



Departamento de Electrónica y Electromagnetismo

Diseño de Circuitos para Tratamiento de Imágenes
Aplicando Técnicas Basadas en Soft Computing

Tesis Doctoral

Nashaat Mohamed Hussein Hassan

Tutor:

Ángel Barriga Barros

Sevilla de 2009



Departamento de Electrónica y Electromagnetismo

Diseño de Circuitos para Tratamiento de Imágenes

Aplicando Técnicas Basadas en Soft Computing

Memoria presentada por

Nashaat Mohamed Hussein Hassan

Para obtener el grado de Doctor

Director:

Dr. Ángel Barriga Barros

Catedrático de Universidad

Dpto. Electrónica y Electromagnetismo

Departamento de Electrónica y Electromagnetismo

Universidad de Sevilla

Agradecimientos

En el desarrollo de este trabajo quiero agradecer la colaboración y ayuda de todas las personas que han contribuido ya sea a través de la discusión o bien proporcionando las facilidades que me han permitido llevarlo a cabo. En primer lugar quiero agradecer a mi profesor, mi tutor y amigo Ángel Barriga Barros. Inicialmente Ángel me dio la oportunidad de empezar esta historia y después ha sido entusiasta y me ha apoyado durante todos estos años. Debo dar un agradecimiento a Iluminada y Santiago que también me han estado apoyando durante todo el tiempo. Agradecimientos especiales a Federico, Gashaw y Piedad. Debo dar un especial agradecimiento a la familia que conforma el Instituto de Microelectrónica de Sevilla del CSIC y al Departamento de Electrónica y Electromagnetismo de la Universidad de Sevilla. En especial a la Unidad Técnica del IMSE y al personal de Secretaría y Administración.

También quiero agradecer el apoyo de varias instituciones con cuya financiación se ha posibilitado este trabajo. La investigación presentada aquí ha sido soportada en parte por una beca de doctorado de la Agencia Española de Cooperación Internacional (AECI), el proyecto TEC2005-04359/MIC del Ministerio Española de Educación y Ciencia y el Gobierno Regional de Andalucía en virtud de conceder el proyecto de excelencia TIC2006-635.

Finalmente, quiero agradecer el apoyo de mi familia, mi esposa Marwa, mi madre, mi hermano Refaei, mi hermano Refat, mi hermano Hussein, mi hermana Amal y a toda

la familia. También quiero agradecer el apoyo de la familia de mi esposa, su madre Nadia, su padre Madany, sus hermanos Ahmad, Mohamed, Omar y Zeyad y a sus hermanas Walaa y Esraa. Y a la familia de mi profesor Margarita y a sus hijos Carlos y Margarita.

Prefacio

La Tesis que se presenta se enmarca en el campo de aplicación del desarrollo de sistemas para el procesado de imágenes. Dentro de esta área se pretende dar soluciones a algunos de los problemas que aparecen a la hora de realizar el tratamiento de bajo nivel de imágenes en sistemas que presentan restricciones tanto de coste como de velocidad de operación. La Tesis pretende abordar cuatro aspectos relacionados con el procesado de imágenes: la compresión de imágenes, la mejora del contraste, la segmentación y la detección de bordes. El desarrollo de los algoritmos de tratamiento de imágenes se afronta desde una perspectiva específica mediante técnicas basadas en *soft computing*. Esta perspectiva permitirá desarrollar estrategias que cumplan con los requisitos impuestos y den lugar a circuitos eficientes.

El primero de los aspectos que se abordan en esta Tesis corresponde a la compresión de imágenes. La necesidad de realizar la compresión proviene de la limitación del ancho de banda en los medios de comunicación así como la necesidad de reducir el espacio de almacenamiento. Las técnicas de compresión de imágenes permiten eliminar la redundancia en la imagen con objeto de reducir la información que es necesario almacenar o transmitir. A la hora de considerar los algoritmos de compresión se considerarán tanto el caso de compresión sin pérdidas y el caso de compresión con pérdidas.

El segundo aspecto que se trata en esta Tesis es el control del contraste. Básicamente el contraste de imágenes puede ser definido como el cambio de la relación de luminancia de los elementos de una imagen. Cuando la variación en la luminancia es baja entonces la imagen tiene poco contraste. El control del contraste de imágenes es una operación necesaria en determinadas aplicaciones de procesamiento de imágenes. Así, por ejemplo, la mejora del contraste permite distinguir objetos en la imagen que no son distinguibles cuando se produce pérdida de contraste.

El tercer y el cuarto aspecto considerados en esta Tesis corresponden a la segmentación de imágenes y la detección de bordes. Los algoritmos de segmentación y de detección de bordes permiten extraer información de las imágenes y reducir los requerimientos necesarios para el almacenamiento de la información. Por otro lado los mecanismos de extracción de bordes se implementan mediante la ejecución de la correspondiente realización software sobre un procesador. Sin embargo en aplicaciones que demanden restricciones en los tiempos de respuesta (aplicaciones en tiempo real) se requiere de implementaciones específicas en hardware. El principal inconveniente de las técnicas de detección de bordes para su realización hardware es la alta complejidad de los algoritmos existentes. Por este motivo se afronta el desarrollo de una técnica que ofrezca resultados adecuados para la detección de bordes en imágenes y simultáneamente permite realizar implementaciones hardware de bajo coste y alta velocidad de procesado.

Esta Tesis se ha organizado en cinco capítulos. El primer capítulo cubre definiciones y conceptos básicos de imágenes digitales. El segundo capítulo trata de la compresión de imágenes. Se describen algunas mostrando las técnicas de compresión y se proponen e implementan nuevas estrategias. El capítulo 3 se centra en el control del contraste. El cuarto capítulo trata la segmentación de imágenes. En este caso nos centramos en la segmentación binaria basada en aplicar un valor umbral. Finalmente en el capítulo quinto se considera el problema de la detección de bordes

Índice

Lista de tablas.....	xiii
Lista de figuras.....	xv
Capítulo 1. Conceptos básicos.....	1
1.1. Representación digital de imágenes.....	2
1.2. Fundamentos y transformaciones de colores.....	7
1.2.1. Imágenes monocromáticas.....	7
1.2.2. Imágenes en color.....	9
1.3. Formatos de almacenamiento de imágenes digitales...15	
1.3.1. Formato BMP.....	15
1.3.2. Formato GIF.....	17
1.3.3. Formato PNG.....	18
1.3.4. Formato JPEG.....	19
1.3.5. Formato TIFF.....	20
1.4. Histograma de la imagen.....	21
1.5. Aplicación del Soft Computing en el procesado de	
Imágenes.....	24
1.6. Resumen.....	25
Capítulo 2. Compresión de imágenes.....	27

2.1. Medida de calidad en compresión de imágenes.....	29
2.2. Run-Length Encoding (RLE).....	31
2.3. Codificación estadística: Codificación de Huffman..	32
2.4. Compresión basada en transformada: JPEG.....	36
2.5. Muestreo uniforme.....	41
2.6. Algoritmos de eliminación de redundancia.....	43
2.6.1. Aproximación por programación dinámica.....	45
2.6.2. Algoritmo geométrico.....	46
2.7. Algoritmos propuestos.....	49
2.7.1. Modificación del algoritmo geométrico.....	49
2.7.2. Particionado del histograma y codificación.....	57
2.8. Diseño e implementación hardware de los algoritmos de compresión propuestos.....	66
2.8.1. Diseño del circuito de compresión de los algoritmos Geométricos.....	67
2.8.2. Diseño del circuito de descompresión de los algoritmos Geométricos.....	72
2.8.3. Circuito de compresión/descompresión de los algoritmos Geométricos.....	76
2.8.4. Diseño del circuito de compresión del algoritmo de particionado del histograma y codificación.....	77
2.9. Resumen.....	80
Capítulo 3. Control del contraste en imágenes.....	83
3.1. Técnicas de control del contraste en imágenes.....	85
3.2. Álgebra de Łukasiewicz.....	93
3.3. Control del contraste mediante operadores de Łukasiewicz.....	95
3.4. Diseño de los operadores básicos de Łukasiewicz... 	109
3.4.1. Realizaciones basadas en redes neuronales.....	109
3.4.2. Realizaciones basadas en lógica combinatorial.....	112

3.4.3. Resultados de implementación.....	114
3.5. Control de contraste aplicando lógica difusa.....	116
3.6. Aplicación de la lógica de Lukasiewicz a la	
aproximación de funciones lineales a tramos.....	128
3.6.1. Funciones de una variable.....	129
3.6.2. Funciones de más variables.....	134
3.6.3. Comparación entre técnicas de realización.....	135
3.7. Resumen.....	137
Capítulo 4. Segmentación de imágenes.....	139
4.1. Segmentación de imágenes.....	140
4.1.1. Segmentación por discontinuidades.....	140
4.2.1. Segmentación por similitud.....	141
4.2. Métodos de umbralizacion.....	143
4.2.1. Método basado en la frecuencia de nivel de gris.....	145
4.2.2. Método de Otsu.....	146
4.2.3. Métodos basados en lógica difusa.....	149
4.3. Cálculo del umbral mediante lógica difusa.....	150
4.4. Diseño del circuito para el cálculo del umbral.....	154
4.5. Resumen.....	164
Capítulo 5. Detección de bordes.....	165
5.1. Algoritmos para la detección de bordes en imágenes	
.....	166
5.1.1 Métodos basados en gradiente.....	166
5.1.2 Métodos basados en Laplaciana.....	171
5.2. Descripción de la técnica de detección de bordes	
Propuesta.....	172
5.2.1 Etapa de filtrado.....	173
5.2.2 Etapa de umbralización y detección de bordes.....	176
5.3. Análisis de calidad.....	178

5.4. Diseño del sistema de detección de bordes.....	180
5.5. Resumen.....	187
Conclusiones.....	189
Apéndice A.....	193
Apéndice B.....	197
Apéndice C.....	201
Apéndice D.....	219
Bibliografía.....	223

Lista de tablas

1.1. Ejemplos de generación de colores RGB.....	9
1.2. Profundidad de color en el formato PNG.....	19
1.3. Ejemplo de distribución de los niveles de gris de una imagen de 64x64 y 8 niveles de intensidad.....	22
2.1. Configuraciones del circuito de compresión de imágenes.....	67
2.2. Coste de las implementaciones en términos de <i>slices</i> ocupados.....	77
2.3. Tiempo de compresión y descompresión de imágenes (en μ seg) para una frecuencia de 100 MHz.....	79
3.1. Recursos consumidos.....	115
3.2. Retraso máximo.....	115
3.3. Coste en <i>slices</i> de las realizaciones sobre FPGA de las funciones $f_1(x)$ y $f_2(x)$	136
3.4. Retraso máximo (en nseg.) en las realizaciones sobre FPGA de las Funciones $f_1(x)$ y $f_2(x)$	136
4.1. Umbrales obtenidos al aplicar los métodos de Otsu, frecuencia de gris y la técnica propuesta.....	162
5.1. Píxeles activos que toman el valor de los bordes y el porcentaje respecto a la imagen total.....	179
5.2. Porcentaje de píxeles que coinciden con el obtenido por Canny.....	180

Lista de figuras

1.1. Formación de una imagen [EDWA99].....	3
1.2. Generación de una imagen digital, a) la imagen continua, b) una línea de barrido de A a B en la imagen continua, c) muestreo, d) cuantización.....	4
1.3. Efecto de reducir la resolución espacial $N \times M$	6
1.4. Efecto de variar el número de niveles de intensidad.....	7
1.5. Operaciones de combinación de imágenes binarias (a y b). (c) $A \vee B$; (d) $A \wedge B$; (e) $A \oplus B$; (f) $\neg A$	8
1.6. Representación de imágenes en color RGB.....	10
1.7. Transformación RGB a HSL.....	11
1.8. Representación tridimensional de los espacios básicos de color de un dispositivo aditivo (RGB) o sustractivo (CMY).....	14
1.9. Estructura de archivos de mapa de bits.....	16
1.10. Imagen BMP con paleta de 4 bits (16 colores).....	17
1.11. Imagen comprimida JPEG a) máxima calidad b) mínima calidad.....	20
1.12. Histograma del ejemplo de la imagen descrita en la tabla 1.3.....	22
1.13. Ejemplos de imágenes con sus histogramas.....	23

2.1. Etapas de un sistema de compresión de imágenes.....	29
2.2. Etapas del proceso de a) compresión JPEG, b) descompresión.....	38
2.3. (a) Muestreo uniforme (k=2), (b) descompresión para k=3.....	42
2.4. Ejemplo de compresión y descompresión aplicando muestreo uniforme: (a) imagen “soccer” original de 64x64 pixels, (b) imagen comprimida con razón de muestreo k=2, (c) imagen descomprimida.....	43
2.5. Error de túnel de la función F con error ε	44
2.6. Aplicación de programación dinámica.....	45
2.7. Esquema de compresión.....	46
2.8. Pseudocódigo del algoritmo de compresión.....	47
2.9. Pseudocódigo del algoritmo de descompresión.....	47
2.10. Cálculo de tangentes en el algoritmo geométrico.....	48
2.11. Ejemplo del algoritmo propuesto.....	50
2.12. a) Imagen “soccer” de 64x64, b) imagen “Lena” de 128x128, c) imagen “Cameraman” de 128x 128.....	52
2.13. Comparación de algoritmos: razón de compresión.....	53
2.14. Comparación de algoritmos: RMSE.....	55
2.15. Comparación de algoritmos para un RMSE prefijado.....	56
2.16. Funciones de pertenencia distribuidas en el universo de discurso de la variable que representa el color de un píxel.....	57
2.17. Representación exacta del universo de discurso.....	58
2.18. Ejemplo de codificación de la imagen.....	59
2.19. Esquema de codificación.....	60
2.20. Resultados de la razón de compresión.....	61
2.21. Resultados del valor de RMSE.....	62
2.22. Comparación de los algoritmos de compresión en función de la razón de compresión.....	64
2.23. Comparación de los algoritmos de compresión en función del RMSE...	65
2.24 Circuito básico común de los algoritmos geométricos de compresión...	68
2.25. Circuito del algoritmo geométrico de compresión AG1.....	69
2.26. Carta ASM del algoritmo geométrico de compresión AG1.....	70
2.27. Circuito del algoritmo geométrico de compresión AG2.....	70
2.28. Circuito del algoritmo geométrico de compresión AG3.....	71

2.29. Esquemático del circuito de compresión.....	72
2.30. Símbolo del circuito de descompresión del algoritmo AG2.....	73
2.31. Descripción algorítmica VHDL de la FSM del algoritmo de descompresión AG3.....	74
2.32. Esquemático del circuito de descompresión AG3.....	75
2.33. Esquema de bloques del circuito de descompresión.....	76
2.34. Símbolo del circuito de cuantización.....	76
2.35. Esquema del sistema de compresión/descompresión.....	78
2.36. Esquemático del circuito de compresión.....	78
2.37. Algoritmo de compresión.....	79
3.1. Transformación lineal.....	87
3.2. Transformación de tramos lineales.....	88
3.3. Diagrama de bloques del circuito de control del contraste de [CHO00].	89
3.4. CDF y su aproximación lineal a tramos (a) de una imagen oscura, (b) de una imagen clara.....	90
3.5. Diagrama de bloques del método de [KIM99].....	90
3.6 Transformación gaussiana.....	92
3.7. Representación gráfica de los operadores de Łukasiewicz.....	94
3.8. a) Imagen original y su histograma, b) imagen resultante de aplicar la suma acotada y su histograma.....	96
3.9. a) Imagen original (356x292), b) $x \oplus y$, c) $x \oplus y \oplus 20$, d) $x \oplus y \oplus 40$..	97
3.10. a) Imagen Mesi (120x89), b) $x \oplus y$, c) $x \oplus y \oplus 20$, d) $x \oplus y \oplus 40$	98
3.11. a) Imagen Dibujo8 (119x80), b) $x \oplus y$, c) $x \oplus y \oplus 20$, d) $x \oplus y \oplus 40$...	99
3.12. Imagen con distintos valores de control de contraste para la suma acotada y los histogramas.....	100
3.13. a) Imagen original y su histograma, b) imagen resultante de aplicar el producto acotado y su histograma.....	100
3.14. Imagen con distintos valores de control de contraste para el producto acotado y los histogramas.....	101
3.15. Imagen con distintos valores de control de contraste para el producto acotado y los histogramas.....	102
3.16. Imagen con distintos valores de control de contraste para el producto	

acotado y los histogramas.....	102
3.17. Imagen con distintos valores de control de contraste para el producto acotado y los histogramas.....	103
3.18. a) Imagen original; suma acotada para mascara de b) 1x2 píxeles, c) 2x2 píxeles y d) 1x2+40 píxeles, e) 2x2+40 píxeles.....	104
3.19. Variación del contraste aplicando la suma acotada a máscaras de 2 píxeles y de 2x2 píxeles con diferentes valores del control C (0, 30 y -30).	105
3.20. El sistema difuso del sistema de toma decisiones en el control de contraste.....	106
3.21. Resultados de aplicar control de contraste: a) imagen original, b) sistema basado en la figura 3.20, c) sistema que incluye de regla (1), d) sistema que incluye la regla (2).....	108
3.22. a) Entidad de una neurona. b) Función de activación de la neurona...	109
3.23. Esquemático del circuito de una neurona de dos entradas.....	110
3.24. Operadores $\min(x,y)$ y $\max(x,y)$ realizados mediante redes neuronales.	110
3.25. Superficies correspondientes a los operadores (a) $\min(x,y)$ y (b) $\max(x,y)$	111
3.26. a) Circuito máximo, b) circuito mínimo.....	111
3.27. Esquema de bloques del operador mínimo.....	113
3.28. Circuito producto acotado.....	113
3.29. Circuito optimizado del producto acotado.....	113
3.30. Circuito suma acotada.....	114
3.31. Sistema para el control de contraste con 3 funciones de pertenencia para los antecedentes, 5 para el consecuente y 9 reglas.....	117
3.32. Sistema para el control de contraste con 5 funciones de pertenencia para los antecedentes, 9 para el consecuente y 25 reglas.....	118
3.33. Superficies correspondientes a la función de control de contraste para el caso de a) 3 funciones de pertenencia en los antecedentes, b) 5 funciones de pertenencia.....	118
3.34. a) Imagen original, b) $x \oplus y$, c) $x \oplus y \oplus f_{3MF}(x,y)$, d) $x \oplus y \oplus f_{5MF}(x,y)$	119

3.35. a) Imagen original, b) $x \oplus y$, c) $x \oplus y \oplus f_{3MF}(x, y)$, d) $x \oplus y \oplus f_{5MF}(x, y)$	120
3.36. a) Imagen original, b) $x \oplus y$, c) $x \oplus y \oplus f_{3MF}(x, y)$, d) $x \oplus y \oplus f_{5MF}(x, y)$	121
3.37. Sistema para el control de contraste con 3 funciones de pertenencia para los antecedentes, 5 para el consecuente y 9 reglas.....	122
3.38. a) Caso de $f(x,y)$ en el rango $[-127,0]$, b) caso de $f(x,y)$ en el rango $[-63,64]$.....	123
3.39. Sistema para el control de contraste con 3 funciones de pertenencia para los antecedentes, 5 para el consecuente y 9 reglas.....	124
3.40. Sistema para el control de contraste con 5 funciones de pertenencia para los antecedentes, 9 para el consecuente y 25 reglas.....	124
3.41. a) Imagen original, b) $x \otimes y$, c) $x \otimes y \otimes f_{3MF}(x, y)$, d) $x \otimes y \otimes f_{5MF}(x, y)$.	125
3.42. a) Caso de $f(x,y)$ en el rango $[-127,0]$, b) caso de $f(x,y)$ en el rango $[-63,64]$.....	126
3.43. Esquema del sistema de control de contraste.....	127
3.44. a) Imagen original, b) $x \oplus y$, c) sistema de control difuso de la figura 3.43.....	128
3.45. Ejemplo de función lineal a tramos.....	130
3.46. Descripción funcional VHDL de la función $f_I(x)$.....	132
3.47. Función VHDL para el producto acotado.....	132
3.48. Realizaciones de la función $f_I(x)$ basada en (a) red neuronal, (b) operadores de Łukasiewicz.....	133
3.49. Superficie correspondiente a la función $f_2(x)$.....	134
3.50. Realización de $f_2(x)$ mediante (a) una red neuronal, (b) operadores de Łukasiewicz basados en lógica combinatorial.....	135
4.1. Ejemplo del algoritmo de segmentación por regiones.....	142
4.2. Clasificación en 3 grupos de los píxeles de una imagen.....	143
4.3. a) Imagen de Lena, b) imagen binaria con $T=0.5$.....	144
4.4. Histograma de la imagen de Lena.....	146

4.5. Funciones de pertenencia para $N=9$, a) antecedente, b) consecuente...	151
4.6. Base de reglas para $N=9$	152
4.7. Cálculo del umbral en Matlab.....	153
4.8. Arquitectura VHDL de la base de conocimiento.....	155
4.9. Paquete VHDL con las definiciones de los tipos de datos y de las funciones.....	156
4.10. a) Símbolo del sistema de generación del umbral, b) Módulo de Inferencia fuzzy (FIM) y divisor, c) bloque FIM con el motor de Inferencia difuso y circuito acumulador.....	157
4.11. Sistema de test basado en una placa de desarrollo Spartan3-Starter Board.....	158
4.12. Sistema de test del circuito de cálculo del umbral.....	159
4.13. a) Inicio de la operación del sistema con la lectura de la imagen de memoria, b) Generación del resultado final con la división. c) Representación de la salida en los displays 7-segmentos.....	160
4.14. Resultados de implementación a) del circuito de generación del umbral, b) del sistema de test.....	161
4.15. a) Ejemplos de imágenes b) método de Otsu, c) frecuencia de nivel de gris, d) técnica propuesta.....	163
5.1. Aplicación del método de Canny a la imagen de Lena.....	169
5.2. Aplicación del método de Roberts a la imagen de Lena.....	169
5.3. Aplicación del método de Sobel a la imagen de Lena.....	170
5.4. Aplicación del método de Prewitt a la imagen de Lena.....	171
5.5. Aplicación del método <i>zero-cross</i> a la imagen de Lena.....	172
5.6. Diagrama de flujo para la detección de bordes.....	172
5.7. a) Imagen con ruido <i>salt&peppers</i> , filtrado b) máximo, c) mínimo.....	173
5.8. Aplicación del filtrado de Kuwahara a un bloque de 3x3 píxeles.....	175
5.9. a) Imagen con ruido <i>salt&peppers</i> , b) salida del filtro Kuwahara.....	175
5.10. a) Imagen de entrada con ruido del tipo <i>salt&peppers</i> , b) salida del filtro basado en la suma acotada de Lukasiewicz.....	176
5.11. Máscara 3x3 para la detección de bordes.....	177
5.12. (a) Imagen binaria de Lena, (b) detección de bordes.....	177
5.13. Orientaciones para la generación de los bordes.....	178
5.14. Diagrama de boques del sistema de detección de bordes.....	181

5.15. Pseudocódigo del algoritmo de detección de bordes.....	181
5.16. Esquema para el procesado de un píxel.....	181
5.17. Diagrama de bloques de la arquitectura 8x3.....	183
5.18. Esquema de una unidad funcional (FU).....	184
5.19. a) Filtro Lukasiewicz, b) lógica de umbralización, c) circuito de detección de bordes.....	184
5.20. FSM de la unidad de control del sistema.....	185
5.21. Cronograma de la operación del circuito.....	186
5.22. Resultados de implementación sobre FPGA.....	186

Capítulo 1

Conceptos básicos

Las imágenes desempeñan un importante papel en nuestra sociedad como un medio de comunicación. La mayoría de los medios de comunicación (por ejemplo, periódicos, televisión, cine, etc) utilizan las imágenes (fijas o en movimiento) como portadores de información. El enorme volumen de información óptica y la necesidad de su tratamiento y transmisión han dado lugar a la necesidad de sistemas específicos de procesamiento de imágenes. El problema de la transmisión de imágenes constituye uno de los campos más fecundos en cuanto a aportes de resultados y propuestas [WILL79] [JOSE04]. Precisamente una de las aplicaciones iniciales de ésta categoría de técnicas de tratamiento de imágenes consistió en mejorar las fotografías digitalizadas de periódicos enviadas por cable submarino entre Londres y Nueva York. La introducción del sistema Bartlane de transmisión de imágenes por cable a principios de la década de 1920 redujo el tiempo necesario para enviar una fotografía a través del Atlántico de más de una semana a menos de tres horas. Un equipo especializado de impresión codificaba las imágenes para transmisión por cable y luego las reconstruía en el extremo de recepción. Otra de las líneas de interés desde los primeros momentos correspondió a la mejora de la calidad visual de estas primeras imágenes digitales. Este aspecto estaba relacionado con la selección de procedimientos de impresión y la distribución de niveles de brillo.

Los esfuerzos iniciados en torno a 1964 en el laboratorio de Jet Propulsion (Pasadena, California) se refería al procesamiento digital de imágenes de la luna procedentes de satélite. A raíz de estos trabajos surgió una nueva rama de la ciencia denominada procesamiento digital de imágenes. Desde entonces dicha rama del conocimiento ha mostrado un enorme crecimiento y ha generado un importante impacto tecnológico en varias áreas como, por ejemplo, en las telecomunicaciones, la televisión, la medicina y la investigación científica. El procesamiento digital de imágenes se refiere a la transformación de una imagen a un formato digital y su procesamiento por computadoras o sistemas digitales. Tanto la entrada y salida del sistema de procesamiento digital de imágenes son imágenes digitales [JAHN05]. El análisis de la imagen digital está relacionado con la descripción y el reconocimiento de los contenidos digitales [CHEN99]. En este caso la entrada del sistema de procesado es una imagen digital y el resultado que se genera tras el procesado es una descripción simbólica de la imagen. Algunos trabajos de investigación en esta área se pueden consultar en [GONZ87, GONZ02, GONZ07, YOUN98, RUSS95, AZRI82, PITA00].

En este capítulo presentamos una introducción general de algunos conceptos básicos sobre las imágenes digitales que serán utilizados en este trabajo de tesis. Finalmente, vamos a realizar una breve discusión sobre el papel de la aplicación del *Soft Computing* en el procesamiento de imágenes.

1.1. Representación digital de imágenes

Una imagen es una distribución de la energía luminosa como una función de la posición espacial. La figura 1.1 ilustra la formación de una imagen. Una fuente de luz (en este ejemplo corresponde al Sol) emite la energía luminosa que incide en un objeto. La energía luminosa puede sufrir diversas transformaciones al incidir en un objeto, tal como ser absorbida por el objeto, ser transmitida a través del objeto y/o reflejarse en el objeto.

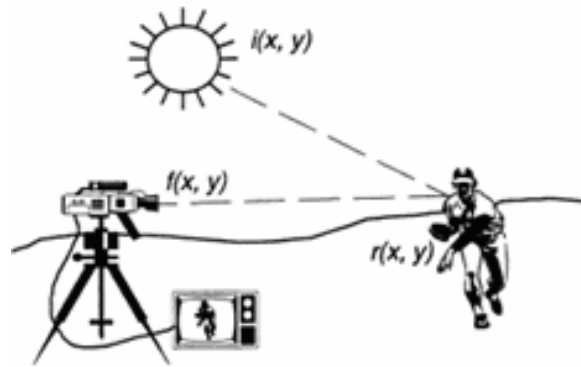


Figura 1.1. Formación de una imagen [EDWA99].

En la figura 1.1 se muestra cómo la luz radiante del Sol se refleja en la superficie del jugador del béisbol y a continuación es capturada por la cámara y transmitida a través de un medio de transmisión a un receptor de televisión. La imagen formada en la cámara se puede expresar matemáticamente como

$$f(x, y) = i(x, y) * r(x, y)$$

La ecuación anterior expresa el modelo de la imagen en la cámara en función de la iluminación de luz $i(x, y)$ y la reflexión del objeto $r(x, y)$. La función $r(x, y)$ varía entre 1 y 0. La función $f(x, y)$ describe la energía luminosa de la imagen con respecto a las coordenadas espaciales x e y . Así pues una imagen es una distribución espacial de la energía luminosa $f(x, y)$. Dicha función sólo puede tomar valores reales positivos [DATE01]. El procesado digital de imágenes requiere transformar la función continua que representa la imagen en una función discreta.

La idea básica del muestreo y la cuantización se ilustra en la figura 1.2 [GONZ07]. La figura 1.2a muestra una imagen $f(x, y)$ continua respecto a las coordenadas x e y así como respecto al valor de la función o amplitud.

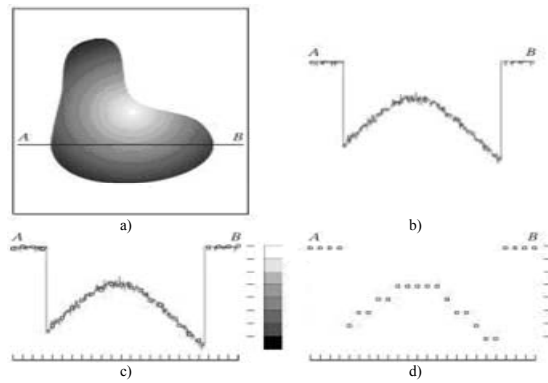


Figura 1.2. Generación de una imagen digital, a) la imagen continua, b) una línea de barrido de A a B en la imagen continua, c) muestreo, d) cuantización.

La digitalización de los valores de las coordenadas se denomina muestreo. La digitalización de los valores de la amplitud corresponde a la cuantización o conversión analógico/digital [GONZ07]. La función unidimensional de la figura 1.2b corresponde a los niveles de intensidad o amplitud de la imagen continua a lo largo del segmento AB en la figura 1.2a. En dicha figura aparece una variación aleatoria de los valores de la función debido al ruido. La figura 1.2c ilustra el muestreo de la función unidimensional. Para ello se toman valores equi-espaciados a lo largo de la línea AB. La ubicación espacial de cada muestra se indica mediante una marca vertical en la parte inferior de la figura. Las muestras se presentan como pequeños cuadrados blancos superpuestos en la función. Si bien el muestreo permite discretizar el número de valores de la función el valor (amplitud) es continuo. La cuantización o discretización de la amplitud permite disponer de la función digital que representa la imagen (figura 1.2d). El proceso de cuantización da lugar a una pérdida de información que se conoce como error de cuantización o ruido de cuantización [WHIT05].

Una imagen digital es una función bidimensional $f(x,y)$ que se representa mediante la siguiente matriz:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ f(M-1,0) & \dots & & f(M-1,N-1) \end{bmatrix}$$

donde M es el número de filas o valores discretos de la variable x y N es el número de columnas o valores discretos de la variable y . El proceso de digitalización requiere tomar decisiones respecto a los valores de muestreo y cuantización (N, M y L para el número de niveles discretos de intensidad). Debido a requerimientos de procesamiento y almacenamiento el número de niveles de intensidad normalmente es una potencia de 2:

$$L = 2^k$$

donde k es el número de bits en cada elemento de la matriz (número de bits para codificar el nivel de intensidad). El número de bits (B) necesarios para almacenar una imagen digitalizada NxM viene dado por

$$B = N \times M \times k$$

La expresión anterior contiene los parámetros que definen la resolución de una imagen. La resolución viene dada por el número de muestras y niveles de intensidad necesarios para realizar la aproximación de la imagen [JAHN05, PITA00, WILL93, RUSS07]. Así el grado de detalle discernible de una imagen depende de los parámetros que definen su resolución. La figura 1.3 muestra una imagen digital con diferentes valores de los parámetros de muestreo (N y M) para un valor fijo del parámetro de cuantización $L=256$ ($k=8$).

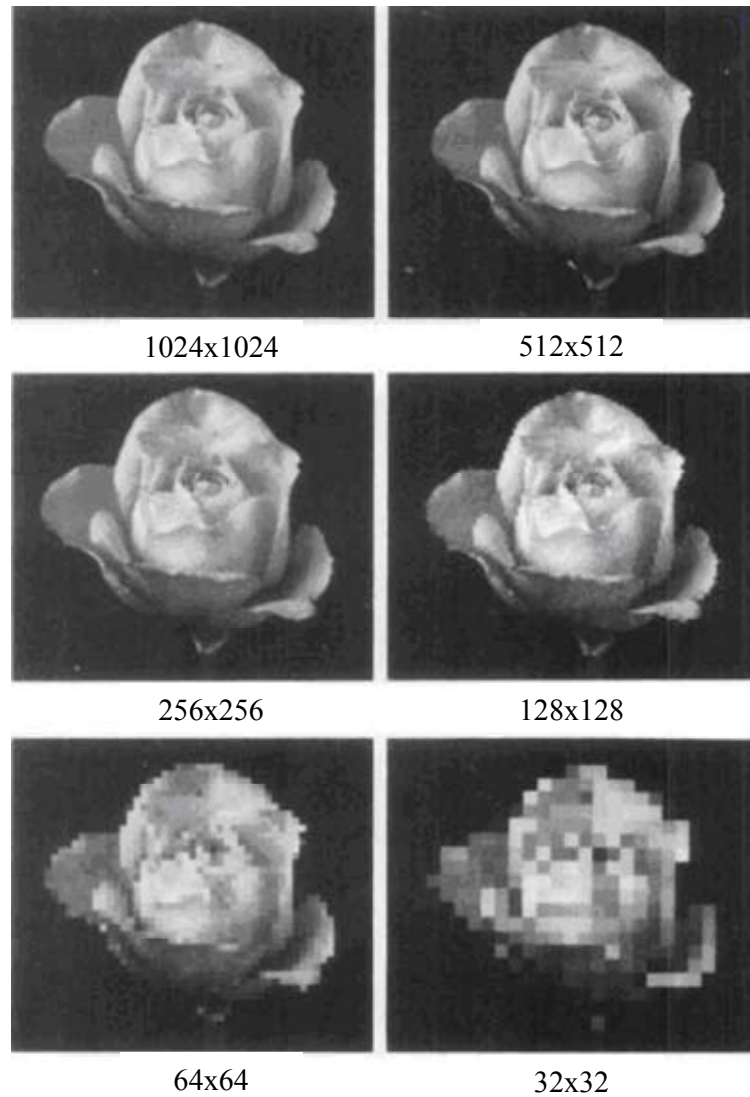


Figura 1.3. Efecto de reducir la resolución espacial NxM.

La figura 1.4 muestra el efecto producido por la variación del número de bits usados para representar los niveles de intensidad en una imagen ($L=256, 2, 4, 32$ respectivamente). En este caso el tamaño de la imagen está fijado al valor $N \times M = 400 \times 400$.



Figura 1.4. Efecto de variar el número de niveles de intensidad.

De lo anterior podemos extraer como conclusión que un elemento de una imagen digital o píxel está representado por bits de información. Un píxel es pues una unidad de información pero no una unidad de medida ya que no se corresponde con un tamaño concreto. Un píxel puede ser muy pequeño (0.1 milímetros) o muy grande (1 metro) [RUSS07].

1.2. Fundamentos y transformaciones de colores

1.2.1. Imágenes monocromáticas

Un mecanismo de cuantización básico consiste en representar cada píxel con dos posibles valores: blanco y negro. En este caso tan sólo se requiere 1 bit para representar el nivel de intensidad. Esta codificación se realiza comparando la intensidad de cada

píxel con un valor denominado umbral. El resultado de la cuantización es una imagen binaria.

Las operaciones con imágenes binarias permite aplicar operadores booleanos (and, or, xor, complemento, etc) para combinar los datos de diferentes imágenes binarias [RUSS02]. La figura 1.5 muestra algunos ejemplos de combinación de imágenes aplicando operadores booleanos píxel a píxel.

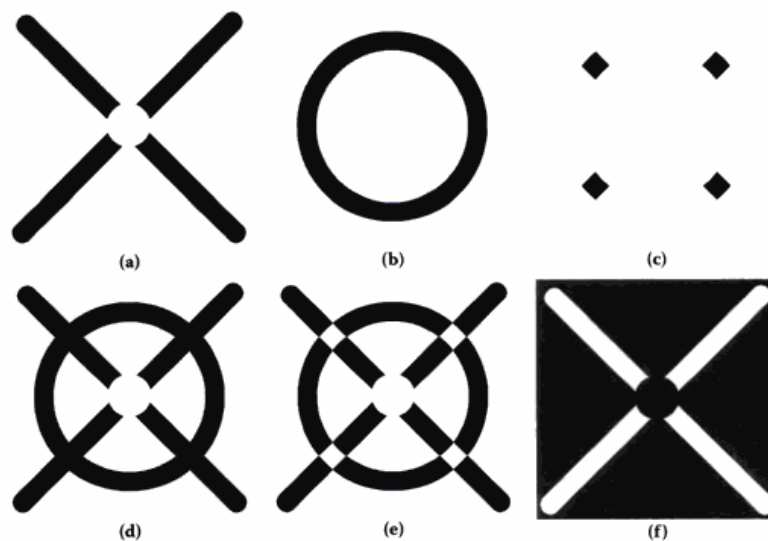


Figura 1.5. Operaciones de combinación de imágenes binarias (a y b).
(c) $A \vee B$; (d) $A \wedge B$; (e) $A \oplus B$; (f) $\neg A$

A medida que se aumenta el número de bits se pueden representar diferentes niveles de intensidad. En el caso de imágenes monocromáticas el número de bits utilizados para cada uno de los píxeles determina el número de los diferentes niveles de brillo disponibles. Una representación típica corresponde a codificar con 8 bits por píxel lo que permite disponer de 256 niveles de gris. Esta representación ofrece suficiente resolución de brillo para el sistema visual humano y proporciona un adecuado margen de ruido. Por otro lado la representación de 8 bits es adecuada ya que corresponde a un byte de información. En algunas aplicaciones, tales como imágenes médicas o en astronomía, se utilizan codificaciones de 12 o 16 bits por píxel [UMBA05].

1.2.2. Imágenes en color

El color se forma mediante la combinación de los tres colores básicos: rojo, verde y azul (en inglés correspondiente a las siglas RGB) y puede expresarse mediante una tripleta de valores de 0 a 1 (R, G, B). La tabla 1.1 muestra ejemplos de colores definidos mediante estas tripletas.

Color	R	G	B
Blanco	1	1	1
Rojo	1	0	0
Amarillo	1	1	0
Verde	0	1	0
Turquesa	0	1	1
Gris	0.5	0.5	0.5
Rojo Oscuro	0.5	0	0
Azul	0	0	1
Aguamarina	0.5	1	0.83
Negro	0	0	0

Tabla 1.1. Ejemplos de generación de colores RGB

Utilizando los 8 bits monocromo estándar como un modelo la imagen en color correspondiente tendría 24 bits por píxel, 8 bits para cada una de las tres bandas de colores (rojo, verde y azul). La figura 1.6 muestra una representación de una imagen en color RGB. Cada píxel (x,y) de la imagen tiene 3 planos que representan la intensidad de cada color R ($I_R(x,y)$), G ($I_G(x,y)$) y B ($I_B(x,y)$).

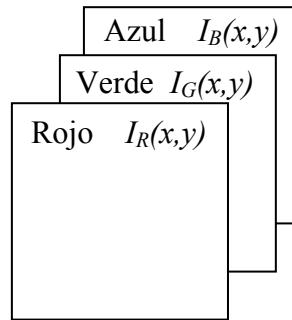


Figura 1.6. Representación de imágenes en color RGB.

En muchas aplicaciones es conveniente transformar la información de color RGB en un espacio matemático que separe la información de brillo de la información de color. A continuación vamos a mostrar algunas de estas transformaciones del modelo de color [UMBA05].

El modelo HSL es una representación en tres planos correspondiente a Tono/Saturación/Luminosidad (*Hue/Saturation/Lightness*). La figura 1.7 muestra la transformación del mapa RGB al mapa HSL. Dicha transformación corresponde a las siguientes relaciones:

$$H = \begin{cases} \theta & \text{si } B \leq G \\ 360 - \theta & \text{si } B > G \end{cases}$$

donde

$$\theta = \cos^{-1} \left\{ \frac{\frac{1}{2}[(R - G) + (R - B)]}{[(R - G)^2 + (R - B)(G - B)]^{1/2}} \right\}$$

$$S = 1 - \frac{3}{(R + G + B)} [\min(R, G, B)]$$

$$L = \frac{(R + G + B)}{3}$$

Estas expresiones asumen que los valores RGB están normalizados entre 0 y 1, y θ se mide en grados desde el eje de color rojo.

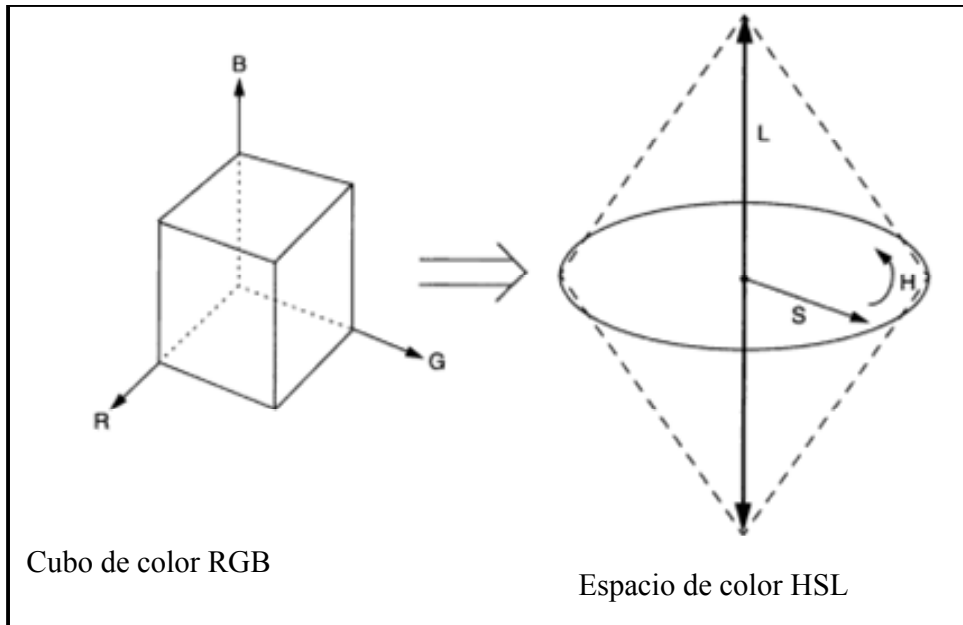


Figura 1.7. Transformación RGB a HSL.

Otro modelo es el denominado YUV. El plano Y indica la luminancia y los planos U y V son los dos componentes de crominancia [QSHI00]. La luminancia Y se puede determinar a partir del modelo RGB mediante la siguiente relación:

$$Y = 0.299R + 0.587G + 0.114B$$

Los otros dos componentes de crominancia U y V se definen como diferencias de color de la siguiente manera.

$$U = 0.493(B - Y)$$

$$V = 0.877(R - Y)$$

Por lo tanto expresando de forma matricial estas relaciones tenemos que:

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Un modelo de color utilizado en los sistemas de televisión NTSC (*National Television Systems Committee*) es el modelo YIQ. Este modelo surgió para compatibilizar la televisión en color con la televisión en blanco y negro que sólo requiere del componente de luminancia. El plano Y corresponde al componente de luminancia. Los parámetros I y Q son generados en relación al método de modulación utilizada para codificar la señal portadora. La relación entre los componentes de crominancia y los parámetros I y Q corresponden a las transformaciones lineales siguientes:

$$\begin{aligned} I &= -0.545U + 0.839V \\ Q &= 0.839U + 0.545V \end{aligned}$$

Por lo tanto es posible expresar YIQ directamente en términos de RGB:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Por su parte el sistema de televisión SECAM (*Sequential Couleur a Memoire*) utiliza el modelo YDbDr. La relación entre YDbDr y RGB es la siguiente.

$$\begin{bmatrix} Y \\ Db \\ Dr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.450 & -0.883 & 1.333 \\ -1.333 & 1.116 & -0.217 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

donde

$$\begin{aligned} Db &= 3.059U \\ Dr &= -2.169V \end{aligned}$$

De los modelos anteriores (YUV, YIQ y YDbDr) se observa que los componentes de crominancia I, Q, Db y Dr son transformaciones lineales de U y V. Por lo tanto estos modelos de color están íntimamente relacionados entre sí.

Los parámetros U y V en el modelo YUV pueden tomar valores positivos y negativos. Con objeto de tener componentes de crominancias no negativas se puede realizar un escalado de los parámetros YUV. Esto da lugar a la codificación YCbCr que se utiliza en la codificación de las normas internacionales de JPEG y MPEG [QSHI00].

La transformación de una señal RGB de 24 bits en el modo YCrCb se realiza de la siguiente manera:

$$\begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 0.257 & 0.504 & 0.98 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} 16 \\ 128 \\ 128 \end{pmatrix}$$

El factor 128 se incluye con objeto de mantener los datos en el rango [0,255] con 8 bits por banda de color en los datos.

Los sistemas de color anteriores son independientes del dispositivo, creando colores coherentes con independencia de los dispositivos concretos para crear o reproducir la imagen (monitores, impresoras, etc.). Estos modos permiten cambiar la luminosidad de una imagen sin alterar los valores de tono y saturación del color, siendo adecuados para transferir imágenes de unos sistemas a otros pues los valores cromáticos se mantienen independientes del dispositivo de salida de la imagen.

Para la impresión a color se utiliza el modelo de color sustractivo CMY. En este modo se restan del color blanco los colores cian, magenta o amarillo (CMY) para generar la gama de colores. La conversión de RGB a CMY se define como sigue (estas ecuaciones que asuman los valores RGB son normalizadas para el rango de (0 a 1)):

$$\begin{aligned} C &= 1 - R \\ M &= 1 - G \\ Y &= 1 - B \end{aligned}$$

El cian absorbe la luz roja, el magenta la verde y el amarillo la luz azul. Por lo tanto para imprimir una imagen RGB normalizado en verde con valores (0, 1, 0) se usan los

valores CMY (1, 0, 1). Para este ejemplo el cian absorbe la luz roja y el amarillo absorbe la luz azul dejando sólo la luz verde que se refleje.

La figura 1.8 muestra la representación tridimensional de los espacios básicos de representación del color dependiente de un dispositivo aditivo (RGB) o sustractivo (CMY).

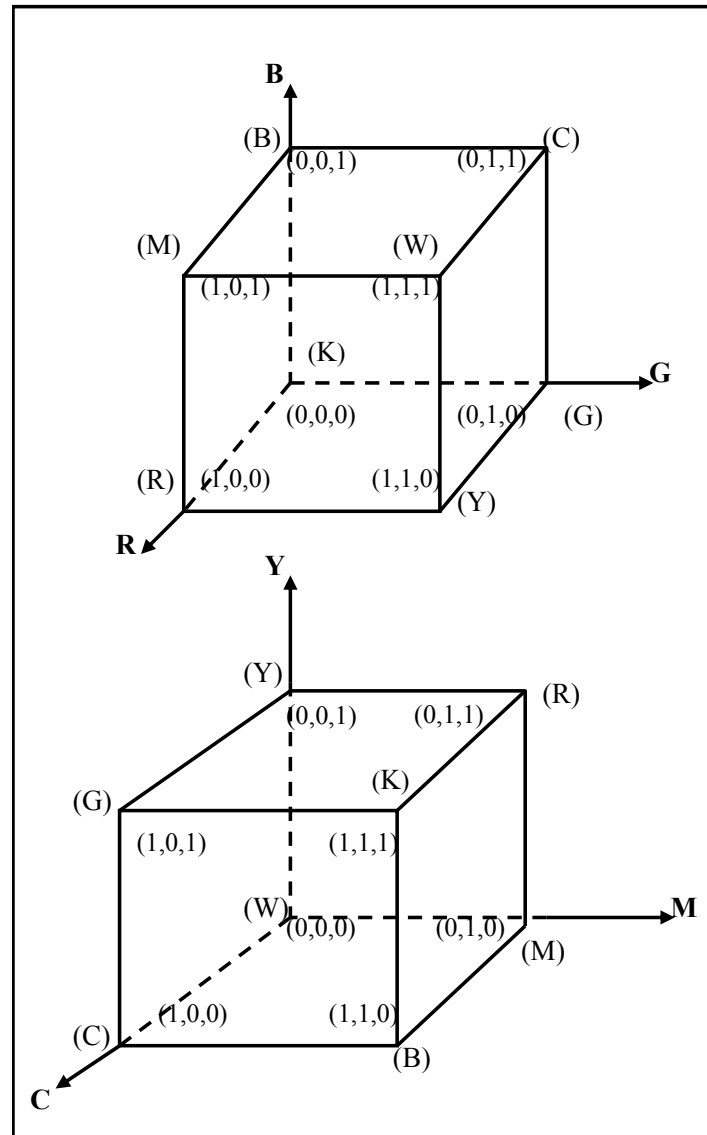


Figura 1.8. Representación tridimensional de los espacios básicos de color de un dispositivo aditivo (RGB) o sustractivo (CMY).

En la práctica esto no permite generar el color negro de manera adecuada por lo que se añade tinta negra y el proceso de impresión se denomina a cuatro colores CMYK. El

modo CMYK, trabaja con cuatro canales de 8 bits (32 bits de profundidad de color), ofreciendo una imagen cuatricromática compuesta por los 4 colores primarios para impresión: Cian (C), Magenta (M), Amarillo (Y) y Negro (K). Se trata de un modelo de color sustractivo en el que la suma de todos los colores primarios produce teóricamente el negro. El principal inconveniente es que este modo sólo es operativo en sistemas de impresión industrial y en las publicaciones de alta calidad ya que, exceptuando los escáneres de tambor que se emplean en fotomecánica, el resto de los digitalizadores comerciales trabajan en modo RGB. El proceso de convertir una imagen RGB al formato CMYK crea una separación de color. En general es mejor convertir una imagen al modo CMYK después de haberla modificado.

1.3. Formatos de almacenamiento de imágenes digitales

Un formato describe la forma en que los datos que representan una imagen son almacenados. Los datos de la imagen deben ser representados en una determinada forma física para ser almacenados y transmitidos. El objetivo de los formatos de imagen es garantizar que los datos se almacenan de acuerdo con un conjunto de reglas previsibles, lo que garantiza la independencia con el dispositivo de información [TERR08].

Existe una gran variedad de formatos de almacenamiento de imágenes [BRIN99, CAMP02, MIAN99, RICH06, SLUD07, WILK98]. La razón de esta proliferación se debe, por un lado, a que hay muchos tipos diferentes de imágenes y aplicaciones con distintas necesidades. Por otro lado también hay razones de cuota de mercado, propiedad de la información y falta de coordinación dentro de la industria de imágenes. Sin embargo existen algunos formatos estándares que son ampliamente utilizados [UMBA05, BURG07].

1.3.1. Formato BMP

El formato BMP (Bit MaP) es un formato definido por Windows para almacenar imágenes. Se ha modificado varias veces desde su creación, pero se ha mantenido estable desde la versión 3 de Windows. El formato BMP soporta imágenes con 1, 2, 4, 8, 16 y 32 bits por píxel aunque los archivos usuales son de 16 y 32 bits por píxel. El

formato BMP permite la compresión de las imágenes, aunque no es muy habitual su utilización comprimida.

El fichero BMP está compuesto por una cabecera con información sobre el formato utilizado para recoger la imagen y el cuerpo del mismo contiene la codificación apropiada de la imagen. En la cabecera se encuentra la información sobre el tamaño de la imagen, la paleta utilizada si la hubiese, el uso o no de la compresión y otras características de la imagen. A continuación se añade el cuerpo de la imagen que contiene los píxeles de la imagen en bruto o bien comprimida con el mecanismo de compresión RLE (*Run Length Encoding*, ver capítulo 2) [SALO04]. La estructura de archivos BMP se muestra en la figura 1.9.

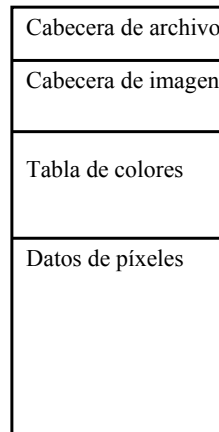


Figura 1.9. Estructura de archivos de mapa de bits

La paleta o tabla de colores es un diccionario en el que se recoge por cada valor posible para la imagen su correspondencia cromática. Dependiendo del número de bits disponibles para la paleta será necesaria o no su inclusión en la imagen. De esta forma, para imágenes con 24 bits no es necesario disponer de paleta, ya que en la propia imagen se indica el color de cada píxel.

En el cuerpo de la imagen se recoge por cada píxel de la misma el color que le corresponde. Si existe paleta de colores será el índice correspondiente en la misma y si no existe indicará directamente el color utilizado. Un ejemplo de una imagen con una paleta de colores de 4 bits se muestra en la figura 1.10.

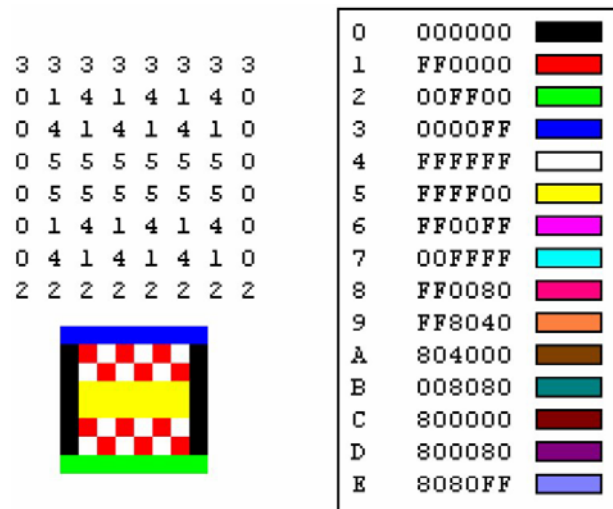


Figura 1.10. Imagen BMP con paleta de 4 bits (16 colores)

La imagen queda representada por los índices de entrada en la paleta de colores. De este modo, la fila superior de la imagen, que es de color azul, queda almacenada con el valor 3, que es el índice correspondiente al azul en la paleta de colores. De esta forma el tamaño de la imagen se ha visto reducido, ya que no ha sido necesario escribir los 24 bits de los colores para cada entrada, aunque también se ha limitado el número de colores posibles a 16. Si el número de colores es muy diverso la inclusión de la paleta de colores también afectará al tamaño de la imagen resultante ya que tiene que incluirse siempre con la imagen. Normalmente la información de la imagen no se recoge como en el ejemplo de la figura 1.10 sino que las filas se encuentran en orden inverso almacenándose en primer lugar la información sobre la última fila de la imagen.

1.3.2. Formato GIF

Las imágenes almacenadas como BMP ofrecen una buena calidad, pero al no ser usual el uso de la compresión los archivos que las contienen tienden a ser de gran tamaño. Por este motivo surgieron otros formatos que permiten la compresión de las imágenes como puede ser el formato GIF (*Graphics Interchange Format*).

El formato GIF fue propuesto por la compañía CompuServer y usa como algoritmo de compresión LZW (*Lempel Ziv Welch*) [WADE94]. En este formato también se usan paletas de colores, pudiendo ir desde los 2 colores hasta los 256. Es posible utilizar este

formato con paletas mayores, aunque no está destinado para este fin y no es muy frecuente.

Al estar limitada la paleta de colores hasta las 256 posibilidades las imágenes que pueden ser almacenadas sin pérdida de información son aquellas que puedan representarse con esta cantidad de colores. Cualquier imagen que supere los 256 colores verá reducida su paleta a este límite, por lo que se perderá información. Por este motivo este formato es recomendable para imágenes pequeñas y de colores sólidos, no estando recomendado su uso en imágenes fotográficas o de color verdadero (paleta de color de 24 bits).

La compresión de este formato se basa en agrupar la información de los colores contiguos que son iguales, de forma que si en la imagen aparecen varios píxeles con el mismo color, en vez de repetir esta información como se hace en el formato BMP, aquí se indica el color y el número de píxeles consecutivos que han de ser coloreados con el mismo.

1.3.3. Formato PNG

El formato PNG (*Portable Network Graphics*) surgió para mejorar algunos aspectos del GIF, como la limitación del número de colores en su paleta, así como para solventar algunos aspectos legales debido al uso de patentes en el formato de compresión.

La paleta de colores en este formato está dividida en canales. En imágenes a color se pueden asimilar estos canales con los niveles de rojo, verde y azul de un píxel, tal y como ocurre en el formato BMP. Cada canal dispone de un número determinado de bits, ampliando así las posibilidades de uso de las paletas. Dependiendo de los canales disponibles en la imagen se dispone de una mayor o menor profundidad de color en la imagen. Las diferentes posibilidades se describen en la tabla 1.2. En la columna bits por canal se indican cuantos bits se utilizan por cada canal para recoger la información del píxel en cada tipo de imagen. El número representado en la tabla indica el número total de bits necesarios por cada píxel y se obtiene multiplicando el número de bits por canal por el número total de canales de la imagen.

Imagen	Bits por canal				
	1	2	4	8	16
Indexada	1	2	4	8	-
Escala de grises	1	2	4	8	16
Escala de grises con transparencias	-	-	-	16	32
Color verdadero	-	-	-	24	48
Color verdadero con transparencias	-	-	-	32	64

Tabla 1.2. Profundidad de color en el formato PNG.

El formato PNG también utiliza la compresión para reducir el tamaño de la imagen. El proceso utilizado para este fin se conoce como deflación y consiste en la utilización conjunta de los algoritmos LZ77 [TASI04] y una codificación de Huffman [BRYA07].

1.3.4. Formato JPEG

JPEG (*Joint Photographic Experts Group*) es un algoritmo diseñado por un comité de expertos del ISO/CCITT estandarizado internacionalmente en el año 1992 que trabaja con imágenes tanto en escala de grises como a color. Hay diferentes versiones y mejoras que se han ido produciendo, existiendo tanto versiones con pérdidas como sin pérdidas. El algoritmo aprovecha la forma en la que el sistema visual humano trabaja para eliminar información de la imagen que no es detectada por el mismo. Así, el ojo humano es más sensible a cambios en la luminancia que en la crominancia y de igual forma detecta mejor cambios de brillo en zonas homogéneas que en aquellas con gran variación.

Dentro del formato JPEG existen diferentes métodos para llevar a cabo alguno de los pasos a desarrollar, dando lugar a varias categorías de imágenes. Como ya se ha comentado anteriormente es posible realizar compresiones de las imágenes con pérdidas o sin pérdidas, incluso ajustar los niveles de pérdidas que se deseen. La figura 1.11 muestra un ejemplo de una imagen JPEG con distintas calidades. El proceso de compresión JPEG se describe con detalle en el capítulo 2.

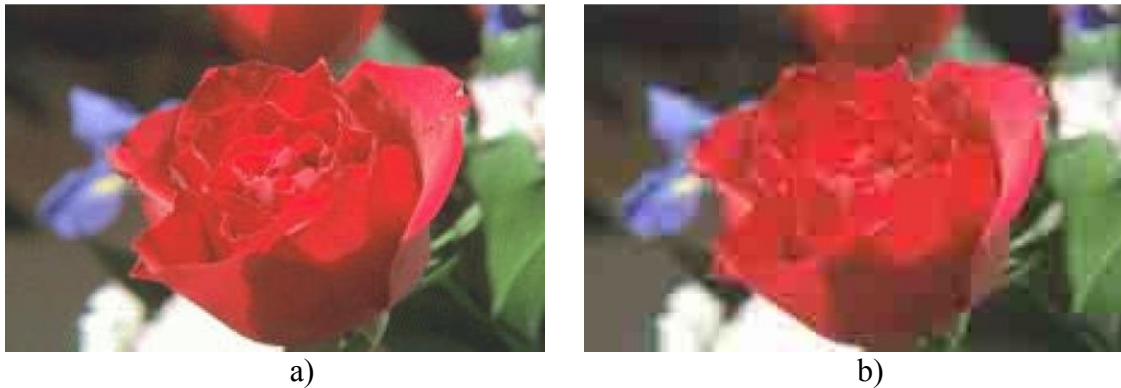


Figura 1.11. Imagen comprimida JPEG a) máxima calidad b) mínima calidad

1.3.5. Formato TIFF

El formato TIFF (*Tagged Image File Format*) fue originalmente desarrollado por la Aldus Corporation en 1980 como un intento de proporcionar un formato estándar para el almacenamiento y el intercambio de imágenes en blanco y negro creados por escáneres y aplicaciones de edición. Este formato se utiliza como contenedor de imágenes. En un fichero TIFF se almacena la imagen y una serie de etiquetas que recogen información sobre la misma, como puede ser el tamaño de la misma, la disposición de la información, la compresión utilizada, etc. Una característica interesante que proporciona este formato es la posibilidad de almacenar más de una imagen en un único fichero.

