



UNIVERSIDAD DE CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR
Ingeniería Técnica en Informática de Sistemas

PROYECTO DE FIN DE CARRERA

Marcas de agua en imágenes digitales

MANUAL TÉCNICO

MANUAL DE USUARIO

MANUAL DE CÓDIGO

Autor: RAÚL PÉRULA MARTÍNEZ

Directora: ÁNGELA ROJAS MATAS

Córdoba, Noviembre de 2009

Dña. Ángela Rojas Matas, Catedrática de Escuelas Universitarias, adscrita al Departamento de Matemáticas.

INFORMA

Que ha dirigido el Proyecto de Fin de Carrera de la Titulación de Ingeniería Técnica en Informática de Sistemas, denominado “Marcas de Agua en Imágenes Digitales”, el cual ha sido realizado por Raúl Pérrula Martínez, reuniendo, a su juicio, las condiciones exigidas en este tipo de trabajo.

Córdoba, noviembre de 2009

Dña. Ángela Rojas Matas

ÍNDICE DE CONTENIDO

ÍNDICE DE CONTENIDO	VI
ÍNDICE DE TABLAS.....	XVIII
CAPÍTULO 1: INTRODUCCIÓN	1
1.1. Criptografía, esteganografía y marcas de agua	2
1.1.1. Criptografía.....	2
1.1.2. Esteganografía	2
1.1.3. Marcas de agua digitales.....	3
1.1.4. Aplicaciones de las marcas de agua digitales.....	4
1.1.5. Requerimientos de una buena marca de agua digital.....	4
1.2. Definición del problema real	5
1.3. Definición del problema técnico	5
1.3.1. Funcionamiento.....	5
1.3.2. Entorno.....	6
1.3.3. Vida esperada	6
1.3.4. Ciclo de mantenimiento.....	6
1.3.5. Competencia	7
1.3.6. Aspecto externo	7
1.3.7. Estandarización.....	7
1.3.8. Calidad y fiabilidad.....	7
1.3.9. Programación de tareas.....	7
1.3.10. Pruebas.....	8
1.3.11. Seguridad.....	8
CAPÍTULO 2: OBJETIVOS.....	11
2.1. Objetivos del proyecto	11
CAPÍTULO 3: ANTECEDENTES	13
3.1. Introducción.....	13
3.2. Descripción de los algoritmos.....	13
3.2.1. Algoritmos de técnicas basadas en correlación	13

Introducción	13
Estructura de la Marca de Agua	14
Proceso de ocultación de la marca	14
Proceso de extracción de la marca	15
3.2.2. Algoritmo de Cox	15
Introducción	15
Estructura de la Marca de Agua	15
Proceso de ocultación de la marca	16
Proceso de extracción de la marca	17
3.2.3. Algoritmos basados en la DCT	17
Introducción	17
Primer Método: DCT	17
Estructura de la Marca de Agua	17
Proceso de ocultación de la marca	17
Proceso de extracción de la marca	19
Segundo Método: DCT y correlación	19
Estructura de la Marca de Agua	19
Proceso de ocultación de la marca	19
Proceso de extracción de la marca	20
3.2.4. Algoritmos basados en CDMA.....	20
Introducción	20
Estructura de la Marca de Agua	21
Proceso de ocultación de la marca	21
Proceso de extracción de la marca	21
3.2.5. Algoritmos basados en la DWT.....	22
Introducción	22
Primer Método: CDMA en el dominio wavelet	23
Estructura de la Marca de Agua	23
Proceso de ocultación de la marca	23
Proceso de extracción de la marca	23
Segundo Método: DWT y la transformada de Haar	24
Estructura de la Marca de Agua.....	24

Proceso de ocultación de la marca	24
Proceso de extracción de la marca	25
Tercer Método: DWT basado en la paridad	26
Estructura de la Marca de Agua	26
Proceso de ocultación de la marca	26
Proceso de extracción de la marca	27
3.2.6. Algoritmo basado en SVD	28
Introducción	28
Primer Método: SVD	28
Estructura de la Marca de Agua	28
Proceso de ocultación de la marca	29
Proceso de extracción de la marca	29
Comentarios	30
Segundo Método: SVD con transformada de Arnold	31
Estructura de la Marca de Agua	32
Proceso de ocultación de la marca	32
Proceso de extracción de la marca	34
Tercer Método: SVD basado en el intercambio de valores	34
Estructura de la Marca de Agua	34
Proceso de ocultación de la marca	34
Proceso de extracción de la marca	36
Comentarios	38
Cuarto Método: SVD basado en el orden en los coeficientes	39
Estructura de la Marca de Agua	39
Proceso de ocultación de la marca	39
Proceso de extracción de la marca	41
Quinto Método: SVD basado en la proximidad a un intervalo	41
Estructura de la Marca de Agua	41
Proceso de ocultación de la marca	41
Proceso de extracción de la marca	42
3.2.7. Algoritmo basado en LSB	42
Introducción	42

Estructura de la Marca de Agua	43
Proceso de ocultación de la marca	43
Proceso de extracción de la marca	43
3.2.8. Algoritmo basado en las Secuencias Caóticas.....	43
Introducción	43
Primer Método: Secuencias Caóticas	44
Estructura de la Marca de Agua.....	44
Proceso de ocultación de la marca	44
Proceso de extracción de la marca	45
Segundo Método: Secuencias Caóticas y DCT.....	45
Estructura de la Marca de Agua.....	45
Proceso de ocultación de la marca	45
Proceso de extracción de la marca	47
3.2.9. Algoritmo basado en PCA.....	47
Introducción	47
Primer Método: PCA	49
Estructura de la Marca de Agua	49
Proceso de ocultación de la marca	49
Proceso de extracción de la marca	49
Segundo Método: PCA para construir la imagen de referencia.....	50
Estructura de la Marca de Agua	50
Proceso de ocultación de la marca	50
Proceso de extracción de la marca	51
CAPÍTULO 4: RESTRICCIONES	53
4.1. Factores dato	53
4.2. Factores estratégicos	53
CAPÍTULO 5: RECURSOS	55
5.1. Recursos humanos	55
5.2. Recursos hardware.....	55
5.3. Recursos software	56
CAPÍTULO 6: ESPECIFICACIÓN DE REQUISITOS	57
6.1. Introducción	57

6.2. Algoritmos para tratado de marcas de agua	58
6.2.1. Inserción de la marca de agua.....	58
6.2.2. Extracción de la marca de agua.....	58
6.3. Pruebas de robustez.....	59
6.3.1. Ataques	59
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES	61
7.1. Introducción.....	61
7.1.1. PSNR.....	61
7.1.2. Similaridad y correlación.....	62
7.2. Tipos de ataques	62
7.3. Pruebas individuales	63
7.3.1. Algoritmos de técnicas basadas en correlación.....	63
7.3.1.1. Pruebas de parámetros	63
7.3.1.2. Pruebas de ataques	69
7.3.1.3. Pruebas de tiempos de ejecución	75
7.3.1.4. Conclusiones individuales.....	76
7.3.2. Algoritmo de Cox.....	77
7.3.2.1. Pruebas de parámetros	77
7.3.2.2. Pruebas de ataques	81
7.3.2.3. Pruebas de tiempos de ejecución	87
7.3.2.4. Conclusiones individuales.....	87
7.3.3. Algoritmos basados en la DCT	88
7.3.3.1. Primer Método: DCT.....	88
7.3.3.1.1. Pruebas de parámetros.....	88
7.3.3.1.2. Pruebas de ataques.....	91
7.3.3.1.3. Pruebas de tiempos de ejecución.....	97
7.3.3.1.4. Conclusiones individuales	98
7.3.3.2. Segundo Método: DCT y correlación	98
7.3.3.2.1. Pruebas de parámetros.....	98
7.3.3.2.2. Pruebas de ataques.....	101
7.3.3.2.3. Pruebas de tiempos de ejecución.....	107
7.3.3.2.4. Conclusiones individuales	108

7.3.4.	Algoritmo basado en CDMA	108
7.3.4.1.	Pruebas de parámetros	108
7.3.4.2.	Pruebas de ataques	111
7.3.4.3.	Pruebas de tiempos de ejecución	117
7.3.4.4.	Conclusiones individuales.....	118
7.3.5.	Algoritmos basados en la DWT	118
7.3.5.1.	Primer Método: CDMA en el dominio wavelet	118
7.3.5.1.1.	Pruebas de parámetros.....	118
7.3.5.1.2.	Pruebas de ataques.....	121
7.3.5.1.3.	Pruebas de tiempos de ejecución.....	127
7.3.4.1.4.	Conclusiones individuales	128
7.3.5.2.	Segundo Método: DWT y la transformada de Haar.....	128
7.3.5.2.1.	Pruebas de parámetros.....	128
7.3.5.2.2.	Pruebas de ataques.....	132
7.3.5.2.3.	Pruebas de tiempos de ejecución.....	137
7.3.5.2.4.	Conclusiones individuales	138
7.3.5.3.	Tercer Método: DWT basado en la paridad	138
7.3.5.3.1.	Pruebas de parámetros.....	138
7.3.5.3.2.	Pruebas de ataques.....	145
7.3.5.3.3.	Pruebas de tiempos de ejecución.....	152
7.3.5.3.4.	Conclusiones individuales	152
7.3.6.	Algoritmos basados en la SVD	153
7.3.6.1.	Primer Método: SVD	153
7.3.6.1.1.	Pruebas de parámetros.....	153
7.3.6.1.2.	Pruebas de ataques.....	155
7.3.6.1.3.	Pruebas de tiempos de ejecución.....	161
7.3.6.1.4.	Conclusiones individuales	161
7.3.6.2.	Segundo Método: SVD con transformada de Arnold.....	161
7.3.6.2.1.	Pruebas de parámetros.....	161
7.3.6.2.2.	Pruebas de ataques.....	164
7.3.6.2.3.	Pruebas de tiempos de ejecución.....	170
7.3.6.2.4.	Conclusiones individuales	171

7.3.6.3. Tercer Método: SVD basado en el intercambio de valores.....	171
7.3.6.3.1. Pruebas de parámetros.....	171
7.3.6.3.2. Pruebas de ataques.....	175
7.3.6.3.3. Pruebas de tiempos de ejecución.....	181
7.3.6.3.4. Conclusiones individuales	182
7.3.6.4. Cuarto Método: SVD basado en el orden de los coeficientes	182
7.3.6.4.1. Pruebas de parámetros.....	182
7.3.6.4.2. Pruebas de ataques.....	185
7.3.6.4.3. Pruebas de tiempos de ejecución.....	191
7.3.6.4.4. Conclusiones individuales	192
7.3.6.5. Quinto Método: SVD basado en la proximidad a un intervalo.....	192
7.3.6.5.1. Pruebas de parámetros.....	192
7.3.6.5.2. Pruebas de ataques.....	196
7.3.6.5.3. Pruebas de tiempos de ejecución.....	202
7.3.6.5.4. Conclusiones individuales	203
7.3.7. Algoritmo basado en LSB	203
7.3.7.1. Pruebas de parámetros	203
7.3.7.2. Pruebas de ataques	204
7.3.7.3. Pruebas de tiempos de ejecución	210
7.3.7.4. Conclusiones individuales.....	211
7.3.8. Algoritmo basado en las Secuencias Caóticas	211
7.3.8.1. Primer Método: Secuencias Caóticas.....	211
7.3.8.1.1. Pruebas de parámetros.....	211
7.3.8.1.2. Pruebas de ataques.....	215
7.3.8.1.3. Pruebas de tiempos de ejecución.....	220
7.3.8.1.4. Conclusiones individuales	221
7.3.8.2. Segundo Método: Secuencias Caóticas y DCT.....	221
7.3.8.2.1. Pruebas de parámetros.....	221
7.3.8.2.2. Pruebas de ataques.....	229
7.3.8.2.3. Pruebas de tiempos de ejecución.....	235
7.3.8.2.4. Conclusiones individuales	235
7.3.9. Algoritmo basado en PCA	236

7.3.9.1. Primer Método: PCA.....	236
7.3.9.1.1. Pruebas de parámetros.....	236
7.3.9.1.2. Pruebas de ataques.....	240
7.3.9.1.3. Pruebas de tiempos de ejecución.....	246
7.3.9.1.4. Conclusiones individuales	246
7.3.9.2. Segundo Método: PCA para construir la imagen de referencia	246
7.3.9.2.1. Pruebas de parámetros.....	246
7.3.9.2.2. Pruebas de ataques.....	251
7.3.9.2.3. Pruebas de tiempos de ejecución.....	257
7.3.9.2.4. Conclusiones individuales	257
7.4. Pruebas comunes	257
7.4.1. Comparativa de tiempos de computación	257
7.4.2. Comparativa de tipos de algoritmos	260
7.4.3. Comparativa de resistencia a ataques.....	262
CAPÍTULO 8: FUTURAS MEJORAS	265
8.1. Introducción.....	265
8.2. Algoritmos.....	265
8.3. Programa	265
8.4. Interfaz.....	266
8.5. Documentación.....	266
BIBLIOGRAFÍA	267
APÉNDICE A: MANUAL DE USUARIO.....	271
A.1. Instalación	271
A.2. Desinstalación.....	274
A.3. Ejemplo de un caso de uso.....	274
APÉNDICE B: MANUAL DE CÓDIGO	277
B.1. Algoritmos	277
B.1.1. Algoritmos de técnicas basadas en correlación	277
Inserción (cor_insertar.m).....	277
Extracción (cor_recuperar.m).....	279
Extracción modificada (cor_recuperarmod.m)	281
B.1.2. Algoritmo de Cox	283

Inserción (cox_insertar.m)	283
Extracción (cox_recuperar.m)	285
B.1.3. Algoritmos basados en la DCT.....	286
B.1.3.1. Primer Método: DCT	286
Inserción (dct_insertar.m)	286
Extracción (dct_recuperar.m)	289
B.1.3.2. Segundo Método: DCT y correlación	291
Inserción (dctcor_insertar.m)	291
Extracción (dctcor_recuperar.m)	293
B.1.4. Algoritmo basado en CDMA.....	296
Inserción (cdma_insertar.m)	296
Extracción (cdma_recuperar.m)	297
B.1.5. Algoritmos basados en la DWT	299
B.1.5.1. Primer Método: CDMA en el dominio wavelet	299
Inserción (dwt_insertar.m)	299
Extracción (dwt_recuperar.m)	300
B.1.5.2. Segundo Método: DWT y la transformada de Haar	302
Inserción (dwtHaar_insertar.m).....	302
Extracción (dwtHaar_recuperar.m).....	304
B.1.5.3. Tercer Método: DWT basado en la paridad	306
Inserción (dwtParidad_insertar.m)	306
Inserción modificada (dwtParidad_insertarmod.m).....	307
Extracción (dwtParidad_recuperar.m)	308
B.1.6. Algoritmos basados en la SVD.....	310
B.1.6.1. Primer Método: SVD	310
Inserción (svd_insertar.m)	310
Extracción (svd_recuperar.m)	311
B.1.6.2. Segundo Método: SVD con transformada de Arnold	312
Inserción (svdArnold_insertar.m)	312
Extracción (svdArnold_recuperar.m)	314
B.1.6.3. Tercer Método: SVD basado en el intercambio de valores.....	315
Inserción (svdIntercambio_insertar.m)	315

Extracción (svdIntercambio_recuperar.m).....	317
Comentarios	318
Ataque 1 (svdIntercambio_ataque.m)	318
Ataque 2(svdIntercambio_ataque2.m)	319
B.1.6.4. Cuarto Método: SVD basado en el orden en los coeficientes.....	320
Inserción (svdOrden_insertar.m)	320
Extracción (svdOrden_recuperar.m)	322
B.1.6.5. Quinto Método: SVD basado en la proximidad a un intervalo	323
Inserción (svdProximidad_insertar.m)	323
Extracción (svdProximidad_recuperar.m)	325
B.1.7. Algoritmo basado en LSB	327
Inserción (lsb_insertar.m)	327
Extracción (lsb_recuperar.m)	328
B.1.8. Algoritmo basado en las Secuencias Caóticas	329
B.1.8.1. Primer Método: Secuencias Caóticas.....	329
Inserción (sc_insertar.m)	329
Extracción (sc_recuperar.m)	331
B.1.8.2. Segundo Método: Secuencias Caóticas y DCT.....	332
Inserción (scdct_insertar.m)	332
Extracción (scdct_recuperar.m)	334
Inserción (scdct_insertar_sin_permutacion.m).....	335
Extracción (scdct_recuperar_sin_permutacion.m).....	337
B.1.9. Algoritmo basado en PCA	338
B.1.9.1. Primer Método: PCA	338
Inserción (pca_insertar.m).....	338
Extracción (pca_recuperar.m).....	340
B.1.9.2. Segundo Método: PCA para construir la imagen de referencia	342
Inserción (pcaReferencia_insertar.m)	342
Extracción (pcaReferencia_recuperar.m).....	344
B.2. Ataques.....	346
B.2.1. Compresión JPEG (JPEG.m)	346
B.2.2. Inserción de ruido Gaussiano (noise.m)	346

B.2.3. Aplicación de un filtro de paso bajo basado en la media (meanFilter.m).....	347
B.2.4. Recortado de la imagen (cropping.m)	347
B.2.5. Escalado de la imagen (scaled.m).....	348
B.2.6. Rotación de la imagen (rotate.m)	348
B.2.7. Simulación de los ataques (simAtaques.m).....	349
B.3. Funciones Auxiliares.....	353
B.3.1. Correlación (correlacion.m)	353
B.3.2. Similaridad (similaridad.m)	354
B.3.3. Cálculo de los errores	355
B.3.3.1. En una matriz (errMatriz.m)	355
B.3.3.2. En un vector (errVector.m).....	355
B.3.4. PSNR (psnr.m).....	356

ÍNDICE DE TABLAS

Tabla 1: Ejemplos de marcas de agua	3
Tabla 2: Inserción y extracción por el algoritmo de técnicas basadas en correlación, k = 5	64
Tabla 3: Inserción y extracción por el algoritmo de técnicas basadas en correlación, k = 15	65
Tabla 4: Inserción y extracción por el algoritmo de técnicas basadas en correlación, k = 25	66
Tabla 5: Inserción y extracción modificada por el algoritmo de técnicas basadas en correlación, k = 1.....	67
Tabla 6: Inserción y extracción modificada por el algoritmo de técnicas basadas en correlación, k = 5.....	68
Tabla 7: Inserción y extracción modificada por el algoritmo de técnicas basadas en correlación, k = 25.....	69
Tabla 8: Ataque compresión JPEG 60% algoritmo técnicas basadas en correlación	70
Tabla 9: Ataque inserción ruido gaussiano algoritmo técnicas basadas en correlación	71
Tabla 10: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo técnicas basadas en correlación	72
Tabla 11: Ataque recortado de la imagen marcada algoritmo técnicas basadas en correlación	73
Tabla 12: Ataque escalado de la imagen marcada algoritmo técnicas basadas en correlación	74
Tabla 13: Ataque rotación de la imagen marcada algoritmo técnicas basadas en correlación	75
Tabla 14: Tiempos de ejecución del algoritmo de técnicas basadas en correlación	76
Tabla 15: Tiempos de ejecución del algoritmo de técnicas basadas en correlación (modificado)	76
Tabla 16: Inserción y extracción por el algoritmo de Cox, alpha = 0.05	78
Tabla 17: Inserción y extracción por el algoritmo de Cox, alpha = 0.1.....	79
Tabla 18: Inserción y extracción por el algoritmo de Cox, alpha = 0.5.....	80
Tabla 19: Ataque compresión JPEG 60% algoritmo de Cox	81
Tabla 20: Ataque inserción ruido gaussiano algoritmo de Cox	82
Tabla 21: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo de Cox.....	83
Tabla 22: Ataque recortado de la imagen marcada algoritmo de Cox.....	84
Tabla 23: Ataque escalado de la imagen marcada algoritmo de Cox.....	85

Tabla 24: Ataque rotación de la imagen marcada algoritmo de Cox	86
Tabla 25: Tiempos de ejecución del algoritmo de Cox	87
Tabla 26: Inserción y extracción por el algoritmo DCT, k = 10	89
Tabla 27: Inserción y extracción por el algoritmo DCT, k = 50	90
Tabla 28: Inserción y extracción por el algoritmo DCT, k = 100	91
Tabla 29: Ataque compresión JPEG 60% algoritmo DCT	92
Tabla 30: Ataque inserción ruido gaussiano algoritmo DCT	93
Tabla 31: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo DCT	94
Tabla 32: Ataque recortado de la imagen marcada algoritmo DCT	95
Tabla 33: Ataque escalado de la imagen marcada algoritmo DCT	96
Tabla 34: Ataque rotación de la imagen marcada algoritmo DCT.....	97
Tabla 35: Tiempos de ejecución del algoritmo DCT	98
Tabla 36: Inserción y extracción por el algoritmo DCT y correlación, k = 1	99
Tabla 37: Inserción y extracción por el algoritmo DCT y correlación, k = 5	100
Tabla 38: Inserción y extracción por el algoritmo DCT y correlación, k = 10	101
Tabla 39: Ataque compresión JPEG 60% algoritmo DCT y correlación	102
Tabla 40: Ataque inserción ruido gaussiano algoritmo DCT y correlación	103
Tabla 41: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo DCT y correlación	104
Tabla 42: Ataque recortado de la imagen marcada algoritmo DCT y correlación	105
Tabla 43: Ataque escalado de la imagen marcada algoritmo DCT y correlación	106
Tabla 44: Ataque rotación de la imagen marcada algoritmo DCT y correlación	107
Tabla 45: Tiempos de ejecución del algoritmo DCT y correlación	108
Tabla 46: Inserción y extracción por el algoritmo CDMA, k = 0.5	109
Tabla 47: Inserción y extracción por el algoritmo CDMA, k = 2	110
Tabla 48: Inserción y extracción por el algoritmo CDMA, k = 5	111
Tabla 49: Ataque compresión JPEG 60% algoritmo CDMA.....	112
Tabla 50: Ataque inserción ruido gaussiano algoritmo CDMA	113
Tabla 51: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo CDMA	114
Tabla 52: Ataque recortado de la imagen marcada algoritmo CDMA	115
Tabla 53: Ataque escalado de la imagen marcada algoritmo CDMA	116
Tabla 54: Ataque rotación de la imagen marcada algoritmo CDMA.....	117
Tabla 55: Tiempos de ejecución del algoritmo CDMA.....	118
Tabla 56: Inserción y extracción por el algoritmo CDMA en el dominio wavelet, k = 0.5	119
Tabla 57: Inserción y extracción por el algoritmo CDMA en el dominio wavelet, k = 2	120
Tabla 58: Inserción y extracción por el algoritmo CDMA en el dominio wavelet, k = 5	121

