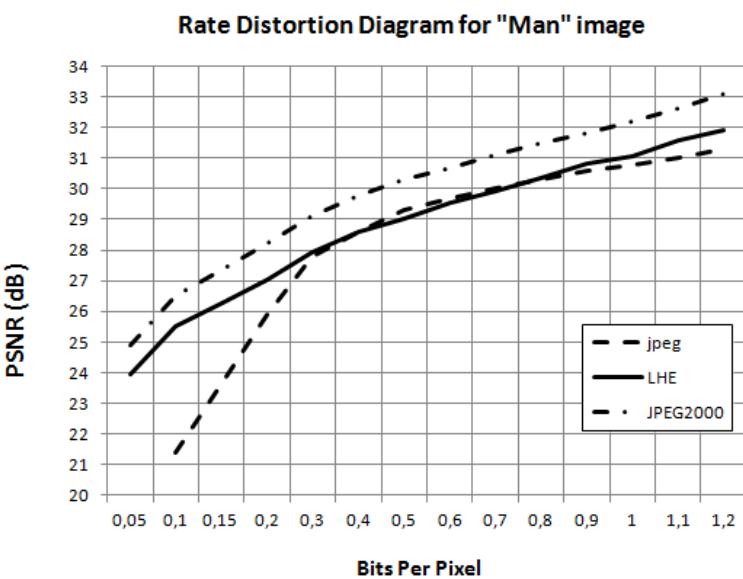


Figura 195. Diagrama de distorsión PSNR para la imagen “Man”

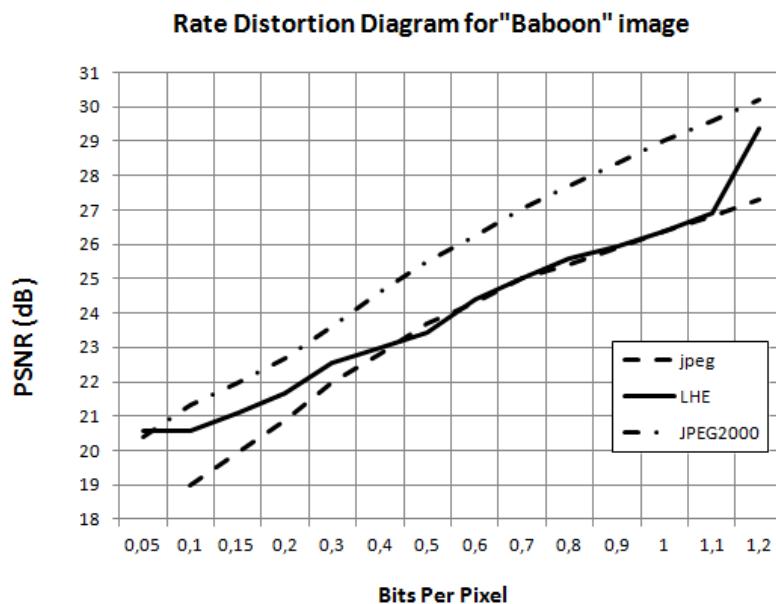
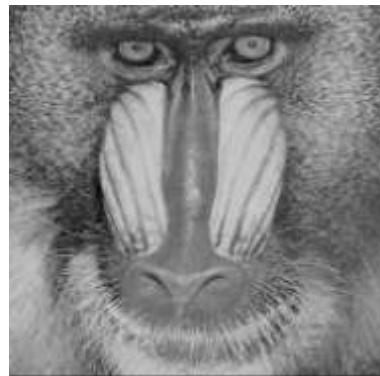


Figura 196. Diagrama de distorsión PSNR para la imagen “Baboon”

### 7.3.3 Tablas de resultados PSNR de la galería miscelánea

image	size	bits per pixel (bpp)												
		0,05	0,1	0,15	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1	
4.1.01 Girl	256x256	25,72531	28,0448	29,74231	31,0551	32,56778	33,81717	34,90861	35,57808	36,36989	37,33899	37,33899	38,38838	38,58838
4.1.02 Couple	256x256	25,57808	27,44894	28,71561	29,93536	31,31839	32,56778	33,81717	34,32869	35,57808	36,36989	37,33899	38,38838	38,58838
4.1.03 Girl	256x256	26,13683	29,38019	30,34929	31,89831	33,35959	34,32869	35,57808	36,36989	37,33899	38,38838	38,58838	43,35959	40,34929
4.1.04 Girl	256x256	24,01461	25,72531	27,44894	29,20988	31,31839	33,35959	34,32869	34,90861	36,36989	36,36989	37,33899	38,38838	38,58838
4.1.05 House	256x256	23,2122	26,03563	27,1271	28,17445	29,20986	29,93536	30,57206	30,80687	31,31839	31,89831	32,56778	32,94566	33,35959
4.1.06 Tree	256x256	20,17896	20,75888	22,18688	22,98533	24,32869	25,57808	26,73201	27,56175	28,30809	29,04595	29,93536	30,80687	31,31839
4.1.07 Jelly beans	256x256	27,79657	31,39831	33,81717	36,36989	38,38838	40,34929	41,35959	43,35959	43,35959	43,35959	43,35959	43,35959	43,35959
4.1.08 Jelly beans	256x256	23,53688	27,11281	30,34929	32,22016	34,32869	36,36989	38,38838	40,34929	40,34929	40,34929	40,34929	40,34929	40,34929
4.2.01 Splash	512x512	27,23175	30,1374	32,22016	34,94564	38,81717	34,32869	34,90861	35,57808	36,36989	37,33899	37,33899	38,38838	38,58838
4.2.02 Girl (Tiffany)	512x512	26,73101	28,30809	29,20988	30,80687	32,56778	33,35959	34,32869	34,90861	35,57808	36,36989	37,33899	38,38838	38,58838
4.2.03 Mandrill (s.k.a. Baboon)	512x512	20,54926	20,54926	21,1085	21,65039	22,56778	22,98533	23,44733	24,38332	25,0345	25,57808	25,95597	26,36989	26,92506
4.2.04 Girl (Lens)	512x512	24,9711	28,17445	29,74231	31,31839	33,35959	34,32869	35,57808	36,36989	37,33899	38,38838	38,38838	40,34929	40,34929
4.2.05 Airplane (F-16)	512x512	22,98533	25,45107	27,44894	29,04595	31,31839	32,56778	33,81717	34,32869	34,90861	36,36989	37,33899	38,38838	38,58838
4.2.06 Sailboat on lake	512x512	21,8674	23,76918	25,0345	26,36989	27,79657	28,73561	29,55748	30,57206	31,31839	31,89831	32,49456	33,35959	34,32869
4.2.07 Peppers	512x512	24,55146	27,44894	28,73561	29,93536	31,59868	32,56778	33,35959	33,81717	34,90861	35,57808	36,36989	37,33899	38,38838
5.1.05 Moon surface	256x256	26,28389	26,92506	27,33899	27,91891	28,30809	28,73561	28,88803	29,20986	29,93809	30,1374	30,57206	30,57206	31,0551
5.1.10 Aerial	256x256	19,85711	19,85711	20,37106	21,23772	22,35889	22,98533	23,67476	24,1168	25,09884	25,29779	26,05865	26,78389	26,9506
5.1.11 Airplane	256x256	24,66098	25,50629	26,03565	26,63081	27,33899	27,79657	28,0448	28,30809	28,88801	29,04595	29,55748	30,1374	30,80687
5.1.12 Clock	256x256	22,82881	23,23122	24,60898	25,23046	26,28389	26,63881	27,33899	27,91891	28,30809	28,58838	29,20986	30,1374	30,34929
5.1.13 Resolution chart	256x256	15,56363	15,56363	17,85731	18,63203	19,78024	20,17890	21,89831	22,36770	23,44733	24,38332	24,78027	29,53530	29,93536
5.1.14 Chemical plant	256x256	21,34562	21,99239	23,44733	23,72171	24,84701	25,72531	26,36989	26,92506	27,56175	27,91891	28,30809	28,44597	29,53748
5.2.08 Couple	512x512	22,2537	23,53688	24,16881	24,76627	26,28389	26,73201	27,33899	27,91891	29,20986	29,74231	30,1374	30,57206	31,0551
5.2.09 Aerial	512x512	20,87986	20,87986	21,62773	22,53171	23,49188	24,38332	25,09884	25,80084	26,28389	26,73201	27,23175	27,67757	28,17445
5.2.10 Stream and bridge	512x512	21,54116	21,54116	22,2537	22,98533	23,58236	24,1168	24,80898	24,9711	25,80084	26,28389	26,54718	27,02491	27,44894
5.3.01 Man	1024x1024	23,9644	25,50629	26,28389	27,02491	27,91891	28,58838	29,04595	29,55748	29,93536	30,34929	30,80687	31,0551	31,59868
5.3.02 Airport	1024x1024	22,56778	23,72171	24,49468	25,23046	25,95597	26,73201	27,1271	27,67757	27,91891	28,44597	28,58838	28,88801	29,80109
7.1.01 Truck	512x512	26,28389	27,1271	27,79657	28,44597	28,88801	29,38019	29,55748	29,93536	30,57206	31,0551	31,11839	31,59868	31,89831
7.1.02 Airplane	512x512	27,91891	29,04595	29,55748	29,93536	30,34929	30,57206	30,80687	31,31839	31,59868	31,89831	32,22016	32,94566	32,94566
7.1.03 Tank	512x512	26,82747	27,44894	28,0448	28,44597	28,88801	29,20986	29,55748	29,74231	30,57206	31,0551	31,11839	31,89831	31,89831
7.1.04 Car and APCs	512x512	26,82747	27,79657	28,44597	29,04595	29,38019	29,93536	30,34929	30,57206	31,0551	31,59868	31,89831	32,22016	32,56778
7.1.05 Truck and APCs	512x512	23,58236	24,68726	25,29779	26,03565	26,73201	27,23175	27,79657	27,91891	28,17445	28,58838	28,38019	29,35748	30,34929
7.1.06 Truck and APCs	512x512	23,81717	24,78627	25,50629	26,19596	26,82747	27,33899	27,91891	28,30809	28,73561	29,55748	30,1374	30,57206	31,0551
7.1.07 Tank	512x512	24,72636	25,80084	26,36989	26,92506	27,56175	28,0448	28,44597	28,73561	29,20986	29,55748	30,34929	30,80687	30,80687
7.1.08 APC	512x512	27,9657	28,44597	28,71561	29,20986	29,55748	29,93536	30,1374	30,34929	30,80687	31,11839	31,59868	32,22016	32,22016
7.1.09 Tank	512x512	25,29779	26,03565	26,28389	26,92506	27,56175	28,0448	28,30809	28,73561	29,04595	29,91516	29,93536	30,1374	30,80687
7.1.10 Car and APCs	512x512	26,82747	27,67757	28,30809	29,04595	29,38019	29,74231	30,1374	30,34929	31,0551	31,11839	31,59868	32,22016	32,56778
7.2.01 Airplane (U-2)	1024x1024	29,04593	29,38019	29,53748	29,55748	29,74231	29,93536	30,1374	30,34929	30,57206	31,0551	31,11839	31,59868	31,89831
boat.512 Fishing Boat	512x512	23,18926	23,86569	25,09884	25,57808	26,63881	27,44894	28,0448	28,73561	29,04595	29,74231	29,93536	30,34929	30,80687
elaine.512 Girl (Elaine)	512x512	25,65107	27,79657	28,30809	28,73561	29,04595	29,38019	29,55748	29,74231	30,34929	30,80687	31,0551	31,11839	31,59868
gray21 21 level step wedge	512x512	29,38019	29,93536	30,34929	30,57206	30,80687	31,0551	31,59868	31,89831	32,22016	32,22016	32,94566	33,35959	33,35959
house House	512x512	22,42538	23,76918	24,9711	26,11863	28,0448	29,55748	30,57206	31,59868	32,56778	33,81717	33,81717	34,50861	35,57808
numbers.512 256 level test pattern	512x512	16,57441	16,57441	16,82747	16,90537	17,36676	17,82076	18,0448	18,51659	18,78077	19,38019	19,91567	20,48157	20,7828
ruler.512 Pixel ruler	512x512	10,36452	10,36452	10,3667	10,72323	10,97411	11,33471	11,43392	11,51552	15,3182	16,70378	24,55146	24,55146	24,55146
testpat.TK General test pattern	1024x1024	16,26689	16,62506	17,65416	18,07043	19,11078	20,26329	21,1271	21,33899	27,67757	28,0448	28,44597	28,88801	28,88801
promedio		23,81	25,32	26,45	27,44	28,60	29,42	30,26	30,82	31,49	31,99	32,46	33,12	33,50

Tabla 29. PSNR para galería Miscelánea usando LHE

image	size	bits per pixel (bpp)												
		0,05	0,1	0,15	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1	1,1
4.1.01 Girl	256x256	22,8	25	27,2	30,6	32,7	34	35	35,7	36,4	37,1	37,6	38,1	38,5
4.1.02 Couple	256x256	22,4	24,85	27,3	30,7	32,6	33,9	34,9	35,8	36,6	37,4	38	38,7	39,1
4.1.03 Girl	256x256	23,2	27,15	31,1	35,1	37,6	39	40,1	40,9	41,4	42	42,4	43	43,4
4.1.04 Girl	256x256	22,2	24,25	26,3	30,8	32,9	34,5	35,7	36,6	37,5	38,3	38,8	39,3	39,9
4.1.05 House	256x256	22,8	24,35	25,9	30,5	32,7	34,1	34,9	35,6	36,4	37,1	37,9	38,5	39
4.1.06 Tree	256x256	19,4	19,4	19,4	23,7	25,5	26,9	27,8	28,7	29,5	30	30,6	31	31,5
4.1.07 Jelly beans	256x256	22,8	26,5	30,2	35,6	38,8	40,7	42,2	43,4	44,3	45,2	46	46,4	46,8
4.1.08 Jelly beans	256x256	22,3	24,2	26,1	31,6	34,6	36,8	38,4	39,5	40,4	41,4	42,3	43,2	43,8
4.2.01 Splash	512x512	23	27,85	32,7	35,5	36,9	37,9	38,7	39,4	39,9	40,4	40,8	41,1	41,6
4.2.02 Girl (Tiffany)	512x512	23,7	26,85	30	32,4	33,6	34,5	35,2	35,9	36,4	36,9	37,4	37,9	38,3
4.2.03 Mandrill (a.k.a. Baboon)	512x512	19	19,95	20,9	22	22,8	23,7	24,3	25	25,4	25,9	26,4	26,8	27,3
4.2.04 Girl (Lena)	512x512	21,9	25,4	28,9	31,7	33,4	34,6	35,4	36,2	36,7	37,3	37,8	38,2	38,5
4.2.05 Airplane (F-16)	512x512	21	24,25	27,5	30,5	32,4	33,9	35,2	36	36,8	37,6	38,2	38,8	39,3
4.2.06 Sailboat on lake	512x512	20,9	22,6	24,3	27,5	29	29,9	30,8	31,5	32,1	32,5	32,9	33,3	33,6
4.2.07 Peppers	512x512	22,2	25,1	28	31,3	32,8	33,8	34,5	35	35,4	35,8	36,1	36,4	36,7
5.1.09 Moon surface	256x256	22,2	24,2	26,2	27,6	28,2	28,6	28,9	29,1	29,4	29,6	29,9	30,1	30,3
5.1.10 Aerial	256x256	18,4	18,4	18,4	21	22,4	23,3	24	24,8	25,3	26	26,3	26,8	27,2
5.1.11 Airplane	256x256	22	24,6	27,2	29,7	30,6	31	31,2	31,4	31,6	31,8	31,9	32,1	32,3
5.1.12 Clock	256x256	21,5	22,3	23,1	26,5	28,1	28,9	29,4	29,7	30	30,2	30,4	30,7	30,9
5.1.13 Resolution chart	256x256	15,8	17,65	19,5	22,1	24,2	25,7	27,5	29,2	30,7	31,9	33,2	34,3	35,5
5.1.14 Chemical plant	256x256	19,9	20,7	21,5	24,1	25,4	26,2	27	27,6	28,1	28,5	28,8	29,2	29,5
5.2.08 Couple	512x512	21,1	22,75	24,4	26,6	27,7	28,5	29,1	29,6	30,1	30,4	30,8	31	31,2
5.2.09 Aerial	512x512	19,2	20	20,8	23,3	24,5	25,4	26	26,7	27,3	27,7	28,1	28,5	28,8
5.2.10 Stream and bridge	512x512	19,7	20,75	21,8	23,3	24	24,7	25,3	25,7	26,1	26,6	26,9	27,2	27,5
5.3.01 Man	1024x1024	21,4	23,65	25,9	27,8	28,6	29,3	29,7	30	30,3	30,6	30,8	31	31,3
5.3.02 Airport	1024x1024	20,5	22,6	24,7	25,9	26,8	27,3	27,7	28	28,3	28,6	28,9	29,1	29,3
7.1.01 Truck	512x512	22,8	24,95	27,1	28,2	29	29,4	29,8	30,1	30,4	30,6	30,8	31	31,2
7.1.02 Airplane	512x512	25,4	27,7	30	30,6	30,8	31	31,1	31,3	31,4	31,6	31,8	32	32,2
7.1.03 Tank	512x512	21,9	24,65	27,4	28,4	28,9	29,3	29,6	29,9	30,2	30,4	30,6	30,8	31
7.1.04 Car and APCs	512x512	22,6	25,05	27,5	28,8	29,4	29,9	30,4	30,7	30,9	31,2	31,4	31,6	31,7
7.1.05 Truck and APCs	512x512	21,1	22,9%	24,8	25,8	26,7	27,3	27,8	28,1	28,5	28,8	29,1	29,3	29,5
7.1.06 Truck and APCs	512x512	21,2	23	24,8	25,9	26,8	27,3	27,8	28,2	28,5	28,8	29,1	29,4	29,6
7.1.07 Tank	512x512	21,5	23,6	25,7	26,8	27,4	27,9	28,3	28,7	28,9	29,2	29,5	29,7	29,9
7.1.08 APC	512x512	24,3	26,55	28,8	29,4	29,8	30	30,3	30,5	30,7	30,9	31	31,2	31,4
7.1.09 Tank	512x512	21,4	23,6	25,8	26,9	27,8	28,2	28,6	29,1	29,4	29,6	29,9	30,2	30,5
7.1.10 Car and APCs	512x512	22,8	25,1	27,4	28,6	29,4	29,9	30,2	30,5	30,8	31	31,2	31,4	31,6
7.2.01 Airplane (U-2)	1024x1024	27	28,35	29,7	30	30,2	30,4	30,6	30,8	31	31,2	31,4	31,5	31,8
boat_512 Fishing Boat	512x512	20,8	22,7	24,6	26,7	27,7	28,5	29,1	29,5	29,9	30,2	30,4	30,7	30,9
elaine_512 Girl (Elaine)	512x512	22,4	24,95	27,5	28,7	29,1	29,4	29,6	29,8	29,9	30,1	30,3	30,4	30,5
gray21 21 level step wedge	512x512	24,1	27,8	31,5	31,7	32	32,2	32,5	32,6	32,8	33,1	33,5	33,7	34,1
house House	512x512	20,6	22,9	25,2	27,6	29,3	30,7	31,8	32,8	33,6	34,3	35	35,7	36,4
numbers_512 256 level test pattern	512x512	16,6	17,1	17,6	18,7	19,6	20,6	21,3	22,4	23,1	23,9	24,6	25,3	25,9
ruler_512 Pixel ruler	512x512	12,5	14,35	16,2	16,2	18,9	19,8	20,9	21,7	22,7	24	25,2	27,1	27,6
testpat_1k General test pattern	1024x1024	19,9	22,05	24,2	28,5	31,2	32,6	32,7	33,9	36,4	37,4	38,2	40,9	42,2
promedio		23,37	23,47	25,57	27,84	29,15	30,03	30,71	31,31	31,85	32,34	32,78	33,24	33,62