Las imágenes almacenadas pueden ser comprimidas utilizando varios métodos entre los que se encuentran el LZW como en el formato GIF y la compresión utilizada en JPEG.

La utilización de etiquetas permite un manejo más preciso de estos formatos. Se permiten el uso de etiquetas privadas o metadatos que aporten mayor información sobre la imagen. Esta información no es tenida en cuenta en aquellas aplicaciones que no entiendan el significado de estas etiquetas. Sin embargo el formato TIFF presenta algunos problemas ya que las disposiciones de los datos de la imagen en el archivo TIFF dan lugar a ficheros de gran tamaño.

1.4. Histograma de la imagen

El histograma de una imagen es una herramienta visual de gran aceptación y utilidad para el estudio de imágenes digitales. Con una simple mirada puede proporcionar una idea aproximada de la distribución de niveles de iluminación, el contraste que presenta la imagen y alguna pista del método más adecuado para manipularla. El histograma de una imagen se puede definir como una función que representa el número de veces que se repiten cada uno de los valores de los píxeles. Así el histograma de una imagen digital con L niveles de iluminación en el rango $[0, L-1]$ es el gráfico de una función discreta de la forma:

$$p_r(r_k) = \frac{n_k}{n}$$

donde r_k es un nivel de iluminación (siendo $k=0,1,2,\dots,L-1$), n_k es el número de píxeles en la imagen con el valor r_k y n es el número total de píxeles de la imagen [GONZ07].

Las intensidades se representan de manera gráfica en el eje de cartesiana de abcisa mientras que el número de ocurrencias para cada intensidad se representan en el eje de ordenadas. La frecuencia de aparición de cada nivel de luminancia en el histograma se muestra de forma relativa debido al hecho de que el valor absoluto puede variar en función del tamaño de la imagen.

La tabla 1.3 muestra un ejemplo para una imagen de 64×64 píxeles codificados con 3 bits (8 niveles intensidad). El histograma de la imagen se muestra en la figura 1.12. Otros ejemplos de imágenes con sus correspondientes histogramas se muestran en la figura 1.13.

r_k	n_k	$p_r(r_k)$
$r_0=0$	790	0.19
$r_0=1$	1023	0.25
$r_0=2$	850	0.21
$r_0=3$	656	0.16
$r_0=4$	329	0.08
$r_0=5$	245	0.06
$r_0=6$	122	0.03
$r_0=7$	81	0.02

Tabla 1.3. Ejemplo de distribución de los niveles de gris de una imagen de 64x64 y 8 niveles de intensidad.

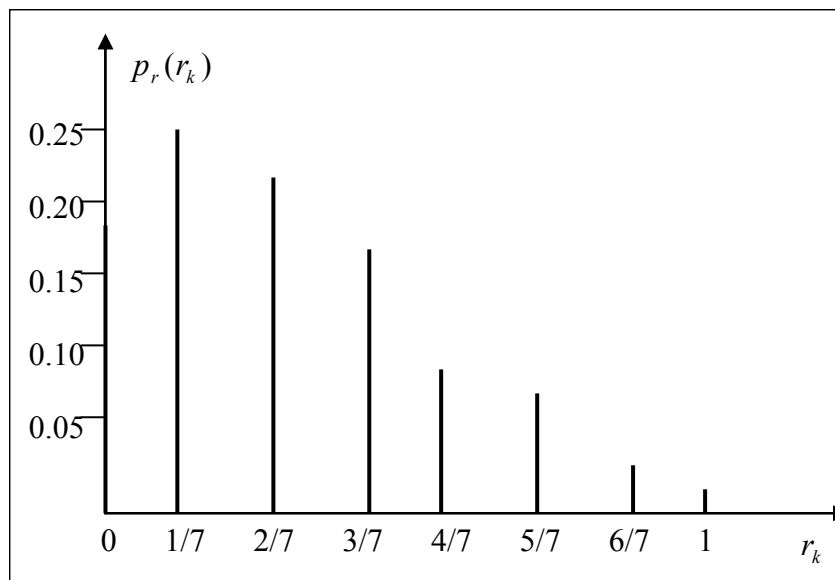


Figura 1.12. Histograma del ejemplo de la imagen descrita en la tabla 1.3.

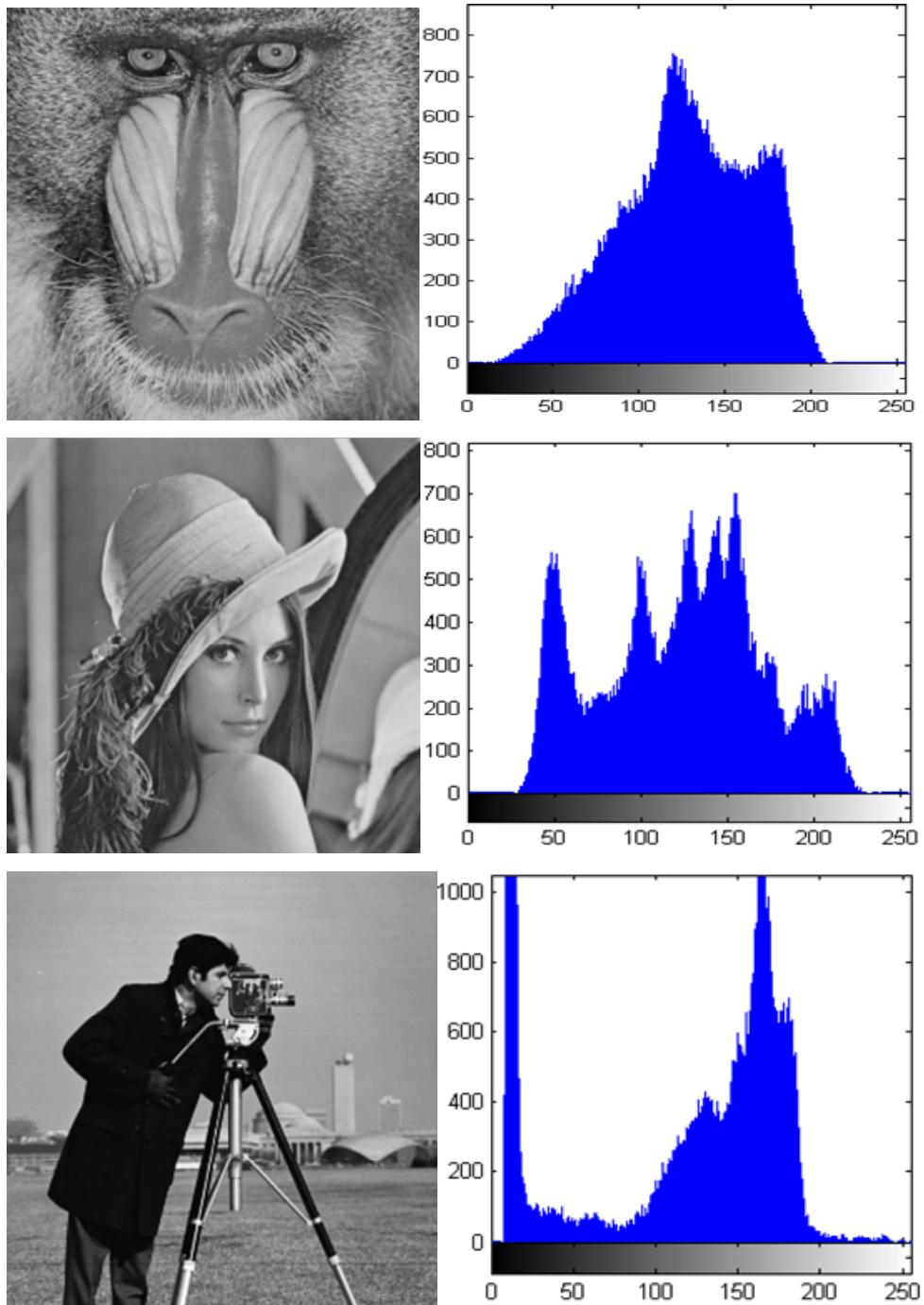


Figura 1.13. Ejemplos de imágenes con sus histogramas.

Muchas aplicaciones de procesamiento digital de imágenes se basan en el procesamiento del histograma. En los sucesivos capítulos plantearemos diferentes técnicas de procesamiento basadas en esta herramienta.

1.5. Aplicación del *Soft Computing* en el procesado de imágenes

El *Soft Computing* es un campo emergente que busca la sinergia de diferentes campos tales como la lógica difusa, las redes neuronales, la computación evolutiva, el razonamiento probabilística, entre otros. Lotfi Zadeh fue el primero en establecer el concepto de *Soft Computing* [ZADE94] planteando la integración de técnicas propias de la lógica difusa con las de redes neuronales y la computación evolutiva.

A diferencia de los sistemas de computación tradicionales, que se basan a la plena verdad y exactitud, las técnicas de *Soft Computing* explotan la imprecisión, la verdad parcial y la incertidumbre para un problema particular. Otra característica del *Soft Computing* consiste en la importancia que desempeña el razonamiento inductivo.

En los últimos años se ha incrementado el interés en el uso de técnicas basadas en *Soft Computing* para resolver problemas de procesado de imágenes cubriendo un amplio rango de dominios tales como análisis de imágenes, mejora y restauración de imágenes, visión artificial, procesado de imágenes médicas, procesado de video, segmentación, codificación y transmisión de imágenes, etc [KERR00, NACH07, REUS06, KAME07, BLOC06, NACH03].

Nuestro interés en la aplicación de técnicas basadas en *Soft Computing* viene dado por el objetivo de disponer de técnicas heurísticas de razonamiento que permitan generar implementaciones hardware de sistemas de procesado de imágenes de bajo coste y una alta velocidad de procesado. Es por ello que en los próximos capítulos se plantea la sinergia de técnicas basadas en el álgebra multivaluada de Łukasiewicz, lógica difusa y redes neuronales. En nuestro caso la conjunción de estas técnicas se realiza atendiendo a criterios de implementación, de razonamiento para la toma de decisiones y como mecanismo de cálculo para aprovechar las particularidades que estos sistemas nos ofrecen.

1.6. Resumen

En este capítulo se han presentando definiciones y conceptos básicos de aspectos relacionados con las imágenes, codificación, formato de almacenamiento, etc. Algunos de los conceptos que aquí se describen se utilizan en el resto de la memoria de tesis. Nos hemos centrado en dar algunas pinceladas sobre ideas generales del proceso de digitalización de imágenes, concepto de resolución, fundamentos del color y la transformación de diferentes representaciones. Estos conceptos junto con los formatos de almacenamiento serán recurrentes en esta tesis. Finalmente se da una muy breve pincelada sobre el campo del *Soft Computing* con objeto de centrar los mecanismos y herramientas de las que se ha hecho uso en el desarrollo de este trabajo.

Capítulo 2

Compresión de imágenes

Las técnicas de compresión de imágenes permiten eliminar la redundancia en la imagen con objeto de reducir la información que es necesario almacenar o transmitir. La redundancia es la correlación que existe entre los píxeles de la imagen. A la hora de considerar el tipo de redundancia que aparece en una imagen podemos realizar una clasificación en tres categorías [RAMI01]:

- 1) Redundancia espacial o correlación entre píxeles próximos.
- 2) Redundancia espectral o correlación entre diferentes planos de color o bandas espectrales.
- 3) Redundancia temporal o correlación entre imágenes adyacentes en una secuencia de imágenes (en aplicaciones de video).

La compresión de imágenes persigue reducir el número de bits necesarios para representar una imagen eliminando la redundancia espacial tanto como sea posible. La eliminación de la redundancia temporal entra dentro del ámbito de la codificación de video.

A la hora de clasificar los algoritmos de compresión se pueden organizar en dos categorías: algoritmos de compresión sin pérdidas y con pérdidas. En los esquemas de compresión sin pérdidas, la imagen reconstruida, después de la compresión, es numéricamente idéntica a la imagen original. Mediante este método de compresión sólo se pueden alcanzar unos modestos resultados. La compresión sin pérdidas se emplea en aplicaciones en las que los datos de la imagen en bruto son difíciles de obtener o bien contienen información vital o importante que debe conservarse como, por ejemplo, ocurre en aplicaciones de diagnósticos médicos por imágenes. Una imagen reconstruida después de una codificación con pérdidas contiene cierta degradación relativa a la original. Esta degradación a menudo es debida a que el esquema de compresión elimina completamente la información redundante. Sin embargo, los esquemas de codificación con pérdidas son capaces de alcanzar una compresión mucho más alta y, bajo condiciones normales de visión, no se aprecian estas pérdidas.

Entre los métodos de compresión sin pérdidas podemos destacar los siguientes:

- *Run-length encoding* (RLE); usado en PCX, BMP, TGA y TIFF.
- Codificación estadística (por entropía) como la codificación Huffman
- Algoritmos basados en diccionario como LZW usado en GIF y TIFF

Entre los métodos de compresión con pérdidas podemos destacar los siguientes:

- Reducción del espacio de colores y submuestreo de cromancia
- Algoritmos basados en transformada tales como DCT (JPEG) o *wavelet*.
- Compresión fractal

En general un sistema de compresión de imágenes consta de tres etapas [GONZ02] como se muestra en la figura 2.1: transformación, cuantificación y codificación. En la transformación se aplica una función que transforma el conjunto de datos de la imagen original en otro conjunto de datos que permita aplicar las estrategias de eliminación de redundancia. Esta fase es reversible y permite obtener la imagen original. Un ejemplo consiste en transformar una imagen bidimensional en una forma de onda lineal. Otro ejemplo lo constituye la aplicación de la transformada discreta del coseno (DCT) sobre

la imagen original. En la fase de cuantificación es cuando se elimina la redundancia. En esta fase es cuando se realiza la pérdida de información. Finalmente la fase de codificación construye la salida basada en un esquema de compresión sin pérdidas.

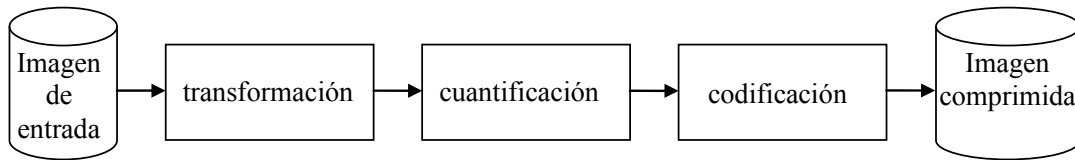


Figura 2.1. Etapas de un sistema de compresión de imágenes.

Nuestro objetivo en este capítulo se centra en presentar nuevos algoritmos de la compresión de imágenes. Los sistemas que presentamos permiten tanto la compresión con pérdidas y sin pérdidas. Los métodos de compresión se han implementado sobre FPGA y dan lugar a una imagen comprimida con una buena calidad y al mismo tiempo circuito de bajo coste y alta velocidad de procesado. Este capítulo se organiza en siete apartados. En el primer apartado se presentan las medidas de calidades en las técnicas de compresión de imágenes. En el segundo apartado se describe el algoritmo *Run-Length Encoding* (RLE). En el apartado tres se muestra la codificación de *Huffman*. En el apartado cuatro se presenta el algoritmo JPEG. En el apartado cinco se trata un algoritmo de muestreo uniforme y a continuación los algoritmos de eliminación de redundancia. En el apartado seis se presentan los algoritmos propuestos. Finalmente, se describe el diseño y la implementación hardware de los algoritmos propuestos.

2.1. Medida de calidad en compresión de imágenes

Para evaluar el rendimiento de cualquier técnica de compresión de imágenes es necesario tener en cuenta dos aspectos: el tamaño que se consigue en la compresión y el error que se comete en la aproximación. El tamaño de la imagen comprimida nos indica la eficiencia en la compresión. Dicha eficiencia puede medirse mediante el parámetro denominado la razón de la compresión. La razón de compresión se define como el cociente entre el tamaño original y el tamaño de la imagen comprimida.

$$CR = \frac{T_{orig}}{T_{comp}} : 1$$

La razón de compresión se referencia frente a la unidad. De esta manera el objetivo de todo algoritmo de compresión consiste en tener una razón de compresión lo mas alta posible. Valores por encima de 1 y alejados indican una buena compresión.

Un segundo aspecto a la hora de evaluar un algoritmo de compresión hace referencia a la calidad de la imagen obtenida. Para medir dicha calidad se requiere de algún parámetro que permita estimar el error que se comente en la transformación. En la literatura existen diversas alternativas a la hora de considerar el error de la aproximación.

El error medio absoluto (MAE, *Mean Absolute Error*) es una medida de la distancia entre la imagen original y la imagen comprimida. Este parámetro viene dado por la expresión siguiente para una imagen de NxM píxeles:

$$MAE = \frac{\sum_{i,j} (img_{orig}(i,j) - img_{final}(i,j))}{N \times M}$$

Otra medida corresponde al error cuadrático medio (MSE, *Mean Squared Error*) que representa la varianza:

$$MSE = \frac{\sum_{i,j} (img_{orig}(i,j) - img_{final}(i,j))^2}{N \times M}$$

Un parámetro muy utilizado es la raíz del error cuadrático medio (RMSE, *Root Mean Squared Error*) que corresponde al error estándar:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{\sum_{i,j} (img_{orig}(i,j) - img_{final}(i,j))^2}{N \times M}}$$

Otro parámetro muy empleado es la relación señal a ruido de pico (PSNR, *Peak Signal-to-Noise Ratio*):

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

donde MAX es el máximo valor que puede tomar un píxel en la imagen. Cuando los píxeles se representan usando B bits entonces $MAX = 2^B - 1$. 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).

2.2. *Run-Length Encoding (RLE)*

La codificación RLE es la forma más sencilla de compresión. Se utiliza ampliamente en la mayoría de formatos de archivos de mapas de bits, tales como TIFF, BMP y PCX. Este tipo de codificación reduce el tamaño físico de una repetición de cadena de símbolos. Consiste en convertir los símbolos idénticos consecutivos en un código formado por el símbolo y el número de veces que se repite. De esta manera la cadena siguiente

ABBBCCCCCDEFGGGGHI

Se codifica por esta otra en la que la longitud se reduce de 18 símbolos a 12:

A3B5CDEF4GHI

A la hora de implementar este esquema de codificación existen dos alternativas. La primera consiste en sustituir una cadena de símbolos consecutivos repetidos por dos bytes. El primer byte contiene un número que representa el número de veces que el símbolo está repetido. El segundo byte contiene al propio símbolo. En el caso de una imagen en blanco y negro cada píxel puede codificarse mediante un sólo byte. En este caso se requiere un bit para el valor del píxel (0 para negro y 1 para blanco) y se emplean los 7 bits restantes para especificar el número de repeticiones.

Puesto que la compresión RLE se basa en la codificación de secuencias con símbolos repetidos, los factores a tener en cuenta a la hora de implementar dicha técnica son:

- 1) el número de repeticiones consideradas para la codificación. Al tener que codificar una secuencia mediante un número y un símbolo, una secuencia demasiado corta podría traducirse en la nula compresión de los datos.
- 2) el número de bits con que se representará el contador de repeticiones.
- 3) el tamaño y naturaleza del alfabeto pues la representación de cada símbolo es determinante para la elección del tamaño de las secuencias con repetición.

2.3. Codificación estadística: Codificación de Huffman

La codificación de Huffman genera un código de longitud variable en el que la longitud de cada código depende de la frecuencia relativa de aparición de cada símbolo; cuanto más frecuente sea un símbolo su código asociado será más corto. Se trata de un código libre de prefijos, es decir, ningún código forma la primera parte de otro código. Esto permite que los mensajes codificados sean no ambiguos. Este enfoque se presentó por primera vez por David Huffman en 1952 para archivos de texto y ha dado lugar a muchas variaciones [GONZ02].

El principio en el que se basa el algoritmo de codificación de Huffman consiste en utilizar un menor número de bits para codificar los datos que se producen con más frecuencia. Los códigos se almacenan en un libro de códigos que pueden ser construidos para cada imagen o un conjunto de imágenes. En todos los casos el libro de códigos debe transmitirse para permitir la decodificación. Vamos a describir un ejemplo que ilustre este proceso. Imaginemos el siguiente flujo de datos:

AAAABCDEEEFFGGGH

La frecuencia de cada símbolo es la siguiente: A: 4, B: 1, C: 1, D: 1, E: 3, F: 2, G: 3, H: 1. A partir de esta frecuencia se puede generar un modelo estadístico que refleja la probabilidad de que cada valor:

A: 0,25, B: 0.0625, C: 0.0625, D: 0.0625, E: 0.1875, F: 0,125, G: 0.1875, H: 0.0625

A partir de estas probabilidades se genera un código de longitud variable que permite describir de forma univoca cada símbolo:

G	A	F	E	D	H	B	C
00	10	110	111	0100	0101	0110	0111

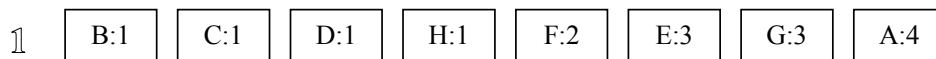
De esta manera la cadena original puede sustituirse por su código y cada símbolo puede reconocerse en la decodificación de manera univoca

AAAABCDEEEFFGGGH

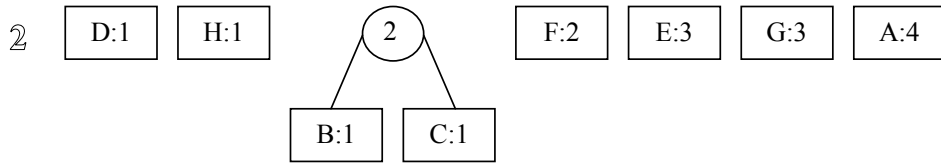
10 10 10 10 0110 0111 0100 111 111 111 110 110 00 00 00 0101

El proceso de asignación del código Huffman emplea un árbol binario en el que las hojas representan a los símbolos y el camino de la raíz a las hojas dan la representación binaria. Para ello el camino de la izquierda codifica un 0 y el de la derecha un 1.

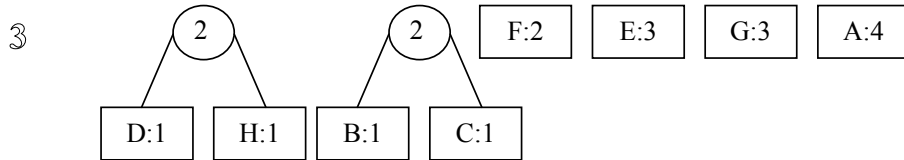
Vamos a ilustrar el procedimiento para el ejemplo anterior. El primer paso consiste en ordenar los símbolos en función de la frecuencia (o de su probabilidad) de menor a mayor. Esta ordenación se mantiene en todo el proceso ya que cada rama tiene un código 0 a la izquierda y 1 a la derecha.



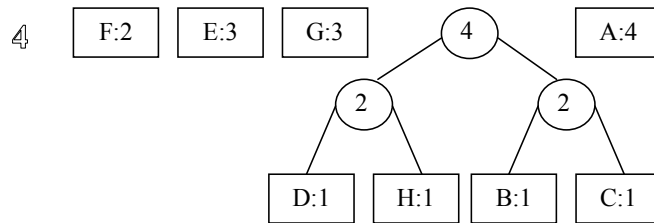
A continuación se agrupa los símbolos de menor frecuencia (B y C) en un árbol binario. La suma de sus frecuencias corresponde al valor de la raíz del árbol. Dicho árbol se ordena en la posición que le corresponde por el valor de la frecuencia agregada.



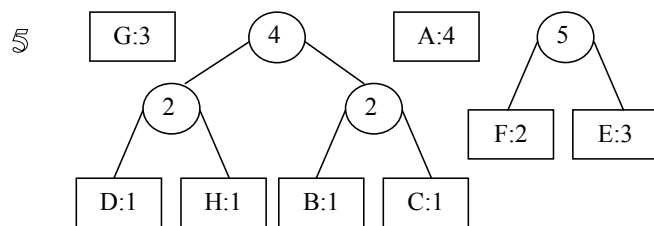
El proceso se repite hasta que todos los símbolos se han agrupado dentro de un único árbol. Así el tercer paso agrupará los símbolos D y H.

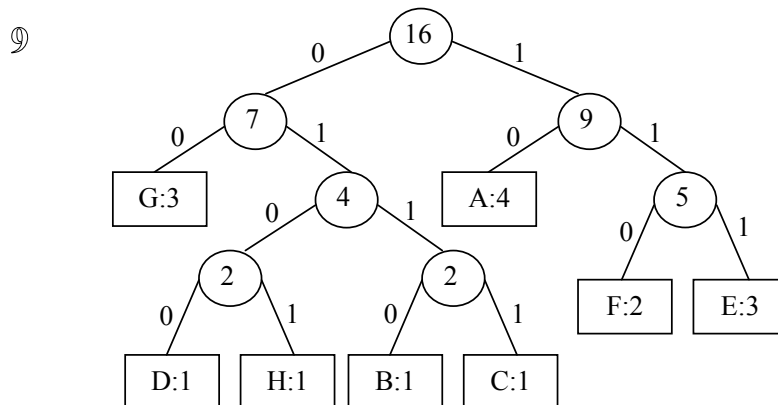
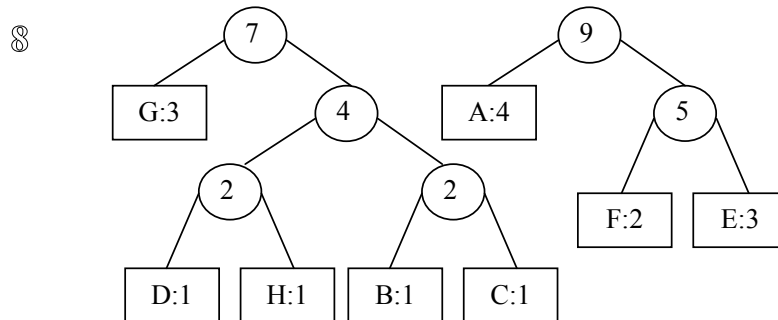
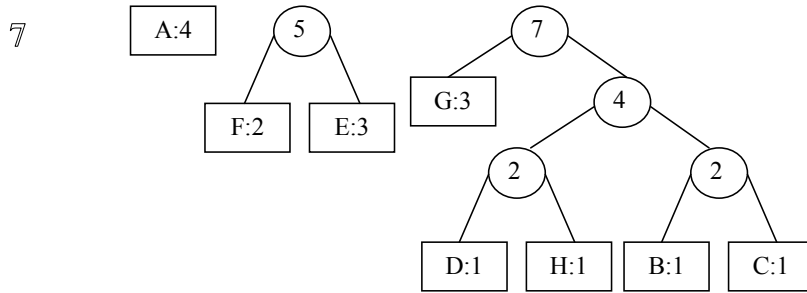
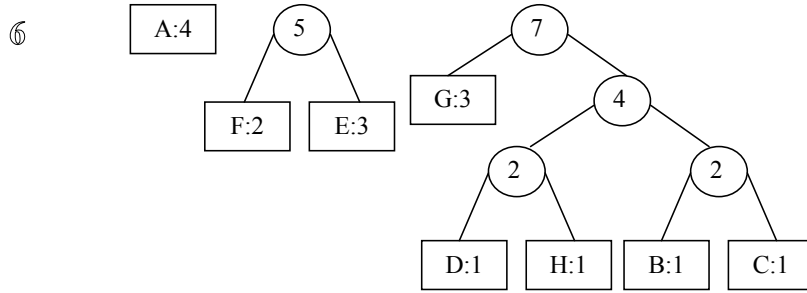


En la siguiente iteración se agrupan los árboles de menor frecuencia y se le asocia un árbol que tiene de frecuencia raíz el valor 4.



En la siguiente iteración se agrupan F y E en un árbol cuya raíz tiene frecuencia 5. Dicho árbol se ordena en la última posición ya que corresponde al de mayor frecuencia. En las sucesivas iteraciones se va construyendo el árbol binario agrupando los sub-árboles de menor frecuencia.





En la última iteración se tiene el árbol binario que asigna los códigos a cada símbolo. Cada rama de la izquierda asigna el valor 0 y el de la derecha el valor 1. Así el código de B es 0110. La longitud del código dependerá de la distancia del símbolo a la raíz. Así la longitud del código de B es de 4 bits mientras que la del código A es de 2

bits. Esta distancia depende de la frecuencia. A menor frecuencia mayor longitud de código.

Para un árbol binario T es posible calcular el número de bits necesarios para codificar una imagen. Así sea x un símbolo (valor de pixel de la imagen). Denominamos el alfabeto como H (conjunto de valores que toman los pixel). La frecuencia $f(x)$ donde $x \in H, \forall x$, corresponde al histograma de la imagen. Sea $d_T(x)$ la profundidad de la hoja x en el árbol T . El número de pixels necesarios para codificar la imagen viene dado por:

$$B(T) = \sum_{x \in H} f(x) \times d_T(x)$$

2.4. Comprensión basada en transformada: JPEG

Una de las técnicas de compresión de imágenes más ampliamente utilizada es el algoritmo JPEG. El nombre JPEG corresponde al acrónimo de *Joint Photographic Experts Group*. Se trata de un algoritmo estándar ISO y CCITT (ahora denominado ITU-T) [CCIT92, JPEG]. El nombre formal del JPEG como estándar es ISO/IEC IS 10918-1 | ITU-T *Recommendation* T.81. El documento que define el estándar IS 10918 tiene actualmente 4 partes:

- Parte 1 – El estándar JPEG básico que define las opciones y alternativas para la codificación de imágenes.
- Parte 2 – Establece el conjunto de reglas y comprobaciones para realizar el software de acuerdo con la parte 1.
- Parte 3 – Establece extensiones para mejorar el estándar incluyendo el formato de ficheros SPIFF.
- Parte 4 – Define métodos de registrar algunos parámetros usados para extender JPEG.

JPEG es un algoritmo de compresión con pérdidas (si bien existe una versión para compresión sin pérdidas) [GONZ02, PENN04]. Una de las características que hacen muy flexible el algoritmo es que es posible ajustar el grado de compresión. Si se especifica una compresión muy alta se perderá una cantidad significativa de calidad, mientras que con una tasa de compresión baja se obtiene una calidad muy parecida a la de la imagen original.

Este algoritmo de compresión utiliza dos características que el sistema visual humano posee para lograr eliminar información de la imagen que el ojo no es capaz de detectar. El primero de ellos consiste en que no tenemos la misma capacidad para apreciar las variaciones de crominancia que las variaciones de luminancia. La segunda, es que somos capaces de detectar ligeros cambios en el tono entre dos zonas de color adyacente pero, cuando la diferencia es grande, esta no tiene por qué ser codificada de forma precisa.

Esta técnica se basa en la aplicación de la transformada discreta del coseno (DCT, *Discrete Cosine Transform*). La DCT se utiliza en muchas técnicas de compresión tales como JPEG, H.261, H.263, MPEG-1, MPEG-2 y MPEG-4 entre otras. La DCT expresa una secuencia de datos finitos en términos de suma de cosenos a diferentes frecuencias. Así las filas de una matriz A de tamaño NxN son funciones coseno:

$$[A]_{ij} = \begin{cases} \sqrt{\frac{1}{N}} \cos\left(\frac{(2j+1)i\pi}{2N}\right) & i = 0; j = 0, 1, \dots, N-1 \\ \sqrt{\frac{2}{N}} \cos\left(\frac{(2j+1)i\pi}{2N}\right) & i = 1, 2, \dots, N-1; j = 0, 1, \dots, N-1 \end{cases}$$

Una ventaja de aplicar la DCT a una matriz de datos que representa una imagen es que permite concentrar mucha información en los primeros coeficientes.

El algoritmo JPEG está constituido por una serie de etapas que se muestran en la figura 2.2. La imagen se descompone en bloques de 8x8 pixels que se procesan de forma casi independiente. En la primera etapa se realiza una transformación del espacio de color para trabajar con valores YUV (o YCbCr). Así si una imagen en color está codificada en formato RGB se transforma al formato YUV. El componente de

luminancia representa en realidad una escala de grises, mientras que los otros dos componentes de crominancia son información de color. Esta transformación se realiza aplicando la siguiente ecuación:

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & -0.312 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \quad \begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 0.257 & 0.504 & 0.98 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} 16 \\ 128 \\ 128 \end{pmatrix}$$

Finalmente, en la fase de transformación del espacio de color, se requiere un cambio de nivel en el color ya que la DCT está centrada en el cero. Lo que se hace es restar a cada valor de color 128 para cambiar el signo de los coeficientes.

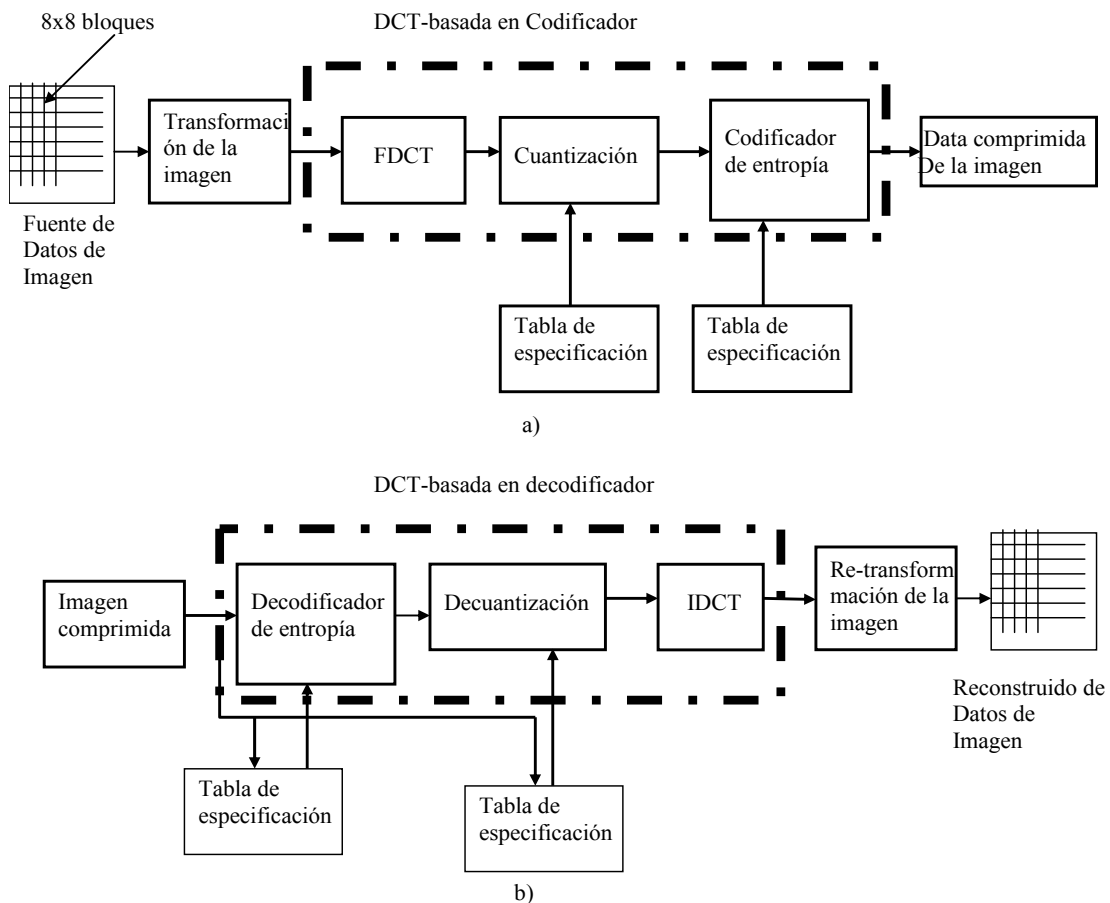


Figura 2.2. Etapas del proceso de a) compresión JPEG, b) descompresión.

La siguiente etapa realiza la DCT. Existen diversas alternativas a la hora de implementar la DCT. Una estrategia consiste en evaluar la siguiente expresión:

$$C = MFM^T$$

donde F es el bloque 8x8 y M es la matriz siguiente:

$$M = \begin{pmatrix} 0.3536 & 0.3536 & -0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & -0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & 0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & 0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & 0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & -0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & -0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & -0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{pmatrix}$$

Vamos a considerar el siguiente ejemplo de imagen 8x8:

$$F = \begin{pmatrix} 154 & 123 & 123 & 123 & 123 & 123 & 123 & 136 \\ 192 & 180 & 136 & 154 & 154 & 154 & 136 & 110 \\ 254 & 198 & 154 & 154 & 180 & 154 & 123 & 123 \\ 239 & 180 & 136 & 167 & 166 & 149 & 136 & 136 \\ 180 & 154 & 136 & 167 & 166 & 149 & 136 & 136 \\ 128 & 136 & 123 & 136 & 154 & 180 & 198 & 154 \\ 123 & 105 & 110 & 149 & 136 & 136 & 180 & 166 \\ 110 & 136 & 123 & 123 & 123 & 136 & 154 & 136 \end{pmatrix}$$

El resultado de realizar la DCT corresponde a la matriz siguiente en la que cada valor se ha redondeado a su entero más cercano:

$$C = \begin{pmatrix} 160 & 38 & 29 & 68 & 29 & 14 & -22 & -8 \\ 30 & 108 & 13 & 31 & 27 & -15 & 18 & -1 \\ -91 & -57 & 1 & -38 & -30 & 4 & 0 & 3 \\ -37 & -81 & -12 & -19 & -13 & 14 & 0 & 1 \\ -33 & 15 & 3 & -17 & 13 & -4 & 9 & -3 \\ -4 & -15 & 23 & -5 & 27 & 15 & 6 & 7 \\ 6 & -1 & 8 & 9 & -18 & -14 & 9 & 10 \\ -7 & 15 & -4 & -10 & 23 & -2 & 9 & 6 \end{pmatrix}$$

Una vez aplicada la DCT a una imagen la siguiente etapa corresponde a la cuantización. En esta etapa es cuando se realiza la pérdida de información. La cuantización se realiza aprovechando las características del ojo humano que se comentó anteriormente. En efecto, puesto que los valores de alta frecuencia no son apreciables pueden eliminarse sin que provoque efecto visual alguno. La cuantización consiste en realizar la siguiente operación:

$$C^*(i, j) = \text{Redondear} (C(i, j)/Q(i, j))$$

Donde Q es la matriz de cuantización. Los valores de la matriz Q se obtienen teniendo en cuenta el comportamiento visual del ojo humano. Así la calidad de la imagen comprimida dependerá de los valores que se hayan considerado en la matriz Q. Un ejemplo de matriz de cuantización es la siguiente:

$$Q = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

El resultado de aplicar la cuantización a nuestro ejemplo da lugar a la siguiente matriz:

$$C^* = \begin{pmatrix} 10 & 3 & 3 & 4 & 1 & 0 & 0 & 0 \\ 3 & 9 & 1 & 2 & 1 & 0 & 0 & 0 \\ -7 & -4 & 0 & -2 & -1 & 0 & 0 & 0 \\ -3 & -5 & -1 & -1 & 0 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

se selecciona uno. De esta manera se reduce la cantidad de información que representa la imagen a la mitad.

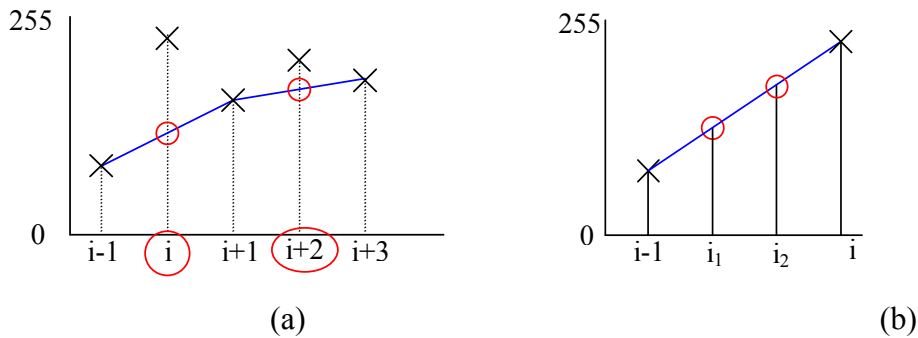


Figura 2.3. (a) Muestreo uniforme ($k=2$), (b) descompresión para $k=3$

Por otro lado el algoritmo de descompresión interpola los píxeles que han sido eliminados de la imagen hasta recuperar los N píxeles de la imagen original. A la hora de realizar esta reconstrucción un mecanismo puede ser repetir cada píxel k veces. Otra alternativa consiste en realizar interpolación lineal como se observa en la figura 2.3b para el caso $k=3$. En este caso se insertan nuevos píxeles entre cada dos mediante aproximación lineal.

La figura 2.4 muestra un ejemplo de muestreo uniforme de la imagen “soccer” de 64×64 píxeles. La razón de muestreo es $k=2$. Se puede observar los efectos de la aproximación tanto en la imagen comprimida (figura 2.4b) como en la descomprimida (figura 2.4c). Puesto que el mecanismo de interpolación se basa en una aproximación lineal a tramos se trata de una técnica de compresión con pérdidas.

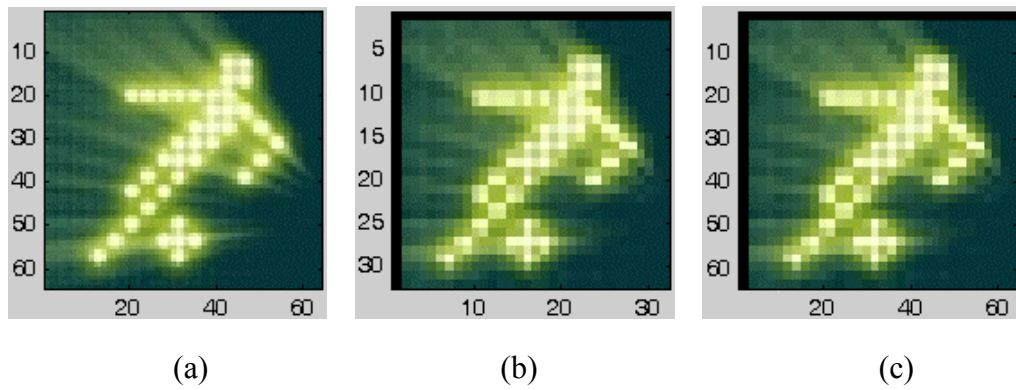


Figura 2.4. Ejemplo de compresión y descompresión aplicando muestreo uniforme: (a) imagen “soccer” original de 64x64 pixels, (b) imagen comprimida con razón de muestreo $k=2$, (c) imagen descomprimida.

Una ventaja de esta técnica es su simplicidad, que permite reducir la complejidad computacional y la alta razón de compresión. Una desventaja es el error de aproximación y la inclusión de artefactos en la imagen así como bordes muy marcados. Esta técnica puede ser útil en imágenes simples formadas por grandes zonas uniformes.

2.6. Algoritmos de eliminación de redundancia.

Una variación de la reducción del muestreo da lugar a los algoritmos de reducción de redundancia. Estos algoritmos exploran la imagen recorriéndola sobre la base de líneas de *scan* de manera que la línea de exploración resultante se diferencie de la original dentro de un determinado error. Entre estos algoritmos destacamos los llamados *fan-based-algorithms* [ROSE90] también llamados codificación lineal a tramos (PLC, *piecewise linear coding*).

El primer paso en un algoritmo PLC es definir el punto de inicio del segmento. Este punto puede ser el primero de la línea o bien el último punto del segmento anterior. A partir de este punto inicial se obtiene el segmento pixel a pixel. En cada paso se calcula el error entre la imagen original y la aproximación correspondiente al segmento considerando el punto actual como extremo de dicho segmento. Cuando se excede un umbral en el error se finaliza el segmento en el punto inmediatamente anterior al actual. De esta manera se garantiza que la nueva aproximación se encuentra dentro de unos límites de error respecto a los datos de la imagen original. El nuevo punto final del

segmento es el punto inicial del siguiente segmento. Este proceso continua hasta que la línea de exploración ha sido procesada completamente. Existen diversas variaciones y modificaciones sobre este método de aproximación. A continuación consideraremos algunas de estas aproximaciones.

La idea básica consiste en realizar la compresión de una función lineal a tramos de manera que el error de la nueva función comprimida esté acotado a un determinado valor ε que se prefija. Así dada una función lineal a tramos $F : [0,1] \rightarrow \mathfrak{R}^k$ especificada como la interpolación de N puntos y un error $\varepsilon \in \mathfrak{R}^k$, construir la función lineal a tramos G de manera que para todo $x \in [1, N]$, $\|F(x), G(x)\|_{\infty} \leq \varepsilon$ y G consiste en el menor número de segmentos que define esa función.

De acuerdo con esto para cada punto (x,y) que define la función F el algoritmo construye los puntos $(x,y+\varepsilon)$ y $(x,y-\varepsilon)$. Esto crea un error de túnel tal y como se muestra en la figura 2.5.

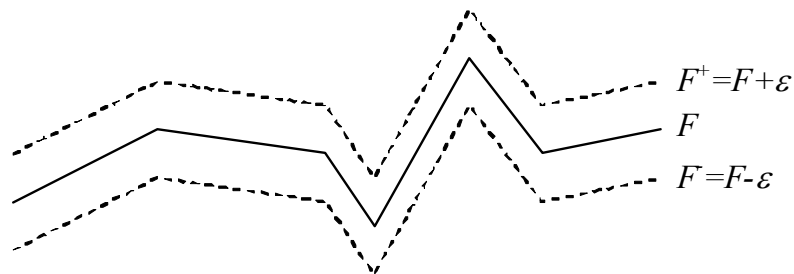


Figura 2.5. Error de túnel de la función F con error ε .

La función descrita en la figura 2.5 es una función escalar lo que da lugar a un túnel en dos dimensiones. En general la función F es un vector de k dimensiones con un error túnel de $k+1$ dimensiones. En el caso particular de una imagen estática en color $k=3$. A partir de ese error el algoritmo de compresión construye una nueva función lineal a tramos (G) que une los dos extremos del túnel con el menor número de tramos.

2.6.1. Aproximación por programación dinámica

En [SAUP98] se aplica programación dinámica para seleccionar el segmento más adecuado dentro de un umbral de error prefijado. Para ello se construye un grafo acíclico con todas las soluciones factibles. En dicho grafo los arcos son las soluciones factibles. Todos los arcos tienen etiqueta 1 por lo que dan una medida del número de segmentos para codificar la señal. La figura 2.6 muestra algunos ejemplos de arcos para soluciones factibles dentro del error de túnel marcado por las líneas discontinuas. Se observa que una solución requiere de 3 segmentos frente a los 8 de la gráfica original.

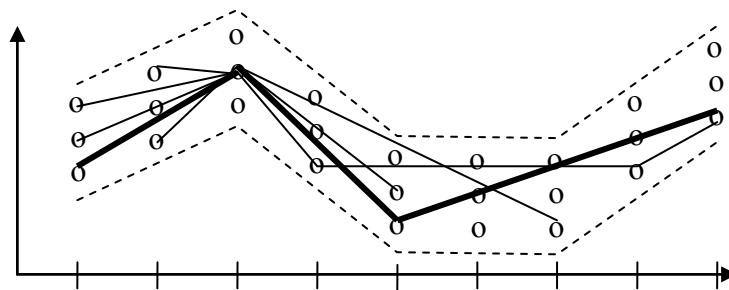


Figura 2.6. Aplicación de programación dinámica.

El principal inconveniente de este tipo de aproximación es la complejidad computacional. En el caso de que se empleen datos discretos con coordenadas enteras (que es el caso más común de codificación de imágenes) el número de segmentos candidatos es muy alto. Así para una imagen con N píxeles y un error ε el número de segmentos viene dado por:

$$\sum_{k=2}^N \binom{N-2}{k-2} (2\varepsilon + 1)^k$$

2.6.2. Algoritmo geométrico

En [BHAS93a,b] se presenta un algoritmo muy adecuado para aplicaciones en tiempo real así como su implementación hardware. El algoritmo se basa en un método voraz (*greedy algorithm*) que selecciona los elementos más prometedores del conjunto de candidatos hasta encontrar una solución. Este mecanismo de resolución voraz no suele dar soluciones óptimas. De hecho el número de segmentos que se obtiene es el doble del número óptimo.

La figura 2.7 muestra el esquema del sistema de compresión. Las secuencias $\{x'\}$ y $\{y'\}$ son codificadas mediante un codificador Huffman. La secuencia $\{y'\}$ no tiene que ser un subconjunto de $\{y\}$.

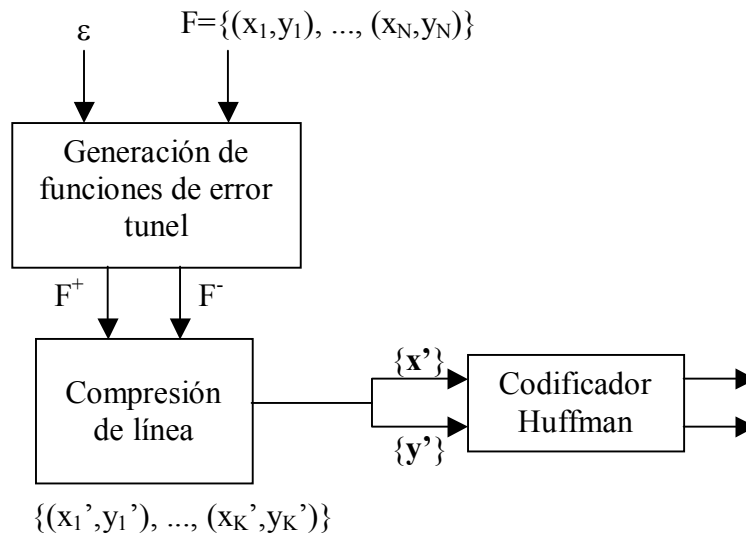


Figura 2.7. Esquema de compresión.

Una imagen bidimensional puede considerarse como una colección de líneas unidimensionales independientes. Sin embargo conviene explotar la correlación bidimensional de la imagen. Para ello el algoritmo de compresión y descompresión de imágenes propuesto en [BHAS93a,b] se ejecuta de la manera descrita en la figura 2.8 y 2.9 respectivamente. En el caso del algoritmo de compresión de imágenes de la figura 2.8 la imagen I es una matriz de $N \times N$ pixels. I_i corresponde la fila- i de la matriz, mientras que $I_i(j)$ corresponde al píxel (i,j) .

```

Compresión
input  $\varepsilon$ 
begin
   $i=0$ 
  while  $i \leq N$  do
    comprime  $I_i$  a la tolerancia  $\varepsilon$ .
    Sea  $J_i$  la forma descomprimida de  $I_i$ .
     $j=i+1$ ;

    while  $\sum_{l=1}^{l=N} |I_j(l) - J_i(l)| \leq \varepsilon N$  do
       $j=j+1$ ; /* la línea  $j$  es ignorada */
    end
  end

```

Figura 2.8. Pseudocódigo del algoritmo de compresión.

```

Descompresión
begin
  for each línea  $i$  comprimida do
    descomprime para obtener  $J_i$ ;
  end
  for each línea  $i$  no comprimida do
    encontrar las líneas comprimida más cercana;
    interpolar linealmente entre ellas para obtener  $J_i$ ;
  end
end

```

Figura 2.9. Pseudocódigo del algoritmo de descompresión.

Uno de los aspectos clave de este algoritmo es la compresión de una fila, es decir la obtención de una función G con el menor número de tramos.

El algoritmo de compresión de filas considera las secuencias de datos $\{x_i, f_i^+\}$ y $\{x_i, f_i^-\}$ para $i=1, 2, \dots, N$. Vamos a llamar s al índice del punto de partida en la etapa p del algoritmo. Partiendo del punto (x_s, f_s) se dibujan las tangentes a las envolventes de los límites superiores e inferiores para el punto con índice $k-1$. Dichas tangentes corresponden a las rectas $a_k x + b_k$ y $a_{k-1} x + b_{k-1}$ tal como se muestra en la figura 2.10. El algoritmo comprueba si es posible incluir el siguiente punto k analizando si se

cumple que f_k^+ está por encima de la tangente inferior y que f_k^- está por debajo de la tangente superior. Esto se realiza comprobando el signo de

$$S_1(k) = f_k^+ - (a_l x_k + b_l) = f_k^+ - a_l(x_k - x_s) - f_s$$

$$S_2(k) = f_k^- - (a_h x_k + b_h) = f_k^- - a_h(x_k - x_s) - f_s$$

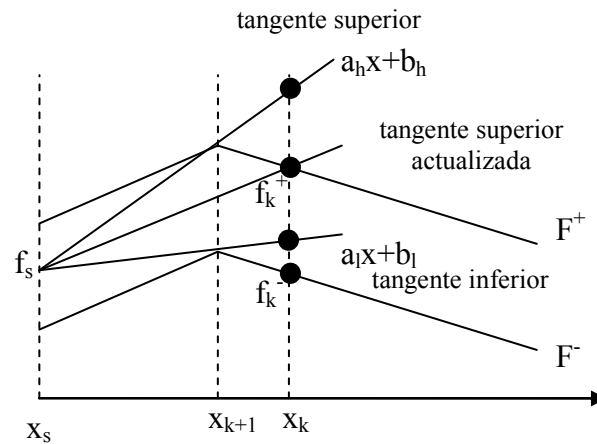


Figura 2.10. Cálculo de tangentes en el algoritmo geométrico.

Si algunas de las condiciones anteriores no se cumple entonces el algoritmo termina y la salida corresponde al punto $(x_{k-1}, g_p(x_{k-1}))$, donde

$$g_p(x_{k-1}) = a_l x_{k-1} + b_l = a_l(x_{k-1} - x_s) + f_s$$

En caso contrario el algoritmo continúa determinando si alguna de las tangentes debe ser actualizada. Las tangentes son actualizadas si f_k^+ está por debajo de la tangente superior o f_k^- está por encima de la tangente inferior. Por lo tanto se comprueban los signos de

$$S_3(k) = f_k^+ - (a_h x_k + b_h) = f_k^+ - a_h(x_k - x_s) - f_s$$

$$S_4(k) = f_k^- - (a_l x_k + b_l) = f_k^- - a_l(x_k - x_s) - f_s$$

Si el test del signo de algunas de estas funciones falla (por ejemplo f_k^+ está debajo de la tangente superior) entonces se calcula una nueva tangente.

En las ecuaciones anteriores está claro que los coeficientes de las rectas vienen dados por las expresiones siguientes:

$$a_h = \frac{f_{k-1}^+ - f_s}{x_{k-1} - x_s} \qquad b_h = f_s - \frac{f_{k-1}^+ - f_s}{x_{k-1} - x_s} x_s$$

$$a_l = \frac{f_{k-1}^- - f_s}{x_{k-1} - x_s} \qquad b_l = f_s - \frac{f_{k-1}^- - f_s}{x_{k-1} - x_s} x_s$$

2.7. Algoritmos propuestos

Vamos a describir a continuación dos algoritmos que proponemos y que intentan paliar algunos de los inconvenientes descritos en los anteriores. El criterio de diseño de estos algoritmos es que su implementación hardware sea de bajo coste y alta velocidad de procesado. De esta manera el campo de aplicación corresponde a sistemas que puedan ser empotrados dentro del propio sensor de visión y que puedan operar en tiempo real. La primera propuesta consiste en una modificación del algoritmo geométrico que ha dado lugar a tres tipos de implementaciones. La segunda propuesta consiste en aplicar un esquema de particionado del histograma y su codificación.

2.7.1. Modificación del algoritmo geométrico

Con objeto de mejorar el resultado de compresión nuestro mecanismo de exploración de la imagen no se realiza fila a fila sino que se aplica a toda la imagen. De esta manera se explora la imagen completa y se obtiene una forma de onda unidimensional. A continuación se aplica el esquema de compresión a la forma de onda de la imagen completa.

Vamos a describir a continuación varios refinamientos del algoritmo de compresión. El objetivo de diseño es reducir la complejidad computacional del

algoritmo con objeto de que el circuito resultante sea de bajo coste y opere a una alta velocidad de procesado.

Una primera simplificación consiste en asignar el mismo valor a los píxeles dentro del error de túnel. La figura 2.11 muestra un ejemplo de esta estrategia. En este ejemplo se describe la secuencia de píxeles $\{x_1, x_2, x_3, x_4, x_5\}$. Los valores que toman dichos píxeles corresponden a la secuencia $\{f_1, f_2, f_3, f_4, f_5\}$. El algoritmo de comprensión se inicia en el punto x_1 cuyo valor es f_1 . Para dicho punto se calculan los límites superior ($F^+=f_1+\varepsilon$) e inferior ($F^-=f_1-\varepsilon$) de acuerdo con un error ε . A continuación se evalúa el siguiente pixel (x_2). Si su valor se encuentra dentro de los límites del error ε se le asocia como nuevo valor f_1 . En caso contrario se toma dicho pixel como punto de inicio y se vuelve a iterar calculando los nuevos límites F^+ y F^- .

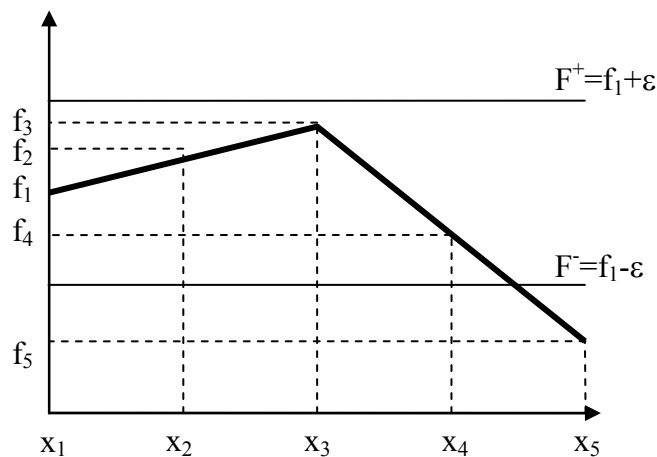


Figura 2.11. Ejemplo del algoritmo propuesto

Como resultado de aplicar este algoritmo la nueva secuencia de salida será $\{f_1, f_1, f_1, f_1, f_5\}$. Se trata de un algoritmo de comprensión con pérdidas en el que la secuencia de valores está acotada por el error ε de la aproximación. Con objeto de poder referenciar este algoritmo en la evaluación que realizaremos a continuación lo vamos a denominar Algoritmo Geométrico 1 (AG1).

El resultado de aplicar el algoritmo anterior es una imagen de $N \times M$ píxeles. En sí mismo no se ha realizado ninguna compresión pero se ha obtenido una imagen en la que el número de códigos (valores de los píxeles) se reduce frente a la imagen original. Por

ello la siguiente etapa en el proceso de compresión consiste en aplicar un esquema de codificación que permita comprimir la información.

Con objeto de mejorar la cantidad de información que se almacena se puede reducir la longitud de la cadena lineal almacenando para cada pixel el número de veces que se repite. En el caso del ejemplo anterior el resultado sería la secuencia:

$$\{(f_1,4), (f_5,1)\}$$

Para el caso de una imagen en la que los píxeles se codifican con 8 bits pasamos de requerir 5 bytes en el caso del algoritmo AG1 a 4 bytes en este caso (que denominaremos AG2).

La codificación del número de repeticiones de un pixel viene limitada por el tamaño de la palabra que estemos considerando (por ejemplo 8 bits en el caso de codificar cada pixel con 1 byte). Una alternativa que consiste en asignar un código para cada repetición del valor del pixel. De esta manera indicamos con 1 si el pixel se repite y con 0 si no se repite. En el ejemplo de la figura 4.9 el resultado será la cadena siguiente:

$$\{f_1,1110,f_5,0\}$$

El pixel con valor f_1 se repite 4 veces por lo que su código corresponde a la subcadena $\{f_1,1110\}$, mientras que el pixel con valor f_5 corresponde a la subcadena $\{f_5,0\}$. En este caso se ha comprimido la información ya que hemos pasado de 40 bits para representar la cadena original a 21 bits. Los bits de la cadena lineal que se obtiene de esta manera se agrupan en palabras de 8 bits. Por lo que en nuestro ejemplo realmente se requieren 3 bytes. De esta manera es posible aplicar una etapa de codificación posterior que realice una mayor compresión de la imagen. Vamos a denominar esta variación del algoritmo de compresión como AG3.

Con objeto de evaluar la calidad de los algoritmos de compresión vamos a compararlos de acuerdo con dos métricas: razón de compresión y error de la aproximación (mediante el RMSE). La figura 2.12 muestra las imágenes de test que se

han empleado. Se tratan de imágenes de 64x64 píxeles y de 128x128 píxeles. El resultado de evaluar la razón de compresión se muestra en la figura 2.13. Se han comparado los algoritmos AG1, AG2 y AG3 con el algoritmo descrito en [BHAS93a,b] que hemos denominado BNK (en referencia a sus autores).