Tabla 59: Ataque compresión JPEG 60% algoritmo CDMA en el dominio wavelet	122
Tabla 60: Ataque inserción ruido gaussiano algoritmo CDMA en el dominio wavelet	123
Tabla 61: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo CDMA en el dominio wavelet.....	124
Tabla 62: Ataque recortado de la imagen marcada algoritmo CDMA en el dominio wavelet.....	125
Tabla 63: Ataque escalado de la imagen marcada algoritmo CDMA en el dominio wavelet.....	126
Tabla 64: Ataque rotación de la imagen marcada algoritmo CDMA en el dominio wavelet.....	127
Tabla 65: Tiempos de ejecución del algoritmo CDMA en el dominio wavelet	128
Tabla 66: Inserción y extracción por el algoritmo DWT y la transformada de Haar, $s = 1$, $t = 5$	129
Tabla 67: Inserción y extracción por el algoritmo DWT y la transformada de Haar, $s = 2$, $t = 10$	130
Tabla 68: Inserción y extracción por el algoritmo DWT y la transformada de Haar, $s = 4$, $t = 20$	131
Tabla 69: Ataque compresión JPEG 60% algoritmo DWT y la transformada de Haar..	132
Tabla 70: Ataque inserción ruido gaussiano algoritmo DWT y la transformada de Haar	133
Tabla 71: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo DWT y la transformada de Haar	134
Tabla 72: Ataque recortado de la imagen marcada algoritmo DWT y la transformada de Haar	135
Tabla 73: Ataque escalado de la imagen marcada algoritmo DWT y la transformada de Haar	136
Tabla 74: Ataque rotación de la imagen marcada algoritmo DWT y la transformada de Haar	137
Tabla 75: Tiempos de ejecución del algoritmo DWT y la transformada de Haar	138
Tabla 76: Inserción y extracción por el algoritmo DWT basado en la paridad, constante = 1.....	140
Tabla 77: Inserción y extracción por el algoritmo DWT basado en la paridad, constante = 10.....	141
Tabla 78: Inserción y extracción por el algoritmo DWT basado en la paridad, constante = 20.....	142
Tabla 79: Inserción modificada y extracción por el algoritmo DWT basado en la paridad, constante = 1	143
Tabla 80: Inserción modificada y extracción por el algoritmo DWT basado en la paridad, constante = 10	144
Tabla 81: Inserción modificada y extracción por el algoritmo DWT basado en la paridad, constante = 20	145

Tabla 82: Ataque compresión JPEG 60% algoritmo DWT basado en la paridad	146
Tabla 83: Ataque inserción ruido gaussiano algoritmo DWT basado en la paridad	147
Tabla 84: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo DWT basado en la paridad	148
Tabla 85: Ataque recortado de la imagen marcada algoritmo DWT basado en la paridad	149
Tabla 86: Ataque escalado de la imagen marcada algoritmo DWT basado en la paridad	150
Tabla 87: Ataque rotación de la imagen marcada algoritmo DWT basado en la paridad	151
Tabla 88: Tiempos de ejecución del algoritmo DWT basado en la paridad.....	152
Tabla 89: Tiempos de ejecución del algoritmo DWT basado en la paridad (modificado)	152
Tabla 90: Inserción y extracción por el algoritmo SVD	154
Tabla 91: Ataque compresión JPEG 60% algoritmo SVD	155
Tabla 92: Ataque inserción ruido gaussiano algoritmo SVD	156
Tabla 93: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo SVD	157
Tabla 94: Ataque recortado de la imagen marcada algoritmo SVD	158
Tabla 95: Ataque escalado de la imagen marcada algoritmo SVD	159
Tabla 96: Ataque rotación de la imagen marcada algoritmo SVD	160
Tabla 97: Tiempos de ejecución del algoritmo SVD	161
Tabla 98: Inserción y extracción por el algoritmo SVD con transformada de Arnold, alpha = 0.3	162
Tabla 99: Inserción y extracción por el algoritmo SVD con transformada de Arnold, alpha = 1	163
Tabla 100: Inserción y extracción por el algoritmo SVD con transformada de Arnold, alpha = 2	164
Tabla 101: Ataque compresión JPEG 60% algoritmo SVD con transformada de Arnold	165
Tabla 102: Ataque inserción ruido gaussiano algoritmo SVD con transformada de Arnold.....	166
Tabla 103: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo SVD con transformada de Arnold	167
Tabla 104: Ataque recortado de la imagen marcada algoritmo SVD con transformada de Arnold	168
Tabla 105: Ataque escalado de la imagen marcada algoritmo SVD con transformada de Arnold	169
Tabla 106: Ataque rotación de la imagen marcada algoritmo SVD con transformada de Arnold.....	170
Tabla 107: Tiempos de ejecución del algoritmo SVD con transformada de Arnold	171

Tabla 108: Inserción y extracción por el algoritmo SVD basado en el intercambio de valores, alpha = 1	172
Tabla 109: Inserción y extracción por el algoritmo SVD basado en el intercambio de valores, alpha = 2	173
Tabla 110: Inserción y extracción por el algoritmo SVD basado en el intercambio, alpha = 20.....	174
Tabla 111: Inserción y extracción por el algoritmo SVD basado en el intercambio de valores, alpha = 200	175
Tabla 112: Ataque compresión JPEG 60% algoritmo SVD basado en el intercambio de valores	176
Tabla 113: Ataque inserción ruido gaussiano algoritmo SVD en el intercambio de valores	177
Tabla 114: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo SVD basado en el intercambio de valores.....	178
Tabla 115: Ataque recortado de la imagen marcada algoritmo SVD basado en el intercambio de valores	179
Tabla 116: Ataque escalado de la imagen marcada algoritmo SVD basado en el intercambio de valores	180
Tabla 117: Ataque rotación de la imagen marcada algoritmo SVD basado en el intercambio de valores	181
Tabla 118: Tiempos de ejecución del algoritmo SVD basado en el intercambio de valores	182
Tabla 119: Inserción y extracción por el algoritmo SVD basado en el orden de los coeficientes, T = 0.001	183
Tabla 120: Inserción y extracción por el algoritmo SVD basado en el orden de los coeficientes, T = 0.012	184
Tabla 121: Inserción y extracción por el algoritmo SVD basado en el orden de los coeficientes, T = 1	185
Tabla 122: Ataque compresión JPEG 60% algoritmo SVD basado en el orden de los coeficientes.....	186
Tabla 123: Ataque inserción ruido gaussiano algoritmo SVD basado en el orden de los coeficientes.....	187
Tabla 124: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo SVD basado en el orden de los coeficientes	188
Tabla 125: Ataque recortado de la imagen marcada algoritmo SVD basado en el orden de los coeficientes	189
Tabla 126: Ataque escalado de la imagen marcada algoritmo SVD basado en el orden de los coeficientes	190
Tabla 127: Ataque rotación de la imagen marcada algoritmo SVD basado en el orden de los coeficientes	191
Tabla 128: SVD basado en el orden de los coeficientes	192

Tabla 129: Inserción y extracción por el algoritmo SVD basado en la proximidad a un intervalo, T = 5	193
Tabla 130: Inserción y extracción por el algoritmo SVD basado en la proximidad a un intervalo, T = 10	194
Tabla 131: Inserción y extracción por el algoritmo SVD basado en la proximidad a un intervalo, T = 60	195
Tabla 132: Inserción y extracción por el algoritmo SVD basado en la proximidad a un intervalo, T = 100	196
Tabla 133: Ataque compresión JPEG 60% algoritmo SVD basado en la proximidad a un intervalo	197
Tabla 134: Ataque inserción ruido gaussiano algoritmo SVD en la proximidad a un intervalo	198
Tabla 135: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo SVD basado en la proximidad a un intervalo	199
Tabla 136: Ataque recortado de la imagen marcada algoritmo SVD basado en la proximidad a un intervalo	200
Tabla 137: Ataque escalado de la imagen marcada algoritmo SVD basado en el proximidad a un intervalo	201
Tabla 138: Ataque rotación de la imagen marcada algoritmo SVD basado en la proximidad a un intervalo	202
Tabla 139: Tiempos de ejecución del algoritmo SVD basado en la proximidad a un intervalo	203
Tabla 140: Inserción y extracción por el algoritmo LSB	204
Tabla 141: Ataque compresión JPEG 60% algoritmo LSB	205
Tabla 142: Ataque inserción ruido gaussiano algoritmo LSB	206
Tabla 143: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo LSB	207
Tabla 144: Ataque recortado de la imagen marcada algoritmo LSB	208
Tabla 145: Ataque escalado de la imagen marcada algoritmo LSB	209
Tabla 146: Ataque rotación de la imagen marcada algoritmo LSB	210
Tabla 147: Tiempos de ejecución del algoritmo LSB	211
Tabla 148: Inserción y extracción por el algoritmo basado en secuencias caóticas, alpha = 1	212
Tabla 149: Inserción y extracción por el algoritmo basado en secuencias caóticas, alpha = 10	213
Tabla 150: Inserción y extracción por el algoritmo basado en secuencias caóticas, alpha = 50	214
Tabla 151: Ataque compresión JPEG 60% algoritmo basado en secuencias caóticas..	215
Tabla 152: Ataque inserción ruido gaussiano algoritmo basado en secuencias caóticas ..	216

Tabla 153: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo basado en secuencias caóticas	217
Tabla 154: Ataque recortado de la imagen marcada algoritmo basado en secuencias caóticas.....	218
Tabla 155: Ataque escalado de la imagen marcada algoritmo basado en secuencias caóticas.....	219
Tabla 156: Ataque rotación de la imagen marcada algoritmo basado en secuencias caóticas.....	220
Tabla 157: Tiempos de ejecución del algoritmo basado en secuencias caóticas	221
Tabla 158: Inserción y extracción por el algoritmo basado en secuencias caóticas y DCT, alpha = 0.1	223
Tabla 159: Inserción y extracción por el algoritmo basado secuencias caóticas y DCT, alpha = 0.3	224
Tabla 160: Inserción y extracción por el algoritmo basado en secuencias caóticas y DCT, alpha = 1	225
Tabla 161: Inserción y extracción por el algoritmo basado en secuencias caóticas y DCT sin permutación, alpha = 0.1	226
Tabla 162: Inserción y extracción por el algoritmo basado en secuencias caóticas y DCT sin permutación, alpha = 0.3	227
Tabla 163: Inserción y extracción por el algoritmo basado en secuencias caóticas y DCT sin permutación, alpha = 1	228
Tabla 164: Ataque compresión JPEG 60% algoritmo basado en secuencias caóticas y DCT	230
Tabla 165: Ataque inserción ruido gaussiano algoritmo basado en secuencias caóticas y DCT	230
Tabla 166: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo basado en secuencias caóticas y DCT	231
Tabla 167: Ataque recortado de la imagen marcada algoritmo basado en secuencias caóticas y DCT	232
Tabla 168: Ataque escalado de la imagen marcada algoritmo basado en secuencias caóticas y DCT	233
Tabla 169: Ataque rotación de la imagen marcada algoritmo basado en secuencias caóticas y DCT	234
Tabla 170: Tiempos de ejecución del algoritmo basado en secuencias caóticas y DCT	235
Tabla 171: Tiempos de ejecución del algoritmo basado en secuencias caóticas y DCT (sin permutación).....	235
Tabla 172: Inserción y extracción por el algoritmo PCA, alpha = 1	237
Tabla 173: Inserción y extracción por el algoritmo PCA, alpha = 4	238
Tabla 174: Inserción y extracción por el algoritmo PCA, alpha = 8.5	239
Tabla 175: Ataque compresión JPEG 60% algoritmo PCA.....	240

Tabla 176: Ataque inserción ruido gaussiano algoritmo PCA	241
Tabla 177: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo PCA	242
Tabla 178: Ataque recortado de la imagen marcada algoritmo PCA	243
Tabla 179: Ataque escalado de la imagen marcada algoritmo PCA	244
Tabla 180: Ataque rotación de la imagen marcada algoritmo PCA.....	245
Tabla 181: Tiempos de ejecución del algoritmo PCA	246
Tabla 182: Inserción y extracción por el algoritmo PCA para construir la imagen de referencia, $s= 1, t = 3$	248
Tabla 183: Inserción y extracción por el algoritmo PCA para construir la imagen de referencia, $s = 6, t = 8$	249
Tabla 184: Inserción y extracción por el algoritmo PCA para construir la imagen de referencia, $s = 20, t = 22$	250
Tabla 185: Ataque compresión JPEG 60% algoritmo PCA para construir la imagen de referencia	251
Tabla 186: Ataque inserción ruido gaussiano algoritmo PCA para construir la imagen de referencia	252
Tabla 187: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo PCA para construir la imagen de referencia.....	253
Tabla 188: Ataque recortado de la imagen marcada algoritmo PCA para construir la imagen de referencia	254
Tabla 189: Ataque escalado de la imagen marcada algoritmo PCA para construir la imagen de referencia	255
Tabla 190: Ataque rotación de la imagen marcada algoritmo PCA para construir la imagen de referencia	256
Tabla 191: Tiempos de ejecución del algoritmo PCA para construir la imagen de referencia	257
Tabla 192: Comparativa de los tiempos de ejecución	259
Tabla 193: Comparativa de los tipos de algoritmos	262
Tabla 194: Comparativa de la resistencia a los ataques	264

CAPÍTULO 1: INTRODUCCIÓN

La criptografía y la esteganografía son técnicas que han sido usadas a lo largo de toda la historia con el propósito de compartir un secreto para comunicarse durante tiempos de guerra y paz. Algunos de los primeros métodos para esconder información fueron, por ejemplo, escribir en tablas que posteriormente serían cubiertas de cera, escribir con tinta invisible o rasurar la cabeza de un mensajero y tatuar un mensaje en su cabellera, una vez le creciera el pelo el mensajero podía llevar el mensaje sin que nadie pudiese apreciarlo. En la Segunda Guerra Mundial se compartió información secreta camuflándola en un inocente mensaje de audio, ya fuese una canción o una grabación de una retransmisión.

Los alemanes inventaron la tecnología de los micropuntos para ocultar la comunicación en 1941. Utilizando micropuntos, los mensajes ni eran ocultados ni cifrados, pero eran invisibles a simple vista ya que su tamaño era muy pequeño. Los avances en criptografía continúan hoy día, por ejemplo se está investigando en la ocultación de un mensaje en una cadena de ADN por el uso de la técnica de esteganografía genómica [3].

Las técnicas de marcas de agua (watermarking) están diseñadas para proteger los derechos de autor (copyright) de ficheros digitales: imágenes, videos, música, etc. Este proyecto se dedicará concretamente al estudio de las técnicas de marcado de imágenes digitales. De este modo, una imagen marcada incorpora la identidad de su autor y la copia de dicha imagen implica la copia de las marcas que identifican a su autor. Existen distintas propiedades que deben cumplir dichos esquemas de protección, como la imperceptibilidad, la capacidad, o la robustez. De entre ellas, la más difícil de obtener es la robustez, que a su vez es la más importante, dado que de ella depende la utilidad de las técnicas en cuestión. En el presente trabajo nos hemos centrado principalmente en las propiedades de robustez e imperceptibilidad de las marcas de agua, donde *robustez* es la resistencia a distorsiones introducidas ya sea de

CAPÍTULO 1: INTRODUCCIÓN

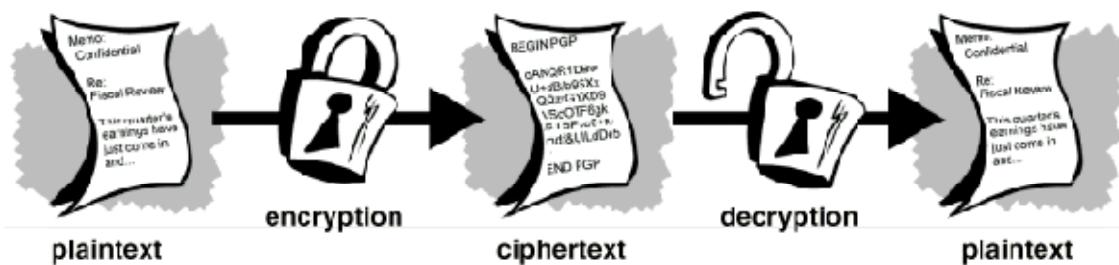
forma no intencionada o mediante un ataque intencionado cuyo objetivo es eliminar la marca de agua, e *imperceptibilidad* que se refiere a si la marca introducida en la imagen original es distingible o no a simple vista. Para evaluar la robustez del método se valorará la recuperación de la marca de agua [2].

1.1. Criptografía, esteganografía y marcas de agua

1.1.1. Criptografía

La criptografía, como el estudio del secreto (crypto) escrito (graphy), puede ser definida como la ciencia que permite cifrar y descifrar información. Por ejemplo, dos personas, que son normalmente conocidas como Alice y Bob, podrán compartir información secreta el uno con el otro. Un intruso, normalmente conocido como Eve, que interceptase dicha información no sería capaz de descifrarla ya que ésta le resultaría ininteligible. La criptografía actual permite a Bob controlar que el mensaje enviado por Alice no esté modificado por Eve (integridad del mensaje) y que el mensaje recibido fue enviado por Alice en realidad (autentificación del mensaje).

Existen algunas palabras específicas que se utilizan cuando se está hablando de criptografía y que se explicarán a continuación. Un mensaje sin cifrar es conocido como texto plano (plaintext). El mensaje resultante de aplicar una técnica criptográfica, dará lugar a un mensaje ininteligible conocido como texto cifrado (ciphertext). El proceso de obtener el texto original a partir del texto cifrado se conoce como descifrado [3].



1.1.2. Esteganografía

Mientras que la criptografía está protegiendo el contenido de los mensajes dando lugar a mensajes ininteligibles, la esteganografía está disimulando su existencia. La esteganografía viene de una palabra griega que significa cubrir lo escrito (estego = cubrir + grafía = escrito). Se trata de coger un medio portador inocuo, como puede ser

CAPÍTULO 1: INTRODUCCIÓN

una imagen digital o un fichero de audio y modificarlo convenientemente para ocultar información secreta.

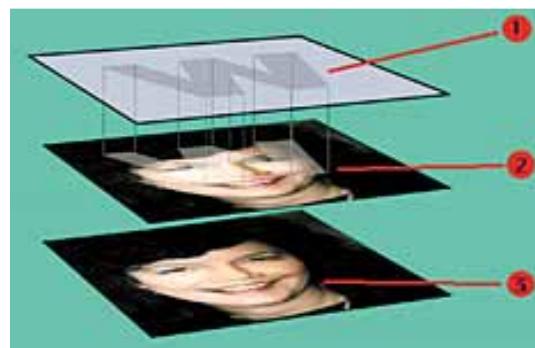
Por ejemplo, Alice podría elegir una imagen familiar suya como medio portador, modificar los pixeles de dicha imagen de manera que no se alterase en exceso pero que se logre ocultar la misma información secreta que se desea enviar a Bob. Eve puede interceptar dicha imagen pero probablemente no sospeche en absoluto de una imagen familiar de Alice. Solo Bob, al recibir la imagen modificada será capaz de recuperar a partir de ella la información secreta que Alice ocultó.

1.1.3. Marcas de agua digitales

Las marcas de agua digitales pretenden proteger los derechos de autor, estas pueden ser visibles o invisibles.



Marca de agua visible



Marca de agua invisible

Tabla 1: Ejemplos de marcas de agua

El método de trabajo de las técnicas de marcado de agua es similar a los métodos esteganográficos. Una técnica de marcado de agua consta de dos partes, la primera parte sería un algoritmo que permita ocultar la marca digital en el fichero cuyos derechos de autor se quiere proteger y una segunda parte en la que se hace uso de un algoritmo de extracción de la marca. También se puede utilizar una clave secreta para reforzar la seguridad. Esta serviría de prevención para evitar que partes no autorizadas, manipulen y recuperasen la marca de agua.

Para el proceso de ocultación, las entradas son la marca de agua, el objeto de cobertura y una clave secreta o pública. La marca de agua usada puede ser texto, una imagen digital o un fichero de audio. En los tres casos la información de la marca de agua se convierte en una secuencia de bits de ceros y unos. El objeto de cobertura cuyos derechos de autor se quieren proteger, será convenientemente modificado para incrustar en el mismo la información relativa a la marca de agua.

CAPÍTULO 1: INTRODUCCIÓN

Para el proceso de extracción de la marca será necesario tener el fichero digital previamente marcado y aplicar el algoritmo correspondiente que nos permita extraer de dicho fichero la marca de agua que se ocultó en él. De esta forma se protegerán los derechos de autor.

1.1.4. Aplicaciones de las marcas de agua digitales

- **Identificación de propiedad:** Las empresas identifican sus productos siempre para defender su copyright. Por ejemplo, la marca de identificación de una compañía de audio en la carátula del CD o la marca del papel manufacturado en una esquina. Estos tipos de marcas pueden ser fácilmente eliminadas cortando la imagen o desgarrando la parte que lo identifica. Las marcas de agua digitales ayudan a superar este problema, ocultándola en forma de bits y formando una parte integral del contenido.
- **Protección de copia:** Uno de los principales objetivos de las marcas de agua digitales consiste en dificultar las copias ilegales o no autorizadas de un determinado producto. Cuando los dispositivos de copia lean los datos, el algoritmo de reconocimiento de marcas de agua debería detectarla y parar la grabación. Esto necesitaría que todos los dispositivos de copia tuvieran un algoritmo de marcas de agua para identificarlas y actuar en consecuencia.
- **Aplicaciones médicas:** El nombre de los pacientes puede ser impreso en los informes de rayos X y los escáneres MRI usando técnicas de marcas de agua visibles. Los informes médicos juegan un importante rol en el tratamiento ofrecido al paciente. Esta información sobre la historia médica de un paciente debe de ser secreta, por lo tanto, es interesante disponer de técnicas que permitan ocultar la historia clínica de un paciente, por ejemplo, en una radiografía de dicho paciente [3].