Tabla 30. PSNR para galería Miselánea usando JPEG

Image	Size	bits per pixel (bpp)													
		0,05	0,1	0,15	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1	1,1	1,2
4.1.01 Girl	256x256	25,8	28,7	30,25	31,8	33,8	35,3	36,4	37,4	38,4	39,24	39,92	40,6	41,18	41,76
4.1.02 Couple	256x256	25,8	28,5	29,95	31,4	33,3	34,9	36,2	37,24	38,28	39,2	40	40,8	41,52	42,24
4.1.03 Girl	256x256	29,2	32,9	34,8	36,7	39	40,9	42,2	43,2	44,2	45,02	45,66	46,3	46,72	47,14
4.1.04 Girl	256x256	25,6	29,3	31,15	33	35,4	36,9	38,2	39,04	39,88	40,68	41,44	42,2	42,76	43,32
4.1.05 House	256x256	24,6	28,4	30,2	32	34	35,4	36,4	37,4	38,4	39,28	40,04	40,8	41,48	42,16
4.1.06 Tree	256x256	19,5	22	23,35	24,7	26,4	27,5	28,6	29,48	30,36	31,18	31,94	32,7	33,3	33,9
4.1.07 Jelly beans	256x256	28,3	32,4	34,9	37,4	40,4	43,3	45,4	47,04	48,68	49,9	50,7	51,5	51,5	51,5
4.1.08 Jelly beans	256x256	25	28,7	31	33,3	36	38,5	40,6	42,24	43,88	45,32	46,56	47,8	48,44	49,08
4.2.01 splash	512x512	31,3	34,6	35,8	37	38,3	39,2	39,9	40,54	41,18	41,76	42,28	42,8	43,26	43,72
4.2.02 Girl (Tiffany)	512x512	29,1	31,4	32,5	33,6	35	36,1	37,2	38	38,8	39,5	40,1	40,7	41,22	41,74
4.2.03 Mandrill (a.k.a. Baboon)	512x512	20,4	21,3	22	22,7	23,6	24,6	25,5	26,26	27,02	27,72	28,36	29	29,6	30,2
4.2.04 Girl (Leia)	512x512	27,3	29,9	31,45	33	34,9	36,2	37,3	37,98	38,66	39,28	39,84	40,4	40,88	41,36
4.2.05 Airplane (F-16)	512x512	25,2	28	29,75	31,5	33,7	35,3	36,6	37,72	38,84	39,78	40,54	41,3	41,9	42,5
4.2.06 Sailboat on lake	512x512	23	25,2	26,5	27,8	29,4	30,5	31,5	32,22	32,94	33,58	34,14	34,7	35,26	35,82
4.2.07 Peppers	512x512	26,7	29,8	31,15	32,5	34,1	35	35,8	36,32	36,84	37,32	37,76	38,2	38,66	39,12
5.1.09 Moon surface	256x256	26	27,2	27,65	28,1	28,6	29	29,4	29,8	30,2	30,58	30,94	31,3	31,68	32,06
5.1.10 Aerial	256x256	18	20	21	22	23,2	24,2	25	25,6	26,2	26,76	27,28	27,8	28,28	28,76
5.1.11 Airplane	256x256	26,1	28,9	29,85	30,8	31,4	31,7	32,2	32,64	33,08	33,54	34,02	34,5	35,02	35,54
5.1.12 Clock	256x256	22,6	25,1	26,3	27,5	28,9	29,8	30,3	30,82	31,34	31,84	32,32	32,8	33,36	33,92
5.1.13 Resolution chart	256x256	16,4	18,8	20,65	22,5	25,9	28,4	30,6	32,36	34,12	35,7	37,1	38,5	39,82	41,14
5.1.14 Chemical plant	256x256	20,7	22,5	23,55	24,6	25,8	26,8	27,5	28,1	28,7	29,26	29,78	30,3	30,68	31,06
5.2.08 Couple	512x512	23,8	25,7	26,65	27,6	28,7	29,5	30,2	30,64	31,08	31,5	31,9	32,3	32,74	33,18
5.2.09 Aerial	512x512	20,4	21,9	22,85	23,8	25	26	26,9	27,54	28,18	28,76	29,28	29,8	30,24	30,68
5.2.10 Stream and bridge	512x512	21,2	22,2	22,85	23,5	24,4	25,2	25,9	26,46	27,02	27,56	28,08	28,6	29,1	29,6
5.3.01 Man	1024x1024	24,9	26,5	27,35	28,2	29,1	29,8	30,3	30,7	31,1	31,48	31,84	32,2	32,64	33,08
5.3.02 Airport	1024x1024	23,4	24,8	25,5	26,2	27	27,6	28,2	28,68	29,16	29,62	30,06	30,5	30,86	31,22
7.1.01 Truck	512x512	26,4	27,5	28,05	28,6	29,3	29,8	30,2	30,56	30,92	31,28	31,64	32	32,42	32,84
7.1.02 Airplane	512x512	29,3	30,2	30,6	31	31,5	31,9	32,4	32,84	33,28	33,74	34,22	34,7	35,22	35,74
7.1.03 Tank	512x512	26,9	27,8	28,2	28,6	29,2	29,7	30,1	30,46	30,82	31,16	31,48	31,8	32,2	32,6
7.1.04 Car and APCs	512x512	27,2	28,2	28,7	29,2	29,9	30,3	30,7	31,06	31,42	31,76	32,08	32,4	32,84	33,28
7.1.05 Truck and APCs	512x512	24,1	25,2	25,8	26,4	27,1	27,7	28,2	28,64	29,08	29,5	29,9	30,3	30,68	31,06
7.1.06 Truck and APCs	512x512	24,2	25,3	25,9	26,5	27,2	27,7	28,2	28,64	29,08	29,52	29,96	30,4	30,76	31,12
7.1.07 Tank	512x512	25,1	26,1	26,6	27,1	27,7	28,2	28,7	29,1	29,5	29,9	30,3	30,7	31,06	31,42
7.1.08 APC	512x512	28,1	28,9	29,25	29,6	30,1	30,5	30,8	31,12	31,44	31,8	32,2	32,6	33,06	33,52
7.1.09 Tank	512x512	25,3	26,2	26,8	27,4	28,1	28,6	29,1	29,58	30,06	30,48	30,84	31,2	31,58	31,96
7.1.10 Car and APCs	512x512	27,1	28,1	28,6	29,1	29,8	30,2	30,5	30,86	31,22	31,56	31,88	32,2	32,62	33,04
7.2.01 Airplane (U-2)	1024x1024	29,6	29,9	30,15	30,4	30,8	31,1	31,4	31,76	32,12	32,5	32,9	33,3	33,8	34,3
boat_512 Fishing Boat	512x512	23,8	25,4	26,35	27,3	28,5	29,3	29,9	30,34	30,78	31,2	31,6	32	32,44	32,88
elaine_512 Girl (Elaine)	512x512	27	28,3	28,7	29,1	29,5	29,9	30,3	30,66	31,02	31,38	31,74	32,1	32,54	32,98
gray21_21 level step wedge	512x512	32	32,6	33,05	33,5	34,2	35	35,8	36,68	37,56	38,4	39,2	40	40,84	41,68
house House	512x512	24	26,1	27,4	28,7	30,4	31,8	33,1	34,18	35,26	36,3	37,3	38,3	39,08	39,86
ibers_512 256 level test pattern	512x512	16,4	17,3	18,15	19	20,4	21,8	23,2	24,28	25,36	26,34	27,22	28,1	28,66	29,22
ruler_512 Pixel ruler	512x512	13,3	15,3	18,05	20,8	24,4	26,9	28,8	30,28	31,76	33,64	35,92	38,2	39,88	41,56
estpat_1k General test pattern	1024x1024	25,2	29,4	31,85	34,3	37,4	39,9	42,3	44,86	47,42	49,82	52,06	54,3	54,66	55,02
promedio		24,64	26,84	28,01	29,17	30,61	31,71	32,63	33,36	34,10	34,76	35,35	36,98	36,44	36,95

Tabla 31. PSNR para galería Miselánea usando JPEG2000

### 7.3.4 Gráficas de resultados SSIM de la galería miscelánea

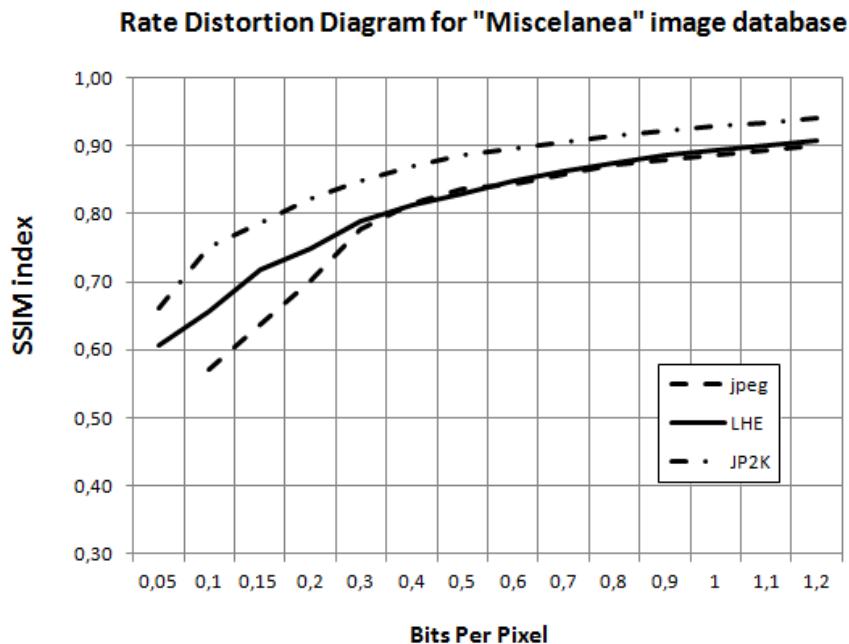


Figura 197. Diagrama de distorsión SSIM para galería “miscelanea” (promediada)

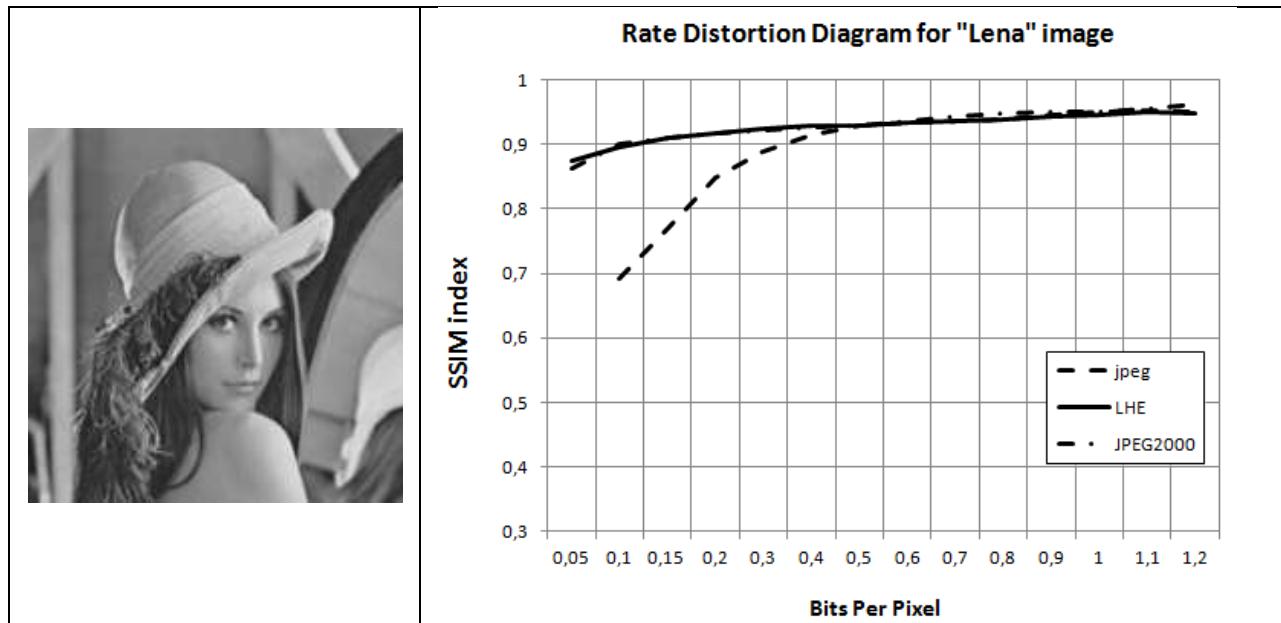


Figura 198. Diagrama de SSIM para la imagen “Lena”

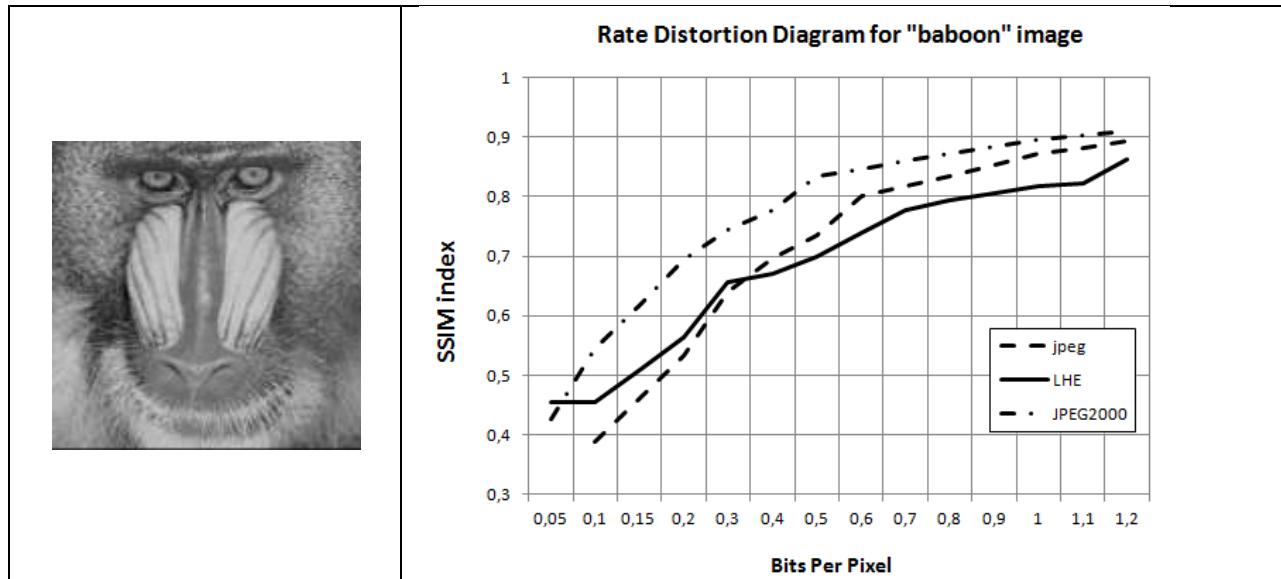


Figura 199. Diagrama de SSIM para la imagen “peppers”

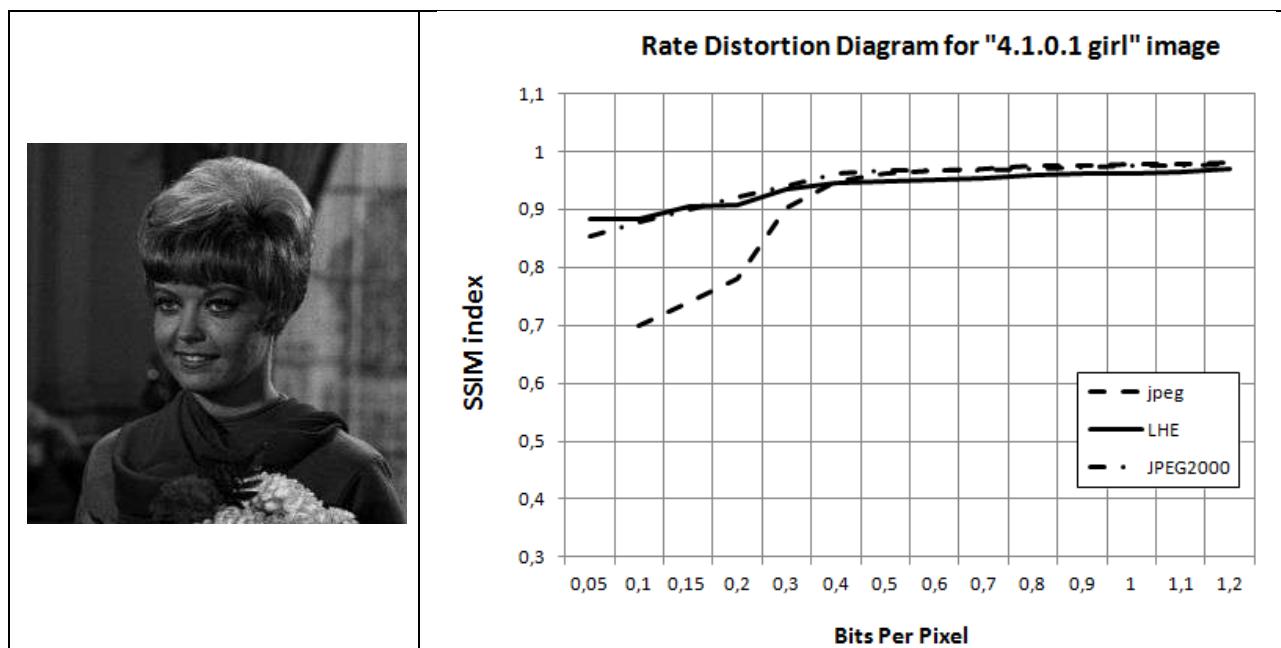


Figura 200. Diagrama de SSIM para la imagen “peppers”

### 7.3.5 Tablas de resultados SSIM de la galería miscelánea

image	size	bits per pixel (bpp)													
		0.05	0.1	0.15	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1	1.2
4.1.01 girl	256x256	0.883789	0.883789	0.90625	0.908203	0.933594	0.944336	0.948242	0.951172	0.953125	0.958008	0.960938	0.962893	0.964844	0.96875
4.1.02 Couple	256x256	0.796875	0.823242	0.844727	0.870117	0.894531	0.90625	0.913086	0.938477	0.935547	0.944336	0.951172	0.959961	0.964844	0.96875
4.1.03 Girl	256x256	0.916016	0.944336	0.9375	0.946289	0.952148	0.95005	0.963867	0.963867	0.961914	0.967773	0.961914	0.972656	0.972656	0.976563
4.1.04 Girl	256x256	0.891602	0.888672	0.888672	0.883789	0.90625	0.907227	0.912109	0.922852	0.931641	0.936523	0.943359	0.948242	0.952148	0.952148
4.1.05 House	256x256	0.880859	0.9375	0.941406	0.961394	0.972656	0.977339	0.978492	0.978516	0.980469	0.981445	0.982422	0.982422	0.982422	0.982422
4.1.06 Tree	256x256	0.504883	0.547852	0.676758	0.733398	0.8125	0.842773	0.876953	0.879461	0.919922	0.933547	0.945313	0.950195	0.957031	0.964844
4.1.07 Jelly beans	256x256	0.577539	0.977539	0.979363	0.975586	0.975586	0.97168	0.976563	0.976563	0.976563	0.976563	0.976563	0.976563	0.976563	0.976563
4.1.08 Jelly beans	256x256	0.981445	0.983398	0.981445	0.979492	0.980469	0.980469	0.979492	0.979492	0.978516	0.978516	0.978516	0.978516	0.978516	0.978516
4.2.01 Splash	512x512	0.394922	0.526167	0.697266	0.730469	0.737305	0.744141	0.760742	0.805641	0.8125	0.849609	0.853516	0.886211	0.876953	0.876953
4.2.02 Girl (tiffany)	512x512	0.447266	0.484175	0.5431945	0.620207	0.677828	0.727539	0.727461	0.832031	0.868164	0.893555	0.933594	0.944336	0.952148	0.960938
4.2.03 Mandrill (a.k.a. Baboon)	512x512	0.436055	0.436055	0.507976	0.563477	0.657227	0.669922	0.699219	0.736281	0.77832	0.794922	0.805664	0.816406	0.823242	0.863281
4.2.04 Girl (Lens)	512x512	0.874023	0.896484	0.90918	0.916992	0.924805	0.928711	0.929688	0.933594	0.936523	0.939453	0.942383	0.946285	0.951172	0.948242
4.2.05 Airplane (F-16)	512x512	0.442383	0.640625	0.790039	0.893555	0.936523	0.951172	0.959961	0.973633	0.980469	0.983398	0.987305	0.988281	0.990234	0.989258
4.2.06 Sailboat on lake	512x512	0.509766	0.608398	0.674805	0.741211	0.795988	0.834961	0.847636	0.86211	0.891602	0.892578	0.905273	0.908203	0.916992	0.925688
4.2.07 Peppers	512x512	0.50293	0.563332	0.676758	0.823242	0.894531	0.914063	0.935547	0.945313	0.996055	0.962891	0.967773	0.970703	0.973633	0.976563
5.1.09 Moon surface	256x256	0.478516	0.488281	0.533203	0.566406	0.600586	0.647461	0.671875	0.686523	0.688477	0.742188	0.77832	0.78418	0.795898	0.797852
5.1.10 Aerial	256x256	0.540039	0.540039	0.601583	0.683594	0.760742	0.805664	0.842773	0.861614	0.886672	0.894511	0.911133	0.916992	0.920898	0.911641
5.1.11 Airplane	256x256	0.598062	0.618789	0.670898	0.686664	0.711914	0.732422	0.741211	0.758789	0.761719	0.761719	0.775391	0.811523	0.818359	0.819844
5.1.12 Clock	256x256	0.557234	0.671289	0.541036	0.477539	0.679968	0.728516	0.720703	0.844727	0.849609	0.879461	0.942383	0.896484	0.894531	0.902344
5.1.13 Resolution chart	256x256	0.566797	0.966797	0.970703	0.972656	0.97168	0.981445	0.984069	0.975566	0.99707	0.992188	0.996094	0.996094	0.996094	0.996094
5.1.14 Chemical plant	256x256	0.488046	0.561523	0.735332	0.740234	0.791016	0.854542	0.849609	0.887695	0.900391	0.916992	0.920898	0.923826	0.932617	0.939453
5.2.08 Couple	512x512	0.518553	0.614258	0.667969	0.681641	0.736281	0.762695	0.791016	0.8125	0.838867	0.834961	0.848033	0.863281	0.873047	0.885742
5.2.09 Aerial	512x512	0.494141	0.495117	0.517578	0.586914	0.65625	0.700195	0.735332	0.756836	0.774414	0.777344	0.783203	0.803711	0.811523	0.829102
5.2.10 Stream and bridge	512x512	0.483398	0.483398	0.541592	0.594727	0.646484	0.689453	0.725586	0.746094	0.776367	0.805664	0.819336	0.821738	0.835938	0.852539
5.3.01 Man	1024x1024	0.650391	0.695113	0.712891	0.716797	0.741211	0.759766	0.770568	0.777344	0.783203	0.805664	0.825195	0.824219	0.844727	0.862305
5.3.02 Airport	1024x1024	0.576172	0.599609	0.618164	0.629883	0.670898	0.676758	0.686653	0.709961	0.729492	0.735352	0.741211	0.755859	0.791016	0.793945
7.1.01 Truck	512x512	0.556641	0.615234	0.651367	0.682617	0.72168	0.741211	0.753589	0.767578	0.797852	0.801758	0.81543	0.834961	0.839844	0.858398
7.1.02 Airplane	512x512	0.623094	0.617188	0.628906	0.642427	0.664063	0.677334	0.685347	0.699313	0.708008	0.731445	0.750977	0.768355	0.77832	0.797578
7.1.03 Tank	512x512	0.520958	0.564453	0.015043	0.643068	0.665039	0.712891	0.734375	0.738281	0.781709	0.807617	0.810547	0.81543	0.853516	0.853516
7.1.04 Car and APCs	512x512	0.559587	0.636203	0.673828	0.71875	0.763672	0.773348	0.792969	0.810547	0.833008	0.84375	0.84375	0.853516	0.863281	0.875977
7.1.05 Truck and APCs	512x512	0.481445	0.560547	0.619141	0.679688	0.717773	0.762695	0.792969	0.801758	0.817383	0.830078	0.856445	0.852539	0.87793	0.886672
7.1.06 Truck and APCs	512x512	0.510742	0.569336	0.629883	0.674805	0.724609	0.750977	0.794922	0.80957	0.84375	0.851563	0.862305	0.887695	0.891602	0.891602
7.1.07 Tank	512x512	0.432617	0.525393	0.583984	0.617188	0.680664	0.706055	0.74707	0.768355	0.774414	0.78125	0.826172	0.835938	0.865234	0.864258
7.1.08 APC	512x512	0.566406	0.581055	0.611281	0.626951	0.648438	0.676758	0.696289	0.716797	0.734375	0.766602	0.775191	0.813477	0.817383	0.817383
7.1.09 Tank	512x512	0.512695	0.589644	0.611328	0.639648	0.662383	0.720703	0.726563	0.769531	0.777344	0.813477	0.817383	0.846668	0.853516	0.862539
7.1.10 Car and APCs	512x512	0.548828	0.628906	0.694336	0.724609	0.763672	0.788133	0.802734	0.814453	0.832031	0.853516	0.860352	0.868164	0.873883	0.883789
7.2.01 Airplane (U-2)	1024x1024	0.612300	0.612305	0.629893	0.630859	0.65918	0.672852	0.699219	0.716797	0.724609	0.762695	0.748047	0.786133	0.859375	0.861328
boat.512 Fishing Boat	512x512	0.563477	0.587893	0.636719	0.648438	0.675781	0.695313	0.708884	0.728516	0.724609	0.750977	0.756789	0.769331	0.782227	0.802734
elaine.512 Girl (Elaine)	512x512	0.633789	0.645508	0.65918	0.666016	0.683594	0.683594	0.702148	0.704102	0.75293	0.737905	0.755859	0.766602	0.775391	0.791016
gray21 21 level step wedge	512x512	0.760742	0.757813	0.760742	0.758789	0.765625	0.775391	0.844727	0.842773	0.856445	0.857422	0.863281	0.882813	0.880859	0.915039
house House	512x512	0.400391	0.581055	0.689435	0.813477	0.838672	0.922852	0.942383	0.947266	0.958006	0.972656	0.975586	0.981445	0.983352	0.987305
numbers.512 256 level test pattern	512x512	0.571289	0.571289	0.619141	0.626953	0.652344	0.718484	0.710938	0.764648	0.777344	0.805664	0.823242	0.849336	0.859598	0.861328
ruler.512 Pixel ruler	512x512	0.207031	0.207031	0.207031	0.350586	0.41133	0.560547	0.661133	0.694336	0.913086	0.936523	0.994141	0.994141	0.994141	0.994141
testpat.TK General test pattern	1024x1024	0.957081	0.970703	0.980469	0.983398	0.984375	0.985352	0.985352	0.975586	0.950195	0.948242	0.954102	0.966328	0.966328	0.966328
promedio		0.61	0.68	0.72	0.75	0.79	0.81	0.83	0.85	0.86	0.88	0.89	0.90	0.91	0.91