En todos los casos se ha aplicado codificación Huffman como etapa final de la aproximación. Podemos observar que las soluciones BNK y AG1 ofrecen resultados similares. Ello justifica la simplificación propuesta en AG1 que permite reducir tanto los recursos de procesado como el tiempo de cómputo. Por otro lado las soluciones AG2 y AG3 también muestran un comportamiento similar entre ellas.

Respecto a la calidad de la aproximación se puede comprobar el buen comportamiento de las soluciones AG2 y AG3 ya que mejoran los resultados obtenidos en BNK y AG1.

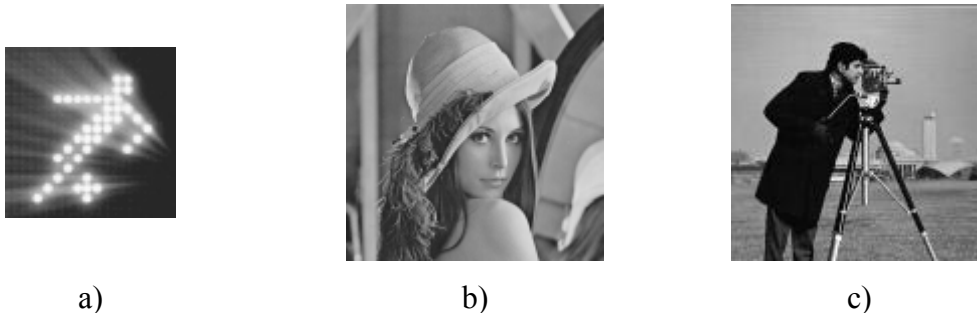
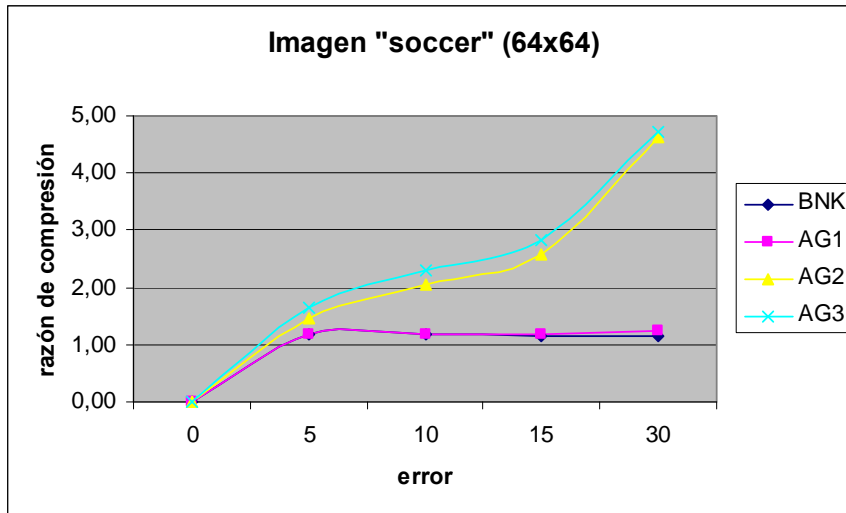


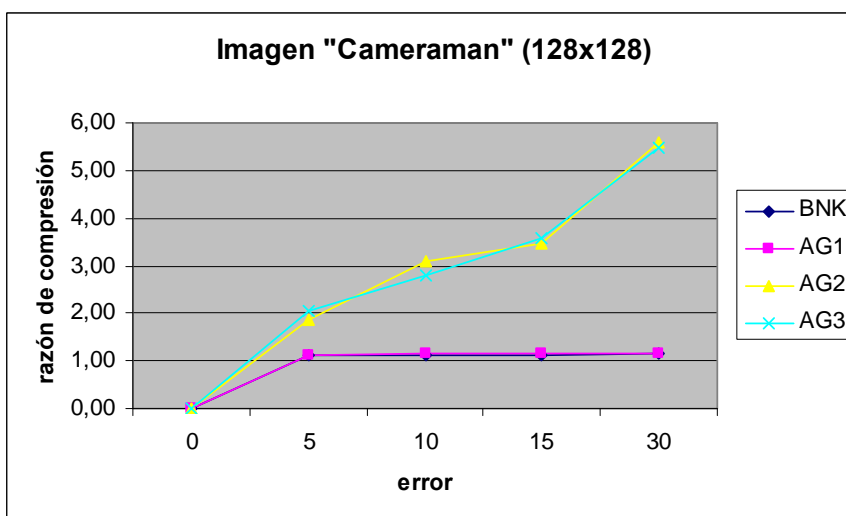
Figura 2.12. a) Imagen “soccer” de 64x64, b) imagen “Lena” de 128x128, c) imagen “Cameraman” de 128x 128.



a)



b)

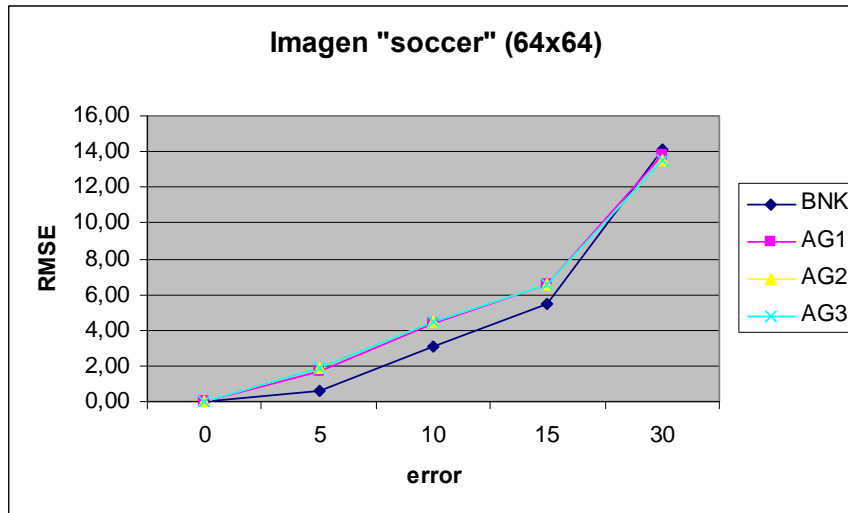


c)

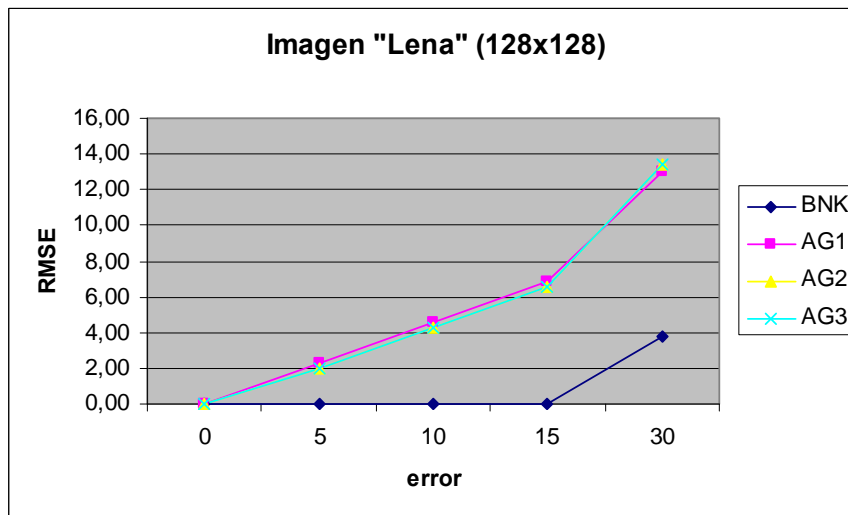
Figura 2.13. Comparación de algoritmos: razón de compresión.

Respecto a los errores en las aproximaciones hemos considerado la raíz del error cuadrático medio (RMSE) como medida de calidad. La figura 2.14 muestra los resultados obtenidos para las imágenes consideradas. Se observa que para valores de error de túnel pequeños (ε entre 1 y 15) los valores del RMSE son similares. Sin embargo el algoritmo BNK muestra un mejor comportamiento frente a las soluciones propuestas.

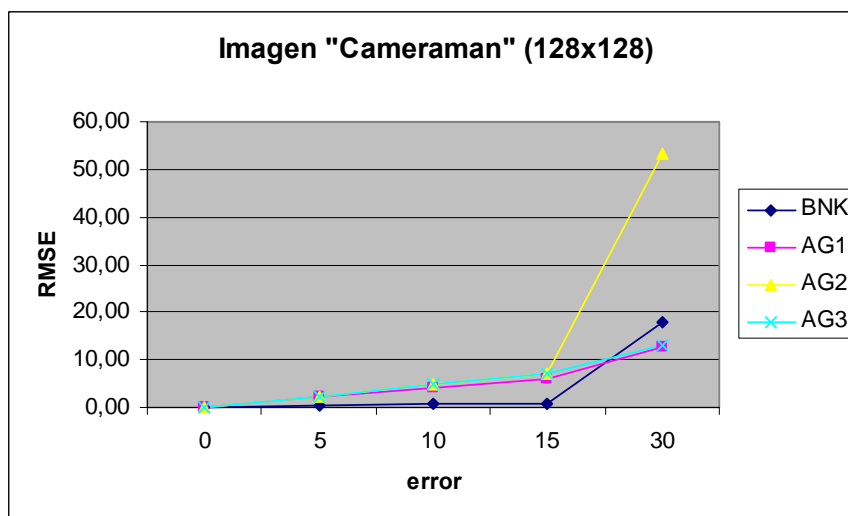
Con objeto de evaluar las prestaciones de los algoritmos de compresión hemos fijado un error de la aproximación y se han comparado las razones de compresión para las imágenes de test. La figura 2.15 muestra estos resultados. Se ha prefijado el valor de la raíz del error cuadrático medio para cada imagen: a) RMSE=0.6 para “soccer”, b) RMSE=0.7 para “lena” y c) RMSE=0.8 para “cameraman”. Esto nos permite determinar la capacidad de compresión de cada algoritmo. Puede observarse que el caso de AG3 ofrece los mejores resultados de compresión en los tres casos.



a)

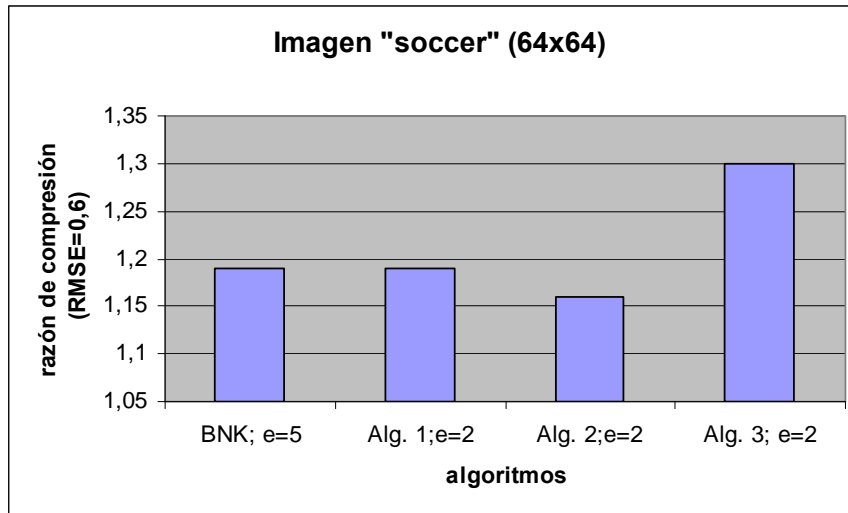


b)

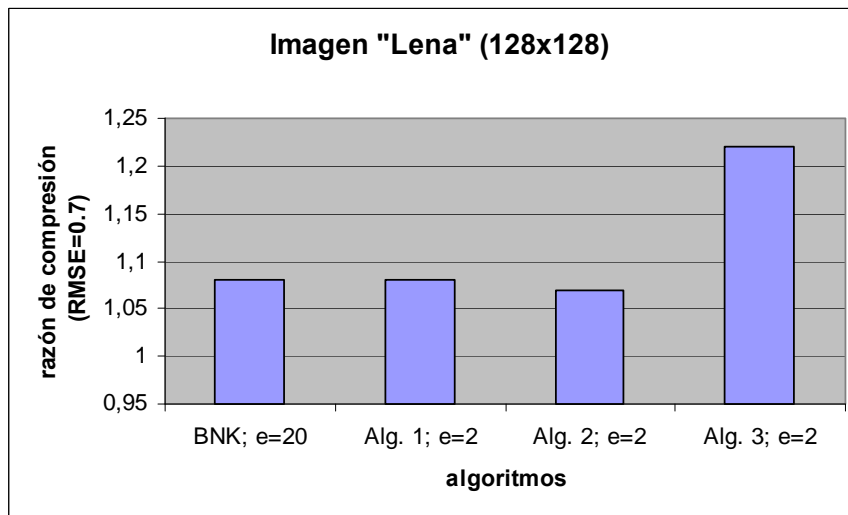


c)

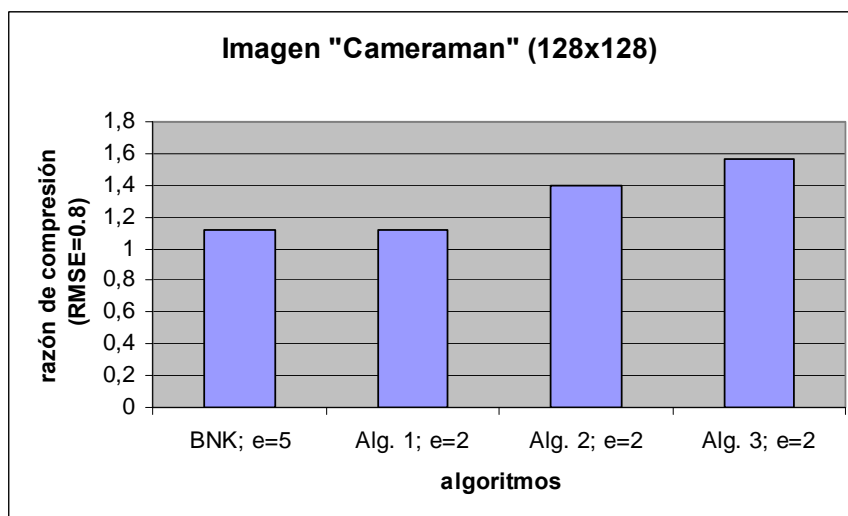
Figura 2.14. Comparación de algoritmos: RMSE.



a)



b)



c)

Figura 2.15. Comparación de algoritmos para un RMSE prefijado.

2.7.2. Particionado del histograma y codificación

El uso de técnicas neuro-difusas en la compresión de imágenes constituye una aplicación muy adecuada por la propia naturaleza del tipo de procesado que se realiza. De hecho la propuesta de este tipo de estrategias no es un hecho reciente [KONG91, STEU96]. La propia naturaleza de los algoritmos de razonamiento aproximado permite ajustar la precisión de las aproximaciones estableciendo compromisos entre la calidad de la imagen y la razón de compresión. Así en [HIRO99] se muestra cómo un incremento en la fuzzificación da lugar a mejores resultados de compresión si bien como contrapartida se obtiene menor calidad en la imagen comprimida.

Los buenos resultados que se han obtenido en la aplicación de la lógica difusa en la compresión de imágenes se ven empañados a la hora de su utilización práctica debido a la complejidad computacional de los algoritmos. Por ello nuestro interés se ha centrado en la realización de estrategias de compresión que permitan ser implementadas en hardware con un bajo coste de recursos, una alta velocidad de procesado y una adecuada relación entre calidad y razón de compresión.

Con objeto de simplificar la presentación vamos a considerar una imagen monocolor de $N \times M$ píxeles. La imagen se codifica con 8 bits por píxel lo que significa que se distinguen 256 tonos de grises. Por lo tanto el universo de discurso corresponde al rango entre 0 y 255. Dicho universo de discurso se divide en un conjunto de etiquetas lingüísticas que representan a conjuntos difusos. En nuestro caso hemos considerado conjuntos representados por funciones de pertenencia triangulares iguales, equiespaciadas y solapadas entre sí de acuerdo con el esquema de la figura 2.16. En el ejemplo mostrado en la figura 3.16 se han empleado 8 etiquetas.

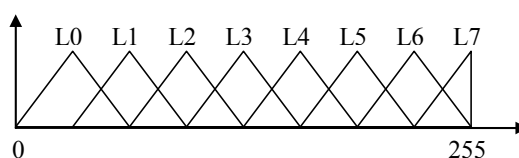


Figura 2.16. Funciones de pertenencia distribuidas en el universo de discurso de la variable que representa el color de un píxel.

El grado de fuzzificación de la imagen va a determinar tanto el grado de compresión como la calidad [HIRO99]. Así en el caso de aplicar un modelo *crisp* de la imagen se obtiene un mecanismo de compresión sin pérdidas ya que cada píxel está unívocamente codificado. Este caso extremo permite la mejor calidad. Una mayor compresión significa aumentar el grado de fuzzificación de la imagen (modelo *fuzzy*). En este caso varios píxeles pueden pertenecer a un conjunto difuso y tener, por lo tanto, el mismo código. Dependiendo de la granularidad del conjunto difuso tendremos mayor compresión (y mayor error en la imagen final).

Nuestra estrategia [BARR07a, b] se basa en considerar un esquema de fuzzificación en el que controlemos la razón de compresión modificando el número de bits que se requiere para representar cada píxel.

La codificación que se realiza de cada píxel se basa en particionar el universo de discurso en conjuntos difusos. En el ejemplo de la figura 2.16 se requieren 3 bits para codificar las etiquetas. El grado de pertenencia a cada etiqueta se codifica con 5 bits. De acuerdo con esto podemos realizar una representación exacta de la imagen al establecer un código único para cada píxel. En este caso el tipo de funciones de pertenencia que empleamos se ilustra en la figura 2.17.

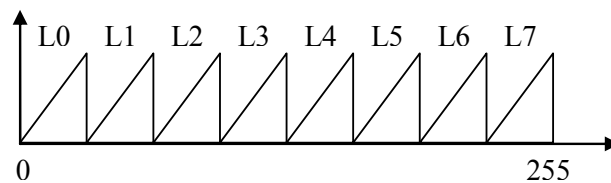


Figura 2.17. Representación exacta del universo de discurso

El tamaño de la imagen requiere $N \times M \times 8$ bits. Con objeto de reducir dicho tamaño (compresión) manteniendo la calidad original (compresión sin pérdida) aplicamos un algoritmo de codificación modificando y adaptando la técnica de codificación RLE (*Run Length Encoding*). Para ello vamos a considerar una imagen como un vector unidimensional de $N \times M$ elementos que corresponden a los píxeles. Cada uno de ellos se codifica mediante la pareja (*etiqueta, grado*). De esta manera cada píxel necesita 8 bits. Sin embargo es común que determinadas zonas de la imagen (píxeles adyacentes)

tengan valores comunes. Nuestra estrategia pretende aprovechar este hecho para reducir el número de bits necesarios ya que cuando los píxeles adyacentes corresponden a la misma etiqueta sólo se requiere especificar el grado de pertenencia como elemento distintivo. Este hecho se ilustra en el ejemplo mostrado en la figura 2.18.

La figura 2.18a muestra un ejemplo que corresponde a una cadena de 6 píxeles codificados con 8 bits cada uno. Los cinco primeros tienen como etiqueta el valor $L1$ y los grados de pertenencia corresponden a valores $g1$ (los tres primeros píxeles) y $g2$ (los dos siguientes). El último píxel de la cadena tiene como etiqueta el valor $L2$ y el grado de pertenencia es $g3$.

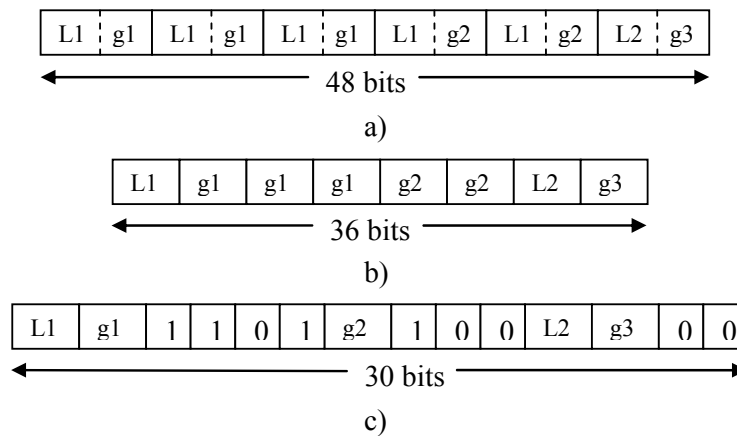


Figura 2.18. Ejemplo de codificación de la imagen.

Una primera aproximación que permite reducir el tamaño de la cadena es eliminar las etiquetas que se repiten en píxeles consecutivos. Así para los 5 primeros píxeles sólo tenemos que especificar la etiqueta $L1$ al comienzo de esa subcadena (figura 2.18b). En la figura 2.18b observamos que existe una redundancia en la información ya que hay repeticiones consecutivas en los grados de pertenencia. Así podemos apreciar que el grado $g1$ se repite tres veces consecutivas y el grado $g2$ se repite dos veces. Podemos aprovechar esa redundancia para optimizar la longitud de la cadena tal y como se ilustra en la figura 2.18c. En dicha figura la cadena comienza por especificar la etiqueta $L1$ y el grado de pertenencia $g1$. A continuación tenemos 2 bits que actúan de *flags* indicando la repetición del grado $g1$. Tras dos bits de control se indica el grado $g2$ y un bit de *flag* indicando su repetición. De esta forma en el ejemplo mostrado se ha reducido en un 37% la longitud de la cadena (de 48 bits a 30).

El esquema de la figura 2.19 muestra la codificación empleada. La subcadena de etiqueta está compuesta por un conjunto de campos que son los siguientes: 1) el campo etiqueta contiene el código de la etiqueta lingüística; 2) el campo grado contiene el código del grado de pertenencia; 3) el campo FCRG (*Flag* de Control de Repetición del Grado) indica con si el grado aparece un vez y con el valor '0' si ya no aparece; 4) el campo FCRL (*Flag* de Control de Repetición de etiqueta/*Label*) indica si la etiqueta se repite en el siguiente píxel. En caso de repetición de la etiqueta el esquema que continúa la secuencia corresponde a la subcadena de grado (FCRG), mientras que en caso contrario (un '0' en el bit FCRL) significa que el siguiente píxel corresponde a otra etiqueta lingüística por lo que se añadirá la subcadena de etiqueta.

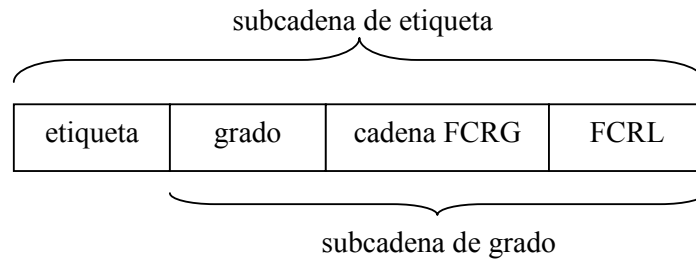
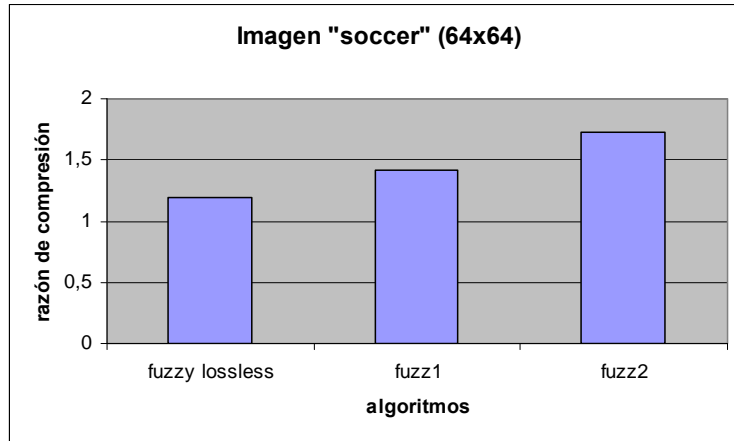


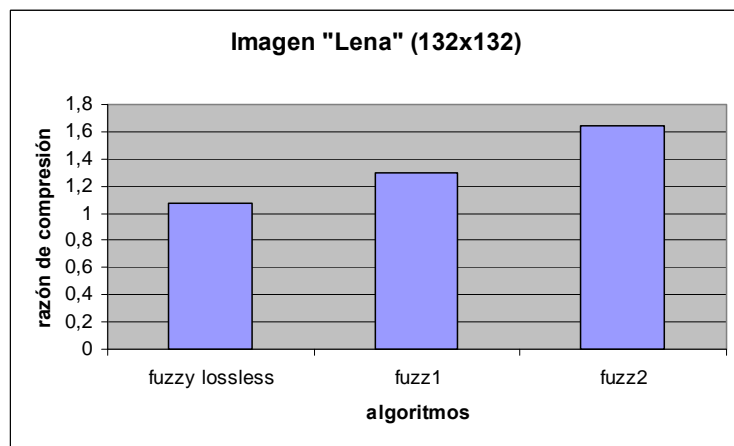
Figura 2.19. Esquema de codificación.

Los resultados obtenidos sobre la razón de compresión se ilustran en la figura 2.20. La aproximación *Fuzzy Lossless* corresponde al caso de compresión sin pérdidas mientras que las soluciones Fuzz1 y Fuzz2 son versiones de compresión difusa con distinta granularidad. Se puede observar que a medida que disminuimos la granularidad (pasamos del algoritmo *Fuzzy Lossless* al Fuzz1 y luego al Fuzz2) se aumenta la razón de compresión. De hecho el caso Fuzz1 es competitivo con JPEG mientras que Fuzz2 lo mejora.

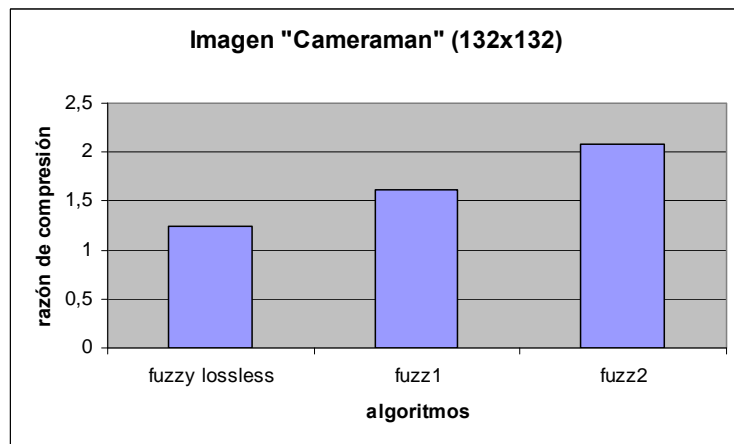
En lo que respecta a la calidad la figura 2.21 muestra los valores del error cuadrático medio para las diferentes imágenes de prueba. La técnica *fuzzy lossless* es una estrategia de compresión sin pérdidas por lo que el error de compresión es cero. Los casos Fuzz1 y Fuzz2 son algoritmos con pérdidas. Se observa que Fuzz1 presenta mayor calidad de la imagen comprimida ya que tiene un valor RMSE menor que Fuzz2.



a)

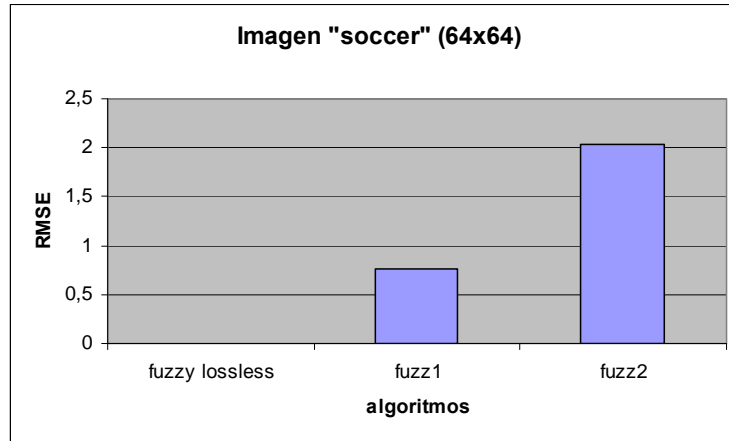


b)

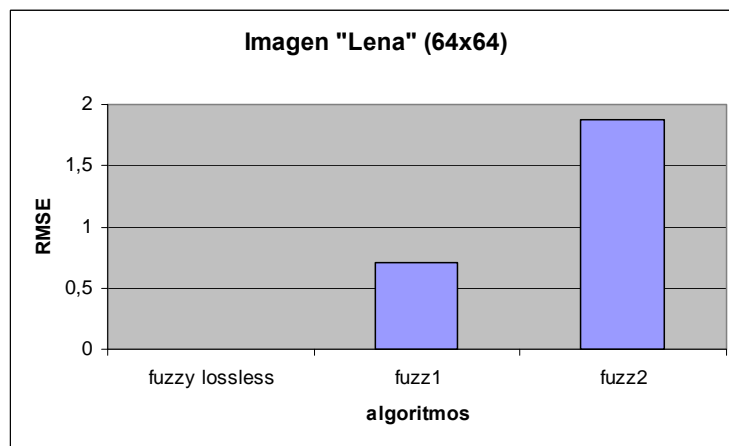


c)

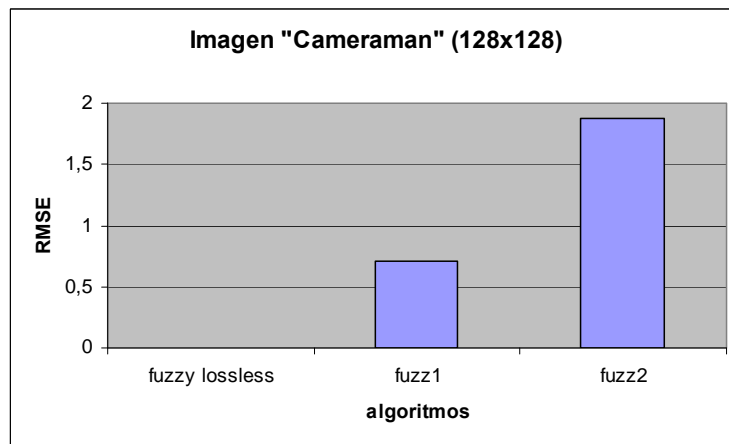
Figura 2.20. Resultados de la razón de compresión.



a)



b)

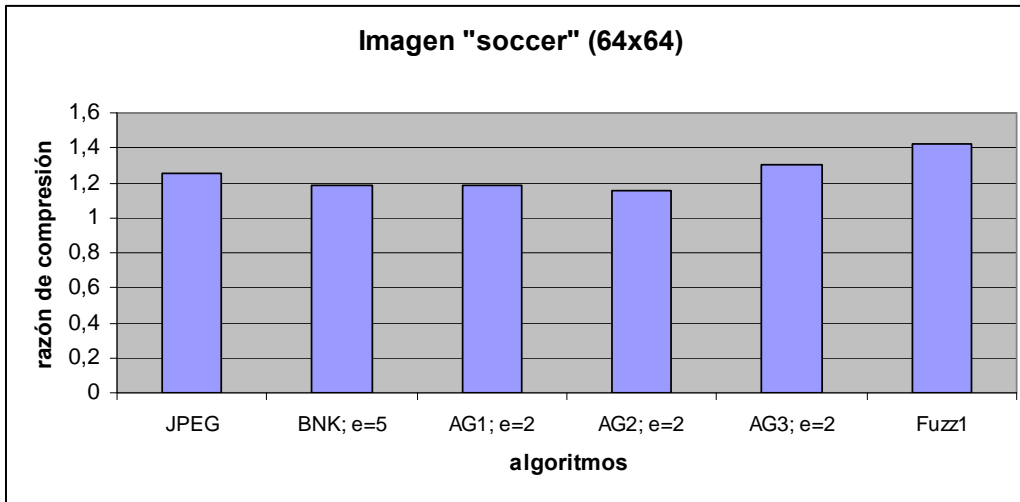


c)

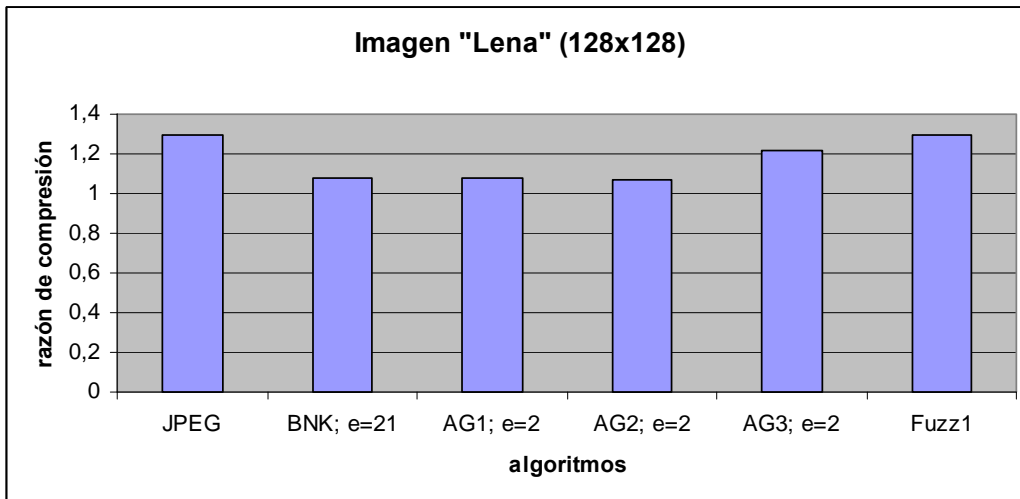
Figura 2.21. Resultados del valor de RMSE.

Un análisis que permita comparar las diferentes estrategias de compresión propuestas puede hacerse a partir de la figura 2.22 y 2.23. En dicha figura se especifican los valores de la razón de compresión y el RMSE para los diferentes algoritmos. Se ha

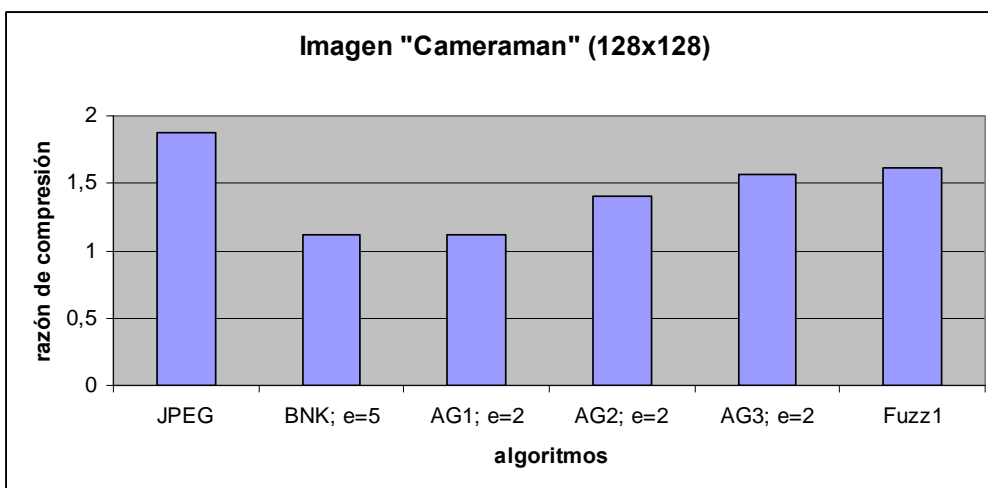
incluido además de los algoritmos AG1, AG2, AG3 y Fuzz1 los algoritmos BNK y JPEG. El algoritmo JPEG que se ha aplicado corresponde al que da lugar a una mejor calidad y por lo tanto un peor comportamiento en la compresión.



a)

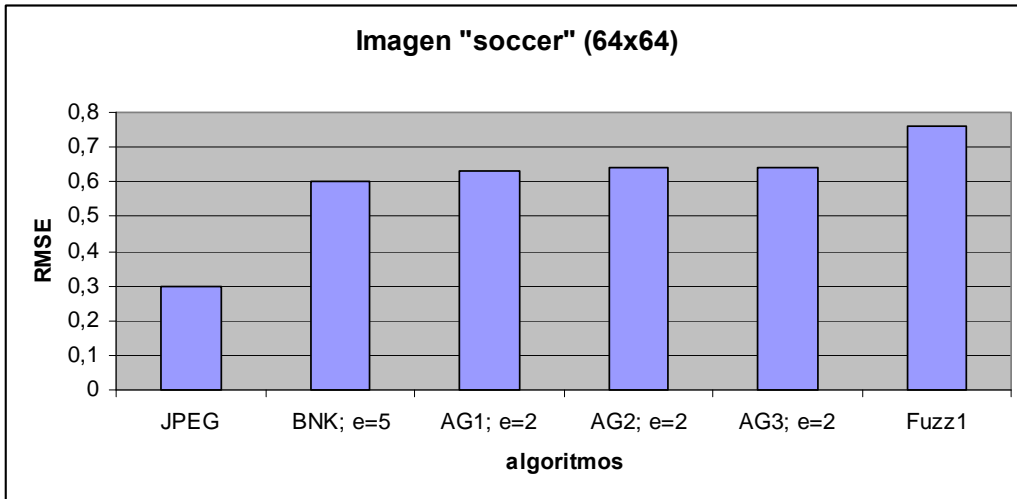


b)

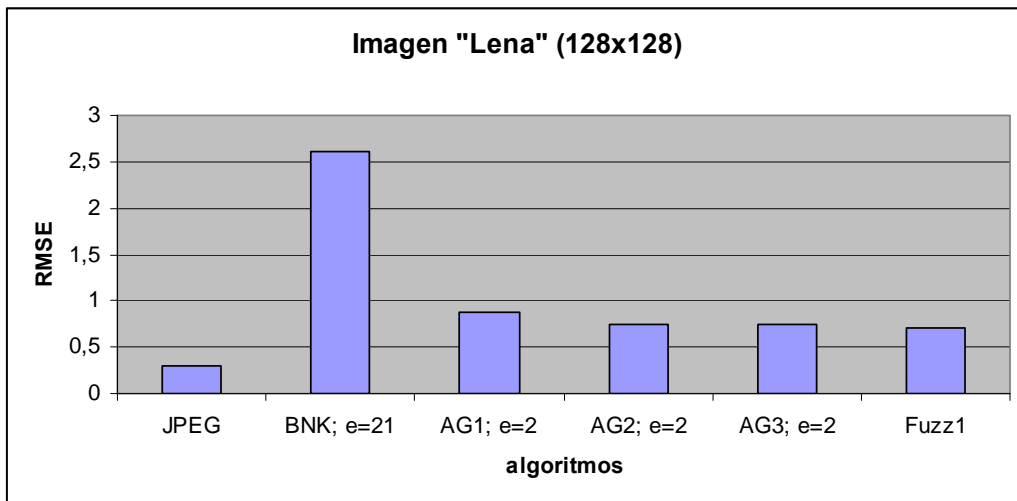


c)

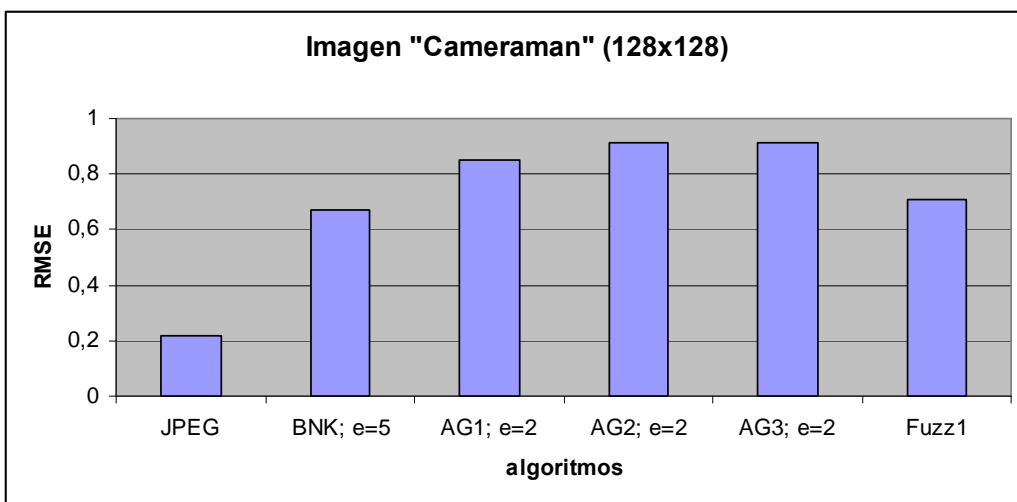
Figura 2.22. Comparación de los algoritmos de compresión en función de la razón de compresión.



a)



b)



c)

Figura 2.23. Comparación de los algoritmos de compresión en función del RMSE.

2.8. Diseño e implementación hardware de los algoritmos de compresión propuestos

A continuación vamos a realizar el diseño e implementación hardware de los algoritmos de compresión de imágenes que hemos propuesto. Los circuitos han sido implementados sobre FPGA con objeto de testar su funcionalidad y comprobar su operación. La implementación hardware del circuito de compresión ha sido realizada sobre dispositivos FPGA de bajo coste Spartan3 XC3S200TQ255 de Xilinx. El diseño se ha realizado a partir de la descripción VHDL de los módulos correspondientes al circuito de compresión y de descompresión. El flujo de diseño se ha basado en las herramientas de síntesis e implementación disponibles en el entorno de desarrollo ISE de Xilinx.

La arquitectura del circuito de compresión de imágenes está particionada en las tres etapas clásicas mostradas en la figura 2.1: transformación, cuantificación y codificación. La etapa de transformación en nuestro caso consiste en transformar una imagen bidimensional en una onda unidimensional. Para ello se realiza una exploración de la imagen por filas. La imagen se encuentra almacenada en una memoria o bien procede de un sensor de visión. En cualquier caso se lee cada píxel de la imagen de manera secuencial.

La etapa de cuantificación corresponde a una aproximación geométrica. Para ello se aplica uno de los algoritmos geométricos que hemos propuesto. Esta etapa genera una imagen en la que los píxeles han sido interpolados dentro de un error ε previamente predefinido. El diseño de esta etapa se describirá en el siguiente subapartado.

Finalmente la etapa de codificación consiste en aplicar uno de los esquemas de codificación propuestos basados en lógica difusa mediante un particionado del histograma de la imagen.

El circuito diseñado permite seleccionar entre cuatro estrategias de cuantificación y tres de codificación mediante determinadas señales de control. La tabla 2.1 muestra las diferentes configuraciones. Las señales Q_1 y Q_0 permiten seleccionar el algoritmo de cuantificación. El valor $Q_1Q_0=00$ no aplica ninguna cuantificación y se utiliza para el caso de compresión sin pérdidas. Por otro lado las señales C_1 y C_0 seleccionan el algoritmo de codificación.

Señales de control	Valores	Algoritmo de cuantificación o codificación
Q_1Q_0	00	Sin cuantificación
	01	Cuantificación AG1
	10	Cuantificación AG2
	11	Cuantificación AG3
C_1C_0	00	Codificación <i>Fuzzy Lossless</i>
	01	Codificación Fuzz1
	10	Codificación Fuzz2
	11	--

Tabla 2.1. Configuraciones del circuito de compresión de imágenes.

2.8.1. Diseño del circuito de compresión de los algoritmos geométricos

El núcleo común de los algoritmos geométricos propuestos es el circuito que contiene los elementos de interpolación basados en el error de túnel. Dicho esquema se ilustra en la figura 2.24. Básicamente está constituido por tres registros que contienen el error de túnel ε , el dato de referencia y el dato de entrada. La interpolación de la imagen se realiza a partir del cálculo de los límites definidos por

$$F_s^+ = x_s + \varepsilon$$

$$F_s^- = x_s - \varepsilon$$

Cada nuevo píxel de entrada se compara con los límites superior e inferior del túnel establecido para el valor de referencia x_s . Mientras el resultado de dicha comparación (C) se encuentre entre dichos límites la salida mantiene el valor de referencia. Cuando se sobrepasan los límites dados por el error ε se actualiza el nuevo valor de referencia x_s .

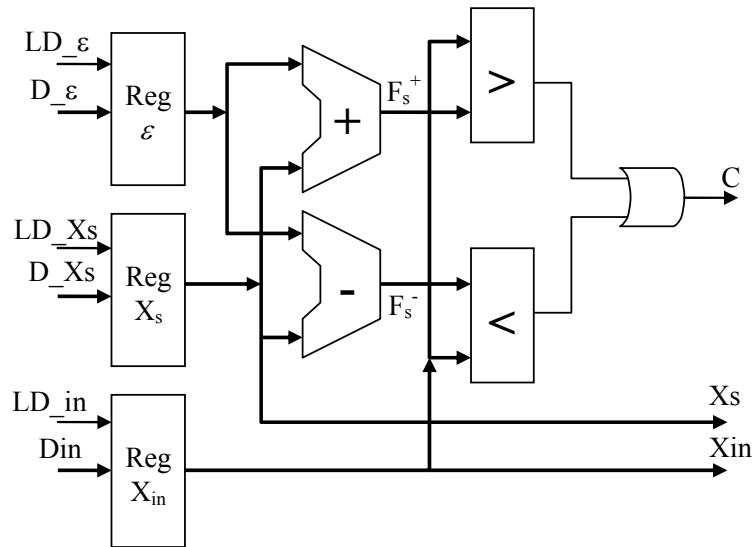


Figura 2.24 Circuito básico común de los algoritmos geométricos de compresión.

El circuito de la figura 2.24 es común para las diferentes alternativas AG1, AG2 y AG3. La diferencia entre éstas corresponde al mecanismo de generación de la salida. Así en el caso de la propuesta AG1 el esquema de la figura 2.25 muestra que requiere de un multiplexor de salida que seleccione el valor de x_s mientras el nuevo píxel se encuentre dentro del error de túnel o bien seleccione el nuevo valor x_{in} en caso contrario.

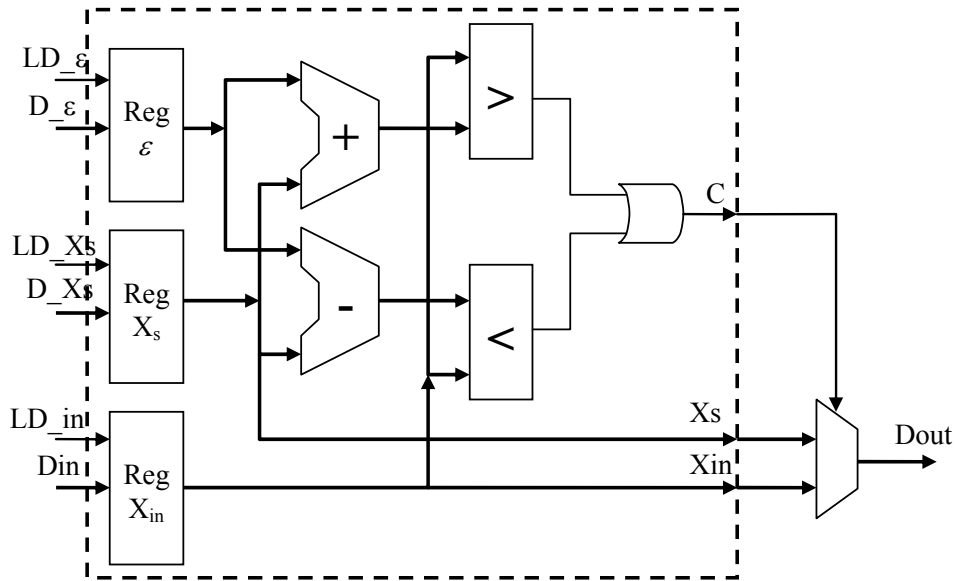


Figura 2.25. Circuito del algoritmo geométrico de compresión AG1.

El control del multiplexor de salida lo realiza la señal C que corresponde al resultado de la comparación. La unidad de control recibe como entrada dicho valor C y se encarga de actualizar el dato en el registro de referencia. La figura 2.26 muestra la carta ASM del algoritmo que implementa la unidad de control. Observamos que se trata de un sistema muy simple con dos estados (NOP y OP).

En la carta ASM del algoritmo de compresión AG1 se observa que el sistema recibe y genera otras señales de control. Así la señal *init* es una entrada que informa al sistema de la validez de los datos de una imagen de entrada. La validez de los datos de salida viene dada por la señal de salida *Dvalid*.

El algoritmo de compresión AG2 suministra el valor del píxel de referencia x_s junto con el valor de repetición de dicho valor. Para ello se requiere de un contador (ver figura 2.27) que se incrementa en cada ciclo mientras el valor del píxel de entrada se encuentre dentro del rango del error de túnel ε . La salida es una palabra de 16 bits que contiene el dato del valor de referencia x_s y el número de repeticiones de dicho dato.

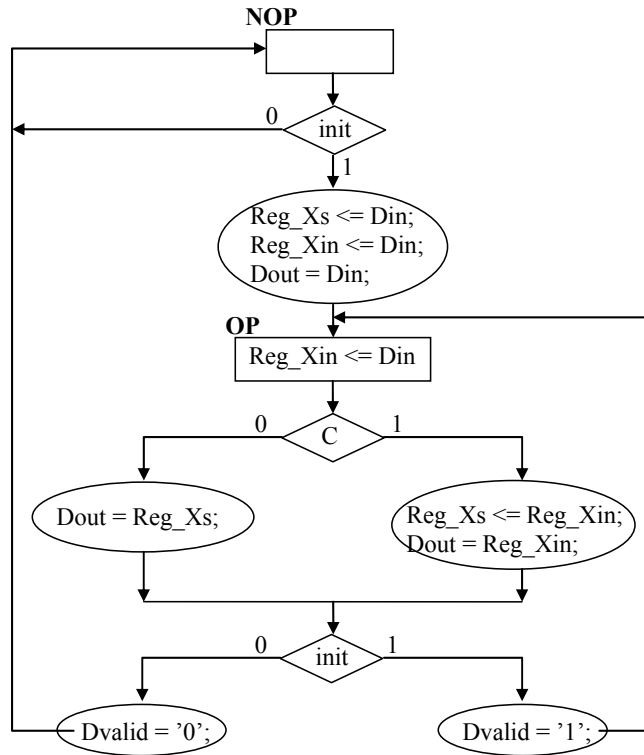


Figura 2.26. Carta ASM del algoritmo geométrico de compresión AG1.

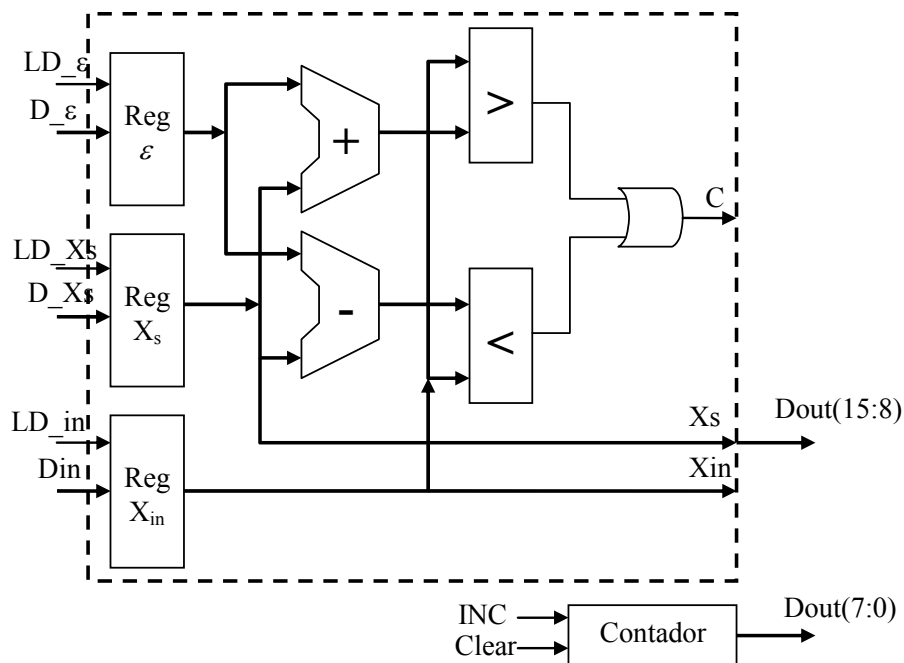


Figura 2.27. Circuito del algoritmo geométrico de compresión AG2.

En el caso del algoritmo de compresión AG3 la figura 2.28 muestra el circuito correspondiente. La salida es una secuencia de bits organizados en palabras de 8 bits. Dicha secuencia contiene el valor del píxel de referencia seguido de tantos '1' como repeticiones existan. Así cada vez que un nuevo píxel se encuentre dentro del túnel de error ε la salida lo indica con un '1'. Un código de final de repetición cuyo valor es '0' aparece cuando el nuevo píxel se encuentra fuera del rango del túnel.

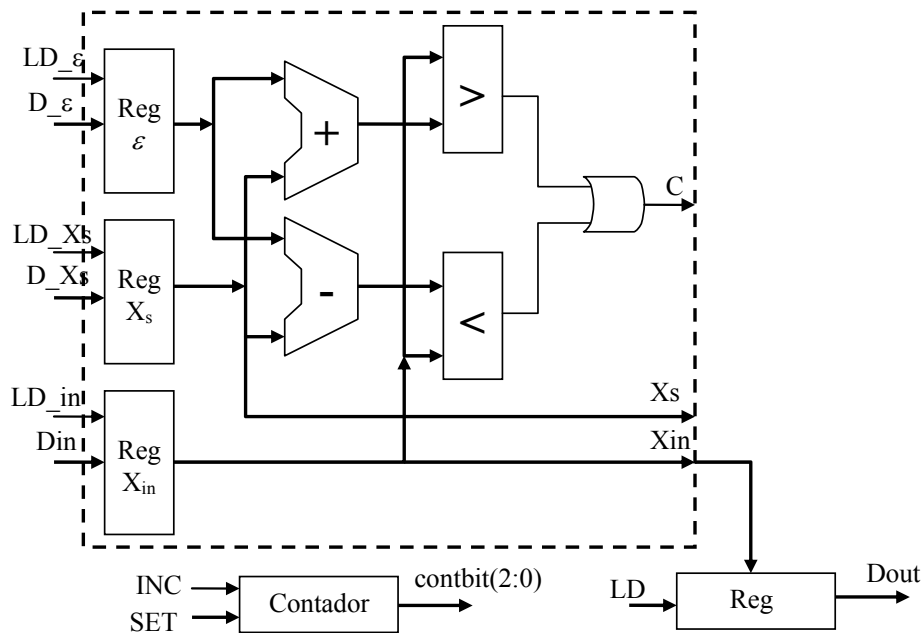


Figura 2.28. Circuito del algoritmo geométrico de compresión AG3.

En este caso el circuito de salida requiere de un contador de tres bits y un registro de salida. El contador actúa de puntero para escribir cada uno de los bits del registro de salida.

La figura 2.29 muestra el esquemático del circuito de compresión que incluye los tres algoritmos propuestos. El bloque UC corresponde a la unidad de control del sistema. Dicha unidad de control se ha diseñado como una máquina de estados finitos para generar las señales que controlan la operación del resto del sistema. El bloque UP corresponde al bloque común de los algoritmos (circuito de la figura 2.24).

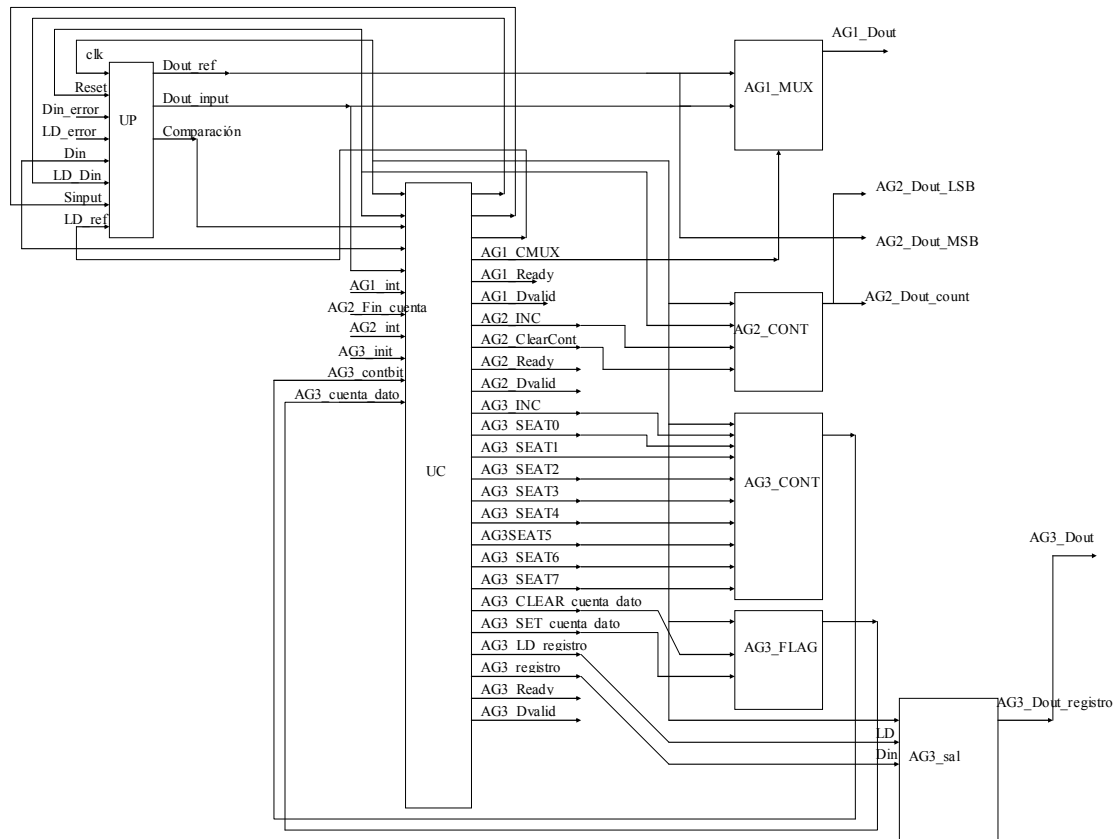


Figura 2.29. Esquemático del circuito de compresión.

2.8.2. Diseño del circuito de descompresión de los algoritmos geométricos

El proceso de descompresión consiste en recuperar la información de la imagen. En nuestro caso, el algoritmo AG1 no requiere ninguna operación de descompresión ya que la imagen comprimida suministra toda la información de cada píxel. En el caso de los algoritmos AG2 y AG3 es necesario recuperar la información de cada píxel por lo que se necesitan circuitos para realizar la descompresión de la imagen comprimida. El mecanismo de descompresión consiste en capturar la información de la imagen comprimida y realizar el cálculo de los valores de los píxeles. Por lo tanto el circuito de descompresión es una maquina de estados finitos (FSM) que realiza la operación de extracción y recuperación de la imagen cuantizada.

En el caso del algoritmo AG2 el circuito de descompresión recibe una palabra de 16 bits en el bus de la entrada “*Din*” que contiene el dato del valor del píxel (en los 8 bits más significativos) y el número de repeticiones de dicho dato (en los 8 bits menos significativos), y genera una palabra de 8 bits en el bus de la salida “*Dout*”. La señal “*init*” activa la operación del circuito. La salida “*Dout*” toma el valor de los 8 bits de la entrada *Din*(15:8) (valor del píxel) tantos ciclos como lo indique la entrada *Din*(7:0). La señal “*Dvalid*” indica que el dato de salida muestra un valor válido del píxel.

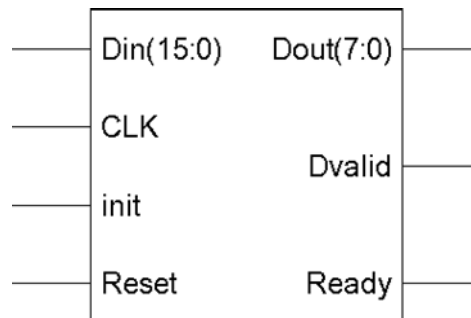


Figura 2.30. Símbolo del circuito de descompresión del algoritmo AG2.