1.1.5. Requerimientos de una buena marca de agua digital

- **Robustez:** Robustez significa que el esquema de marcado empleado debería ser capaz de preservar la marca de agua bajo varios ataques. Si el fichero a proteger es una imagen digital, el ataque podría ser una rotación, una traslación, un recortado, un escalado o bien, someter a la imagen a varios tipos de filtros. Este tipo de ataques no debería afectar a la recuperación de la marca de agua.

- **Calidad de la imagen:** El marcado debería realizarse de manera que no afecte a la calidad del fichero digital que está siendo marcado. Por ejemplo, si se está marcando una imagen digital, los cambios en dicha imagen no deberían ser detectados por el ojo humano.
- **Capacidad de carga de la imagen:** Es muy importante encontrar la máxima cantidad de información que pueda ser guardada u ocultada en una imagen. Dependiendo de la arquitectura del ordenador los tipos de datos pueden variar en tamaño. Esto afecta directamente a la robustez y al impacto perceptual. Si se insertan muchos datos en una imagen (muchos más que la capacidad de carga) puede ser perjudicial para la calidad de las imágenes ya que se reduce drásticamente.
- **Fiabilidad de la marca de agua:** Siempre existe la posibilidad de que un usuario conozca el algoritmo exacto para la detección y renderización de la marca de agua. La única manera de asegurar la marca de agua es evitar que se conozca la clave secreta que se ha usado en el proceso de marcado. Si un usuario malicioso conociese el algoritmo exacto de ocultación, debería ser prácticamente imposible que encontrara la clave exacta que se empleó durante la ocultación. Esto es muy importante para la fiabilidad o fortaleza de la marca de agua.

1.2. Definición del problema real

El problema real que se desea resolver con el presente proyecto es conocer las características, funcionamiento y calidad de los algoritmos que aplican técnicas de marcado en imágenes digitales.

A partir del estudio e implementación de estas técnicas de marcado digital, se quiere determinar las características, tales como la robustez o vulnerabilidad, para finalmente realizar un estudio comparativo de los mismos, analizando sus ventajas e inconvenientes y obteniendo una serie de conclusiones para cada algoritmo en particular.

1.3. Definición del problema técnico

1.3.1. Funcionamiento

CAPÍTULO 1: INTRODUCCIÓN

El objetivo de este proyecto será el de poder comprobar las características de cada uno de los algoritmos de marcado implementados. Para ello se tomarán imágenes guardadas en el disco, se mostrarán por pantalla, y se posibilitará la aplicación de algoritmos de marcado para insertar marcas de agua en ellas, o bien para tratar de extraer las marcas de agua de las imágenes marcadas. Así mismo, se realizarán pruebas de eficiencia para comprobar la calidad y robustez que tiene cada uno de los algoritmos analizados.

1.3.2. Entorno

El proyecto se desarrollará utilizando el programa de desarrollo matemático Matlab, el cual tiene su propia sintaxis como lenguaje de programación. También se hará uso de sus múltiples funciones, como de sus toolbox, que son paquetes con funciones específicas que hacen la aplicación de los algoritmos más sencillos. Para mostrar algún ejemplo de funcionamiento, por ejemplo ocultar y extraer una marca de agua, se hará uso de la utilidad de Matlab para realizar una interfaz gráfica que es la creación de una GUI.

En cuanto a las personas a las que está destinado el proyecto, hay que decir que al tratarse de un proyecto de fin de carrera, no tiene interés comercial, por lo que sólo les servirá a las personas a las que les resulte útil para su propia investigación o para una posterior mejora.

1.3.3. Vida esperada

Se trata de un proyecto de fin de carrera, y por lo tanto no está pensado para ser una aplicación comercial, por lo que no tiene sentido hablar de vida esperada. Lo que sí puede ser objeto de renovación o actualización serán los propios algoritmos utilizados. En tal caso, podrán ser éstos sustituidos por otros algoritmos más actuales, y con características que mejoren las técnicas actualmente publicadas.

1.3.4. Ciclo de mantenimiento

Ya que el proyecto está destinado mayoritariamente a la investigación y estudio de algoritmos de diferentes autores, no se indicarán las posibles fases de mantenimiento que deberían ser realizadas por personas ajena a este proyecto.

1.3.5. Competencia

Este proyecto no entrará en mercado, ya que no se trata de una aplicación comercial y, por lo tanto, no es necesario considerar este aspecto.

Sin embargo, los algoritmos que se llevarán a estudio, podrían llegar a ser útiles para que alguna empresa o grupo independiente hiciese uso comercial y obtener beneficio de ellos.

1.3.6. Aspecto externo

Como se ha comentado anteriormente se hará uso de la utilidad de Matlab para la creación de una interfaz gráfica a partir de una GUI. Como el proyecto no trata sobre una aplicación como tal, sino que es un estudio de investigación, la interfaz gráfica que se realizará será lo más sencilla posible para mostrar simplemente como se realizaría una inserción y extracción de una marca de agua.

1.3.7. Estandarización

Este aspecto se considerará desde el principio del proyecto, ya que se procurará que exista una utilización de herramientas y procedimientos lo más estándar posible, ya sea en la metodología de la programación, como en el desarrollo de la documentación que acompañará al proyecto.

1.3.8. Calidad y fiabilidad

Se intentará tomar todas las medidas necesarias para realizar los algoritmos lo más óptimos posibles siempre adecuándose al máximo, para minimizar así la ocurrencia de fallos, ya sea con la definición del algoritmo según el autor o con alguna modificación que la mejore.

1.3.9. Programación de tareas

CAPÍTULO 1: INTRODUCCIÓN

Las tareas que se pretenden realizar son las siguientes:

- 1.- Revisión bibliográfica sobre el marcado digital de imágenes en artículos de investigación.
- 2.- Estudio de los algoritmos a utilizar para cada técnica de marcado, y su implementación en Matlab.
- 3.- Obtención de los resultados de las ejecuciones de los algoritmos considerando las distintas técnicas de marcado.
- 4.- Estudio comparativo y análisis de los resultados obtenidos en la fase anterior.
- 5.- Extracción de conclusiones, analizando la robustez y vulnerabilidad de cada algoritmo.
- 6.- Redacción de la documentación del proyecto.

1.3.10. Pruebas

Las pruebas que se realizarán para este proyecto serán las de comprobación del correcto funcionamiento de cada uno de los algoritmos estudiados.

Para cada prueba se realizarán los siguientes pasos, primero se diseñará la prueba correspondiente al algoritmo, en dicho diseño se tendrá en cuenta los diferentes parámetros y opciones que el algoritmo contenga, seguidamente se observará la respuesta que ofrece y por último se solucionará el problema en el caso que exista.

Todo esto se podrá observar de una manera más detallada en el apartado correspondiente a las pruebas de este proyecto.

1.3.11. Seguridad

En cuanto a lo que a la seguridad respecta, al ser un proyecto fin de carrera, la documentación, códigos y aplicaciones que se realicen quedarán de cara al uso público con lo que cualquier persona podrá tener acceso. Esto no quiere decir que se pueda hacer uso comercial del mismo, ya que el proyecto pertenecerá a la Universidad de

CAPÍTULO 1: INTRODUCCIÓN

Córdoba y al propio autor, sirviendo de ayuda e información solamente a futuros estudios de investigación en el tema.

CAPÍTULO 1: INTRODUCCIÓN

CAPÍTULO 2: OBJETIVOS

2.1. Objetivos del proyecto

El objetivo principal de este proyecto es el estudio, análisis e implementación de diferentes técnicas de inserción y extracción de marcas de agua propuestas en artículos de investigación recientemente publicados. Entre estas técnicas se distinguirán tres tipos:

- Las del dominio espacial.
- Las del dominio de las frecuencias.
- Otros dominios.

Se utilizará cada una de estas técnicas para insertar una marca de agua en un medio portador (imágenes en formato .bmp), así como para recuperar dicha marca de agua.

Lo que se pretende es poder realizar un estudio de varios algoritmos de cada técnica, para poder determinar la robustez de cada uno en comparación con los demás. De esta forma podremos determinar características tales como la complejidad, la eficiencia o la vulnerabilidad de cada algoritmo.

Para cada técnica, se presentan distintos algoritmos como veremos a

CAPÍTULO 2: OBJETIVOS

continuación:

- Dominio Espacial:
 - Método Basado en Correlación [2].
 - Algoritmo de Cox [4].
 - Método del Bit Menos Significativo (LSB) [1].
- Dominio de las Frecuencias:
 - Transformada Discreta del Coseno (DCT) [1].
 - Acceso Múltiple por División de Código (CDMA) [3, 6, 11].
 - Transformada Discreta Wavelet (DWT) [1, 7, 14].
- Otros dominios:
 - Descomposición en Valores Singulares (SVD) [8, 9, 10, 12, 13, 16, 17, 18, 19, 20, 21, 22].
 - Métodos basados en Secuencias Caóticas [15, 23].
 - Métodos basados en Análisis de Componentes Principales (PCA) [5, 24, 25, 26, 27].

CAPÍTULO 3: ANTECEDENTES

3.1. Introducción

El uso de marcas de agua como tales no es una ciencia nueva, pueden rastrearse antecedentes incluso entre los antiguos griegos. En publicaciones de Cox [4] por ejemplo, se recogen aplicaciones de los últimos 50 años, aunque la mayor parte de las aplicaciones digitales son posteriores a 1994. La abrumadora mayoría de los trabajos se centran en imágenes, audio, video y después, en mucha menor medida, se abordan los casos de los textos, bases de datos, software, etc.

Muchas empresas utilizan marcas de agua visibles con el logotipo de la empresa para sus documentos oficiales. También se puede ver cómo grandes organismos han hecho uso de las marcas de agua para la protección de la autenticidad y verificación de documentos oficiales, tales como el DNI electrónico o los billetes de las antiguas pesetas o de los actuales euros, e incluso en determinadas marcas de folios en blanco.

3.2. Descripción de los algoritmos

3.2.1. Algoritmos de técnicas basadas en correlación

Introducción

Una forma sencilla de insertar una marca de agua en una imagen en el dominio espacial es sumarle un patrón de ruido pseudo-aleatorio a los valores de intensidad de sus píxeles para luego aprovecharse de las propiedades de la correlación. Dicho patrón

CAPÍTULO 3: ANTECEDENTES

está compuesto por los números enteros $\{-1, 0, 1\}$ y se llama pseudo-aleatorio porque no es aleatorio totalmente, sino que se genera a partir de una clave, por lo tanto, a igual clave obtenemos igual patrón.

Después, para introducir la marca en la imagen tenemos:

$$I_w(x,y) = I(x,y) + k * W(x,y)$$

Donde $I_w(x,y)$ es la imagen con la marca de agua, $I(x,y)$ es la imagen original, $W(x,y)$ es una máscara de patrones del mismo tamaño que la imagen y k es el factor de ganancia.

Para recuperar la marca de agua, el mismo algoritmo generador de ruido pseudo-aleatorio es inicializado con la misma clave, y después se realiza la correlación entre el patrón de ruido y la imagen marcada. Si la correlación excede un cierto umbral, la marca de agua es detectada [2].

Estructura de la Marca de Agua

La marca de agua será una máscara de patrones del mismo tamaño que la imagen.

Proceso de ocultación de la marca

Para el algoritmo de ocultación de la marca se dividirá la máscara $W(x,y)$ en tantos bloques como información en bits se quiere enviar. La longitud máxima de la marca a insertar será de: $(W_w * H_w)/S^2$, donde W_w es el ancho de la marca de agua, H_w es el alto de la marca de agua y S es el tamaño del bloque que se ha elegido. Por ejemplo, para una imagen de tamaño 512x512 y si se cogen bloques de tamaño 16, se tendría que la longitud máxima de la marca a insertar sería $(512*512)/16^2 = 1024$.

Se recorrerá cada posición del mensaje a incrustar, en donde, si el valor de la celda es cero se inserta el patrón previamente creado, es decir, la secuencia pseudo-aleatoria, en el bloque de la marca $W(x,y)$ y en caso de ser 1 se rellena el bloque con ceros.

Finalmente, una vez obtenida la marca $W(x,y)$, se realizará la operación mencionada anteriormente $I_w(x,y) = I(x,y) + k * W(x,y)$ para insertar la marca.

Proceso de extracción de la marca

Para recuperar la marca insertada se realizará la correlación de cada bloque de la imagen que posea la marca de agua con el patrón pseudo-aleatorio. Si dicha operación supera un cierto umbral (en este caso configurado manualmente), se reconstruye el mensaje con un “cero”, en caso contrario con “uno”.

Aunque la técnica anterior no necesitaría la imagen original para la extracción de la marca, existe la posibilidad de tener la imagen original y a partir de ella extraer la marca de la imagen marcada de la siguiente manera: $W(x,y) = (I_W(x,y) - I(x,y)) / k$.

3.2.2. Algoritmo de Cox

Introducción

Aquí se ilustra qué marca de agua no debería ser puesta en regiones perceptualmente insignificantes de una imagen (o su espectro) a no ser que algunas señales comunes o procesos geométricos afectasen estas componentes. Por ejemplo, una marca de agua oculta en el espectro de altas frecuencias de una imagen puede ser fácilmente eliminada con una pequeña degradación de la imagen por cualquier proceso que directa o indirectamente realice un filtrado de paso bajo. El problema es cómo insertar una marca de agua en las regiones más significantes perceptualmente del espectro de manera que se preserve la fidelidad. Sin embargo, muchos pequeños cambios podrían hacer a las imágenes muy susceptibles al ruido.

Para solventar este problema, el dominio de las frecuencias de la imagen ofrece esta vista como un canal de comunicaciones, y correspondientemente, la marca de agua es vista como una señal que es transmitida por este medio. Los ataques y las señales distorsionadas no intencionadas son tratados como ruido que la señal inmersa debe tener como algo inmune. Aunque se use esta metodología para ocultar marcas de agua en datos, del mismo modo podría ser aplicada para enviar cualquier tipo de mensaje por medio de datos [4].

Estructura de la Marca de Agua

En su más básica implementación, una marca de agua consiste en una secuencia de números reales $X = x_1, \dots, x_n$. En la práctica, se crea una marca de agua donde cada valor x_i es elegido independientemente de acuerdo a $N(0,1)$ (donde $N(\mu;$

CAPÍTULO 3: ANTECEDENTES

σ^2) denota una distribución normal con media μ y varianza σ^2). Se asume que los números están representados por una razonable pero finita precisión y se ignoran los posibles errores de redondeo insignificantes que pudiesen existir. Después se introducirá la notación para describir la inserción y extracción de la marca de agua y se describirá cómo dos marcas de agua (la original y la recuperada, posiblemente corrupta) pueden ser comparadas. Este procedimiento explora el hecho de que cada componente de la marca de agua sea elegida mediante una distribución normal. Sería posible utilizar otro tipo de distribuciones, incluso elegir un x_i uniformemente de $\{1,-1\}$, $\{0,1\}$ o $[0,1]$. Sin embargo, como se dice, tales distribuciones dejan una particular vulnerabilidad para ataques usando documentos múltiplemente marcados [4].

Proceso de ocultación de la marca

Se extrae de cada documento D una secuencia de valores $V = v_1, \dots, v_n$, en los cuales se inserta una marca de agua $X = x_1, \dots, x_n$ para obtener una secuencia ajustada de valores $V' = v'_1, \dots, v'_n$. V' es insertada seguidamente en el documento en lugar de V para obtener un documento marcado D' . Uno o más atacantes podrían después alterar D' , produciendo un nuevo documento D^* . Dado D y D^* , se extrae una posible marca de agua corrupta X^* y es comparada con X por significancia estadística. Se extrae X^* sacando primeramente un conjunto de valores $V^* = v^*_1, \dots, v^*_n$ de D^* (usando información sobre D) y después generando X^* de V^* y V .

Cuando se inserta X en V para obtener V' se especifica un parámetro de escalado α que determina la amplitud la cual X modifica V . Tres fórmulas para calcular V' son:

$$v'_i = v_i + \alpha x_i \quad (1)$$

$$v'_i = v_i(1 + \alpha x_i) \quad (2)$$

$$v'_i = v_i e^{\alpha x_i} \quad (3)$$

La ecuación 1 es siempre invertible, y las ecuaciones 2 y 3 son invertibles si $v_i \neq 0$. Dado V^* se puede, por lo tanto, calcular la función inversa para derivar X^* de V^* y V . La ecuación 1 podría no ser apropiada cuando los valores de v_i varían ampliamente. Si $v_i = 10^6$ entonces añadiendo 100 podría ser insuficiente para establecer una marca, pero si $v_i = 10$, añadiendo 100 distorsionaría este valor inaceptablemente. La inserción basada en las ecuaciones 2 o 3 son más robustas en diferentes escalas. Se nota que las ecuaciones 2 y 3 dan similares resultados cuando αx_i es pequeño. También, cuando v_i es positivo, la ecuación 3 es equivalente a $\lg(v'_i) = \lg(v_i) + \alpha x_i$, y podría ser vista como

CAPÍTULO 3: ANTECEDENTES

una aplicación de la ecuación 1 para el caso donde se usan los logaritmos de los valores originales. Para este método se ha utilizado la ecuación 2 [4].

Proceso de extracción de la marca

Como en el proceso de ocultación, la marca de agua es extraída mediante la misma ecuación, salvo que donde antes se encontraba la imagen original, ahora habrá que utilizar la imagen marcada, despejando así los valores de la marca de agua.

Para evaluar la calidad de la marca extraída se le realizará la similaridad con la marca original y algunas marcas creadas aleatoriamente, con ello se podrá saber cuánta calidad tiene después de un posible ataque o simplemente después de la extracción.

3.2.3. Algoritmos basados en la DCT

Introducción

El clásico y más popular dominio de procesamiento de imágenes es la Transformada Discreta del Coseno, o DCT. La DCT permite a una imagen descomponerse en diferentes bandas de frecuencia, haciéndolo mucho más fácil que ocultar información marcada en bandas de frecuencias medias de una imagen. Las bandas de frecuencias medias son elegidas de manera que evitan las partes visuales más importantes de la imagen (bajas frecuencias) sin sobreexponerse a eliminar la compresión y los ataques de ruido (altas frecuencias) [1].

Primer Método: DCT

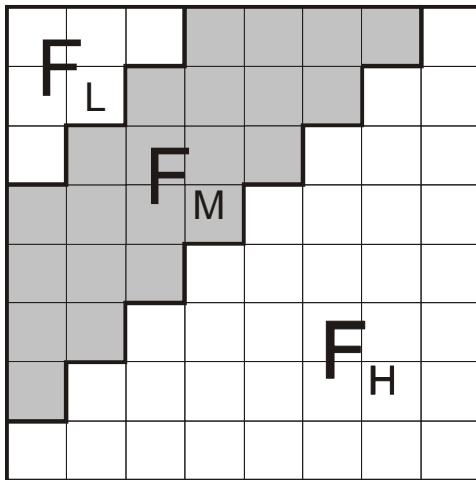
Estructura de la Marca de Agua

La marca de agua será una imagen binaria de tamaño MxN.

Proceso de ocultación de la marca

CAPÍTULO 3: ANTECEDENTES

Esta técnica utiliza la comparación de los coeficientes de bandas medias DCT para codificar un solo bit en un bloque DCT. Para comenzar, definimos las frecuencias de banda media (F_M) de un bloque DCT 8x8 como se muestra en la figura.



Definición de regiones DCT

F_L se usa para denotar los componentes de bajas frecuencias del bloque, mientras que F_H se usa para denotar los componentes de altas frecuencias. F_M es elegido tanto para las regiones de ocultación como para proporcionar resistencia adicional a las técnicas de pérdida de compresión, mientras se evita modificaciones significativas a la imagen de cobertura.

Las siguientes dos localizaciones $B_i(u_1, v_1)$ y $B_i(u_2, v_2)$ son elegidas de la región F_M para comparación. En vez de elegir las localizaciones de manera arbitraria, muy robustas para la compresión, son escogidas en base a la elección de los coeficientes en la tabla de cuantización JPEG recomendada.

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

Valores de cuantización usados en el esquema de compresión JPEG

CAPÍTULO 3: ANTECEDENTES

Basándose en la tabla, podemos observar que los coeficientes (4,1) y (3,2) o (1,2) y (3,0) serían candidatos apropiados para la comparación, ya que sus valores de cuantización son iguales. El bloque DCT codificará un “1” si $B_i(u_1, v_1) > B_i(u_2, v_2)$; en otro caso codificará un “0”. Los coeficientes son intercambiados después si el tamaño relativo de cada coeficiente no concuerda con el bit que está para codificarse.

El intercambio de tales coeficientes no debería alterar la imagen marcada ya que, por lo general, los coeficientes DCT de medias frecuencias tienen magnitudes similares. La robustez de la marca de agua puede ser mejorada introduciendo una constante “fuerte” a la marca de agua, tal que $B_i(u_1, v_1) - B_i(u_2, v_2) > k$. Los coeficientes que permiten este criterio son modificados aunque el uso de ruido aleatorio satisface la relación. Incrementando k se reduce el riesgo de detección de errores pero ocurre que existe degradación adicional en la imagen [1].

Proceso de extracción de la marca

En este caso, la extracción de la marca de agua se realiza a partir de la imagen marcada y siguiendo el procedimiento inverso a como se marcó la imagen original. Es decir, se comparan los coeficientes y dependiendo como se haya codificado, se guardará un 1 o un 0 en el bit que toque de la marca de agua.

Segundo Método: DCT y correlación

Estructura de la Marca de Agua

La marca de agua será una imagen binaria de tamaño MxN.

Proceso de ocultación de la marca

Otra posible técnica que mezcla la aplicación de la DCT y la utilización de la correlación para las marcas es, ocultar una secuencia W en las frecuencias medias de un bloque DCT. Podemos modular un bloque DCT dado x,y usando la ecuación que se muestra en la figura.

CAPÍTULO 3: ANTECEDENTES

$$I_{W_{x,y}}(u,v) = \begin{cases} I_{x,y}(u,v) + k * W_{x,y}(u,v), & u,v \in F_M \\ Ix,y(u,v), & u,v \notin F_M \end{cases}$$

Ocultación de marca de agua CDMA en frecuencias medias DCT

Para cada x,y del bloque 8×8 de la imagen, se calcula primero el bloque DCT. En ese bloque, los componentes de frecuencias medias F_M son añadidos a la pn secuencia W , multiplicada por un factor de ganancia k . Los coeficientes en las bajas y medias frecuencias son copiados en las partes de la imagen que no sufren transformaciones. Cada bloque es transformado inversamente para darnos la imagen marcada final I_W .