Tabla 32. SSIM para galería Miscelánea usando LHE

image	size	bits per pixel (bpp)												
		0.05	0.1	0.15	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1
4.1.01 Girl	256x256	0.696242	0.73877	0.779297	0.90332	0.947296	0.962891	0.965797	0.970703	0.975866	0.976561	0.977539	0.978516	0.980469
4.1.02 Couple	256x256	0.651367	0.696289	0.741211	0.879883	0.908209	0.950195	0.956964	0.965682	0.972636	0.975566	0.977539	0.978516	0.979492
4.1.03 Girl	256x256	0.931641	0.950684	0.969727	0.970503	0.982422	0.982422	0.984375	0.984375	0.984375	0.986328	0.986328	0.986328	0.987303
4.1.04 Girl	256x256	0.837422	0.791016	0.724609	0.878908	0.896484	0.908203	0.919922	0.935547	0.945313	0.953125	0.955961	0.963867	
4.1.05 House	256x256	0.899414	0.921387	0.943399	0.948242	0.967773	0.974609	0.976363	0.981445	0.982422	0.983398	0.983398	0.984375	
4.1.06 Tree	256x256	0.638672	0.638672	0.638672	0.78418	0.863281	0.893555	0.896484	0.927734	0.931641	0.939453	0.956055	0.955078	0.958984
4.1.07 Jelly beans	256x256	0.953125	0.947754	0.942383	0.96875	0.980469	0.987305	0.990234	0.991211	0.993164	0.992188	0.992188	0.993164	
4.1.08 Jelly beans	256x256	0.928713	0.933594	0.958477	0.95457	0.961594	0.979492	0.986128	0.989258	0.991211	0.991211	0.991211	0.991211	0.992188
4.2.01 Splash	512x512	0.640625	0.731445	0.822266	0.949219	0.97168	0.983398	0.985352	0.993164	0.995117	0.996094	0.995117	0.99707	
4.2.02 Girl (Tiffany)	512x512	0.404297	0.580937	0.702578	0.801758	0.898438	0.931641	0.951173	0.966797	0.96875	0.969727	0.976261	0.982422	0.984375
4.2.03 Mandrill (a.k.a. Baboon)	512x512	0.389648	0.461426	0.533203	0.639648	0.696289	0.735352	0.799805	0.818359	0.833008	0.853116	0.871094	0.881816	0.892378
4.2.04 Girl (Lens)	512x512	0.691406	0.769531	0.847656	0.889648	0.915039	0.927734	0.933594	0.936523	0.939453	0.945313	0.949219	0.952148	0.951172
4.2.05 Airplane (F-16)	512x512	0.613281	0.683105	0.732923	0.793943	0.927734	0.953125	0.978316	0.977539	0.983352	0.990234	0.992188	0.993164	
4.2.06 Sailboat on lake	512x512	0.489258	0.550781	0.612305	0.727539	0.791016	0.822266	0.850506	0.87207	0.885742	0.895508	0.910156	0.916016	
4.2.07 Peppers	512x512	0.547852	0.663086	0.77832	0.864258	0.90332	0.938477	0.950195	0.955078	0.962891	0.965862	0.969727	0.972656	0.974609
5.1.09 Moon surface	256x256	0.389648	0.453125	0.516600	0.583984	0.618164	0.661133	0.682617	0.696289	0.727539	0.740234	0.743164	0.780273	0.795858
5.1.10 Aerial	256x256	0.522463	0.522461	0.522461	0.74707	0.799805	0.818359	0.858398	0.879883	0.883789	0.911133	0.911133	0.921875	0.926758
5.1.11 Airplane	256x256	0.420898	0.526855	0.632813	0.729492	0.764648	0.776367	0.799805	0.807617	0.810547	0.822266	0.826172	0.831055	0.837891
5.1.12 Clock	256x256	0.570313	0.60793	0.645508	0.911333	0.973566	0.976063	0.980469	0.983398	0.984375	0.985352	0.986328	0.986328	
5.1.13 Resolution chart	256x256	0.961514	0.962891	0.930867	0.967977	0.968797	0.972856	0.982422	0.991211	0.993104	0.993117	0.993117	0.996094	
5.1.14 Chemical plant	256x256	0.463867	0.547963	0.630859	0.772461	0.835938	0.867188	0.885742	0.892578	0.907227	0.923828	0.928711	0.936523	0.941406
5.2.08 Couple	512x512	0.421875	0.515137	0.608396	0.730469	0.77832	0.813523	0.830078	0.837891	0.843703	0.856445	0.865234	0.873047	0.880859
5.2.09 Aerial	512x512	0.435678	0.510742	0.566406	0.671875	0.730469	0.762695	0.772461	0.781215	0.796875	0.813477	0.828125	0.833008	0.848633
5.2.10 Stream and bridge	512x512	0.369141	0.451172	0.533203	0.656525	0.706055	0.74707	0.780273	0.797852	0.8125	0.846668	0.852539	0.858398	0.865234
5.3.01 Man	1024x1024	0.518555	0.588867	0.65918	0.724609	0.746994	0.758789	0.769531	0.777344	0.782227	0.788086	0.798828	0.804688	0.813477
5.3.02 Airport	1024x1024	0.40332	0.499023	0.594727	0.639648	0.671828	0.694336	0.706055	0.722656	0.738281	0.75291	0.757813	0.764648	0.773438
7.1.01 Truck	512x512	0.429888	0.536623	0.643555	0.696289	0.72168	0.753906	0.777344	0.794922	0.801758	0.8123	0.821189	0.831055	0.833984
7.1.02 Airplane	512x512	0.603518	0.622509	0.641062	0.670898	0.682617	0.703125	0.705078	0.719727	0.730469	0.750977	0.766602	0.784086	0.803711
7.1.03 Tank	512x512	0.388672	0.492187	0.595703	0.657227	0.695313	0.722656	0.746094	0.757813	0.770058	0.78123	0.791392	0.810547	0.830078
7.1.04 Car and APCs	512x512	0.411133	0.543945	0.676758	0.74707	0.77832	0.795898	0.8125	0.830078	0.838914	0.849609	0.852539	0.860352	0.865234
7.1.05 Truck and APCs	512x512	0.352539	0.473145	0.59375	0.6875	0.743164	0.764818	0.813477	0.821289	0.84082	0.839844	0.846668	0.864258	0.875
7.1.06 Truck and APCs	512x512	0.401367	0.522484	0.641602	0.695313	0.737305	0.761719	0.805664	0.816406	0.826172	0.831055	0.837891	0.851563	0.849609
7.1.07 Tank	512x512	0.329102	0.456543	0.583984	0.676758	0.701172	0.734375	0.745117	0.78203	0.800781	0.815336	0.830078	0.842277	0.854492
7.1.08 APC	512x512	0.49707	0.559082	0.621094	0.670988	0.693359	0.701172	0.719727	0.740234	0.75	0.756789	0.772463	0.782227	0.792969
7.1.09 Tank	512x512	0.386211	0.493652	0.621094	0.667969	0.732422	0.754883	0.78418	0.798628	0.81543	0.820311	0.832031	0.836914	0.850586
7.1.10 Car and APCs	512x512	0.392578	0.534806	0.663594	0.772461	0.797852	0.814453	0.827148	0.841797	0.854492	0.853516	0.863281	0.867188	0.873047
7.2.01 Airplane (U-2)	1024x1024	0.584961	0.612793	0.640625	0.694053	0.68457	0.700195	0.714844	0.726563	0.734375	0.743164	0.759766	0.768555	0.777344
boat_512 Fishing Boat	512x512	0.46582	0.529297	0.592773	0.660158	0.703125	0.725586	0.742188	0.760742	0.773591	0.78125	0.791018	0.798828	0.804688
elaine_512 Girl (Elaine)	512x512	0.505859	0.554199	0.602539	0.665039	0.686664	0.696288	0.695313	0.711934	0.733398	0.738281	0.744142	0.756836	0.762695
gray21 21 level step wedge	512x512	0.71582	0.744141	0.772461	0.727539	0.787109	0.787109	0.793945	0.811523	0.842773	0.832031	0.847656	0.851563	0.867188
house House	512x512	0.574219	0.654785	0.735352	0.822266	0.890625	0.926758	0.951172	0.960938	0.967773	0.972656	0.978516	0.982445	0.984375
numbers_512 256 level test pattern	512x512	0.632813	0.667969	0.703125	0.743164	0.771484	0.783203	0.788086	0.813477	0.828125	0.824219	0.833008	0.833984	0.836914
ruler_512 Pixel ruler	512x512	0.643463	0.775879	0.904297	0.904297	0.917969	0.929688	0.968203	0.957031	0.981998	0.994141	0.995117	0.99707	0.988261
testpat_1K General test pattern	1024x1024	0.985352	0.985352	0.985352	0.985352	0.985352	0.985352	0.985352	0.970113	0.962109	0.982125	0.94922	0.928125	0.910156
promedio		0.57	0.64	0.70	0.78	0.82	0.84	0.84	0.86	0.87	0.88	0.89	0.89	0.90

Tabla 33. SSIM para galería Miscelánea usando JPEG

Image	Size	bits per pixel (bpp)													
		0,05	0,1	0,15	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1	1,1	1,2
4.1.01 Girl	256x256	0,853516	0,876953	0,898444	0,919922	0,94043	0,961914	0,966797	0,96719	0,96758	0,96914	0,97188	0,974609	0,97578	0,97695
4.1.02 Couple	256x256	0,785625	0,842773	0,87549	0,908203	0,923828	0,951172	0,96582	0,96934	0,97285	0,97656	0,98047	0,984375	0,98555	0,98672
4.1.03 Girl	256x256	0,958984	0,975586	0,97949	0,983398	0,985352	0,985352	0,986328	0,98672	0,98711	0,9873	0,9873	0,987303	0,9875	0,9877
4.1.04 Girl	256x256	0,825195	0,881836	0,896	0,910156	0,917969	0,935547	0,939453	0,94609	0,95273	0,9582	0,9625	0,966797	0,96934	0,97188
4.1.05 House	256x256	0,898438	0,947206	0,95898	0,970703	0,978516	0,980469	0,982422	0,9832	0,98398	0,98438	0,98438	0,984375	0,98457	0,98477
4.1.06 Tree	256x256	0,509766	0,695313	0,76416	0,833008	0,899414	0,924805	0,931641	0,94414	0,95664	0,96504	0,96934	0,973633	0,97676	0,97988
4.1.07 Jelly beans	256x256	0,962891	0,980469	0,98145	0,982422	0,985352	0,988281	0,991211	0,99121	0,9916	0,99238	0,993164	0,99316	0,99316	0,99316
4.1.08 Jelly beans	256x256	0,929688	0,953125	0,96582	0,978516	0,983352	0,988281	0,989258	0,98965	0,99004	0,99043	0,99082	0,991211	0,9916	0,99199
4.2.01 splash	512x512	0,957031	0,978363	0,98389	0,991211	0,993164	0,995117	0,995117	0,9959	0,99668	0,99727	0,99766	0,998047	0,99805	0,99805
4.2.02 Girl (Tiffany)	512x512	0,754883	0,936533	0,94629	0,956055	0,964844	0,967773	0,976563	0,97813	0,97969	0,98164	0,98398	0,986328	0,9873	0,98828
4.2.03 Mandrill (a.k.a. Baboon)	512x512	0,427734	0,344922	0,61963	0,694336	0,744141	0,776367	0,833008	0,84629	0,85957	0,87207	0,88379	0,895508	0,90312	0,91074
4.2.04 Girl (Lena)	512x512	0,862305	0,901167	0,90918	0,916992	0,921875	0,926758	0,929688	0,93633	0,94297	0,94727	0,94922	0,951172	0,95605	0,96094
4.2.05 Airplane (F-16)	512x512	0,894336	0,938664	0,94727	0,963867	0,972656	0,979492	0,983332	0,98848	0,9916	0,99375	0,99492	0,996094	0,99648	0,99688
4.2.06 Sailboat on lake	512x512	0,555664	0,644531	0,7085	0,772461	0,819336	0,844727	0,867188	0,88164	0,89609	0,90684	0,91387	0,920898	0,92852	0,93613
4.2.07 Peppers	512x512	0,763672	0,864258	0,89941	0,93457	0,954102	0,964844	0,969727	0,97324	0,97676	0,97949	0,98145	0,983398	0,98457	0,98574
5.1.09 Moon surface	256x256	0,44043	0,542969	0,59277	0,642578	0,670898	0,709561	0,753906	0,78477	0,81563	0,83633	0,84688	0,857422	0,86816	0,87891
5.1.10 Aerial	256x256	0,352359	0,351758	0,65381	0,755859	0,832031	0,866211	0,913086	0,92441	0,93574	0,94629	0,95605	0,96582	0,96777	0,96973
5.1.11 Airplane	256x256	0,65332	0,725586	0,73535	0,745117	0,761719	0,793945	0,808594	0,821188	0,83516	0,84863	0,8623	0,875977	0,89082	0,90566
5.1.12 Clock	256x256	0,621094	0,911133	0,94385	0,976563	0,979492	0,980469	0,983422	0,98438	0,98633	0,9877	0,98848	0,989258	0,98984	0,99043
5.1.13 Resolution chart	256x256	0,962891	0,96382	0,96289	0,959961	0,958984	0,989258	0,993164	0,99395	0,99473	0,99551	0,99629	0,99707	0,99746	0,99785
5.1.14 Chemical plant	256x256	0,364258	0,625977	0,69482	0,763672	0,832031	0,867188	0,891602	0,90527	0,91895	0,93086	0,94102	0,951172	0,9543	0,95742
5.2.08 Couple	512x512	0,55957	0,641802	0,71289	0,78418	0,817383	0,838867	0,854492	0,86074	0,86699	0,87519	0,88594	0,896484	0,90313	0,90977
5.2.09 Aerial	512x512	0,477539	0,553711	0,62207	0,69043	0,727539	0,756836	0,779297	0,79297	0,80664	0,81914	0,83047	0,841797	0,85527	0,86875
5.2.10 Stream and bridge	512x512	0,445313	0,541016	0,5918	0,642578	0,706055	0,745117	0,804688	0,821148	0,83828	0,8543	0,86953	0,884766	0,89335	0,90234
5.3.01 Man	1024x1024	0,662109	0,702148	0,71826	0,734375	0,75293	0,756838	0,772461	0,7877	0,80293	0,81504	0,82402	0,833008	0,83906	0,84512
5.3.02 Airport	1024x1024	0,577148	0,597666	0,61963	0,641602	0,663086	0,683594	0,685547	0,68789	0,69023	0,69609	0,70547	0,714844	0,73555	0,75629
7.1.01 Truck	512x512	0,568359	0,605469	0,65527	0,705078	0,75	0,763672	0,78438	0,81035	0,83652	0,85488	0,86543	0,875977	0,88398	0,89199
7.1.02 Airplane	512x512	0,607422	0,626953	0,6416	0,63625	0,675781	0,729492	0,740234	0,75664	0,77305	0,79355	0,81816	0,842773	0,86074	0,87871
7.1.03 Tank	512x512	0,512695	0,576172	0,624541	0,672852	0,733398	0,780273	0,825195	0,8373	0,84941	0,86348	0,87949	0,895508	0,90254	0,90957
7.1.04 Car and APCs	512x512	0,612305	0,678711	0,71338	0,748047	0,788086	0,814453	0,829102	0,846229	0,86348	0,87578	0,88332	0,890625	0,898883	0,90703
7.1.05 Truck and APCs	512x512	0,510742	0,625	0,67871	0,732422	0,794648	0,803664	0,821289	0,85137	0,88145	0,90078	0,90938	0,917969	0,92617	0,93438
7.1.06 Truck and APCs	512x512	0,538086	0,628906	0,68408	0,739258	0,771484	0,806641	0,817383	0,83613	0,85488	0,87051	0,88301	0,895508	0,90441	0,9127
7.1.07 Tank	512x512	0,482422	0,580078	0,61475	0,649414	0,725586	0,753906	0,826172	0,84219	0,8582	0,87012	0,87793	0,885742	0,89492	0,9041
7.1.08 APC	512x512	0,578125	0,607422	0,62988	0,652344	0,69313	0,75293	0,761729	0,78125	0,80078	0,8127	0,81699	0,822189	0,83457	0,84785
7.1.09 Tank	512x512	0,531203	0,598613	0,64307	0,6875	0,719277	0,746094	0,779852	0,82246	0,84707	0,86582	0,87871	0,891602	0,9	0,9084
7.1.10 Car and APCs	512x512	0,631836	0,700195	0,73926	0,77832	0,823242	0,845703	0,856445	0,86855	0,88066	0,89063	0,89844	0,90625	0,91348	0,9207
7.2.01 Airplane (U-2)	1024x1024	0,615234	0,62207	0,62891	0,635742	0,683594	0,731445	0,749023	0,7623	0,77559	0,78672	0,7957	0,804688	0,825	0,84531
boat,512 Fishing Boat	512x512	0,561523	0,610352	0,63965	0,668945	0,705078	0,717773	0,75293	0,77715	0,80137	0,82422	0,8457	0,867188	0,87559	0,88398
laine,512 Girl (Elaine)	512x512	0,637695	0,646484	0,653132	0,660156	0,664063	0,685547	0,724609	0,74531	0,76602	0,78066	0,78926	0,797852	0,80527	0,8127
gray21 21 level step wedge	512x512	0,804688	0,836914	0,85156	0,866211	0,892578	0,912109	0,936523	0,94902	0,96152	0,96914	0,97188	0,974609	0,97773	0,98086
house House	512x512	0,610352	0,852539	0,89209	0,931641	0,949219	0,967773	0,975586	0,98105	0,98652	0,99004	0,99116	0,993164	0,99395	0,99473
ibers,512 256 level test pattern	512x512	0,517578	0,587891	0,61621	0,644531	0,677734	0,730469	0,750977	0,76227	0,77441	0,78711	0,80078	0,814453	0,8207	0,82695
ruler,512 Pixel ruler	512x512	0,820313	0,890625	0,92676	0,962891	0,97663	0,987905	0,988258	0,99238	0,99551	0,99746	0,99824	0,999023	0,99922	0,99941
estpat.1k General test pattern	1024x1024	0,958008	0,926758	0,95459	0,962422	0,989258	0,991211	0,993164	0,99473	0,99629	0,99766	0,99883	1	1	1
promedio		0,66	0,75	0,79	0,82	0,85	0,87	0,89	0,90	0,91	0,92	0,93	0,93	0,94	0,94

Tabla 34. SSIM para galería Miscelánea usando JPEG2000

### 7.3.6 Algunos detalles comparativos de la galería miscelánea



Figura 201. Comparación entre JPEG (izquierda) y LHE (derecha) a 0.1bpp de imagen “Lena”

En el siguiente ejemplo se aprecia como los bordes de las imágenes LHE aparecen algo difuminados en fuertes compresiones debido a los algoritmos de downsampling (media) e interpolación utilizados (en este caso bicúbica).



Figura 202. Comparación entre JPEG2000 (izquierda) y LHE (derecha) a 0.1bpp de imagen “Lena”

El ruido de cuadrícula de JPEG nunca aparece en LHE, ni siquiera en compresiones muy intensas.



Figura 203. Comparación entre JPEG (izquierda) y LHE (derecha) a 0.3bpp de imagen “Lena”



Figura 204. Comparación entre JPEG (izquierda), LHE (centro) y JP2K (izquierda) a 0.1bpp de imagen “4.1.01-girl”

## 7.4 Resultados en color

A continuación se presentan algunos resultados de ejemplo en color, obtenidos usando la primera versión del mecanismo de downsampling basado en 3 mallas (sin Elastic Downsampling y con peores resultados). La aplicación de color basado en el modelo YUV ha sido validado de este modo, y dicha validez es extensible al downsampling elástico pues se basa en el mismo principio: más muestras en las zonas con más relevancia perceptual, aunque sea un mecanismo de downsampling más avanzado.

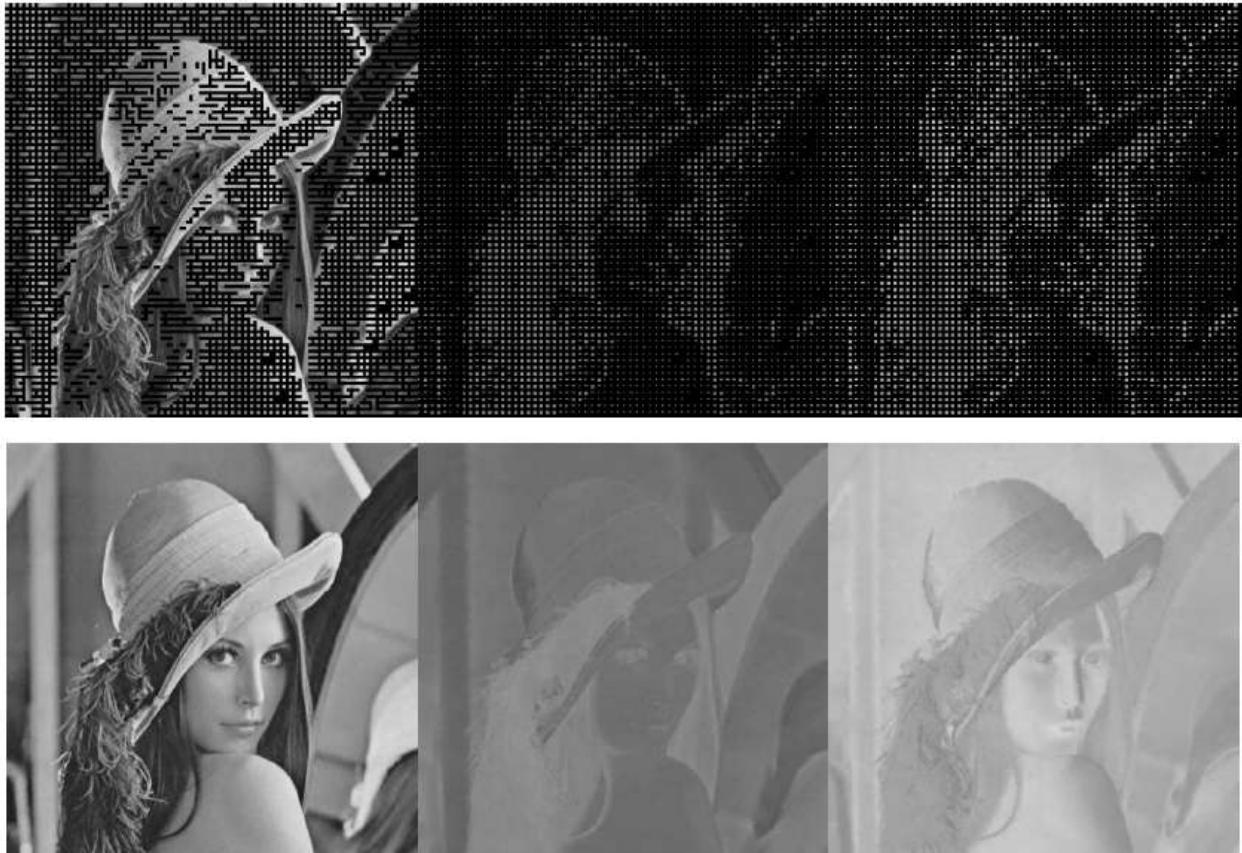


Figura 205. Luminancia, Crominancia U, Crominancia V sub-muestreadas en YUV420 y reconstruidas



Figura 206. Lena en color a 1.6bpp y PSNR 33,22dB

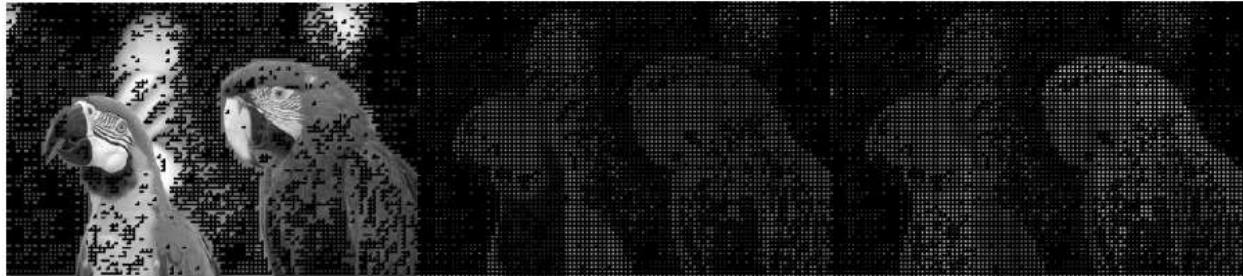


Figura 207. Luminancia, Crominancia U, Crominancia V sub-muestreadas en YUV420 y reconstruidas



Figura 208. Lena en color a 1.37bpp y PSNR 35,82dB

## 7.5 Resultados de video

Para el análisis del codificador de vídeo LHE básico se han escogido algunas secuencias de la galería de videos para test bajo licencia “creative commons” disponibles en la web de la universidad de Mannheim [73]. La comparativa se ha realizado con H264 aunque no es una comparación en igualdad de condiciones ya que el codificador de vídeo LHE básico tan solo codifica información diferencial y para codificar cada frame no tiene en cuenta más que el frame justamente anterior, mientras que H264 utiliza frames P (los cuales usan como referencias uno o más frames anteriores) y frames B (que usan como referencias tanto frames anteriores como posteriores), además H264 utiliza cálculo de vectores de movimiento lo cual representa ahorros enormes de información, aunque también involucra muchos cálculos.