El circuito de descompresión del algoritmo AG3 recibe una secuencia de bits organizados en palabras de 8 bits en el bus de la entrada “*Din*”. Dicha secuencia contiene el valor del píxel y los bits de control que definen las repeticiones del píxel. La FSM que extrae la información sólo requiere 5 estados. La figura 2.31 muestra la descripción algorítmica VHDL de dicha máquina y la figura 2.32 muestra el esquemático del circuito de descompresión. Las variables de estado son “actual” y “*proximo*”. La variable “dato” contiene el dato de entrada “*Din*”. La variable “*contbit*” corresponde a la salida de un contador que se utiliza para examinar cada uno de los bits del dato de entrada.

```

CC: process(actual,init,dato,contbit)
begin
  case actual is
    when NOP =>
      LD<="11111111"; registro <= "00000000";
      Dvalid_dummy <='0';
      Ready <= '0';
      INC <= '0'; Clear_cont<='0';
      if init='1' then
        proximo <= FIRST_DATA;
      else
        proximo <= NOP;
      end if;
    when FIRST_DATA =>
      LD<="11111111"; registro <= dato;
      Dvalid_dummy <='1';
      Ready <= '0';
      INC <= '0'; Clear_cont<='1';
      proximo <= OP;
    when OP =>
      LD<="00000000"; registro <= dato;
      if (dato(contbit)='1') then
        Dvalid_dummy <='1';
        proximo <= OP;
      else
        Dvalid_dummy <='0';
        proximo <= NEW_DATA;
      end if;
      INC <= '1'; Clear_cont<='0';
      if contbit=7 then
        Ready <= '0';
      else
        Ready <= '1';
      end if;
    when NEW_DATA =>
      if contbit=0 then
        Dvalid_dummy <='1';
        Ready <= '0';
        INC <= '0'; Clear_cont<='1';
        LD<="11111111"; registro <= dato;
        proximo<=OP;
      else
        Dvalid_dummy <='0';
        Ready <= '0';
        INC <= '0'; Clear_cont<='0';
        registro(7-contbit downto 0)<=dato(7 downto contbit);
        LD(7-contbit downto 0)<=(others=>'1');
        LD(7 downto 8-contbit)<=(others=>'0');
        proximo<=NEW_DATA2;
      end if;
    when NEW_DATA2 =>
      Dvalid_dummy <='1';
      Ready <= '1';
      INC <= '0'; Clear_cont<='0';
      registro(7 downto 8-contbit)<=dato(contbit-1 downto 0);
      LD(7 downto 8-contbit)<=(others=>'1');
      LD(7-contbit downto 0)<=(others=>'0');
      proximo<=OP;
  end case;
end process;

```

Figura 2.31. Descripción algorítmica VHDL de la FSM del algoritmo de descompresión AG3.

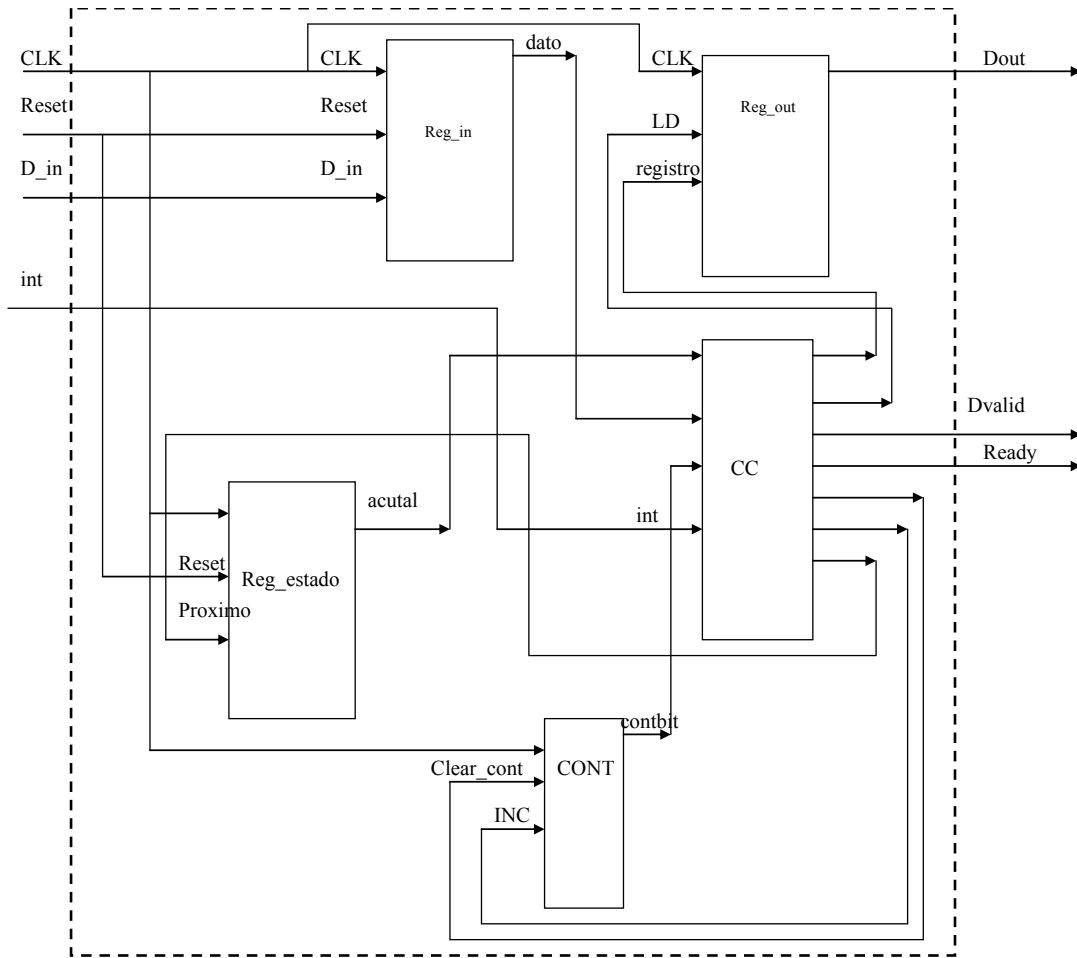


Figura 2.32. Esquemático del circuito de descompresión AG3.

La figura 2.33 muestra el esquema de bloques del circuito de descompresión. Dicho esquema contiene las FSM correspondientes a los algoritmos AG2 y AG3 así como las señales de control y selección correspondientes.

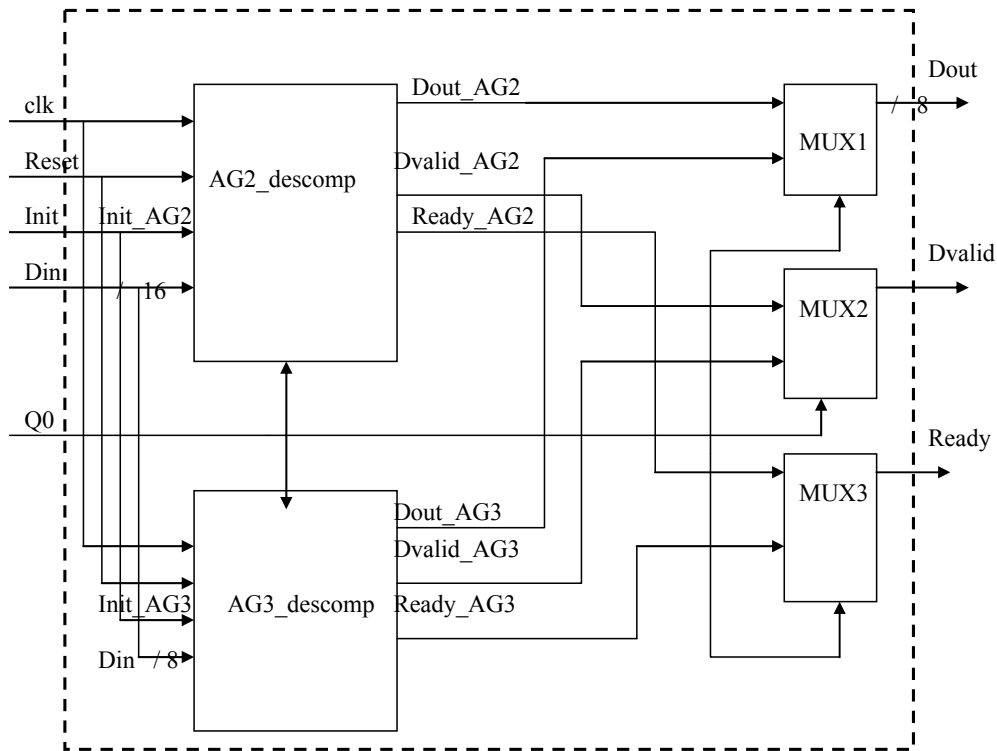


Figura 2.33. Esquema de bloques del circuito de descompresión.

2.8.3. Circuito de compresión/descompresión de los algoritmos geométricos

En este apartado vamos a describir el circuito completo de compresión/descompresión de los algoritmos geométricos propuestos. También mostraremos algunos resultados de implementación de dichos circuitos sobre FPGA. La figura 2.34 muestra el símbolo del circuito y el significado de las señales de entrada/salida.

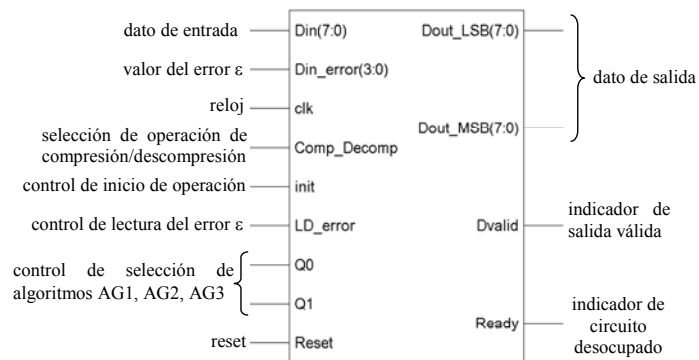


Figura 2.34. Símbolo del circuito de cuantización

El sistema de cuantización se ha implementado sobre FPGA de Xilinx Spartan3 haciendo uso de las herramientas de síntesis (XST) e implementación (XFlow) del entorno de diseño ISE de Xilinx. Los resultados de la implementación se muestran en la tabla 2.2.

	Compresión	Descompresión
AG1	26	--
AG2	34	20
AG3	123	57
Completo (AG1, AG2, AG3)	147	86

Tabla 2.2. Coste de las implementación en términos de *slices* ocupados/.

Como se muestra en la tabla el área ocupada por el sistema completo de los algoritmos de compresión ha sido de 147 *slices* lo que supone un 7% de ocupación del dispositivo FPGA seleccionado. El tamaño en número de puertas equivalentes es de 2351 puertas. En el caso del sistema completo de los algoritmos de descompresión, el área ocupa ha sido de 86 *slices* lo que supone un 4% de ocupación del dispositivo FPGA seleccionado. El tamaño en número de puertas equivalentes es de 1344 puertas.

2.8.4. Diseño del circuito de compresión del algoritmo de particionado del histograma y codificación

La etapa de codificación se ha realizado implementando los esquemas que hemos denominado *Fuzzy Lossless*, *Fuzz1* y *Fuzz2*. El esquema del sistema se ilustra en la figura 2.35. Los bloques compresión y descompresión leen la imagen secuencialmente por el bus de entrada *Din* y generan las salidas en el bus *Dout*. La selección de uno de los bloques se realiza con la señal *codec*. El sistema permite tres niveles de codificación que se selecciona con las señales de control C_1C_0 .

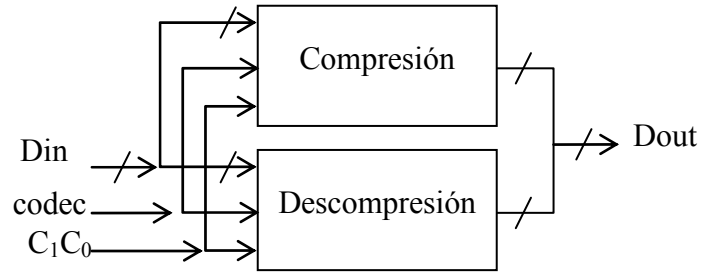


Figura 2.35. Esquema del sistema de compresión/descompresión.

Los bloques compresión y descompresión se han diseñado mediante maquinas FSM que realizan el algoritmo descrito en el apartado anterior. La figura 2.36 muestra el esquemático del circuito de compresión. El algoritmo de la FSM del circuito de compresión de imágenes se ilustra en la figura 2.37.

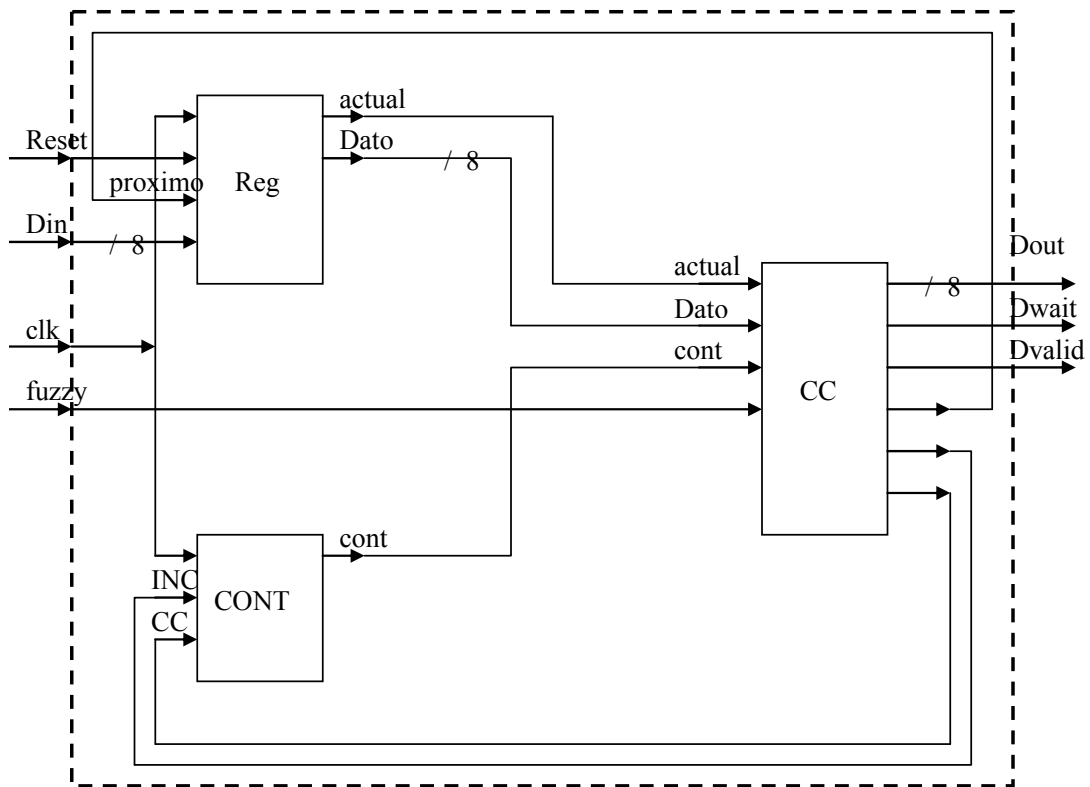


Figura 2.36. Esquemático del circuito de compresión.

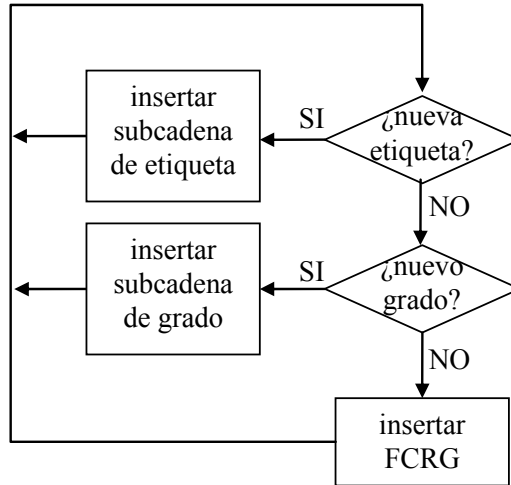


Figura 2.37. Algoritmo de compresión.

La selección de la granularidad de la compresión de la imagen es programable pudiéndose seleccionar entre las aproximaciones *Fuzzy Lossless*, *Fuzz1* y *Fuzz2*. El circuito recibe la imagen de manera secuencial y procesa un pixel en cada ciclo de reloj. De esta forma el tiempo que se requiere para comprimir una imagen dependerá del tamaño de la misma. La tabla 2.3 muestra los tiempos de compresión necesarios para imágenes de diferente tamaño. La frecuencia de operación del circuito es de 100 MHz.

	<i>Fuzzy lossless</i>		Fuzz2	
	compresión	descompresión	compresión	descompresión
Soccer (64x64)	125.4	100	108	85
Peppers (115x115)	435.7	344.5	355.2	281
Lena (222x208)	1466.2	1166.7	1250.4	990.5
Cameraman(256x256)	2016.8	1602.5	1509.8	1226.7
Nosveo (389x433)	4481	3606.8	2850.9	2451.8
Goldhill (512x512)	8630.7	6800.6	6984.9	5517.6

Tabla 2.3. Tiempo de compresión y descompresión de imágenes (en μseg) para una frecuencia de 100 MHz.

Básicamente las operaciones que se necesitan consisten en comparaciones, desplazamientos y contadores. Ello da lugar a un circuito que requiere de pocos recursos de procesado. El área ocupada por el sistema completo ha sido de 330 *slices* lo que

supone un 17% de ocupación del dispositivo FPGA seleccionado. El tamaño en número de puertas equivalentes es de 4377 puertas.

La implementación del sistema para imágenes en color se ha realizado sobre un dispositivo FPGA XC2V1500. El circuito ha ocupado 630 *slices* y 51 IOBs que suponen un total de 8709 puertas equivalentes. La frecuencia de operación máxima ha sido de 158 MHz. Por otro lado se ha implementado el compresor JPEG de [OPEN]. Dicho circuito ha ocupado un área de 6105 *slices*, 13 BRAM, 2 multiplicadores de 18x18 bits y 77 IOBs que suponen un total de 970707 puertas equivalente. La frecuencia máxima de operación ha sido de 26 MHz.

Podemos observar que nuestra propuesta presenta un menor coste en área ya que dicho circuito supone un 0,6% del tamaño del compresor JPEG. Por otro lado se multiplica por 6 la frecuencia de operación lo que da lugar a menores tiempos requeridos para la compresión de imágenes.

2.9. Resumen

En este capítulo nos hemos planteado el problema de la compresión de imágenes. Se han propuesto diversas soluciones a dicho problema. Algunas de estas soluciones se basan en algoritmos geométricos de interpolación de funciones lineales a tramos. Otras técnicas se basan en un particionado del universo de discurso del histograma y su correspondiente codificación.

Se ha realizado una implementación hardware de un circuito de compresión que incorpora las diferentes técnicas. Así dicho circuito está constituido por dos etapas: cuantificación y codificación. La primera etapa utiliza las técnicas de interpolación lineal a tramos mientras que la segunda etapa emplea el sistema de particionado y codificación. El circuito resultante es programable y permite realizar compresión con pérdidas y compresión sin pérdidas.

Los circuitos han sido implementados sobre dispositivos FPGA de bajo coste Spartan3 de Xilinx con objeto de testar su funcionalidad y comprobar su operación. El

diseño se ha realizado a partir de la descripción VHDL de los módulos correspondientes al circuito de compresión y de descompresión. El objetivo del diseño ha sido reducir la complejidad computacional de los algoritmos que hacen la compresión de imágenes de forma que el circuito resultante es de bajo costo y opera a alta velocidad de procesado.

Capítulo 3

Control del contraste en imágenes

Los sistemas sensoriales humanos (visión, audición, olfato, gusto, tacto, temperatura) se organizan para responder fuertemente a los cambios temporales y espaciales en la energía física del estímulo. Cuando hay un cambio temporal en la energía aplicada al sensor (ojo, oído, nariz, lengua, piel) hay al principio una respuesta fuerte. A continuación los sentidos se adaptan rápidamente (responden menos) al uso constante y continuado de la energía.

Nuestro sistema visual satisface dos tipos de exigencias: en primer lugar ver tanto con iluminaciones débiles como con iluminaciones muy brillantes (tener un perceptivo) y en segundo lugar discriminar la diferencia existente entre dos objetos que reflejan intensidades lumínicas muy próximas entre sí. Así, podemos ver con niveles de iluminación tan bajos como los existentes cuando estamos completamente adaptados a la oscuridad o con niveles altos como cuando el sol se refleja en la nieve. También podemos discriminar entre dos objetos que se diferencien en menos del 1% de la cantidad de luz que reflejan. Para resolver estas situaciones el sistema visual dispone de dos mecanismos. El primero es la adaptación rápida, en la que la retina cambia su rango operativo (rango de intensidad lumínica) unas tres décimas de segundo después de producirse el cambio en el nivel lumínico. El segundo mecanismo es el de adaptación

local, por medio del cual partes diferentes de la retina se adaptan a niveles de iluminación diferentes.

La luminancia describe la energía del estímulo más bien que cambios de la energía, así que no es bastante por sí mismo. El concepto del “contraste de la luminancia” o simplemente “contraste” fue desarrollado con el propósito de describir los cambios de la energía. Existen muchas propuestas de medida del contraste. Básicamente el contraste puede definirse como el cambio de la luminancia relativa de los elementos de una imagen. Por lo tanto corresponde a la diferencia de luminancia que existe entre dos puntos de una imagen. El histograma de la imagen es una herramienta útil para examinar el contraste en la imagen [CHEN06].

Nuestro interés en este capítulo se centra en describir un mecanismo del control del contraste. Esta técnica se basa en la aplicación de los operadores del álgebra de Lukasiewicz (suma-acotada y producto-acotado) con objetivo de realizar una simplificación del diseño de los circuitos que controlen el contraste en las imágenes (circuitos con bajo coste y una alta velocidad).

Este capítulo se organiza en seis apartados. En el primer apartado se muestran las técnicas del control del contraste. En el apartado siguiente se presenta una breve introducción al álgebra de Lukasiewicz. En el apartado tres se describe la técnica de control del contraste mediante los operadores de Lukasiewicz. El diseño de dichos operadores es tratado en el siguiente apartado. A continuación se describe un refinamiento del algoritmo de control aplicando lógica difusa. Finalmente se presenta otra aplicación de los operadores de Lukasiewicz para la aproximación lineal a tramos de funciones.

3.1. Técnicas de control del contraste en imágenes

Una definición de contraste es el contraste de Weber y es la más comúnmente empleada del contexto de la iluminación. Consiste en la diferencia entre dos luminancias dividido por la luminancia más baja.

$$C = \frac{L_{\max} - L_{\min}}{L_{\min}}$$

Otra definición de uso frecuente en fotografía es la de contraste simple. En este caso se especifica la diferencia entre las partes brillantes y oscuras del cuadro. Esta definición no es útil para las luminancias del mundo real debido a su gama dinámica mucho más alta y las características logarítmicas de la respuesta del ojo humano.

$$C = \frac{L_{\max}}{L_{\min}}$$

Otra definición de contraste es el contraste de pico a pico o contraste de Michelson que mide la relación entre la extensión y la suma de las dos luminancias. Esta definición se utiliza típicamente en teoría del procesamiento de señal, para determinar la calidad de una señal respecto a su nivel de ruido. En el contexto de la visión tal ruido se podría causar por la luz dispersada introducida en la trayectoria de la visión por un elemento translúcido que oscurece en parte la escena detrás de él.

$$C_M = \frac{L_{\max} - L_{\min}}{L_{\max} + L_{\min}}$$

Otro tipo de medida del contraste de una imagen puede ser la varianza que viene dada por la expresión siguiente:

$$\sigma^2 = \frac{1}{MN} \sum_{k=1}^L (k - \bar{k})^2 n_k$$

Donde M y N son el tamaño de la imagen, k es el valor de luminancia que pertenece al rango de valores desde 1 a L , n_k es la frecuencia del nivel de luminancia k , \bar{k} es el valor medio de la distribución de luminancias,

$$\bar{k} = \frac{1}{MN} \sum_{k=1}^L kn_k$$

Cuando todos los píxeles presentan el mismo nivel de gris su varianza es cero y cuando mayor diferencia hay entre todos los posibles pares de píxeles entonces la varianza es mayor.

Por otra parte los valores $\{p_k = n_k/MN; k=1,2,\dots,L\}$ constituyen una distribución de probabilidad sobre el conjunto de los tonos de luminancia pues $\sum_{k=1}^L p_k = 1$. Se puede utilizar la entropía como una medida de contraste [KHEL91]:

$$H = - \sum_{k=1}^L p_k \ln p_k$$

Cuando la distribución de los tonos de luminancia de los píxeles es uniforme ($p_k = 1/L$) entonces la entropía alcanza su valor máximo (que es $\ln(L)$) que corresponde a una imagen con máximo contraste. Esto sugiere que una medida normalizada en el intervalo $[0,1]$ del contraste de una imagen sea $H/\ln(L)$. Obsérvese que la entropía es también una medida de incertidumbre que cuando vale cero corresponde a máxima información y al mismo tiempo mínimo contraste, mientras que para una imagen con distribución uniforme, que corresponde al máximo contraste, la incertidumbre o falta de información es también máxima.

Debido al proceso de digitalización de imágenes los píxeles están codificados por un número limitado de bits. Así por ejemplo en el caso de 8 bits para imágenes monocromáticas supone distinguir entre 256 niveles de gris. Si la gama de la variación en el brillo de la imagen es mucho más pequeña que la gama dinámica de la cámara entonces la gama real de números será mucho menor que la gama completa de 0 a 255. Es decir, la imagen obtenida en la salida de los sensores de la cámara no tiene porqué

cubrir la gama completa. En muchas situaciones la imagen registrada tendrá una gama mucho más pequeña de los valores del brillo. Estos valores pueden encontrarse en la zona media de la gama (valores grises intermedios) o hacia los extremos brillante u oscuro de la gama.

La visibilidad de los elementos que forman una imagen puede ser mejorada estirando el contraste con objeto de reasignar los valores de píxeles para cubrir la gama disponible entera. Esto significa que los píxeles se interpolan entre los valores extremos del rango dinámico.

Un mecanismo usual de mejora del contraste consiste en realizar una interpolación lineal [CHEN06, KIM99, KIM01, CHO00]. Esta técnica de expansión lineal del contraste permite incrementar la discriminación visual y resulta útil cuando la imagen tiene variaciones de luminancia que permiten discernir entre los elementos que la forman. Una manera sencilla consiste en aplicar la función mostrada en la figura 3.1. En este caso el valor del nuevo píxel viene dado por la siguiente expresión:

$$g(i,j) = a + b * f(i,j)$$

Donde, los valores a y b se calculan de acuerdo con la figura 3.1. Esta transformación aumenta el contraste de la imagen pues estira los valores de los tonos de luminancia de la imagen hasta ocupar el rango completo.

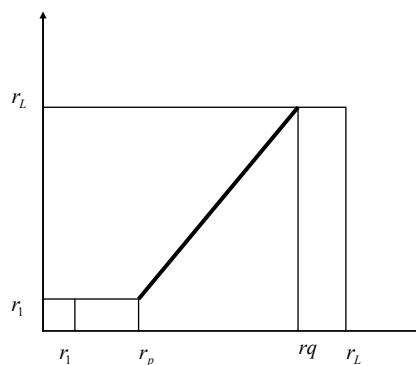


Figura 3.1. Transformación lineal

Observamos que si $f(i,j) = r_k$ entonces el nuevo valor de luminancia del píxel (i,j) , que representaremos por s_k viene determinado por la expresión siguiente:

$$s_k = \frac{r_l - r_1}{r_q - r_p} (r_k - r_p) + r_1$$

Un tipo de transformación un poco más general que la anterior para mejorar el contraste es la siguiente:

$$v = \begin{cases} \alpha u & \text{si } 0 \leq u \leq a \\ \beta(u - a) + v_a & \text{si } a \leq u < b \\ \gamma(u - b) + v_b & \text{si } b \leq u < L - 1 \end{cases}$$

Donde u representa el tono o nivel de luminancia de la imagen original y v el nivel de luminancia de la imagen transformada.

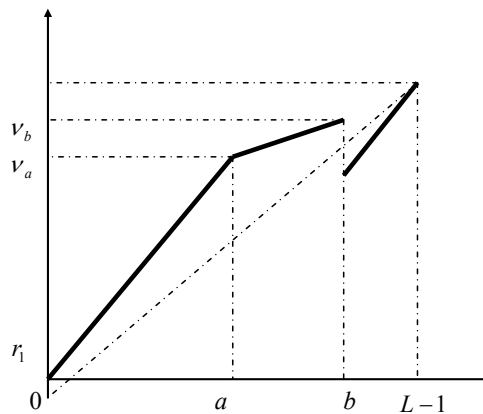


Figura 3.2. Transformación de tramos lineales

Con esta transformación realizamos una modificación de los tonos de luminancia de los píxeles de la imagen en función de los parámetros α, β y γ . La transformación de la figura 3.2 mejora el contraste en el tramo $[0, a]$, pues $\alpha > 1$, y también mejora el contraste en el tramo $[b, L-1]$ pues $\gamma > 1$. Por lo tanto dicha transformación mejora el contraste de los píxeles más oscuros y, también, el de los píxeles más claros.

En [KIM01] se describe un método basado en la siguiente expresión:

$$y_i = (x_i - L_{\min})x(M + US)$$

donde $(M+US)$ es el valor del peso, US corresponde a un parámetro de control seleccionado por el usuario y M es un peso que se calcula de la siguiente manera:

$$M = \begin{cases} 1 + 2^{-n} & \text{si } MSB = 1 \\ 2^m + 2^{-n} & \text{en otro caso} \end{cases}$$

donde MSB es el bit más significativo en la diferencia con el valor L_{min} y los índices m y n son valores enteros y se calculan mediante una heurística específica. La figura 3.3 muestra el esquema de bloques del circuito que implementa este método de control del contraste. La arquitectura del sistema está compuesta por el módulo MM que realiza la resta entre el píxel de entrada y el valor L_{min} , el bloque WD (para calcular peso M), el bloque WE que expande el rango del histograma y está formado por un desplazador y sumadores. El circuito ha sido diseñado en VHDL y se ha realizado la síntesis con la herramienta de Synopsys [CHO00]. Se ha implementado en una tecnología CMOS de $0.25\mu\text{m}$. El circuito generado tiene un coste de 2,317 puertas.

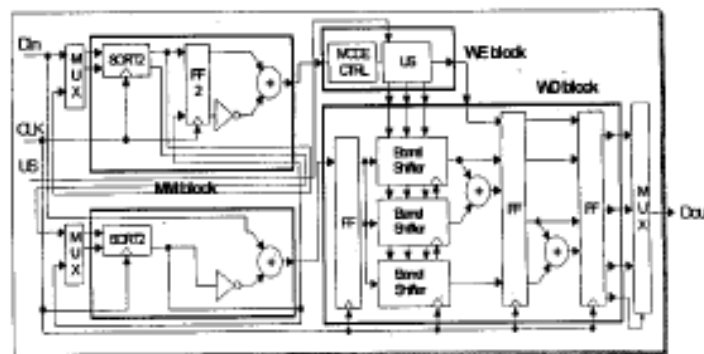


Figura 3.3. Diagrama de bloques del circuito de control del contraste de [CHO00].

El método descrito en [KIM99] se basa en la aproximación de funciones lineales a tramos de la función de densidad acumulativa (CDF, *Cumulative Density Function*). Dicha función CDF se obtiene realizando una acumulación secuencial del histograma. La figura 3.4 muestra el CDF de dos imágenes (una oscura y otra clara) y su aproximación lineal a tramos. En este caso la transformación de la imagen se realiza mediante interpolación lineal:

$$F'(p_n) = scale \times \left(F(p_n) - \frac{256n}{N} \right) + \frac{256n}{N}$$

donde $F(p_n)$ y $F'(p_n)$ son los valores antiguos y nuevos del píxel p_n , N es el número total de tramos y n es el tramo del píxel p_n .

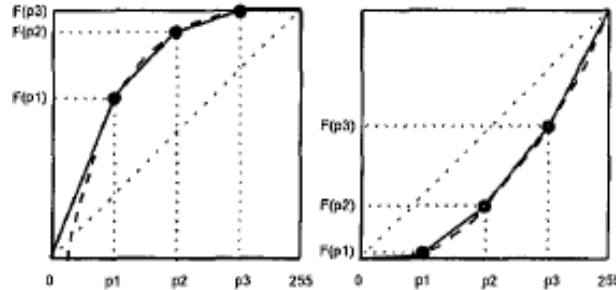


Figura 3.4. CDF y su aproximación lineal a tramos (a) de una imagen oscura, (b) de una imagen clara.

Esta técnica se aplica en imágenes de video. El principal problema surge en que el cálculo del CDF es costoso en tiempo por lo que no se recalcula entre *frames* sucesivos que contienen imágenes similares. La figura 3.5 muestra el diagrama de bloques del circuito de control del contraste aplicando este método para una aproximación del CDF con cuatro tramos (de acuerdo con la figura 3.4). Está constituido por cuatro módulos. El módulo de cálculo del CDF requiere tres comparadores, tres contadores y memoria para almacenar dicha función. El filtro mediana temporal se utiliza para aliviar el problema de cambios abruptos en la imagen entre *frames* consecutivos.

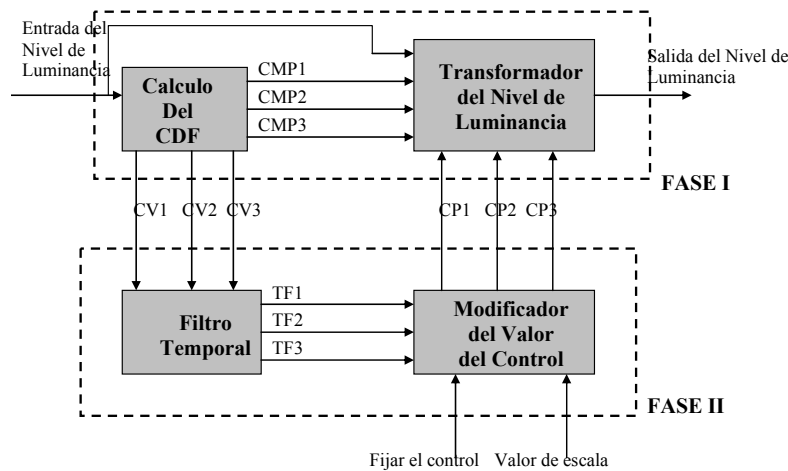


Figura 3.5. Diagrama de bloques del método de [KIM99]

Los autores no dan datos de implementación del circuito. Sin embargo podemos sacar algunas conclusiones. En primer lugar se propone una implementación para un conjunto fijo de tramos por lo que no se puede garantizar una aproximación al CDF adecuada para cualquier imagen. El coste en área es grande ya que cada módulo requiere de varios circuitos aritméticos y de almacenamiento. El coste en tiempo puede ser alto ya que el cálculo del CDF resulta costoso y la solución mediante el filtro no garantiza que pueda aplicarse en videos de alta definición (HDV) a 24-fps.

Otras técnicas se basan en transformaciones locales de los píxeles y reciben el nombre de operaciones de punto. Las operaciones de punto o funciones punto a punto requieren en cada paso, conocer el valor de intensidad de un solo píxel, al cual se le aplica la transformación deseada. Una vez realizada la transformación el píxel no es necesario ya en todo el resto del algoritmo, por lo tanto este tipo de operaciones se denominan de memoria cero.

Las operaciones de punto son ejecutadas más eficientemente con tablas de búsqueda (*Look-Up Tables*, LUTs). Las LUTs son simples vectores que usan el valor del píxel actual como índice del vector. El nuevo valor es el elemento del vector almacenado en esa posición. La nueva imagen se construye repitiendo el proceso para cada píxel. Usando LUTs se evitan cálculos repetidos e innecesarios. Cuando se trabaja con imágenes de, por ejemplo, 8 bits solamente se necesitan calcular 256 valores. En este caso el tamaño de la imagen es indiferente pues el valor de cada píxel de la imagen es un número entre el 0 y 255 y el resultado de tabla de búsqueda produce otro número entre 0 y 255. Estos algoritmos pueden ser implementados sin utilizar ninguna memoria intermedia ya que la imagen de salida puede ser almacenada en el mismo espacio de memoria que la entrada.

Una de las transformaciones no lineales más utilizadas es la transformación gaussiana (ver figura 3.6) que viene dada por la función:

$$g(i,j) = \frac{\phi\left(\frac{f(i,j)-0.5}{\sigma\sqrt{2}}\right) + \left[\frac{0.5}{\sigma\sqrt{2}}\right]}{\phi\left(\frac{0.5}{\sigma\sqrt{2}}\right)}$$

Donde los corchetes en la expresión $[x]$ representan la parte entera de x , y

$$\phi(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-y^2} dy$$

Esta transformación aumenta el contraste de la imagen al hacer más oscuras las partes oscuras y más claras las partes claras.

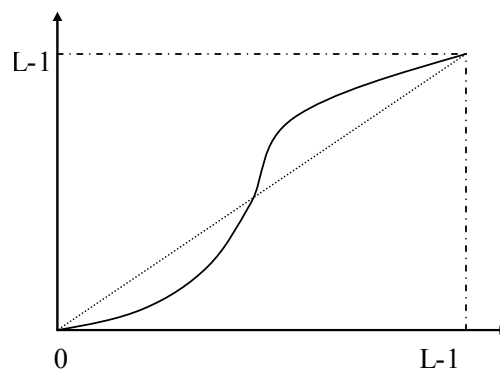


Figura 3.6 Transformación gaussiana.

También se puede usar la función potencia:

$$g(i, j) = c[f(i, j)]^p$$

donde $c = (L-1)^{1-p}$. Para valores de p menores que 1 esta transformación aumenta el contraste de las partes oscuras de la imagen, mientras que para valores mayores que 1 aumenta el contraste de las partes claras.

Otra aproximación no lineal es la descrita en [BOCC01] que se basa en la siguiente expresión:

$$I(x, y) = I_{orig}(x, y)e^{\phi(c(x, y))}$$

3.2. Álgebra de Łukasiewicz

El desarrollo de los conceptos teóricos de las lógicas multivaluadas se inició en la década de los años 20 por Jan Łukasiewicz, quién estableció la generalización de la lógica clásica a la lógica multivaluada. Posteriormente, a finales de los años 50 C.C. Chang formalizó el álgebra multivaluada basada en la lógica de Łukasiewicz. En 1965 Lofti A. Zadeh estableció los fundamentos de la lógica difusa. Ésta puede considerarse como una variante de la lógica multivaluada en la que el grado de incertidumbre tiene distintos niveles. Por lo tanto las lógicas multivaluadas constituyen los fundamentos teóricos en los que se asienta la lógica difusa [NGUY03, NOLA03].

Un álgebra de Łukasiewicz es una sextupla $A = (A, \oplus, \otimes, \neg, 0, 1)$ que satisface las siguientes propiedades [NOLA99]:

- $x \oplus (y \oplus z) = (x \oplus y) \oplus z$
- $x \oplus 0 = x$
- $\neg 0 = 1$ y $\neg 1 = 0$
- $x \oplus (\neg x \otimes y) = y \oplus (\neg y \otimes x)$
- $x \oplus y = y \oplus x$
- $x \oplus 1 = 1$
- $\neg(\neg x \oplus \neg y) = x \otimes y$

El álgebra multivaluada coincide con el álgebra booleana si se verifica la idempotencia ($x \oplus x = x$).

Los operadores vienen definidos de la siguiente forma:

- Suma acotada: $x \oplus y = \min(1, x + y)$
- Producto acotado: $x \otimes y = \max(0, x + y - 1)$
- Implicación: $x \rightarrow y = \min(1, 1 - x + y)$
- Complemento: $\neg x = 1 - x$

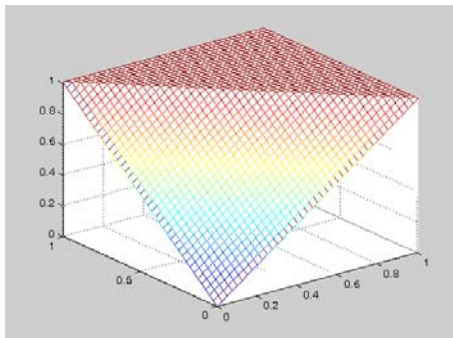
Las siguientes conectivas son de utilidad:

- Operador máximo: $x \vee y = \max(x, y) = (x \otimes \neg y) \oplus y$

- Operador mínimo: $x \wedge y = \min(x, y) = (x \oplus \neg y) \otimes y$

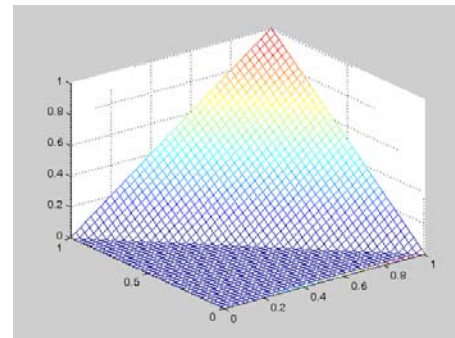
siendo $x, y \in [0, 1]$.

Con objeto de visualizar el significado de los operadores, la figura 3.7 muestra la representación gráfica de cada uno de ellos.



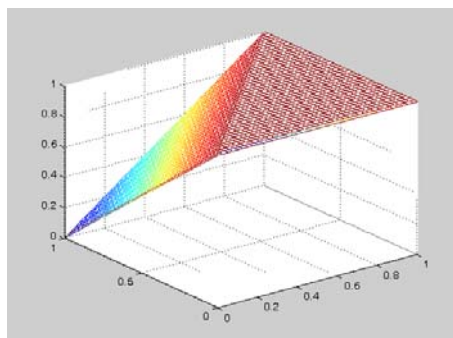
$$x \oplus y = \min(1, x + y)$$

(a)



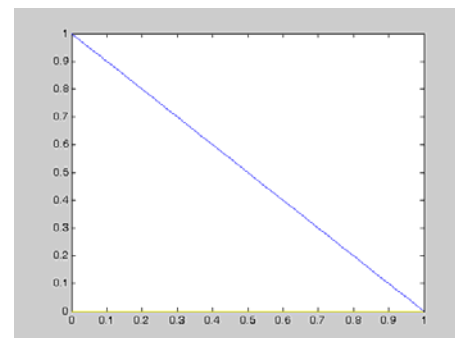
$$x \otimes y = \max(0, x + y - 1)$$

(b)



$$x \rightarrow y = \min(1, 1 - x + y)$$

(c)



$$\neg x = 1 - x$$

(d)

Figura 3.7. Representación gráfica de los operadores de Łukasiewicz.

3.3. Control del contraste mediante operadores de Łukasiewicz

La aplicación de los operadores de Łukasiewicz en una imagen da lugar a una transformación de la distribución de los niveles de los píxeles. Dicha transformación produce una traslación de niveles bajo a niveles alto o de niveles alto a niveles bajo, es decir, la aplicación de los operadores de Łukasiewicz la mayoría de los niveles de gris de la imagen sufren un desplazamiento en el histograma.

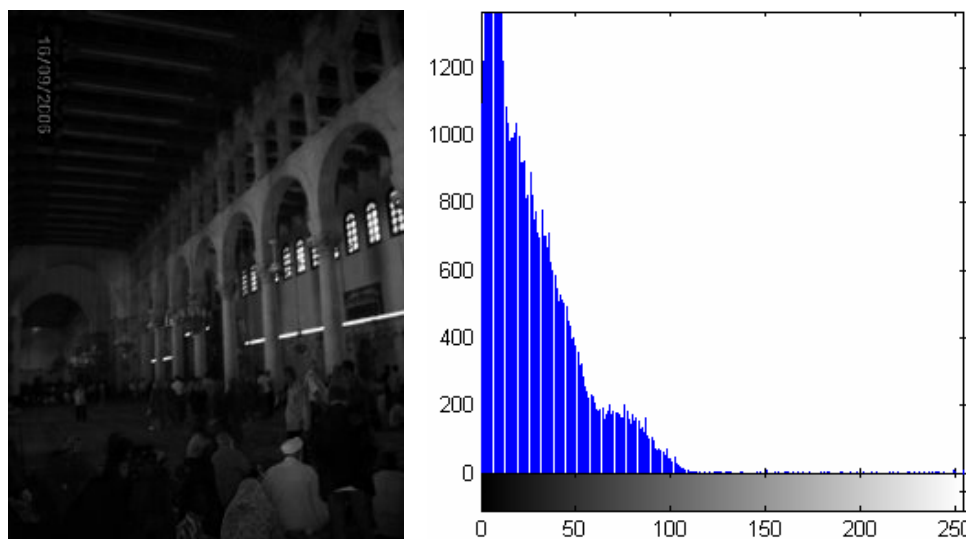
El operador suma acotada actúa como un filtro paso baja y realiza un desplazamiento de los píxeles de la imagen a la zona de los niveles más altos. De esta forma se obtiene una imagen más clara. La figura 3.8b muestra el efecto de aplicar la suma acotada a píxeles consecutivos de la imagen original en la figura 3.8a. Se puede observar el desplazamiento de los píxeles hacia el blanco por lo que se obtiene una imagen más clara.

El control del contraste aplicando la suma acotada puede realizarse introduciendo un parámetro adicional que permita regular el desplazamiento de la frecuencia:

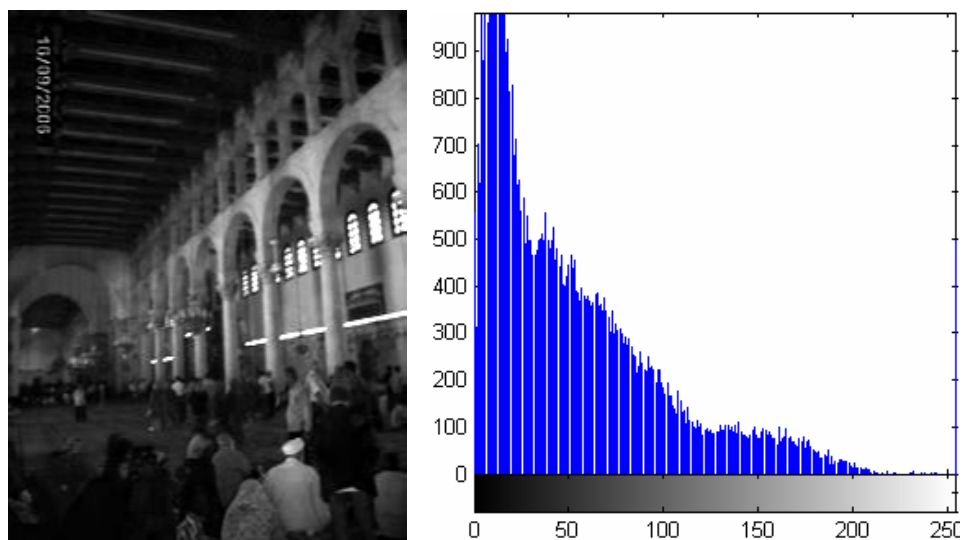
$$x \oplus y \oplus C$$

donde C es el parámetro de control del contraste. El rango de valores que puede tomar C se encuentra en el intervalo $[-255,255]$. En nuestro caso hemos implementado C en el rango $[-128,127]$ ya que se ha codificado con 8 bits.

Las figuras 3.9-3.12 muestran la aplicación de la suma acotada en varias imágenes con distintos valores del parámetro de control C . Se han considerado tanto imágenes en gris como imágenes en color. Puede observarse que el desplazamiento de los niveles al blanco aclara la imagen. Con objeto de comparar el efecto que se produce al aplicar la suma acotada se han considerado varios valores del parámetro de control.

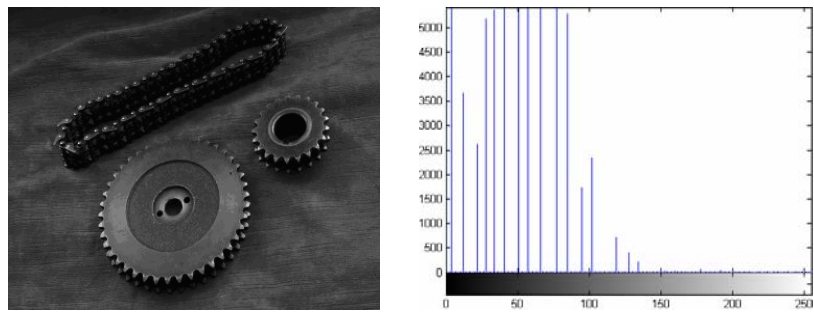


a)

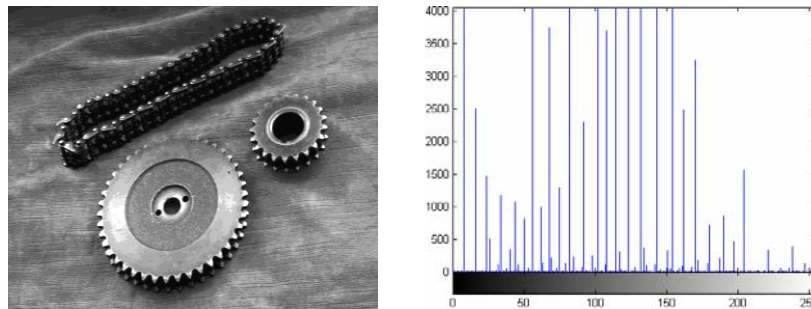


b)

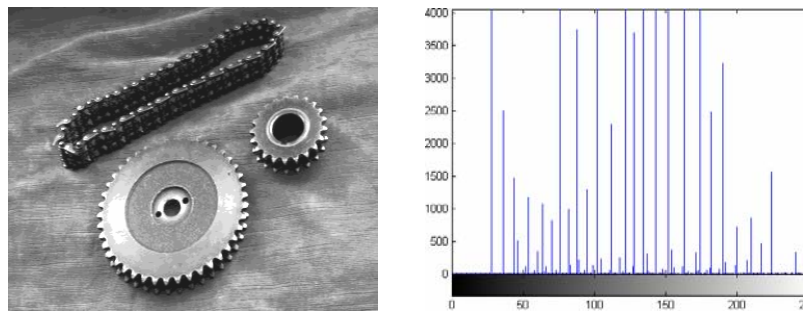
Figura 3.8. a) Imagen original y su histograma. b) Imagen resultante de aplicar la suma acotada y su histograma.



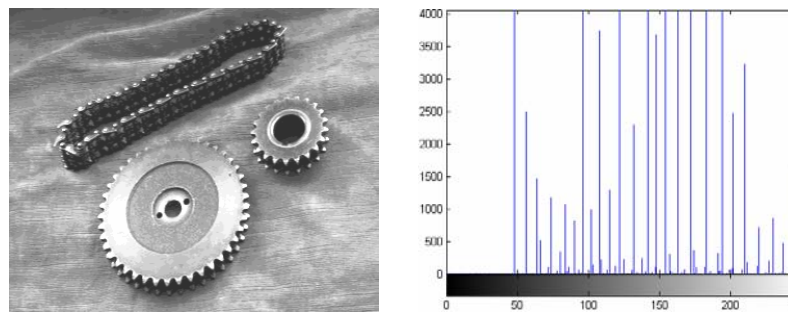
a)



b)



c)



d)

Figura 3.9. a) Imagen original (356x292), b) $x \oplus y$, c) $x \oplus y \oplus 20$, d) $x \oplus y \oplus 40$.

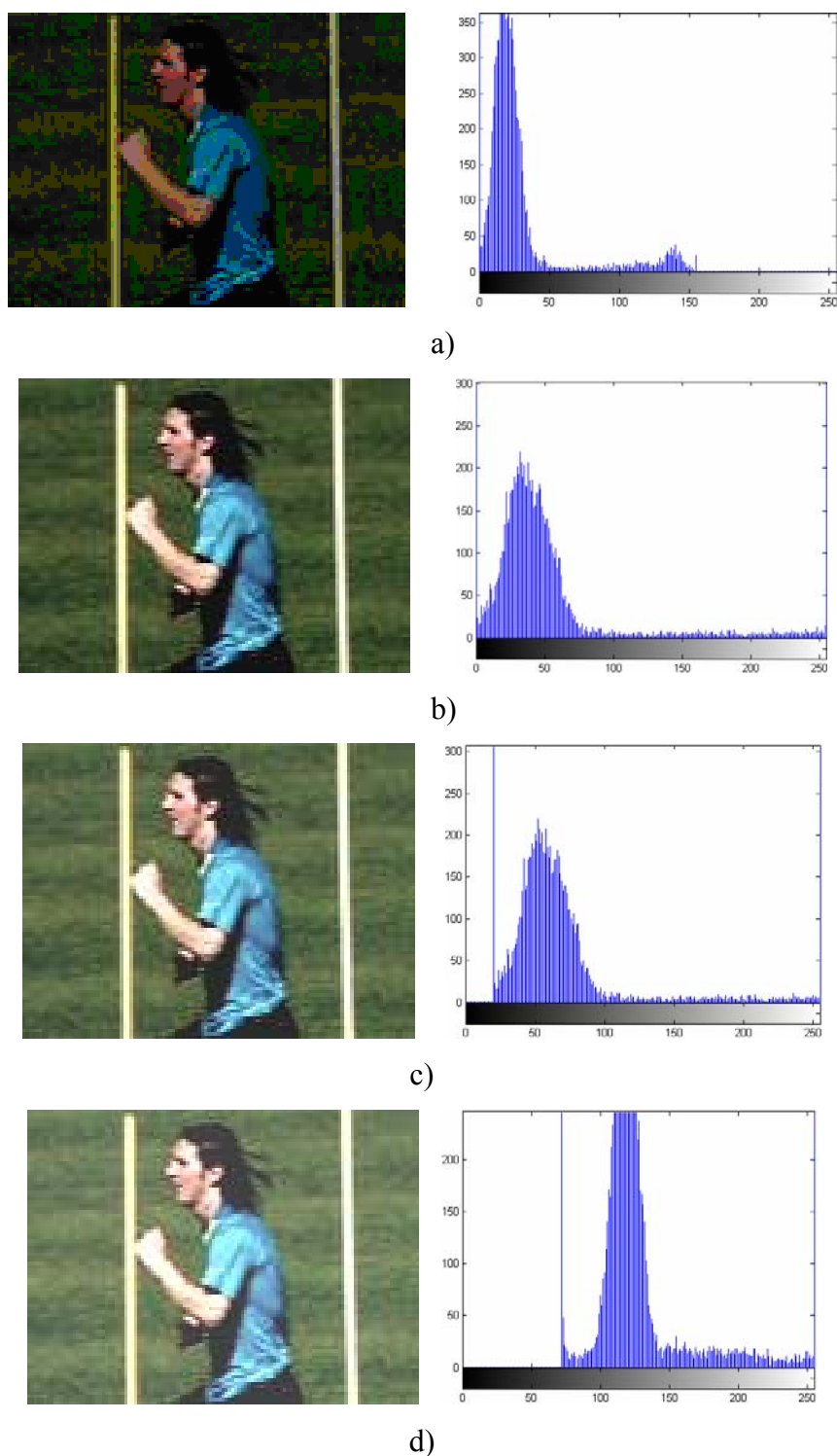


Figura 3.10. a) Imagen Mesi (120x89), b) $x \oplus y$, c) $x \oplus y \oplus 20$, d) $x \oplus y \oplus 40$.

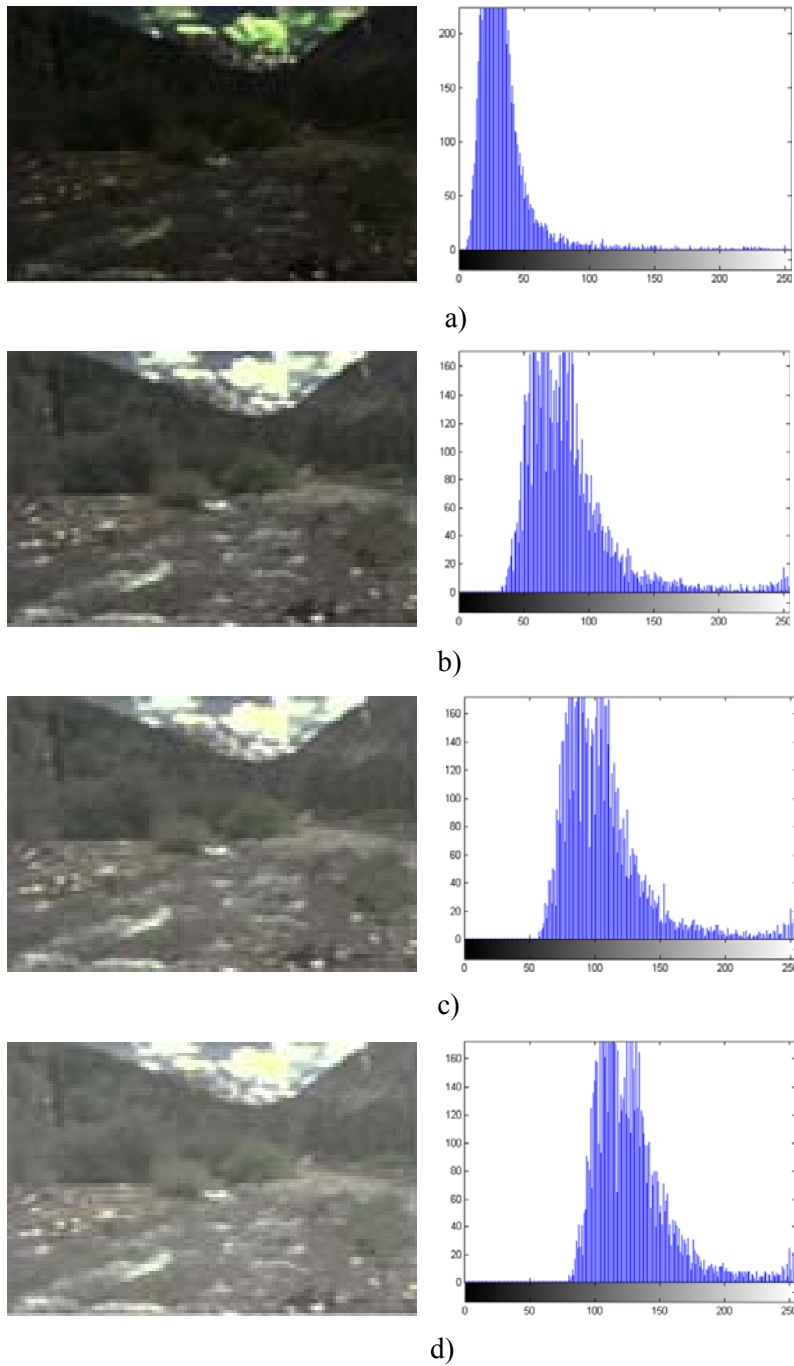


Figura 3.11. a) Imagen original (119x80), b) $x \oplus y$, c) $x \oplus y \oplus 20$, d) $x \oplus y \oplus 40$.

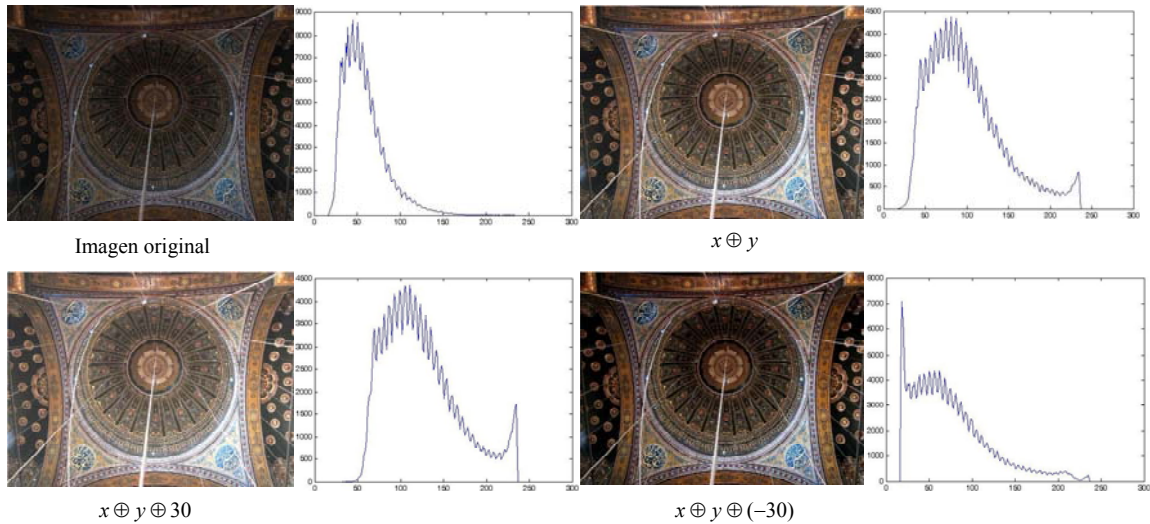


Figura 3.12. Imagen con distintos valores de control de contraste para la suma acotada y los histogramas.

La operación complementaria a la suma acotada corresponde al producto acotado. Este operador da lugar a un desplazamiento del histograma hacia el negro. Este efecto se observa en la figura 3.13 que muestra el resultado de aplicar el producto acotado y su histograma.