El procedimiento de marcado puede ser hecho algo más adaptativo alterando ligeramente el proceso de ocultación con el método mostrado más abajo.

$$I_{W_{x,y}}(u,v) = \begin{cases} I_{x,y}(u,v) * (1 + k * W_{x,y}(u,v)), & u,v \in F_M \\ Ix,y(u,v), & u,v \notin F_M \end{cases}$$

Carga de imagen DCT marca de agua CDMA

Esta leve modificación escala la fuerza del marcado basándose en el tamaño de los coeficientes particulares usados. Una k grande puede, por tanto, ser usada por coeficientes de magnitud mayor, en efecto, fortaleciendo la marca de agua en regiones que puede proporcionarlo; cediendo en las que no puede.

Proceso de extracción de la marca

Como detección, la imagen es dividida en bloques iguales de 8×8 , y se realiza la DCT. La misma pn secuencia es comparada para valores de medias frecuencias del bloque transformado. Si la correlación entre las secuencias excede un umbral T , un “1” es detectado para ese bloque; de otro modo, un “0” es detectado. De nuevo, k denota la fuerza del marcado, donde incrementando k se incrementa la robustez de la marca de agua a expensas de calidad.

3.2.4. Algoritmos basados en CDMA

Introducción

La técnica de Acceso Múltiple por División de Código o CDMA, es una técnica de amplio espectro, que extiende lo transmitido o el mensaje de banda limitada sobre una ancha banda de frecuencias, que es mucho más que la actual amplitud de banda

CAPÍTULO 3: ANTECEDENTES

mínima requerida y a esta señal de gran amplitud de banda se le llama señal de propagación. Un modo de ampliar la amplitud de banda es el uso de la modulación. Secuencias pseudo-aleatorias o secuencias PN pueden ser usadas como secuencias de difusión. Para aplicaciones de mercado, puede ser usado un generador de números pseudo-aleatorios para determinar los pixeles donde se ocultarán los datos de la marca de agua usando una “semilla” o “clave”.

Uno podría pensar que usar la técnica de difusión del espectro podría ser un aparente desperdicio del espectro. Pero se consiguen varias cosas:

- Inmunidad de ruido y distorsión multicamino.
- Puede ser usado para esconder y cifrar señales, esta propiedad ayuda en el principio de marcado usando CDMA.
- Varios usuarios pueden usar independientemente la misma amplitud de banda con una interferencia muy pequeña.
- Los bajos valores de la autocorrelación y el cruce de correlaciones de secuencias PN hacen de ello una herramienta útil para el marcado [3].

Esta técnica es muy utilizada actualmente; aunque basados en los mismos principios, otros autores han hecho distintas propuestas [6, 11].

Estructura de la Marca de Agua

La marca de agua será una imagen binaria de tamaño MxN.

Proceso de ocultación de la marca

Para ocultar la marca de agua en la imagen original lo que se hará es comprobar los bits de la marca original, en el caso de que el bit sea 0 se utilizará la siguiente ecuación:

$$I_w = I + k * pn_secuencia$$

donde I_w es la imagen marcada, I es la imagen original, k es el factor de ganancia y la pn secuencia es una secuencia generada pseudoaleatoriamente con una semilla que será una clave secreta.

Proceso de extracción de la marca

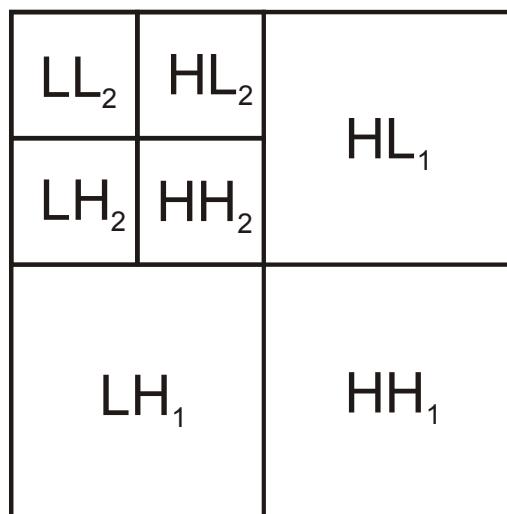
CAPÍTULO 3: ANTECEDENTES

La extracción de la marca de agua oculta en la imagen marcada se hará de manera que, primero se generarán secuencias pseudoaleatorias, se calculará el vector de correlaciones entre la imagen marcada y estas marcas pseudoaleatorias y se establecerá un umbral que será la media del vector de correlaciones. En este momento, se comprobará si el valor que toca del vector de correlaciones es mayor que el umbral definido, en el caso de que así sea se reconstruirá la marca poniendo en ese bit el valor de 0, siendo el resto de valores que no sean modificados puestos a 1 al principio de la comprobación.

3.2.5. Algoritmos basados en la DWT

Introducción

Otro posible dominio donde ocultar marcas de agua es el dominio wavelet. La Transformada Discreta Wavelet o DWT separa una imagen en una aproximación mínima de la imagen (LL) además de las componentes, horizontal (HL), vertical (LH) y diagonal (HH). El proceso puede ser repetido para obtener diferentes niveles de la descomposición wavelet, como en la figura, que se muestra hasta un segundo nivel.



Nivel 2 – Transformada Discreta Wavelet

Una de las muchas ventajas de la transformada wavelet es que se piensa que es el más preciso modelo del Sistema Visual Humano o HVS comparado con la Transformada Rápida de Fourier (FFT) o la DCT. Esto permite usar marcas de agua de alta energía en regiones donde el HVS se conoce que es menos sensible, tales como las bandas detalladas de alta resolución {LH, HL, HH}. Ocultando marcas de agua en estas regiones permite incrementar la robustez de la marca de agua, sin tener un gran impacto en la calidad de la imagen [1].

Primer Método: CDMA en el dominio wavelet

Estructura de la Marca de Agua

La marca de agua será una imagen binaria de dimensiones MxN.

Proceso de ocultación de la marca

Una de las técnicas más sencillas es usar una técnica de ocultación similar a la usada en la DCT, la ocultación de una secuencia CDMA en la banda detallada de acuerdo con la ecuación mostrada.

$$I_{W_{u,v}} = \begin{cases} W_i + \alpha |W_i| x_i, & u, v \in HL, LH \\ W_i & u, v \in LL, HH \end{cases}$$

donde w_i denota el coeficiente de la imagen transformada, x_i el bit de la marca de agua a ser ocultado, y α el factor de ganancia.

Además, como la ocultación usa los valores de la transformada, el proceso de ocultación debería ser bastante adaptativo; almacenando la mayoría de la marca de agua en los coeficientes.

Proceso de extracción de la marca

Para detectar la marca de agua se genera la misma secuencia pseudoaleatoria usada en la generación CDMA y se determina su correlación con las dos bandas transformadas. Si la correlación excede algún umbral, se detecta la marca de agua.

Esto puede ser fácilmente extendido para ocultar en una imagen múltiples marcas de agua. Como en el dominio espacial, se usa una semilla para cada PN secuencia, que son después añadidas a los coeficientes. Durante la detección, si la correlación excede el umbral T para una secuencia en particular, un “1” es recuperado; en otro caso es un “0”. El proceso de recuperación itera la PN secuencia hasta que todos los bits de la marca de agua hayan sido recuperados.

Segundo Método: DWT y la transformada de Haar

Estructura de la Marca de Agua

La marca de agua W es una secuencia de bits pseudoaleatoria generada por una semilla. Se define como sigue:

$$W = \{w(k) | 1 \leq k \leq n, w(k) \in \{1, -1\}\}$$

[7].

Proceso de ocultación de la marca

Una imagen con tamaño $M \times N$ pixeles es transformada en coeficientes wavelet por la transformada wavelet de un nivel y se extrae la subbanda LL. La subbanda extraída LL es descompuesta en cuatro subbandas y las tres subbandas de altas frecuencias (LH, HL, HH) se ponen a cero. Despues se realiza la transformada inversa wavelet de donde se obtiene la subbanda LL. La información idx de la localización de lo ocultado en el proceso se obtiene de la marca de agua ordenando $|LL-LL'|$. Finalmente, la información de la marca de agua es ocultada en la subbanda LL haciendo $LL = LL' \pm k * w(idx(i))$, donde k es un factor para controlar la intensidad de ocultado y w es una secuencia pseudoaleatoria binaria con tamaño de 100 bits generados por una semilla, w pertenece a $\{1, -1\}$.

La imagen original X es una imagen en niveles de gris con $M \times N$ pixeles. Se define como sigue:

$$X = \{x(i,j) | 0 \leq i \leq M-1, 0 \leq j \leq N-1, 0 \leq x(i,j) \leq 255\}$$

Primero, la imagen original es modificada en el dominio de la transformada. Despues, una marca de agua es ocultada en la imagen original de acuerdo a los valores diferentes entre la imagen original y su imagen de referencia.

Entrada: Una imagen original X y una marca de agua W .

Salida: Una imagen marcada X_w y una secuencia idx con las localizaciones.

Paso 1: Transformar la imagen original en coeficientes wavelet de un nivel de la transformada.

Paso 2: Poner las tres subbandas de altas frecuencias (LH, HL y HH) a cero.

Paso 3: Realizar la transformada inversa y obtener su imagen de referencia.

CAPÍTULO 3: ANTECEDENTES

Paso 4: Calcular las diferencias entre la imagen original X y su imagen de referencia X' . Después obtener la localización $idx(i,j)$ tal que $s < |x(i,j) - x'(i,j)| < t$, donde s y $t \in Z^+$.

Paso 5: De acuerdo con el resultado del paso 4, aleatoriamente se selecciona algunas localizaciones para ocultar. La marca de agua es ocultada como sigue:

$$\begin{aligned} x_w(idx(i,j)) \\ = \begin{cases} x'(idx(i,j)) + \alpha & \text{if } w(k) = 1 \text{ and } s < x(idx(i,j)) - x'(idx(i,j)) < t \\ x'(idx(i,j)) - \alpha & \text{if } w(k) = -1 \text{ and } s < x'(idx(i,j)) - x(idx(i,j)) < t \end{cases} \end{aligned}$$

donde $\alpha = \text{round}[(s+t)/2]$, que es el ancho visual para balancear el sacrificio entre la robustez y la imperceptibilidad.

Nótese que el proceso de ocultación no requiere repetición. Además, la secuencia idx de las localizaciones de ocultado debería mantenerse como una clave secreta para la extracción de las subsecuencias de la marca de agua.

Proceso de extracción de la marca

El proceso de extracción de la marca de agua no requiere la imagen original. Se utiliza la secuencia de localizaciones de ocultado para extraer la marca de agua.

Entrada: Una imagen marcada X_w y la secuencia idx de locaciones de ocultado.

Salida: Una marca de agua W' .

Paso 1: Transformar la imagen marcada en coeficientes de un nivel de la transformada wavelet.

Paso 2: Poner las tres subbandas de altas frecuencias (LH, HL y HH) a cero.

Paso 3: Realizar la transformada inversa wavelet y obtener su imagen de referencia X'_w .

Paso 4: De acuerdo a la secuencia de localizaciones de ocultado, obtener los bits ocultos de la marca de agua por la siguiente formula.

$$w'(k) = \begin{cases} 1 & \text{if } x_w(idx(i,j)) \geq x'_w(idx(i,j)) \\ -1 & \text{if } x_w(idx(i,j)) < x'_w(idx(i,j)) \end{cases}$$

Paso 5: Comparar la marca de agua extraída W' con la marca original W . La similaridad se mide por la siguiente fórmula:

$$Sim(W, W') = \frac{W \cdot W'}{\sqrt{W' \cdot W'}}$$

Tercer Método: DWT basado en la paridad

Estructura de la Marca de Agua

Se supone una imagen en escala de grises I de tamaño MxN para ser protegida por la inserción de una marca de agua binaria w, se puede suponer que w es de tamaño M/4 x N/4 (por ejemplo, el tamaño de la subbanda de baja frecuencia después de dos niveles de descomposición), si no, de una imagen binaria M/4 x N/4 se puede construir w de varias maneras, una opción sería la repetición periódica de w [14].

Proceso de ocultación de la marca

Primero se descompone la imagen original en dos niveles usando la transformada wavelet.

En el proceso de ocultación de la marca de agua, se usa una clave secreta K para generar una permutación aleatoria de vectores R_p y C_p de los vectores fila y columna, Row = [1, 2, ..., M/4] y Col = [1, 2, ..., N/4] de la marca de agua W, y se permutan las posiciones de los píxeles de la marca de agua W usando R_p y C_p como sigue:

$$w_p(R_p(i), C_p(j)) = w(i, j) \quad (1)$$

donde w_p representa la marca de agua permutada.

Para un ocultamiento robusto de la marca de agua, la marca de agua será ocultada en la subbanda de bajas frecuencias después de dos niveles de descomposición. Siendo C_{LL} los coeficientes de bajas frecuencias y C_{LL}^* los coeficientes correspondientes de la imagen marcada. El algoritmo de ocultación se puede representar como sigue:

$$\text{Obtener } d(i, j) = \left\lfloor \frac{C_{LL}(i, j)}{Q(i, j)} \right\rfloor,$$

$$\text{Si } w_p(i, j) \neq \text{mod}(d(i, j), 2),$$

$$C_{LL}^*(i, j) = (d(i, j) + 1) * Q(i, j);$$

CAPÍTULO 3: ANTECEDENTES

$$\text{Si no, } C_{LL}^*(i,j) = C_{LL}(i,j).$$

Donde Q es la matriz constante para controlar la fuerza de la marca ocultada, y $\lfloor \cdot \rfloor$ es la operación de obtener la parte entera de un número.

Entonces, después de hacer la inversa de la DWT, se adquirirá la imagen marcada. La decisión de la matriz Q es un problema clave para la robustez de la marca de agua. Cuanto mayor sea Q , la marca de agua será más robusta. Puede haber dos métodos para decidir los elementos de Q , un modo simple sería poner todos los elementos de Q con el mismo número constante $Const$, el otro sería un modo adaptativo, acordando los coeficientes correspondientes de la subbanda detallada. Es decir,

$$Q_1(i,j) = Const \quad i = 1, \dots, \frac{M}{4}, j = 1, 2, \dots, \frac{N}{4} \quad (2)$$

$$Q_2(i,j) = \alpha \log_2 \left(2 + \frac{|C_{HL}^2(i,j)| + |C_{LH}^2(i,j)|}{2} \right) \quad i = 1, \dots, \frac{M}{4}, j = 1, 2, \dots, \frac{N}{4} \quad (3)$$

donde C_{HL}^2 y C_{LH}^2 son los coeficientes detallados de las direcciones horizontal y vertical del segundo nivel de la descomposición wavelet respectivamente, y α es un parámetro para controlar la fuerza de la marca ocultada.

Para la fórmula, cuando la imagen tiene textura compleja en la posición (i, j) , el coeficiente $Q(i, j)$ es grande, se puede tener una marca ocultada bastante fuerte. Mientras que cuando $Q(i, j)$ es pequeño, la imagen es plana en (i, j) , se puede tener una marca suave, donde α es un parámetro para controlar la fuerza de la ocultación de la marca. Por tanto, se puede obtener robustez pero menor degradación de la imagen original.

Proceso de extracción de la marca

Para la extracción de la marca, se llevan a cabo las operaciones inversas. Primero, se transforman las imágenes marcadas por la DWT de dos niveles. Se calcula la matriz de cuantización por (2) o (3) para los coeficientes de la subbanda de bajas frecuencias C_{LL} . Para cada elemento de los coeficientes se hace lo siguiente:

$$\text{Obtener } d(i,j) = \left\lfloor \frac{C_{LL}(i,j)}{Q(i,j)} \right\rfloor,$$

$$w^*(i,j) = \text{mod}(d(i,j), 2);$$

Para detectar la marca de agua, se usa la similaridad entre dos marcas:

$$\text{sim}(X, X') = 1 - (\| X - X' \|)^2 / \text{card}(X') \quad (4)$$

donde X y X' es la marca original y la extraída respectivamente.

3.2.6. Algoritmo basado en SVD

Introducción

SVD es una técnica numérica usada para diagonalizar matrices en análisis numérico. Es un algoritmo desarrollado para una variedad de aplicaciones. Las principales propiedades de la SVD desde el punto de vista de las aplicaciones de procesamiento de imágenes son: 1) los valores singulares (SVs) de una imagen tienen muy buena estabilidad, es decir, cuando una pequeña perturbación es añadida a la imagen, sus SVs no cambian significativamente; y 2) los SVs representan propiedades intrínsecas algebraicas de la imagen.

Desde el punto de vista del álgebra lineal, se puede observar que una imagen discreta es un vector de entradas escalares no negativas, las cuales podrían ser guardadas como una matriz. Denotando una imagen como A . Se supone que A es una matriz cuadrada, donde se representa o el dominio de números reales o el dominio de números complejos. La SVD se define como:

$$A = USV^t \quad (1)$$

donde U y V son matrices unitarias y S es una matriz diagonal.

Para la elección de la marca de agua, se calcula la SVD de una imagen A de dimensiones $N \times N$ para obtener dos matrices ortogonales U y V y una matriz diagonal S . Aunque se supone que A será una matriz cuadrada (imagen) por conveniencia, otras imágenes no cuadradas podrán ser procesadas del mismo modo. Se mantiene que esta es una de las ventajas del método SVD sobre algún otro esquema de marcado que no puede tratar directamente matrices no cuadradas [8].

Primer Método: SVD

Estructura de la Marca de Agua

La marca de agua con dimensiones $M \times N$ es una secuencia pseudoaleatoria de números pertenecientes a una normal $N(0,1)$, con media 0 y varianza 1.

Proceso de ocultación de la marca

Se añade una marca de agua W (también representada como una matriz) en la matriz S y se realiza la SVD en la nueva matriz $S + \alpha W$ para obtener U_W , S_W , y V_W ($S + \alpha W = U_W S_W V_W^H$, el superíndice H denota la matriz de transposición de Hermite, pero cuando una matriz A es real, entonces A^H es simplemente A^T , es decir, la transpuesta de A), donde la constante positiva α es el factor de escala que controla la fuerza de la marca de agua para ser insertada. Después se obtiene la imagen marcada A_W por la multiplicación de las matrices U , S_W , y V^T . Es decir, si las matrices A y W representan la imagen original y la marca de agua respectivamente, se obtiene la imagen marcada A_W por los siguientes tres pasos:

$$A \Rightarrow USV^H \quad (2)$$

$$S + \alpha W \Rightarrow U_W S_W V_W^H \quad (3)$$

$$A_W \Leftarrow US_W V^H \quad (4)$$

Proceso de extracción de la marca

En la detección de la marca de agua, dadas las matrices U_W , S , V_W , y la posible imagen distorsionada A_W^* , uno puede extraer una posible marca de agua corrupta W^* si se hace la operación inversa de lo antedicho. Es decir:

$$A_W^* \Rightarrow U^* S_W^* V^{*H} \quad (5)$$

$$D^* \Leftarrow U_W S_W^* V_W^H \quad (6)$$

$$W^* \Leftarrow \frac{1}{\alpha}(D^* - S) \quad (7)$$

Mientras que algunos algoritmos de marcado requieren la imagen original para extraer la marca de agua, este método usa tres matrices para extraer la marca de agua y no necesita información adicional.

La similaridad entre W (la marca de agua original) y W^* (la marca de agua extraída) es medida por la definición de correlación. Por conveniencia se considera que W y W^* son vectores unidimensionales (1-D) y se calcula su coeficiente de correlación $c(W, W^*)$ de manera estándar. Para una marca de agua bidimensional (2-D) (tal como una imagen de un logo de una compañía), se puede simplemente tratar la marca de agua como un vector 1-D o calcular el coeficiente de correlación 2-D directamente.

Comentarios

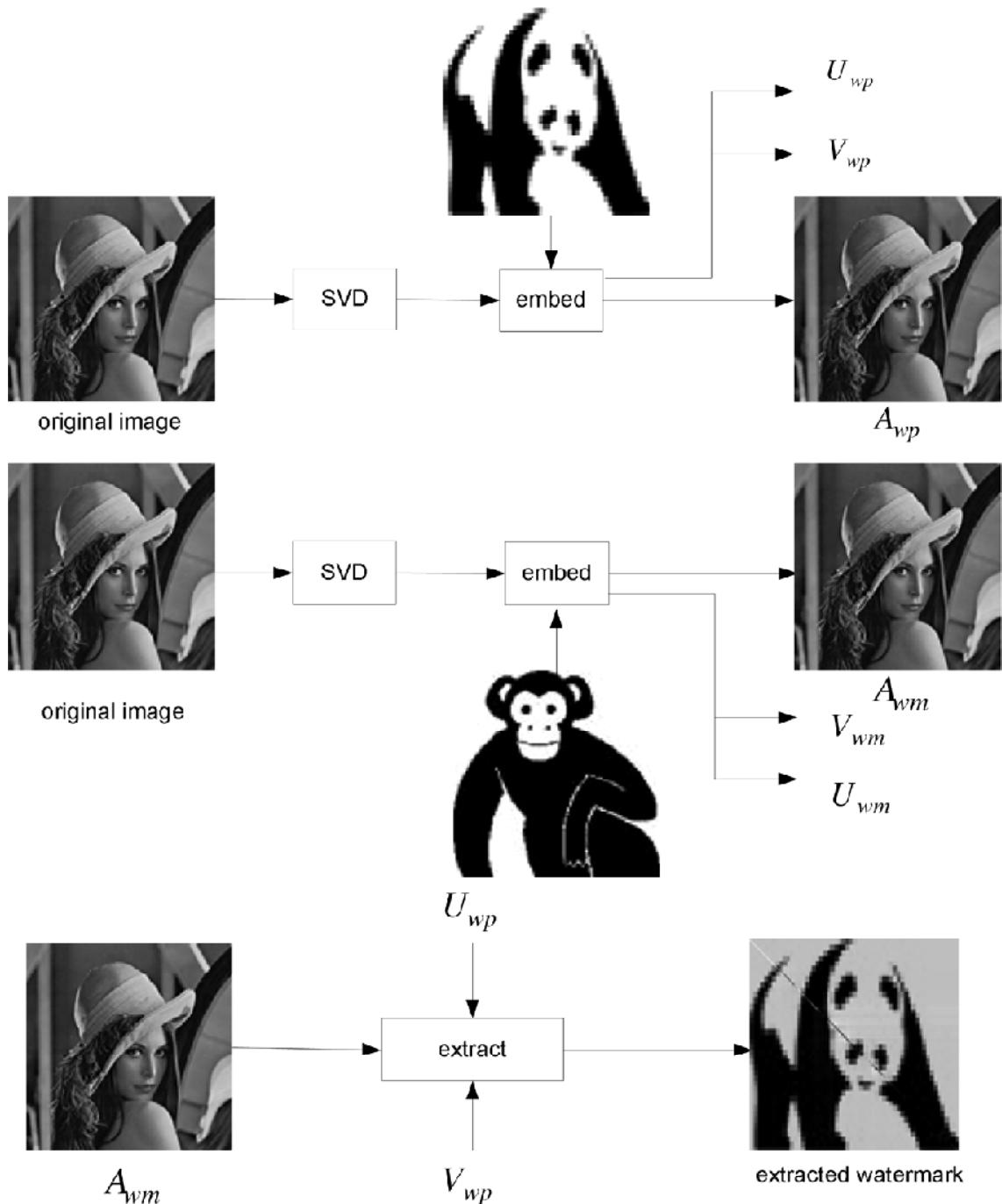
Este algoritmo ha tenido una difusión muy amplia en la literatura científica y ha sido muy referenciado, sin embargo, en otros artículos se demuestra una debilidad importante del método de marcado.