La herramienta escogida para realizar las codificaciones h264 ha sido ffmpeg [74]. FFmpeg es una colección de software libre que puede grabar, convertir (transcodificar) y hacer streaming de audio y vídeo. Incluye libavcodec, una completa biblioteca de códecs.

En las gráficas siguientes aparecen dos referencias de H264. Una de ellas (“fast”) utiliza menos cálculos y elimina el uso de frames “B” (bidireccionales) aunque no por ello deja de emplear vectores de movimiento. Esta sería la configuración adecuada para una aplicación “cloud-gaming”, donde no se pueden usar frames “B”.

Teniendo en cuenta todo esto, se pueden medir los resultados obtenidos en su justa medida y valorar el potencial que puede llegar a tener un codificador de video LHE avanzado que incorporase mejoras como el cálculo de vectores a partir de la malla o una interpolación avanzada que evite los bordes difuminados. En muchos casos LHE con una simple codificación diferencial que apenas consumiría 1ms, se acerca mucho a la configuración “fast” de h264, por lo que revela un alto potencial. En los vídeos en los que el uso de vectores de movimiento representa una mayor ventaja, h264 se distancia más de LHE.

Todos los videos han sido procesados en luminancia, no en crominancia, tanto para LHE como para h264. Junto a cada gráfica se ha añadido un fotograma del video correspondiente para identificarlo más fácilmente.

- Una conclusión general observable (y mejorable) son los bordes difuminados de LHE, sobre todo a bajos bit-rates, debidos a los algoritmos de downsampling e interpolación utilizados. Este aspecto ya lo he mencionado como conclusión al codificar imágenes fijas y lógicamente en los resultados de video es igualmente apreciable.
- Otra conclusión interesante es el paralelismo de la tendencia de la curva de distorsión PSNR entre h264 y LHE. Son paralelas, lo cual es prometedor. Esto no siempre ocurre entre diferentes algoritmos de compresión. Por ejemplo entre JPEG y JPEG2000 o LHE la caída del PSNR de JPEG a partir de 0.3-0.2 bpp tiene como consecuencia unas curvas de PSNR que no son paralelas a las de LHE o a JPEG2000.

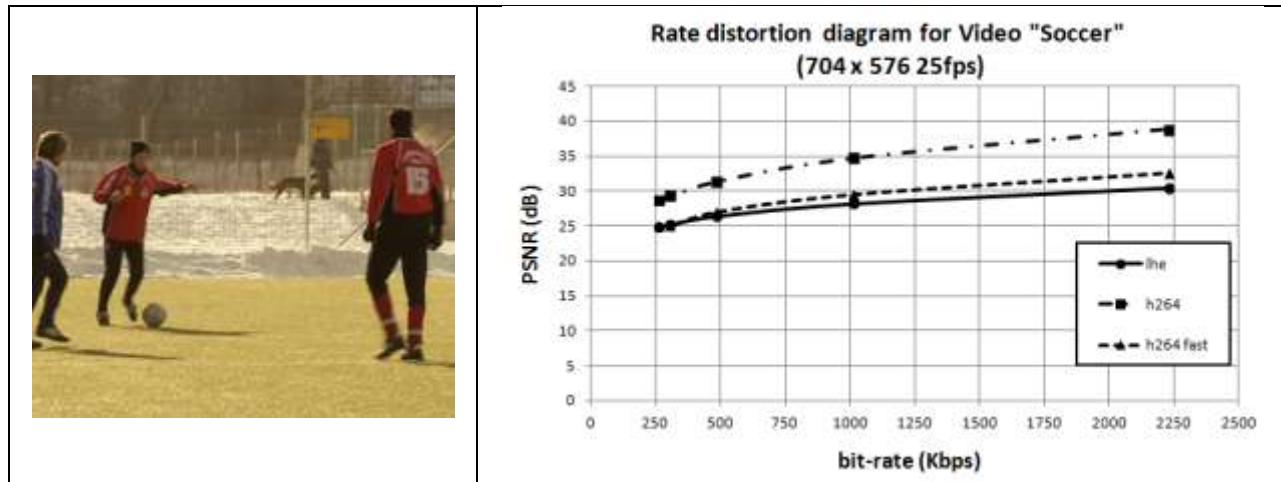


Figura 209. Video soccer



Figura 210. Video soccer LHE a 270 kbps y 930 kbps

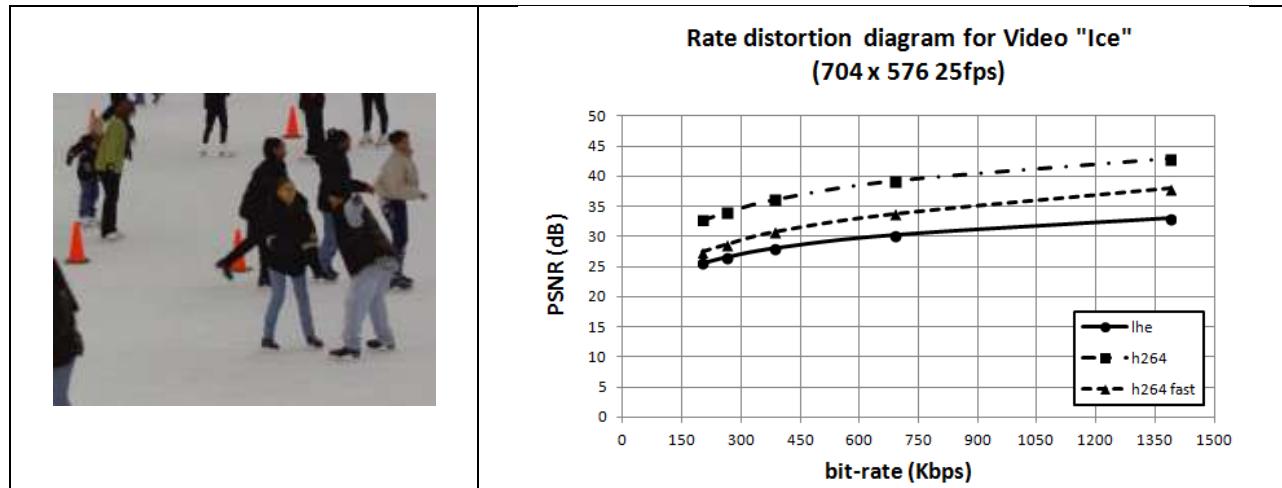


Figura 211. Video "ice"

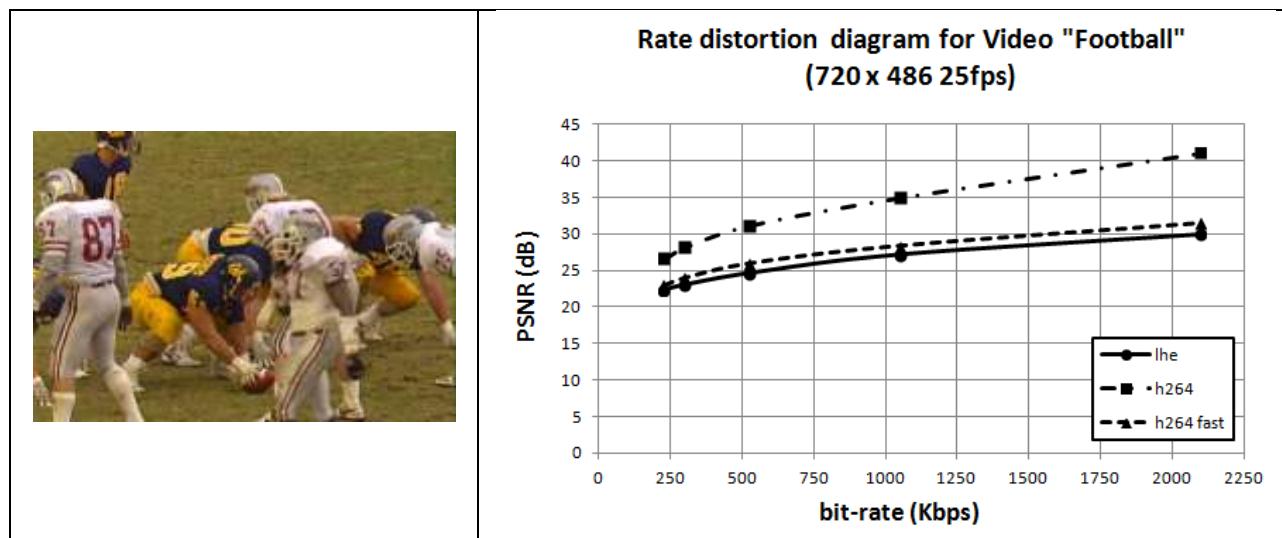


Figura 212. Video football



Figura 213. Video football LHE a 300 kbps y 1000 kbps

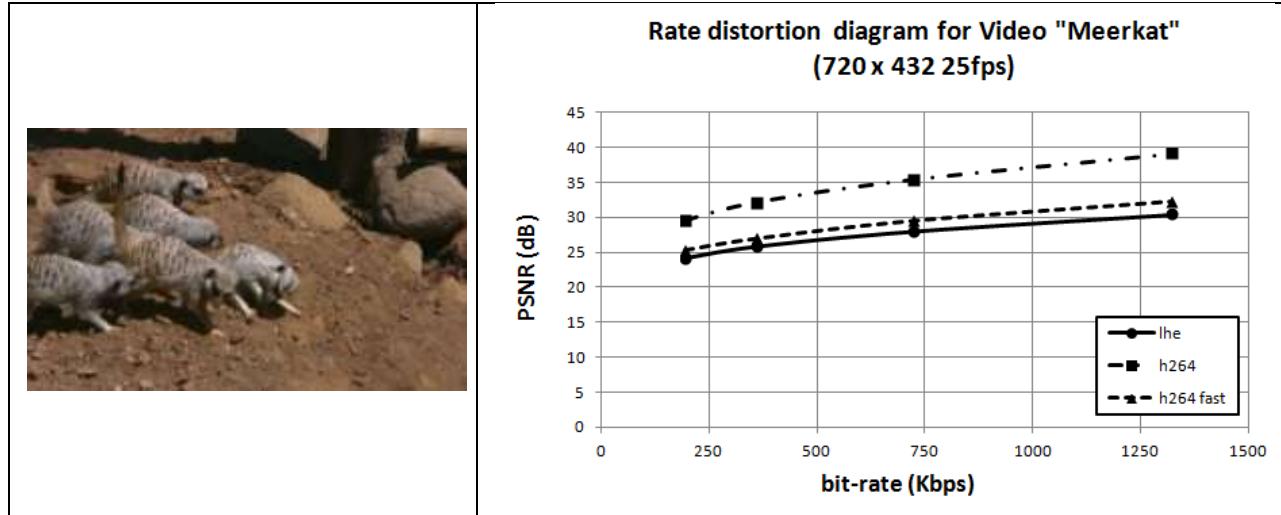


Figura 214. Video Meerkat



Figura 215. Video Meerkat LHE a 320 kbps y 1300 kbps

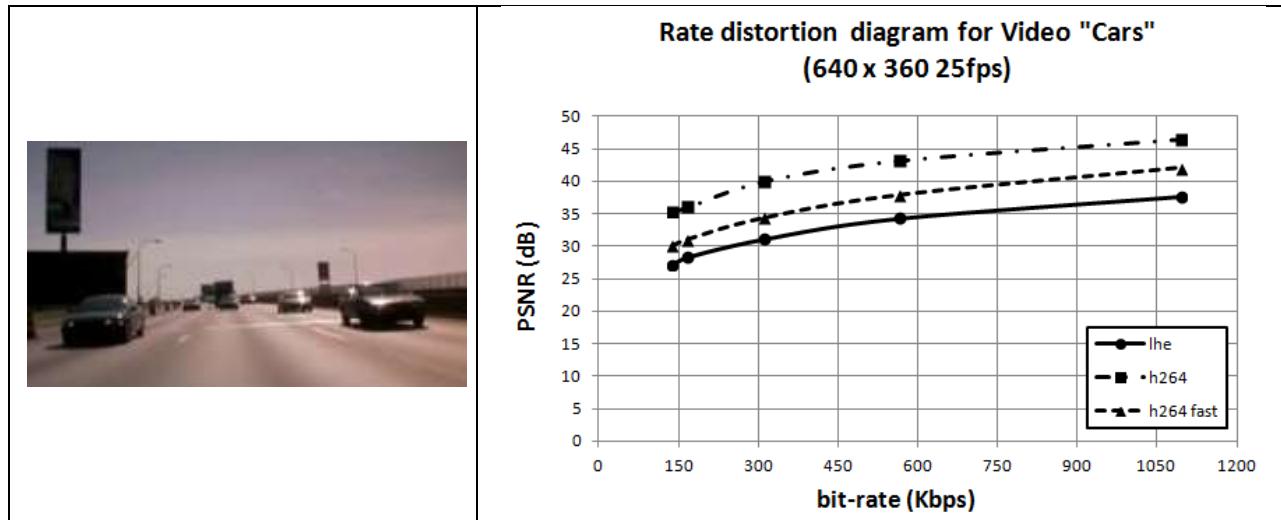


Figura 216. Video Cars

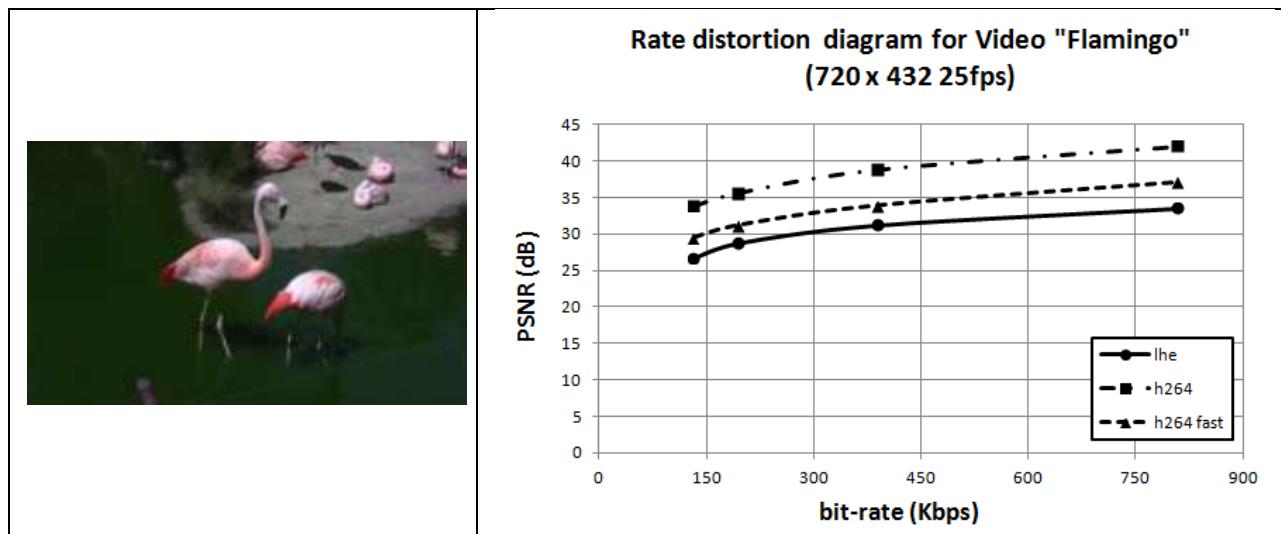


Figura 217. Video flamingo



# Capítulo 8

## Conclusiones y líneas de trabajo futuro

En este capítulo se extraen las principales conclusiones de esta tesis, mediante el análisis de los objetivos que se marcaron en la sección 1.2. Además se analiza el marco de trabajo en el que se ha realizado esta tesis doctoral. Por último, se plantean una serie de líneas futuras de investigación con las que continuar la labor comenzada en este trabajo.

### 8.1 Análisis de los objetivos

Desde el principio el objetivo fundamental de esta tesis ha sido lograr la definición de un algoritmo que no efectúe operaciones de transformación al dominio de la frecuencia, ahorrando el coste computacional que ello supone y dando una oportunidad al dominio del espacio, lograr más calidad, más rapidez y en definitiva, tratar de buscar este resultado haciendo cosas diferentes a las que estamos acostumbrados a ver en los trabajos sobre tratamiento digital de imágenes.

Aunque con esta tesis he logrado los objetivos enumerados al principio, mis verdaderos objetivos están lejos de lo que he alcanzado hasta ahora. Son objetivos fuera del alcance de una sola tesis, pero este trabajo es la demostración de su viabilidad. Es la demostración de que con LHE podemos lograr la compresión de imágenes más rápida de cuantas existen y no por ello de peor calidad. Y ello supone una alternativa para llegar a lograr un “cloud-gaming” de latencia inferior al milisegundo y un ahorro de energía en dispositivos móviles que ejecuten este algoritmo para codificar y visionar contenidos multimedia.

Estos eran los objetivos presentados en el primer capítulo:

- Creación de un algoritmo básico de compresión (sin creación ni uso de patentes) de complejidad lineal y fundamentado en la ley de Weber de la percepción:

En el capítulo 3 se han presentado los fundamentos de LHE, basado en la ley de weber y la respuesta fisiológica del ojo humano. También se ha presentado su reducida complejidad temporal (lineal) y su reducida complejidad espacial, pues con muy poca memoria es posible ejecutar el algoritmo. Como conclusión se ha obtenido un algoritmo de muy bajo coste computacional en comparación con algoritmos existentes, que responde al objetivo planteado. Además como conclusión se puede afirmar que es posible comprimir en el dominio del espacio y con operaciones lineales, al tiempo que se logra una calidad igual o superior al formato de referencia JPEG.

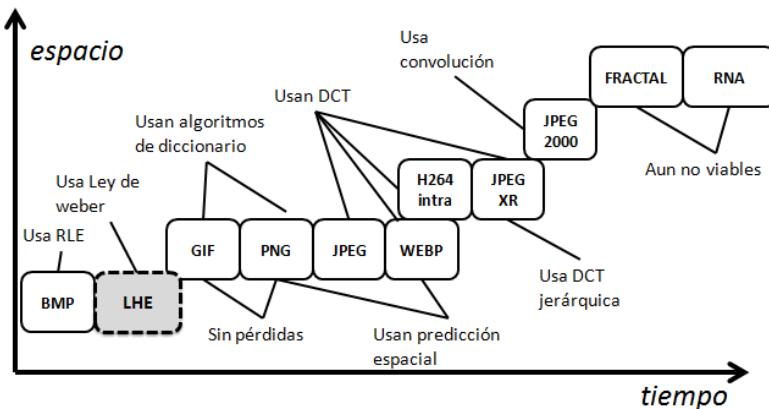


Figura 218. Complejidad comparada de LHE

- Creación de una métrica de relevancia perceptual que permita medir la relevancia que tiene para el observador cualquier área de la imagen para poder sub-muestrear la imagen original en base a la métrica, asignando más muestras en zonas relevantes y menos en zonas poco relevantes (sub-muestreado “elástico”).

En el capítulo 4 se ha presentado la métrica de Relevancia Perceptual (PR) basada en el análisis de los hops. Esta métrica está relacionada con la “interpolabilidad” del área evaluada y por lo tanto sirve para la toma de decisión de un mayor o menor downsampling en cada dirección del espacio. En este capítulo se ha realizado el análisis comparativo del downsampling elástico basado en la métrica PR, frente al downsampling homogéneo convencional, obteniendo resultados superiores en todos los bit-rates.

Una conclusión aplicable a las imágenes de alta resolución sobre las métricas es que pueden ser efectuadas sobre una versión reducida de la imagen por SPS (Single Pixel Selection) sin afectar al resultado del cálculo y acelerando la velocidad de codificación.

Una de las conclusiones adicionales ha sido la necesidad de mejora de los algoritmos de downsampling (actualmente promedio), siendo una posibilidad el cálculo de la mediana, que permitirá mantener bordes abruptos en los bloques sub-muestreados.

- Creación de los algoritmos de interpolación elástica para visualizar las imágenes comprimidas.

En el capítulo 5 he presentado los bloques funcionales del decodificador, entre los que se encuentra la interpolación elástica. Estos algoritmos son dependientes del tipo de interpolación ya sea “vecino cercano”, “bilineal” o “bicúbica”. Otras interpolaciones más avanzadas basadas en algoritmos “Pixel-Art” también son posibles y podrían mejorar los resultados. Esta mejora sería complementaria a la mejora en downsampling con mediana y el uso conjunto de ambas podría suponer la eliminación de bordes difuminados presentes actualmente en LHE, tanto en sub-muestreado como en interpolación.

- Creación de un algoritmo completo que usando el sub-muestreado elástico y la ley de weber permita comprimir linealmente y con mayor calidad que JPEG.

Los capítulos 3, 4 y 5 comprenden toda la descripción del algoritmo LHE y los resultados de calidad se analizan en el capítulo 7, donde se supera a JPEG ampliamente a bajos bit-rates.

Una de las conclusiones de los resultados de este algoritmo es la ausencia de efecto de bloques, algo frecuente en otros algoritmos que usan transformación al dominio de la frecuencia.

Otro de los conceptos introducidos como conclusión ha sido la “calidad equivalente” con un significado físico concreto para comprimir con mayor o menor calidad usando LHE.

Se ha constatado que el algoritmo definido, al igual que JPEG2000, es capaz de alcanzar tasas de compresión fuera del alcance de JPEG, de 0.05bpp.

Se puede concluir que con mejoras futuras sobre LHE (como algunas de las mencionadas en las líneas de trabajo futuras) es previsible que logren incrementos de calidad adicionales respecto el formato de referencia.

- Validación del algoritmo con bases de datos de imágenes de referencia usando diferentes métricas estándares de calidad (PSNR y SSIM).

El capítulo 7 evalúa el algoritmo en comparación con JPEG y JPEG2000 contra dos bases de datos de imágenes de referencia (Kodak y USC) y las dos métricas de calidad (PSNR y SSIM). Los resultados son prometedores y revelan posibles mejoras futuras para LHE, como las mencionadas para el downsampling e interpolación.

Aunque los resultados de calidad han sido satisfactorios, la ventaja fundamental de LHE es computacional y ello permite ahorros en tiempo de codificación y decodificación y por ende en ahorro energético, esencial en dispositivos móviles.

- Validación del modelo de color YUV (YUV 444, YUV 422 y YUV 420) con el nuevo algoritmo.

El capítulo 7 expone algunos resultados en color realizados con la primera versión de LHE, que validan el modelo YUV para el algoritmo.

Una de las conclusiones más importantes de la aplicabilidad del modelo YUV es que se pueden ejecutar las métricas de relevancia perceptual sobre la señal de luminancia y utilizarlas para definir el nivel de sub-muestreado de luma, no requiriendo el cómputo de las métricas para las señales de crominancia.

- Creación del mecanismo básico de compresión de video, es decir, codificación de información diferencial temporal y primera comparativa con el codificador estándar de mayor difusión (H264).