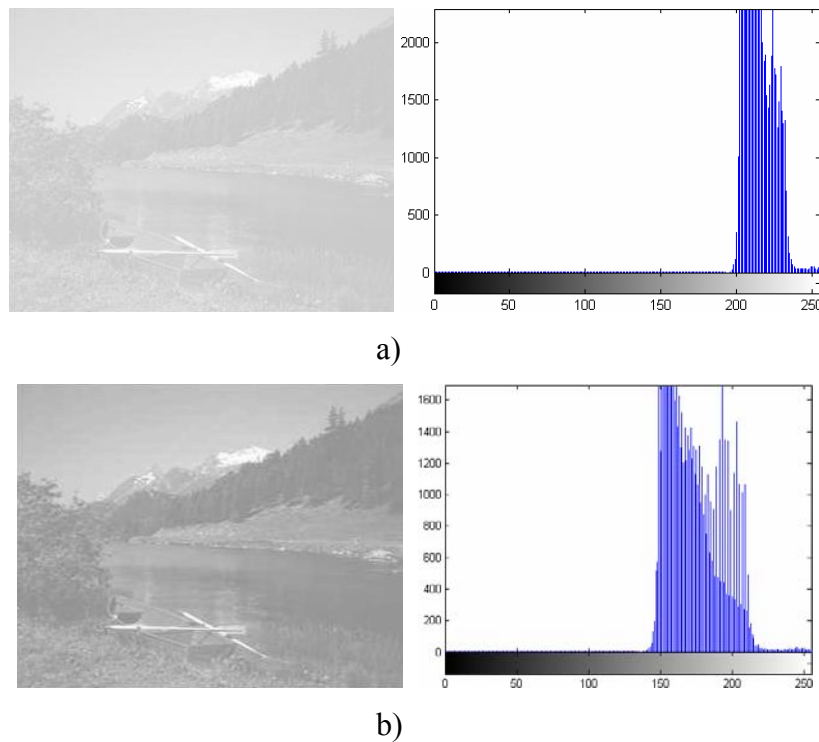


Figura 3.13. a) Imagen original y su histograma, b) imagen resultante de aplicar el producto acotado y su histograma.

El control del contraste aplicando el producto acotado se realiza mediante el parámetro C en la siguiente expresión:

$$x \otimes y \otimes C$$

Las figuras 3.14-3.17 muestran la aplicación del producto acotado en varias imágenes con distintos valores del parámetro de control C . Puede observarse que el desplazamiento de los niveles al negro oscurece la imagen.

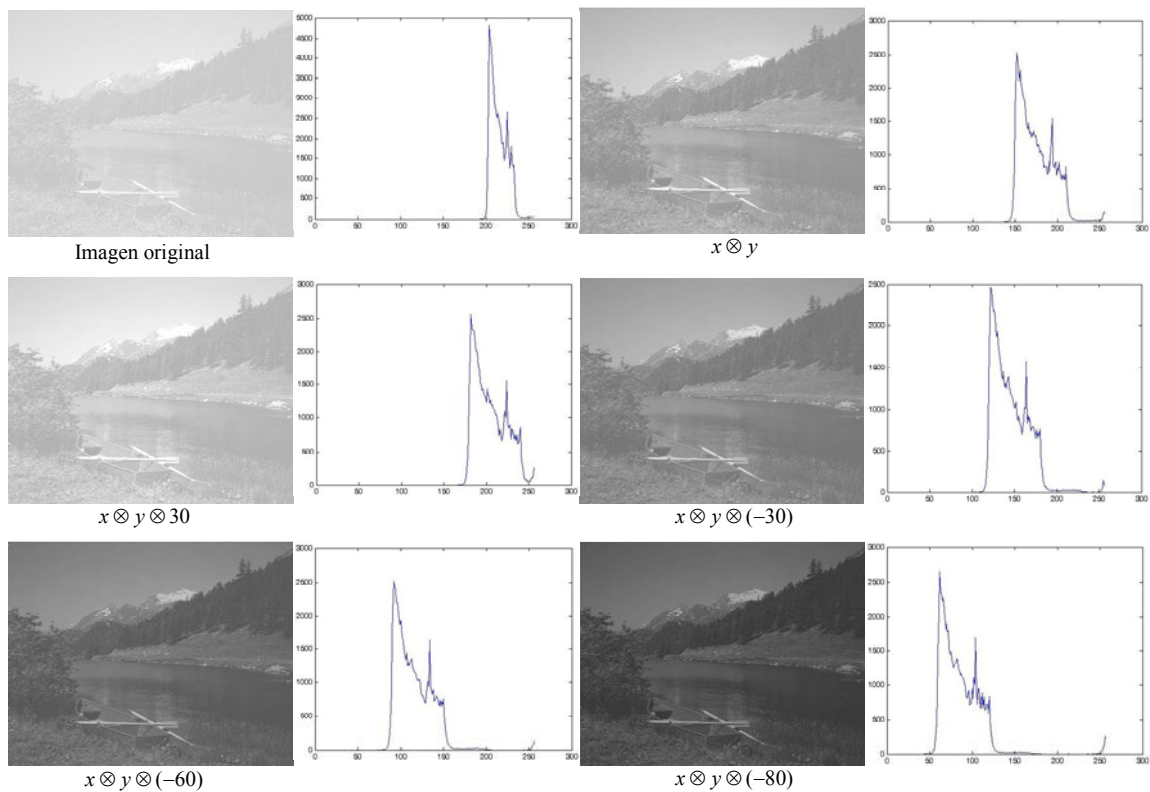


Figura 3.14. Imagen con distintos valores de control de contraste para el producto acotado y los histogramas.

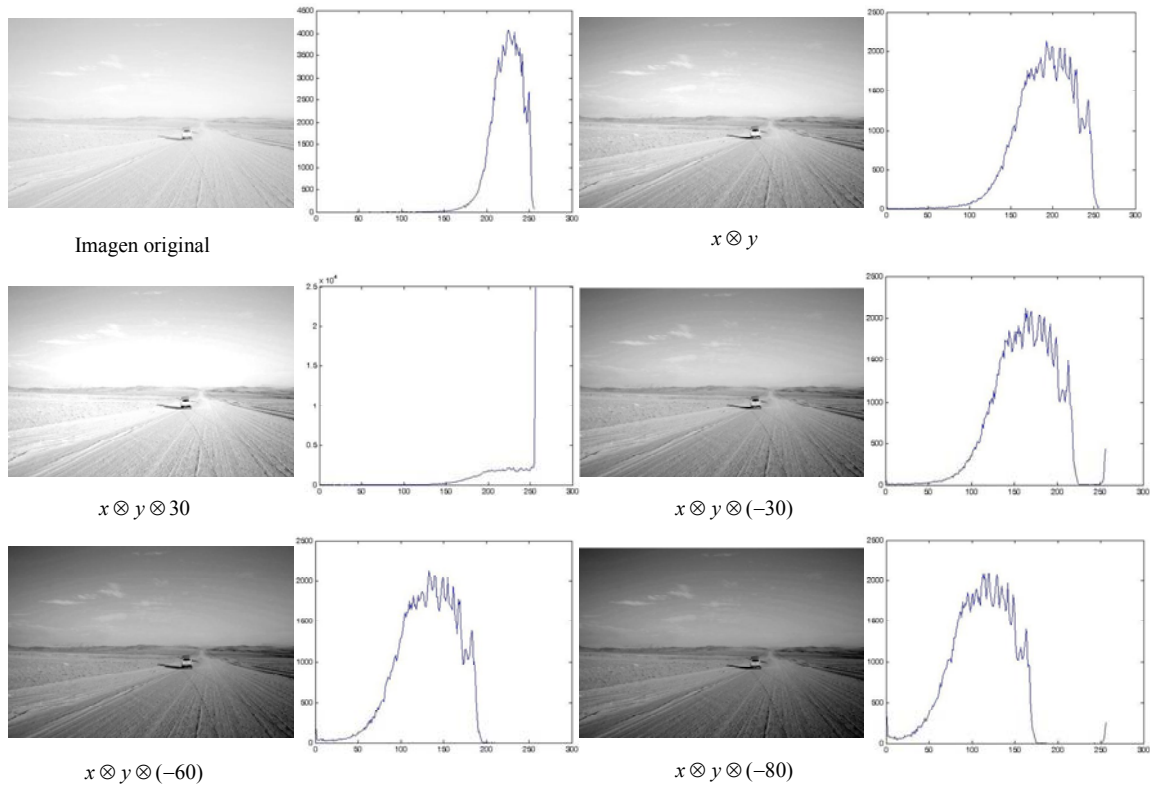


Figura 3.15. Imagen con distintos valores de control de contraste para el producto acotado y los histogramas.

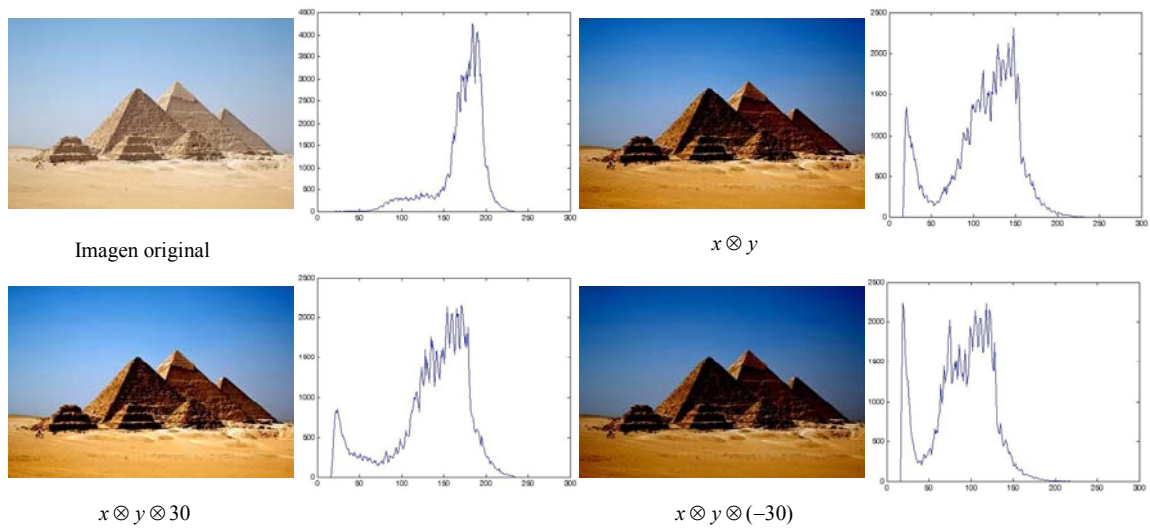


Figura 3.16. Imagen con distintos valores de control de contraste para el producto acotado y los histogramas.

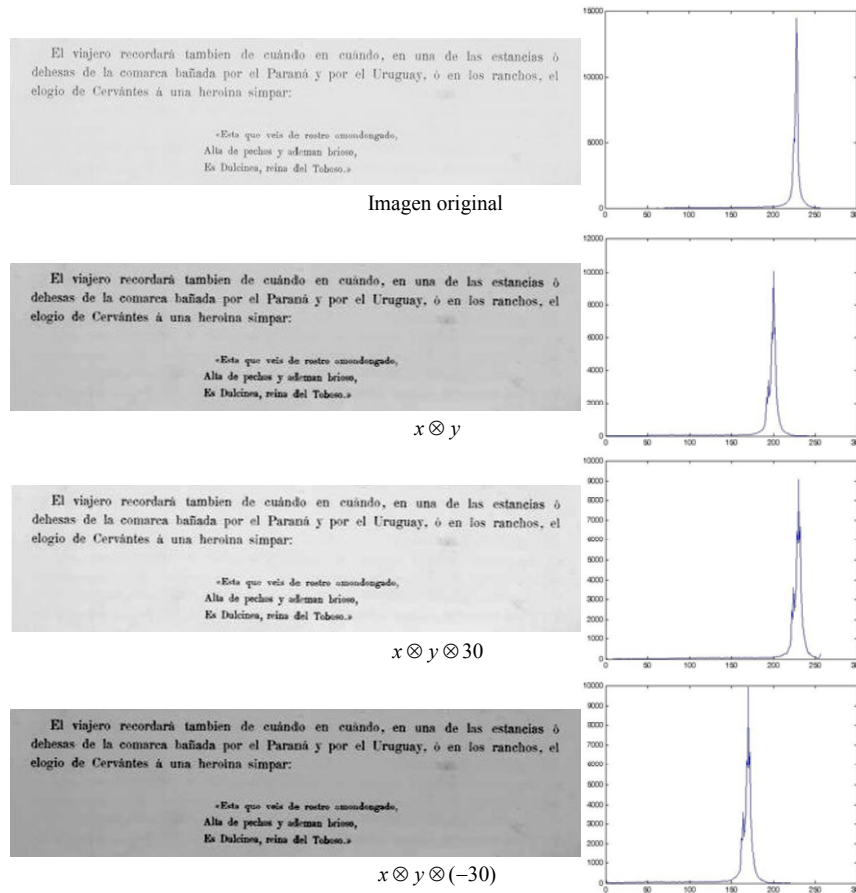


Figura 3.17. Imagen con distintos valores de control de contraste para el producto acotado y los histogramas.

La técnica que proponemos para el control del contraste es una técnica local que modifica la imagen punto a punto. El proceso de transformación se realiza aplicando una máscara que recorre la imagen. Dicha máscara transforma el valor de un píxel bajo la influencia de los píxeles que le rodean. En los ejemplos considerados hasta ahora se han aplicado los operadores de suma acotada y producto acotado a dos píxeles consecutivos de la imagen. Sin embargo es posible considerar otros casos de máscaras. Así las figuras 3.18 y 3.19 muestran el efecto sobre la variación del contraste al aplicar la suma acotada a máscaras de 1x2, 2x2 con diferentes valores del parámetro de control C

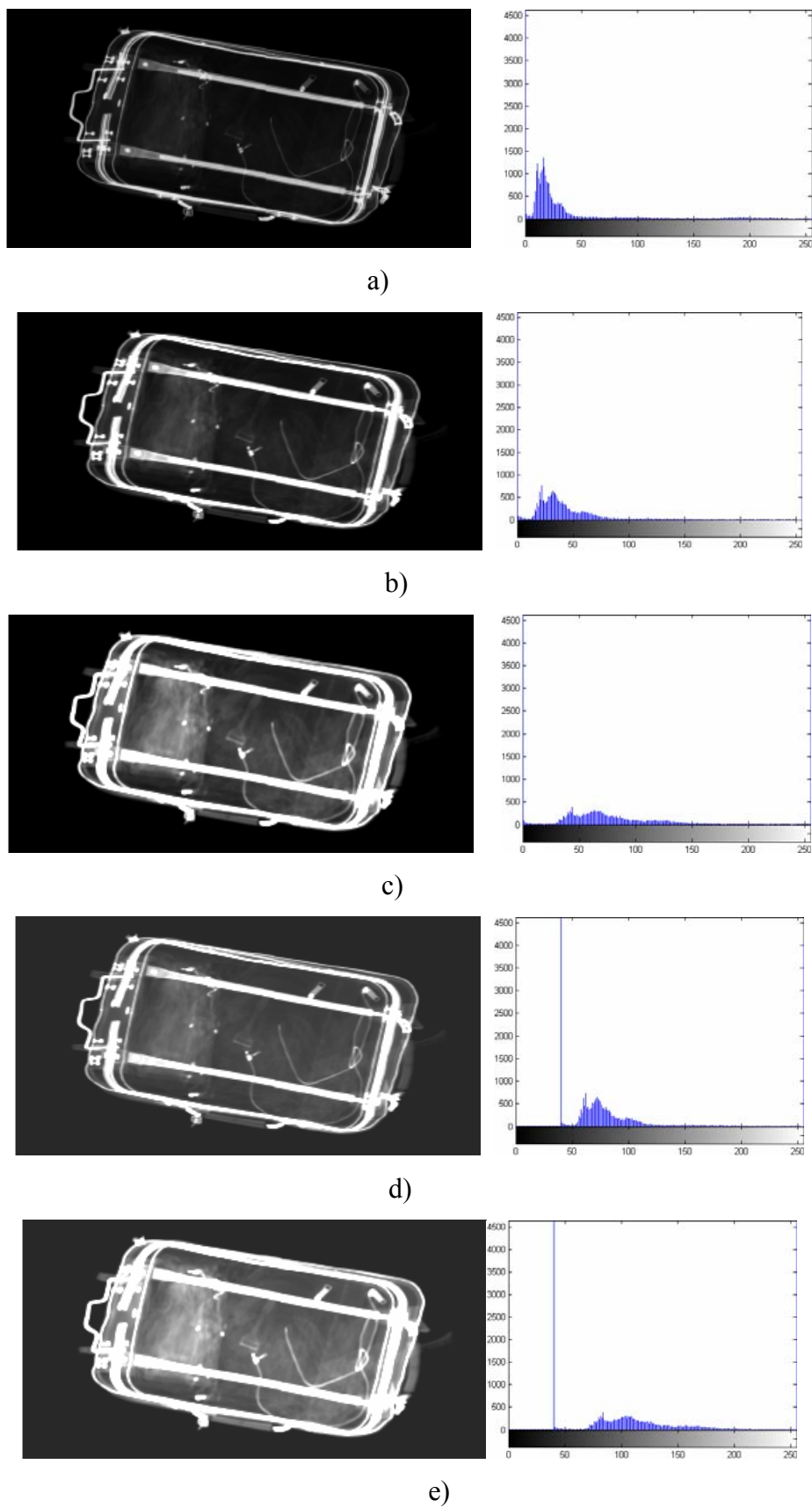


Figura 3.18. a) Imagen original. Suma acotada para mascara de b) 1x2 píxeles, c) 2x2 píxeles y d) 1x2 píxeles y $C=40$, e) 2x2 píxeles y $C=40$.

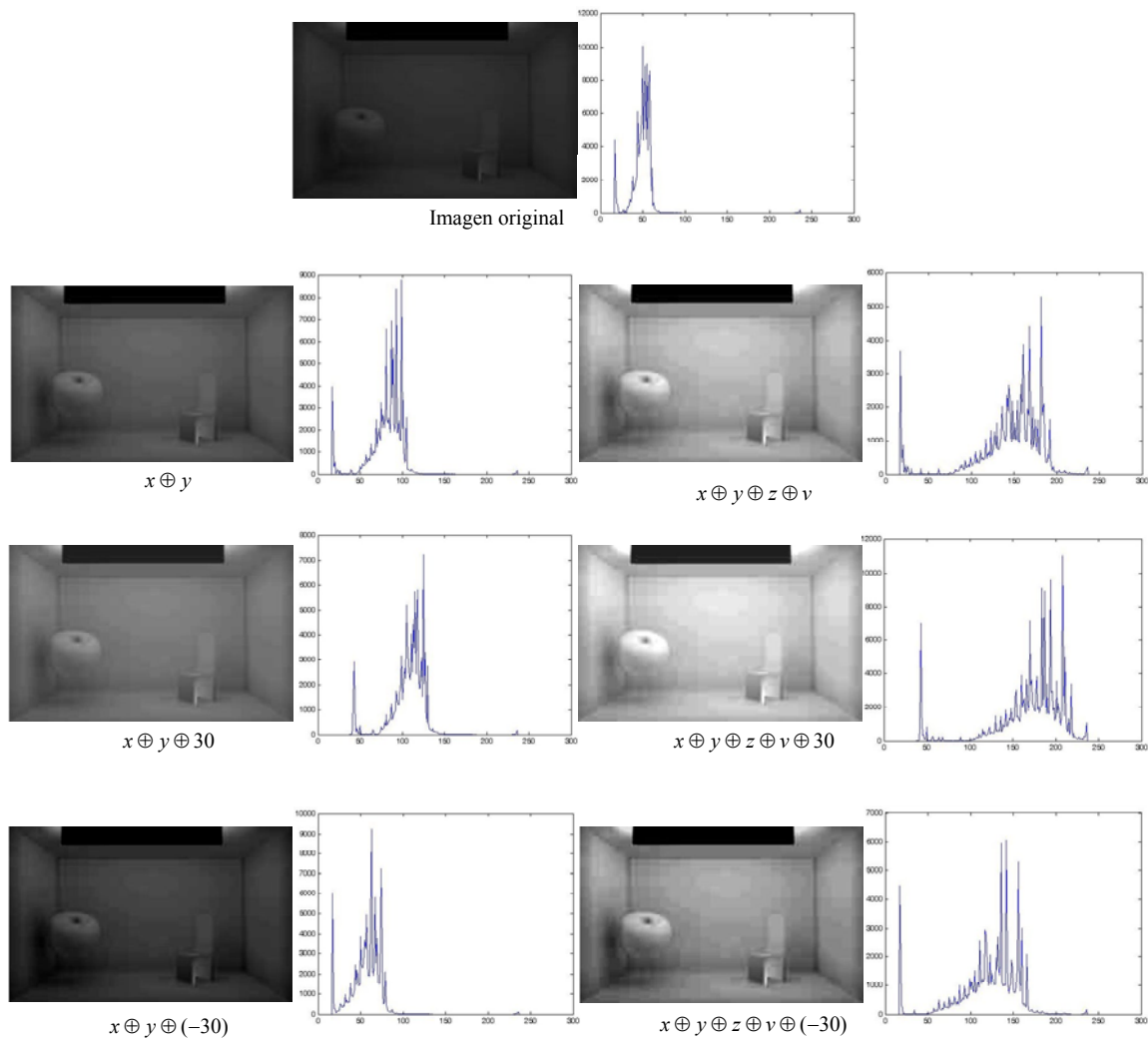


Figura 3.19. Variación del contraste aplicando la suma acotada a máscaras de 2 píxeles y de 2x2 píxeles con diferentes valores del control C (0, 30 y -30).

En la técnica que proponemos para el control del contraste se aplica cada uno de los dos operadores (suma acotada y producto acotado) dependiendo de las características de la imagen. Así la suma acotada se utiliza en el caso de las imágenes oscuras mientras que el producto acotado debe ser aplicado en imágenes claras. Sin embargo, en general, las imágenes pueden tener zonas con diferentes características. Es decir pueden coexistir zonas oscuras y zonas claras en la misma imagen. Por ello conviene adaptar el mecanismo de control de contraste a las características locales de la imagen. Para ello se ha considerado un sistema de toma de decisión que determine el tipo de operador que debe aplicarse en cada momento (suma acotada en la zona oscura de la imagen y el producto acotado en la zona clara).

El sistema de toma de decisiones se basa en un mecanismo de inferencia basado en lógica difusa. Las especificaciones del sistema difuso para el caso de una máscara 1x2 píxeles se muestra en la figura 3.20 y corresponde a las funciones de pertenencia de los antecedentes y del consecuente así como la base de reglas. Las funciones de pertenencia de los antecedentes son tres funciones triangulares distribuidas en el universo de discurso correspondiente al rango de valores [0,255] y solapadas de dos en dos. Por su parte las funciones de pertenencia del consecuente son 3 funciones de tipo singleton Z1, Z2 y Z3 que corresponden a cada uno de los tres mecanismos de generar la salida. Así la función Z1 da lugar a realizar la suma acotada mientras que Z3 supone aplicar el producto acotado. El caso de que la salida sea Z2 significa que no se realiza ningún cambio de contraste y por lo tanto la salida corresponde al valor de la entrada.

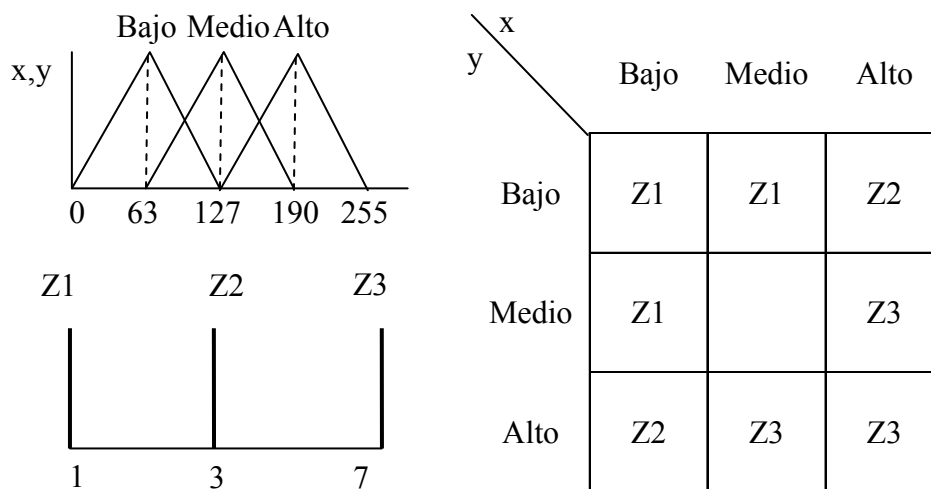


Figura 3.20. El sistema difuso del sistema de toma de decisiones en el control de contraste.

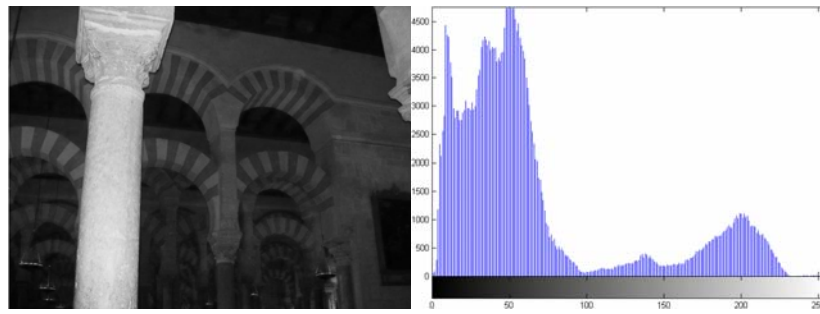
La base de reglas de la figura 3.21 contiene 8 reglas. Cuando el contraste es bajo se aplica la suma acotada o bien el producto acotado mientras que si el contraste es alto la salida no cambia respecto a la entrada. Podemos observar que el caso correspondiente al antecedente “Si x es Medio e y es Medio” no tiene definida ninguna regla. Se han considerado otras dos variantes de esta base de regla. Una de estas variantes corresponde a incluir la siguiente regla:

$$\text{“Si } x \text{ es Medio e } y \text{ es Medio entonces } z \text{ es } Z2\text{”} \tag{1}$$

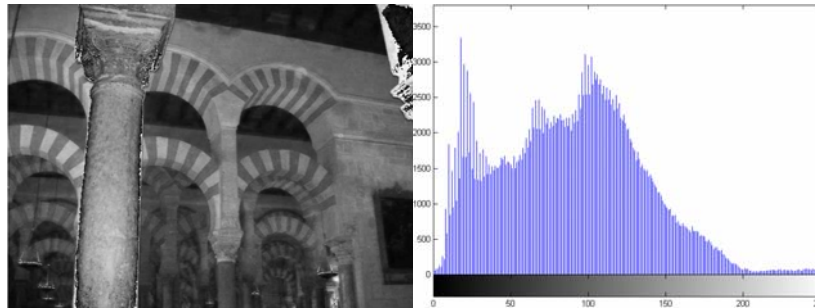
La otra variante incluye la regla siguiente:

“Si x es Medio e y es Medio entonces z es Z1” (2)

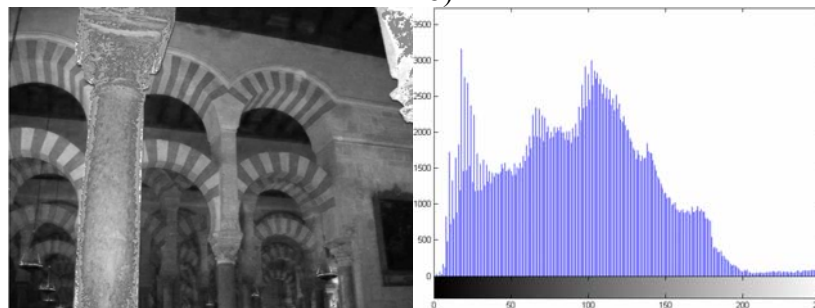
La figura 3.21 muestra los resultados obtenidos para cada uno de las bases de reglas. Como puede observarse los tres casos dan lugar a resultados muy similares. El caso de la figura 3.21d da lugar a imágenes con artefactos así como un menor contraste con respecto a las otras dos soluciones.



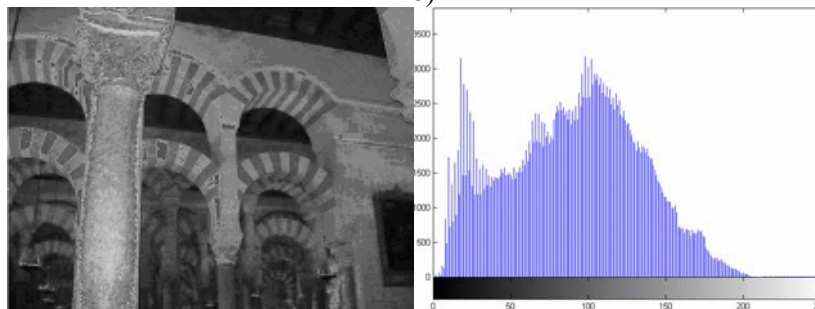
a)



b)



c)



d)

Figura 3.21. Resultados de aplicar control de contraste: a) imagen original, b) sistema basado en la figura 3.20, c) sistema que incluye de regla (1), d) sistema que incluye la regla (2).

3.4. Diseño de los operadores básicos de Łukasiewicz

En este apartado vamos a considerar el diseño de los operadores básicos de Łukasiewicz. A la hora de plantear la estrategia de diseño de los operadores básicos hemos primado, como criterio de diseño la reducción en el retraso y la menor ocupación de área. Puesto que los operadores que vamos a tratar van a ser las primitivas que permitirán construir sistemas para el procesamiento de imágenes resulta prioritaria la optimización de dichos bloques básicos. Las estrategias de realización van a proporcionar diversas estructuras de circuitos con diferentes prestaciones. Al afrontar los diseños vamos a considerar, básicamente, dos estrategias de realización: basada en redes neuronales y basada en lógica combinacional.

3.4.1. Realizaciones basadas en redes neuronales

En una primera estrategia de implementación de los operadores *min* y *max* vamos a realizar el diseño mediante redes neuronales. Para ello nos basaremos en [CAST98] donde se demuestra que dada una función de activación $\psi(x) = 1 \wedge (x \vee 0)$ es posible representar la red neuronal correspondiente como una combinación de proposiciones del cálculo de Łukasiewicz.

La figura 3.22a muestra la estructura de una neurona. La salida es el resultado de aplicar una función a la suma ponderada de las entradas. Dicha función recibe el nombre de función de activación.

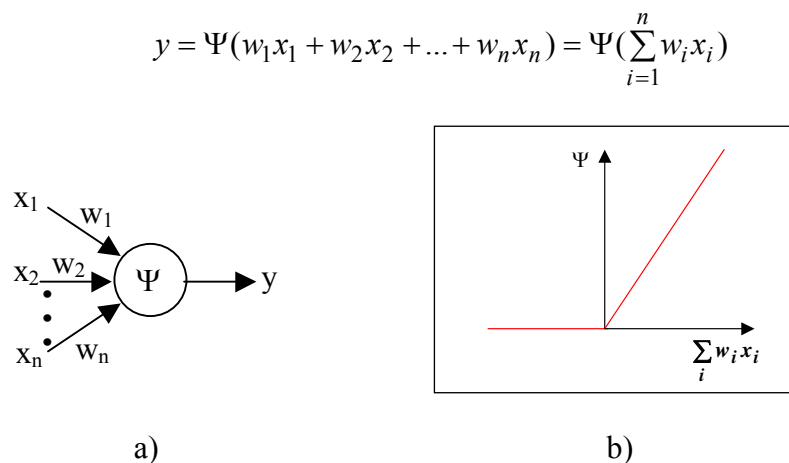


Figura 3.22. a) Entidad de una neurona. b) Función de activación de la neurona.

La función de activación se ilustra gráficamente en la figura 3.22b. Se trata de una función lineal que se anula para valores negativos de la variable independiente. Dicha variable independiente corresponde a la suma ponderada de las variables de entrada. La figura 3.23 muestra el esquemático del circuito que implementa una neurona de dos entradas.

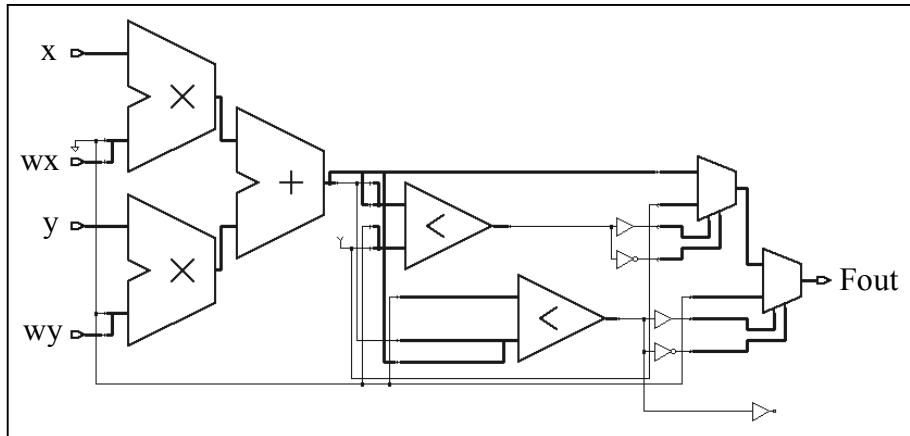


Figura 3.23. Esquemático del circuito de una neurona de dos entradas.

De acuerdo con lo anterior en [AMAT02] se propone las realizaciones de los operadores $\min(x,y)$ y $\max(x,y)$ tal y como se ilustra en la figura 3.24. Se trata de una red de una sola capa cuya salida suministra el comportamiento que se muestra en la figura 3.25 y que corresponde a las superficies de dichos operadores.



Figura 3.24. Operadores $\min(x,y)$ y $\max(x,y)$ realizados mediante redes neuronales.

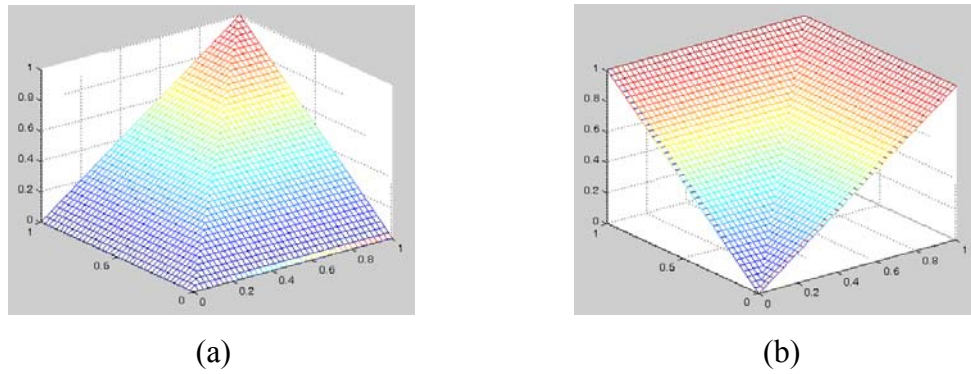


Figura 3.25. Superficies correspondientes a los operadores (a) $\min(x,y)$ y (b) $\max(x,y)$.

De acuerdo con la figura 3.24a tenemos que:

$$\min(x, y) = \Psi(y) - \Psi(y - x)$$

Sin embargo puesto que $y \in [0,1]$ entonces $\Psi(y) = y$ por lo que podremos simplificar la expresión anterior empleando sólo una única neurona:

$$\min(x, y) = y - \Psi(y - x)$$

El diseño del operador máximo es idéntico al del operador mínimo. La expresión que define el operador máximo es la siguiente:

$$\max(x, y) = \Psi(y) + \Psi(x - y) = y + \Psi(x - y)$$

La figura 3.26 muestra los esquemas de circuito de los operadores mínimo y máximo. Podemos observar que se trata de un circuito combinacional. La primera etapa la constituye una neurona (bloque *net*) y la segunda es un sumador o bien un restador.

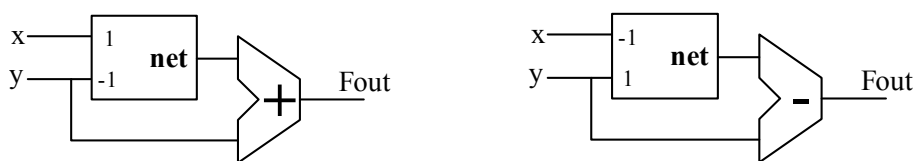


Figura 3.26. a) Circuito máximo, b) circuito mínimo.

La realización de los operadores producto y suma acotados se basa en las expresiones siguientes:

$$x \oplus y = \min(1, x + y) = 1 \wedge (x + y)$$
$$x \otimes y = \max(0, x + y - 1) = 0 \vee (x + y - 1)$$

Al considerar la realización de estos operadores como una red neuronal se puede simplificar el diseño ya que tan sólo se requiere una neurona ya que:

$$x \oplus y = \Psi(1) - \Psi(1 - x - y) = 1 - \Psi(1 - x - y)$$
$$x \otimes y = \Psi(0) + \Psi(x + y) = \Psi(x + y)$$

Por lo tanto el coste de las realizaciones de estos operadores vendrá determinado por el coste de una neurona.

El último bloque que vamos a considerar es el operador implicación. Este operador también puede simplificarse ya que:

$$x \rightarrow y = \min(1, 1 - x + y) = 1 - \Psi(x - y)$$

De acuerdo con esta expresión el circuito que se obtiene es muy similar al del operador máximo.

3.4.2. Realizaciones basadas en lógica combinacional

Una realización alternativa a las redes neuronales consiste en implementar los operadores mediante lógica combinacional [HUSS05a,b]. En este caso las primitivas corresponden a puertas lógicas digitales. Partiendo de una descripción funcional del operador se obtienen las ecuaciones lógicas. Así, por ejemplo, los operadores *min* y *max* se realizan mediante comparadores de acuerdo con el esquema lógico que se ilustra en la figura 3.27.

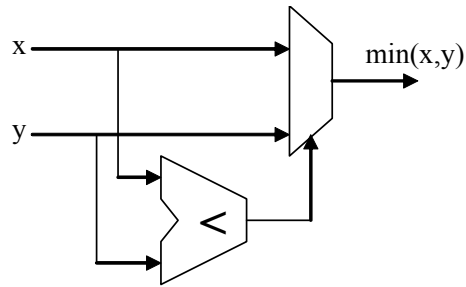


Figura 3.27. Esquema de bloques del operador mínimo.

De manera análoga a los operadores anteriores el producto acotado está constituido por un sumador y un restador en la etapa de entrada y un comparador que selecciona el origen de datos de salida.

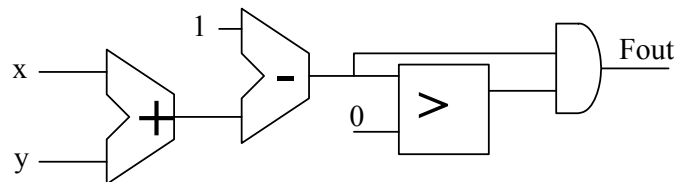


Figura 3.28. Circuito producto acotado.

Si se realiza una codificación de las variables en complemento a dos la etapa de entrada de la figura 3.28 se simplifica por un sumador con la salida complementada y el comparador consiste en conectar el bit más significativo del multiplexor. Con lo cual el circuito simplificado se muestra en la figura 3.29.

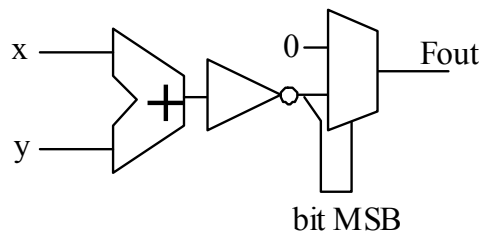


Figura 3.29. Circuito optimizado del producto acotado.

El circuito correspondiente a la suma acotada requiere un sumador en la etapa de entrada para realizar la operación $x+y$. A continuación se compara con 1 y la salida del

comparador controla la selección de canales del multiplexor que actúa sobre el restador de salida.

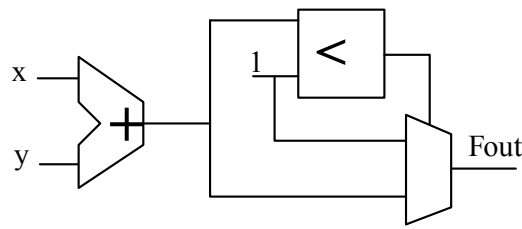


Figura 3.30. Circuito suma acotada.

Por último el operador de implicación es un circuito idéntico al máximo salvo en la etapa de entrada que emplea un restador en lugar de un sumador.

3.4.3. Resultados de implementación

Los circuitos descritos anteriormente han sido implementados mediante la herramienta de síntesis de Xilinx XST. Dicha herramienta se encuentra incluida en el flujo de diseño de FPGA del entorno de desarrollo ISE de Xilinx. Hemos realizado la síntesis e implementación sobre FPGA de la familia Spartan 2E. Ello nos ha permitido disponer de prototipos de forma rápida. De esta manera hemos podido comparar las prestaciones de cada implementación. Las restricciones de síntesis que hemos empleado han primado como criterio de optimización la velocidad frente a la reducción de área y se ha promovido compartir recursos frente a una duplicación de los mismos.

La tabla 3.1 muestra una comparación entre las diferentes estrategias de diseño de cada operador. En este caso se trata de una comparación que tiene en cuenta los recursos consumidos por los dispositivos.

Operador	Basado en red neuronal			Basado en lógica combinacional		
	<i>slices</i>	LUTs	puertas	<i>slices</i>	LUTs	puertas
min/max	21	39	447	8	16	120
\otimes	27	51	474	13	25	204
\oplus	27	49	552	10	18	153
\rightarrow	21	39	402	8	16	141

Tabla 3.1. Recursos consumidos

De acuerdo con la tabla 3.1 cada operador se ha implementado de acuerdo con dos estrategias: mediante una red neuronal y mediante lógica combinacional. Para cada estrategia se dan tres datos: *slices*, LUTs y puertas. Todos esos datos son diferentes medidas del mismo concepto, el área ocupada.

Los *slices* corresponden a los bloques lógicos que definen la funcionalidad del circuito en el FPGA. Los *slices* se constituyen con tablas de búsquedas (LUT, *LookUp Table*). Esas LUT se implementan mediante una memoria SRAM. Finalmente las puertas representan una medida de puertas equivalentes de un circuito lógico sobre un ASIC. Una puerta equivalente equivale a una puerta AND de dos entradas.

Se puede observar que las realizaciones basadas en lógica combinacional dan lugar a mejores resultados (circuitos con menor área) que el caso de estrategia basada en red neuronal.

La tabla 3.2 muestra una comparación de las diferentes realizaciones en función del retraso máximo. Puesto que se trata de circuitos combinacionales este retraso mide la frecuencia máxima de operación del circuito.

Operador	Basado en red neuronal	Basado en lógica combinacional
min/max	19 ns	16 ns
\otimes	20 ns	17 ns
\oplus	20 ns	15 ns
\rightarrow	17 ns	14 ns

Tabla 3.2. Retraso máximo

Podemos observar en la tabla que las realizaciones basadas en lógica combinacional presentan mejores prestaciones que los casos basados en red neuronal. En el caso de las realizaciones basadas en red neuronal se obtienen frecuencias de operación que oscilan entre los 50 MHz y los 59 MHz. En el caso de las realizaciones basadas en lógica combinacional se obtienen frecuencias de operación en el rango entre los 59 MHz y los 71 MHz.

3.5. Control de contraste aplicando lógica difusa

La técnica de control del contraste que se ha presentado se basa en realizar una transformación del histograma de la imagen aplicando los operadores suma acotada y producto acotado. Estos operadores dan lugar a un desplazamiento y expansión de los valores del histograma. El control de este efecto se realiza mediante un parámetro C que permite regular la intensidad de la transformación. La variación del contraste en una imagen no tiene porque ser uniforme. Así pueden existir regiones donde el contraste sea menor que en otras zonas de la imagen. Por ello el parámetro C debería adaptarse a cada región de la imagen con objeto de mejorar la calidad de la transformación. Así la expresión que regula el contraste mediante la suma acotada viene dada por la siguiente expresión:

$$x \oplus y \oplus f(x, y)$$

donde x e y son píxeles de la imagen y el parámetro de control es la función $f(x,y)$.

La función de control de contraste $f(x,y)$ depende de las características de cada imagen y permite adaptar la operación de control de contraste de manera local. En nuestro caso se ha optado por aplicar una heurística que determina dicha función aplicando un criterio de decisión mediante un mecanismo de inferencia basado en lógica difusa. Así el sistema de toma de decisiones se basa en criterios de proximidad, esto es, si los valores de los píxeles están muy cercanos (poco contraste) la función $f(x,y)$ debe ser alta mientras que si los píxeles están alejados la función debe ser baja.

La figura 3.31 muestra las especificaciones del sistema difuso para el control de contraste. Las funciones de pertenencia corresponden a tres funciones triangulares equiespaciadas y con grado de solapamiento de dos. La salida del sistema está compuesta por 5 funciones de pertenencia de tipo *singleton*. La base de regla detalla la heurística descrita anteriormente, es decir,

Si x es Bajo e y es Bajo entonces $f(x,y)$ es Muy bajo (F1)

Si x es Bajo e y es Medio entonces $f(x,y)$ es Bajo (F2)

Si x es Bajo e y es Alto entonces $f(x,y)$ es Medio (F3)

...

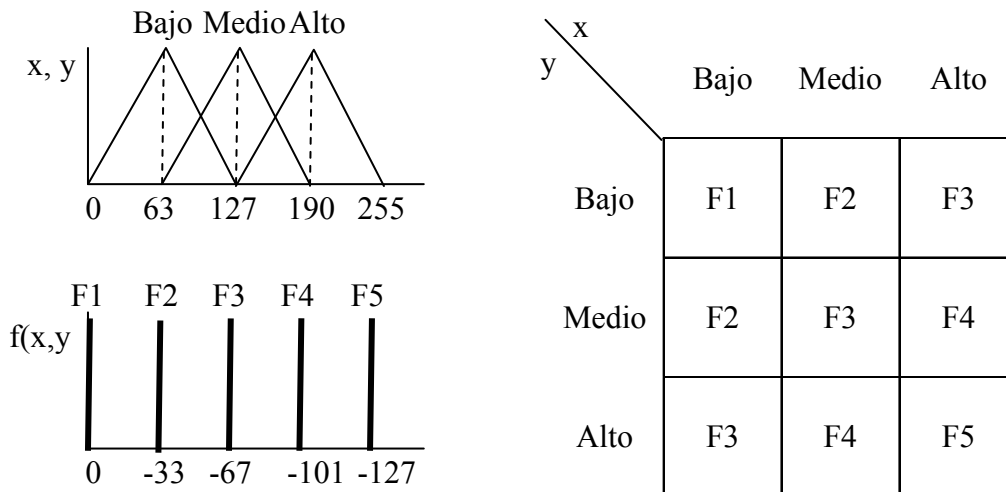


Figura 3.31. Sistema para el control de contraste con 3 funciones de pertenencia para los antecedentes, 5 para el consecuente y 9 reglas.

El grado de control se puede incrementar aumentando el número de funciones de pertenencia. Así la figura 3.32 muestra un caso con 5 funciones de pertenencia para los antecedentes y 9 funciones de pertenencia para el consecuente. En este caso la base de reglas contiene 25 reglas.

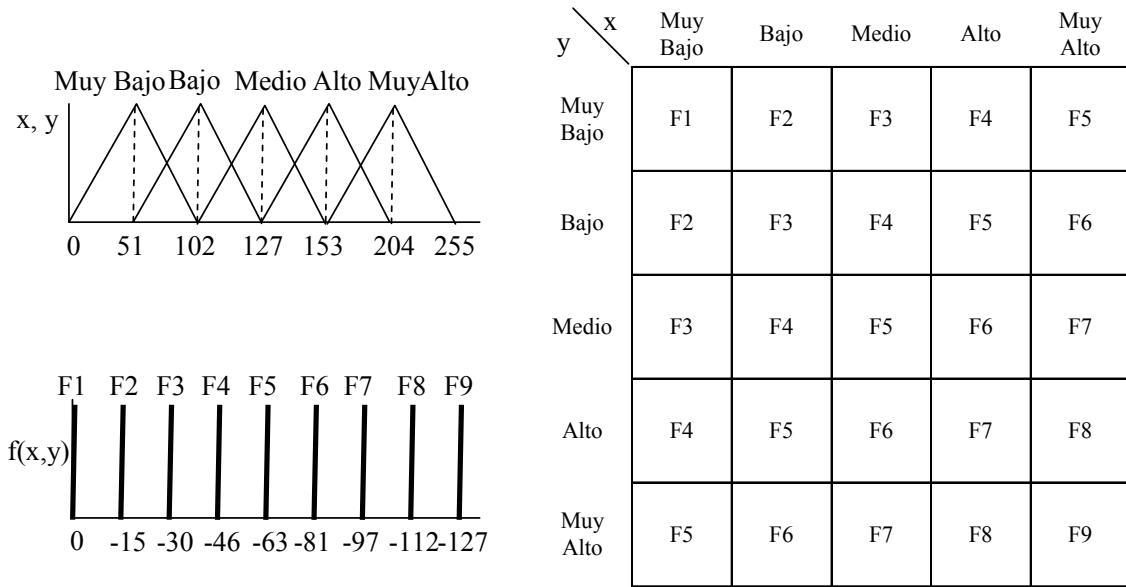


Figura 3.32. Sistema para el control de contraste con 5 funciones de pertenencia para los antecedentes, 9 para el consecuente y 25 reglas.

La figura 3.33 muestra las superficies correspondientes a la función de control de contraste para los casos de 3 y 5 funciones de pertenencia en los antecedentes. Puede observarse que cuando se aumenta el número de funciones de pertenencia los cambios en la superficie de control se suavizan.

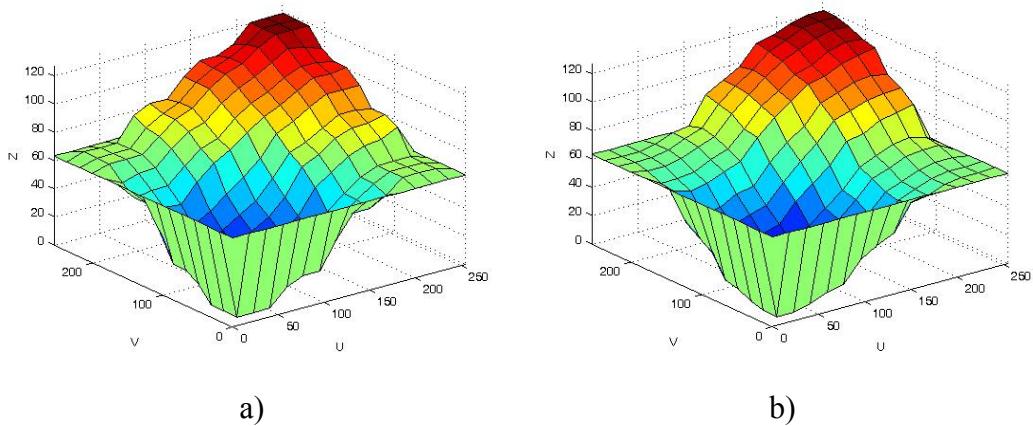


Figura 3.33. Superficies correspondientes a la función de control de contraste para el caso de a) 3 funciones de pertenencia en los antecedentes, b) 5 funciones de pertenencia.

La figura 3.34 muestra un ejemplo de aplicación del control de contraste. El caso de la figura 3.34b corresponde a la suma acotada, la figura 3.34c corresponde a la función de control basada en el sistema difuso con 3 funciones de pertenencia en los

anteriores y la figura 3.34d corresponde a la función de control con 5 funciones de pertenencia en los antecedentes. Puede observarse en la figura 3.34 que en la zona de la imagen correspondiente a la columna se puede apreciar los efectos del control del contraste. Se observa que cuando no se establece control los valores de la columna se saturan (toman el valor blanco) por lo que se pierde contraste. Sin embargo cuando se aplica un control local (casos c y d) se mejora el contraste en la zona de la columna. El efecto de mejora del contraste también aparece cuando se aumenta el número de funciones de pertenencia ya que se suavizan las transiciones en la superficie de control. Las figuras 3.35 y 3.36 muestran otros ejemplos en los que se aprecian efectos similares.

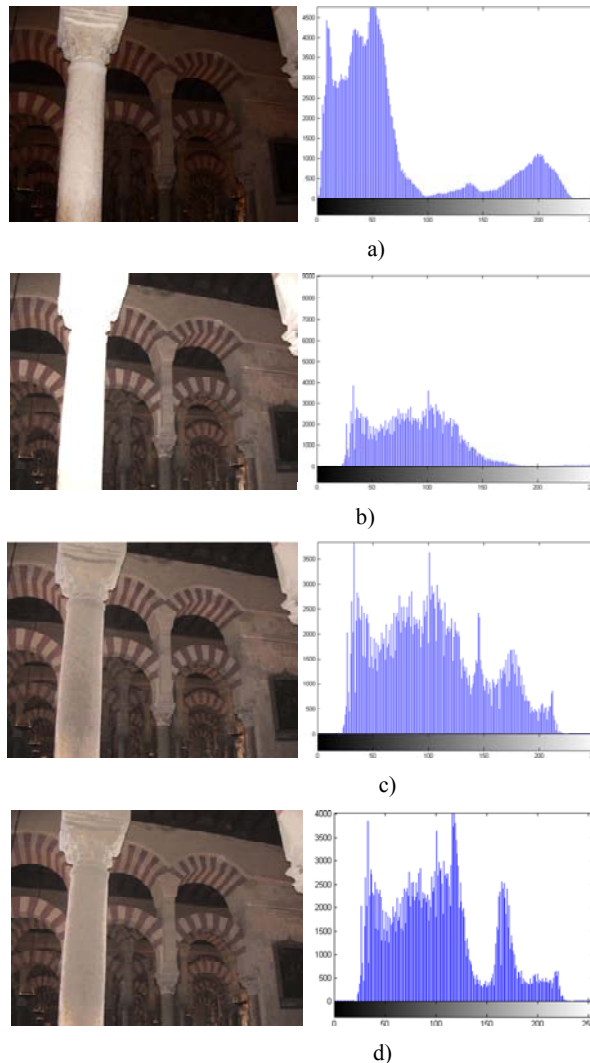


Figura 3.34. a) Imagen original, b) $x \oplus y$, c) $x \oplus y \oplus f_{3MF}(x, y)$,
d) $x \oplus y \oplus f_{5MF}(x, y)$.

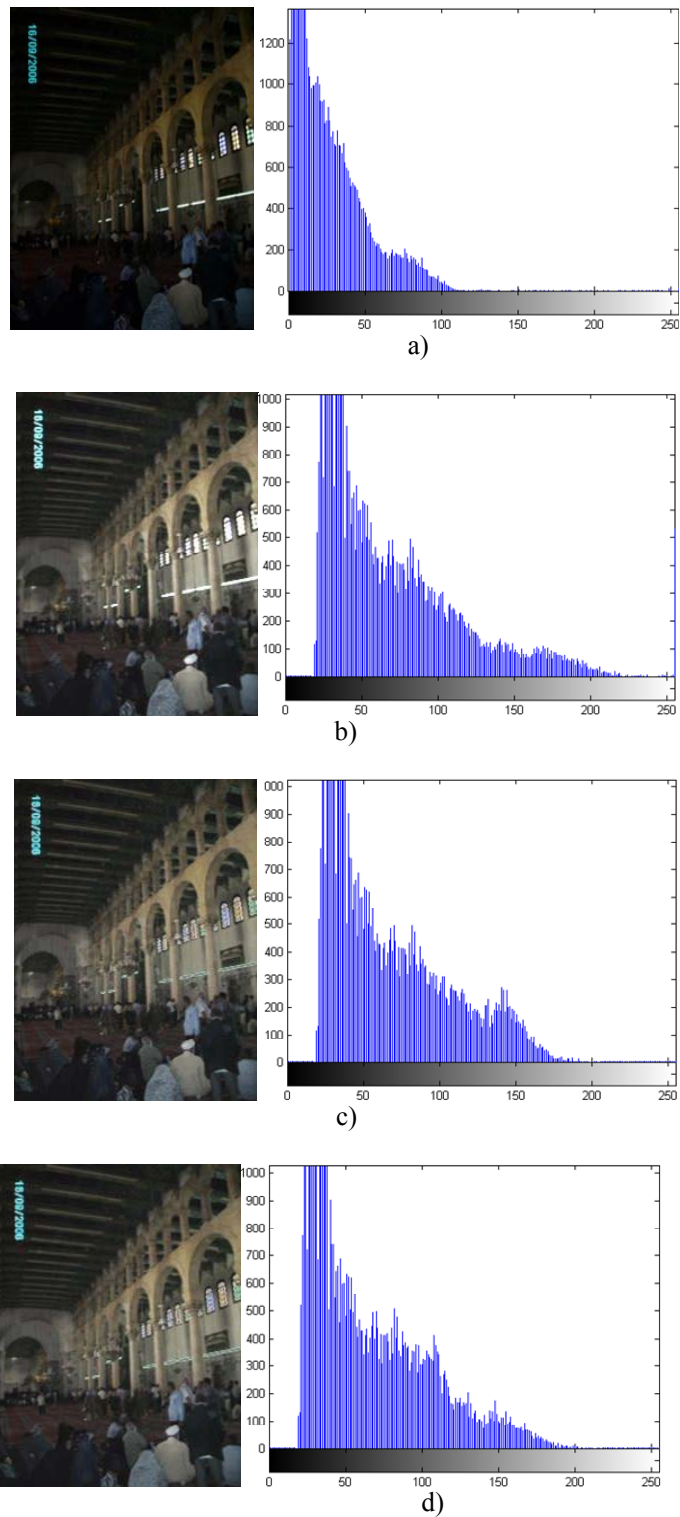
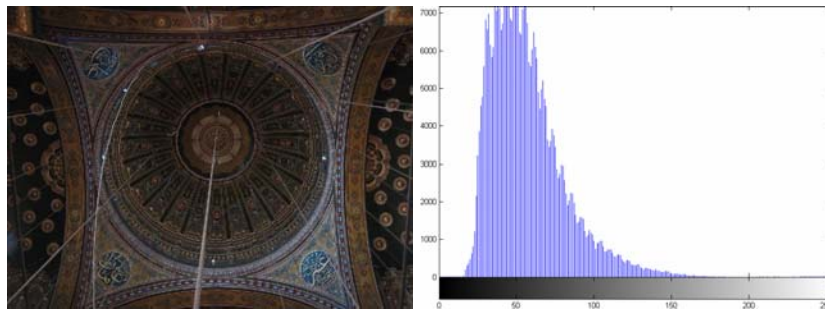
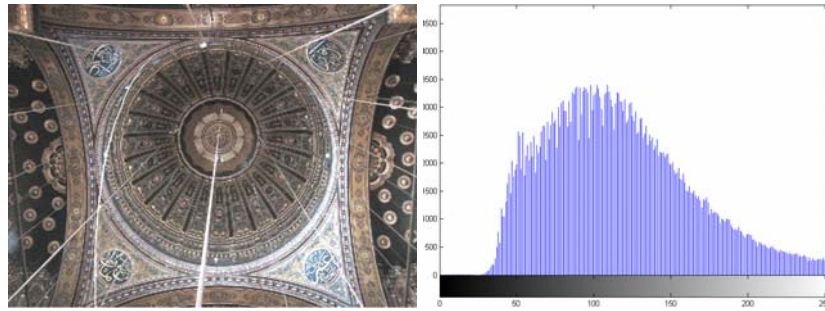


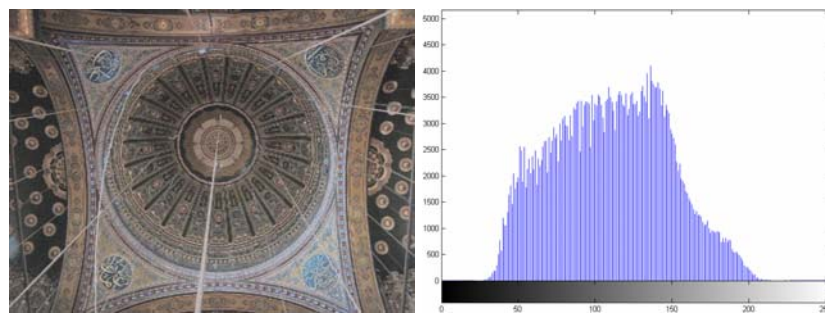
Figura 3.35. a) Imagen original, b) $x \oplus y$, c) $x \oplus y \oplus f_{3MF}(x, y)$,
d) $x \oplus y \oplus f_{5MF}(x, y)$.