Primer comentario:

Los valores singulares de diferentes imágenes (pero con propiedades geométricas ligeramente similares) difieren relativamente poco. Por tanto, si se usa en la fórmula (6) una matriz S_W^* , obtenida por la SVD de una imagen completamente no conectada con la imagen marcada, se obtiene una buena estimación de W por (7). Esto significa que el detector devolverá una respuesta positiva (se detectará la marca de agua W) para imágenes arbitrarias, las cuales llegan a su entrada. Debería ser enfatizado que esto no es la desventaja del detector. Esto es un resultado de falsa concepción para insertar información de las marcas de agua en valores singulares de la imagen. Desde luego, el algoritmo de marcado con muy alta probabilidad de respuesta de falso positivo no puede ser usado para proteger la propiedad legítima incluso si es no invertible [9].

Segundo comentario:

Se discute que el método de marcado anterior es fundamentalmente defectuoso. Notar que en (3), S es una matriz diagonal, las matrices U_w and V_w representan los subespacios de una marca de agua W ligeramente modificada, que solo difiere de la marca de agua original en los valores diagonales. Como se sabe, el subespacio SVD puede preservar la mayor información de una imagen (como es usado en (4)). Por tanto, en el paso de detección (6), no importa qué valor de la matriz diagonal S^* toma, la matriz resultante D^* está definida en el mismo subespacio SVD por la marca de agua modificada diagonalmente. En otras palabras, (6) efectivamente “destruye” la información de la marca de agua en D^* sin importar que A_W^* y que W se utilicen. Notar que (7) cambia solo los valores diagonales de D^* . Por tanto, la marca extraída W^* tendrá alta correlación con la marca de agua W que con A_W^* [10].



Como última cosa que comentar, es decir que este método es bastante conocido y por ello hay otros autores que han querido proponer sus propios algoritmos, algunos tienen modificaciones y añaden la combinación de otro método, pero las bases siguen siendo las mismas [13, 20, 21].

Segundo Método: SVD con transformada de Arnold

CAPÍTULO 3: ANTECEDENTES

La transformada bidimensional de Arnold, popularmente llamada mapa de la cara del gato porque la transformación fue llevada a la práctica por primera vez con la imagen de un gato, fue introducida por V. I. Arnold cuando estudiaba problemas ergodinámicos de mecánica clásica.

Esta transformación lleva a cabo una permutación pseudoaleatoria de los pixeles de una imagen mediante un proceso iterativo. Se puede utilizar para cifrar una imagen por ejemplo.

Suponiendo que el tamaño de la imagen original es $M \times M$, (x, y) son las coordenadas del pixel de la imagen y (x', y') son las coordenadas después de la transformación. La transformada bidimensional de Arnold se define como:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} (\text{mod } M)$$

donde $x, y \in \{0, 1, \dots, M-1\}$.

En este caso se ha empleado la matriz $\{\{1, 1\}, \{1, 2\}\}$, pero se podrían emplear otras matrices. De hecho, se podría emplear cualquier matriz cuadrada A y de cualquier orden, la única restricción es que, si la imagen es $M \times M$ entonces la matriz debe tener inversa módulo M , lo cual ocurre siempre que el máximo común divisor del determinante de A y M valga 1, es decir, $\text{mcd}(|A|, M) = 1$.

Las posiciones de cada pixel en la imagen son transformadas usando la formula anterior. Después de recorrer todos los pixeles en la imagen, se consigue la imagen codificada. Además, uno puede transformar la imagen repetidamente para generar diferentes resultados hasta alcanzar los requerimientos. Debido a la periodicidad de la transformada de Arnold, la imagen puede ser recuperada después de cierto número de iteraciones.

Se asume que la imagen original A tiene un tamaño de $N \times N$ donde $N = 2^n$, la marca de agua W tiene un tamaño de $M \times M$ donde $M = 2^m$, y $n \geq m$ [12].

Estructura de la Marca de Agua

La marca de agua será una imagen en escala de grises de tamaño $M \times M$ e inferior al tamaño de la imagen original.

Proceso de ocultación de la marca

CAPÍTULO 3: ANTECEDENTES

(1) Procesamiento de la marca de agua

Para aumentar la seguridad y la robustez de la operación de recortado en el esquema de marcado propuesto, la marca de agua W es codificada espacialmente usando la transformada de Arnold antes de ocultarla en la imagen de cobertura. Se baraja cada posición de la marca de agua k veces respectivamente, donde el parámetro k puede ser después usado como clave secreta cuando se recupere la marca de agua.

(2) Bloque SVD de la imagen de cobertura

Se cogen bloques 8×8 de la imagen original y se le aplica la SVD a cada bloque, de ello se guarda el primer elemento de cada vector S que se calcula en cada iteración y se almacenan estos valores.

(3) Ocultación de la marca de agua

Se añade a cada valor del pixel de la marca de agua codificada W' el máximo SV de cada bloque correspondiente:

$$S + \alpha W'$$

donde S es una matriz $M \times M$ compuesta por todos los máximos SVs; α es el factor de escalado que controla la fuerza de la ocultación de la marca de agua; W' es la marca de agua codificada.

De acuerdo a las propiedades de luminancia y textura del HVS, se puede ocultar la marca de agua en la imagen de cobertura con diferentes valores para el factor de ganancia α para eludir el daño a la calidad visual. Por simplicidad, se selecciona una constante como factor de escalado.

(4) Por último, se realiza la imagen marcada A_w por inversión de los bloques SVD transformados:

$$U_{ij} S'_{ij} V^H_{ij} \rightarrow B'_{ij}$$

donde S'_{ij} es obtenido por modificación del máximo SV de S_{ij} con el correspondiente valor del pixel de la marca de agua codificada con el factor de escala α ; B'_{ij} es el bloque no coincidente al final de la imagen marcada, la cual constituye la imagen marcada A_w final.

Por supuesto, el periodo de la transformación de Arnold k (como clave secreta), el tamaño de bloque 8×8 , la matriz S que contiene todos los máximos SV obtenidos de los bloques SVD de la imagen de cobertura y el factor de escala α deben de ser guardados ya que son requeridos para la extracción de la marca de agua.

Proceso de extracción de la marca

En este esquema, la imagen original no es necesaria, pero los parámetros provistos en el proceso de ocultación mencionados anteriormente si son necesarios. La posible marca de agua corrupta W^* puede ser extraída como sigue:

(1) Primero, a la imagen marcada posiblemente atacada A_w^* se le calcula la SVD por bloques de tamaño 8x8.

$$B'_{ij}^* = U_{ij}^* S'_{ij}^* V'^H_{ij}.$$

(2) El máximo SV de cada bloque S'_{ij}^* es cogido para constituir una nueva matriz S'_{ij}^* . La marca de agua codificada posiblemente distorsionada puede ser obtenida realizando el cálculo:

$$W^* = (S'^* - S)/\alpha$$

(3) Por último, la marca de agua espacialmente codificada es una vez más permutada usando la transformación de Arnold, con periodo de permutación P-k, donde P es el periodo de transformación de la imagen. La marca de agua posiblemente distorsionada W^* es obtenida en su forma original.

Tercer Método: SVD basado en el intercambio de valores

Estructura de la Marca de Agua

La marca de agua será una imagen binaria de tamaño PxP.

Proceso de ocultación de la marca

El esquema de marcado propuesto para un logo binario está basado en la SVD de bloques de una imagen. Se supone que el tamaño de una imagen N×N. Esta imagen es dividida en $(N/4) \times (N/4)$ bloques no solapados cuyo tamaño es 4×4.

Para comenzar el trabajo, se explora un modo basado en la función hash del esquema de Rabin para decidir la posición de los bloques ocultados. Primero, se elige dos números primos grandes, p y q. Se hace Z = pxq, donde p y q son secretos y Z es

CAPÍTULO 3: ANTECEDENTES

publica. Después, aleatoriamente se eligen dos valores, k_1 y k_2 , como semillas secretas para calcular la posición de los bloques ocultados como sigue:

$$X_i = X_{i-1}^2 \bmod Z, \quad X_0 = k_1^2 \bmod Z \quad (1)$$

$$Y_i = Y_{i-1}^2 \bmod Z, \quad Y_0 = k_2^2 \bmod Z \quad (2)$$

$$x_i = X_i \bmod \left(\frac{N}{4}\right), \text{ and} \quad (3)$$

$$y_i = Y_i \bmod \left(\frac{N}{4}\right) \quad (4)$$

(x, y) es la posición de los bloques ocultados del iésimo bit en el canal de bits de la marca de agua. Además, dos posiciones arbitrarias deben ser diferentes. El proceso anterior puede ser considerado como un generador de números pseudoaleatorios seguro.

De acuerdo con lo previamente comentado, hay $(N/4) \times (N/4)$ bloques no solapados. El tamaño es 4×4 para cada bloque. Y hay $P \times P$ valores binarios que necesitan ser ocultados. Para conseguir alta robustez, cada valor binario de la marca de agua debería ser ocultado en tres bloques diferentes. Por tanto, el canal de bits de la marca de agua debe ser copiado tres veces para obtener $P \times P \times 3$ valores binarios, y debe haber $P \times P \times 3$ posiciones diferentes que son generadas por el generador de números pseudoaleatorios, es decir, $1 \leq i \leq P \times P \times 3$. Se asume que B_j , donde $j = x_i \times (N/4) + y_i$ y $1 \leq j \leq (N/4) \times (N/4)$, es el bloque correspondiente de (x_i, y_i) en esta imagen y W_i es el iésimo valor binario en el canal de bits de las tres marcas de agua originales. Así, $(N/4) \times (N/4)$ debe ser tan largo como $P \times P \times 3$.

El algoritmo de ocultación

Entrada: Coordenada (x_i, y_i) , bloque B_j , y bit de la marca de agua W_i

Salida: una imagen marcada

Paso 1: Poner $i = 1$.

Paso 2: Calcular la SVD del bloque correspondiente, B_j , de (x_i, y_i) . Obtener tres matrices

que son U_j , S_j , y V_j . Se asume que $S_j = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 \\ 0 & 0 & 0 & \sigma_4 \end{bmatrix}_j$

CAPÍTULO 3: ANTECEDENTES

Paso 3: Poner σ_3 igual que σ_2 . Obtener S'_j que es

$$\begin{bmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma'_3 & 0 \\ 0 & 0 & 0 & \sigma_4 \end{bmatrix}_j$$

Paso 4: Poner σ_2 igual a $\sigma_2 + \delta \times W_i$, donde δ es una constante. Si $\sigma_1 < \sigma_2 + \delta \times W_i$, $\sigma_1 = \sigma_2 + \delta \times W_i$. Se obtiene S''_j que es

$$\begin{bmatrix} \sigma'_1 & 0 & 0 & 0 \\ 0 & \sigma'_2 & 0 & 0 \\ 0 & 0 & \sigma'_3 & 0 \\ 0 & 0 & 0 & \sigma_4 \end{bmatrix}_j$$

Paso 5: Reconstruir el bloque marcado, BW_j , que es igual a $U_j \times S''_j \times V^T_j$.

Paso 6: Poner $i = i + 1$. Volver al paso 2 hasta que $i = P \times P \times 3$.

De acuerdo con el algoritmo de ocultación propuesto, el número de valores singulares que son modificados es como máximo 3. Además, el mayor valor singular, σ_1 , es normalmente más grande que σ_2 para un bloque en una imagen. Así, la condición de $\sigma_1 < \sigma_2 + \delta \times W_i$ rara vez ocurre. En el algoritmo propuesto, la diferencia entre el bloque marcado y el bloque original es pequeña. La razón es que casi solo se modifica el segundo y tercer valor singular cuando $W_i = 1$ y solo se modifica el tercer valor singular cuando $W_i = 0$ [16].

Proceso de extracción de la marca

En el proceso de extracción, se puede extraer directamente la marca de agua de la imagen marcada, y extraer la marca de agua con bastante precisión si la imagen marcada no está modificada por alguna técnica de procesamiento de imágenes. Para empezar el proceso de extracción, se debe dar la clave secreta original que es (k_1, k_2) para obtener (x_i, y_i) por las ecuaciones (1) a (4), y asumir que el bloque correspondiente de (x_i, y_i) es BW'_j que es probablemente la distorsión de BW_j de la imagen marcada.

En el algoritmo de ocultación propuesto, la diferencia entre el segundo y el tercer valor singular es poner a 0 o a δ . Así, si la diferencia entre los segundos y terceros valores que son calculados de la imagen marcada es mayor que un umbral, el bit de la marcada de agua será considerado como 1, de otro modo será considerado como 0. De acuerdo a los resultados de los experimentos, es mejor que este umbral sea $\delta/2$. Además, de tres veces ocultada la marca de agua, esta puede ser determinada por estas tres marcas de agua. Se puede determinar el valor binario de la marca de agua extraída por una fuerte votación en "1" o "0" de estas tres marcas de agua. Así, la tasa de corrección de la marca de agua extraída será promovida. La tasa de corrección

CAPÍTULO 3: ANTECEDENTES

de la marca de agua extraída es medida por el BCR (Bit Correction Ratio). La fórmula del BCR es demostrada como sigue:

$$BCR = \frac{\sum_{i=1}^{P \times P} \overline{W_i \oplus W'_i}}{P \times P} \times 100\%$$

donde W_i es el iésimo valor binario en el canal de bits de la marca de agua original, W'_i es el iésimo valor binario en el canal de bits de la marca de agua extraída, y \oplus representa un operador de OR exclusivo. Obviamente, cuando el BCR es mayor, la similaridad entre la marca de agua original y la extraída es alta.

El algoritmo de extracción de la marca de agua es el que sigue.

El algoritmo de extracción

Entrada: (x_i, y_i) y BW'_j

Salida: la marca de agua

Paso 1: Poner $i = 1$.

Paso 2: Calcular los SVD del correspondiente bloque, BW'_j , de (x_i, y_i) . Obtener tres

matrices que son UW_j , SW_j , y VW_j . Asumir que $SW_j = \begin{bmatrix} \sigma w_1 & 0 & 0 & 0 \\ 0 & \sigma w_2 & 0 & 0 \\ 0 & 0 & \sigma w_3 & 0 \\ 0 & 0 & 0 & \sigma w_4 \end{bmatrix}_j$

Paso 3: Si $\sigma w_2 - \sigma w_3 > \delta/2$, poner $WT_i = 1$; de otro modo, poner $WT_i = 0$.

Paso 4: Poner $i = i+1$. Ir al Paso 2 hasta que $i = P \times P \times 3$.

Paso 5: Poner $i = 1$.

Paso 6: Si $WT_i + WT_{i+pxp} + WT_{i+pxpx2} \geq 2$, poner $W'_i = 1$; de otro modo, poner $W'_i = 0$.

Paso 7: Poner $i = i + 1$. Ir al paso 6 hasta que $i = P \times P$.

Ya que los valores singulares de un bloque en una imagen no cambian enormemente cuando una pequeña interferencia es añadida a la imagen, usando el algoritmo de extracción, la marca de agua puede aún ser extraída de la imagen marcada que es procesada por algún esquema de procesamiento de imágenes.

Comentarios

A este algoritmo, como a otro de los anteriores, también tiene un comentario que dice que no es tan bueno como parece, por ello hace dos ataques observando que la robustez no es buena y que para identificar a un autor no es muy fiable.

Ataque a la robustez

Este ataque invalida la afirmación de los diseñadores de que el esquema es robusto. Los pasos del ataque son los que siguen:

1. Se denota a la imagen marcada, I_w como una matriz NxN. I_w es dividido en bloques 4x4 no solapados B_j ($1 \leq j \leq (N/4) \times (N/4)$).
2. Se pone $j = 1$.
3. Se realiza la SVD en B_j

$$B_j = U_j S_j V_j^T$$

Se asume que $S_j = \begin{bmatrix} S_1 & 0 & 0 & 0 \\ 0 & S_2 & 0 & 0 \\ 0 & 0 & S_3 & 0 \\ 0 & 0 & 0 & S_4 \end{bmatrix}$

4. Se pone $s2 = s3$.
5. Se realiza la SVD en el bloque modificado, BM_j como:

$$BM_j = U_j S'_j V_j^T$$

donde S'_j son los valores singulares modificados.

6. Se incrementa j en 1 y se vuelve al paso 3 hasta que todos los bloques hayan sido procesados, por ejemplo, hasta que $j = (N/4) \times (N/4)$.

Durante los pasos de extracción, la marca de agua fallará para ser extraída de la imagen marcada modificada debido a $s2 = s3$. Esto es porque basándose en los pasos de extracción de la marca de agua, el bit de la marca de agua es siempre 0 y la condición es siempre falsa. No se ha necesitado conocer los pares de clave secreta para determinar la posición de los bloques ocultados. Por lo tanto, la robustez esta invalidada.

Ataques en pruebas de propiedad

Considerar como ejemplo ilustrativo un escenario donde Alice es la propietaria de una imagen I , y ella oculta su marca de agua W en la imagen I , para obtener la imagen marcada I_w . Un atacante Bob, obtiene I_w con éxito y repite el proceso de ocultación con su propia marca de agua WB en I_w , para obtener la imagen marcada I_{wb} . Esto, de hecho, elimina la marca de agua de Alice W de la imagen marcada I_w . Surge una disputa cuando Alice reclama que ella es la propietaria original de I pero ella no

CAPÍTULO 3: ANTECEDENTES

puede extraer su propia marca de agua W de la imagen marcada I_{wb} , así que ha sido falsificada por Bob. Por otra parte, Bob puede extraer su propia marca de agua WB de I_{wb} , ya que él ha sobreescrito la marca de agua de Alice W con su propia marca de agua WB usando el mismo esquema.

Ambos de los ataques funcionan porque el espacio de ocultación en el que la marca de agua ha sido ocultada, puede ser fácilmente modificado para cambiar la marca de agua ocultada.

Además, ambos de estos ataques directamente invalidan la afirmación de los diseñadores de que sus esquemas son robustos y pueden ser usados para la protección de los derechos de autor de la imagen [22].

Cuarto Método: SVD basado en el orden en los coeficientes

Estructura de la Marca de Agua

La marca de agua W será una imagen binaria consistente en $w \cdot h$ bits, donde $W = (w_1, w_2, \dots, w_{w \cdot h})$ y $w_i \in (0, 1)$ [18].

Proceso de ocultación de la marca

Para utilizar las características del dominio SVD para ocultar una marca de agua, los coeficientes de las componentes D y U eran explorados. Aquí, se encontraron dos características importantes de las componentes D y U. La primera característica, el número de coeficientes distintos de cero en la componente D podían ser usados para determinar la complejidad de un bloque (matriz). Generalmente, el mejor número de coeficientes distintos de cero indicarían mejor complejidad. Para un esquema de marcado basado en bloques, un bloque más complejo era favorecido para ocultar una marca de agua con perceptibilidad. La segunda característica, la relación entre los coeficientes en la primera columna de la componente U podían ser preservados cuando era realizado el procesamiento general de la imagen. Ambas características soportaban la idea de desarrollar un esquema de marcado basado en una SVD robusta.

En el esquema de marcado propuesto, la imagen de cobertura será una imagen en escala de grises. La imagen de cobertura será particionada en bloques de $n \cdot n$ pixeles. Y después los bloques serán transformados por la SVD. El número de coeficientes distintos de cero en la componente D de cada bloque serán seleccionados de acuerdo al generador de números pseudoaleatorios (PRNG) y las características de

CAPÍTULO 3: ANTECEDENTES

la componente D. Usando el PRNG se incrementa la seguridad del marcado. Aplicando las características de la componente D se previene bloques lisos para ser seleccionados y beneficia la percepción de la imagen marcada.

En cada bloque seleccionado, es examinada la relación entre la primera columna de coeficientes ($\mu_{1,1}, \mu_{2,1}, \dots, \mu_{n,1}$) en la componente U. Esta relación puede ser tomada como la diferencia de magnitud entre los coeficientes vecinos. La diferencia de magnitud puede ser un valor positivo o negativo. De otro modo, una relación negativa sería asignada. La relación puede ser preservada cuando sea realizado el procesamiento general de la imagen. En otras palabras, cuando un coeficiente tenga una magnitud más grande que otra, la relación positiva no será fácilmente afectada por el procesamiento de la imagen.

De acuerdo a las características de la componente U, parecería que si una relación positiva es hallada, un valor de bit a 1 estaría oculto. De otro modo, un valor de bit a 0 estaría insertado. Por ello, los coeficientes de la componente U podrían ser modificados ocultando una marca de agua. En la modificación de los coeficientes, dos escenarios deben ser discutidos. Primero, si la diferencia de magnitudes coincide con la marca insertada (por ejemplo, la relación positiva coincidiendo un valor de bit a 1 o la relación negativa coincidiendo un valor de bit a 0), los coeficientes son conservados. Segundo, si la diferencia de magnitudes no coincide con la marca insertada, los coeficientes deben ser modificados. Sin embargo, la modificación de los coeficientes de la componente U podrían alterar los valores del pixel original y degradar la calidad de la imagen marcada. A mayor modificación de la componente U, es más la distorsión de la calidad de imagen y la fuerza de robustez. De otra manera, se consigue que una poca modificación implique una mejor calidad de imagen y una débil resistencia. Esto es, en otras palabras, una compensación entre robustez y calidad.