En el capítulo 6 se expone el mecanismo básico para sumar información diferencial temporal codificada con LHE y los resultados obtenidos se exponen en el capítulo 7 en comparación con H264.

Dos son las conclusiones más importantes de estos resultados:

1. A bajos bit-rates LHE produce bordes difuminados que deben resolverse en mejoras futuras, lo cual era previsible considerando los resultados en imagen fija
2. La curva de distorsión de LHE es paralela a la de H264, para cualquier bit-rate, lo cual es prometedor. Esto no siempre ocurre entre diferentes algoritmos de compresión. Por ejemplo entre JPEG y JPEG2000 o LHE la caída del PSNR de JPEG a partir de 0.3-0.2 bpp tiene como consecuencia unas curvas de PSNR que no son paralelas a las de LHE o a JPEG2000.

Los objetivos logrados evidencian el potencial de LHE como algoritmo de compresión de imagen pero también se abre un rango de aplicaciones de cada una de sus partes en el campo del tratamiento digital de imágenes.

En particular la métrica de relevancia perceptual y el downsampling elástico pueden ser útiles no solo

para el algoritmo LHE sino para técnicas de análisis e identificación de imágenes de pre-procesado digital. El resumen de contribuciones de esta tesis se presenta en la siguiente tabla:

Contribución original principal	Contribución derivada o secundaria	Descripción
<b>Algoritmo basado en el modelado del ojo humano</b>	Modelado del brillo medio: Predicción del color de fondo	Se utiliza una técnica original de predicción.
	Modelado de la ley de weber	Mecanismo de distribución de hops en el rango disponible basado en la ley de weber y cálculo de la razón geométrica que relaciona los hops.
	Número óptimo de hops	9 en total. Hop nulo, 4 positivos y 4 negativos.
	Modelado del “Umbral de detección”	Basado en la fracción de weber y afecta al menor hop (H1)
	Modelado de la “acomodación al brillo medio”	Mecanismo de compactación y expansión de hops
<b>Métrica de Relevancia Perceptual</b>	Fórmula de la métrica basada en la no interpolabilidad de la información sensorial	Permite medir la relevancia de la información sensorial para la percepción del sujeto
	Umbral de saturación	Saturación sensorial y aplicación para proteger bordes frente a ruido
	Expansión de histograma de PR	Su aplicación en la métrica PR es una contribución original
	Cuantización geométrica	Demostrable por la relación inversa entre PR y PPP
<b>Downsampling elástico</b>	Concepto de downsampling elástico	Constituye una contribución original, e introduce el concepto PPP.
	Transformación de PR a PPP	Relación entre PR, PPP y el nivel de calidad QL
	malla de bloques sin tamaño fijo	Sin tamaño fijo. Relaciona tanto la métrica PR como el downsampling elástico para su aplicación óptima
	Nivel de calidad QL y calidad equivalente	Otorga significado físico al nivel de compresión deseado QL
	Restricción rectangular	Permite la separabilidad de la operación de downsampling así como la posibilidad de ejecutar un segundo LHE
<b>Arquitectura del codificador y decodificador de imágenes</b>	Primer LHE reducido por SPS	Otorga más velocidad en alta resolución y los resultados de la métrica de PR son iguales.
	Mecanismo de fronteras para la codificación por LHE	Para codificar por LHE los bloques sub-muestreados
	Codificadores binarios de malla y hops	Son codificadores de entropía basados en redundancia espacial y distribución estadísticas
	Interpolación de costuras	Soluciones para las interpolaciones bilineal y bicúbica
<b>Arquitectura del codificador y decodificador de vídeo</b>	Función diferencial temporal por tramos	Absorbe los errores de cuantización de LHE
	Suma de información diferencial temporal de diferente resolución	Solución mediante 3 reglas
	Solución a la compensación de movimiento mediante el análisis de evolución de valores de la métrica de PR de la información diferencial temporal	Permitirá el cálculo instantáneo de vectores de movimiento basados en LHE

## 8.2 Difusión de resultados

Las ideas y contribuciones de esta tesis han sido de gran utilidad en el desarrollo del proyecto de investigación “VideoXperience: Mejora Efectiva de la Experiencia de Usuario en la Nueva Era de Servicios Digitales mediante la Provisión de nuevas Tecnologías de Supercompresión en Streaming”. Este proyecto forma parte del subprograma INNPACTO del Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica 2008-2011, y está financiado por el Ministerio de Ciencia e Innovación, actual Ministerio de Economía y Competitividad. Los objetivos primordiales del proyecto son dos:

- Caracterizar el dimensionamiento de Internet para poder ofrecer servicios de vídeo de alta calidad con una experiencia de usuario medible y similar a los actuales sistemas de TDT e IPTV desplegados por operadores.
- Cubrir el gap existente entre los resultados obtenidos con el primer objetivo y la capacidad de las redes actuales. Para ello se desarrollará un nuevo sistema de codificación de imagen y video capaz de satisfacer dicha experiencia de usuario en Internet sobre cualquier red de acceso fija o móvil. Esto reducirá el coste por byte, aumentará la capacidad de las redes existentes y mejorará la experiencia de usuario.

Esta tesis ha estado centrada en el segundo de los objetivos del proyecto VideoXperience. Y en ese contexto se ha publicado un artículo sobre LHE, en la revista IET-image Processing, anexo en esta memoria.

## 8.3 Líneas de trabajo futuro

A lo largo del trabajo realizado he identificado muchas líneas de trabajo futuro con las que continuar mejorando y evolucionando el algoritmo, tanto en imagen fija como en video. LHE acaba de nacer y requiere muchas mejoras para llegar a explotar todo su potencial

- **Mejora de velocidad en el primer LHE:** mediante un sub-muestreo por SPS de la imagen original se puede acelerar el cálculo de métricas tal como ha quedado demostrado. Esto es especialmente útil en HD y UHD para reducir el tiempo de cálculo de métricas al mínimo. El sub-muestreo por SPS no consume tiempo porque simplemente aplicaríamos LHE en muestras equiespaciadas de la imagen original, por lo que esto no supone ningún procesamiento adicional.
- **Mejora de la predicción:** La predicción que utiliza LHE es siempre la misma y se basa en dos pixels cercanos. Sin embargo tras el primer LHE necesario para calcular las métricas, podríamos cambiar la fórmula de la predicción en función del valor de la métrica de relevancia perceptual en cada esquina del bloque. Dichos valores pueden revelar que para un determinado bloque lo más adecuado es usar el pixel izquierdo como predicción, o el pixel superior. Es decir, para cada bloque en función de las métricas obtenidas, se escogería una predicción entre un juego de predicciones al estilo de PNG. La predicción vendría impuesta por los valores de la métrica PR calculada en las esquinas de cada bloque, de modo que no haría falta almacenarla. Mejoraría la

predicción sin aumentar la información necesaria, por lo que obtendríamos una reducción del bit-rate necesario a igual calidad.

- **Evolución del downsampling elástico** para que genere como resultado no un conjunto de bloques sino una imagen compacta donde las zonas de mayor detalle aparezcan más grandes que las zonas de menor detalle. Esta evolución podría constituir un mecanismo de pre-procesado digital de imágenes útil para cualquier otro algoritmo.
- **Codificación aritmética de los hops:** cada bloque puede codificarse aritméticamente sin necesidad de incluir el número de símbolos que contiene, pues dicho número es calculable a partir de las métricas cuantizadas de la malla y el QL (nivel de calidad). Esto permite explotar la codificación aritmética de un modo muy óptimo, sin el perjuicio de tener que indicar el número de símbolos.
- **Capacidad de transparencias:** igual que en video la función de “dy” posee un tramo vertical para indicar que  $Y_1=Y_2$ , en imágenes fijas se puede definir algo parecido para dotar a LHE de capacidad e de transparencias. Se puede definir una función con un tramo horizontal que defina una transparencia. Las transparencias así definidas se corresponderían con un intervalos de  $\pm 2$ , exactamente igual a lo que hacemos en el video con su franja vertical. De este modo se podrían definir varias franjas de transparencia en la imagen.

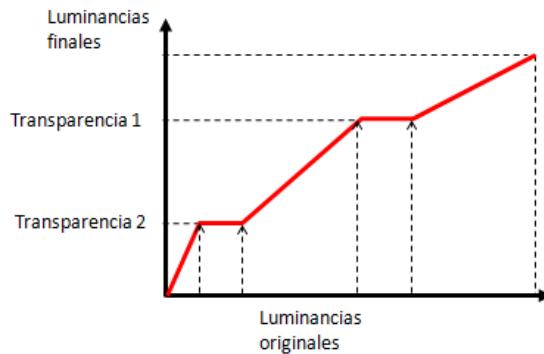


Figura 219. Posibles transparencias en LHE

- **Compresión sin pérdidas:** LHE podría estructurarse en capas, de modo que en la primera capa podría haber 9 hops y se podría hacer una segunda capa correctora con menos tipos de hops, al fin y al cabo, los saltos más grandes no podrán estar presentes. A esta segunda capa se podría sumar una tercera y quizás con ello podríamos ya alcanzar la compresión sin perdidas. Las capas correctoras se podrían beneficiar de usar menos hops y de la codificación RLC por lo que quizás ocuparían muy poco.
- **Mejora del sub-muestreo:** experimentar con el uso de la mediana en lugar de la media. Minimizaría el emborronamiento de bordes en los bloques sub-muestreados.
- **Mejora de la interpolación:** LHE difumina los bordes de las imágenes, sobre todo en compresiones intensas debido al efecto combinado del downsampling y la interpolación. Es esperable que LHE pueda mejorar en este aspecto mediante un downsampling basado en mediana y una interpolación con algoritmo similar a los de “pixel art” (descritos en el capítulo de estado del arte), como XBR [65], aunque estos algoritmos deberían ser adaptados o reformulados para poder actuar de forma “elástica”.



Figura 220. Interpolación por vecino, bicúbica y HQX

- **Compensación de movimiento basada en evolución de la métrica de relevancia perceptual de los nodos de la malla de LHE:** Puesto que actualmente en video LHE codifica la información diferencial, la evolución de la métrica de relevancia perceptual en los nodos de la malla transporta información de movimiento. Si algo se mueve hacia un vértice, su métrica aumenta. Si algo se aleja, su métrica disminuye. La evaluación de la evolución de estas métricas puede proporcionar un mecanismo casi instantáneo de cálculo de vectores, de complejidad  $O(1)$  pues el número de nodos de la malla es constante e independiente de la resolución.
- **Mejoras de la implementación:** introducir el color en el algoritmo actual, implementar la ejecución paralela del algoritmo y hacerlo todo como un plugin para la herramienta ffmpeg de modo que LHE pueda ser utilizado en herramientas de cloud gaming como “gaming anywhere”.
- **Difusión:** impulsar el algoritmo en organismos de estandarización como W3C para definir un nuevo estándar de compresión con pérdidas de menor complejidad a los actuales.
- **Aplicación a Audio:** la ley de weber gobierna los sentidos de la vista y el oído, y precisamente por ello LHE tiene potencial para codificar audio de forma eficiente.



# Apéndice I: Artículo publicado

IET Image Processing

Review Article



## Logarithmical hopping encoding: a low computational complexity algorithm for image compression

*Jose Javier Garcia Aranda<sup>1</sup> , Marina Gonzalez Casquete<sup>1</sup>, Mario Cao Cueto<sup>2</sup>, Joaquin Navarro Salmeron<sup>2</sup>, Francisco Gonzalez Vidal<sup>2</sup>*

<sup>1</sup>Video Architecture Department, Alcatel-Lucent, Madrid, Spain

<sup>2</sup>Departamento de Ingeniería de Sistemas Telemáticos (DIT), Universidad Politécnica de Madrid (UPM), Madrid, Spain

<sup>✉</sup>E-mail: jose\_javier.garcia\_aranda@alcatel-lucent.com

ISSN 1751-9659

Received on 29th January 2014

Accepted on 10th December 2014

doi: 10.1049/iet-ipr.2014.0421

[www.ietdl.org](http://www.ietdl.org)

**Abstract:** LHE (logarithmical hopping encoding) is a computationally efficient image compression algorithm that exploits the Weber-Fechner law to encode the error between colour component predictions and the actual value of such components. More concretely, for each pixel, luminance and chrominance predictions are calculated as a function of the surrounding pixels and then the error between the predictions and the actual values are logarithmically quantised. The main advantage of LHE is that although it is capable of achieving a low-bit rate encoding with high quality results in terms of peak signal-to-noise ratio (PSNR) and image quality metrics with full-reference (FSIM) and non-reference (blind/referenceless image spatial quality evaluator), its time complexity is  $O(n)$  and its memory complexity is  $O(1)$ . Furthermore, an enhanced version of the algorithm is proposed, where the output codes provided by the logarithmical quantiser are used in a pre-processing stage to estimate the perceptual relevance of the image blocks. This allows the algorithm to downsample the blocks with low perceptual relevance, thus improving the compression rate. The performance of LHE is especially remarkable when the bit per pixel rate is low, showing much better quality, in terms of PSNR and FSIM, than JPEG and slightly lower quality than JPEG-2000 but being more computationally efficient.

### 1 Introduction

There are three main concepts that set the limits for image compression techniques: image complexity [1], desired quality and computational cost. This paper presents logarithmical hopping encoding (LHE) algorithm, a computationally efficient algorithm for image compression.

The proposed algorithm relies on Weber-Fechner law, which states that subjective sensation is proportional to the logarithm of the stimulus intensity [2]. LHE applies this law to prediction errors instead of the stimulus itself (in this case the original luminance and chrominance signals). More concretely, LHE estimates a luminance and chrominance prediction for each pixel (using the surrounding pixels) and then encodes the prediction error using a set of logarithmically distributed and dynamically adjusted values. This procedure is performed in the space domain, avoiding the need of any costly transformation to the frequency domain, and therefore reducing the computational complexity.

This approach can be enhanced by including a pre-processing stage to estimate the perceptual relevance of the image blocks. As will be explained later, the perceptual relevance estimation can be obtained, in an efficient manner, using the output codes of the logarithmical quantiser. This pre-processing stage allows the algorithm to perform region of interest (ROI) coding [3].

The remaining of this paper is organised as follows. Section 2 surveys the most relevant work related to the proposed algorithm. Section 3 describes the structure and the workflow of LHE. In Section 4, an enhanced version of the algorithm is presented. Section 5 shows the main results that have been obtained in the evaluation stage of the algorithm. Finally, Section 6 summarises the main contributions of this paper and outlines the future lines of work.

### 2 Related work

LHE can be defined as a spatial domain image compression algorithm. In the literature of image compression, spatial domain-based algorithms have been extensively studied.

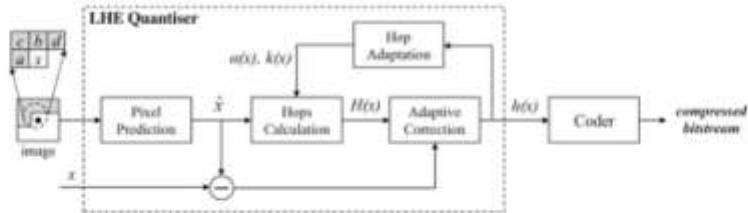
In [4], a spatial domain image compression algorithm is proposed. This algorithm encodes the difference between the minimum pixel value of an  $m \times n$  pixel block and the current pixel. For each block, an 11 bits header is included in order to represent the minimum value of the block (8 bits) and the number of pixels required to encode the pixel difference with respect to the minimum (3 bits).

The authors of [5] present a modified approach to the previous algorithm where the final number of bits is reduced significantly by reducing the overhead bits. In [6], a variation of the previous algorithm that encodes the difference between adjacent pixels is proposed.

In [7], a logarithmic function is used as a pre-processing stage for an image compression algorithm. This algorithm comprises four stages: logarithmic transform, neighbouring difference, repeat reduction and Huffman encoding. In this paper, the logarithmic function is used to reduce the range of the difference between neighbouring pixels.

LHE also has similarities to ADPCM (adaptive differential pulse-code modulation) [8]. ADPCM uses an adaptive predictor and an adaptive quantiser. The quantiser levels for a given pixel are generated by scaling the levels used for the previous pixel by a factor that depends on the reconstruction level used for the previous pixel. ADPCM dynamically adapts the quantiser step size to the input signal, and the set of possible unary codes is linearly distributed for each sample. However, LHE unary codes are logarithmically distributed for each sample. This change in step distribution provides better results than ADPCM. For example, according to [8], Lena image at 1.2 bpp encoded with ADPCM provides a peak signal-to-noise ratio (PSNR) value of 30.24 dB whereas LHE provides 39.1 dB.

LOC-O-I [9] is the algorithm at the core of the ISO/ITU standard for lossless and near-lossless compression of continuous-tone images. JPEG-LS, LOC-O-I uses the prediction of samples based on a finite subset of available past data and the context modelling of the prediction error. The purpose of this context modelling is to exploit high order structures, for example, texture patterns, by



**Fig. 1** LHE basic: block diagram

analysing the level of activities, such as smoothness and edginess of the neighbouring samples. This context modelling provides a probabilistic model for the prediction residual (or error signal), which can be efficiently used in combination with Golomb-Rice codes. LHE uses a similar approach, but focused on lossy image compression and taking into account the logarithmical nature of human perception.

### 3 LHE: basic algorithm

The basic algorithm of LHE is based on the prediction of colour space values (e.g. YUV) of each pixel from the previous ones. The errors of the predicted values are encoded using a set of logarithmically distributed possible values of luminance and chrominance, which are called hops. The main blocks of the basic algorithm of LHE, grouped as LHE quantiser, are depicted in Fig. 1. Detailed information about these blocks is provided in the following subsections.

#### 3.1 Pixel prediction

LHE uses the YUV colour space to represent the pixel information, YUV is defined in terms of one luminance value (Y) and two chrominance components (UV). The human eye has fairly little spatial sensitivity to colour, thus luminance component has far more impact on the image detail than the chrominance. For a given pixel, LHE predicts each colour component (Y, U or V) as the average of the colour component of the top (*b*) and left (*a*) pixels, as in the following equation

$$\hat{x} = \frac{a + b}{2} \quad (1)$$

The prediction of each colour component should be computed individually. Therefore, for a given pixel *x*,  $\hat{x}$  represents the predicted value of the luminance (Y) or chrominance (UV). In the remaining of this section, luminance will be used as an example of the three colour components.

As the predictions of the pixels depend on the previous ones, the first pixel of the image is not processed by the LHE quantiser. Thus, its colour components are included uncompressed in order to allow the decoder to process subsequent pixels (see Section 3.5 'LHE decoding').

#### 3.2 Logarithmical hops

As aforementioned, LHE encodes the errors of the predicted colour components from a set of possible logarithmically distributed values (called hops) for each pixel,  $H(x) = \{h_{-N}, h_{-N+1}, \dots, h_{-1}, h_0, h_1, \dots, h_{N-1}, h_N\}$ . The null hop  $h_0$  means that the error associated with the predicted colour component is lower than the one achieved by a different hop value. The smallest positive and negative hops,  $h_1$  and  $h_{-1}$ , are not logarithmically assigned. LHE algorithm adjusts automatically, within a certain range, the value of the hops  $h_1$  and  $h_{-1}$  for each pixel depending on the previously encoded pixel

through the parameter  $\alpha(x)$ , which is described in Section 3.4 'hop adaptation'. The first time LHE is executed, an initial fixed value for the parameter  $\alpha$  is used, for example,  $\alpha=8$ .

The following equation details the different values of the set of hops  $H(x)$  for a given pixel

$$h_i = \begin{cases} 0, & \text{if } i = 0 \\ \alpha(x), & \text{if } i = 1 \\ -\alpha(x), & \text{if } i = -1 \\ h_{i-1} \cdot (255 - \hat{x}/k(x))^{1/100}, & \text{if } i > 1 \\ h_{i+1} \cdot (\hat{x}/k(x))^{1/100}, & \text{if } i < -1 \end{cases} \quad (2)$$

The image compression rate of LHE depends on the number of hops is considered ( $2N+1$ ), the smallest non-null hops ( $h_1$  and  $h_{-1}$ , defined by the parameter  $\alpha(x)$ ) and the parameter  $k(x)$ . The higher the cardinality of  $H$ , the lower the compression rate of LHE. The parameters  $\alpha(x)$  and  $k(x)$  are responsible for the compactness of the set of hops  $H(x)$  for a given pixel.

Different values of  $k(x)$  allow expanding and shrinking the range covered by the set of logarithmical hops. In image areas where there are high component fluctuations, a low value of  $k(x)$  covers the maximum range and provides better results. On the other hand, in soft detailed areas, a high value of  $k(x)$  shrinks the set of hops, gaining more accuracy for small changes on colour component. The value of  $k(x)$  is determined locally, at each pixel, taking into account the set of surrounding hops

$$k(x) = f(h(a), h(b), h(c), h(d)) \quad (3)$$

For each combination of hops corresponding to the pixels *a*, *b*, *c* and *d* (pixel positions are shown in Fig. 1), there is an optimal value for  $k(x)$ , which minimises the error when a new hop for *x* is chosen. Although a formula could be defined, a pre-calculated table of optimal  $k(x)$  values can be generated testing over all pixels from all images from an image database, and therefore setting the best values for any type of image. This strategy avoids deducing the 'best logic' for the formula and therefore simplifies the problem.

#### 3.3 Adaptive correction

The adaptive correction module takes into account two parameters: the set of possible hops  $H(x)$ , computed in the previous module, and the error associated to the predicted colour component, *e*.

$$e = x - \hat{x} \quad (4)$$

The output of this module is the hop  $h(x)$  from the set  $H(x)$ , that is,  $h(x) \in H(x)$ , that is closer to the above described error. In other words, the hop  $h(x)$  is the quantised error made by the LHE algorithm in the colour component prediction  $\hat{x}$ . This hop  $h(x)$  is the output of the LHE quantiser

$$h(x) = \arg_{h_i} \min(|h_i - e|), \quad h_i \in H(x) \quad (5)$$

In the particular case that there are two different hops with the same

**Table 1** Statistical compression for five hops

Quantized error (float)	Code, bits
$b_0$	1
$b_1$	01
$b_{-1}$	001
$b_2$	0001
$b_{-2}$	00001

distance to the error  $e$ , the hop with the smaller value is chosen. The reason behind this approach is that in statistical compression of images, smaller codes are assigned to small hops (see Section 3.5 ‘coder’).

$$\text{If } \exists (h_j, h_k) \in H(x) \mid |h_j - c| = |h_k - c| \Rightarrow h(x) = h_j \mid i = \min(|j|, |k|) \quad (6)$$

### 3.4 Hop adaptation

Once the quantised error of the actual pixel is assigned, the hop adaptation module updates the parameter  $\alpha(x)$ , which has the same absolute colour component value as the smallest non-null hops  $h_1$  and  $h_{-1}$ , for the next pixel. The parameter  $\alpha(x)$  varies within a certain fixed range  $[\alpha_{\min}, \alpha_{\max}]$ , for example, [4, 8]. According to (2) the parameter  $\alpha(x)$  is used for computing the set of possible hops  $H$  of the next pixel. As aforementioned, an initial start value for the parameter  $\alpha$  is fixed for the first pixel encoded by LHE, for example,  $\alpha=8$ .