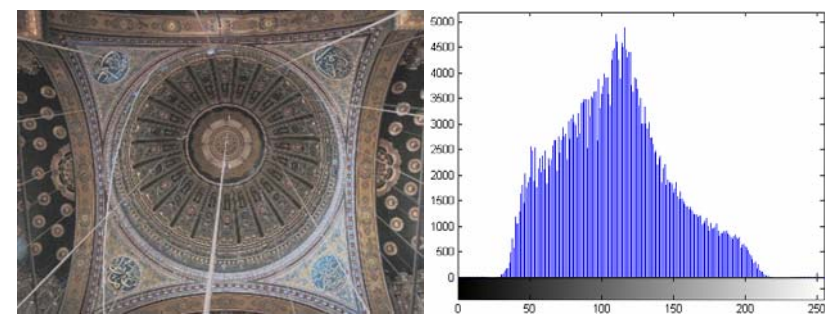
a)



b)



c)



d)

Figura 3.36. a) Imagen original, b) $x \oplus y$, c) $x \oplus y \oplus f_{3MF}(x, y)$,

d) $x \oplus y \oplus f_{5MF}(x, y)$.

La elección del rango de valores de la función $f(x,y)$ influye en la imagen final. Así una opción puede ser emplear un rango de valores positivos y negativos de la función. En este caso los valores del parámetro $f(x, y)$ son valores del rango $[-63;64]$. La figura 3.37 muestra las especificaciones del sistema difuso para el control de contraste: funciones de pertenencia de antecedentes y consecuente y base de reglas.

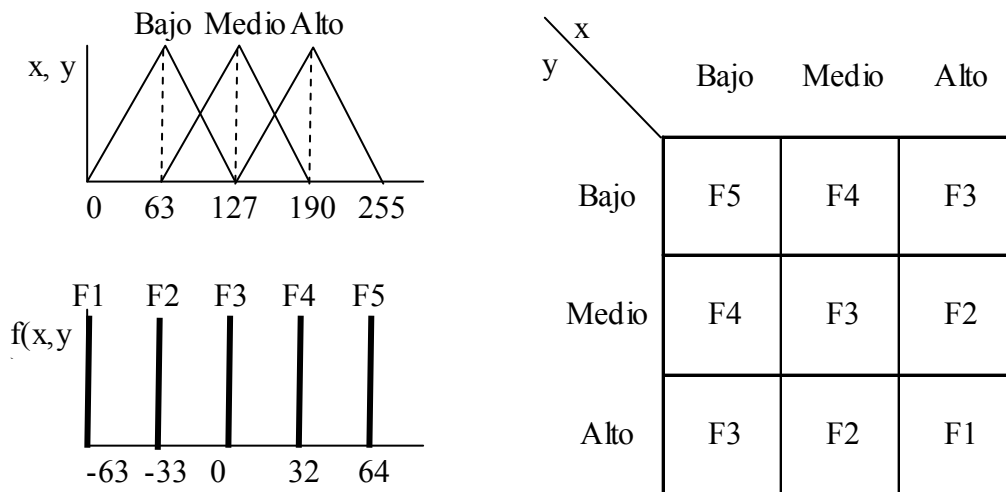
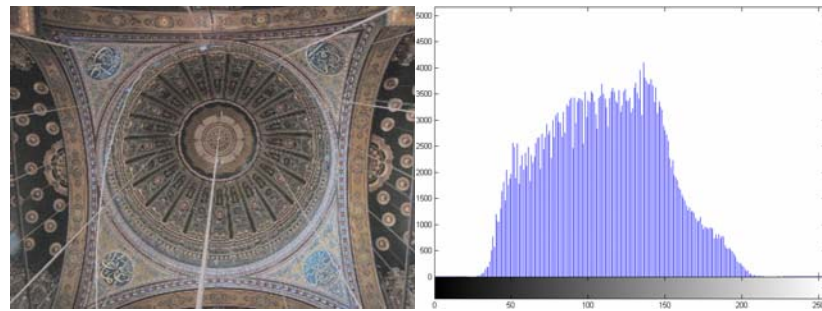
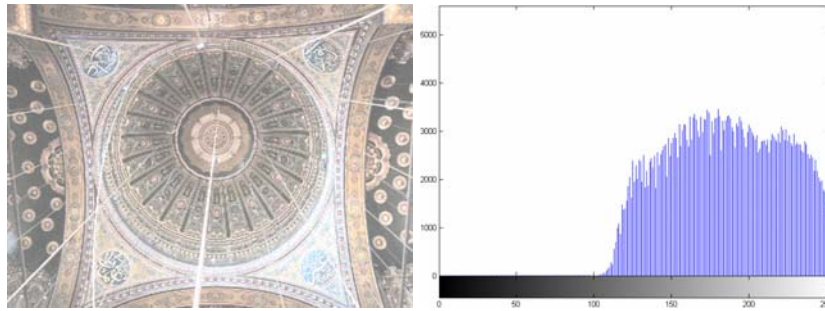


Figura 3.37. Sistema para el control de contraste con 3 funciones de pertenencia para los antecedentes, 5 para el consecuente y 9 reglas.

El resultado de aplicar este sistema da lugar a una reducción del contraste de la imagen. Esto se observa en la figura 3.38 donde se comparan los resultados de aplicar la función $f(x,y)$ en el rango $[-127,0]$ (figura 3.38a) y el caso $f(x,y)$ en el rango $[-63,64]$ (figura 3.38b). Este último caso da lugar a imágenes más claras concentrándose el histograma en niveles de brillo altos.



b)



b)

Figura 3.38. a) Caso de $f(x,y)$ en el rango $[-127,0]$, b) caso de $f(x,y)$ en el rango $[-63,64]$.

En el caso de la aplicación del producto acotado la expresión que regula el contraste viene dada por la siguiente expresión:

$$x \otimes y \otimes f(x, y)$$

De la misma manera que en el caso de la suma acotada el cálculo de la función de control de contraste se basa en un motor de inferencia difuso basado en la base de conocimientos mostrada en la figura 3.39 o bien en el de la figura 3.40 para los casos de 3 o 5 funciones de pertenencia en los antecedentes.

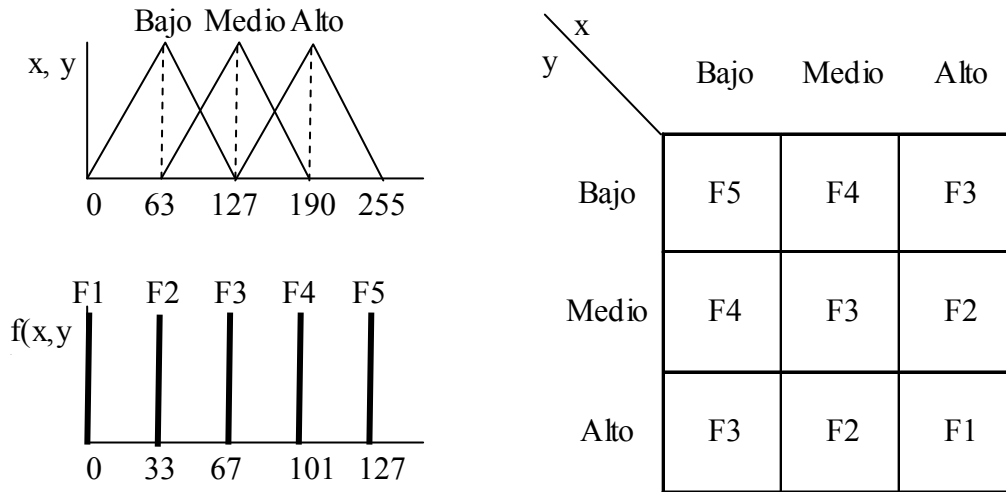


Figura 3.39. Sistema para el control de contraste con 3 funciones de pertenencia para los antecedentes, 5 para el consecuente y 9 reglas.

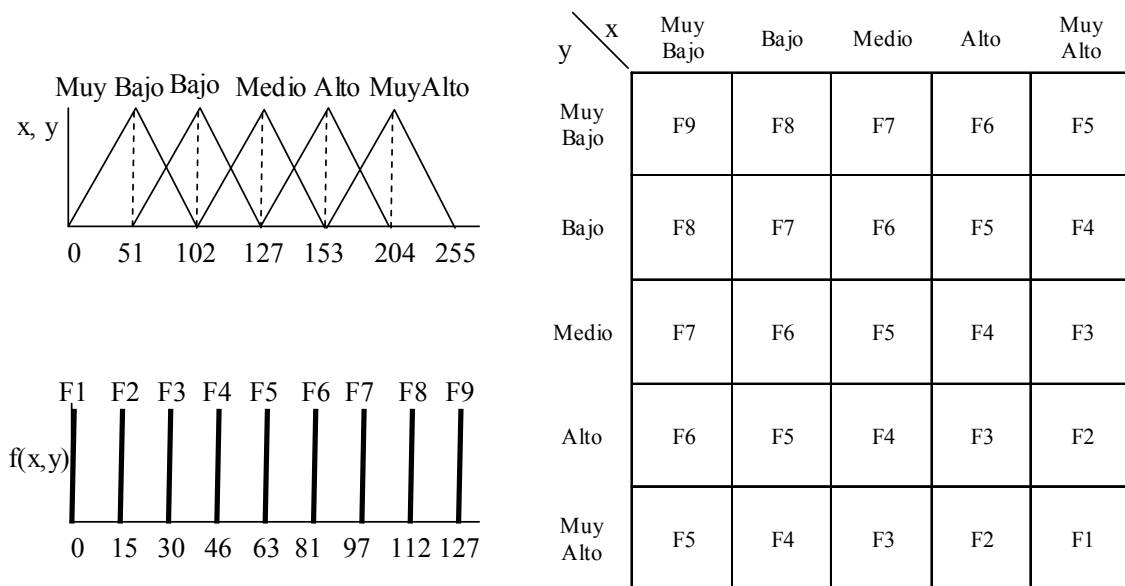


Figura 3.40. Sistema para el control de contraste con 5 funciones de pertenencia para los antecedentes, 9 para el consecuente y 25 reglas.

Los resultados que se obtienen de la aplicación del producto acotado se muestran en la figura 3.41. En el caso de la figura 3.41b se muestran los resultados corresponden al producto acotado sin adaptación mientras que las figuras 3.41c y 3.41d corresponden a los sistemas difusos con 3 y 5 funciones de pertenencia en los antecedentes respectivamente.

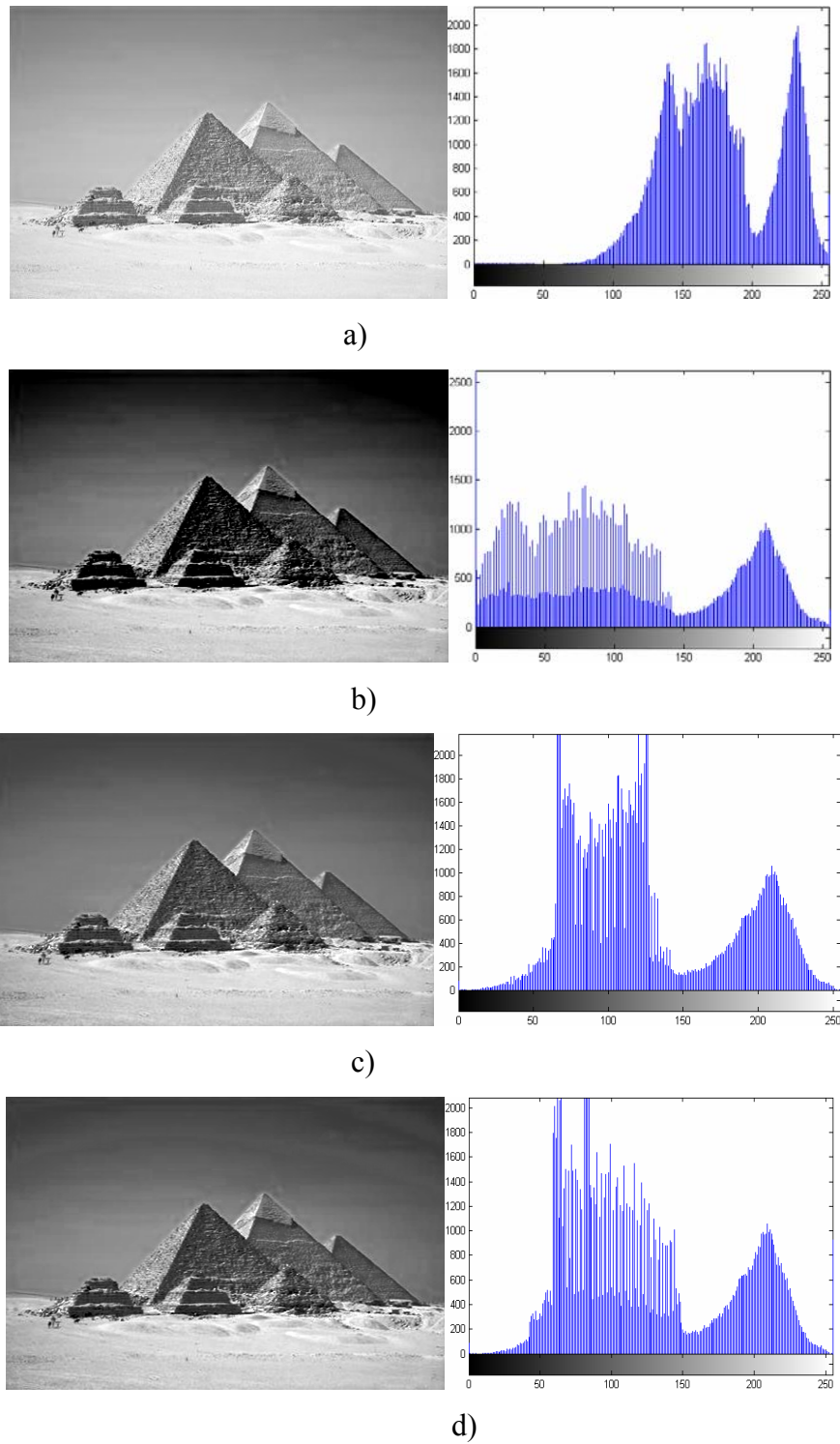


Figura 3.41. a) Imagen original, b) $x \otimes y$, c) $x \otimes y \otimes f_{3MF}(x, y)$,
 d) $x \otimes y \otimes f_{5MF}(x, y)$.

También en este caso se ha comprobado que cuando el rango de valores de la función $f(x,y)$ cubre valores positivos y negativos se reduce el contraste. Este hecho

puede observarse en la figura 3.42. Por lo tanto en este caso la solución que ofrece mejores resultados corresponde a una función positiva en el rango $[0,127]$.

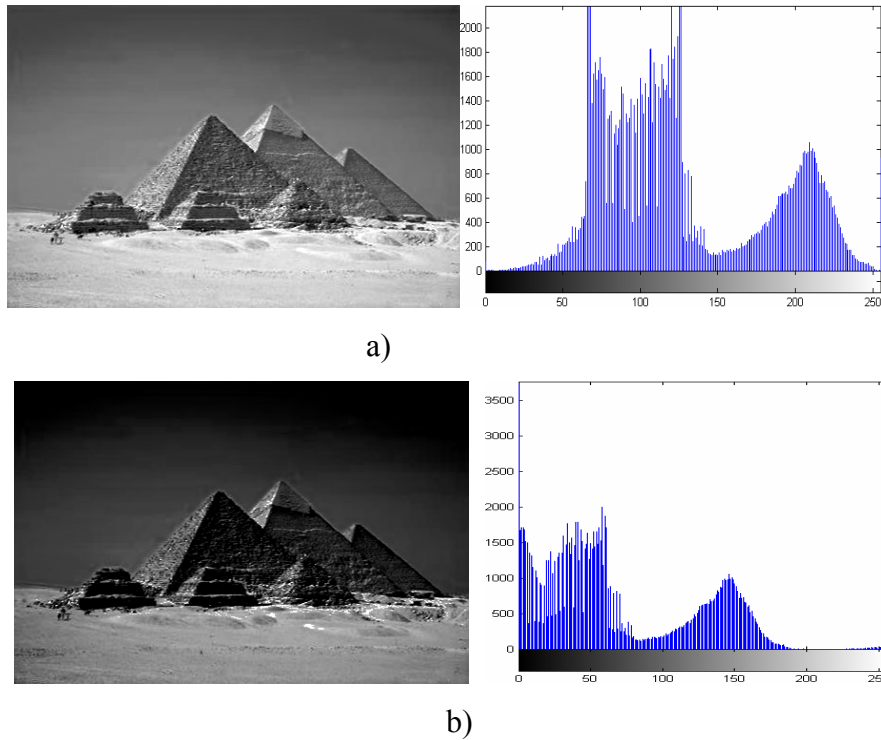


Figura 3.42. a) Caso de $f(x,y)$ en el rango $[0,127]$, b) caso de $f(x,y)$ en el rango $[-63,64]$.

De lo discutido anteriormente es posible extraer como conclusión que la función $f(x,y)$ para el caso de la suma acotada debe tomar valores negativos mientras que si se considera el producto acotado deberá ser positiva.

De acuerdo con nuestra estrategia de control de contraste el sistema que proponemos se basa en aplicar una máscara que recorre la imagen. En función del contraste local el sistema decide aplicar el operador más adecuado. Esta toma de decisión se realiza con el sistema difuso discutido en el apartado 3.3 cuya base de conocimiento se muestra en la figura 3.20. De acuerdo con esta estrategia el sistema global se compone de 3 motores de inferencia difusos como se ilustra en la figura 3.43. Los sistemas FIM1 y FIM2 generan las funciones de control de contraste asociadas a la suma acotada y el producto acotado respectivamente. El sistema FIM3 corresponde al sistema de toma de decisión que selecciona la fuente del valor de salida. Finalmente es posible añadir un parámetro

adicional C que permita al usuario realizar un control específico. De esta manera la funcionalidad del sistema viene dada por la siguiente expresión:

$$x' = \begin{cases} x \oplus y \oplus f(x,y) \oplus C & \text{si } z = Z1 \\ x & \text{si } z = Z2 \\ x \otimes y \otimes f(x,y) \otimes C & \text{si } z = Z3 \end{cases}$$

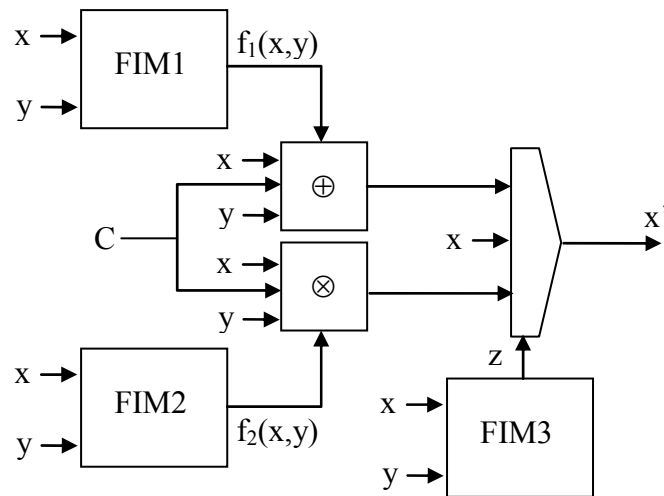


Figura 3.43. Esquema del sistema de control de contraste.

En la figura 3.44 se ilustra el resultado de la aplicación del operador suma acotada (figura 3.44b) y el resultado del sistema de control difuso (figura 3.44c).

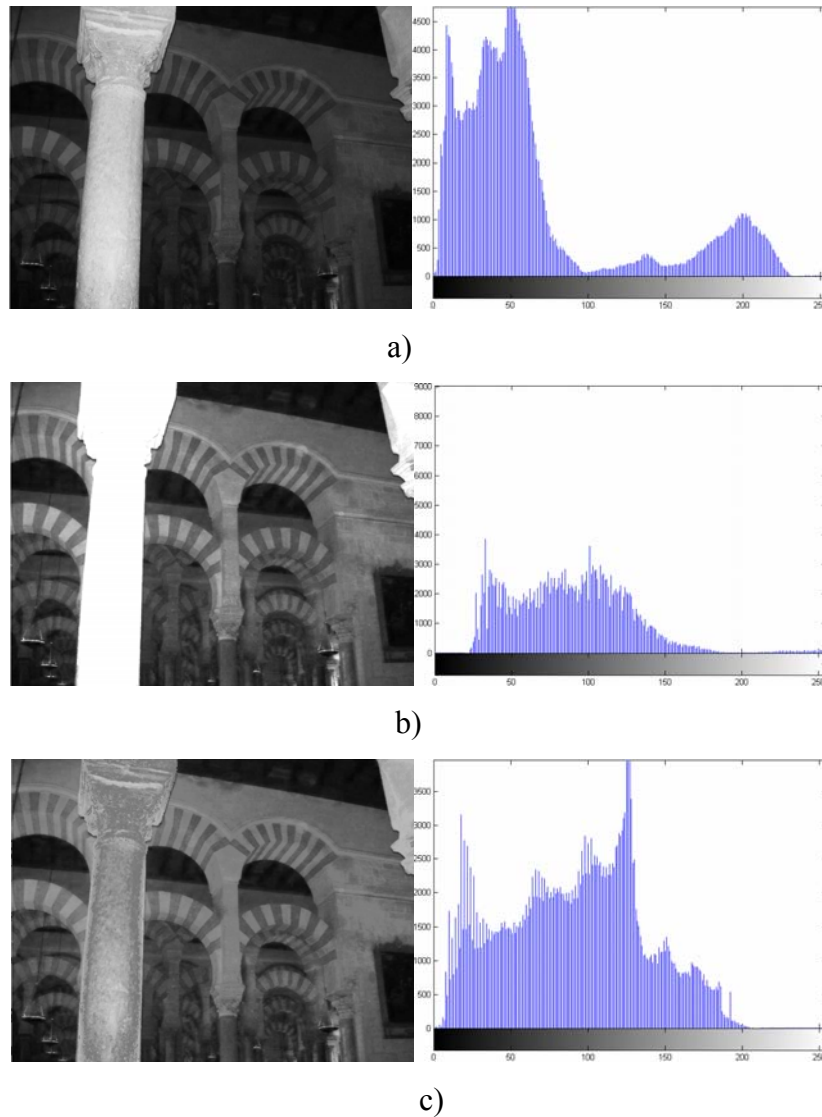


Figura 3.44. a) Imagen original, b) $x \oplus y$, c) sistema de control difuso de la figura 3.43.

3.6. Aplicación de la lógica de Łukasiewicz a la aproximación de funciones lineales a tramos

Una aplicación interesante de los operadores de Łukasiewicz consiste en la interpolación lineal a tramos de funciones. La interpolación lineal de funciones tiene aplicaciones en muchas técnicas de procesamiento de imágenes. En este apartado vamos a mostrar un mecanismo que permite realizar la interpolación de funciones lineales a tramos mediante estos operadores.

Una primera aproximación al problema de la interpolación lineal de funciones se establece en [OVCH02] donde se especifica que una función lineal a tramos y el conjunto de sus distintos componentes ($\{g_1, \dots, g_n\}$), puede describirse mediante la siguiente expresión:

$$f(x) = \bigvee_{j \in J} \bigwedge_{i \in S_j} g_i(x) \quad \forall x$$

donde los elementos de la familia $\{S_j\}_{j \in J}$ son subconjuntos incomparables (respecto a \subseteq) de $\{1, \dots, n\}$.

Con objeto de ilustrar la aproximación de funciones mediante los operadores de Łukasiewicz vamos a considerar dos casos. El primero corresponde a una función de una variable y, a continuación, se tratará el caso de una función de dos variables.

3.6.1. Funciones de una variable

Sea la función lineal a tramos que se muestra en la figura 3.45. Dicha función viene descrita por la expresión siguiente:

$$f_1(x) = \begin{cases} g_1 = 0 & x < 3 \\ g_2 = x - 3 & 3 < x < 5 \\ g_3 = \frac{1}{3}x + \frac{1}{3} & 5 < x < 8 \\ g_4 = -\frac{3}{2}x + 15 & 8 < x < 10 \\ g_5 = 0 & x > 10 \end{cases}$$

Una realización directa de esta función a partir de la expresión anterior da lugar a la técnica de aproximación basada en *splines* de orden 1. En este caso cada tramo viene determinado por la recta $g_i = m_i x + n_i$.

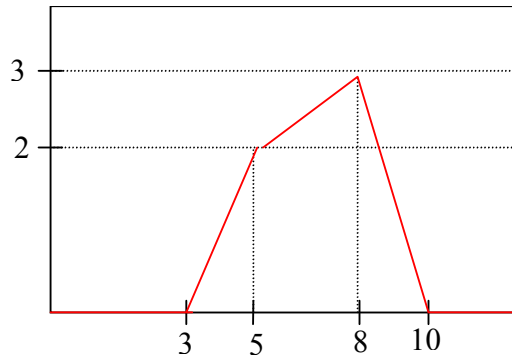


Figura 3.45. Ejemplo de función lineal a tramos.

De acuerdo con las dos estrategias de diseño empleadas en la sección 3.4 hemos obtenido diferentes implementaciones de la función $f_1(x)$. Una primera estrategia consiste en realizar la implementación de la función mediante una red neuronal. La figura 3.48a muestra el esquema de dicha red neuronal. Se trata de una red de tres capas. Ya vimos en la sección 3.4 que la función de activación de las neuronas viene dado por la expresión:

$$\psi(x) = 1 \wedge (x \vee 0)$$

Otro tipo de realización puede inferirse al aplicar el álgebra de Łukasiewicz [HUSS05a,b]. En este caso es posible obtener una expresión de la función haciendo uso de los operadores de Łukasiewicz. En efecto puede comprobarse que una función $g_i = m_i x + n_i$ puede expresarse como $g_i = (1 + n_i) \otimes m_i x$, por lo que la función anterior viene dada por la siguiente expresión:

$$f_1(x) = \begin{cases} g_1 = 0 & x < 3 \\ g_2 = -2 \otimes x & 3 < x < 5 \\ g_3 = \frac{4}{3} \otimes \frac{1}{3} x & 5 < x < 8 \\ g_4 = 16 \otimes -\frac{3}{2} x & 8 < x < 10 \\ g_5 = 0 & x > 10 \end{cases}$$

Por otro lado, la función $f_1(x)$ puede expresarse de la siguiente forma:

$$f_1(x) = (g_2 \wedge g_3 \wedge g_4) \vee 0 = (((g_2 \oplus \neg g_3) \otimes g_3) \oplus \neg g_4) \oplus g_4$$

donde los términos g_2 , g_3 y g_4 vienen dadas por las funciones descritas en la expresión previa.

La figura 3.46 muestra la descripción funcional VHDL de la función $f_1(x)$. Para realizar la función se ha aplicado aritmética en punto fijo y representación de los números signo en complemento a 2. En este caso se ha considerado una precisión de 4 bits para la entrada. Dicha entrada corresponde a un número entero entre 0 y 15. La salida se codifica con 8 bits de los que los cuatro más significativos corresponden a la parte entera y los cuatro menos significativos a la parte decimal.

De esta manera la constante $1/3$ se codifica como “0.0101” y la constante $-3/2$ corresponde al valor binario “10.1000” que corresponde a un número negativo en complemento a 2.

Se observa que el producto acotado se ha descrito mediante la función mostrada en la figura 3.47. La figura 3.48b muestra el circuito que se obtiene al realizar la síntesis de la expresión anterior con XST sobre un FPGA de Xilinx.

```

entity F1 is
  port(x    : in  std_logic_vector(3 downto 0);
        Fout: out std_logic_vector(7 downto 0));
end F1;

architecture funcional of F1 is
  constant untercio: std_logic_vector(4 downto 0):="00101";
  constant menostresmedios: std_logic_vector(5 downto 0):="101000";
begin
  process(x)
    variable dato : integer range 0 to 15;
    variable sal: std_logic_vector(7 downto 0);
    variable sal1: std_logic_vector(9 downto 0);
    variable sal2: std_logic_vector(10 downto 0);
  begin
    dato := CONV_INTEGER('0'&x);
    if dato>=3 and dato<=5 then
      sal := Lprod("1111100000","00"&x&"0000");
    elsif dato>=5 and dato<=8 then
      sal1:=untercio*('0'&x);
      sal := Lprod("0000010101",sal1);
    elsif dato>=8 and dato<=10 then
      sal2 := menostresmedios*('0'&x);
      sal := Lprod("0100000000",sal2(9 downto 0));
    else
      sal := "00000000";
    end if;
    Fout <= sal;
  end process;
end funcional;

```

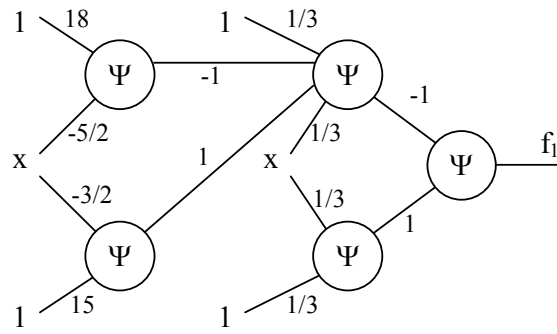
Figura 3.46. Descripción funcional VHDL de la función $f_1(x)$.

```

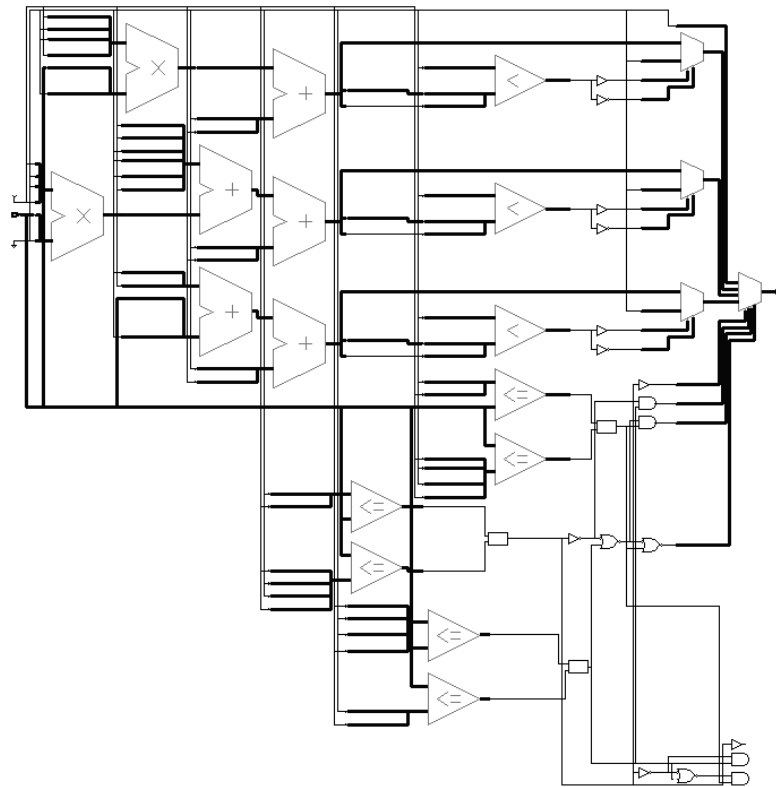
function Lprod(x,y: std_logic_vector(9 downto 0)) return std_logic_vector is
  variable suma: std_logic_vector(9 downto 0);
begin
  suma := x + y + "111110000";
  if CONV_INTEGER(suma(7 downto 4))>0 then
    return suma(7 downto 0);
  else
    return "00000000";
  end if;
end Lprod;

```

Figura 3.47. Función VHDL para el producto acotado.



(a)



(b)

Figura 3.48. Realizaciones de la función $f_1(x)$ basada en (a) red neuronal, (b) operadores de Łukasiewicz.

3.6.2. Funciones de más variables

Vamos a considerar un ejemplo de una función con dos variables descrita en [AMAT02] que tiene la expresión siguiente:

$$f_2(x) = (3x \wedge 1) \wedge ((x + y) \wedge 1)$$

Podemos describir la función $f_2(x)$ mediante los operadores de Łukasiewicz:

$$F_2(x) = [(x \oplus y) \oplus \{-3x \otimes 1\} \oplus 0] \otimes [(3x \oplus 0) \otimes 1]$$

La superficie correspondiente a esta función se muestra en la figura 3.49. De nuevo hemos realizado el diseño de dicha función mediante circuitos sobre FPGA empleando las dos aproximaciones (red neuronal y operadores de Łukasiewicz).

La figura 3.50a muestra el esquema correspondiente a la red neuronal. Se trata de una red de dos capas con cuatro neuronas. Dicha red neuronal se ha obtenido a partir de la siguiente expresión de la función $f_2(x)$:

$$f_2(x) = 1 - \psi(1 - x - y) - \psi(-(\psi(1 - x - y) + \psi(1 - 3x)))$$

La figura 2.50b muestra el esquema del circuito obtenido al sintetizar la expresión de la función $f_2(x)$ empleando los operadores de Łukasiewicz.

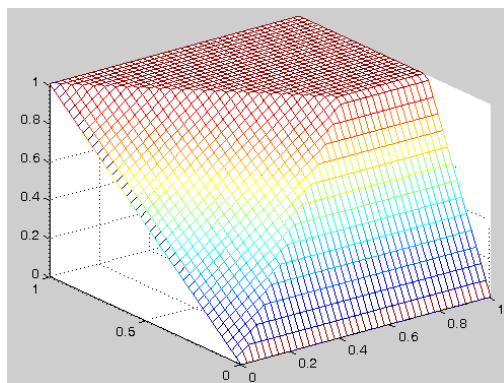


Figura 3.49. Superficie correspondiente a la función $f_2(x)$.

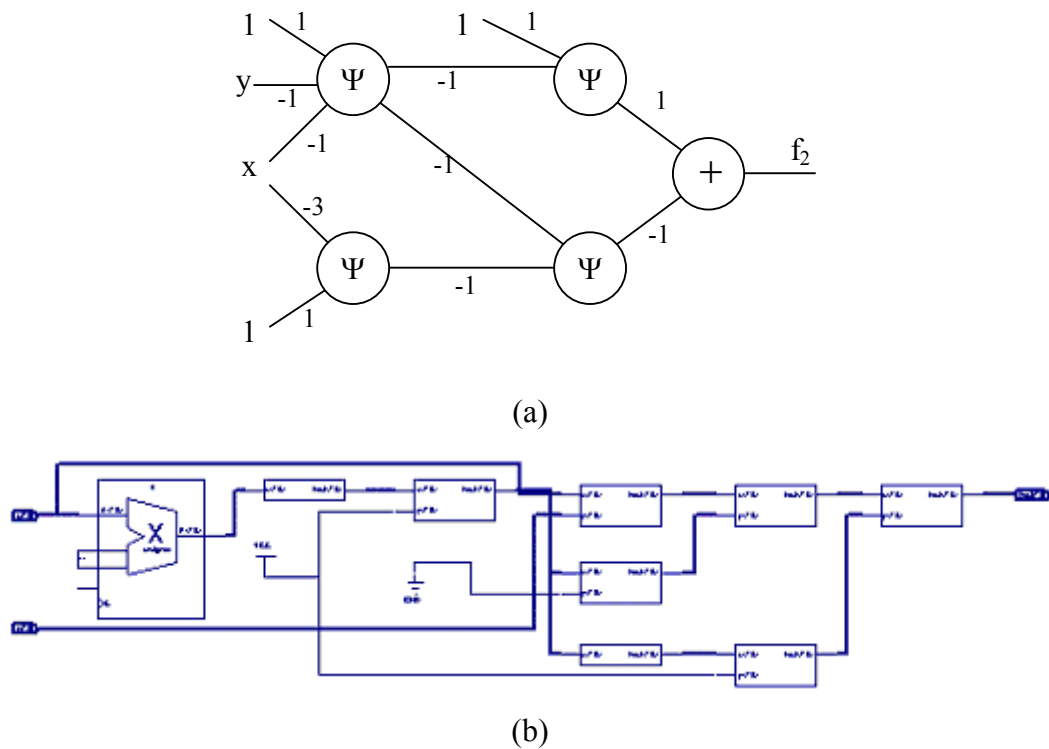


Figura 3.50. Realización de $f_2(x)$ mediante (a) una red neuronal, (b) operadores de Łukasiewicz basados en lógica combinacional.

3.6.3. Comparación entre técnicas de realización

En este apartado vamos a analizar las características de las diferentes implementaciones de las funciones $f_1(x)$ y $f_2(x)$. Estos resultados nos pueden dar pistas sobre cual debe ser la estrategia de realización hardware más adecuada para realizar la interpolación de funciones lineales a tramos. La tabla 3.3 muestra los resultados referentes al área ocupada por la realización de las funciones $f_1(x)$ y $f_2(x)$ sobre FPGA de Xilinx.

En el caso de la función $f_1(x)$ se ha considerado una precisión de 4 bits para la entrada. Dicha entrada corresponde a un número entero. La salida se codifica con 8 bits de los que los cuatro más significativos corresponden a la parte entera y los menos significativos a la parte decimal. El coste de área se expresa en términos de *slices* ocupados en el FPGA.

	Función $f_1(x)$	Función $f_2(x)$
Red neuronal	162	36
Oper. Łukasiewicz	25	32

Tabla 3.3. Coste en *slices* de las realizaciones sobre FPGA de las funciones $f_1(x)$ y $f_2(x)$.

De la observación de la tabla 3.3 podemos observar que las realizaciones basadas en redes neuronales requieren más recursos hardware que la aplicación de los operadores de Łukasiewicz como bloques lógicos combinacionales. Ello se debe a que en el caso de las redes neuronales se requiere más multiplicadores (sobre todo en el ejemplo de la función $f_1(x)$).

En la tabla 3.4 se muestran los retrasos máximos de las realizaciones anteriores. Estos retrasos no indican la frecuencia máxima de operación del sistema. Observamos que la el circuito que aproxima la función $f_1(x)$ puede operar a una frecuencia de 18 MHz en el caso de una realización mediante red neuronal o 47 MHz en el caso de emplear los operadores de Łukasiewicz. Por otro lado el circuito que aproxima la función $f_2(x)$ opera a 38 MHz en el caso de red neuronal y 34 MHz en el caso del uso de operadores de Łukasiewicz.

	Función $f_1(x)$	Función $f_2(x)$
Red neuronal	55 ns	26 ns
Oper. Łukasiewicz	21 ns	29 ns

Tabla 3.4. Retraso máximo (en nseg.) en las realizaciones sobre FPGA de las funciones $f_1(x)$ y $f_2(x)$.

3.7. Resumen

En este capítulo se ha presentando una técnica para el control del contraste en imágenes basada en la aplicación de operadores del álgebra de Lukasiewicz. En concreto se hace uso de los operadores suma-acotada y producto-acotado. Estos operadores dan lugar a una transformación de la distribución de los niveles de luminancia en el histograma de la imagen. Dicha transformación consiste en un desplazamiento y expansión de la distribución de los niveles de luminancia. El control del contraste aplicando la suma acotada o el producto acotado se realiza introduciendo un parámetro adicional que permita regular el desplazamiento de la frecuencia.

La técnica que se ha propuesto para el control del contraste es una técnica local que modifica la imagen punto a punto. El proceso de transformación se realiza aplicando una máscara que recorre la imagen. También se ha considerado un mecanismo que selecciona el parámetro de control aplicando un sistema basado en lógica difusa. El motor de inferencia difuso toma la decisión del valor que debe tener el parámetro de control dependiendo del contraste local de la máscara considerada. Esto permite adaptar el contraste para cada región de la imagen de manera independiente.

El principal objetivo del algoritmo de control de contraste que se propone es poder generar el circuito con restricciones de coste y alta velocidad de procesado. La implementación hardware de los operadores suma acotada y producto acotado se ha realizado siguiendo dos estrategias de diseño: una estrategia basada en redes neuronales y otra basada en lógica combinatorial.

Capítulo 4

Segmentación de imágenes

La segmentación de imágenes constituye una etapa en muchos procesos de análisis de imágenes. Mediante la segmentación se divide la imagen en las partes u objetos que la constituyen. Para ello la imagen se divide en regiones de las que se extraen una serie de parámetros que permiten clasificar las distintas regiones de la imagen en los distintos objetos que la constituyen. En el caso de considerar una única región la imagen se divide en objeto y fondo. El nivel al que se realiza esta subdivisión depende de la aplicación en particular, es decir, la segmentación terminará cuando se hayan detectado todos los objetos de interés para la aplicación. La segmentación es un paso imprescindible en diversos procesos de tratamiento de imagen. Entre otros, es necesaria para tomar medidas sobre una región, para realizar reconstrucciones tridimensionales de una zona de la imagen, para la clasificación o diagnóstico automático o para reducir la información de las imágenes.

Este capítulo se organiza en cuatro apartados. En el primer apartado se describen las técnicas de segmentación de imágenes. Nuestro interés se centra en las basadas en el cálculo de umbral. Por ello en el segundo apartado se describen los métodos basados en umbral. A continuación se presenta la técnica que se propone. Finalmente, en el último apartado se describe el diseño e implementación del circuito de umbralización.

4.1. Segmentación de imágenes

Los algoritmos de segmentación de imagen generalmente se basan en dos propiedades básicas de los niveles de luminancia de la imagen: discontinuidad y similitud. Dentro de la primera categoría se intenta dividir la imagen basándonos en los cambios bruscos en el nivel de luminancia. Un ejemplo dentro de esta categoría es la detección de puntos, detección de líneas y detección de bordes en la imagen. Dentro de la segunda categoría se aplican técnicas de umbrales, crecimiento de regiones, y técnicas de división y fusión.

4.1.1. Segmentación por discontinuidades

Entre las técnicas basadas en los cambios bruscos en el nivel de luminancia o técnicas basadas en discontinuidad está la técnica de detección de puntos. Esta técnica se basa en el uso de una máscara que recorre la imagen con objeto de detectar las discontinuidades:

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

El vector de productos puede calcular como:

$$w'x = 8x_5 - (x_1 + x_2 + x_3 + x_4 + x_6 + x_7 + x_8 + x_9)$$

En una zona de nivel de luminancia constante el resultado de esta operación sería 0. Por otro lado, si la máscara se centra en un punto aislado (x_5), cuya intensidad es mayor que el del fondo, entonces el resultado sería mayor que 0, [GONZ87].

$$|w'x_9| > T$$

Donde el T es un valor no es negativo denominado umbral.

La detección de punto no tiene en cuenta la orientación de los bordes de los objetos. Una manera de considerar la orientación de dichos bordes la proporciona el mecanismo de detección de líneas de una imagen. Para ello se consideran se consideran las máscaras siguientes:

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} \quad \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

Puede observarse que la respuesta máxima resulta cuando el borde coincide con una de las orientaciones determinadas por la máscara. De esta manera empleando una máscara 3x3 es posible detectar las 4 orientaciones siguientes: 0°, 45°, 90° y 135°.

4.1.2. Segmentación por similitud

El método de segmentación por regiones consiste en dividir la imagen en regiones uniformes. Un algoritmo que realiza esta subdivisión es el siguiente:

1. Sea I_l la región de la imagen completa.
2. Seleccionamos una condición P.
3. Para toda región R_i : Si $P(R_i) = \text{FALSO}$ entonces Subdividir R_i en cuatro regiones disjuntas.
4. Repetir 3 hasta que no existan más regiones que dividir.

La aplicación del algoritmo se ilustra en la figura 4.1a. Para la representación de las sucesivas subdivisiones se utiliza un árbol cuaternario. La figura 4.1b muestra dicha representación para el ejemplo de la figura 4.1a. La raíz del árbol corresponde a la imagen completa I, y cada nodo identifica a cada una de las subdivisiones.

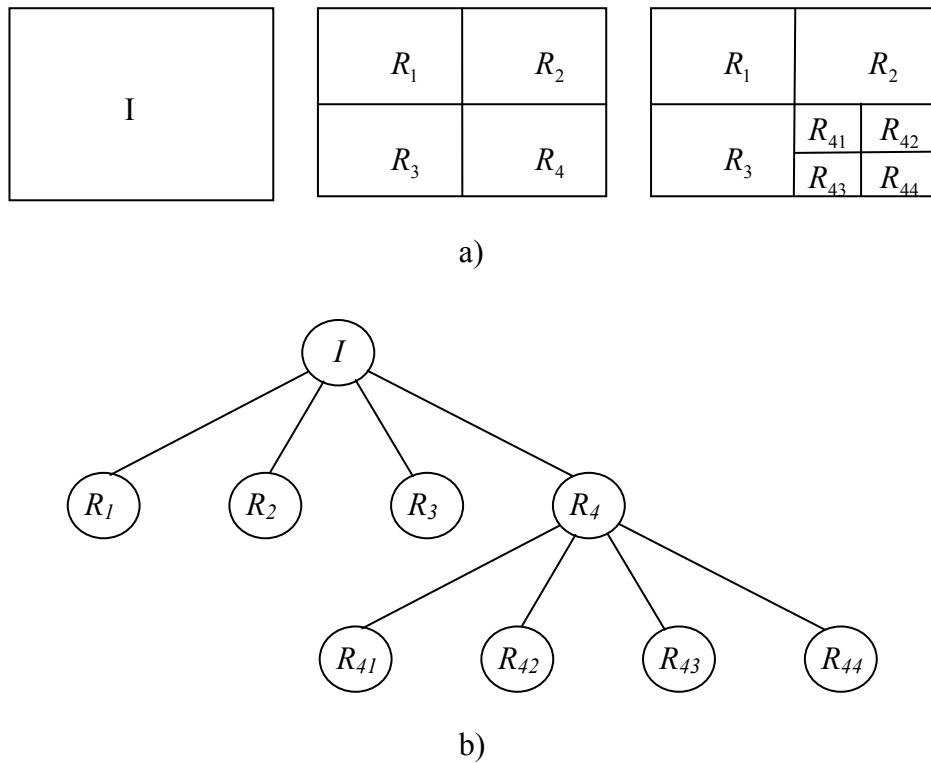


Figura 4.1. Ejemplo del algoritmo de segmentación por regiones.

Las técnicas de segmentación basada en agrupamientos clasifican los píxeles estadísticamente, sin tener en cuenta su situación espacial. Es decir no emplean información de regiones o de bordes sino sólo información de la intensidad o nivel de luminancia de cada punto. El proceso de segmentación por agrupación consiste en la división de los datos en grupos de objetos similares. Para medir la similitud entre objetos se suelen utilizar diferentes formas de distancia: distancia euclídea, distancia de Manhattan, distancia de Minkowski, etc. En el ejemplo de la figura 4.2 se puede extraer tres grupos en la imagen.

El problema más sencillo de segmentación se presenta cuando la imagen está formada por un sólo objeto que tiene intensidad luminosa homogénea sobre un fondo con un nivel de intensidad diferente. En este caso la imagen se puede segmentar en dos regiones utilizando una técnica basada en un parámetro umbral. Esta técnica de aplicar un valor de umbral es un método sencillo pero eficiente de separar los objetos del fondo. Ejemplos de aplicaciones en los que se emplea la umbralización son el análisis de documentos, procesado de mapas, imágenes de endoscopia, extracción de bordes, detección de objetos, etc. La elección del valor del umbral se puede hacer a partir del

histograma. Si el fondo tiene también intensidad luminosa homogénea, entonces el histograma es bimodal y el umbral que se debe tomar es el que corresponde al mínimo local que está entre los dos máximos del histograma. En ocasiones no se distinguen bien las dos zonas modales (los máximos) y se debe aplicar una técnica de variación espacial del umbral.

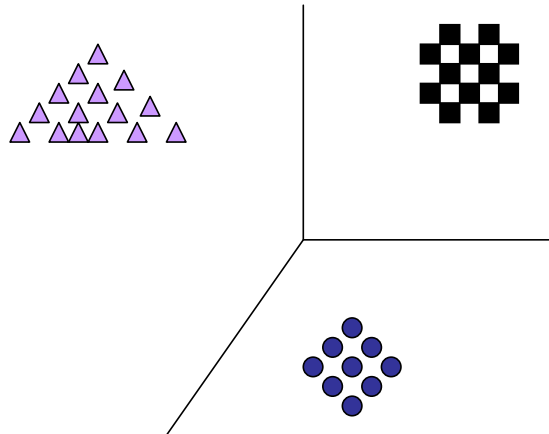


Figura 4.2. Clasificación en 3 grupos de los píxeles de una imagen.

Muchos de los algoritmos de umbralización se basan en una umbralización binaria. Los procedimientos de umbralización binario pueden extenderse a multi-nivel disponiendo de múltiples umbrales T_1, T_2, \dots, T_n para segmentar la imagen en $n+1$ regiones [LIAO01, CAO02, OH06]. La umbralización multi-nivel basada en un histograma multidimensional aplica técnicas de crear agrupamientos (*pattern clustering*) para realizar la segmentación de imágenes.

Entre las técnicas de umbralización las que se basan en aplicar lógica difusa permiten resolver el problema de la información imprecisa de la imagen [FORE00, HUAN95, KAUF75, YAGE79].

4.2. Métodos de umbralización

Las técnicas basadas en umbralizar una imagen permiten clasificar los píxeles en dos categorías (imagen en blanco y negro). Esta transformación se realiza para

establecer una distinción entre los objetos de la imagen y el fondo. El modo de generar esta imagen binaria es realizando la comparación de los valores de los píxeles con un umbral T . Así sea G el conjunto de valores de los píxeles de la imagen:

$$G = \{0,1,\dots,L-1\} \left\{ \begin{array}{ll} 0 & \text{negro} \\ L-1 & \text{blanco} \end{array} \right\}$$

Se define un umbral $T \in G$ de manera que se realiza la transformación que viene dada de la siguiente manera:

$$y_{i,j} = \begin{cases} 0 & \text{si } x_{i,j} < T \\ L-1 & \text{si } x_{i,j} > T \end{cases} \quad (1)$$

donde $x_{i,j}$ es un píxel de la imagen original e $y_{i,j}$ es el píxel correspondiente a la imagen binaria. En el caso de una imagen monocolor en la que los píxeles se codifican con 8 bits el rango de valores que toman los píxeles corresponde al rango entre 0 y 255 ($L=256$). Es usual expresar dicho rango en valores normalizados entre 0 y 1. La figura 4.3 muestra un ejemplo para el caso de la imagen de Lena.



Figura 4.3. a) Imagen de Lena, b) imagen binaria con $T=0.5$

Existen muchos métodos para seleccionar el valor del umbral T . En [SEZG04] se analizan 40 métodos de selección del umbral. Estos métodos se clasifican en seis categorías de acuerdo con la información que es empleada: métodos basados en la forma del histograma, agrupamientos, entropía, métodos basados en atributos de los objetos, métodos espaciales y locales.

Los métodos de cálculo del umbral más extendidos son los basados en el análisis del histograma. El histograma es una relación entre los niveles de grises de una imagen con la frecuencia de los valores de grises de la imagen. La figura 4.4 muestra el histograma de la imagen de Lena (figura 4.3a).

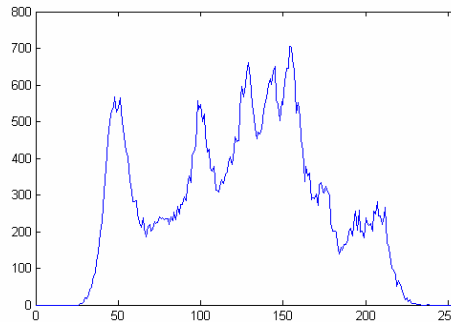


Figura 4.4. Histograma de la imagen de Lena.

4.2.1. Método basado en la frecuencia de nivel de gris

Una técnica básica de cálculo del umbral se basa en la frecuencia de nivel de gris. En este caso el umbral T se calcula mediante la siguiente expresión:

$$T = \sum_{i=1}^L p_i i \quad (2)$$

donde i es el nivel de gris que toma valores entre 1 y 256 para el caso de una codificación con 8 bits, p_i representa la frecuencia de nivel de gris (también denominada probabilidad de nivel de gris). Para una imagen con n pixels y n_i pixels con el nivel de gris i :

$$p_i = n_i / n \quad \text{y} \quad \sum_{i=1}^L p_i = 1 \quad (3)$$

4.2.2. Método de Otsu

La técnica de Otsu [OTSU78] calcula el umbral óptimo maximizando la varianza entre clases. Para ello realiza una búsqueda exhaustiva para evaluar el criterio de maximizar la varianza entre clases. Un inconveniente del método de Otsu es el tiempo requerido para seleccionar el valor del umbral.

En el caso de la umbralización en dos niveles los píxeles se clasifican en dos clases: C_1 , con niveles de gris $[1, \dots, t]$; y C_2 , con niveles de gris $[t+1, \dots, L]$. Entonces, la distribución de probabilidad de los niveles de gris para las dos clases es:

$$C_1 : \frac{p_1}{w_1(t)}, \dots, \frac{p_t}{w_1(t)} \quad (4)$$

$$C_2 : \frac{p_{t+1}}{w_2(t)}, \frac{p_{t+2}}{w_2(t)}, \dots, \frac{p_L}{w_2(t)} \quad (5)$$

donde

$$w_1(t) = \sum_{i=1}^t p_i \quad w_2(t) = \sum_{i=t+1}^L p_i$$

Las medias para las clases C_1 y C_2 son

$$\mu_1 = \sum_{i=1}^t \frac{ip_i}{w_1(t)} \quad \mu_2 = \sum_{i=t+1}^L \frac{ip_i}{w_2(t)}$$

Sea μ_T la intensidad media de toda la imagen. Es fácil demostrar que

$$w_1\mu_1 + w_2\mu_2 = \mu_T \quad w_1 + w_2 = 1$$

Usando análisis discriminante Otsu definió la variancia entre clases de una imagen umbralizada como

$$\sigma^2_B = w_1(\mu_1 - \mu_T)^2 + w_2(\mu_2 - \mu_T)^2 \quad (6)$$

Para una umbralización de dos niveles Otsu verificó que el umbral óptimo t^* se elige de manera que σ_B^2 sea máxima; esto es

$$t^* = \max_t \{\sigma_B^2(t)\} \quad 1 \leq t \leq L \quad (7)$$

El método de Otsu puede extenderse fácilmente a múltiples umbrales. Asumiendo que hay $M-1$ umbrales, $\{t_1, t_2, \dots, t_{M-1}\}$, los cuales dividen a la imagen en M clases:

C_1 para $[1, \dots, t_1]$, C_2 para $[t_1 + 1, \dots, t_2]$, ... , C_i para $[t_{i-1} + 1, \dots, t_i]$, ... y C_M para $[t_{M-1}, \dots, L]$,

Los umbrales óptimos $t_1^*, t_2^*, \dots, t_{M-1}^*$ se eligen maximizando σ_B^2 :

$$\{t_1^*, t_2^*, \dots, t_{M-1}^*\} = \max_{t_1, t_2, \dots, t_{M-1}} \{\sigma_B^2(t_1, t_2, t_{M-1})\} \quad (8)$$

$$1 \leq t_1 < \dots < t_{M-1} < L$$

donde
$$\sigma_B^2 = \sum_{k=1}^M w_k (\mu_k - \mu_T)^2 \quad (9)$$

con
$$w_k = \sum_{i=C_k} p_i \quad \mu_k = \sum_{i=C_k} \frac{ip_i}{w_k}$$

ω_k es conocido como momento acumulado de orden cero de la k -ésima clase C_k , y el numerador de la última expresión es conocido como momento acumulado de primer orden de la k -ésima clase C_k ; esto es,

$$\mu(k) = \sum_{i=C_k} ip_i$$

Independientemente del número de clases que se consideren durante el proceso de umbralización la suma de las funciones de probabilidad acumulada de las M clases son

iguales a 1 y la media de la imagen es igual a la suma de las medias de las M clases ponderadas por sus correspondientes probabilidades acumuladas, esto es,

$$\sum_{k=1}^M w_k = 1 \quad (10)$$

$$\mu_T = \sum_{k=1}^M w_k \mu_k \quad (11)$$

Usando las expresiones (10) y (11) la varianza entre clases en la ecuación (9) de la imagen umbralizada puede describirse de la siguiente forma

$$\sigma_B^2(t_1, t_2, \dots, t_{M-1}) = \sum_{k=1}^M w_k \mu_k^2 - \mu_T^2 \quad (12)$$

Debido a que el segundo término en la expresión (12) depende de la elección de los umbrales $\{t_1, t_2, \dots, t_{M-1}\}$ los umbrales óptimos $\{t_1^*, t_2^*, \dots, t_{M-1}^*\}$ pueden ser elegidos maximizando una varianza entre clase modificada $(\sigma_B')^2$, definida como la sumatoria de los términos del lado derecho de la expresión (12). En otras palabras, los valores de los umbrales óptimos $\{t_1^*, t_2^*, \dots, t_{M-1}^*\}$ se eligen por

$$\{t_1^*, t_2^*, \dots, t_{M-1}^*\} = \underset{t_1, t_2, \dots, t_{M-1}}{\text{Max}} \{\sigma_B^2(t_1, t_2, t_{M-1})\} \quad (13)$$

donde,
$$(\sigma_B')^2 = \sum_{k=1}^M w_k (\mu_k - \mu_T)^2 \quad (14)$$

De acuerdo al criterio de la expresión (8) para $(\sigma_B)^2$ y al de la expresión (13) para $(\sigma_B')^2$, para encontrar los umbrales óptimos el campo de búsqueda para el máximo $(\sigma_B)^2$ y para el máximo $(\sigma_B')^2$ es $1 < t_1 < L - M + 1, t_1 + 1 < t_2 < L - M + 2, \dots, y t_{M-1} + 1 < t_{M-1} < L - 1$

Esta búsqueda exhaustiva involucra $(L - M + 1)^{M-1}$ combinaciones posibles. Además, comparando la expresión (14) con la (9), encontramos que la resta en la

expresión (9) no es necesaria. Así, la expresión (14) es mejor que la expresión (9) ya que elimina $M(L - M + 1)^{M-1}$ restas del cálculo de los umbrales.

4.2.3. Métodos basados en lógica difusa

Las técnicas que aplican lógica difusa para el cálculo del umbral se basan principalmente en tres tipos de medidas de vaguedad [FORE00]: entropía, medida de Kaufmann y medida de Yager.

La técnica basada en la entropía consiste en minimizar la dispersión del sistema. Así los píxeles de la imagen se agrupan en dos clases correspondientes a los objetos y al fondo. Huang y Wang [HUAN95] consideran la media de los datos correspondientes a cada clase es μ_0 y μ_1 . La función de pertenencia de cada clase viene definida por:

$$u_x(x) = \begin{cases} \frac{1}{1 + \frac{|x - \mu_0|}{x_{\max} - x_{\min}}} & \text{si } x < T \\ \frac{1}{1 + \frac{|x - \mu_1|}{x_{\max} - x_{\min}}} & \text{si } x > T \end{cases}$$

El cálculo del umbral T se basa en la entropía de un conjunto difuso, que se calcula usando la función de Shannon:

$$H_f(x) = -x \log x - (1 - x) \log(1 - x)$$

El umbral será aquél que minimice la entropía de los datos:

$$E(T) = \frac{1}{M} \sum_i H_f(\mu_x(i))h(i)$$

La medida de Kaufmann se define como [KAUF75]:

$$D(A) = \left[\sum_{x \in X} |\mu_A(x) - \mu_C(x)|^w \right]^{\frac{1}{w}}$$

Este método se basa en usar la distancia al conjunto A como métrica. Cuando $w=1$ se utiliza la distancia de Hamming mientras que si $w=2$ se trata de la distancia Euclídea.

El método de Yager [YAGE79] se basa en la distancia entre un conjunto difuso y su complementario. Así se basa en minimizar la siguiente función:

$$D_2(T) = \sqrt{\sum_i |\mu_x(i) - \mu_x^-(i)|^2}$$

donde $\mu_x^-(i) = 1 - \mu_x(i)$.

4.3. Cálculo del umbral mediante lógica difusa

La técnica que se propone en este trabajo consiste en aplicar lógica difusa en el cálculo del umbral T . Básicamente, desde un punto de vista formal, esta técnica se basa en el cálculo de la media aplicada al histograma de la imagen. Un aspecto ventajoso de esta técnica es que el mecanismo de cálculo mejora los tiempos de cómputo ya que tan sólo requiere recorrer una vez la imagen y permite calcular de manera directa el valor del umbral. Desde el punto de vista de la realización hardware se dispone de una arquitectura de elemento de procesado difuso de bajo coste que permite realizar la inferencia tal y como se verá en una próxima sección.