Para conservar la calidad de la imagen y mantener una fuerte robustez del esquema de marcado, la modificación del coeficiente está más considerada. Para que la complejidad de cada bloque sea mejor, la diferencia de magnitud entre coeficientes vecinos tenía que coincidir ambos con la marca de agua ocultada y un umbral predefinido en el que se incrementase la robustez de la marca. En otras palabras, cuando la diferencia de magnitud coincide con la marca de agua pero sea más pequeña que el umbral predefinido de la diferencia de magnitudes, ambos coeficientes tienen que ser muy modificados. Ambos coeficientes modificados no solo reducen la percepción de la imagen si no que también aumentan la robustez a resistir ataques. Además, si la segunda posibilidad descrita anteriormente se diese, la diferencia de magnitud entre dos coeficientes modificados debería ser mejor o igual que un umbral predefinido. Esto significa que la distancia entre dos coeficientes modificados debe ser más grande aún contra ataques.

Proceso de extracción de la marca

El proceso de extracción de la marca es similar al de ocultación. Los primeros tres pasos del proceso de extracción de la marca son los mismos que en el proceso de ocultación, excepto que la imagen de cobertura es remplazada por la imagen marcada. Una vez un bloque ocultado sea detectado de acuerdo a las características de la componente D y el PRNG, la relación entre los coeficientes de la componente U son examinados. Si una relación positiva es detectada, al valor del bit de la marca extraída se le asigna un 1. De otro modo, el valor del bit de la marca extraída se pone a 0. Estos valores de bit extraídos forman la marca extraída. La marca de agua extraída puede ser identificada por el ojo humano o comparada con la marca de agua original.

Comentar también que existe otro método creado por otros autores y que tiene las mismas bases que el anterior [17].

Quinto Método: SVD basado en la proximidad a un intervalo

Estructura de la Marca de Agua

La marca de agua es una imagen binaria de tamaño P×P.

Proceso de ocultación de la marca

Es aplicada la transformación SVD basada en bloques, $1 \leq p \leq N/2$ y $1 \leq q \leq N/2$ con tamaño de bloque de $M \times M$.

La imagen marcada $w(i, j)$, $1 \leq i \leq N/2M$ y $1 \leq j \leq N/2M$ es ocultada en las columnas de cada bloque de la matriz U. Para cada bloque $M \times M$ de la matriz U, u_{11} (primera fila primera columna) y u_{21} (segunda fila primera columna) son modificados como sigue:

$$u_{diff} = |u_{11}| - |u_{21}|$$

$$\text{Si } (w(i, j) = 1 \& u_{diff} < \alpha) \text{ o } (w(i, j) = 0 \& u_{diff} > \alpha)$$

$$u_{21} = -u_{21} - (\alpha + u_{diff}) / 2$$

$$u_{11} = -u_{11} + (\alpha + u_{diff}) / 2$$

CAPÍTULO 3: ANTECEDENTES

donde '| |' indica el valor absoluto. La modificación anterior es aplicada a los coeficientes de cada bloque de la matriz U. Aquí, α es una constante.

Después de la modificación de los coeficientes de la matriz U, la inversa de la SVD se aplica a cada bloque para obtener la imagen marcada [19].

Proceso de extracción de la marca

Se aplica la transformación SVD a la imagen marcada. Los elementos y la matriz U generada del paso previo son comparados para generar la marca de agua:

para $1 \leq i \leq N/2, 1 \leq j \leq N/2$

si $|u_{11}| > |u_{21}|$ entonces $w(i, j) = 1$

sino $w(i, j) = 0$.

3.2.7. Algoritmo basado en LSB

Introducción

El método más sencillo de ocultación de marcas de agua, sería ocultar la marca de agua en el bit menos significativo del objeto de cobertura. Dado la extraordinaria alta capacidad del canal usando la cobertura entera para transmisión en este método, un pequeño objeto podría ser ocultado múltiples veces. Incluso si la mayor parte de esta se pierde debido a ataques, con que una sola marca de agua sobreviviese sería considerado un éxito.

La sustitución del Bit Menos Significativo o LSB, sin embargo, a pesar de su simplicidad trae gran cantidad de inconvenientes. Aunque podría sobrevivir a transformaciones tales como recortado, alguna adición de ruido o pérdida de compresión es probable rechazar la marca de agua. Un constante ataque mejor sería simplemente poner los bits LSB de cada pixel a uno, por lo menos rechazando la marca de agua con un insignificante impacto en el objeto de cobertura. Además, una vez es descubierto el algoritmo, la ocultación de la marca de agua podría ser fácilmente modificada por una parte intermediaria.

Una mejora en la sustitución básica LSB seria usar un generador de números pseudo-aleatorios, para determinar los pixeles a ser usados para ocultar, basado en una "semilla" o clave dada. La seguridad de la marca de agua seria mejorada ya que la marca de agua podía no ser ya fácilmente vista por la parte intermediaria. El algoritmo,

CAPÍTULO 3: ANTECEDENTES

sin embargo, aún sería vulnerable reemplazando el LSB con una constante. Incluso en localizaciones que no estaban usadas para bits de marcado, el impacto de la sustitución en la imagen de cobertura sería insignificante. La modificación LSB provee ser una simple y poderosa herramienta para la esteganografía, sin embargo, falta la robustez básica que las aplicaciones marcadas requieren [1].

Estructura de la Marca de Agua

La marca de agua será una imagen binaria de tamaño MxN.

Proceso de ocultación de la marca

Para ocultar la marca de agua en una imagen en escala de grises habrá que tratar la imagen y la marca como bits, de esta forma cada bit de la marca de agua se insertará en la imagen de forma que sustituirá el último bit o menos significativo. Así, se conseguirá una imagen marcada con la marca de agua deseada.

Proceso de extracción de la marca

Una vez obtenida una imagen marcada por este método se podrá recuperar la marca de agua que contiene de la misma forma que se marcó, recuperando el último bit de la imagen marcada se formará la marca que podrá ser comparada con la original.

3.2.8. Algoritmo basado en las Secuencias Caóticas

Introducción

Los sistemas caóticos son deterministas, sistemas que son gobernados por dinámicas no lineales. Estos sistemas muestran comportamiento determinista que es muy sensible a sus condiciones iniciales, de modo que los resultados son no correlados y parecen aleatorios.

Una categoría de sistemas caóticos son los mapas caóticos. Un mapa caótico, que puede ser considerado como una función caótica bidimensional, es una herramienta que reorganiza los pixeles de una imagen y rompe la continuidad espacial.

CAPÍTULO 3: ANTECEDENTES

Si se transforma la imagen caótica resultante en el dominio de las frecuencias, el número de coeficientes significantes es altamente incrementado en comparación con la respectiva imagen transformada [23].

Hasta la fecha, la mayoría de las técnicas de marcado disponibles en libros son generadas por secuencias de números pseudoaleatorios. Sin embargo, las secuencias generadas por iteración de mapas caóticos constituyen una alternativa diferente a las secuencias de marcas de agua pseudoaleatorias. Tales secuencias podrían ser generadas por iteración de una función caótica con una semilla y un valor inicial, y esta secuencia podría ser cuantificada para formar una marca de agua binaria. El caos es conocido por ser un sistema altamente sensible a su estado inicial y la semilla, y un ligero cambio en el estado inicial y la semilla puede provocar el caos variando enormemente después de algunas iteraciones.

Un gran número de funciones caóticas ha sido propuesto para propósito de generación de marcas de agua. Uno de los más simples y muy estudiados es el mapa logístico escrito en su forma recursiva $X_{n+1} = \lambda \cdot X_n \cdot (1 - X_n)$. Aquí $0 \leq X_n \leq 1$, y $0 \leq \lambda \leq 4$, λ es llamado el parámetro de bifurcación. Dependiendo de los valores de λ , la dinámica de este sistema puede cambiar dramáticamente. Para $3.57 \leq \lambda \leq 4$, la secuencia es no periódica, no convergente y muy sensible al valor inicial. Una señal caótica unidimensional es inicialmente producida y después alabeada para generar una marca de agua base bidimensional. A fin de obtener la marca de agua base bidimensional usando escaneado raster para la señal caótica puede ser convertido en una imagen bidimensional que podría ser usada como marca de agua. El punto de secuencia inicial (condición inicial del mapa) es considerado como la clave de la marca de agua [15].

Primer Método: Secuencias Caóticas

Estructura de la Marca de Agua

La marca de agua será una imagen de dimensiones MxN.

Proceso de ocultación de la marca

La imagen original de cobertura I será protegida con una marca de agua W de tamaño MxN. El proceso de ocultación de la marca de agua es escalado por un factor de ganancia α (el valor de la marca de agua es 0 o 1). En el dominio espacial, la ocultación es realizada como:

CAPÍTULO 3: ANTECEDENTES

$$I_W = I + \alpha W \quad (1)$$

donde

I_W : la imagen marcada;

I : la imagen original;

α : el factor de ganancia;

W : la marca de agua.

Proceso de extracción de la marca

La detección y extracción de la marca de agua es otro paso importante en los algoritmos de marcado digitales. Por tanto, para que un esquema de marcado sea efectivo, es importante poner un umbral apropiado que minimice el número de falsos negativos y positivos. A fin de poner un umbral apropiado, se correla la marca de agua extraída con un número grande de marcas de agua aleatorias y se oculta la marca de agua. El umbral debería ser más grande que el valor máximo de correlación entre la marca extraída y las marcas de agua aleatorias y ser mucho menor que la correlación entre la marca de agua extraída y la ocultada.

La respuesta de la correlación debido a la marca de agua correcta es mucho más fuerte que la respuesta a la correlación de las marcas de agua incorrectas.

Segundo Método: Secuencias Caóticas y DCT

Estructura de la Marca de Agua

La marca de agua es una secuencia pseudoaleatoria de tamaño N formada por números pertenecientes a una normal $N(0,1)$, con media 0 y varianza igual a 1.

Proceso de ocultación de la marca

Se presenta un esquema de marcado basado en una función caótica de dos dimensiones, llamada “toral automorphism”, que no es más que una variación sencilla del mapa del gato de Arnold. Esta función caótica cíclica, con cada aplicación en una imagen cuadrada, cambia de lugar sus pixeles. Después es aplicada T veces, donde T es

CAPÍTULO 3: ANTECEDENTES

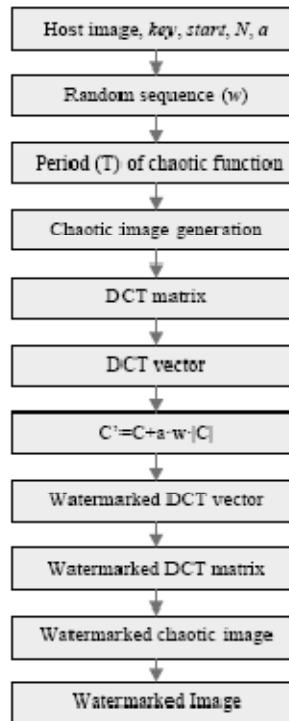
el periodo de la función, los pixeles vuelven a su localización inicial. Si (x, y) son las coordenadas iniciales de un pixel, las coordenadas resultantes de la función caótica (x', y') son dadas por:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ l & l+1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \bmod N \quad (1)$$

donde N denota el ancho de la imagen y l es un parámetro entero, que afecta al periodo T de la función caótica.

A fin de ocultar la información en la imagen de cobertura ciertos parámetros deben ser especificados: un entero el cual derive de la secuencia aleatoria (key), el coeficiente inicial ($start$), el número total de coeficientes DCT que serán afectados (N), y el factor de ganancia (α).

Los pasos del algoritmo de ocultación son mostrados en la siguiente figura:



1. Una secuencia aleatoria (w) es generada de acuerdo a la clave. La secuencia consiste en N números aleatorios que son normalmente distribuidos con media cero y varianza uno.
2. El periodo T de la función caótica (1) para la imagen particular es calculado.
3. La función caótica (1) es aplicada $[T/3]$ veces a la imagen de cobertura creando la imagen caótica. La imagen caótica parecerá más a ruido.
4. Los coeficientes DCT de la imagen caótica, en su totalidad, son calculados formulando la matriz respectiva DCT.
5. Se produce un vector por escaneo en zigzag de la matriz DCT.

CAPÍTULO 3: ANTECEDENTES

6. Cada coeficiente DCT desde el comienzo hasta el comienzo+N-1 es alterado de acuerdo a la siguiente regla:

$$C' = C + \alpha \cdot w \cdot |C|$$

donde α denota el factor de ganancia, C' denota el coeficiente alterado, C denota el coeficiente inicial, y w denota el respectivo elemento de la secuencia aleatoria.

7. Los coeficientes alterados de la matriz DCT marcada por escaneo inverso en zigzag.
8. La imagen marcada caótica es generada por aplicación de la función inversa DCT a la matriz DCT marcada.

La función caótica (1) es aplicada $T - [T/3]$ veces a la imagen caótica marcada, produciendo la imagen marcada final [23].

Proceso de extracción de la marca

Para el proceso de extracción de la marca de agua es básicamente seguir los pasos del proceso de ocultación pero de forma inversa. Es decir, primero se permuta la imagen marcada, después se calcula la DCT a la misma y seguidamente se extraen los coeficientes en zigzag. Por último, se comprobará si la correlación entre las marcas es alta o no.

3.2.9. Algoritmo basado en PCA

Introducción

Una imagen $I(m,n)$ con tamaño $M \times N$ donde m y n toman valores enteros desde 0 hasta $M-1$ y $N-1$. Se escribe la transformación de la imagen como:

$$[T(u, v)] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(m, n) \varphi^{(u,v)}(m, n) \quad (1)$$

donde $[\varphi(u,v)]$ es una matriz de transformación (matriz básica), $[T(u,v)]$ la transformación de la imagen. La transformación inversa puede ser definida como:

$$[I(i, v)] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} T(m, n) \varphi^{-1}(u,v)(m, n) \quad (2)$$

donde $[\varphi(u,v)]^{-1}$ es la matriz de transformación inversa. Es notado que la transformada discreta de Fourier usa funciones sinusoidales como funciones básicas, DCT usa funciones de coseno, y la transformación wavelet usa una wavelet particular como

CAPÍTULO 3: ANTECEDENTES

función básica. El PCA usa la matriz básica obtenida por búsqueda de los vectores propios de la matriz de imagen de correlación.

Se extrae los componentes principales de los subpixeles de una imagen encontrando la matriz de transformación $[\varphi]$. Cada subpixel es transformado por la matriz transformada PCA $[\varphi]$. Las marcas de agua son ocultadas en los componentes significantes de los subpixeles que son seleccionados.

Los siguientes pasos aplicados a la imagen original encuentran la matriz de transformación $[\varphi]$ y la imagen marcada.

Paso 1. Primero, se partitiona la imagen en un número de subimágenes por conveniencia en la implementación numérica. Se considera cada subimagen como un vector (vector de pixeles); los vectores de datos de la imagen pueden ser expresados como: $I = (i_1, i_2, i_3, \dots, i_m)^T$ donde el vector i_i es la i ésima subimagen y T denota la matriz transpuesta, cada subimagen tiene n^2 pixeles. Cada vector i_i tiene dimensión n^2 .

Paso 2. Se calcula la matriz de covarianza C_x de subimágenes y los vectores propios, valores propios de la matriz de covarianza: $C_x = E(I - m_i)(I - m_i)^T$ donde $m_i = E(I)$ son vectores con la media de cada subvector i_i . Cada subimagen podría ser descompuesta en coeficientes no correlados encontrando primero los vectores propios (función básica de transformación) y los valores propios correspondientes a la matriz de covarianza: $C_x\Phi = \lambda_x\Phi$. La matriz $[\varphi]$ formada por los vectores propios $\Phi = (e_1, e_2, e_3, \dots, e_n)$. Los valores propios λ , ($\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots \geq \lambda_n$) y vectores propios $[\varphi]$ son ordenados en orden descendiente. La matriz $[\varphi]$ es una matriz ortogonal llamada función básica de PCA.

Paso 3. Transformar subimágenes en coeficientes no correlados.

La imagen original espectralmente correlada I puede ser descorrelada por la matriz básica $[\varphi]$, y se obtiene la imagen propia Y , los valores correspondidos en la matriz y_i son los componentes principales de cada subpixel donde $Y = \Phi^T I = (y_1, y_2, \dots, y_n)^T$.

Una vez las componentes principales se han calculado por la función básica PCA, se tiene que decidir qué marca de agua se va a ocultar. Desde que la descomposición está hecha bloque por bloque de acuerdo al máximo de la varianza, la primera componente contiene un conjunto de información, y la última componente corresponde principalmente a ruido. Las marcas de agua son ocultadas en las componentes perceptualmente significantes de los subpixeles que son seleccionados. La imagen es marcada por las siguientes condiciones:

- Las primeras componentes son normalmente conservadas.

CAPÍTULO 3: ANTECEDENTES

- La marca de agua es ocultada en algunas de las componentes principales de los coeficientes no correlados de la subimágen.

Paso 4. Recuperar la marca de agua puede ser realizado por la siguiente ecuación: $I = (\Phi^T)^{-1}Y = \Phi Y$ [24].

Primer Método: PCA

Estructura de la Marca de Agua

En este método propuesto la marca de agua consiste de secuencias de números pseudoaleatorias de longitud M con valores w normalmente distribuidos $W = (w_1, w_2, \dots, w_M)$.

Proceso de ocultación de la marca

En este método basado en las propiedades del PCA, se selecciona un conjunto de coeficientes en cada subbloque para poner la marca de agua modificando los coeficientes desde que la marca de agua es colocada en los componentes perceptualmente significantes de la señal. El esquema es para ocultar la marca de agua en componentes predefinidas de cada subbloque de coeficientes no correlados PCA. Los coeficientes ocultados son modificados por la siguiente ecuación:

$$y' = y + \xi y_i w_i \quad (3)$$

donde $i = 1, 2, \dots, m$, ξ es un parámetro de escalado para controlar la fuerza de la marca de agua y y' es los coeficientes marcados. La imagen marcada I' es recibida por aplicación del proceso inverso de PCA [24].

Proceso de extracción de la marca

La detección de la marca de agua es aplicada a la imagen marcada I' . Los coeficientes no correlados de los subbloques de I' son calculados aplicando la función básica de imagen PCA Φ y los coeficientes que fueron ocultados en las marcas de agua son seleccionados para generar el vector de coeficientes marcados $Y^* = (y_1^*, y_2^*, \dots, y_M^*)$.

El valor de correlación entre el código marcado W y posiblemente el coeficiente corrupto Y^* es calculado para detectar la marca:

CAPÍTULO 3: ANTECEDENTES

$$CV = \frac{WY^*}{M} = \frac{1}{M} \sum_{i=1}^M w_i \cdot y_i^* \quad (4)$$

La correlación entre las marcas de agua es calculada, primero por la marca W, y después por 1000 marcas diferentes. El CV de correlación puede ser usado para determinar si una marca dada esta presente o no. Para la detección de la marca de agua, es definido un umbral T por:

$$T = \frac{\xi}{2M} \sum_{i=1}^M y_i \quad (5)$$

Para este método existen otros artículos en los cuales se hace mención y se aplica a imágenes en color y video [25, 26].

Segundo Método: PCA para construir la imagen de referencia

Estructura de la Marca de Agua

La marca de agua es una secuencia aleatoria teniendo valores entre -1 y 1 generados por una semilla [27].

Proceso de ocultación de la marca

La imagen de cobertura esta en escala de grises con dimensiones MxN. El método está basado en selección de algunos pixeles aleatoriamente por la diferencia de la imagen original y su referencia satisfaciendo la condición ($s < |A(i,j) - A'(i,j)| < t$) y modificando después de acuerdo con el valor de la marca de agua. Los pixeles seleccionados son guardados como clave secreta para usar en la extracción. El método es semiciego en este sentido. Es usado el PSNR para medir la calidad de la imagen marcada. La imagen de referencia es obtenida usando PCA o bloques PCA. En la reconstrucción son usados b vectores principales, y la imagen de referencia es obtenida. En la aproximación de bloques PCA, la imagen es dividida en bloques no solapados (el tamaño del bloque debería ser apropiado con el tamaño de la imagen), el PCA es hecho para cada bloque, y cada bloque es reconstruido usando d vectores principales. Los bloques de referencia son reunidos siempre para obtener la referencia basada en bloques PCA.

Para ocultar la marca de agua en la imagen original se realizan los siguientes pasos:

CAPÍTULO 3: ANTECEDENTES

- Obtener la imagen de referencia de la imagen de cobertura por PCA o bloques PCA, A'
- Obtener la diferencia entre la imagen original y la de referencia, $A - A'$
- Encontrar los pixeles que satisfagan $s < |A(i,j) - A'(i,j)| < t$, P ; s y t son positivos, y enteros
- Seleccionar algunos de ellos aleatoriamente (p_i, p_j) y modificarlos:

$$A_w(p_i, p_j) = \begin{cases} A'(p_i, p_j) + \alpha, & \text{if } w(k) = 1 \text{ and } A(p_i, p_j) > A'(p_i, p_j) \\ A'(p_i, p_j) - \alpha, & \text{if } w(k) = -1 \text{ and } A(p_i, p_j) < A'(p_i, p_j) \end{cases}$$

Aquí, $\alpha = (s+t)/2$, $k = 1$ hasta $N \times N$.

Proceso de extracción de la marca

Para extraer la marca de agua de la imagen marcada se realizan los siguientes pasos:

- Encontrar la imagen de referencia de la imagen marcada, A'_w
- Extraer la marca de agua de acuerdo a la diferencia entre la imagen marcada y su referencia en las localizaciones ocultadas como:
$$w'(k) = \begin{cases} 1, & \text{if } A_w(p_i, p_j) \geq A'_w(p_i, p_j) \\ -1, & \text{if } A_w(p_i, p_j) < A'_w(p_i, p_j) \end{cases}$$
- Comparar las marcas extraída y original mediante la similaridad:
$$\text{Sim}(w, w') = \frac{w \cdot w'}{\sqrt{w \cdot w'}}$$

CAPÍTULO 3: ANTECEDENTES

CAPÍTULO 4: RESTRICCIONES

4.1. Factores dato

Los factores dato son restricciones impuestas bien por el cliente o bien por las características particulares del proyecto. En este proyecto no existen consideraciones por parte del cliente, sólo las impuestas por el tipo de proyecto.