The adjustment of the value  $\alpha$  is based on the following rules:

- If the assigned hops of two consecutive pixels  $\{h(x-1), h(x)\}$  are small, that is, they are either null hops  $h_0$  or the smallest non-null hops  $\{h_{-1}, h_1\}$ , then the updated value  $a(x)$  becomes one unit smaller than the smallest positive non-null hop  $h_1$ , up to a certain minimum given by  $a_{min}$ .

$$\text{If } \{h(x-1), h(x)\} \in \{h_{-1}, h_0, h_1\} \Rightarrow \alpha(x) = \max(h_{-1}, 1, \alpha_{-1}) \quad (7)$$

- If the quantised error  $h(x)$ , assigned in the previous module, is different to the null hop or the smallest non-null hops,  $\{h_{-1}, h_1\}$ , then the updated value  $a$  is set to its maximum  $a^*$ .

$$\text{If } \beta_0 \neq 0, \text{ then } b_0(b_0 - b_1) = a = a_0. \quad (8)$$

35 *Cader*

The coder module translates the quantised hops of all pixels into a compressed stream of bits. One possible approach for the compression technique can be based on the existing redundancy of

images across its axes, that is, any pixel is generally similar to the previous one, and thus small hops are more frequently assigned. Although different compression techniques can be applied, this paper recommends the use of Huffman coding algorithm. It has variable-length codes for defining the quantised errors (called hops) based on its frequency of appearance.

However, analysis over two image databases [10, 11] reveals that the smallest hops are assigned in more than 90% of pixels. Therefore, in order to avoid the frequency analysis and enable real-time encoding, an effective statistical compression of hops can be achieved by assigning the smaller codes to the smaller hops. Table I shows an example of a LHE statistical coder with five hops codes.

### 3.6 LHE decoding

The LHE decoder performs similar operations as in the LHE quantiser but in the reverse order. The following lines described the phases of the LHE decoding process for a given pixel  $x$ . It should be noted that previous pixels to  $x$  has been already decoded and therefore the value of the parameter  $a(x)$  for this pixel has been already computed.

1. The binary stream is translated into symbols, in this case, into a certain hop value  $h(x)$ .
  2. The current predicted pixel  $\hat{x}$  is computed as the average of the colour components of the top and left pixels, as in (1).
  3. Given the value of  $\hat{x}$  and  $a$ , the set of hops  $H(x)$  for this pixel are computed by following (2). At this moment, the colour component value of  $h(x)$  is known.
  4. The decoded colour component  $x'$  is computed as follows

$$x' = \pm + h(t) \quad (9)$$

5. Finally, the new value for the parameter  $\alpha$  is computed, following the rules described in Section 3.4 hop adaptation.

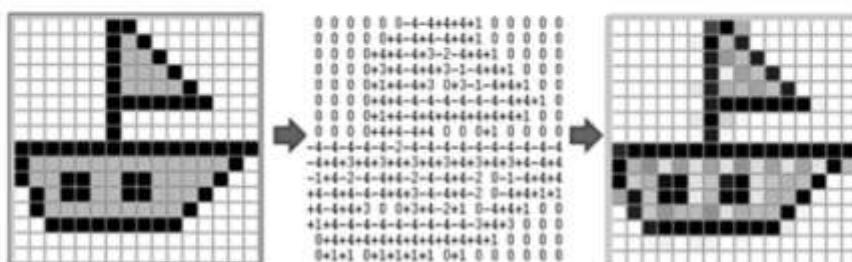
As aforementioned, the first pixel colour components are included in raw-format in the binary stream, in order to enable the decoding process of the subsequent pixels.

Fig. 2 shows the process of encoding and decoding a figure with the basic LHE algorithm. The LHE quantiser assigns a symbol to each pixel (in this case, nine hops are considered). Following the above described steps, the hop symbols are decoded as pixel colour components.

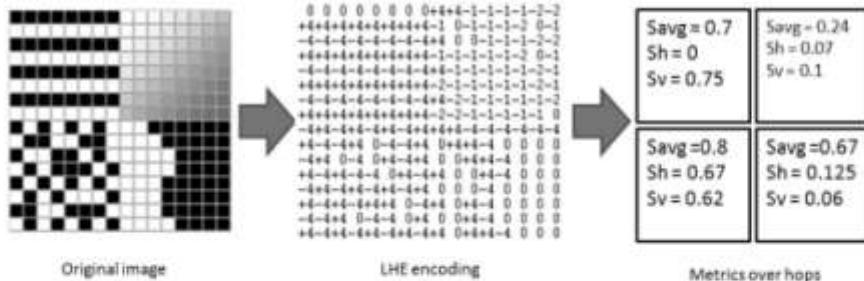
#### 4 LHE-enhanced algorithm

### 4.1 Motivation

In [3, 12], a method known as a region of interest coding is introduced. The main idea behind ROI coding is to segment an



**Fig. 2** LHE encoding and decoding examples



**Fig. 3** Perceptual relevance metric

image into an ROI and the background. If the ROI is coded with higher fidelity than the background, a high compression ratio with good subjective quality can be achieved. ROI coding relies in the fact that the background is less perceptually relevant than the ROI, so the coding errors made in the background are more likely to remain unnoticed than if those errors are made in the ROI.

Analogously, LHE adopts the idea of the perceptual relevance of the different regions of an image, but instead of trying to distinguish between an ROI and the background, LHE generalises the ROI concept, by evaluating the perceptual relevance of each block ( $8 \times 8$  pixels) of the image and encoding each of these blocks accordingly.

#### 4.2 Perceptual relevance evaluation

Intuitively, the perceptual relevance of an image block can be defined as a measure of the importance of the block regarding to the complete image, as perceived by a human viewer [13]. This subsection explains how the perceptual relevance is estimated in the LHE algorithm.

Throughout the development of the LHE algorithm, we realised that LHE-quantised luminance, that is, the output of the LHE quantiser when using luminance as input, can be used as an estimator of the perceptual relevance of an image block. Three metrics have been defined to estimate the perceptual relevance of each block:

- $S_{avg}$ : absolute hop index average (normalised to 0–1). This metric gives a measurement of the size of the hops. A value close to 1 indicates high luminance/chrominance fluctuations, and therefore, it suggests a complex region such as fur or sea foam and so on. It is equivalent to the entropy measurement.
  - $S_h$ : number of changes of hop index's sign when scanning the symbols horizontally (normalised to 0–1). A value close to 0 indicates that the block has low information in the horizontal direction.
  - $S_v$ : number of changes of hop index's sign when scanning the symbols vertically (normalised to 0–1). A value close to 0 indicates that the block has low information in the vertical direction.

In Fig. 3, these metrics are calculated over an example image. As can be seen in the example, blocks containing low information at the horizontal direction, have a low  $S_h$  value. This allows detecting when a block can be down sampled horizontally. These metrics also make possible the distinction of complex regions (which have high  $S_{avg}$ ,  $S_h$  and  $S_v$  values) from soft regions (with low  $S_{avg}$ ,  $S_h$  and  $S_v$  values) and edges (high  $S_{avg}$  value but low  $S_h$  or  $S_v$  depending on the edge direction).

The following subsections explain two methods (based on the perceptual relevance metrics) aimed at improving the performance of the basic LHE algorithm. The architecture of the enhanced algorithm is depicted in the following image (Fig. 4).

The enhanced version of the LHE algorithm uses the aforementioned perceptual relevance metrics as input for a new module: downsampling. This module is aimed to reduce the amount of information to be coded, improving the compression rate while maintaining the subjective quality. This module is described in more detail in the following subsection.

### 4.3 Downsampling

The downsampling module uses the perceptual relevance evaluation to reduce the information to be encoded. For each block, it evaluates the perceptual relevance metrics defined in the previous subsection and it decides if the block can be resized, thus reducing the number of pixels that have to be processed.

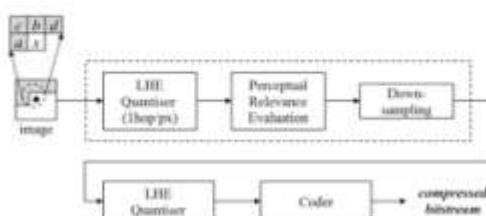
The decision to resize an image block depends on a set of thresholds that are applied to  $S_{avg}$ ,  $S_h$  and  $S_v$ . More concretely, six thresholds are defined, a maximum and a minimum threshold for each perceptual relevance metric. Depending on the actual value of the perceptual relevance metrics with respect to the thresholds, the downsampling module can estimate the type of content of the image block, and its suitability to be resized.

Table 2 summarises the detection rules that are applied to identify the type of content of an  $8 \times 8$  block and the resizing strategy that is

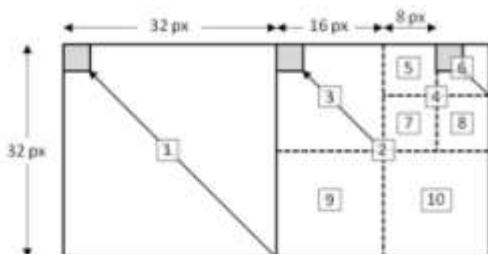
Table 2. Downsampling strategies

Type of content	Detection rule*	Resizing strategy	Binary code
plain luminance	Savg1 and Sh1 and Sv1	4 × 4 pixels (vertical and horizontal)	11
gradated or soft details	Savg1 and Sh1	8 × 4 pixels (horizontal)	01
	Savg1 and Sv1	4 × 8 pixels (vertical)	10
strong and fuzzy details, for example, hair	Savg1 and Sh1	8 × 4 pixels (horizontal)	01
	Savg1 and Sv1	4 × 8 pixels (vertical)	10
other	none threshold is exceeded	no resizing	00

\*<sub>1</sub> = minimum threshold exceeded and \*<sub>2</sub> = maximum threshold exceeded



**Fig. 4** Enhanced LHE algorithm



**Fig. 5** Recursive downsampling procedure

applied in each case. It also shows the binary code that is assigned to each block in order to identify how the block has been resized (vertically, horizontally or both).

Once the image blocks have been analysed (and resized) by the downsampling module, they will be used as input for the LHE quantiser. As can be seen, the main advantage of the

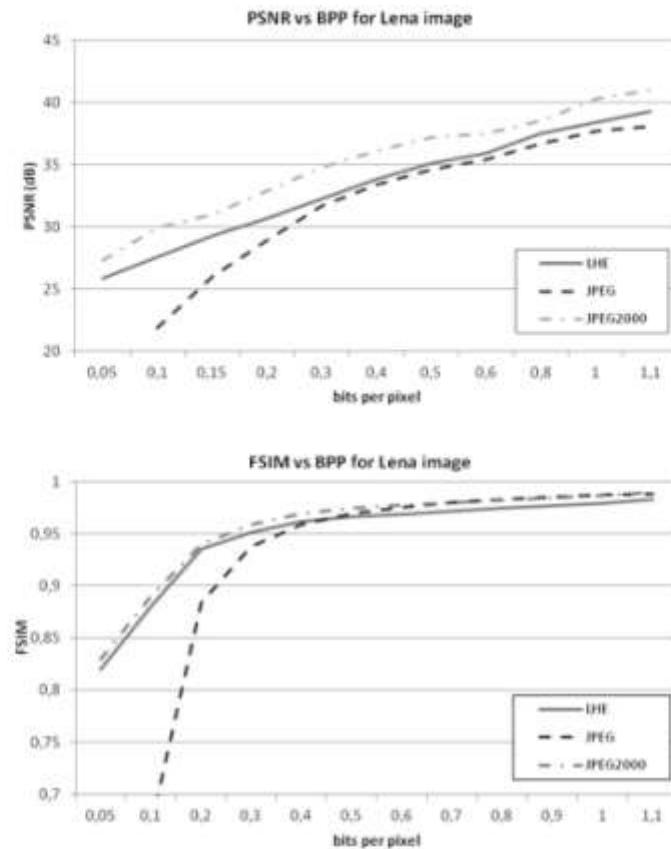
downsampling process is that a certain amount of the 64 pixels blocks will be replaced by resized versions of 16 or 32 pixels.

The threshold selection establishes a trade-off between image quality and compression rate. If the thresholds are very restrictive (maximum and minimum thresholds close to 1 and 0, respectively), more quality (and less compression rate) will be achieved. More details about the threshold effect will be given in Section 5.

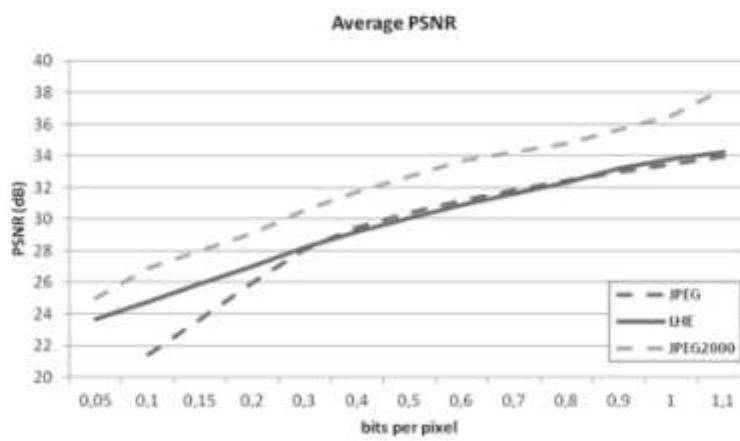
In order to take the maximum advantage of the downsampling module, a recursive procedure, using different block sizes, can be used. Let 'n' be number of iterations of this recursive procedure. First, the image is divided into macroblocks of  $2^{2n} \times 2^{2n}$  pixels. For each of these macroblocks, the perceptual relevance metrics are computed, and the resizing rules are checked. If the macroblock can be resized (according to the aforementioned rules) it will be resized and the next macroblock will be processed. If the macroblock cannot be resized, then it is divided into four blocks and the previously described downsampling technique is applied recursively to each of these inner blocks. This recursion is applied while the block size is equal or bigger than  $8 \times 8$  pixels. The calculation of the perceptual relevance metrics for a macroblock does not require additional processing, because the additive nature



**Fig. 6** JPEG, LHE and JPEG2000 comparison



**Fig. 7** PSNR and FSIM R-D diagrams for 'Lena' image



**Fig. 8** Average PSNR diagram

```

//Pixel Prediction
 $\hat{x} = (a + b)/2$ 

//Best Hop Selection
 $error_{min} = |x - \hat{x}|$ 
 $hop_{selected} = 0$ 

//Positive Hops
if ( $x > \hat{x}$ ) then
    foreach ( $h_i$  in  $h_1 \dots h_4$ ) do //hops can be precomputed
        error =  $|x - h_i|$ 
        if ( $error < error_{min}$ ) then
             $hop_{selected} = i$ 
             $error_{min} = error$ 
        else
            break
        end
    end
else
    ...
end

//Negative Hops (same loop for  $h_{-1} \dots h_{-4}$ )
else
    ...
end

```

**Fig. 9 LHE quantisation algorithm**

of these metrics allows them to be computed by adding the corresponding metrics of the macroblock inner  $8 \times 8$  blocks. The perceptual relevance metrics for each  $8 \times 8$  block are given by the perceptual relevance evaluation module, so the computational overhead in the downsampling module is low.

The following figure shows an example where the recursive downsampling procedure is applied to a  $32 \times 64$  pixels image, using  $n = 3$  levels of iterations (Fig. 5).

The numbers enclosed in boxes represent the processing order for each block and macroblock. As can be seen, in the first place, the  $32 \times 32$  macroblock on the left is processed. As it can be resized, no further processing is required. Next, the  $32 \times 32$  macroblock on the right is processed. This macroblock cannot be resized so its  $16 \times 16$  pixels inner macroblocks will be processed. The first  $16 \times 16$  macroblock (on the top and the left) can be resized, so the algorithm continues with the next one. The second  $16 \times 16$  macroblock cannot be resized, so it is required to process its  $8 \times 8$  inner blocks. This process continues until the complete image has been analysed.

In the decoder side, the downsampling procedure can be easily reverted by applying an interpolation algorithm over the downsampled version of the decoded blocks in order to restore their original size.

LHE does not specify which downsampling or interpolation technique should be used. However, the selected technique will have an effect over the performance and the quality achieved by LHE.

## 5 Experimental results

The following experimental results have been obtained by applying LHE algorithm with nine hops and by using pixel averaging downsampling and bilinear interpolation.

### 5.1 Image quality

LHE has been tested using two image databases: Kodak lossless true colour image suite [10] and USC-SIPI image database (miscellaneous volume) [11]. LHE provides, in most cases, a good

objective quality (PSNR) but the even better subjective quality, because bigger errors are located at pixels with strong contrast with surrounding ones, where subjective quality impact is minimised.

Edge information is vitally important in the perception of images [14]. However, this information is usually distorted when the image is encoded using DCT (discrete cosine transform) or other frequency-based technique [15, 16]. Fig. 6 shows a subjective quality comparison between JPEG, LHE and JPEG2000. It can be seen that the LHE edges are cleaner than JPEG edges. Furthermore, LHE noise has less impact on the subjective quality compared to typical DCT-based algorithms noise because of the lack of visible artefacts at block boundaries. JPEG2000 obtains higher PSNR values and slightly better subjective quality than LHE. At low bitrates the most important LHE degradation is because of the strong downsampling performed by the encoder in certain image areas, which may lead to blurriness when they are interpolated by the decoder. Despite the aforementioned effect, in terms of subjective quality, LHE is closer to JPEG2000 than JPEG.

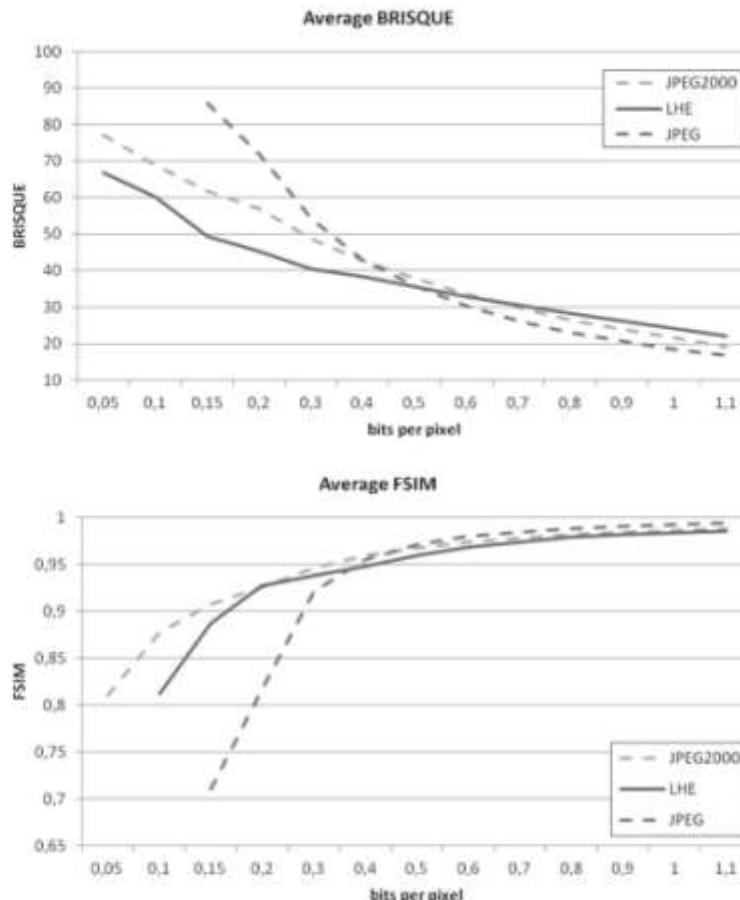
Fig. 7 curves are the PSNR and FSIM rate-distortion ( $R-D$ ) diagrams for Lena image. LHE performs better than JPEG at low bitrates and provides quite similar quality at high bitrates. Regarding FSIM, LHE follows JPEG2000 trend.

Fig. 8 shows the average PSNR using all the images contained in the Kodak lossless true colour image suite and the USC-SIPI image database. Analogously to the case of Lena image, for all the images of the databases, LHE outperforms JPEG at low bit rates.

To provide more evidences of the LHE quality performance the following figure shows a comparison between LHE, JPEG and JPEG2000 using blind/referenceless image spatial quality evaluator (BRISQUE) [17], a metric that evaluates the loss of 'naturalness'

**Table 3** LHE complexity (linear) without parallelisation

Image resolution, px	184k (429 × 429)	414k (720 × 576)	921k (1280 × 720)	2073k (1920 × 1080)
quantisation time, ms	2.8	6.7	15	29.49



**Fig. 10** Image quality metrics: non-reference (BRISQUE) and full-reference (FSIM)

in the image as a consequence of the compression process; and FSIM [18], a full reference image quality metric consistent with subjective evaluations. It should be noted that quality and BRISQUE are inversely proportional while quality and FSIM are directly proportional.

## 5.2 Algorithm performance

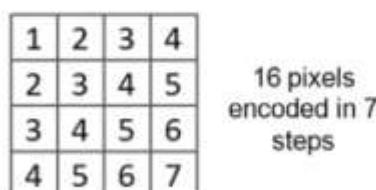
One of the main advantages of LHE is its simplicity. LHE time complexity is  $O(n)$  and its memory complexity is  $O(1)$ . A simplified implementation of the LHE algorithm for nine hops is described in Fig. 9.

It can be observed that LHE only requires a constant number of basic operations per pixel (assignments and comparisons), that is, its complexity is  $O(n)$ . In contrast, DCT and DWT (discrete wavelet transform) have  $O(n \log n)$  or higher computational complexity [19]. This advantage makes LHE an extremely fast procedure. Our Java prototype (without parallelisation) tested on Intel i5-3320@2.6GHz achieves the quantisation times that are shown in Table 3, confirming the linear complexity of LHE.

Using parallelisation, the encoding time can be significantly reduced. LHE allows the parallel processing of those pixels which have the top and left pixels processed (needed for colour components prediction). Given an image of  $N \times N$  pixels, the

complete parallel process will take  $2N-1$  steps. In Fig. 11 pixels are labelled with a number. Pixels labelled with the same number can be encoded in parallel (in the same step).

This parallelisation strategy can be applied to blocks instead of pixels. Thus, the enhanced LHE algorithm can take benefit from parallelisation, where every block is downsampled at different ratio depending on its perceptual relevance metrics. In this case, the encoding of the blocks is made in the same order as depicted in Fig. 11. This figure refers to pixels but the same parallelisation strategy can be applied to blocks composed of  $M \times M$  pixels.



**Fig. 11** Parallel processing of  $N \times N$  pixels in  $2N-1$  steps

## 6 Conclusions

LHE is a lossy compression algorithm suitable for static images based on adaptive logarithmic quantisation. The main advantages of the algorithm are its low computational complexity  $O(n)$  and its performance in terms of image quality, specially at low bitrates. Two main contributions make possible this performance. In the first place, LHE proposes a logarithmic quantisation of the error between pixel colour component predictions and the actual value of such components. This quantisation is based on Weber-Fechner law and it has been proven as a linear and quality effective compression procedure. In the second place, a downsampling strategy, based on perceptual relevance metrics (calculated using LHE-quantised luminance), provides a fast procedure to protect the image ROIs, optimising the overall image quality. Furthermore, the algorithm design supports the parallel processing of different image blocks, thus reducing the encoding time.

The results of this paper point to several interesting directions for future work:

- Image quality improvement based on more complex pixel prediction, downsampling strategies and interpolation techniques.
- Modification of LHE algorithm for lossless image compression.
- Application of LHE approach in the time-domain for video and audio compression.

## 7 Acknowledgment

The authors wish to thank the Spanish Science & Tech Ministry which funds this research through 'INNPACTO' innovation program IPT-2011-1683-430000.