El sistema difuso tiene una entrada que corresponde al píxel que se va a evaluar y una salida que corresponde al resultado de la inferencia difusa. Una vez leída la imagen completa la salida muestra el valor del umbral T . Básicamente la operación que realiza el sistema difuso corresponde al cálculo del centro de gravedad del histograma de la imagen de acuerdo con la siguiente expresión:

$$T = \frac{\sum_{i=1}^M \sum_{j=1}^R \alpha_{ij} c_{ij}}{\sum_{i=1}^M \sum_{j=1}^R \alpha_{ij}} \quad (16)$$

donde T es el umbral, M es el número de píxeles de la imagen, R es el número de reglas del sistema difuso, c es el consecuente de cada regla y α es el grado de activación de la regla.

Para realizar la inferencia difusa se realiza un particionado del universo de discurso del histograma en un conjunto de N funciones de pertenencia equidistribuidas. La figura 4.5 muestra un ejemplo de particionado para $N=9$. Se han empleado funciones de pertenencia triangulares ya que son más fácilmente implementables en hardware. Dichas funciones tienen un solapamiento de 2 lo que permite limitar el número de reglas activas.

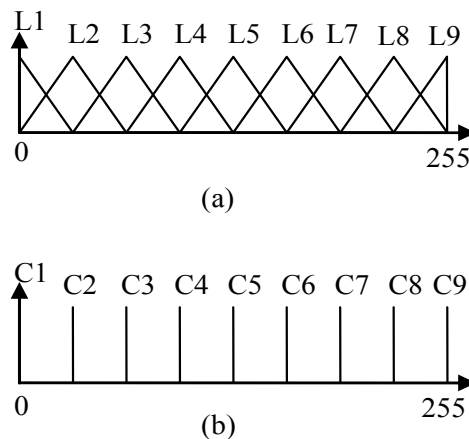


Figura 4.5. Funciones de pertenencia para $N=9$, a) antecedente, b) consecuente.

Las funciones de pertenencia del consecuente son funciones de tipo *singleton* equidistribuidas en el universo de discurso del histograma. El uso de funciones de pertenencia de tipo *singleton* para el consecuente permite aplicar métodos de defuzzificación simplificados como es el caso de la media difusa. Este método de defuzzificación puede interpretarse como aquél en el que cada regla propone una conclusión con un determinado “peso” definido por su grado de activación. La acción

global de las reglas se obtiene calculando el promedio de las diferentes conclusiones ponderadas por sus grados de activación. Estas características de procesamiento basado en reglas activas y método de defuzzificación simplificados permiten obtener realizaciones hardware de bajo coste y alta velocidad de operación.

La base de reglas del sistema de la figura 4.6 emplea las funciones de pertenencia mostradas en la figura 4.5. Observamos que el número de reglas viene dado por el número de funciones de pertenencia. La base de conocimiento (funciones de pertenencia y base de reglas) es común para todas las imágenes por lo que los valores pueden almacenarse en una memoria de tipo ROM.

if x is L1 then c is C1;
if x is L2 then c is C2;
if x is L3 then c is C3;
if x is L4 then c is C4;
if x is L5 then c is C5;
if x is L6 then c is C6;
if x is L7 then c is C7;
if x is L8 then c is C8;
if x is L9 then c is C9;

Figura 4.6. Base de reglas para $N=9$.

Es posible optimizar la expresión de la ecuación (4) si el sistema está normalizado. En este caso la suma extendida a la base de reglas de los grados de activación toma el valor 1 ($\sum_{j=1}^R \alpha_{ij} = 1$). Por lo tanto la ecuación (4) se transforma en:

$$T = \frac{1}{M} \sum_{i=1}^M \sum_{j=1}^R \alpha_{ij} c_{ij} \quad (17)$$

El sistema evalúa en cada instante de tiempo un píxel y realiza la inferencia de acuerdo con la base de reglas de la figura 4.6. La salida del sistema acumula el resultado correspondiente al numerador de la ecuación (17). La salida final se genera con el último píxel de la imagen que permite realizar la división.

La figura 4.7 muestra una función en Matlab que calcula el valor del umbral.

```
% ANTECEDENTE
a=newfis('tipper');
a=addvar(a,'input','U',[0 255]);
a=addmf(a,'input',1,'U1','trimf',[0 0 31]);
a=addmf(a,'input',1,'U2','trimf',[0 31 63]);
a=addmf(a,'input',1,'U3','trimf',[31 63 95]);
a=addmf(a,'input',1,'U4','trimf',[63 95 127]);
a=addmf(a,'input',1,'U5','trimf',[95 127 159]);
a=addmf(a,'input',1,'U6','trimf',[127 159 191]);
a=addmf(a,'input',1,'U7','trimf',[159 191 223]);
a=addmf(a,'input',1,'U8','trimf',[191 223 255]);
a=addmf(a,'input',1,'U9','trimf',[223 255 255]);

% CONSECUENTE
a=addvar(a,'output','Z',[0 255]);
a=addmf(a,'output',1,'Z1','trimf',[0 0 1]);
a=addmf(a,'output',1,'Z2','trimf',[30 31 32]);
a=addmf(a,'output',1,'Z3','trimf',[62 63 64]);
a=addmf(a,'output',1,'Z4','trimf',[94 95 96]);
a=addmf(a,'output',1,'Z5','trimf',[126 127 128]);
a=addmf(a,'output',1,'Z6','trimf',[158 159 160]);
a=addmf(a,'output',1,'Z7','trimf',[190 191 192]);
a=addmf(a,'output',1,'Z8','trimf',[222 223 224]);
a=addmf(a,'output',1,'Z9','trimf',[254 255 255]);

% BASE DE REGLAS
ruleList=[
    1 1 1 1
    2 2 1 1
    3 3 1 1
    4 4 1 1
    5 5 1 1
    6 6 1 1
    7 7 1 1
    8 8 1 1
    9 9 1 1];
a=addrule(a,ruleList);

% INFERENCIA
Y=evalfis(img,a);
T=sum(Y)/(N*M);
```

Figura 4.7. Cálculo del umbral en Matlab.

4.4. Diseño del circuito para el cálculo del umbral

A la hora de implementar el sistema difuso que genera el umbral se ha empleado la metodología de diseño descrita en [BARR06]. Dicha metodología se basa en el lenguaje de descripción de hardware VHDL como la manera de describir y modelar el sistema en alto nivel. Con objeto de conseguir un modelado del comportamiento del módulo de inferencia difuso se ha optado por un estilo de descripción VHDL en el que se describen de manera independiente la estructura del sistema (conjuntos difusos, base de regla) y los operadores del sistema (conectivas, operaciones difusas). Esto permite independizar la descripción de la estructura del sistema de los mecanismos de procesado. La descripción del sistema difuso es sintetizable de manera que es posible generar la realización hardware haciendo uso de herramientas convencionales de síntesis de circuitos.

La figura 4.8 muestra la arquitectura VHDL del sistema difuso. La base de reglas se describe en el cuerpo de la arquitectura. Corresponde a la base de 9 reglas de la figura 4.6. Cada regla está constituida por dos componentes: antecedente de la regla y consecuente. El antecedente es una expresión de la variable de entrada relacionada con la etiqueta lingüística. El consecuente establece el valor lingüístico de la salida de la regla.


```

architecture knowledge base of threshold is
  signal R: consec;
  -- MF for x
  constant L1: triangle:=((0, 0, 31),(0, 32));
  constant L2: triangle:=((0, 31, 63),(32,32));
  constant L3: triangle:=((31, 63, 95),(32,32));
  constant L4: triangle:=((63, 95, 127),(32,32));
  constant L5: triangle:=((95,127, 159),(32,32));
  constant L6: triangle:=((127,159,191),(32,32));
  constant L7: triangle:=((159,191,223),(32,32));
  constant L8: triangle:=((191,223,255),(32,32));
  constant L9: triangle:=((223,255,255),(32,0));

  --MF for z
  constant C1: integer := 0;
  constant C2: integer := 31;
  constant C3: integer := 63;
  constant C4: integer := 95;
  constant C5: integer := 127;
  constant C6: integer := 159;
  constant C7: integer := 191;
  constant C8: integer := 223;
  constant C9: integer := 255;
begin
  R(1) <= rule( (x = L1), C1 );
  R(2) <= rule( (x = L2), C2 );
  R(3) <= rule( (x = L3), C3 );
  R(4) <= rule( (x = L4), C4 );
  R(5) <= rule( (x = L5), C5 );
  R(6) <= rule( (x = L6), C6 );
  R(7) <= rule( (x = L7), C7 );
  R(8) <= rule( (x = L8), C8 );
  R(9) <= rule( (x = L9), C9 );

  Zout<=defuzz(R);
end knowledge_base;

```

Figura 4.8. Arquitectura VHDL de la base de conocimiento.

El mecanismo de procesado de la operación difusa ‘*is*’ (=) y la inferencia ‘*then*’ (*rule(,)*) no están definidas en la descripción VHDL. Solamente se ha descrito la estructura de la base de conocimiento. De esta manera la descripción es de alto nivel ya que no asume ningún criterio específico de implementación. Tan solo describe la base de conocimiento en términos de comportamiento de la base de reglas.

Las etiquetas lingüísticas representan el rango de valores de los conjuntos difusos dentro del universo de discurso para las variables entrada y salida. Estas etiquetas se describen mediante funciones con objeto de procesar el grado de pertenencia de un cierto valor de la variable. Las funciones de pertenencia asociadas a las etiquetas lingüísticas pueden ser triangulares o trapezoidales. La zona declarativa de la arquitectura VHDL de la figura 4.8 muestra las definiciones de dichas funciones de pertenencia.

El tipo de dato *triangle* se encuentra definido en un paquete VHDL denominado “xfvhdfunc”. Dicho tipo contiene las definiciones de los puntos que definen un triángulo así como las pendientes. Por su parte la base de reglas expresa el conocimiento de la figura 4.6. La función *rule* también está definida en el paquete VHDL. Dicha función realiza la inferencia de la regla. El conjunto de reglas se evalúa de manera concurrente ya que las instrucciones de asignación de señal en el cuerpo de la arquitectura VHDL son concurrentes. El operador “=” también ha sido redefinido aprovechando las propiedades de sobrecarga de funciones en VHDL. La figura 4.9 muestra el paquete VHDL con las definiciones de los tipos de datos y las nuevas funciones.

```
PACKAGE xfvhdfunc IS
  type points is array (1 to 3) of integer;
  type slopes is array (1 to 2) of integer;

  type triangle is record
    point: points;
    slope: slopes;
  end record;
  . . .
  -- Consequents
  type two_int is array (0 to 1) of integer;
  type consec is array (1 to RULES) of two_int;

  -- FUNCTIONS
  function "=" (x: integer; y: triangle) return integer;
  . . .
  function rule (x: integer; y: integer) return two_int;
  function defuzz(x: consec) return integer;
END xfvhdfunc;
```

Figura 4.9. Paquete VHDL con las definiciones de los tipos de datos y de las funciones.

Las funciones empleadas en la descripción del sistema difuso han sido descritas de acuerdo con las restricciones de VHDL para síntesis. Esto ha permitido generar el circuito que implementa el sistema difuso haciendo uso de herramientas convencionales de síntesis de circuitos. El resultado corresponde a un circuito combinacional que realiza la inferencia difusa. La figura 4.10 muestra el símbolo del sistema de generación del umbral. Dicho sistema está compuesto por dos bloques (figura 4.10b): el módulo de inferencia difuso (FIM) y el circuito de división por M de acuerdo con la ecuación (17). El divisor es un módulo IP de [ASIC] diseñado por Richard Herveille. Nosotros hemos reescrito el código Verilog en VHDL debido a problemas con la herramienta de

simulación ModelSim. La figura 4.10c muestra el esquema del circuito FIM constituido por el motor de inferencia difuso y la etapa acumuladora que realiza la suma extendida a todos los píxeles de la imagen.

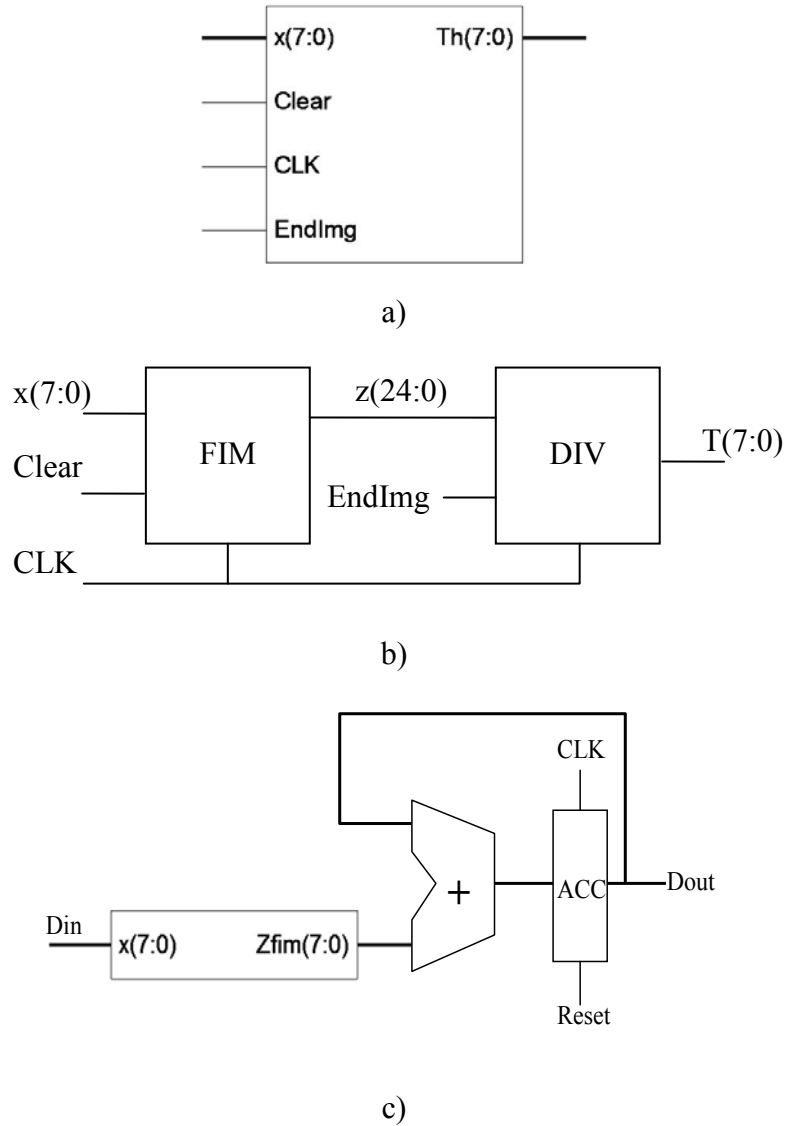


Figura 4.10. a) Símbolo del sistema de generación del umbral, b) Módulo de inferencia fuzzy (FIM) y divisor, c) bloque FIM con el motor de inferencia difuso y circuito acumulador.

El modulo de inferencia difuso junto con la etapa divisora generan el valor del umbral de la imagen. El circuito ha sido implementado sobre un dispositivo FPGA de la familia Spartan3 de Xilinx. El sistema de test se ha implementado sobre una placa de desarrollo Spartan3-Starter Board [XILI05]. La figura 4.11 muestra una fotografía de dicho sistema. El dispositivo FPGA contiene tanto la imagen que debe ser procesada, el circuito de generación de umbral y muestra el resultado sobre los displays 7-segmentos de la placa. El resultado del valor del umbral se muestra en hexadecimal. En la imagen se observa el caso del procesamiento de la imagen de Lena.

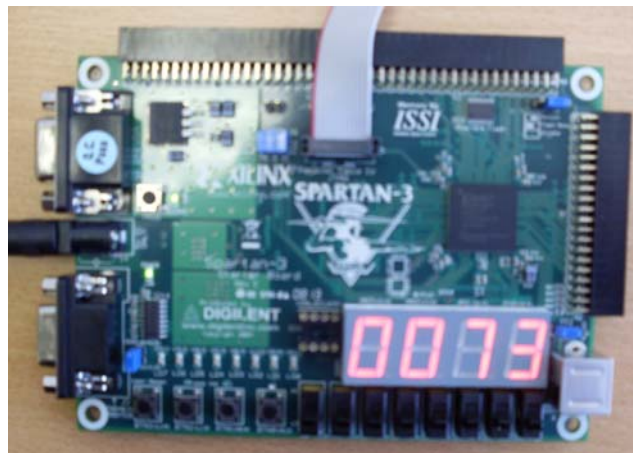


Figura 4.11. Sistema de test basado en una placa de desarrollo Spartan3-Starter Board.

La figura 4.12 muestra el esquema del sistema de test. El bloque UMBRAL lee la imagen de la memoria y genera el valor del umbral. Dicho valor se representa en hexadecimal sobre el display 7-segmentos de la placa de desarrollo mediante los valores LED y AN. Las señales div0 (que indica un error de división por cero) y ovf (que indica overflow) encienden dos diodos leds de la placa. La señal de reloj CLK proviene de un oscilador de 50 MHz. Por otro lado las señales Reset e Init son generadas por dos pulsadores. En nuestro sistema el tamaño de la memoria de imagen es de $2^{14}=16384$ bytes debido a restricciones de recursos e el dispositivo FPGA. Por lo tanto las imágenes que vamos a tratar van a ser de 128x128 píxeles.

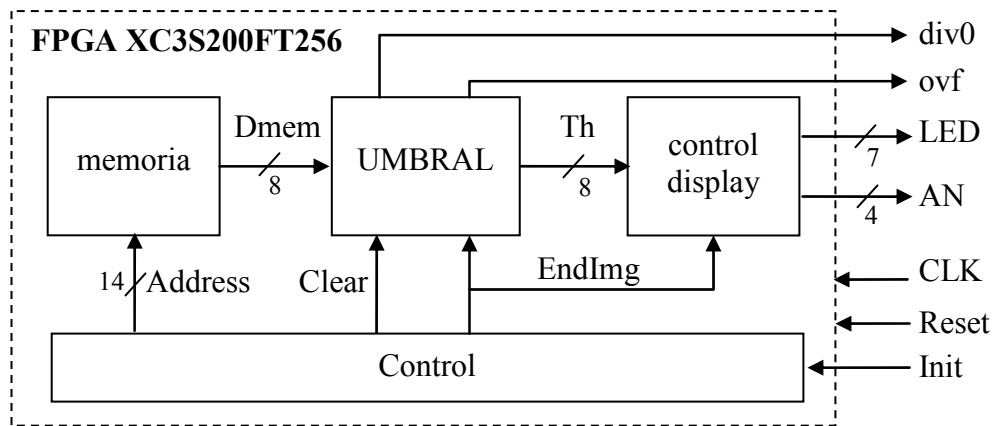
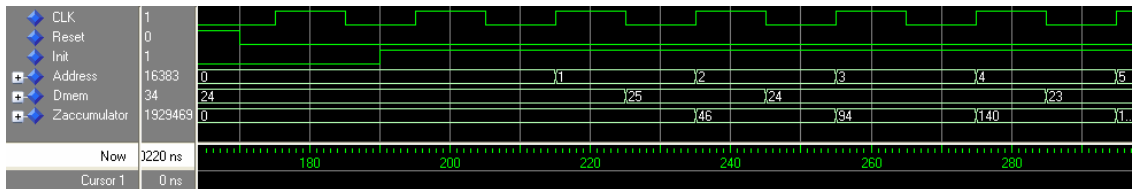


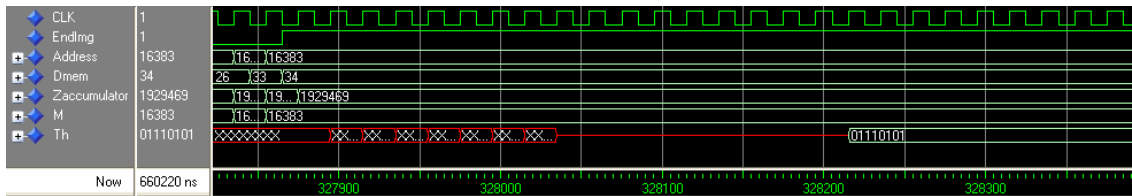
Figura 4.12. Sistema de test del circuito de cálculo del umbral.

La operación del sistema puede observarse de los cronogramas de la figura 4.13. En la figura 4.13a se muestra el comienzo de la operación del sistema. Tras el pulso de Reset se inicia la lectura de la memoria con la señal Init. En cada ciclo de reloj se lee una palabra de memoria y se realiza la inferencia. El resultado se acumula en el acumulador del circuito UMBRAL (ver figura 4.10). Una vez leída la imagen se activa la señal EndImg tal y como se muestra en la figura 4.13b. Ello inicia el proceso de división. La división requiere tantos ciclos de reloj como bits tenga el divisor. En nuestro caso se requieren 17 ciclos de reloj.

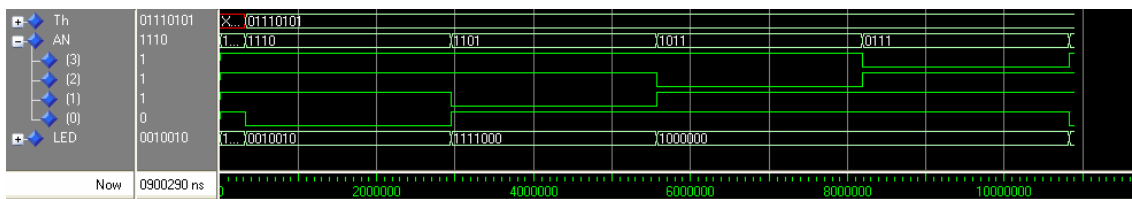
La última etapa del sistema de test corresponde a la representación del umbral en los display 7-segmentos. Para ello, de acuerdo con las especificaciones de los displays disponibles en la placa de desarrollo, el circuito de control de los displays genera las señales de alimentación de los ánodos de los diodos leds con una anchura de 2 ms. El proceso de escritura está multiplexado en el tiempo, tal y como se muestra en la figura 4.13c, ya que los datos de los cátodos son comunes para todos los displays.



a)



b)



c)

Figura 4.13. a) Inicio de la operación del sistema con la lectura de la imagen de memoria, b) Generación del resultado final con la división. c) Representación de la salida en los displays 7-segmentos.

El área ocupada sobre un FPGA de la familia Spartan3 de Xilinx ha sido de 1.070 *slices* que equivalen a un total de 49.799 puertas equivalentes. La figura 4.14a muestra los resultados de la utilización de recursos en el FPGA del circuito de generación del umbral. Por otro lado la figura 4.14b muestra los resultados correspondientes al sistema de test completo. Puede observarse que la circuitería adicional supone un aumento poco significativo en cuanto a recursos salvo en lo que respecta a los requerimientos de la memoria de la imagen.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	969	3,840	25%	
Number of 4 input LUTs	1,438	3,840	37%	
Logic Distribution				
Number of occupied Slices	1,070	1,920	55%	
Number of Slices containing only related logic	1,070	1,070	100%	
Number of Slices containing unrelated logic	0	1,070	0%	
Total Number of 4 input LUTs	1,575	3,840	41%	
Number used as logic	1,438			
Number used as a route-thru	126			
Number used as Shift registers	11			
Number of bonded IOBs	72	173	41%	
IOB Flip Flops	10			
Number of MULT18x18s	6	12	50%	
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	49,799			
Additional JTAG gate count for IOBs	3,456			

a)

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	1,005	3,840	26%	
Number of 4 input LUTs	1,482	3,840	38%	
Logic Distribution				
Number of occupied Slices	1,093	1,920	56%	
Number of Slices containing only related logic	1,093	1,093	100%	
Number of Slices containing unrelated logic	0	1,093	0%	
Total Number of 4 input LUTs	1,650	3,840	42%	
Number used as logic	1,482			
Number used as a route-thru	157			
Number used as Shift registers	11			
Number of bonded IOBs	17	173	9%	
IOB Flip Flops	3			
Number of Block RAMs	8	12	66%	
Number of MULT18x18s	6	12	50%	
Number of GCLKs	2	8	25%	
Total equivalent gate count for design	574,781			
Additional JTAG gate count for IOBs	816			

b)

Figura 4.14. Resultados de implementación a) del circuito de generación del umbral, b) del sistema de test.

A una frecuencia de 50 MHz el circuito puede procesar una imagen de 128x128 pixels en 328 μ s. En el caso de una imagen VGA (1024x768) el tiempo requerido para calcular el umbral es de 15,73 ms.

En tabla 4.1 se muestran algunos resultados del cálculo del umbral para imágenes de 128x128 píxeles. La última columna corresponde a los resultados obtenidos con el circuito propuesto mientras que la tercera columna corresponde al cálculo obtenido

mediante simulación con Matlab. La diferencia en los resultados obtenidos se debe a la precisión en la aritmética empleada. Tanto las columnas correspondientes al método de Otsu como al método basado la frecuencia de cada nivel de gris han sido obtenidos con la precisión que ofrece Matlab. La figura 4.15 muestra algunos ejemplos de la aplicación del umbral para realizar la segmentación de imágenes. En el apéndice A se muestran otros resultados de segmentación para otras imágenes.

	Método de Otsu	Frecuencia del nivel de gris	Método propuesto	
			valor teórico	salida del circuito
Lena	117	123	121	115
Cameraman	87	118	120	85
Peppers	101	103	102	96
Barbara	112	112	111	122
Nosveo	119	47	47	49
Pirata	99	40	39	52
Baboon	128	129	128	122
Goldhill	133	111	112	145
Tuneldevertigo	133	108	109	117

Tabla 4.1. Umbrales obtenidos al aplicar los métodos de Otsu, frecuencia de gris y la técnica propuesta.



Fig 4.15. a) Ejemplos de imágenes b) método de Otsu, c) frecuencia de nivel de gris, d) técnica propuesta

4.5. Resumen

En este capítulo nos hemos centrado en estudiar el problema de la segmentación de imágenes. Se ha propuesto una técnica de segmentación binaria basada en el cálculo de un umbral. Para el cálculo del umbral se ha hecho uso de un motor de inferencia difuso. Así nuestra técnica se basa en aplicar un sistema de inferencia difuso como mecanismo de cómputo para obtener el promedio de la frecuencia de los valores de los píxeles. Esta estrategia permite incrementar la velocidad de procesado ya que tan solo se requiere procesar la imagen una vez para generar el valor del umbral frente a otras técnicas que requieren procesar la imagen varias veces. Esto hace que el sistema propuesto sea muy adecuado en aplicaciones que demanden tiempo real en el procesado de imágenes.

Capítulo 5

Detección de bordes

Los algoritmos de detección de bordes permiten extraer información de las imágenes y reducir los requerimientos necesarios para el almacenamiento de la información. Un borde se define como un cambio abrupto en la luminosidad entre un píxel y otro adyacente. La detección de bordes suele constituir una de las primeras etapas del procesamiento de imágenes. Se trata de una operación previa a otras operaciones de procesamiento como es la detección de contornos, reconocimiento de objetos, clasificación de imágenes, etc. Por lo tanto la eficiencia de esas operaciones de procesamiento depende en gran medida de los resultados obtenidos en la detección de bordes.

Uno de los principales problemas en aplicaciones que requieran bajos tiempos de respuesta se debe a que los algoritmos que suelen aplicarse en la detección de bordes son costosos en cuanto a tiempo de procesamiento. Normalmente la detección de bordes se realiza por software debido al alto coste de recursos hardware que dichos algoritmos requieren. Nuestro interés en este capítulo se centra en mostrar un algoritmo de detección de bordes que sea eficiente en cuanto a la calidad de la generación de bordes y en términos de recursos hardware y velocidad de procesamiento.

Este capítulo se organiza en cuatro apartados. En el primer apartado se clasifican los algoritmos de detección de bordes y se describen algunos de ellos. A continuación, en el segundo apartado, se presenta el algoritmo de detección de bordes que proponemos. Para ello se describen cada una de las etapas del proceso de detección de bordes. El tercer apartado contiene un análisis de la calidad del algoritmo propuesto. En este análisis se hace una comparación de nuestra técnica con las otras técnicas que suelen ser las más frecuentemente utilizadas. Finalmente, en el último apartado, se realiza el diseño del circuito de detección de bordes y su implementación hardware sobre dispositivos FPGA.

5.1. Algoritmos para la detección de bordes en imágenes

La mayoría de las técnicas de detección de bordes pueden clasificarse en dos categorías: técnicas basadas en gradiente y métodos basados en Laplaciana. Las técnicas basadas en gradiente usan la primera derivada de la imagen y buscan el máximo y el mínimo de esa derivada. Ejemplos de este tipo de estrategia son: el método de Canny [CANN86], el método de Sobel, el método de Roberts [ROBE65], el método de Prewitt [PREW70], etc. Por otro lado, las técnicas basadas en Laplaciana buscan el cruce por cero de la segunda derivada de la imagen. Un ejemplo de esta técnica es método *zero-crossing* [MARR80].

Normalmente los mecanismos de extracción de bordes se implementan mediante la ejecución de la correspondiente realización software sobre un procesador. Sin embargo en aplicaciones que demanden restricciones en los tiempos de respuesta (aplicaciones en tiempo real) se requiere de implementaciones específicas en hardware. El principal inconveniente de las técnicas de detección de bordes para su realización hardware es la alta complejidad de los algoritmos existentes.

5.1.1. Métodos basados en gradiente

Las técnicas basadas en gradiente usan la primera derivada de la imagen y buscan el máximo y el mínimo de esa derivada. El motivo de emplear la primera derivada es debido a que toma el valor de cero en todas las regiones donde no varía la intensidad y

apunta en la dirección de la máxima variación (su módulo es proporcional a dicha variación). Por tanto un cambio de intensidad se manifiesta como un cambio brusco en la primera derivada. Esta característica es usada para detectar un borde. Así en el caso de funciones bidimensionales $f(x,y)$:

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

La magnitud viene dada por:

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2}$$

Para una imagen digital el gradiente de fila puede aproximarse por la diferencia de píxeles adyacentes de la misma fila:

$$G_x(i,j) = \frac{\partial f(x,y)}{\partial x} \approx f(x,y) - f(x-1,y)$$

Esto es:

$$G_x(i,j) = \begin{bmatrix} -1 & 1 \end{bmatrix} \times \begin{bmatrix} f(x-1,y) \\ f(x,y) \end{bmatrix}$$

De igual manera el gradiente de columna puede obtenerse mediante:

$$G_y(i,j) = \frac{\partial f(x,y)}{\partial y} \approx f(x,y) - f(x,y-1)$$

$$G_y(i,j) = \begin{bmatrix} f(x,y-1) & f(x,y) \end{bmatrix} \times \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

El gradiente de fila G_x y de columna G_y en cada punto se obtienen mediante la convolución de la imagen con las máscaras H_x y H_y , esto es:

$$G_x(i, j) = F(i, j) \otimes H_x(i, j)$$

$$G_y(i, j) = F(i, j) \otimes H_y(i, j)$$

La magnitud puede aproximarse por:

$$|G(i, j)| = \sqrt{G_x^2 + G_y^2} \approx |G_x(i, j)| + |G_y(i, j)|$$

El operador de detección de bordes de Canny fue desarrollando en la Universidad de Berkeley en 1986 y se basa en un algoritmo de múltiples fases para detectar un amplio rango de bordes [CANN86]. Es sin duda el operador más utilizado en la detección de bordes. El objetivo de Canny era encontrar un algoritmo óptimo para la detección de bordes. Un detector óptimo significa que debe tener las tres cualidades siguientes: en primer lugar debe proporcionar una buena detección, es decir, el algoritmo debe marcar tantos bordes reales como sea posible; en segundo lugar debe indicar una buena localización, o sea, los bordes marcados deben estar lo más cerca posible del borde de la imagen real; por último debe generar una mínima respuesta, es decir, un borde dado debe ser marcado sólo una vez y, además, el ruido presente en la imagen no debería crear falsos bordes.

El algoritmo de Canny consiste en tres grandes pasos:

- Obtención del gradiente: en este paso se calcula la magnitud y orientación del vector gradiente en cada píxel.
- Supresión no máxima: en este paso se logra el adelgazamiento del ancho de los bordes, obtenidos con el gradiente, hasta lograr bordes de un píxel de ancho.
- Histéresis de umbral: en este paso se aplica una función de histéresis basada en dos umbrales; con este proceso se pretende reducir la posibilidad de aparición de contornos falsos.

La figura 5.1 muestra el resultado de aplicar el método de Canny a la imagen de Lena.



Figura 5.1. Aplicación del método de Canny a la imagen de Lena

Una técnica que permite calcular de manera simple y rápida el gradiente espacial en una imagen consiste en aplicar el operador Roberts Cross [ROBE65]. El operador de Roberts consiste de una par de máscaras de convolución:

$$G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

A partir de una de las mascarar se obtiene la otra mediante una rotación de 90° .

La principal razón para el uso del operador Roberts es su velocidad de cómputo. La figura 5.2 muestra el resultado de aplicar el operador de Roberts a la imagen de Lena.



Figura 5.2. Aplicación del método de Roberts a la imagen de Lena

Otro método similar al anterior consiste en aplicar el operador de Sobel. Los valores G_x y G_y se pueden calcular a través de máscaras de 3×3 . La propiedad de este operador está en suavizar la imagen, eliminar parte del ruido.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Estas máscaras obtienen su mejor rendimiento con bordes que tienen orientaciones vertical y horizontalmente. En la figura 5.3 se muestra la salida de imagen de Lena una vez aplicado el operador de Sobel.

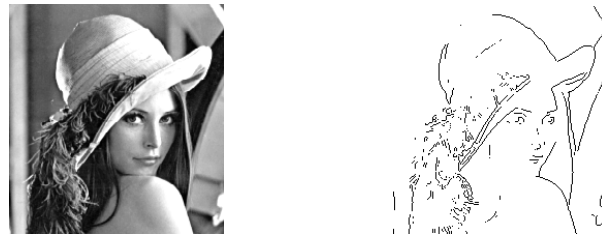


Figura 5.3. Aplicación del método de Sobel a la imagen de Lena

El operador de Prewitt [PREW70] es similar al de Sobel diferenciándose en los coeficientes de las máscaras. La magnitud y la dirección del gradiente se obtienen mediante las siguientes mascarar:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

La diferencia entre este método y el de Sobel se debe a que en el caso del operador de Prewitt las máscaras dan más peso al área de estimación ya que el factor de escala es 1/6 mientras que en el caso del operador de Sobel es de 1/8. La figura 5.4 muestra el resultado de aplicar el operador de Prewitt a la imagen de Lena.



Figura 5.4. Aplicación del método de Prewitt a la imagen de Lena

5.1.2. Métodos basados en Laplaciana

Cuando los bordes no son muy abruptos los operadores de gradiente no funcionan bien. En este caso resulta más eficiente el uso de derivadas de segundo orden. El más utilizado es el operador Laplaciana que se define como:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Los principales inconvenientes de esta técnica son: alta sensibilidad al ruido, produce doble borde y no detecta direcciones. Para atenuar la influencia del ruido se puede pasar un filtro gaussiano a la imagen y a continuación hacer el laplaciano para detectar bordes. Esta secuencia de operaciones es equivalente a aplicar un filtro que sea el laplaciano del filtro gaussiano. Esto es lo que se denomina operador laplaciano generalizado o también denominado filtro LOG.

Las técnicas basadas en Laplaciana buscan el cruce por cero de la segunda derivada de la imagen. El método *zero-cross* [MARR80] se basa en la aplicación de las siguientes mascarar:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

En la figura 5.5 se ha aplicado el método *zero-cross* a la imagen de Lena.



Figura 5.5. Aplicación del método *zero-cross* a la imagen de Lena

5.2. Descripción de la técnica de detección de bordes propuesta

El método de detección de bordes que se presenta en este apartado se aplica a la luminosidad de la imagen. Una imagen es una matriz bidimensional de píxeles cuyos valores pertenecen a un determinado rango. Vamos a considerar cada píxel codificado con 8 bits lo que da lugar a un conjunto de 256 posibles valores de tonos de gris. Por lo tanto una imagen es una función de dos variables (dimensiones) en el rango de 0 a 255.

El proceso de detección de bordes en una imagen consta de un conjunto de etapas que se muestran en la figura 5.6 [BARR08, HUSS08a,b]. La primera etapa realiza un filtrado de la imagen con objeto de eliminar ruido. A continuación se aplica un umbral T que clasifica los píxeles de la imagen en dos categorías obteniéndose una imagen en blanco y negro. Finalmente se realiza la detección de bordes.

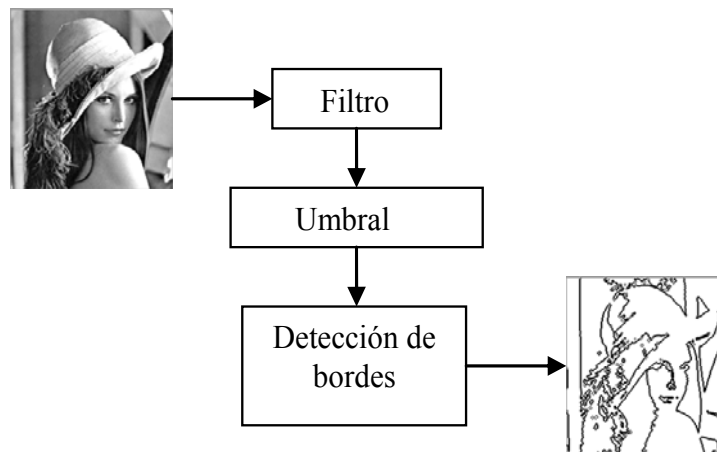
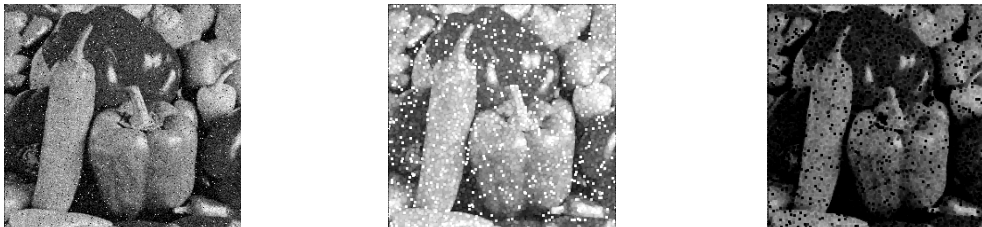


Figura 5.6. Diagrama de flujo para la detección de bordes

5.2.1. Etapa de filtrado

El filtrado permite mejorar los detalles de bordes en imágenes o bien permite reducir o eliminar patrones de ruido. La etapa de filtrado tiene por objetivo eliminar todos aquellos puntos que no suponen ningún tipo de información de interés. El ruido corresponde a información no deseada que aparece en la imagen. Proviene principalmente del propio sensor de captura (ruido de cuantización) y de la transmisión de la imagen (error al transmitir los bits de información). Básicamente consideramos dos tipos de ruido: gaussiano e impulsivo (*salt&peppers*). El ruido gaussiano tiene su origen en diferencias de ganancias del sensor, ruido en la digitalización, etc. El ruido impulsivo se caracteriza por la aparición de píxeles con valores arbitrarios normalmente detectables porque se diferencian mucho de sus vecinos más próximos. Una manera de eliminar estos tipos de ruido es mediante el empleo de un filtro paso de bajo. Dicho tipo de filtro realiza un suavizado de la imagen reemplazando valores altos y bajos por valores promedio.

Algunos de los filtros que suelen aplicarse en esta etapa son los filtros de máximo y mínimo se utilizan para eliminar el ruido *salt&peppers*. El filtro de máximo se utiliza en la selección del mayor valor del conjunto de valores de los niveles de gris mientras que el filtro mínimo selecciona el menor valor. El filtro mínimo funciona mejor en el caso de las imágenes con ruido de tipo sal (valores altos) mientras que el máximo funciona mejor en el caso de imágenes con ruido de tipo pimienta (valores bajos). En figura 5.7 se muestra un ejemplo de aplicar filtrado máximo y mínimo a una imagen con ruido *salt&peppers*.



a)

b)

c)

Figura 5.7. a) Imagen con ruido *salt&peppers*, filtrado b) máximo, c) mínimo.

Otro tipo de filtro corresponde al filtro de punto medio. Este filtro calcula el valor promedio de un conjunto de valores de los niveles de gris. Para ello basta con calcular el valor medio del máximo y mínimo valor dentro de la ventana:

$$I_1 \leq I_2 \leq I_3 \leq I_4 \leq \dots \leq I_N$$
$$\text{Punto Medio} = \frac{(I_1 + I_N)}{2}$$

Este tipo de filtro se utiliza para la eliminación del ruido gaussiano y del ruido uniforme. En el caso del filtro gaussiano sin embargo se aplica la siguiente expresión:

$$g_{\sigma}(i, j) = c e^{-\frac{(i^2 + j^2)}{2\sigma^2}}$$

donde, c es una constante que se calcula para todos los coeficientes, σ es la varianza. Valores típicos son $c = 1/6.1689$ y $\sigma = 1$.

El filtro gaussiano, presenta como principales características una simetría de rotación y el comportamiento de que a mayor σ se obtiene un mayor suavizado.

Otro tipo de filtro que tiene como objeto suavizar imágenes sin distorsionar los detalles de las mismas intentando mantener los bordes es el filtro de Kuwahara. Este filtro parte de una máscara de tamaño $N \times N$ y la divide en cuatro regiones.

$$N = 4L + 1$$

Para cada una de las regiones L , se calcula la media del nivel de gris y la varianza. El valor de salida asociado será la media del nivel de gris de la región con menor varianza. Así la figura 5.8 muestra un bloque de 3×3 píxeles. El cálculo de X_{media} y $X_{varianza}$ en una región L viene dado por:

$$X_{media}(L) = \frac{(X_{(i,j)} + X_{(i,j+1)})}{2}$$

$$X_{\text{varianza}}(L) = \frac{((X_{(i,j)} - X_{\text{media}}(L))^2 + (X_{(i,j+1)} - X_{\text{media}}(L))^2)}{2}$$

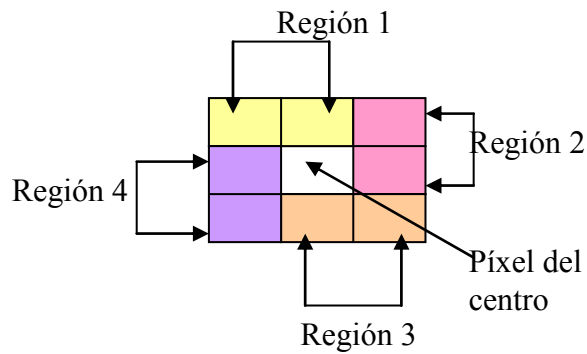
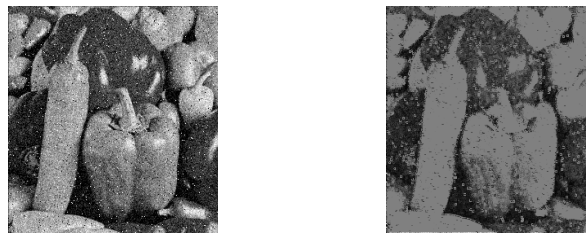


Figura 5.8. Aplicación del filtrado de Kuwahara a un bloque de 3x3 píxeles.

En figura 5.9 se muestra la salida del filtro Kuwahara de una imagen con ruido *salt&peppers*.



a)

b)

Figura 5.9. a) Imagen con ruido *salt&peppers*, b) salida del filtro Kuwahara.

La operación de filtrado que se realiza en el sistema propuesto se basa en el operador suma acotada de Lukasiewicz. Dicho operador proviene del álgebra Lukasiewicz y se define como:

$$\text{SumaAcotada}(x, y) = \min(1, x + y)$$

donde $x, y \in [0,1]$. La principal ventaja de aplicar este operador se debe a la simplicidad de la realización hardware.

El filtro basado en la suma acotada de Lukasiewicz realiza un suavizado de la imagen y resulta adecuado en ruidos tipos *salt&peppers* así como gaussiano. La figura 5.10 muestra el efecto de aplicar este tipo de filtrado.



Figura 5.10. a) Imagen de entrada con ruido del tipo *salt&peppers*, b) salida del filtro basado en la suma acotada de Lukasiewicz

La aplicación del filtro se ha realizado empleando una mascara basada en una matriz de 3x3. De esta manera para el píxel x_{ij} se aplica la mascara para obtener el nuevo valor y_{ij} del pixel de acuerdo con la expresión siguiente:

$$y_{ij} = \min\left(1, \frac{1}{8} \sum_{k=-1}^1 \sum_{l=-1}^1 x_{i+k, j+l}\right)$$

5.2.2. Etapas de umbralización y detección de bordes

La imagen de entrada para la detección de borde es una imagen binaria cuyos píxeles pueden tomar el valor 0 (negro) o 1 (blanco). Para ello se realiza la comparación entre cada píxel de la imagen con un valor umbral T . En el capítulo anterior ya se han discutido diversos mecanismos para el cálculo del umbral. A partir de la imagen binaria los bordes aparecen cuando tiene lugar un cambio entre blanco y negro entre dos píxeles.

$$a_{edge} = \begin{cases} 0 & \text{if } a - b \neq 0 \\ 1 & \text{if } a - b = 0 \end{cases}$$

Donde a y b son el valor de píxeles consecutivos, y a_{edge} es el valor resultante.

La generación del borde consiste en determinar si cada píxel tiene vecinos con valores diferentes. Dado que la imagen es binaria esta operación de detección de borde se obtiene calculando la operación lógica *xor* entre píxeles vecinos utilizando una máscara 3x3.

$$y_{(i,j)} = \oplus x_{(i,j)} = x_{(i-1,j-1)} \oplus x_{(i-1,j)} \oplus \dots \oplus x_{(i+1,j+1)}$$

$x_{i-1,j-1}$	$x_{i-1,j}$	$x_{i-1,j+1}$
$x_{i,j-1}$	$x_{i,j}$	$x_{i,j+1}$
$x_{i+1,j-1}$	$x_{i+1,j}$	$x_{i+1,j+1}$

Figura 5.11. Máscara 3x3 para la detección de bordes.

La figura 5.12 muestra un ejemplo de la aplicación del operador *xor* en la imagen binaria.



Figura 5.12. (a) Imagen binaria de Lena, (b) detección de bordes

Usando la máscara de 3x3, es posible refinar la generación de bordes detectando su orientación. Para ello pueden considerarse las cuatro orientaciones se indica en la figura 5.13. Este refinamiento se realiza calculando la operación *xor* sobre los 3 píxeles de cada orientación. De esta forma, para una orientación horizontal tendremos

$$y_{(i,j)} = x_{(i,j-1)} \oplus x_{(i,j)} \oplus x_{(i,j+1)}$$

Para el caso de 45 ° será

$$y_{(i,j)} = x_{(i+1,j-1)} \oplus x_{(i,j)} \oplus x_{(i-1,j+1)}$$

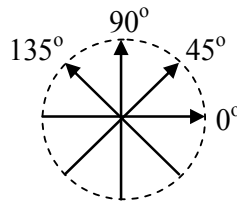


Figura 5.13. Orientaciones para la generación de los bordes.

5.3. Análisis de calidad

Con objeto de caracterizar el algoritmo de detección de bordes que proponemos vamos a aplicarlo a un banco de imágenes de test. Se trata de un conjunto de 12 imágenes de diferente tamaño empleadas usualmente en la literatura. Las imágenes se ilustran en el apéndice B.

Sobre el banco de imágenes hemos aplicado diversos algoritmos de detección de bordes: métodos de Canny, Prewitt, Sobel, Roberts y *zero-cross*. También hemos aplicado el método basado en la suma acotada de Lukasiewicz. Las imágenes obtenidas se muestran en el apéndice C.

A la hora de realizar un análisis de calidad de los métodos de detección de bordes no hemos encontrado en la literatura ninguna técnica que permita evaluar dichos métodos. Básicamente la bondad de un método se obtiene comparando el resultado con la detección de bordes que realizaría una persona [HEAT96, SALO96]. Indudablemente esta manera de evaluar un método se basa en criterios subjetivos. De hecho distintas personas dan lugar a la detección de distintos bordes para una misma imagen. Existen algunas iniciativas que intentan estandarizar este mecanismo de evaluación [MART01].

Con objeto de extraer información analítica de las imágenes obtenidas nosotros hemos usado como referencia el método de Canny ya que es el considerado, de manera generalizada en la literatura, como el más preciso. A la hora de evaluar la calidad de nuestra aproximación se han considerado dos parámetros: el número de píxeles activos (esto es, el número de píxeles que forman los bordes) y la coincidencia con la

aproximación de Canny (el número de píxeles activos que coinciden con el método de Canny). Los resultados obtenidos se ilustran en las tablas 5.1 y 5.2.

En sí mismo el número de píxeles activos (tabla 5.1) no indica una medida de calidad pero sí que da información sobre la capacidad de discriminación del algoritmo, o sea la capacidad de localizar bordes o discontinuidad en la imagen. Pude observarse que la técnica que proponemos da lugar a una mayor detección de discontinuidades en la imagen seguida por el método de Canny.

En lo que se refiere a la precisión en la generación de bordes la tabla 5.2 muestra que la técnica propuesta es la técnica que más coincide con la técnica de Canny en la mayoría de las imágenes (tiene una porcentaje más alta de píxeles que coinciden con el obtenido por Canny). Eso significa que la técnica propuesta da una calidad buena para detectar los bordes de imágenes.

	Canny	Prewitt	Sobel	Roberts	Zero cross	T=0.5
Baboon	40088	12427	12678	6495	31501	59826
512x512	15.3%	4.7%	4.8%	2.5%	12.0%	22.8%
Baboon_small	1869	485	490	244	1147	3303
115x115	14.1%	3.7%	3.7%	1.8%	8.7%	25.0%
Barbara	25321	13530	14439	6223	16887	31119
512x512	9.7%	5.2%	5.5%	2.4%	6.4%	11.9%
Cameraman	5747	2509	2503	2341	3847	8644
256x256	8.8%	3.8%	3.8%	3.6%	5.9%	13.2%
Goldhill	28098	10748	10783	7336	22584	46107
512x512	10.7%	4.1%	4.1%	2.8%	8.6%	17.6%
Lena	4005	1834	1900	1702	3151	7940
222x208	8.7%	4.0%	4.1%	3.7%	6.8%	17.2%
Nosveo	5425	3283	3310	4328	4741	5762
389x433	3.2%	1.9%	2.0%	2.6%	2.8%	3.4%
Peppers	16068	6967	7021	6634	10955	20437
512x512	6.1%	2.7%	2.7%	2.5%	4.2%	7.8%
Peppers_small	1298	685	704	603	1029	2823
115x115	9.8%	5.2%	5.3%	4.6%	7.8%	21.3%
Pirata	1702	1022	1033	1142	983	1862
150x200	5.7%	3.4%	3.4%	3.8%	3.3%	6.2%
Soccer	260	175	173	172	275	541
64x64	6.3%	4.3%	4.2%	4.2%	6.7%	13.2%
Tuneldevertigo	5407	5231	5258	6722	5386	14050
368x381	3.9%	3.7%	3.8%	4.8%	3.8%	10.0%

Tabla 5.1. Píxeles activos que toman el valor de los bordes y el porcentaje respecto a la imagen total.

	Prewitt	Sobel	Roberts	Zero cross	T=0.5
Baboon 512x512	18.5%	18.3%	7.3%	32.5%	32.8%
Baboon_small 115x115	15.5%	14.9%	4.3%	27.7%	31.5%
Barbara 512x512	28.2%	28.2%	5.5%	27.5%	29.5%
Cameraman 256x256	29.1%	28.4%	19.0%	30.0%	39.7%
Goldhill 512x512	23.8%	23.7	11.0%	32.0%	36.6%
Lena 222x208	30.3%	30.8%	19.0%	32.2%	40.0%
Nosveo 389x433	41.4%	41.2%	37.6%	36.0%	16.5%
Peppers 512x512	28.6%	28.4%	16.5%	30.0%	23.4%
Peppers_small 115x115	33.8%	34.4%	21.0%	31.9%	47.4%
Pirata 150x200	38.2%	38.0%	30.5%	24.7%	26.1%
Soccer 64x64	28.5%	29.2%	25.0%	46.9%	8.5%
Tuneldevertigo 368x381	68.8%	68.0%	57.4%	45.6%	40.5%

Tabla 5.2. Porcentaje de pixeles que coinciden con el obtenido por Canny.

5.4. Diseño del sistema de detección de bordes

El diagrama de bloques del sistema que realiza la detección de bordes se muestra en la figura 5.14. La imagen es leída píxel a píxel por el circuito de control de memoria. Dicho circuito provee en cada ciclo de reloj un píxel de 8 bits. Dicho píxel se almacena en la memoria RAM mediante la intervención del circuito de control de la memoria. Durante la lectura de la imagen se realiza el cálculo del umbral por el circuito de generación del umbral. Una vez almacenada la imagen en la memoria dicho circuito suministra el valor del umbral de la imagen.

A continuación el circuito de detección de bordes inicia su operación leyendo de la memoria 8 píxeles en cada ciclo de reloj (dos palabras de 32 bits). De esta manera el circuito de detección de bordes es capaz de producir 4 salidas en paralelo que se almacenan en la memoria. Cada dato generado corresponde a un píxel de la imagen de bordes. Esta imagen es binaria por lo que sólo se requiere un bit para representar el valor del píxel (0 si se trata de un píxel de un borde y 1 si el píxel es del fondo de la imagen). La nueva imagen de bordes se almacena en la memoria RAM.

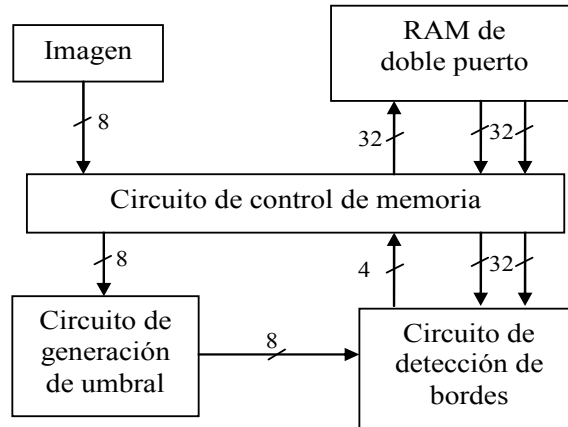


Figura 5.14. Diagrama de bloques del sistema de detección de bordes

El algoritmo de detección de bordes está constituido por tres etapas [BARR09]. La figura 5.15 muestra el pseudocódigo de este algoritmo. En la primera etapa se realiza el filtrado mediante la suma acotada de Lukasiewicz. A continuación se aplica el umbral con objeto de obtener una imagen binaria. En la tercera etapa se obtiene la imagen de bordes. Para ello se emplea una máscara 3x3 de manera que sólo los píxeles vecinos al que se procesa son de interés. Así si los valores de los píxeles vecinos son iguales significa que no hay borde mientras que si son diferentes se considera que el píxel es un borde.

```

%***** LUKASIWICZ BOUNDED-SUM FILTER *****
for i=1:N
  for j=1:M
    imgi,j=min(1,scale*(imgi-1,j-1+imgi-1,j+imgi-1,j+1+
                    imgi,j-1+imgi,j+imgi,j+1+
                    imgi+1,j-1+imgi+1,j+imgi+1,j+1));

%***** THRESHOLDING *****
for i=1:N
  for j=1:M
    if imgi,j>THRESHOLD imgi,j=1;
    else imgi,j=0;

%***** EDGE DETECTION *****
for i=1:N
  for j=1:M
    edgei,j=EDGE( imgi-1,j-1,imgi-1,j,imgi-1,j+1,
                  imgi,j-1,imgi,j,imgi,j+1,
                  imgi+1,j-1,imgi+1,j,imgi+1,j+1);

```

Figura 5.15. Pseudocódigo del algoritmo de detección de bordes.

La figura 5.16 muestra el esquema de procesamiento del sistema para el cálculo de un píxel. Los píxeles 1 al 9 corresponden a la máscara 3x3 que recorre la imagen. La Unidad Funcional (UF) realiza el procesamiento de los datos almacenados en los registros que contienen la máscara.

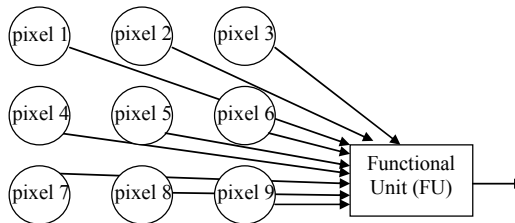


Fig. 5.16. Esquema para el procesamiento de un píxel.

Con objeto de mejorar el tiempo de procesamiento de la imagen se ha extendido la arquitectura del sistema a una matriz 8x3 como se muestra la figura 5.17. Cada Unidad Funcional (UF) opera sobre una máscara 3x3 de acuerdo con el esquema de la figura 5.16. Los datos se almacenan en los registros de entrada (R3, R6, R9, ...) y se desplazan en cada ciclo de reloj de acuerdo con el conexionado mostrado en la figura. En el tercer ciclo de reloj las unidades funcionales operan sobre los datos y generan las salidas. La imagen se recorre por columnas. En cada ciclo de reloj la máscara avanza una columna en la imagen. Los píxeles entran por la derecha y pasan de una etapa a otra hasta salir por la izquierda. Es una arquitectura sistólica basada en una topología lineal y permite procesar varios píxeles en paralelo.

El sistema recibe dos entradas de datos de 32 bits (8 píxeles). Dichos datos provienen de una memoria de doble puerto que almacena la imagen de entrada. El circuito recibe también como dato de entrada el umbral de la imagen (T) que se ha calculado previamente. El circuito genera como salida los 4 bits correspondientes a las salidas de los píxeles almacenados en R5, R8, R11 y R14.

La unidad funcional opera sobre la máscara de datos 3x3 y genera el valor de salida correspondiente al primer elemento de la máscara (un píxel en la figura 5.17). El diagrama de bloques de una unidad funcional se muestra en la figura 5.18. El circuito se compone de dos etapas de *pipeline* de manera que los datos tienen una latencia de dos

ciclos de reloj. La primera etapa realiza el filtrado de la imagen y aplica el umbral T. El detector de borde de salida opera sobre la mascara binaria (imagen en blanco y negro).

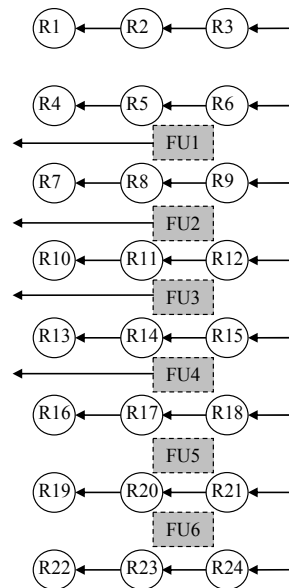


Figura 5.17. Diagrama de bloques de la arquitectura 8x3.

La figura 5.19 muestra los circuitos correspondientes a los diferentes bloques de la unidad funcional (FU). Como podemos observar en la figura 5.19a el filtro basado en la suma acotada de Lukasiewicz recibe los datos de entrada almacenados en los registros R1 a R9 de la mascara 3x3. Estos datos son escalados por el factor 0.125 que consiste en dividir por 8, esto es realizar tres desplazamientos a la izquierda. La suma de los datos se compara (utilizando el acarreo generado como control de la comparación) con el valor 1. La lógica de umbral (figura 5.19b) consiste en comparar el dato del píxel con el valor umbral. La salida es una imagen binaria (blanco/negro) por lo que sólo requiere un bit. Finalmente el circuito de detección de bordes recibe una imagen 3x3 binaria. Consiste en realiza la operación *xor* de los bits. Si todos los bits de la mascara son iguales la salida corresponde al fondo, mientras que si algún bit es diferente la salida corresponde a un borde.