Las restricciones iniciales serán las impuestas por la programación de los distintos algoritmos seleccionados para formar parte del estudio.

4.2. Factores estratégicos

Los factores estratégicos son aquellos que se derivan del diseño del software y que condicionarán las distintas propuestas que se lleven a cabo para cada elemento.

Se podrá considerar que formarán parte de los factores estratégicos la elección del entorno de trabajo, el lenguaje de programación, las herramientas de desarrollo, la interfaz simple de usuario para el ejemplo, el formato de las imágenes con las que se trabajará, etc.

Como entorno de trabajo se ha utilizado el sistema operativo Windows por la limitación que tiene trabajar con un programa realizado para una plataforma específica como es Matlab. Por otro lado, Matlab ofrece, como software matemático y multifuncional que es, muchas ventajas para la manipulación de imágenes.

Por lo dicho, se ha utilizado el lenguaje de programación que Matlab tiene, aunque también ofrece la oportunidad de utilizar lenguajes como C o C++, dando la

CAPÍTULO 4: RESTRICCIONES

posibilidad de empotramiento de código. Esto es una ventaja, ya que el programa en sí se encuentra implementado en dichos lenguajes y la sintaxis que ofrece es muy similar, lo que hace fácil la programación ya que en los estudios cursados los lenguajes de programación dados han sido estos en su mayoría. Así, se tiene todas las ventajas que estos lenguajes proporcionan junto al que el propio Matlab ya posee al ser software matemático con mucha precisión en sus cálculos.

En cuanto a la interfaz simple para mostrar el ejemplo de presentación, el propio Matlab ya proporciona una característica para realizarla y por ello se ha hecho uso de ella, aun no habiendo mucha documentación sobre esta ni información sobre desarrollos en internet. Se podría haber elegido alguna librería gráfica como QT4 que se puede desarrollar en C++, pero al no haber tiempo de auto aprender al manejo de la misma por eso se decidió hacerlo con la propia utilidad que Matlab proporciona.

Si se habla sobre el formato de las imágenes que se van a utilizar para la inserción y extracción de las marcas de agua, el formato que se ha escogido ha sido el .bmp en blanco y negro y escala de grises. Esta elección ha sido ya sea por que los autores de los algoritmos estudiados las han utilizado o por simpleza en el desarrollo. En cualquier caso, se podría replantearse el uso de otro formato de imágenes para el estudio de los algoritmos.

CAPÍTULO 5: RECURSOS

5.1. Recursos humanos

Los recursos humanos del presente proyecto serán el desarrollador del proyecto y la directora del mismo:

- Directora del Proyecto
 - Doctora Ángela Rojas Matas.
- Alumno desarrollador del proyecto
 - Raúl Pérula Martínez.

5.2. Recursos hardware

Básicamente hará falta un ordenador que soporte las necesidades para la ejecución de Matlab, estas son:

Windows

Operating System	Processors	Disk Space	RAM

CAPÍTULO 5: RECURSOS

32-bit MathWorks™ Products			
Windows® XP Service Pack 2 or Service Pack 3	Intel® Pentium 4 and above Intel Celeron Intel Xeon Intel Core AMD Athlon 64 AMD Opteron AMD Sempron	660 MB (MATLAB® only)	512 MB (At least 1024 MB recommended)
Windows Vista™ Service Pack 1			

Graphics

- 16-, 24-, or 32-bit OpenGL capable graphics adapter [28].

5.3. Recursos software

Los recursos software necesarios para el desarrollo del proyecto son, tener un sistema operativo Windows XP Service Pack 2 or Service Pack 3 o Windows Vista™ Service Pack 1 y tener instalada la aplicación MATLAB & Simulink Student Version - Release 2009a en dicho sistema operativo.

Para la visualización de las imágenes valdrá el visor de imágenes del propio sistema operativo.

Para la redacción de la documentación se hará uso de la aplicación de Microsoft Office, Microsoft Word 2007.

CAPÍTULO 6: ESPECIFICACIÓN DE REQUISITOS

6.1. Introducción

Este estudio tiene como objetivo la comparación de los diferentes métodos que existen para el tratamiento de las marcas de agua digitales sobre imágenes en escala de grises con formato Windows Bitmap (.bmp).

Con ello se podrá realizar un análisis de los diferentes algoritmos existentes hasta ahora, basándose en los que tienen más relevancia y desechar los que se basan en modificaciones de los mismos. Así, se obtendrá una comparación de los diferentes métodos para que pueda servir como referencia al resto de investigadores y desarrolladores, ya que hasta el momento no se ha realizado ningún estudio comparativo con esta relevancia en el campo.

Principalmente, la función de los algoritmos se basa en dos subfunciones, la primera es la inserción de la marca de agua y la segunda es la extracción de la misma.

El estudio se puede dividir funcionalmente en varios apartados:

- Algoritmos para tratado de marcas de agua.
- Ataques.
- Pruebas de robustez.
- Ayuda para el manejo de los algoritmos.

6.2. Algoritmos para tratado de marcas de agua

6.2.1. Inserción de la marca de agua

Como entradas para realizar la imagen marcada se necesitan la imagen y la marca de agua originales, la marca de agua original podrá ser un logotipo o una secuencia generada pseudoaleatoriamente.

Una vez cargadas ambas, se realizará el procesamiento de la imagen en los diferentes dominios haciendo uso de uno de los algoritmos estudiados.

La descripción detallada de los algoritmos se encuentra en el capítulo de antecedentes donde se podrá hacer un seguimiento de cuál es el funcionamiento de los mismos.

Como salidas se proporcionarán la imagen resultante del marcado, el tiempo de ejecución que ha tenido y el valor de PSNR de la imagen marcada. Opcionalmente se podrá hacer una visualización previa de la imagen marcada.

6.2.2. Extracción de la marca de agua

Como entradas para realizar la extracción de la marca de agua de la imagen marcada, dependiendo del algoritmo que se aplique, hará falta la imagen original, la marca original o algún dato auxiliar.

Cuando se tengan los datos necesarios se aplicará el procedimiento necesario basado en los algoritmos estudiados.

La descripción detallada de los algoritmos se encuentra en el capítulo de antecedentes donde se podrá hacer un seguimiento de cuál es el funcionamiento de los mismos.

Como salidas se obtendrán, dependiendo del algoritmo de extracción y del tipo de marca de agua que se haya utilizado, la marca de agua extraída el tiempo de ejecución que ha tenido, el número de errores que se han cometido respecto a la marca de agua original, la similaridad o correlación o la opción de poder hacer una vista previa de la marca de agua extraída.

6.3. Pruebas de robustez

Para probar la robustez que ofrecen los diferentes algoritmos estudiados se harán diferentes pruebas para comprobar cómo evolucionan a diferentes valores de los parámetros que utilizan.

Se realizarán una serie de ataques para observar cuánta resistencia tienen las imágenes marcadas a éstos y si la extracción de la marca de agua se realiza con éxito o fracaso.

6.3.1. Ataques

Como entrada se cogerá la imagen marcada, se le aplicará los seis ataques elegidos; que aunque se podrían haber hecho otros, se ha elegido éstos porque son los más utilizados en este ámbito. Como salida se obtendrá las imágenes atacadas que, seguidamente, se someterán a los algoritmos de extracción de la marca de agua para comprobar la resistencia a los mismos.

CAPÍTULO 6: ESPECIFICACIÓN DE REQUISITOS

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

7.1. Introducción

El análisis de los algoritmos se hará de dos maneras para que el estudio de los mismos sea el más completo de los que se pudiesen hacer. Para comenzar se hará un estudio individual de cada algoritmo, haciendo una visión de cómo se comportan los parámetros que el algoritmo contempla, la calidad que tiene la imagen que se obtiene después de la inserción de una marca de agua, la imagen que se extrae como marca de agua de la imagen marcada o el gráfico de similaridad o correlación entre las secuencias pseudoaleatorias utilizadas, el número de errores que existe en la marca de agua extraída, etc. De aquí se sacarán las conclusiones en cuanto a cada algoritmo.

Una vez realizado este estudio individual, se hará un estudio común entre los algoritmos que puedan ser comparables, con lo que se sacarán las conclusiones en cuanto a qué algoritmo se debería de elegir para según qué utilización se le deseé dar.

Para cuantificar la calidad de la imagen marcada se utilizará el PSNR y para la comparación entre marcas de agua de secuencias pseudoaleatorias se hará uso de la similaridad o correlación, los definiciones completas de estos se encuentran más extensamente explicados abajo.

7.1.1. PSNR

Para comparar la imagen original con la imagen marcada se utilizó la métrica PSNR (Peak Signal-to-Noise Ratio), la cual nos brinda una medida aproximada (ya que no tiene en cuenta algunas propiedades del modelo visual humano) de la modificación que sufre una imagen al insertarle una marca de agua, donde para valores de PSNR por debajo de los 30 dB corresponde a imágenes pobres y deterioradas, para valores entre 30 y 50 dB aumenta la calidad percibida y para valores por encima de los 50 dB se tiene imágenes de muy buena calidad [2].

$$PSNR = \frac{XY \max_{x,y} p_{x,y}^2}{\sum_{x,y} ((P_{x,y} - \bar{P}_{x,y})^2)}$$

7.1.2. Similaridad y correlación

Es altamente improbable que la marca extraída X^* sea idéntica a la marca original X . Se mide la similaridad de X y X^* por:

$$sim(X, X^*) = \frac{X^* \cdot X}{\sqrt{X^* \cdot X^*}}$$

Sería posible ofrecer otro tipo de medidas, por ejemplo, incluyendo el coeficiente estándar de correlación. Para decidir si X y X^* coinciden, uno determina si la $sim(X, X^*) > T$, donde T es un umbral. Se pone el umbral de detección para evitar falsos negativos (detecciones perdidas) y falsos positivos (alarmas falsas) [4].

7.2. Tipos de ataques

Para comprobar la efectividad de los algoritmos propuestos por los autores, a las imágenes marcadas se les va a realizar una serie de ataques para ver qué resistencia tienen a éstos cuando se extrae la marca. Los ataques que se realizarán son:

- Compresión JPEG
- Inserción de ruido gaussiano
- Aplicación de un filtro de paso bajo basado en la media
- Recortado de la imagen marcada
- Escalado de la imagen marcada
- Rotación de la imagen marcada

7.3. Pruebas individuales

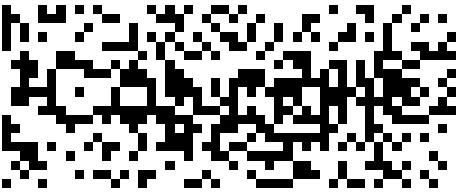
7.3.1. Algoritmos de técnicas basadas en correlación

7.3.1.1. Pruebas de parámetros

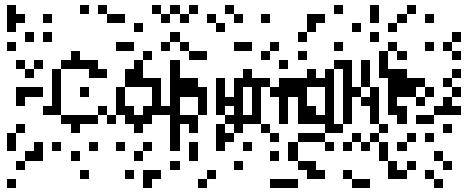
Este algoritmo utilizará la imagen de Lena de dimensiones 512x512, para la marca de agua se usará el logotipo del copyright normal, como clave para inicializar la semilla se utilizará la imagen key y de tamaño de bloque 16x16. Las pruebas se harán para diferentes valores de k, que es el factor de ganancia.

Como datos que se recogerán se encuentran las imágenes marcadas, las marcas extraídas, los valores de PSNR, el número de errores de la marca extraída y los tiempos de computación en la inserción y extracción.

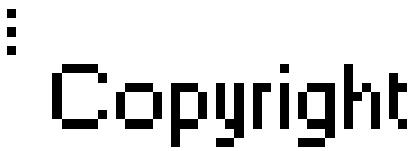
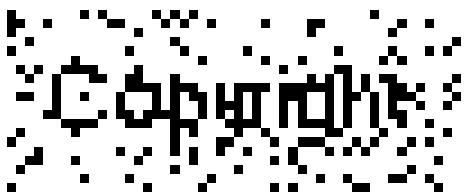
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Parámetros:	k = 5
	
Imagen Original	Imagen Marcada
	PSNR: 47.0562 dB
	Calidad: Muy Buena
: Copyright	
Marca Original	Marca Extraída
	Número de Errores: 304 de 1000
Tabla 2: Inserción y extracción por el algoritmo de técnicas basadas en correlación, k = 5	

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Parámetros:	$k = 15$
	
Imagen Original	Imagen Marcada
	PSNR: 37.5138 dB
	Calidad: Regular
: Copyright	
Marca Original	Marca Extraída
	Número de Errores: 180 de 1000
Tabla 3: Inserción y extracción por el algoritmo de técnicas basadas en correlación, $k = 15$	

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Parámetros:	$k = 25$
	
Imagen Original	Imagen Marcada
	PSNR: 33.0768 dB
	Calidad: Mala
	
Marca Original	Marca Extraída
	Número de Errores: 131 de 1000
Tabla 4: Inserción y extracción por el algoritmo de técnicas basadas en correlación, $k = 25$	

Prueba con el algoritmo de extracción modificado.

Parámetros:	$k = 1$
	
Imagen Original	Imagen Marcada
PSNR:	61.0356 dB
Calidad:	Muy Buena
Marca Original	Marca Extraída (Modificado)
	Número de Errores: 0 de 1000

Tabla 5: Inserción y extracción modificada por el algoritmo de técnicas basadas en correlación, $k = 1$

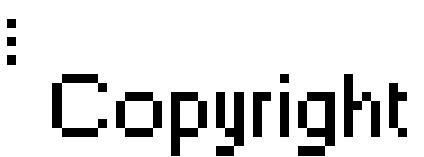
Parámetros:	k = 5
	
Imagen Original	Imagen Marcada
	PSNR: 47.0562 dB
	Calidad: Muy Buena
: Copyright	: Copyright
Marca Original	Marca Extraída (Modificado)
	Número de Errores: 0 de 1000
Tabla 6: Inserción y extracción modificada por el algoritmo de técnicas basadas en correlación, k = 5	

Parámetros:	$k = 25$
	
Imagen Original	Imagen Marcada
	PSNR: 33.0768 dB
	Calidad: Mala
: Copyright	: Copyright
Marca Original	Marca Extraída (Modificado)
	Número de Errores: 0 de 1000
Tabla 7: Inserción y extracción modificada por el algoritmo de técnicas basadas en correlación, $k = 25$	

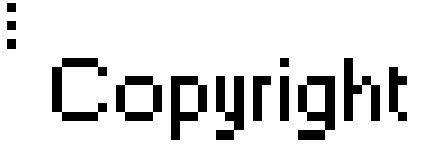
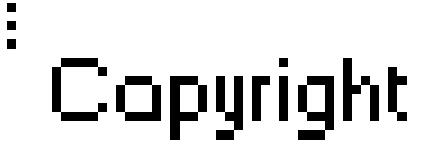
7.3.1.2. Pruebas de ataques

Se hará una prueba de cada ataque de los definidos, en ellas se comprobará la resistencia del algoritmo a los mismos, para estas pruebas se cogerá la imagen marcada que mejores resultados dio en las pruebas anteriores. Las extracciones se han realizado con el algoritmo de extracción modificado ya que ofrece mejores resultados que el de los autores.

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del Ataque:	Compresión JPEG, 60%
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 71 de 1000
Tabla 8: Ataque compresión JPEG 60% algoritmo técnicas basadas en correlación	

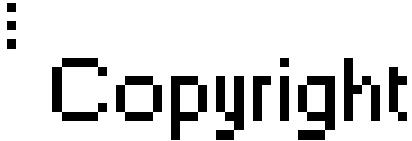
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del Ataque:	Inserción de ruido gaussiano
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 1 de 1000
Tabla 9: Ataque inserción ruido gaussiano algoritmo técnicas basadas en correlación	

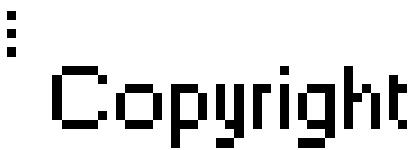
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del Ataque:	Aplicación de un filtro de paso bajo basado en la media
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
Número de Errores:	105 de 1000
Tabla 10: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo técnicas basadas en correlación	

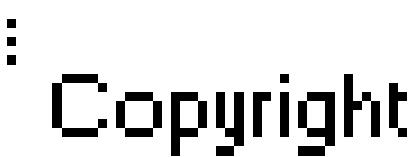
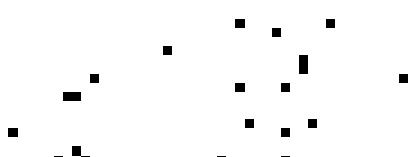
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del Ataque:	Recortado de la imagen marcada
	
Imagen Marcada	Imagen Atacada
	Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben tener las mismas dimensiones.
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: No Necesario
Tabla 11: Ataque recortado de la imagen marcada algoritmo técnicas basadas en correlación	

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del Ataque:	Escalado de la imagen marcada, doble tamaño
	
Imagen Marcada	Imagen Atacada
	Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben tener las mismas dimensiones.
Marca Extraída	Marca Extraída del Ataque
Número de Errores:	No Necesario
Tabla 12: Ataque escalado de la imagen marcada algoritmo técnicas basadas en correlación	

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del Ataque:	Rotación de la imagen marcada, 90°
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 124 de 1000
Tabla 13: Ataque rotación de la imagen marcada algoritmo técnicas basadas en correlación	

7.3.1.3. Pruebas de tiempos de ejecución

Por último, se comprobará la eficiencia del algoritmo en tiempo de computación, para ello se cogerá una media de diez ejecuciones del mismo.

Algoritmo:	Técnicas basadas en correlación	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	0.2496	0.3744
Ejecución 2	0.2496	0.2808
Ejecución 3	0.2184	0.3744
Ejecución 4	0.2496	0.3432
Ejecución 5	0.2496	0.3432
Ejecución 6	0.2184	0.3276
Ejecución 7	0.2496	0.2808
Ejecución 8	0.2496	0.2964
Ejecución 9	0.2496	0.2652
Ejecución 10	0.2496	0.2964
Media	0.2433 seg.	0.3182 seg.

Tabla 14: Tiempos de ejecución del algoritmo de técnicas basadas en correlación

Algoritmo:	Técnicas basadas en correlación (modificado)	
	Inserción de la marca de agua	Extracción de la marca de agua (modificado)
Ejecución 1	0.2496	0.2028
Ejecución 2	0.2496	0.1560
Ejecución 3	0.2184	0.1716
Ejecución 4	0.2496	0.1716
Ejecución 5	0.2496	0.2028
Ejecución 6	0.2184	0.1248
Ejecución 7	0.2496	0.0936
Ejecución 8	0.2496	0.1716
Ejecución 9	0.2496	0.1716
Ejecución 10	0.2496	0.1872
Media	0.2433 seg.	0.1653 seg.

Tabla 15: Tiempos de ejecución del algoritmo de técnicas basadas en correlación (modificado)

7.3.1.4. Conclusiones individuales

Después del estudio individual que se acaba de realizar se puede concluir que:

- En el algoritmo sin modificar, un valor del parámetro k bajo ofrecerá una **gran calidad de imagen** pero una extracción de la **marca de agua mala**. En cambio, un valor alto, ofrecerá una **mala calidad de imagen** pero una obtención de la **marca más buena**, aún así, la extracción de la **marca** que se consigue es **muy mala**.

- En el algoritmo modificado, un valor del parámetro **k bajo** ofrecerá una **calidad de imagen muy buena** y la extracción de la **marca** de agua será **perfecta**; un valor de **k alto** hará que la **calidad de la imagen decrezca** pero que la extracción de la **marca** siga siendo **perfecta**.
- Se puede concluir que el método modificado, aunque necesita la imagen original, es mejor que el método no modificado.
- En cuanto a los ataques:
 - Es resistente a: inserción de ruido gaussiano.
 - Es vulnerable a: compresión JPEG al 60%, filtro de paso bajo de la media, rotación.
 - No es aplicable a: recortado, escalado.

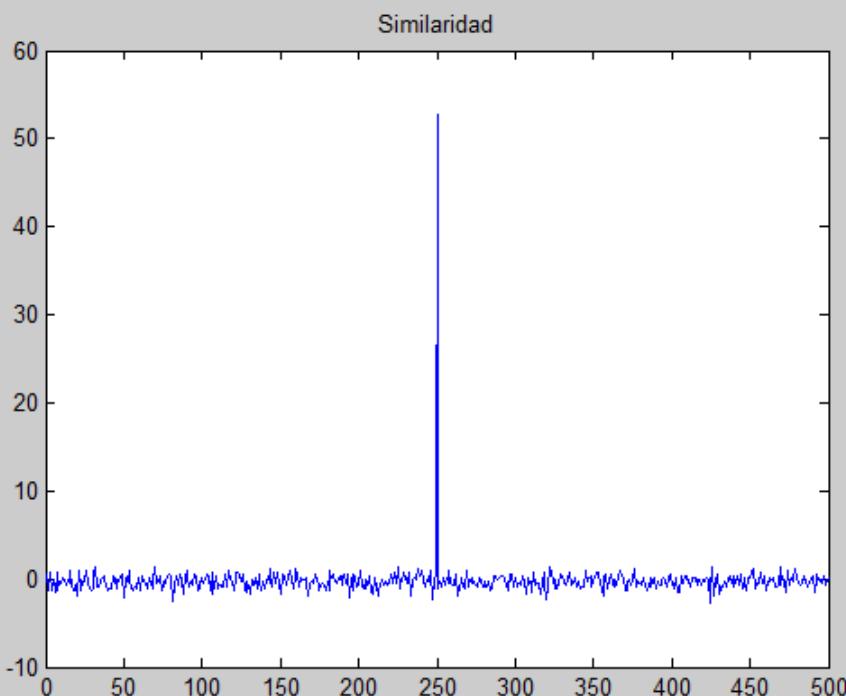
7.3.2. Algoritmo de Cox

7.3.2.1. Pruebas de parámetros

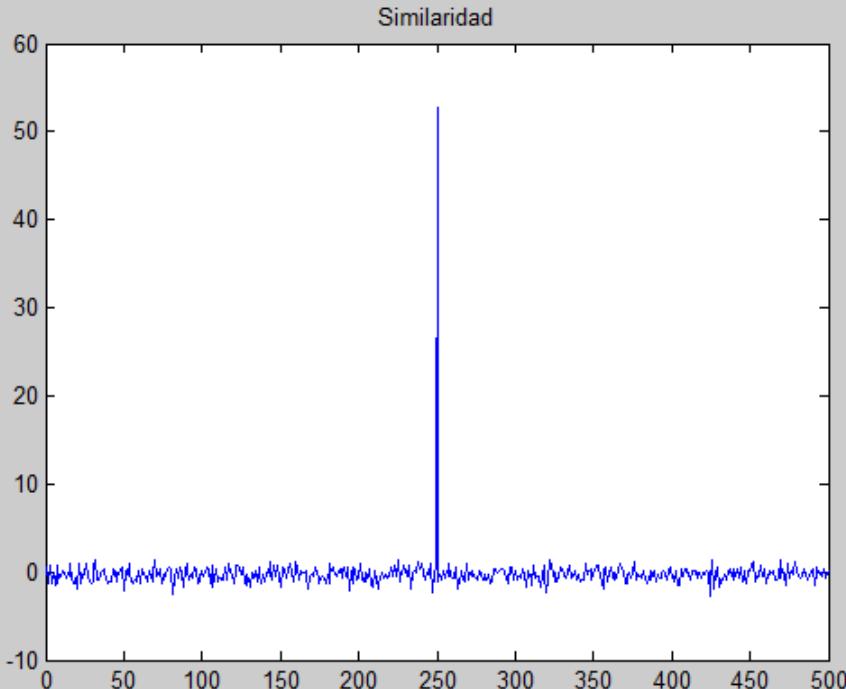
Este algoritmo utilizará la imagen de Lena de dimensiones 512x512, para la marca de agua se usará una secuencia pseudoaleatoria de tamaño 2500 y como clave para inicializar la semilla será el valor 10. Las pruebas se harán para diferentes valores de alpha.