## 8 References

- 1 Lloyd, S.: 'Measures of complexity: a nonexhaustive list', *IEEE Control Syst. Mag.*, 2001, **21**, (4), pp. 7-8
- 2 Jayant, N., Johnston, J., Safranek, R.: 'Signal compression based on models of human perception', *Proc. IEEE*, 1993, **81**, (10), pp. 1383-1422
- 3 Cnud, D.S., Ebshini, T., Larsson, M., Askleif, J., Christopoulos, C.: 'Region of interest coding in JPEG2000 for interactive client/server applications', *IEEE Third Workshop on Multimedia Signal Processing*, 1999, pp. 389-394
- 4 Hasan, S.A., Hasan, M.: 'Spatial domain lossless image data compression method', *Int. Conf. on Information and Communication Technologies (ICICT)*, July 2011, pp. 1-4
- 5 Hasan, M., Nur, K., Noor, T.B., Shakur, H.B.: 'Spatial domain lossless image compression technique by reducing overhead bits and run length coding', *Int. J. Comput. Sci. Inf. Technol.*, 2012, **3**, (2), pp. 3650-3654
- 6 Hasan, M., Nur, K.: 'A novel spatial domain lossless image compression scheme', *Int. J. Comput. Appl.*, 2012, **39**, (15), pp. 25-28
- 7 Huang, S.C., Chen, L.G., Chang, H.C.: 'A novel image compression algorithm by using Lap-Esp transform', *Proc. 1999 IEEE Int. Symp. on Circuits and Systems (ISCAS'99)*, July 1999, vol. 4, pp. 17-20
- 8 Rabban, M., Jones, P.W.: 'Adaptive DPCM', in 'Digital image compression techniques', (SPIE - The International Society for Optical Engineering Press, 1991)
- 9 Weinberger, M.I., Seshadri, G., Sapiro, G.: 'The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS', *IEEE Trans. Image Process.*, 2000, **9**, (6), pp. 1309-1324
- 10 Krish, 'Kodak lossless true color image suite', <http://www.rtk.us/graphics/kodak>, accessed January 2014
- 11 'USC-SIPI image database', <http://sipi.usc.edu/databases/database.php?volume=msic>, accessed January 2014
- 12 Christopoulos, C.A., Askleif, J., Larsson, M.: 'Efficient methods for encoding regions of interest in the upcoming JPEG2000 still image coding standard', *IEEE Signal Process. Lett.*, 2000, **7**, (9), pp. 247-249
- 13 Moreno, J.M.: 'Perceptual criteria on image compression', Ph.D. dissertation, DCC, UAB, Barcelona, Spain, 2011
- 14 Zhang, C.N., Wu, X.: 'A hybrid approach of wavelet packet and directional decomposition for image compression', *IEEE Canadian Conf. on Electrical and Computer Engineering*, Edmonton, Alta, Canada, December 1999, vol. 2, pp. 735-740
- 15 Zhou, Z., Venetsanopoulos, A.N.: 'Morphological methods in image coding', *IEEE Int. Conf. on Acoustics, Speech, and Signal Process (ICASSP-92)*, San Francisco, CA, USA, March 1992, vol. 3, pp. 481-484
- 16 Popovic, I., Withers, W.D.: 'Locating edges and removing ringing artifacts in JPEG images by frequency-domain analysis', *IEEE Trans. Image Process.*, 2007, **16**, (5), pp. 1470-1474
- 17 Mital, A., Moorthy, A.K., Bovik, A.C.: 'No-reference image quality assessment in the spatial domain', *IEEE Trans. Image Process.*, 2012, **21**, (12), pp. 4695-4708
- 18 Zhang, L., Zhang, D., Mou, X., Zhang, L.: 'FSIM: a feature similarity index for image quality assessment', *IEEE Trans. Image Process.*, 2011, **20**, (8), pp. 2378-2396
- 19 Kok, C.W.: 'Fast algorithm for computing discrete cosine transform', *IEEE Trans. Signal Process.*, 1997, **45**, (3), pp. 757-768

A continuación se incluye otro artículo recientemente submitido (aun no publicado) a la misma revista

## Logarithmical Hopping Encoding (LHE): physiological fundamentals

Jose J. García Aranda<sup>1</sup>, Joaquín Navarro Salmerón<sup>2</sup>, Mario Cao Cueto<sup>2</sup>, Marina González Casquete<sup>2</sup>, Francisco González Vidal<sup>2</sup>

<sup>1</sup> Video Architecture Department

Alcatel-Lucent

Madrid, Spain

jose\_javier.garcia\_aranda@alcatel-lucent.com

<sup>2</sup> Departamento de Ingeniería de Sistemas Telemáticos (DIT)

Universidad Politécnica de Madrid (UPM)

Madrid, Spain

{navarro, mcao, mgonzalez,vidal}@dit.upm.es

**Abstract-** LHE (Logarithmical Hopping Encoding) is a computationally efficient spatial domain lossy image compression algorithm that models and exploits the physiological behaviour of the human eye. In this article, the physiological fundamentals of LHE are reviewed and the LHE equations are simplified and improved. Three different behaviours of the human eye are modelled by LHE: the logarithmical response to stimulus (Weber-Fechner law), the minimum contrast threshold for change detection and the adaptation process of the eye to the local average luminance. The aforementioned behaviours are modelled linearly, resulting in a final linear algorithm; its time complexity is  $O(n)$  and its memory complexity is  $O(1)$ . This feature makes it suitable for building image and video coders for applications in which a fast and predictable encoding time is required. Thanks to its physiological fundamentals, the encoding errors produced by LHE algorithm are mainly condensed in abrupt borders and noisy areas where the algorithm accuracy is lower but human tolerance is higher. LHE provides a fixed bit rate which depends on each image structure.

**Keywords-** image processing; image compression; image representation

### I. Introduction

Lossy transform compressors, such as JPEG, JP2K, H264 intra-frame among other popular formats, started to become popular when in 1965 the first Fast Fourier Transform (FFT) [1] basic ideas were born. Thenceforth, lots of improvements in frequency domain processing have emerged, usually focused on improving the compression rates by compacting more the signal energy. As a consequence, the computational complexity of these compressors [2] has significantly increased.

While frequency transformations are complex operations that represent a significant cost for nowadays real-time applications, algorithms based on space domain would improve computational complexity as they do not require frequency transformations such as FFT or Discrete Cosine Transform (DCT). However, they have not been as popular as frequency domain lossy compressors. In this context, the LHE algorithm works in the space domain thereby breaking with the trend of lossy "transform compressors" and improving performance and quality. This is possible by using a physiological model of the human eye, which is specifically designed to reduce the amount of information to be encoded.

In this paper we analyze the physiological fundamentals of LHE algorithm. This algorithm has been designed applying a methodology that includes, at first stage, the study of physiology of human eye perception. At second stage, the human eye behaviour is modelled discarding any non-linear approach, in order to build a fast and low computational algorithm. Finally, the proposed models are iteratively tested against image databases [3] [4], selecting the most reasonable approach in terms of computational costs and quality (PSNR).

The remainder of this paper is structured as follows. Section II provides a throughout state of the art analysis. Section III describes the main human eye physiological behaviours which LHE exploits to achieve linear time image compression. Once the foundations of LHE are introduced in section IV, details about the parallelizable nature of LHE and the proposed entropy encoder are given in the sections V and VI respectively. Finally, some comparative results are exposed and the main conclusions of the paper are discussed.

## II. Related Work

By taking into account the properties and limitations of the Human Visual System (HVS), images can be more efficiently compressed, colours more accurately reproduced, prints better rendered, among other major advantages [5].

Most image lossy compressors take into account three main limitations of HSV:

First, human colour perception is not directly related to the cone responses, but rather to their differences [5]. These are represented by an achromatic channel and two opponent-colour channels, which code red-green and blue-yellow colour differences. In image processing this coding is exploited in several colour spaces such as Y CbCr, where Y is the luminance channel and Cb, Cr the colour-difference channels.

Another exploited HSV physiological behaviour is the perception on multiple channels that are tuned to different ranges of spatial frequencies and orientations. This behaviour is well matched by a multi-resolution filter bank or wavelet decomposition.

Finally, human eye ability to differentiate between intensity levels is vitally important in image processing, especially in digital image processing where images are presented as a discrete set of intensity levels. Some lossy image compressors (like JPEG) take into account that the human eye is more sensitive to small changes in brightness in smooth image areas (low-frequency areas) than the same changes taking place in areas where the brightness has very large variations (high-frequency). This fact is one of the basics JPEG principles, applied in its quantization phase. In this manner, JPEG uses a lossy form of compression based on the DCT. Image is splitted into blocks of 8x8 pixels and, for each block, DCT is applied. The amplitudes of the different frequency components obtained are quantized considering that high-frequency components should be stored with lower accuracy than the low-frequency components. This

strategy is used in most of the transform compressors (JPEG, JP2K, H264 intra-frame, JPEG-XR, WebP, etc.).

Other additional human eye physiological behaviours can be taken into account without frequency transformation, in space domain. These additional physiological behaviours constitute the LHE basis and they are described in the next section.

### III. Human eye physiological behaviour

The behaviour of the human eye involves focus capabilities, movement, and response to stimulus. When the human eye is watching a screen with images or videos, its behaviour is simplified because the distance from eye to screen is nearly constant and an accommodation process (focus) does not take place. In this context, three different behaviours occur simultaneously: logarithmical response to stimulus (Weber-Fechner law), minimum contrast threshold for change detection and adaptation process of the eye to the local average luminance.

#### Logarithmical response to stimulus:

The Weber-Fechner law [6], states a quantitative relation between physical stimulus and how it is perceived.

$$S = C \cdot \log \frac{I}{I_0} \quad (1)$$

Where C is a constant, I is the intensity of the stimulus, S the perception (difference between luminance of an object and its background) and  $I_0$  is the minimum intensity of stimulus that human eye can perceive. Basically, this law establishes that if a stimulus intensity grows like a geometric progression, the perception will evolve as an arithmetic progression. This phenomenon relies on the pupil area changing mechanism, which decreases linearly when light is increased geometrically [7].

In conclusion we can state that the human eye perception is non-linear and that it encodes sensory information based on the geometric fluctuations of the image luminance. This logarithmical information is transmitted from the eye to the brain for further processing.

**Minimum contrast threshold for change detection:**

The detection threshold of the human eye [8] is what is called Weber fraction [9], which is basically the minimum luminance change with respect to the image background that can be detected by the human eye [7] [10]. It is expressed as:

$$\text{Weber fraction} = \frac{\Delta I}{I} \quad (2)$$

Consider an area with intensity  $I + \Delta I$  surrounded by a background of intensity  $I$ , as shown in Fig. 1 (a). The minimum value of  $\Delta I$  noticed by the human eye is the value of the Weber fraction.

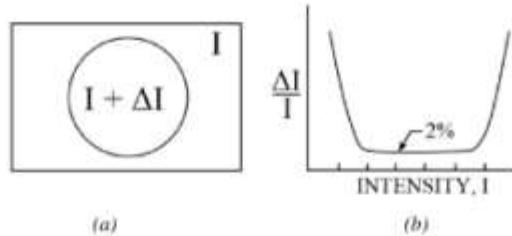


Fig. 1. Weber fraction for human eye

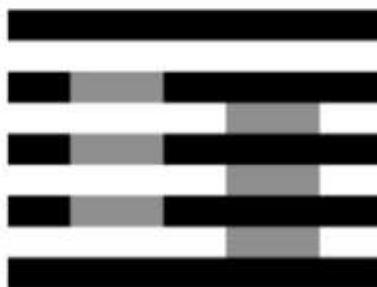
Typically, the minimum value of the Weber fraction of the human eye is 0.02, i.e. the human eye can perceive a difference between two image areas if their luminance ratio is at least 2%. However, this 2% is increased for extreme luminance values, i.e. the eye becomes less sensitive in very dark or very bright image areas [11] (see Fig. 1 (b)).

Many times we can begin to recognize or distinguish objects by increasing the level of lighting. This explains how when go from a low intensity to medium intensity, the Weber fraction gets reduced up to 2%. If we continue increasing the light level, after certain level we enter in glare

zone and the fraction of Weber increases again and our ability to differentiate objects gets reduced.

#### Adaptation process of the eye to the local average luminance:

The following example (Fig. 2) [12] created by Edward H. Adelson, Professor of Vision Science at MIT illustrates the process of adaptation of the human eye to the local average brightness



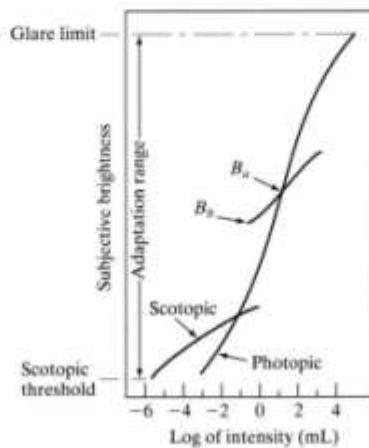
*Fig. 2. Adaptation to the local average luminance*

Although the two shades of gray are identical, we perceive more brilliant the left one. It is because the white stripes increase the local average brightness on the left. In general, any tone far away from the local average brightness is perceived by human eye with less accuracy, either by excess or defect. Because of the same reason, any very dark object placed on a white background is perceived by human eye as black. In other words, at higher contrast ratios, human eye has lower accuracy because the eye has only its maximum accuracy at tones near of local average brightness.

In fact, once the human eye adapts to the local average brightness, it is not able to perceive more than 50 levels of different brightness [13]. Tones far from local average brightness are perceived without accuracy [14], which sometimes results in eye overshoot around limits of regions of different intensities.

The total range of light intensity levels to which the HVS can adapt is on the order of magnitude of  $10^{10}$ - from the scotopic threshold to the glare limit (Fig. 3). However, the visual system

cannot operate over the complete range simultaneously. Due to the adaptation to the local average luminance ( $B_a$  in Fig. 3) [15], the number of differentiable levels of grey by human eye is around 50, from black ( $B_b$  in Fig. 3) to a certain quantity above  $B_a$ .



*Fig. 3. Range of subjective brightness sensations showing a particular adaptation level*

#### IV. Modelling of human eye physiological behaviour

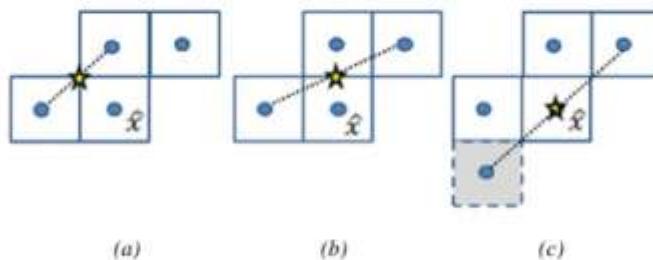
The previous section described the main physiological characteristics of the human eye. In this section, these characteristics are modelled and exploited to design an efficient image compression algorithm in terms of computational complexity and compression rate.

Modelling the behaviour of the eye's response to the Weber's law involves the definition of two major aspects:

- Compute the background intensity for each pixel.
- Encode information using logarithmic hops between the computed background value and the signal.

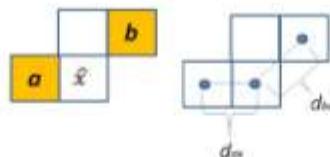
The background intensity may be defined as an estimation of the signal value, since there is no difference between the background and the signal in smooth or graded areas. We can consider different options for the computation of this background intensity (see Fig. 4).

The prediction strategy, shown in (Fig. 4 b), produces a better approximate background signal than the one that uses the left and upper pixel (Fig. 4 a) because the X coordinate of  $\hat{x}$  is more centered. To center the vertical coordinate Y we should have to choose the bottom-left instead of left pixel but as LHE operates in scanlines, said pixel has not yet been calculated (Fig. 4 c).



*Fig. 4. Alternatives for background computation*

We estimate the background colour ("hop null" or  $h_0$ ) of the component (Y, U or V) of each pixel as the weighted average of the colour components of the left pixel (a) and upper-right (b), according with their corresponding distances to the pixel (Fig. 5).



$$d_{ax} = 1, \quad d_{bx} = \sqrt{2}$$

$$\text{weight } a = \frac{d_{bx}}{(d_{ax} + d_{bx})}$$

$$\text{weight } b = \frac{d_{ax}}{(d_{ax} + d_{bx})}$$

$$\hat{x} = \text{weight}_a \cdot a + \text{weight}_b \cdot b$$

*Fig. 5. Background computation*

LHE defines a set of positive and negative hops referenced to the computed prediction of background colour ( $h_0 = \hat{x}$ ), with which LHE will code the logarithm of the difference between the signal and prediction. The hops are related to each other through a geometrical

ratio  $r$ . The Basic LHE algorithm transforms each original signal value (pixel) into the closest hop and only stores the hop number. If two hops are located at the same distance from the signal value, then the smaller hop is chosen. The prediction  $h_0$  can take any value of the available interval (0-255). Therefore we consider two ratios to distribute properly the hops across the interval:  $r$  for positive hops and  $r'$  for negative hops.

$$\begin{aligned} h_2 &= h_1 \cdot r & h_3 &= h_2 \cdot r & h_4 &= h_3 \cdot r \\ h_{-2} &= h_{-1} \cdot r' & h_{-3} &= h_{-2} \cdot r' & h_{-4} &= h_{-3} \cdot r' \end{aligned} \quad (3)$$

The smallest considered hop is  $h_1$  (or  $h_{-1}$ ), and its absolute value represents the modeling of the threshold for change detection. Any signal change below  $h_0 \pm h_1/2$  will not be reflected because in this case  $h_0$  is closer than  $h_1$  to the signal. The Weber fraction  $\frac{(h_1/2)}{h_0}$  should be around 2% for medium values of  $h_0$ , and therefore  $h_1$  should be  $h_1 = 0.04 \cdot h_0$ .

A simplification assigning a constant value to  $h_1 = 4$  results in a graph (Fig. 6) that matches approximately the Weber fraction for low and medium values of  $h_0$  detecting around 2% for medium values of  $h_0$ . Glare range is not interesting because display devices are calibrated to avoid it.

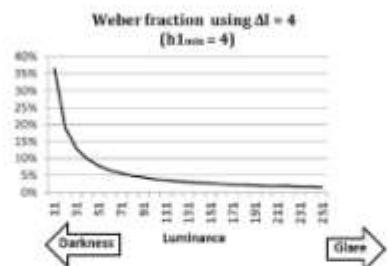


Fig. 6. Modelling the threshold for change detection

As aforementioned, LHE only stores the identifier of the closest hop to the signal and not the value of the corresponding luminance. Therefore LHE reduce the original degrees of freedom (256) into just 9 possibilities (one null hop plus 4 positive hops and 4 negative hops).

The ratio ( $r$  and  $r'$ ) of the hops distribution should avoid exceeding the maximum signal value (which is 255). Therefore the ratio must be bumped to a maximum value. The maximum values which limit the maximum hop to the 80% of the positive and negative ranges are calculated as follows:

$$r_{\max\_pos} = \left( \frac{0.8 \cdot (255 - h_0)}{h_1} \right)^{1/3}, \quad (4)$$

$$r_{\max\_neg} = \left( \frac{0.8 \cdot h_0}{h_1} \right)^{1/3}$$

A high value of “ $r$ ” spreads out the set of hops across the available interval (suitable for areas of high fluctuations of brightness) and a low value of “ $r$ ” compact all hops in a very tiny interval (suitable for soft areas). An intermediate value ( $r = 2.5$ ) offers good results at any type of area. The value of “ $r$ ” is different for positive hops and negative hops.

$$r = \min (2.5, r_{\max\_pos}) \quad (5)$$

$$r' = \min (2.5, r_{\max\_neg})$$

The number of possible hops depends on the trade-off between quality and needed quantity of information that has to be stored. Experimental results reveal 9 hops as a suitable value (Fig. 7). Beyond this number, the increase on quality does not compensate the storage costs.

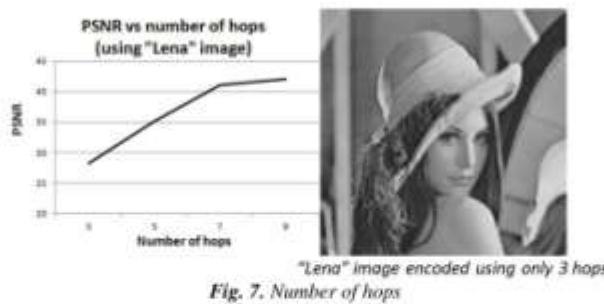


Fig. 7. Number of hops

The adaptation modelling of the human eye to the local average luminance is achieved by modifying the value of  $h_1$ . The modeling of the threshold for change detection is achieved by assigning a minimum value for  $h_1 = 4$ . In turn, to model the adaptation process, the maximum value of  $h_1$  is modified based on signal fluctuations. It should be taken into account that, since

all hops are related each other through “r”, the modification of  $h_1$  affects all hops. For example if  $h_1$  is increased, then all hops are dispersed, i.e. they can reach farther luminance values.

If the signal is stable, the assigned hops will be small and the value of  $h_1$  will be reduced gradually up to  $h_{1\ min} = 4$ . This strategy compacts the hops in a reduced interval around the background estimation  $h_0$ , gaining accuracy around this centered value. On the contrary, if a large hop is presented,  $h_1$  will be increased immediately to a maximum value, thereby expanding the hops coverage to a greater range of luminance and enabling a better accuracy approaching the signal by a hop. The following pseudo-code (Fig. 8) summarizes this adaptive process:

```
//tunning hop1 for the next hop
min_h1=4
max_h1=10
small_hop=false
for each pixel do
    if (selected_hop<=h1 && selected_hop>=h-1) then
        small_hop=true
    else
        small_hop=false
    endif
    if (small_hop && previous_small_hop) then
        h1=h1-1
        if (h1<min_h1) then
            h1=min_h1
        endif
    else
        h1=max_h1
    endif
    previous_small_hop=small_hop
```

*endfor*

*Fig. 8. Adaptation mechanism pseudocode*

Once each hop is assigned, the LHE algorithm stores the identifier of the hop and not the value of the corresponding intensity. Taking into account that the intervals defined by each hop are not equal in size, neither symmetrical, the most suitable value of luminance for each hop must be the centre of the covered interval.

Considering the overall process, Basic LHE allows to transform the signal into hops and the original image can be decoded from hops [16] (see Fig. 9).



*Fig. 9. Encoding and decoding*

## V. Parallel computation

LHE time consumption is strictly linear with the number of pixels of the whole image ( $N$ ).

Taking into account that multiple hops can be computed simultaneously, a drastic reduction of the processing time can be achieved by a LHE implementation based on a parallelization strategy. The only constraint to compute a hop is that the left and up-right hops should be computed previously. In the Fig. 10, an 8x8 image ( $M=8$ ) is represented, and every pixel is labelled with a number. All pixels with the same label can be computed at the same step. The image is computed in 22 steps.

1	2	3	4	5	6	7	8
2	4	5	6	7	8	9	10
3	6	7	8	9	10	11	12
4	8	9	10	11	12	13	14
5	10	11	12	13	14	15	16
6	12	13	14	15	16	17	18
7	14	15	16	17	18	19	20
8	16	17	18	19	20	21	22

*Fig. 10. Parallel implementation*

The following formula (eq. 6) shows the relationship between serial computation time and parallel computation.

$$t_p = \frac{3M - 2}{M^2} \cdot t_s \cong \frac{3}{M} \cdot t_s \quad (6)$$

Where:

$t_p$  is the time consumed by parallel implementation.

$t_s$  is the time consumed by a serial implementation.

M is the number of pixels of the largest side of the image.

This parallelization strategy reduces the computation time up to 170 times (for a 512x512 image) or up to 426 times (for a 1280 x 720p image), but depends on the number of available parallelizable processor resources.

## VI. From Hops to bits

This stage is simple and fast, prioritizing speed versus compression ratio. A static Huffman entropy compressor is part of this process, which can be overcome by other more advanced (and computationally complex) entropy encoders.

The process that transforms hops into binary codes comprises two steps:

- First the hops are converted into symbols, taking into account the spatial redundancies of the image. The same symbol identifier sometimes will represent one hop and sometimes will represent a different one, according to certain rules of redundancy. This step transforms the hop indexes into a more compacted distribution of symbol identifiers.
- Then, an entropy compressor (static Huffman) is executed, assigning variable length codes to the symbol identifiers.

Regarding spatial redundancy, any pixel is usually similar to the one located at its left (horizontal redundancy) and/or to the one located at its top (vertical redundancy). In LHE this results in the more frequent hop being the hop  $h_0$ , and / or the hop whose value corresponds to the hop located at the upper position ("up" hop).