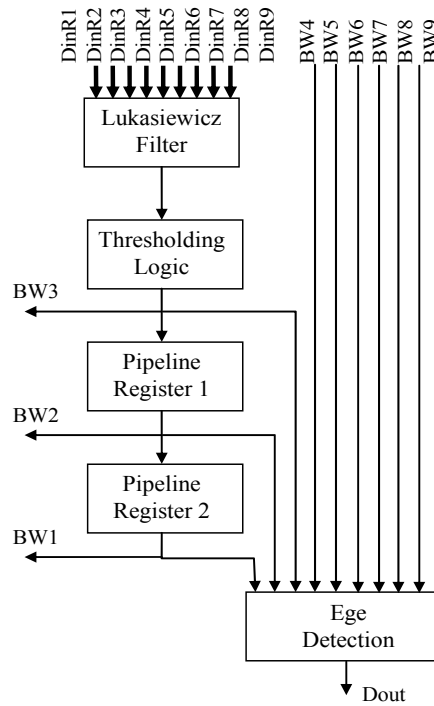


Figura 5.18. Esquema de una unidad funcional (FU)

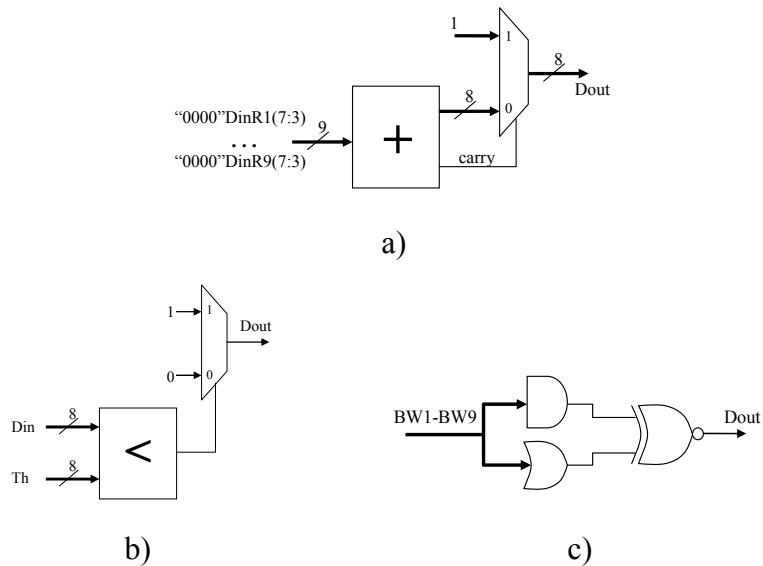


Figura 5.19. a) Filtro Lukasiewicz, b) lógica de umbralización, c) circuito de detección de bordes.

La máquina de estado que controla el sistema se muestra en la figura 5.20. Dicha máquina dispone de 4 estados. La máscara recorre la imagen por columnas. Cada vez que se inicia una fila se requieren dos ciclos de reloj para inicializar los registros de la máscara (CYCLE1 y CYCLE2). En el siguiente ciclo (PROCESSING) se realiza el procesamiento de los datos de manera que en los ciclos sucesivos se procesan los datos de la siguiente columna.

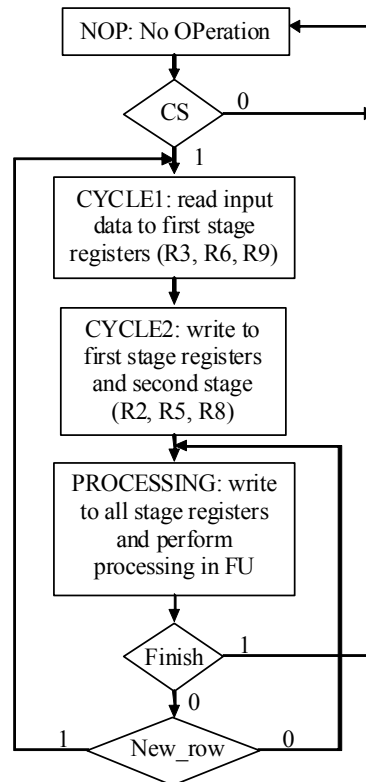


Figura 5.20. FSM de la unidad de control del sistema.

La figura 5.21 muestra el cronograma del circuito. Se puede observar que con la bajada de la señal CS se inicia la operación del sistema. Al tercer ciclo de reloj se activa la señal Dvalid que indica que se dispone de una salida válida. Los datos de entrada se suministran en cada ciclo de reloj. Una vez activada Dvalid vemos que en los siguientes ciclos se originan datos de salida válidos (ya que Dvalid=1).

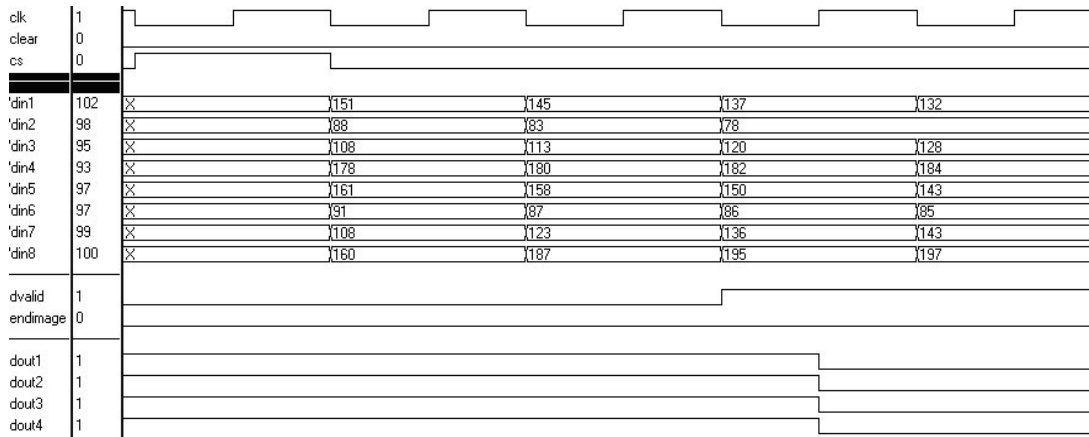


Figura 5.21. Cronograma de la operación del circuito.

El sistema ha sido implementado sobre un FPGA de la familia Spartan3 XC3S200FT255 de Xilinx. La figura 5.22 muestra los resultados de implementación del sistema. El circuito ocupa un área de 293 slices que supone un coste de 5.361 puertas equivalentes. Puede operar con una frecuencia de reloj de 77 MHz, si bien en las pruebas realizadas hemos empleado una placa que dispone de un reloj a 50 MHz. Esto supone que el tiempo requerido para procesar una imagen es de 0.335 ms lo que permite procesar 2985 imágenes por segundo.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	150	3,840	3%	
Number of 4 input LUTs	397	3,840	10%	
Logic Distribution				
Number of occupied Slices	293	1,920	15%	
Number of Slices containing only related logic	293	293	100%	
Number of Slices containing unrelated logic	0	293	0%	
Total Number of 4 input LUTs	423	3,840	11%	
Number used as logic	397			
Number used as a route-thru	26			
Total equivalent gate count for design	5,361			

Figura 5.22. Resultados de implementación sobre FPGA.

5.5. Resumen

Se ha realizado la implementación hardware de un sistema de detección de bordes que utiliza el sistema de segmentación de imágenes propuesto en el capítulo anterior. El acondicionamiento de la imagen para su posterior procesamiento se basa en un filtro que utiliza el operador suma acotada de Lukasiewicz. A continuación se realiza la segmentación binaria de la imagen. Finalmente la etapa de detección de bordes se realiza mediante la operación *xor* de los píxeles de una máscara de acuerdo con diferentes orientaciones. Con objeto de aumentar el *throughput* se ha realizado una implementación del sistema que permite procesar 4 píxeles en paralelo mediante una arquitectura sistólica.

Conclusiones

La Tesis aborda cuatro aspectos relacionados con el procesado de imágenes: la compresión de imágenes, la mejora del contraste, la segmentación y la detección de bordes. A continuación se presentan las principales conclusiones de este trabajo.

1. Compresión de imágenes

El criterio del diseño es reducir la complejidad computacional de los algoritmos que hacen la compresión de imágenes de forma que el circuito resultante es de bajo costo y opera a alta velocidad de procesamiento.

1.1. Se han presentado dos estrategias para la compresión de imágenes estáticas. Una basada en una aproximación lineal a tramos y la otra basada en el particionado del histograma y su codificación.

1.2. La estrategia de compresión basada en la aproximación lineal ha dado lugar a tres técnicas de compresión. Todas se basan en acotar el error de la aproximación a un valor prefijado.

1.3. La principal característica de estas técnicas es la simplicidad de los circuitos y al mismo tiempo una relación entre razón de compresión y calidad aceptables.

1.4 La estrategia de compresión basada en el particionado del histograma y su codificación permite aplicar mecanismos de compresión con pérdidas y sin pérdidas.

1.5. Se ha diseñado un circuito que conjuga las diferentes técnicas de compresión/descompresión que tiene una buena relación coste-velocidad de operación. De los resultados de los circuitos podemos observar que nuestra propuesta presenta un menor coste en área ya que dicho circuito supone un 0,6% del tamaño de un compresor JPEG. Por otro lado se multiplica por 6 la frecuencia de operación lo que da lugar a menores tiempos requeridos para la compresión de imágenes.

2. Control del contraste de imágenes

2.1 Se ha realizado la propuesta de una técnica para el control del contraste en imágenes basada en la aplicación de operadores del álgebra de Lukasiewicz.

2.2 Se han implementado dichos operadores mediante dos estrategias de diseño: una basada en circuitos neuronales y la otra basada en lógica combinacional.

2.3 La técnica que se ha propuesto para el control del contraste es una técnica local que modifica la imagen punto a punto. El proceso de transformación se realiza aplicando una máscara que recorre la imagen. Se ha considerado un mecanismo que selecciona el parámetro de control aplicando un sistema basado en lógica difusa. El motor de inferencia difuso toma la decisión del valor que debe tener el parámetro de control dependiendo del contraste local de la máscara considerada. Esto permite adaptar el contraste para cada región de la imagen de manera independiente.

2.4. Se ha realizado un mecanismo de control de contraste que se adapte a las características locales de la imagen. Para ello se ha considerado un sistema de toma de decisión que determine el tipo de operador más adecuado que debe aplicarse en cada momento (suma acotada en la zona oscura de la imagen y el producto acotado en la zona clara). El sistema de toma de decisiones se basa en un mecanismo de inferencia basado en lógica difusa.

2.5 Se ha establecido un mecanismo para realizar la interpolación de funciones lineales a tramos aplicando el álgebra de Łukasiewicz.

3. Segmentación de imágenes

3.1 Se ha propuesto una técnica de segmentación de imágenes basada en la umbralización binaria. El cálculo del valor del umbral se realiza mediante un sistema difuso. En este caso se aprovechan las características de procesamiento de los sistemas difusos como mecanismo de cálculo del umbral.

3.2 Se ha realizado el diseño e implementación del circuito de cálculo del umbral. Dicho circuito tiene como ventaja la alta velocidad de procesamiento ya que tan sólo se requiere recorrer la imagen una vez. Esto hace que resulte muy eficiente en aplicaciones que demanden tiempo real en la respuesta del sistema.

4. Detección de bordes de imágenes

4.1 Se ha propuesto una estrategia para la detección de bordes en imágenes que resulta ser muy adecuada para su implementación hardware.

4.2 Se ha diseñado e implementado el circuito de detección de bordes. Dicho circuito ha sido optimizado para permitir el procesamiento en paralelo de varios píxeles mediante una arquitectura sistólica basada en la aplicación de etapas de *pipeline*.

4.3 La técnica propuesta presenta aceptables resultados en cuanto a la calidad de la imagen de acuerdo con la comparación realizada con otras técnicas referenciadas en la literatura.

Trabajos futuros













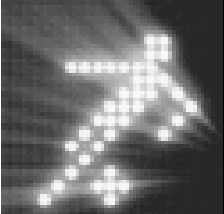










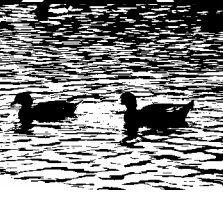

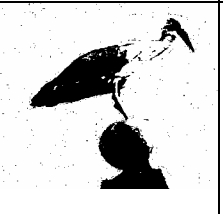

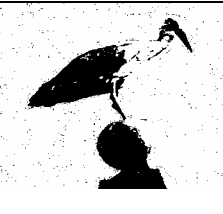
Las características de bajo coste y alta velocidad de procesamiento de los circuitos obtenidos en cada una de las actividades desarrolladas hacen que estos sistemas sean adecuados para incorporarse en el sensor de visión. Ello requiere paralelizar los algoritmos y/o adaptar los sistemas con el sensor en el mismo chip.



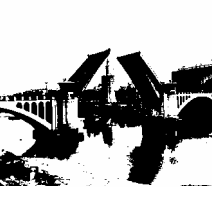

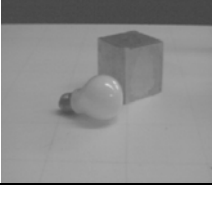
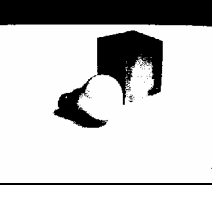

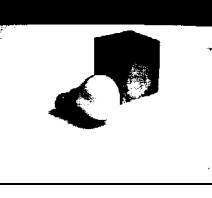
Otra línea de trabajo se refiere a la realización e implementación hardware de sistemas de procesamiento de imágenes de alto nivel. Ello requiere de la aplicación de procesamiento de bajo nivel, de acuerdo con este trabajo de tesis, como etapa inicial y un procesamiento posterior de alto nivel que permita realizar aplicaciones tales como reconocimiento de objetos, seguimiento controlado por visión, procesamiento de sistemas distribuidos de visión, etc.

Apéndice A

Resultados de segmentación binaria en imágenes

En la tabla siguiente se muestran nueve imágenes monocromáticas. Las imágenes se han utilizados como un banco de prueba para analizar los resultados de técnicas de la segmentación binaria. En la tabla se muestran los resultados de las imágenes aplicando tres estrategias de segmentación binaria: técnica de Otsu, una técnica se basa en la frecuencia de nivel de gris y la técnica propuesta en esta Tesis.

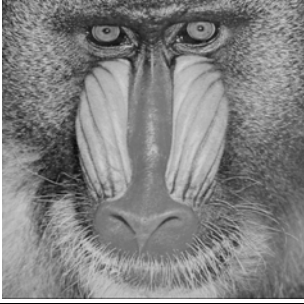

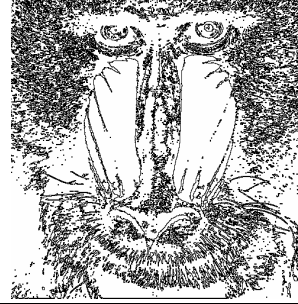







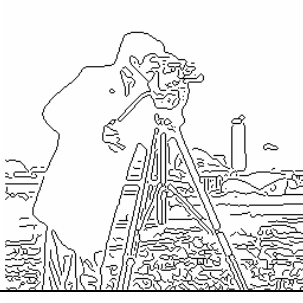
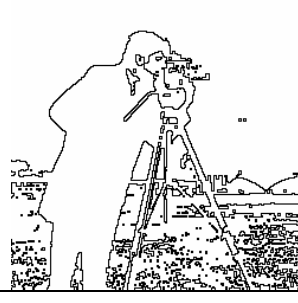

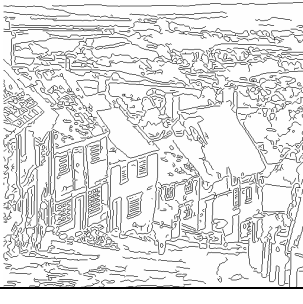

	Imagen Original	Técnica Otsu	Técnica frecuencia	Técnica propuesta
Pirata (200x150)				
Baboon (512x512)				
Goldhill (512x512)				
Soccer (64x64)				
Tuneldevertigo (381x368)				
patos1_356x292 (356x292)				
ciconia2_356x292 (356x292)				





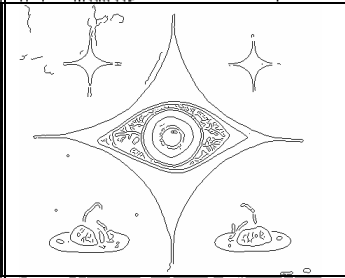
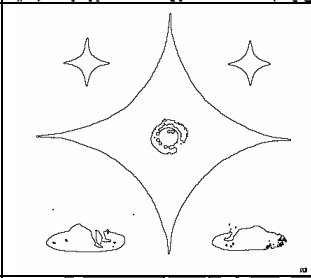


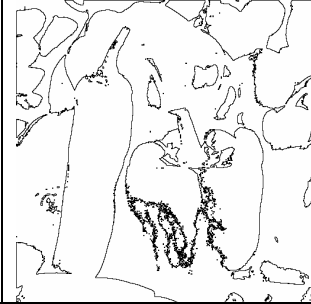


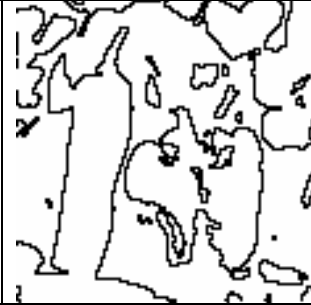


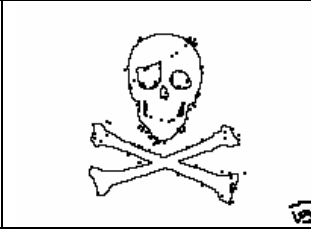
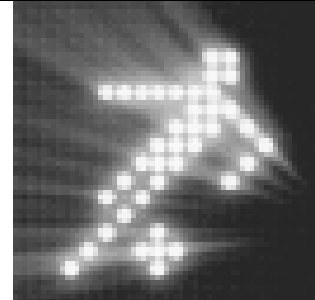
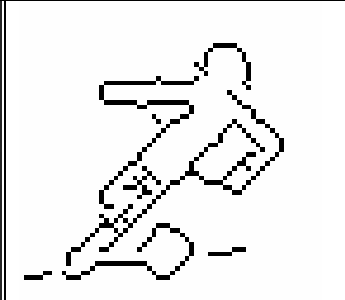
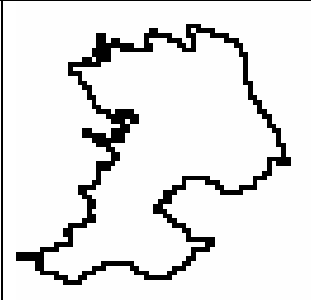
puente_356x2 92 (367x301)				
Lambara (367x301)				

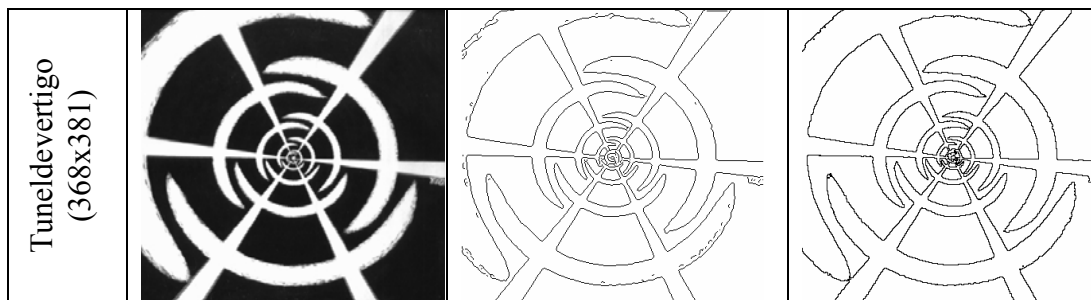
Apéndice B

Resultados de detección de bordes en imágenes

Las siguientes imágenes monocromáticas se han utilizado como un banco de prueba para comparar técnicas de detección de bordes. Se han comparando dos técnicas: el método de Canny y la técnica propuesta en esta Tesis.

	Imagen Original	Método Canny	Método propuesto
Baboon (512x512)			
Baboon (115x115)			
Barbara (512x512)			
Cameraman (256x256)			
Goldhill (512x512)			

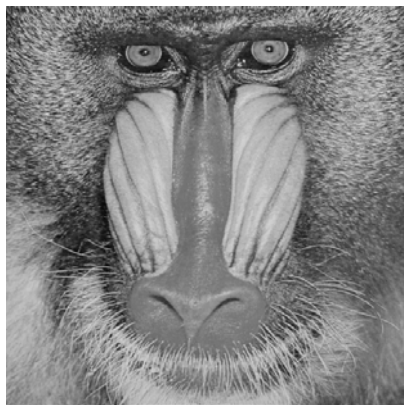
<p>Lena (222x208)</p>			
<p>Nosveo (389x433)</p>			
<p>Peppers (512x512)</p>			
<p>Peppers (115x115)</p>			
<p>Pirata (150x200)</p>			
<p>Soccer (64x64)</p>			



Apéndice C

Comparación de distintas técnicas de detección de bordes

En este apéndice se presenta la comparación de los resultados de imágenes monocromáticas de aplicar distintas técnicas de detección de bordes. Sobre cada imagen se aplican seis técnicas: técnicas de Canny, Sobel, Prewitt, Roberts, Zerocross, y la técnica propuesta en esta Tesis.



Baboon (512x512)



Canny



Sobel



Prewitt



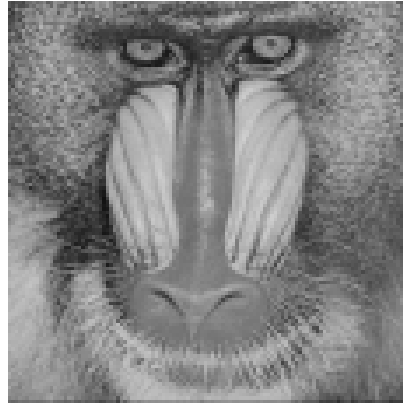
Roberts



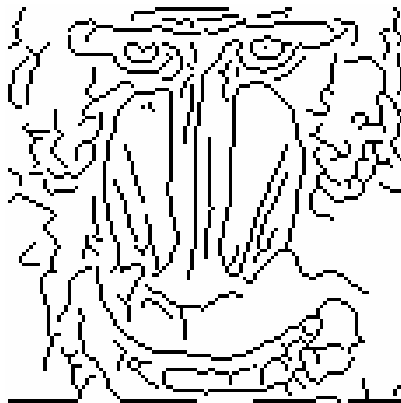
Zerocross



Propuesto



Baboon (115x115)



Canny



Sobel



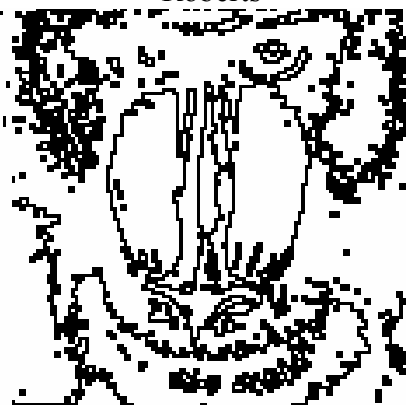
Prewitt



Roberts



Zerocross



Propuesto



Barbara (512x512)



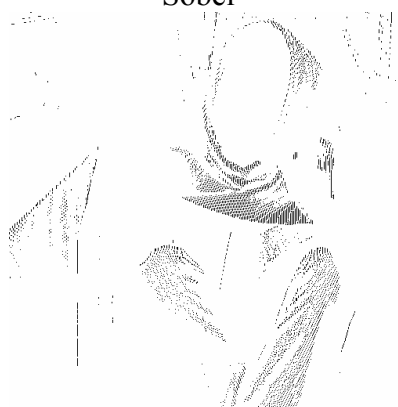
Canny



Sobel



Prewitt



Roberts



Zerocross



Propuesto



Barbara_128 (128x128)



Canny



Sobel



Prewitt



Roberts



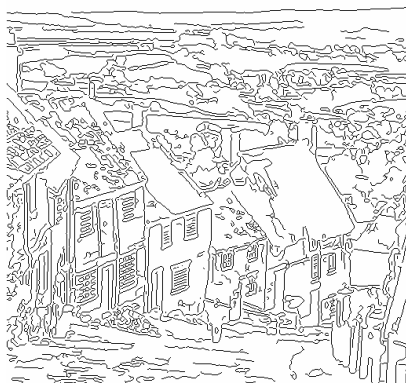
Zerocross



Propuesto



Goldhill (512x512)



Canny



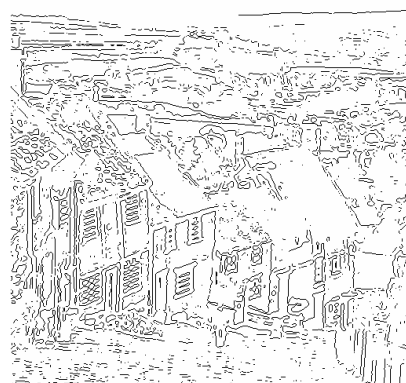
Sobel



Prewitt



Roberts



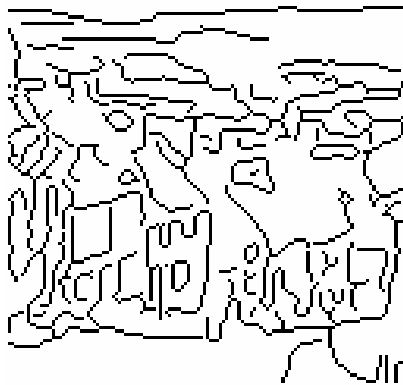
Zero-cross



Propuesto



Goldhill_128 (128x128)



Canny



Sobel



Prewitt



Roberts



Zerocross



Propuesto



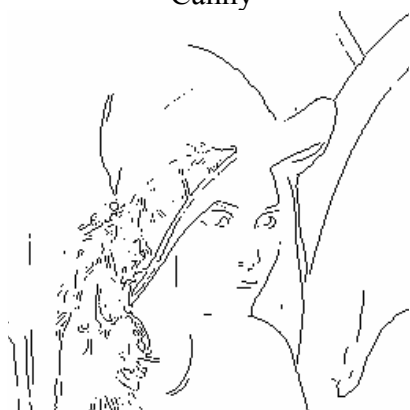
Lena_256 (256x256)



Canny



Sobel



Prewitt



Roberts



Zerocross



Propuesto



Lena_128 (128x128)



Canny



Sobel



Prewitt



Roberts



Zerocross



Propuesto



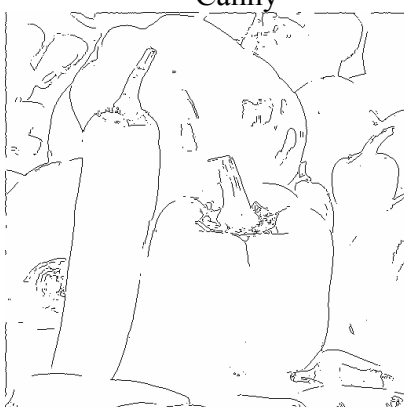
Peppers (512x512)



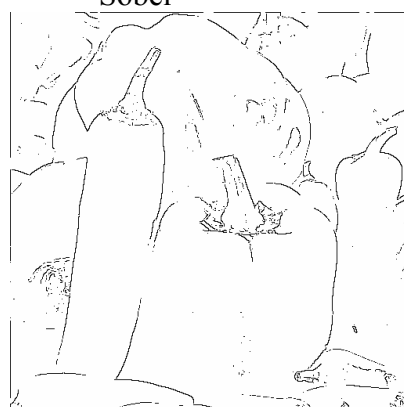
Canny



Sobel



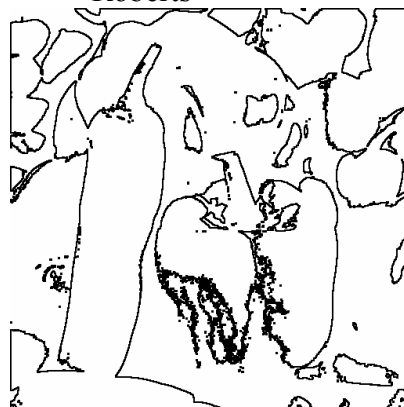
Prewitt



Roberts



Zerocross



Propuesto



Peppers_115 (115x115)



Canny



Sobel



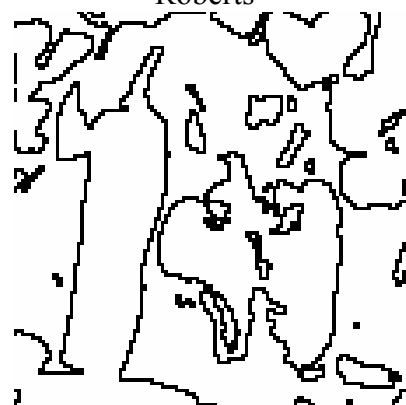
Prewitt



Roberts



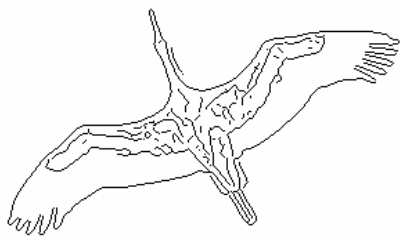
Zerocross



Propuesto

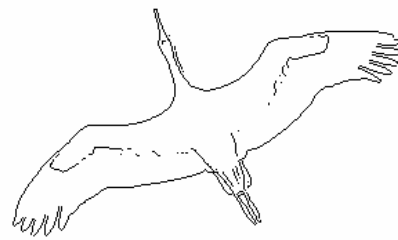


ciconia1_356x292 (356x292)



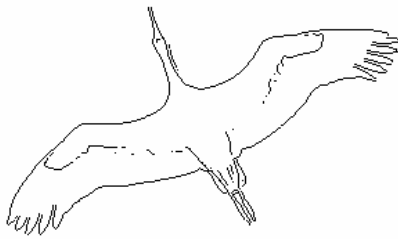
by Lygeum

Canny



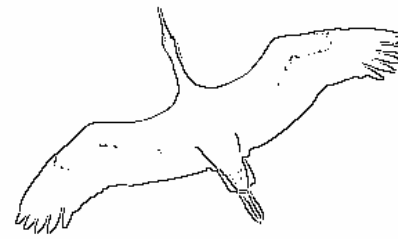
by Lygeum

Sobel



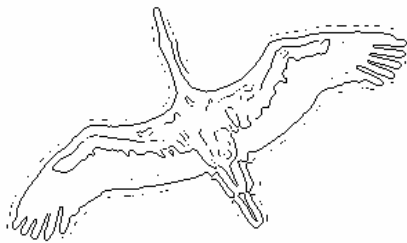
by Lygeum

Prewitt



by Lygeum

Roberts



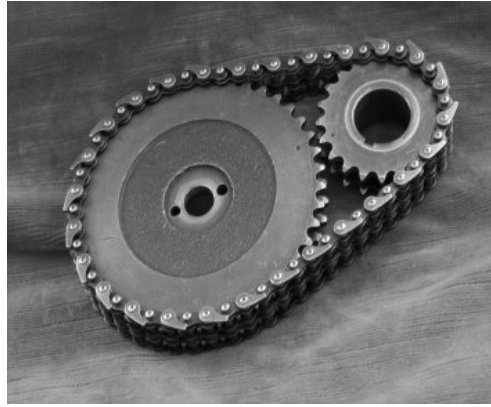
by Lygeum

Zero-cross

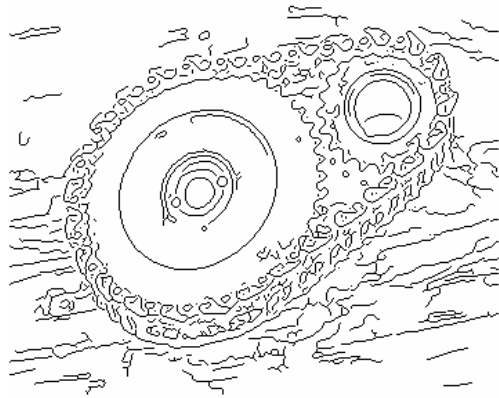


by Lygeum

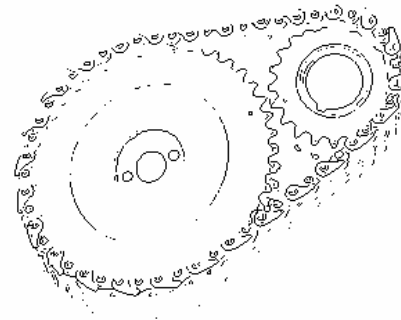
Propuesto



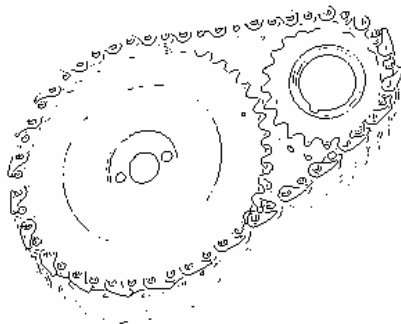
piezas_356x292 (356x292)



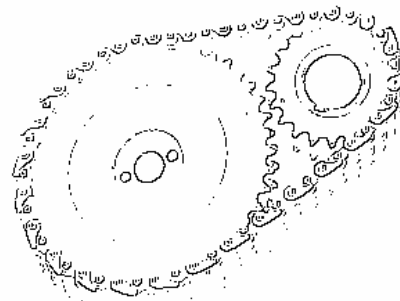
Canny



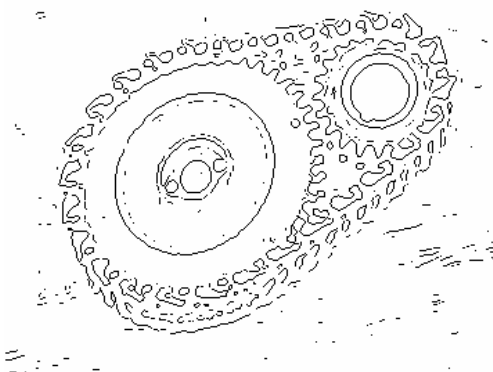
Sobel



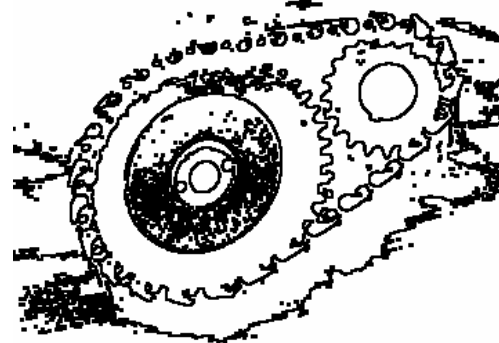
Prewitt



Roberts



Zerocross



Propuesto



punte_356x292 (356x292)



Canny



Sobel



Prewitt



Roberts



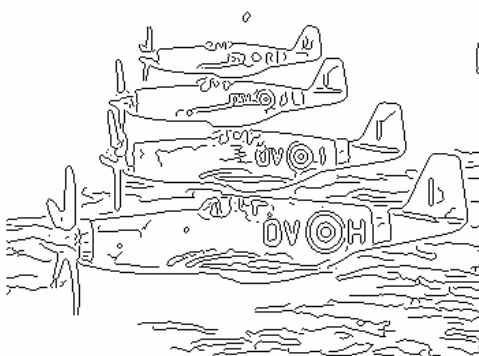
Zerocross



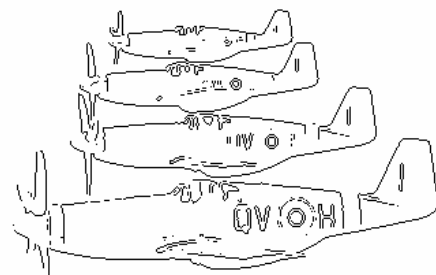
Propuesto



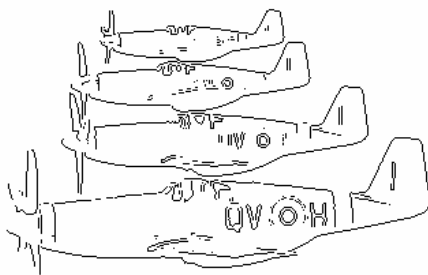
Avion1 (350x292)



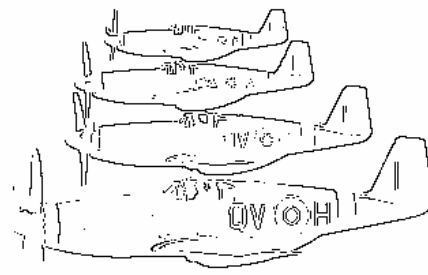
Canny



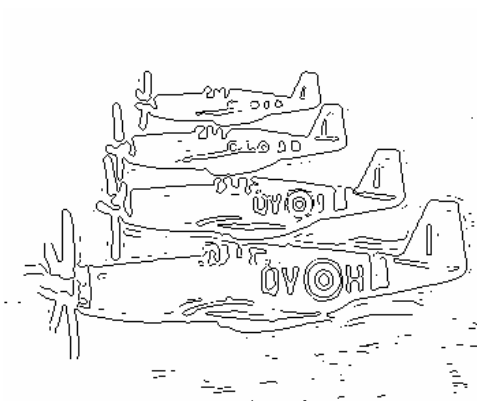
Sobel



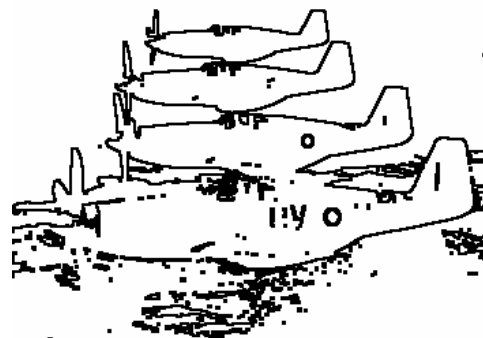
Prewitt



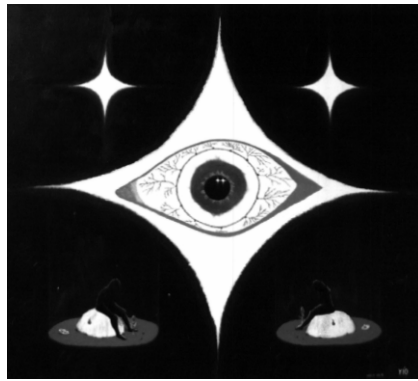
Roberts



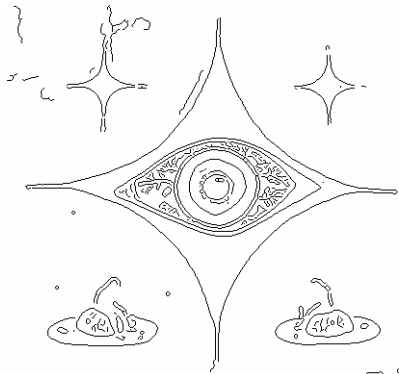
Zero-cross



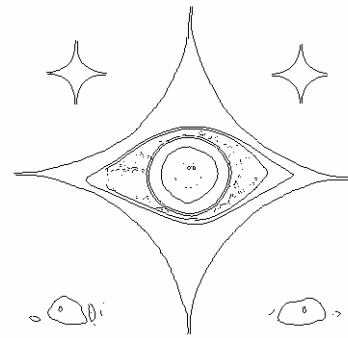
Propuesto



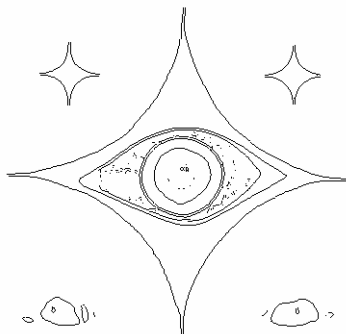
Nosveo (433x389)



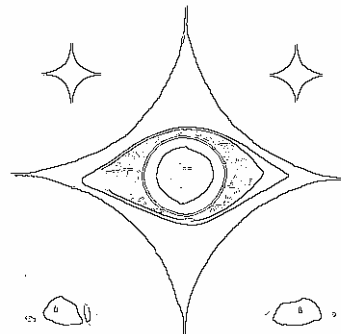
Canny



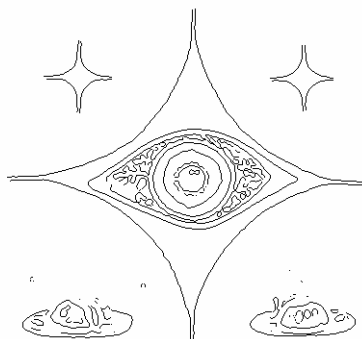
Sobel



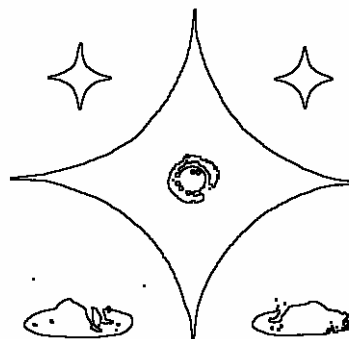
Prewitt



Roberts



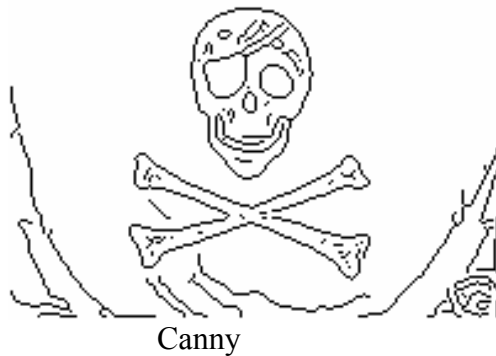
Zerocross



Propuesto



Pirata (200x150)



Canny



Sobel



Prewitt



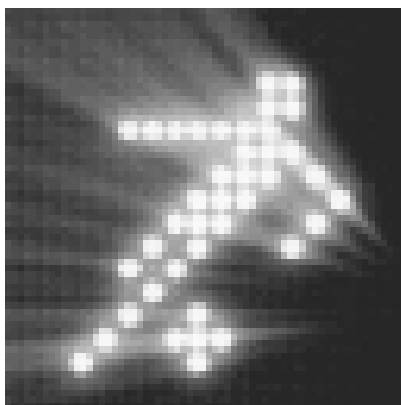
Roberts



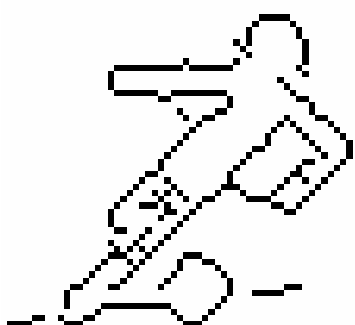
Zerocross



Propuesto



Soccer_64 (64x64)



Canny



Sobel



Prewitt



Roberts



Zerocross

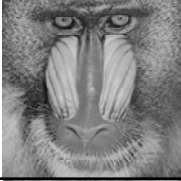






























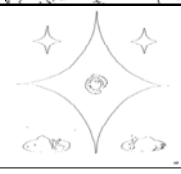
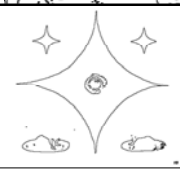
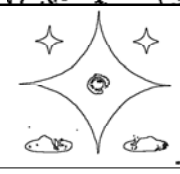
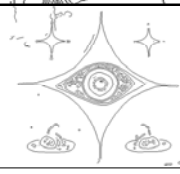
























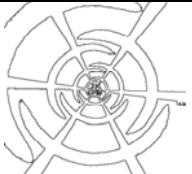
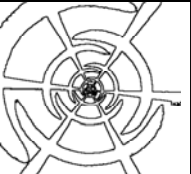

Propuesto

Apéndice D

Resultados de aplicar distintas mascarar en técnicas de detección de bordes

Los resultados que han mostrando en el apéndice B de la técnica propuesta en esta Tesis para la detección de bordes, corresponden a una mascara 3x3. En este apéndice mostramos los resultados pero en el caso de corresponder distintas mascarar en los tamaños de 1x2, 2x2 y 3x3 píxeles.

	Imagen Original	1x2	2x2	3x3	Canny
Baboon (512x512)					
Baboon (115x115)					
Barbara (512x512)					
Camerman (256x256)					
Goldhill (512x512)					
Lena (222x208)					
Nosveo (389x433)					

Peppers (512x512)					
Peppers (115x115)					
Pirata (150x200)					
Soccer (64x64)					
Tuneldevertigo (368x381)					

Bibliografía

[AMAT02] P. Amato, A. Di Nola, B. Gerla: “Neural Networks and Rational Łukasiewicz Logic”, Proceedings NAFIPS. Annual Meeting of the North American. pp. 506-510, 2002.

[ASIC] ASIC.ws: <http://www.asics.ws/>

[AZRI82] A. Rosenfeld, A. C. Kak: “Digital Picture Processing”, Volume 1, Academic Press, 1982.

[BARR06] A. Barriga, S. Sánchez Solano, P. Brox, A. Cabrera, I. Baturone: “Modelling and Implementation of Fuzzy Systems based on VHDL”, International Journal of Approximate Reasoning, vol. 41, issue 2, pp. 164-278, 2006.

[BARR07a] A. Barriga, N.M. Hussein: “Implementación de un circuito para compresión de imágenes aplicando lógica difusa”, XIII Taller Iberchip (IWS'2006), Lima (Perú), Marzo 2007.

[BARR07b] A. Barriga, N.M. Hussein: “Image Compression based on Fuzzy Logic and its Hardware Implementation”, Int. Symposium on Innovations in Intelligent System and Applications (INISTA'2007), Estambul (Turquía), June 2007.

[BARR08] A. Barriga, N.M. Hussein Hassan: “Design and Implementation of an Edge Detection Circuit Based on Soft Computing”, International Conference on Industrial Informatics and Systems Engineering (IISE 2008) Glasgow (UK), 22-24 July 2008.

[BHAS93a] V. Bhaskaran, B.K. Natarajan, K. Konstantinides: “Lossy Compression of Images Using Piecewise-Linear Approximation” Hewlett-Packard Technical Report HP-93-10, Feb. 1993.

[BHAS93b] V. Bhaskaran, B.K. Natarajan, K. Konstantinides: “Optimal piecewise-linear compression of images”, IEEE Data Compression Conference (DCC’93), pp. 168-177, Utah, Apr. 1993.

[BLAN08] J. Blanc-Talon, S. Bourennane, W. Philipps: “Advanced Concepts for Intelligent Vision Systems”, International Conference, ACIVS 2008, Juanles –Pins, France, October 20-24, Springer, 2008.

[BLOC06] I. Bloch, A. Petrosino, A. Tettamanzi: “Fuzzy Logic and Applications”, 6th International Conference, WILF 2005, Crema, Italy, Springer, 2006.

[BOCC01] G. Boccignone, M. Ferrero: “Multiscale Contrast Enhancement”, Electronics Letters, Vol.37 No.12, June 2001

[BRIN99] R. Brinkmann: “The Art and Science of Digital Compositing”, Morgan Kaufmann, 1999.

[BRYA07] D. Salomon, G. Motta, D. Bryant: “Data Compression: The Complete reference”, Springer, 2007.

[BURG07] W. Burger: “Digital Image Processing: An Algorithmic Introduction Using Java”, Springer, 2007.

[CAMP02] J. B. Cambell: “Introduction to Remote Sensing”, Taylor & Francis, 2002.

[CANN86] J.F. Canny: “A computational approach to edge detection”, IEEE Trans. Pattern Analysis and Machine Intelligence, pp. 679-698, 1986.

[CAO02] L. Cao, Z.K. Shi, E.K.W. Chenp: “Fast automatic multilevel thresholding method”, Electronics Letters, vol. 38, no. 16, pp.868-870, 2002.

[CARR94] L. Carretero, A. Fimia, Y A. Meléndez: “An Entropy Study of Imaging Quality in Holographic Optical Elements”, Optics Letters, Vol. 19, Issue 17, pp. 1355-1357, 1994.

[CARR96] L. Carretero, A. Fimia, Y A. Belendez: “Experimental Evaluation of Entropy for Transmission Holographicoptical Elements”, Applied Physics B: Lasers and Optics, Springer-1996.

[CAST98] J.L. Castro, E. Trillas: “The logic of neural networks”, Mathware and Soft Computing, vol 5, pp. 23-27, 1998.

[CCIT92] CCITT: “Information Technology –Digital Compression and Coding of Continous-Tone Still Images- Requirements and Guidelines. Recommendation T.81”, CCITT, 1992.

[CHAO06] Z. Chao, Z. Jia-shu, C. Hui: “Local Fuzzy Entropy-Based Transition Region Extraction and Thresholding”, International Journal of Information Technology, vol.12, no. 6, pp. 19-25, 2006

[CHEN99] C.W. Chen, Y-Q. Zhang: “Visual Information Representation, Communication, and Image Processing”, CRC Press, 1999.

[CHEN06] Z.Y. Chen, B.R. Abidi, D.L. Page, M.A. Abidi: “Gray-Level Grouping (GLG): An Automatic Method for Optimized Image Contrast Enhancement”, IEEE Transaction on Image Processing, Vol. 15, No.8, Agust 2006.

[CHO00] H.H. Cho, C.H. Choi, B.H. Kwon, M.R. Choi: "A Design of Contrast Controller for Image Improvement of Multi-Gray Scale Image ", 2nd IEEE Asia Pacific Conference on ASICs, pp. 131-133, Aug 2000.

[DATE01] C. J. Date, Sergio Luis Maria Ruiz Faudón, and Felipe López Garmino: "Introducción a los sistemas de bases de datos", Pearson Educación, 2001

[EDWA99] Edward R. Dougherty, "Electronic Imaging technology", Spie Press, 1999.

[ELIZ97] J.J.E Elizondo, L.E.P Maestre: "Fundamentos para el procesamiento de imágenes", Universidad Autónoma de Baja California UABC, 1997.

[FORE00] F-Vargas, M.G. Rojas-Camacho: "New formulation in image thresholding using fuzzy logic", 11th Portuguese Conf. on Pattern Recognition, pp. 117-124, 2000.

[FORT01] L.Fortuna, R. Caponetto, M. Lavorgna, G. Rizzotto, G. Nunnari, M.G. Xibilia: "Soft computing: New Trends and Applications", Springer, 2001.

[GONZ87] R.C. Gonzalez, P.A. Wintz: "Digital Image Processing", Addison-Wesley Publishing Company, 1987.

[GONZ96] R.C. Gonzalez, R Woods: "Procesamiento digital de imagenes", Ediciones Díaz de Santos, 1996.

[GONZ02] R.C. González, R.E. Woods: "Digital Image Processing", Prentice Hall Int, 2002.

[GONZ07] R.C. González, R.E. Woods: "Digital Image Processing", Prentice Hall Int, 2007.

[GRAU03] J.F.P Grau, "Técnicas de análisis de imagen", Tesis Doctoral, Universitat de València, 2003.

[HASS06] N.M.H Hassan, A Barriga: “Lineal Image Compression Based on Łukasiewicz’s Operators”, International Conference in Lecture Notes in Computer Science, Springer, 2006.

[HEAT96] M.D. Heath: “A Robust Visual Method For Assessing The Relative Performance Of Edge Detection Algorithms”, MsC Thesis, University of South Florida, 1996.

[HIRO99] K. Hirota, W. Pedrycz, “Fuzzy Relational Compression”, IEEE Trans. in System, Man and Cybernetics – Part B: Cybernetics, vol. 29, no. 3, pp. 407-415, June 1999.

[HUAN95] Huang, L.K., Wang, M.J: “Image thresholding by minimizing the measure of fuzziness”, Pattern Recognition, vol. 28, pp. 41-51, 1995.

[HUSS05a] N.M. Hussein Hassan, A. Barriga, S. Sánchez-Solano: “Implementación de funciones lineales a tramos mediante operadores de Łukasiewicz”, XI Workshop IBERCHIP, Salvador de Bahía (Brasil), 2005.

[HUSS05b] N. M. Hussein, A. Barriga, S. Sánchez-Solano: “Piecewise Linear Function Interpolation Using Łukasiewicz’s Operators”, Int. Symposium on Innovations in Intelligent Systems and Applications (INISTA’2005), Istanbul (Turkey), 2005.

[HUSS08a] N.M. Hussein Hassan, A. Barriga: “Hardware Implementation of a Soft Computing Technique for Edge Detection”, International Conference of Signal and Image Engineering (ICSIE 08), London (UK), Jul. 2008

[HUSS08b] N.M.Hussein, A. Barriga: “Implementación sobre FPGA de una Técnica Basada en Soft Computing para la Detección de Bordos en Imágenes”, Jornadas de Computación Reconfigurable y Aplicaciones (JCRA’2008), Móstoles, Madrid, Sept. 2008.

[HUSS09] N.M. Hussein Hassan, A. Barriga: “High Speed Soft Computing based Circuit for Edges Detection in Images”, in L. Gelman, N. Balkan, and S.I. Ao (editors): “Advances in Electrical Engineering and Computational Science”, Springer, 2009.

[JAHN05] B. Jahne: “Digital Image Processing”, Springer, 2005

[JOSE04] J.A. Garcia, R. Rodriguez-Sanchez, J. Fernandez-Valdivia: “Progressive Image Transmission”, Spie Press, 2004.

[JPEG] <http://www.jpeg.org/>

[KAME07] M. Kamel, A. Campilho: “Image Analysis and Recognition”, 4th International Conference, ICIAR 2007, Motreal. Canada, August 22-24, Springer, 2007.

[KAUF75] A. Kaufmann: “Introduction to the theory of fuzzy subsets”, Academic Press, 1975.

[KERR00] E.E. Kerre, M. Nachtgeael: “Fuzzy Techniques in Image Processing”, Springer, 2000.

[KHEL91] A. Khellaf, A. Beghdadi, H. Dupoisot: “Entropic Contrast Enhancement”, IEEE Transactions On Medical Imaging, vol. 10, no. 4, pp- 589-592, Dec. 1991.

[KIM99] S.Y. Kim, D. Han, S.J. Choi and J.S. Park: “Image Contrast Enhancement Based on the Piecewise-Linear Approximation of CDF “, IEEE Transactions on Consumer Electronics, Vol. 45, No. 3, pp. 828-834, Aug. 1999.

[KIM01] H.C Kim, B.H. Kwon, M.R. Choi “An Image Interpolator with Image Improvement for LCD Controller”, IEEE Transactions on Consumer Electronics, vol. 47, no. 2, pp. 263-271, May 2001.

[KIRI02] N.V. Kirianaki, S.Y. Yurish, O. Shapk: “Data Acquisition and Signal Processing for Smart Sensors”, John Wiley and Sons, 2002.

[KONG91] S.-G. Kong, B. Kosko: "Adaptive Fuzzy System for Transform Image Coding", International Joint Conference on Neural Networks, IJCNN-91-Seattle, July 1991.

[LIAO01] P.-S., Liao, T.-S. Chen, P.C. Chung: "A Fast Algorithm for Multilevel Thresholding", Journal of Information Science and Engineering, vol. 17, pp. 713-727, 2001.

[MARR80] D. Marr, E. Hildreth: "Theory of Edge Detection", Proceedings of the Royal Society London, pp. 187-217, 1980.

[MART01] D. Martin, C. Fowlkes, D. Tal, J. Malik: "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics", International Conference of Computer Vision (ICCV), 2001.

[MIAN99] J. Miano: "Compressed Image File Formates", Addison-Wesley, 1999.

[NACH03] M. Nachtegael, D.V.D Weken, D.V De Ville, E. E. Kerre: "Fuzzy Filters for Image Processing" Springer, 2003.

[NACH07] M. Nchtegael, D.V der Weken, E.E. Kerre, W. Philipps: "Soft Computing in Image Processing", Springer, 2007.

[NGUY03] H.T. Nguyen, V. Kreinovich, A. Di Nola: "Which truth values in fuzzy logics are definable?," Int. J. Intelligent Systems, Wiley Interscience, vol. 18, no. 3, pp. 1057-1064, Oct. 2003.

[NOLA99] A. Di Nola, "Algebraic analysis of Łukasiewicz logic: " ESSLLI, Summer School, Utrecht, 1999.

[NOLA03] A. Di Nola, A. Lettieri: “Formulas of Łukasiewicz’s logic represented by hyperplanes,” 10th International Fuzzy Systems Association World Congress (IFSA), Istanbul, Turkey, pp. 189-194, 2003.

[OH06] J.-T. Oh and W.-H. Kim: “EWFCM Algorithm and Region-Based Multi-level Thresholding”, LNAI, vol. 4223, pp. 864-873, 2006.

[OPEN] <http://www.opencores.org>

[OTSU78] N. Otsu: “A Threshold Selection Method from Gray Level Histogram”, IEEE Trans. on Systems, Man and Cybernetics, vol. 8, pp. 62-66, 1978.

[OVCH02] S. Ovchinnikov: “Max-min representation of piecewise linear functions”, Contributions to Algebra and Geometry, vol 43, pp. 297-302, 2002.

[PENN04] W.B. Pennebaker, J.L. Mitchell, “JPEG: Still Image Data Compression Standard”, Kuwer Academic Pub., 2004.

[PITA00] I. Pitas: “Digital Image Processing Algorithms and Applications”, John Wiley&Sons, 2000.

[PREW70] J.M.S. Prewitt: “Object enhancement and extraction”, in A. Rosenfeld and B. S. Lipkin, editors, “Picture Processing and Psychophysics”, Academic Press, New York, pp. 75-149, 1970.

[QSHI00] Q.Y. Shi, H. Sun: “Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards”, CRC Press, 2000.

[RAMI01] J. Ramírez, “Nuevas estructuras RNS para la síntesis VLSI de sistemas de procesamiento digital de señales”, Tesis Doctoral, Universidad de Granada, 2001.

[REUS06] B. Reusch: “Computational intelligence, Theory and Applications”, International Conference Fuzzy Days, Dortmund, Germany, Sept. 18-20, Springer, 2006.

[RICHA03] R.B. Merrill: “Vertical Color Filter Detector Group and Array,” U. S. Patent 6,632,701, Oct. 2003.

[RICH06] J.A. Richards, X. Jia: “Remote Sensing Digital Image Analysis”, Birkhauser, 2006.

[ROBE65] L.G. Roberts: “Machine perception of three-dimensional solids”, in J. T. Tippet et al., editor: “Optical and Electro-Optical Information Processing”, MIT Press, Cambridge, Massachusetts, pp. 159-197, 1965.

[ROSE90] C. Rosenberg: “A Lossy Image Compression Based on Nonuniform Sampling and Interpolation of Image Intensity Surfaces”, M.S. Thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1990.

[RUSS07] Russ, J.C, “The Image Processing Handbook”, Boca Raton, Florida, CRC Press, 1995.

[SALO96] M. Salotti, F. Bellet, C. Garbay: “Evaluation of Edge Detectors: Critics and Proposal”, Proc. Workshop Performance Characteristics Vision Algorithms, 1996.

[SALO04] David Salomon, “Data compression: The Complete Reference”, Springer, 2004.

[SAUP98] D. Saupe: “Optimal Piecewise Linear Image Coding”, SPIE Visual Communication and Image Processing (VCIP’98), San Jose, Jan. 1998.

[SCOT05] E.S. Umbaugh: “Computer Imaging: Digital Image Analysis and Processing”, CRC Press, 2005.

[SEZG04] M. Sezgin, B. Sankur: “Survey over image thresholding techniques and quantitative performance evaluation”, *Journal of Electronic Imaging*, vol. 13, pp. 146-165, 2004.

[SLUD07] G. Sluder, D. E. Wolf: “Digital Microscopy”, Academic Press, 2007.

[STEU96] A. Steudel, M. Glesner: “Image Coding with Fuzzy Region-Growing Segmentation”, In Proc. IEEE Int. Conf. on Image Proc., Lausanne, Suisse, September, 1996.

[TASI04] T. Acharya, Ping-Sing Tasi: “JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures”, John Wiley and Sons, 2004.

[TERR08] M. Terras: “Digital Images for the Information Professional”, Ashgate Publishing, Ltd, 2008.

[UMBA05] S.E. Umbaugh: “Computer Imaging : Digital Image Analysis and Processing”, CRC Press, 2005.

[VERD02] J.M.A Verde, J. Pujol: “Tecnología del Color”, Universitat de València, 2002.

[VIDA03] M. Vidaurrazga, W.Domínguez: “Evaluación de la Calidad de Imágenes Medicas”, Memorias V Congresos de la sociedad Cubana de Bioingeniería, Habana, Junio 10 al 13 de 2003.

[WADE94] G. Wade: “Signal Coding and Processing”, Cambridge University Press, 1994.

[WHIT05] J.C. Whitaker: “The Electronics Handbook”, CRC Press, 2005.

[WILL79] W K. Pratt: “Image Transmission Techniques”, Academic Press, 1979.

[WILL93] W.B. Pennebaker, J.L. Mitchell: “JPEG Still Image Data Compression Standard”, Springer 1993.

[XILI05] Xilinx: “Spartan3 Starter Kit Board User Guide”, Xilinx, 2005.

[YAGE79] R.R. Yager: “On the measure of fuzziness and negation. Part 1: membership in the unit interval”. *Int. Journal of Genet. Syst.*, vol. 5, pp. 221-229, 1979.

[YANG03] X. Yang, D. Ruan, K. Qin, J. Liu: “Lattice-valued logic: Fuzziness and Soft-computing”, Springer, 2003.

[YOUN88] I.T. Young: “Sampling Density and Quantitative Microscopy”. *Analytical and Quantitative Cytology and Histology*, vol. 10, pp. 269-275 1988.

[YOUN98] I.T. Young, J. J. Gerbrands, L.J.V Vilet: “Fundamental of Image Processing”, Delft University of Technology, Netherlands, 1998.

[ZADE94] L.A. Zadeh: “Fuzzy Logic, Neural Networks, and Soft Computing”, *Communications of the ACM*, Vol.37 No.3, pages 77-84, March 1994.