The strategy to perform the translation from hops into symbols consists of checking the hop to translate in the order shown in Table 1 until match the hop. The number of checks until match the hop is the value of the symbol.

Table 1. Check order

Hop	Check order
$h_0$	1
up	2
$h_1$	3
$h_{-1}$	4
$h_2$	5
$h_{-2}$	6
$h_3$	7
$h_{-3}$	8
$h_4$	9
$h_{-4}$	10

After hops to symbol conversion, a static Huffman is applied, resulting in the final Basic LHE compressed file.

## VII. Experimental results

The following results are obtained using samples (Fig. 11) from the test image databases [3] and [4], using “Basic LHE” algorithm (not “Advanced LHE” algorithm [16]). Each image is compressed at certain non-eligible bit-rate. This limitation of “Basic LHE” has been overcome by “Advanced LHE”.

Non-eligible bit-rate implies that Rate-Distortion diagrams are not applicable for “Basic LHE” but in spite of it, different examples at fixed bit-rates can be analyzed and compared with JPEG and/or JPEG2000. Basic LHE offers an intermediate level of quality, keeping a simple and linear computation.



*Fig. 11. Test images*

The resulting PSNR and SSIM of Basic LHE encoding are shown in Table 2 and Table 3. For SSIM computation we have used the “kanzi” library [17]. Every image is compressed at the same bit-rate by all compressors. In each case the bit rate is obtained after LHE binary encoding process.

Table 2. Quality PSNR results of Basic LHE

Gallery	Image name	bit-rate	PSNR (dB)		
			JPEG	LHE	JP2K
USC-miscelanea	Lena	1,8	40,8	43,35	44
	Baboon	2,7	33,6	34,9	38,32
	Peppers	2	38,8	40,34	43
Kodak	4-1-01	1,8	41,2	43,35	44,5
	kodim07	1,7	42,6	43,35	46,7
	Kodim14	2,2	37,2	38,58	41,4
	Kodim16	1,9	40,7	43,35	44,7
	kodim22	2	39	40,3	42,8

Table 3. Quality SSIM results of Basic LHE

Gallery	Image name	bit-rate	SSIM		
			JPEG	LHE	JP2K
USC-miscelanea	Lena	1,8	0,9639	0,9765	0,9806
	Baboon	2,7	0,9668	0,9785	0,9945
	Peppers	2	0,9814	0,9892	0,9937
Kodak	4-1-01	1,8	0,9833	0,9794	0,9843
	kodim07	1,7	0,9941	0,9883	0,9957
	Kodim14	2,2	0,9863	0,9902	0,9942
	Kodim16	1,9	0,9902	0,9863	0,9941
	kodim22	2	0,9746	0,9794	0,9917

LHE coding time is strictly linear with N (number of pixels of the whole image), taking  $1,5 \cdot 10^{-5}$  milliseconds per pixel (experimental Java prototype tested on Intel i5-3320@2.6Ghz). The results show that LHE takes 18 ms for a 1280x960 gray image without any parallelization and 0.05 ms using maximum parallel implementation.

A rough comparison with JPEG compression time may be achieved considering that libjpeg v6b (running in 2.8 GHz Intel Intel Xeon W3530 quad-core) takes around 35ms for 1280x960 gray image [18]. This CPU is 9% slower than our CPU running the LHE Java prototype, but allows certain level of comparison.

### VIII. Conclusions

This paper presents the close relationship between LHE quantization mechanism and the human eye physiological behaviour.

LHE establishes the modeling of the human eye with linear algorithms as the basis of the encoder. As aforementioned, the obtained results are promising in terms of image quality and computation speed. This feature makes it suitable for building image and video coders for applications in which a fast and predictable encoding time is required.

The physiological behaviours of the eye modelled or applicable in LHE are summarized in Table 4.

Table 4. Summary of eye physiological behaviours versus LHE

Human eye behaviour	Applicability on LHE
Logarithmical response	LHE makes a logarithmical quantization of signal into hops
Weber fraction	LHE sets a minimum value for $h_1$
Adaptation to local average brightness	LHE compacts the hops if signal does not change significantly, reducing $h_1$
More sensitive to luminance than chrominance	LHE is compatible with YUV model

Thanks to LHE simplicity, compression times using a LHE non-optimized Java prototype are faster than JPEG implementations written in C or assembler, while obtaining higher compression rates.

The results of this paper point to several interesting directions for future work:

- Application of LHE approach in the time-domain for audio compression, using a linear model for hearing sense
- Advanced compression techniques for entropy compressor.

#### IX. Acknowledgement

The authors wish to thank to Carlos M. Lentisco Sánchez for his review and technical tips

The authors wish to thank the Spanish Ministry of Economy and Competitiveness which funds this research through "RETOP" innovation program RTC-2015-4061-7.

#### References

- [1] J. Cooley y J. W. Tukey, «An Algorithm for the Machine Calculation of Complex Fourier Series,» *Mathematics of Computation*, vol. 19, nº 90, p. 297–301, 1965.
- [2] L. P. Yaroslavsky, «Fast Transforms in Image Processing: Compression, Restoration, and Resampling,» *Advances in Electrical Engineering*, vol. 2014, pp. 1-23, 2014.
- [3] "USC-SIPI Image Database," [Online]. Available: <http://sipi.usc.edu/database/database.php?volume=misc>. [Accessed june 2014].
- [4] "Kodak Lossless True Color Image Suite," [Online]. Available: <http://r0k.us/graphics/kodak/>. [Accessed June 2014].
- [5] M. J. Nadenau, S. Winkler, D. Alleysson y M. Kunt, «Human Vision Models for Perceptually Optimized Image Processing - A review,» *Proceedings of the IEEE*, 2000.
- [6] K. H. Norwiche, "The empirical law of sensation and perception," in *Information, sensation and perception*, Toronto, Internet by Biopsychology.org, 2003, pp. 15-33.
- [7] S. Fernandez and M. Martin, "Tema 1. Luz y sistema visual humano," in *Master de Física de los sistemas de diagnóstico, tratamiento y protección en ciencias de la salud*, Valladolid, Escuela Técnica Superior de Ingenieros de Telecomunicación de Valladolid, 2010, pp. 1-12.
- [8] G. A. Gestcheider, "Chapter 1. Psychophysical measurement of thresholds:differential sensitivity," in *Psychophysics: the fundamentals*, New Jersey, Lawrence Erlbaum Associates Inc, 1997, pp. 1-15.
- [9] K. H. Norwiche, "Chapter 12. Differential Thresholds, weber fractions and JND's," in *Information, Sensation and Perception*, Ontario, Internet by Biopsychology.org, 2003, pp.

130-142.

- [10] S. Olavarria, "Tema 1. Elementos de percepción visual," in *Apuntes de TV digital*, Valparaíso, Departamento de Electrónica de la Universidad Técnica Federico Santa María, 2015, pp. 1-7.
- [11] H. Davson, "Section 7. some general aspects of vision," in *Physiology of the eye*, London, Macmillan Academic and Professional Ltd, 1990, pp. 236-240.
- [12] E. H. Adelson, "Lightness Perception and Lightness Illusions," in *New Cognitive Neurosciences*, Cambridge, MIT press, 2000, pp. 339-351.
- [13] E. Kreit, L. M. Mäthger, R. T. Hanlon, P. B. Dennis, R. R. Naik, E. Forsythe and J. Heikenfeld, "Biological versus electronic adaptive coloration: how can one inform the other?," *Journal of the Royal Society Interface*, pp. 1-14, 2012.
- [14] B. HB, "Chapter 16. The physical limits of visual discrimination," in *Photophysiology: action of light on animals and microorganisms*, London, Academic Press Inc, 1964, pp. 163-201.
- [15] M. A. Joshi, "Digital Image Processing," in *Digital Image Processing: An algorithm approach*, new Delhi, Prentice-Hall, 2006, pp. 1-37.
- [16] J. J. García Aranda, M. González Casquete, M. Cao Cueto, J. Navarro Salmerón and F. González Vidal, "Logarithmical hopping encoding: a low computational complexity algorithm for image compression," *IET Image Processing*, vol. 9, no. 8, p. 643 –651, 2015.
- [17] GitHub, "Kanzi: Java & Go code for manipulation and compression of data and images," [Online]. Available: <https://github.com/flanglet/kanzi>. [Accessed March 2015].
- [18] "libjpeg-turbo," 28 july 2015. [Online]. Available: <http://www.libjpeg-turbo.org/About/Performance>. [Accessed 4 August 2015].

# Bibliografía

- [1] S. Wolfram, *A new kind of science*, Champaign, IL: Wolfram Media, 2002.
- [2] M. Stamp, «Once Upon a Time-Memory Tradeoff,» Department of Computer Science of San Jose State University, San Jose, 2006.
- [3] L. Vadillo, «Las infraestructuras de telecomunicaciones en el futuro de los videojuegos,» *bit*, vol. 1, nº 190, p. 39.42, 2012.
- [4] Z.-Y. Wen and H.-F. Hsiao, "QoE-driven performance analysis of cloud gaming services," in *IEEE 16th International Workshop on Multimedia Signal Processing*, Jakarta, 2014.
- [5] Z. Li, R. Grosu, P. Sehgal, S. A. Smolka, S. D. r. Stolle y E. Zadok, «On the Energy Consumption and Performance of Systems Software,» de *SYSTOR '11 Proceedings of the 4th Annual International Conference on Systems and Storage*, New York, 2011.
- [6] J. Cooley y J. W. Tukey, «An Algorithm for the Machine Calculation of Complex Fourier Series,» *Mathematics of Computation*, vol. 19, nº 90, p. 297–301, 1965.
- [7] L. P. Yaroslavsky, «Fast Transforms in Image Processing: Compression, Restoration, and Resampling,» *Advances in Electrical Engineering*, vol. 2014, pp. 1-23, 2014.
- [8] J. J. Garcia Aranda, *Telecommunicaciones superfáciles*, Madrid: ALcatel-lucent, 2010.
- [9] C. E. Shannon, «A mathematical theory of communication,» *Bell System Technical Journal*, 1948.
- [10] R. Penrose, *Cycles of Time*, Bodley Head, 2010.
- [11] D. May, «The return of innovation,» December 2005. [En línea]. Available: <http://www.cs.bris.ac.uk/~dave/iee.pdf>. [Último acceso: May 2015].
- [12] S. W. Golomb, «Run-length encodings,» *IEEE Trans Info Theory*, vol. 12, nº 3, pp. 399-401, 1966.
- [13] M. J. P. W. Rabbani, «Ch 9. Lossy predictive coding,» de *Digital Image Compression Techniques*, SPIE Press,, 1991, pp. 87-101.
- [14] R. Tyagi, *Image compression using DPCM with LMS algorithm*, International Society of Thesis Publication, 2012.

- [15] D. A. Huffman, «A Method for the Construction of Minimum-Redundancy Codes,» *Proceedings of the I.R.E*, 1952.
- [16] A. Desoki, «Compression of text and binary files using adaptive Huffman coding techniques,» de *Southeastcon '88., IEEE Conference Proceedings*, Knoxville, TN, 1988.
- [17] D. E. Knuth, «Dynamic Huffman Coding,» *Journal of Algorithms*, vol. 6, nº 2, pp. 163-180, 1985.
- [18] J. S. Vitter, «Design and analysis of dynamic Huffman codes,» *Journal of the ACM*, vol. 34, nº 4, pp. 825-845, 1987.
- [19] A. Said, *Introduction to Arithmetic Coding - Theory and Practice*, Palo Alto: Academic Press, 2004.
- [20] M. Villar Senín y D. Fernández López, «Departamento de Tecnologías de información y comunicaciones,» 2005. [En línea]. Available: <http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/Imagenyvideo/compresion/3.1.2.2.htm>. [Último acceso: march 2015].
- [21] D. Patel, V. Bhogan y A. Janson, «Simulation and Comparison of Various Lossless Data Compression Techniques based on Compression Ratio and Processing Delay,» *International Journal of Computer Applications*, vol. 81, nº 14, pp. 31-35, 2013.
- [22] CompuServe, «Graphics Interchange Format (GIF) Specification,» 1987. [En línea]. Available: <ftp://ftp.ncsa.uiuc.edu/misc/file.formats/graphics.formats/gif87a.doc>. [Último acceso: may 2015].
- [23] W3C, «Portable Network Graphics (PNG) Specification (Second Edition),» 10 November 2003. [En línea]. Available: <http://www.w3.org/TR/2003/REC-PNG-20031110>. [Último acceso: may 2015].
- [24] A. W. Paeth, «Image File Compression Made Easy,» de *Graphics Gems II*, San Diego, Academic Press, 1991.
- [25] W. B. Pennebaker y J. L. Mitchell, *Still Image Data Compression Standard*, Springer US, 1993.
- [26] J. committee, «JPEG,» [En línea]. Available: <http://www.jpeg.org/jpeg/index.html>. [Último acceso: January 2015].
- [27] S. W. Smith, «JPEG (Transform Compression),» de *The Scientist and Engineer's Guide to Digital Signal Processing*, San Diego, California Technical Publishing, 1998, pp. 481-502.
- [28] A. Mammeri, B. Hadjou y A. Khoumsi, «A Survey of Image Compression Algorithms for Visual Sensor Networks,» *ISRN Sensor Networks*, vol. 2012, nº Article ID 760320, 2012.

- [29] D. Austin, «What is JPEG?», *Notices of the AMS*, vol. 55, nº 2, pp. 226-229, 2008.
- [30] W. Pratt, *Digital Image Processing*, Jhon Wiley & Sons, 1978.
- [31] R. Gonzalez y R. Woods, *Tratamiento Digital de Imágenes*, Addison-Wesley, 1996.
- [32] H. Aalen, «PackJPEG», 2013. [En línea]. Available: <http://www.elektronik.htw-aalen.de/packjpg/index.htm>. [Último acceso: may 2015].
- [33] «jpg/jpeg lossless image compression test», MaximumCompression (lossless data compression software benchmarks), 2011. [En línea]. Available: <http://www.maximumcompression.com/data/jpg.php>. [Último acceso: may 2015].
- [34] M. Stirner y G. Seelmann, «Improved redundancy reduction for JPEG files», de *Picture coding symposium*, 2007.
- [35] M. Shensa, «The discrete wavelet transform: wedding the a trous and Mallat algorithms», *IEEE Transactions on Signal Processing*, vol. 40, nº 10, pp. 2464 - 2482, 2002.
- [36] S. B. Z.-E. S-Medouakh, «Entropy Encoding EBCOT (Embedded Block Coding with Optimized Truncation) In JPEG2000», *IJCSI International Journal of Computer Science*, vol. 4, nº 1, pp. 531-536, 2011.
- [37] S. Medouakh y Z.-E. Baarir, «Study of the Standard JPEG2000 in Image Compression», *International Journal of Computer Applications*, vol. 18, nº 1, pp. 27-33, 2011.
- [38] Y. Sun y J. Xin, «Efets of JPEG XR Compression settings On iris recognition system», de *Computer Analisys of Images and Patterns*, Sevilla, 2011.
- [39] J. Bankoski, P. Wilkins y Y. Xu, «Technical overview of VP8, an open source video codec for the web», 2012. [En línea]. Available: <http://static.googleusercontent.com/media/research.google.com/es//pubs/archive/37073.pdf>. [Último acceso: may 2015].
- [40] «WEBP, nuevo formato de imagen para la web», november 2010. [En línea]. Available: <http://mosaic.uoc.edu/2010/11/15/webp-nuevo-formato-de-imagen-orientado-a-la-web/>. [Último acceso: may 2015].
- [41] «Lossy WebP», Google, April 2015. [En línea]. Available: <https://developers.google.com/speed/webp/docs/compression>. [Último acceso: may 2015].
- [42] S. Perez-Becker, «Compresión fractal de imágenes», 2012. [En línea]. Available: [http://www.redmat.unam.mx/foro/volumenes/vol029/Compresion\\_Fractal.pdf](http://www.redmat.unam.mx/foro/volumenes/vol029/Compresion_Fractal.pdf). [Último acceso: may 2015].

- [43] J. D. Gomez V., «Compresión de imágenes digitales utilizando redes neuronales multicapa (backpropagation) y RBR's,» *Scientia et Technica*, vol. 11, nº 29, pp. 101-106, 2005.
- [44] A. Domingo Ajenjo, Tratamiento digital de imágenes, Madrid: Anaya multimedia, 1994.
- [45] J. Kopf y D. Lischinski, «Pixel Art multicomparison,» 2011. [En línea]. Available: <http://research.microsoft.com/en-us/um/people/kopf/pixelart/supplementary/index.html>. [Último acceso: 2015].
- [46] «Pixel Scalers,» datagenetics, 2013. [En línea]. Available: <http://www.datagenetics.com/blog/december32013/index.html>. [Último acceso: may 2015].
- [47] A. M. A. B. A. Mittal, «No-reference image quality assessment in the spatial domain,» *IEEE Transactions Image Processing*, vol. 21, nº 12, p. 4695-4708, 2012.
- [48] Z. Wang y A. C. Bovik, «A universal image quality index,» *IEEE signal processing letters*, vol. 9, nº 3, pp. 81 - 84, 2002.
- [49] GitHub, «Kanzi: Java & Go code for manipulation and compression of data and images,» [En línea]. Available: <https://github.com/flanglet/kanzi>. [Último acceso: March 2015].
- [50] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Thomas Stockhammer y W. Thomas, «Video coding with H.264/AVC: Tools, Performance, and Complexity,» *IEEE CIRCUITS AND SYSTEMS MAGAZINE*, pp. 7-28, 2004.
- [51] M. T. Pourazad, C. Doutre, M. Azimi y P. Nasiopoulos, «HEVC: The New Gold Standard for Video Compression,» *IEEE CONSUMER ELECTRONICS MAGAZINE*, vol. 1, nº 3, pp. 36-46, 2012.
- [52] F. Bossen, B. Bross y D. Flynn, «HEVC Complexity and Implementation Analysis,» *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, vol. 22, nº 12, pp. 1685-1696, 2012.
- [53] k. H. Norwich, «The empirical law of sensation and perception,» de *Information, sensation and perception*, Toronto, Internet by Biopsychology.org, 2003, pp. 15-33.
- [54] S. Fernandez y M. Martin, «Tema 1. Luz y sistema visual humano,» de *Master de Física de los sistemas de diagnóstico, tratamiento y protección en ciencias de la salud*, Valladolid, Escuela Técnica Superior de Ingenieros de Telecomunicación de Valladolid, 2010, pp. 1-12.
- [55] K. H. Norwich, «Chapter 12. Differential Thresholds, weber fractions and JND's,» de *Information, Sensation and Perception*, Ontario, Internet by Biopsychology.org, 2003, pp. 130-142.
- [56] H. Davson, «Section 7. some general aspects of vision,» de *Physiology of the eye*, London, Macmillan Academic and Professional Ltd, 1990, pp. 236-240.

- [57] G. A. Gestcheider, «Chapter 1. Psycophysical measurement of thresholds:differential sensivity,» de *Psycophysics: the fundamentals*, New Jersey, Lawrence Erlbaum Associates Inc, 1997, pp. 1-15.
- [58] S. Olavarria, «Tema 1. Elementos de percepción visual,» de *Apuntes de TV digital*, Valparaiso, Departamento de Electrónica de la Universidad Técnica Federico Santa María, 2015, pp. 1-7.
- [59] B. HB, «Chapter 16. The physical limits of visual discrimination,» de *Photophysiology: action of light on animals and microorganisms*, London, Academic Press Inc, 1964, pp. 163-201.
- [60] E. H. Adelson, «Lightness Perception and Lightness Illusions,» de *New Cognitive Neurosciences*, Cambridge, MIT press, 2000, pp. 339-351.
- [61] «Kodak Lossless True Color Image Suite,» [En línea]. Available: <http://r0k.us/graphics/kodak/>. [Último acceso: June 2014].
- [62] M. Podpora, G. Paweł Korbas y A. Kawala-Janik, «YUV vs RGB – Choosing a Color Space for Human-Machine Interaction,» de *Federated Conference on Computer Science and Information Systems*, Warsaw, 2014.
- [63] «USC-SIPI Image Database,» [En línea]. Available: <http://sipi.usc.edu/database/database.php?volume=misc>. [Último acceso: june 2014].
- [64] J. J. Garcia Aranda, M. Gonzalez, M. Cao, J. Navarro y F. Gonzalez, «Logarithmical hopping encoding: a low computational complexity algorithm for image compression,» *IET Image Processing*, vol. 9, nº 8, p. 643 –651, 2015.
- [65] J. Kopf y D. Lischinski, «Depixelizing Pixel Art,» de *Proceedings of SIGGRAPH*, Article no. 99, 2011.
- [66] B. Goldstein E., *Sensación y percepción*, Mexico: International Thomson Editores, 1999.
- [67] K. H. Norwich, «Perception as a Choice among Alternatives,» de *Information, Sensation and Perception*, Ontario, Biopsychology.org, 2003, pp. 8-14.
- [68] A. Brigas Hidalgo, *Psicología. Una ciencia con sentido humano*, Mexico: Esfinge, 2010.
- [69] G. Humphrey, «The psychology of the gestalt,» *Journal of Educational Psychology*, vol. 15, nº 7, p. 401–412, 1924.
- [70] D. Todorovic, «Gestalt principles,» *Scholarpedia*, vol. 3, nº 12, p. 5345, 2008.
- [71] J. González Labra, *Introducción a la psicología del pensamiento*, Madrid: Trotta, 2009.

- [72] J. Covacevic y M. Vetterli, «The commutativity of up/downsampling in two dimensions,» *IEEE transactions on Information Theory*, vol. 37 , nº 3, pp. 695-698, 1991.
- [73] J. Kiess, «University of Mannheim,» [En línea]. Available: <http://ls.wim.uni-mannheim.de/de/pi4/research/projects/retargeting/test-sequences/>. [Último acceso: April 2015].
- [74] «ffmpeg,» [En línea]. Available: [www.ffmpeg.org](http://www.ffmpeg.org).

# Acrónimos

ADPCM	Adaptive Differential Pulse Code Modulation
BMP	Bit Map
BPP	bits per pixel
CF	compression Factor
CIE	International Commission on Illumination (abreviado CIE por su nombre en frances: Commission internationale de l'éclairage)
CMYK	Cian, Magenta, Yellow , blacK
DCT	Discrete Cosine Transform
DM	Delta Modulation
DPCM	Differential Pulse Code Modulation
EBCOT	Embedded Block Coding with Optimized Truncation
EPX	Eric Pixel eXpander
FIF	Fractal Image Format
GIF	Graphics Interchange Format
GOP	Group of pictures
HD	High Definition
HSB	Hue, Saturation, Brightness
HSL	Hue, saturation, luminance
HSV	Hue, Saturation, Value
HSV	Human Vision System
ISO	International Organization for Standardization
ITU	International Telecommunication Union
JPEG	Joint Photographic Experts Group
LHE	Logarithmical Hopping Encoding
LZ	Lempel Ziv
LZW	Lempel Ziv Welch
MNG	Multiple Image Network Graphics
MOS	Mean Opinion Score
MPEG	The Moving Picture Experts Group

MSE	Mean Squared Error
NN	Nearest Neighbour
PCM	Pulse Code Modulation
PNG	PNG is Not GIF
PPP	pixels per pixel
PR	Perceptual Relevance
PSNR	Peak Signal to Noise Ratio
QL	Quality level
QoE	Quality of Experience
RD	Rate-Distortion
RGB	Red, Green, Blue
RLC	Run Length Coding
RLE	Run Length Encoding
RNA	Red Neuronal Artificial
SPS	Simple Pixel Selection
SSIM	Structured similarity Index Metric
TIFF	Tagged Image File Format
WMA	Windows Media Audio
WMV	Windows Media Video