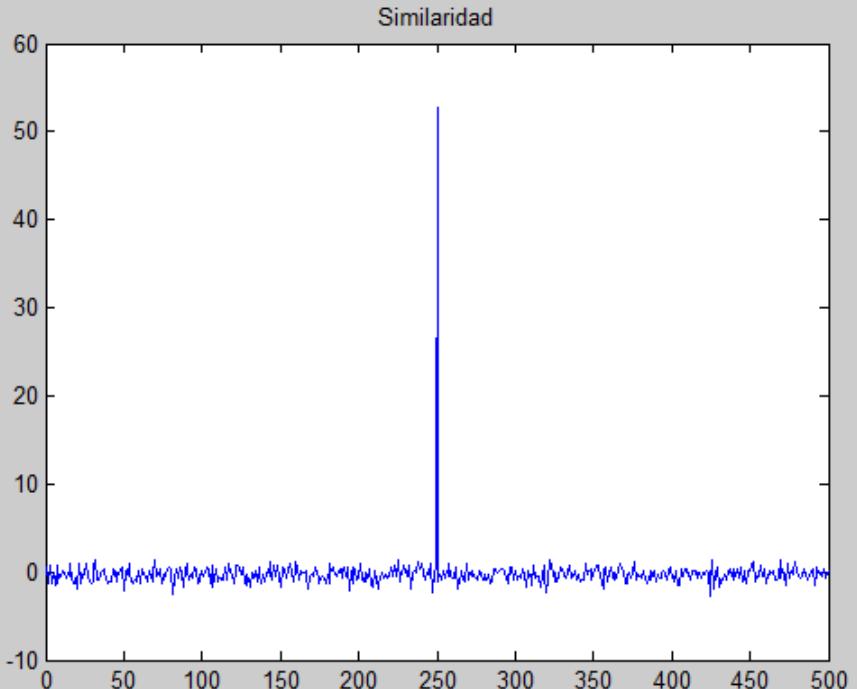
Como datos que se recogerán se encuentran la imagen marcada, la gráfica de similaridad, los valores de PSNR y los tiempos de computación en la inserción y extracción.

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Parámetros:	alpha = 0.05														
															
Imagen Original	Imagen Marcada														
	PSNR: 39.6942 dB														
	Calidad: Buena														
 <p>Similaridad</p> <table border="1"> <tr> <td>0</td> <td>50</td> </tr> <tr> <td>50</td> <td>10</td> </tr> <tr> <td>10</td> <td>20</td> </tr> <tr> <td>20</td> <td>30</td> </tr> <tr> <td>30</td> <td>40</td> </tr> <tr> <td>40</td> <td>50</td> </tr> <tr> <td>50</td> <td>60</td> </tr> </table>		0	50	50	10	10	20	20	30	30	40	40	50	50	60
0	50														
50	10														
10	20														
20	30														
30	40														
40	50														
50	60														
Similaridad															
Número de Errores:	No Necesario														
Tabla 16: Inserción y extracción por el algoritmo de Cox, alpha = 0.05															

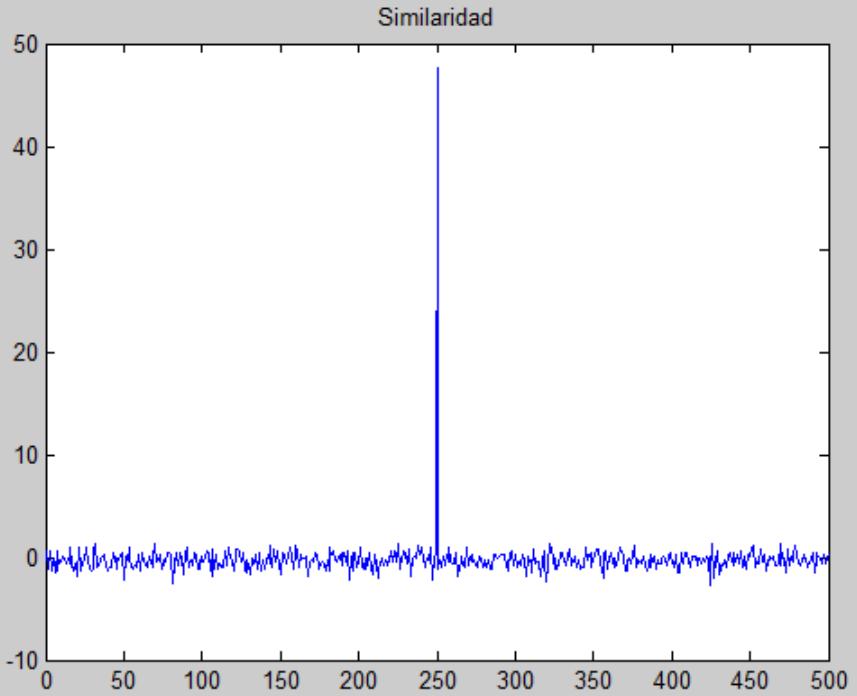
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Parámetros:	alpha = 0.1																						
																							
Imagen Original	Imagen Marcada																						
	PSNR: 33.7133 dB																						
	Calidad: Buena																						
 <p>Similaridad</p> <table border="1"> <tr> <td>0</td> <td>50</td> <td>100</td> <td>150</td> <td>200</td> <td>250</td> <td>300</td> <td>350</td> <td>400</td> <td>450</td> <td>500</td> </tr> <tr> <td>-10</td> <td>-5</td> <td>0</td> <td>5</td> <td>10</td> <td>15</td> <td>20</td> <td>25</td> <td>30</td> <td>35</td> <td>40</td> </tr> </table>		0	50	100	150	200	250	300	350	400	450	500	-10	-5	0	5	10	15	20	25	30	35	40
0	50	100	150	200	250	300	350	400	450	500													
-10	-5	0	5	10	15	20	25	30	35	40													
<p>Similaridad</p> <table border="1"> <tr> <td>Número de Errores:</td> <td>No Necesario</td> </tr> </table>		Número de Errores:	No Necesario																				
Número de Errores:	No Necesario																						
Tabla 17: Inserción y extracción por el algoritmo de Cox, alpha = 0.1																							

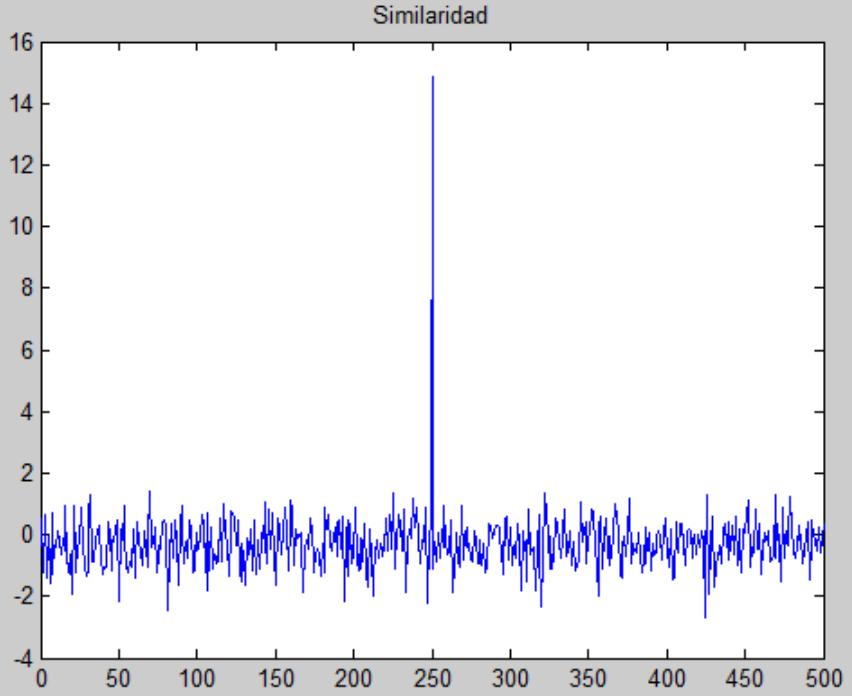
Parámetros:	alpha = 0.5
	
Imagen Original	Imagen Marcada
	PSNR: 19.7854 dB
	Calidad: Muy Mala
 <p>The graph plots 'Similaridad' (Similarity) against an index from 0 to 500. The y-axis ranges from -10 to 60. The x-axis ranges from 0 to 500. A blue line shows a noisy baseline around zero, with a single sharp vertical blue line reaching up to approximately 53 at index 250.</p>	
Similaridad	
Número de Errores:	No Necesario
Tabla 18: Inserción y extracción por el algoritmo de Cox, alpha = 0.5	

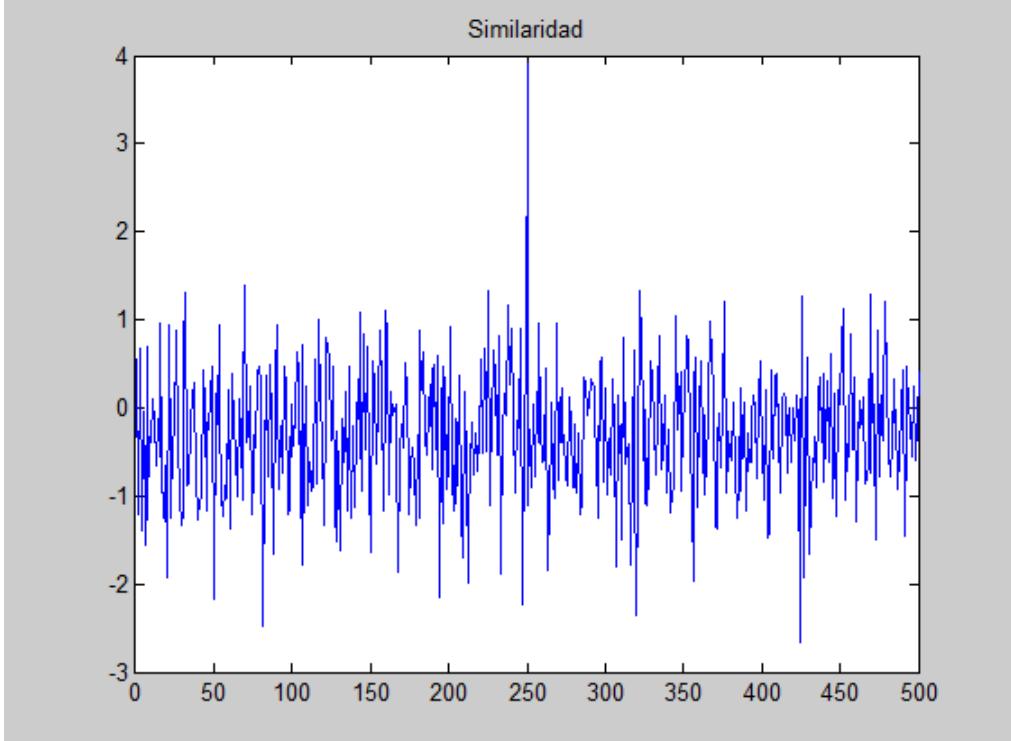
7.3.2.2. Pruebas de ataques

Se hará una prueba de cada ataque de los definidos, en ellas se comprobará la resistencia del algoritmo a los mismos, para estas pruebas se cogerá la imagen marcada que mejores resultados dio en las pruebas anteriores.

Nombre del ataque:	Compresión JPEG, 60%																		
																			
Imagen Marcada	Imagen Atacada																		
 <p>Similaridad</p> <table border="1"> <tr> <td>50</td> <td>40</td> <td>30</td> <td>20</td> <td>10</td> <td>0</td> <td>-10</td> </tr> <tr> <td>0</td> <td>50</td> <td>100</td> <td>150</td> <td>200</td> <td>250</td> <td>300</td> <td>350</td> <td>400</td> <td>450</td> <td>500</td> </tr> </table>		50	40	30	20	10	0	-10	0	50	100	150	200	250	300	350	400	450	500
50	40	30	20	10	0	-10													
0	50	100	150	200	250	300	350	400	450	500									
Número de Errores:	No Necesario																		
Tabla 19: Ataque compresión JPEG 60% algoritmo de Cox																			

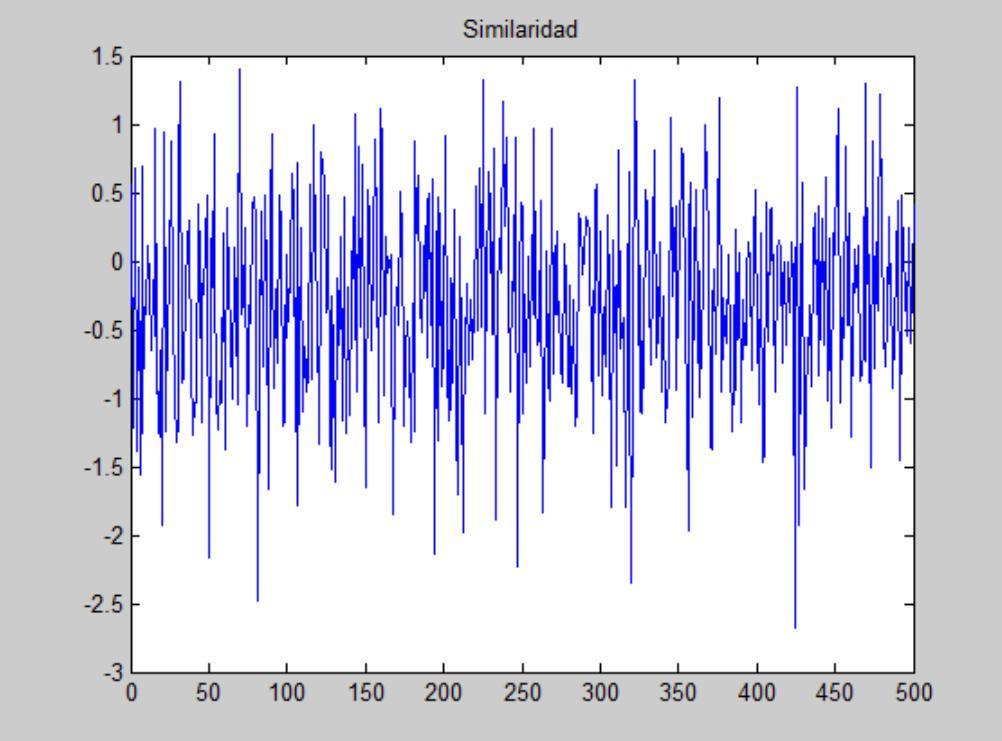
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

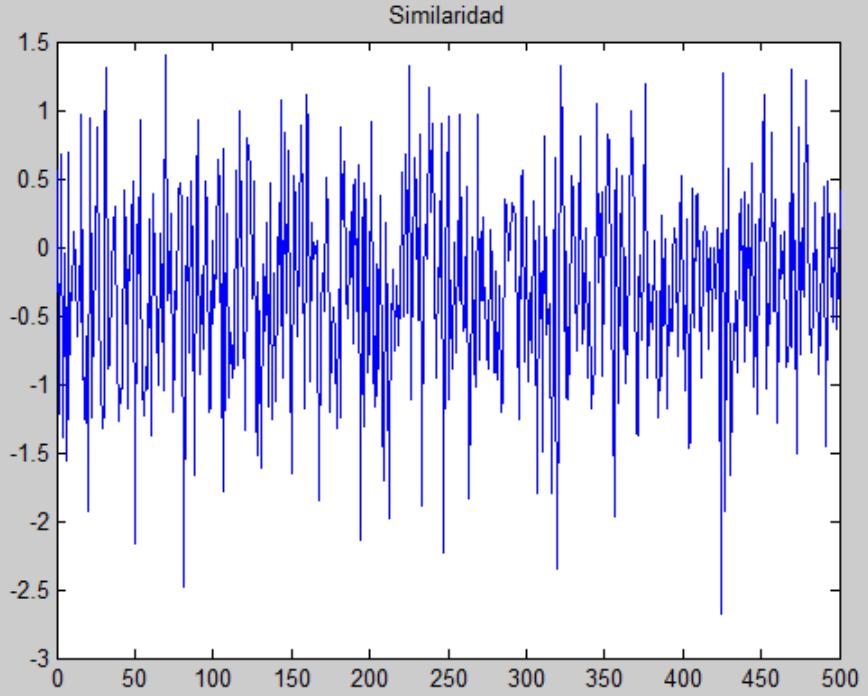
Nombre del ataque:	Inserción de ruido gaussiano
	
Imagen Marcada	Imagen Atacada
	
Número de Errores:	No Necesario
Tabla 20: Ataque inserción ruido gaussiano algoritmo de Cox	

Nombre del ataque:	Aplicación de un filtro de paso bajo basado en la media
	
Imagen Marcada	Imagen Atacada
	
Número de Errores:	No Necesario
Tabla 21: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo de Cox	

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del ataque:	Recortado de la imagen marcada
	
Imagen Marcada	Imagen Atacada
Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen atacada debe tener al menos las mismas dimensiones que la imagen marcada.	
Similaridad	
Número de Errores:	No Necesario
Tabla 22: Ataque recortado de la imagen marcada algoritmo de Cox	

Nombre del ataque:	Escalado de la imagen marcada, doble tamaño
	
Imagen Marcada	Imagen Atacada
	
Similaridad	
Número de Errores:	No Necesario
Tabla 23: Ataque escalado de la imagen marcada algoritmo de Cox	

Nombre del ataque:	Rotación de la imagen marcada, 90°
	
Imagen Marcada	Imagen Atacada
	
Número de Errores:	No Necesario
Tabla 24: Ataque rotación de la imagen marcada algoritmo de Cox	

7.3.2.3. Pruebas de tiempos de ejecución

Por último, se comprobará la eficiencia del algoritmo en tiempo de computación, para ello se cogerá una media de diez ejecuciones del mismo.

Algoritmo:	Cox	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	0.5616	0.4056
Ejecución 2	0.5460	0.3588
Ejecución 3	0.5772	0.3432
Ejecución 4	0.4524	0.4368
Ejecución 5	0.4992	0.3744
Ejecución 6	0.5148	0.3744
Ejecución 7	0.5616	0.3900
Ejecución 8	0.4992	0.3900
Ejecución 9	0.5304	0.3588
Ejecución 10	0.4992	0.3432
Media	0.5241 seg.	0.3775 seg.

Tabla 25: Tiempos de ejecución del algoritmo de Cox

7.3.2.4. Conclusiones individuales

Después del estudio individual que se acaba de realizar se puede concluir que:

- Este algoritmo consigue **recuperar la marca** bastante **bien**, dependiendo del valor del parámetro **alpha** variará la calidad de la imagen marcada, cuanto más **pequeño** sea el valor de alpha **más calidad** tendrá la imagen marcada, en cambio, cuanto **más alto** sea dicho valor, **menor calidad** tendrá.
- En cuanto a los ataques:
 - Es resistente a: compresión JPEG al 60%, inserción de ruido gaussiano, filtro de paso bajo de la media.
 - Es vulnerable a: escalado, rotación.
 - No es aplicable a: recortado.

7.3.3. Algoritmos basados en la DCT

7.3.3.1. Primer Método: DCT

7.3.3.1.1. Pruebas de parámetros

Este algoritmo utilizará la imagen de Lena de dimensiones 512x512, para la marca de agua se usará el logotipo del copyright normal y de tamaño de bloque 8x8. Las pruebas se harán para diferentes valores de k, que es el factor de ganancia.

Como datos que se recogerán se encuentran la imagen marcada, la marca extraída, los valores de PSNR, el número de errores de la marca extraída y los tiempos de computación en la inserción y extracción.

Parámetros:	$k = 10$
	
Imagen Original	Imagen Marcada
	PSNR: 43.2654 dB
	Calidad: Muy Buena
	
Marca Original	Marca Extraída
	Número de Errores: 0 de 1000
Tabla 26: Inserción y extracción por el algoritmo DCT, $k = 10$	

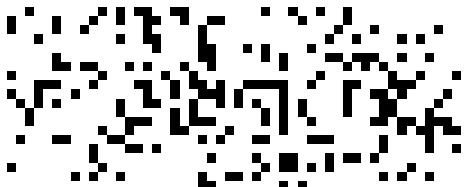
Parámetros:	k = 50
	
Imagen Original	Imagen Marcada
	PSNR: 33.8132 dB
	Calidad: Regular
: Copyright	: Copyright
Marca Original	Marca Extraída
	Número de Errores: 0 de 1000
Tabla 27: Inserción y extracción por el algoritmo DCT, k = 50	

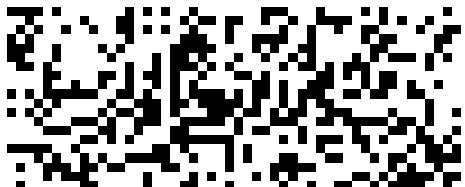
Parámetros:	$k = 100$
	
Imagen Original	Imagen Marcada
	PSNR: 28.3601 dB
	Calidad: Mala
: Copyright	: Copyright
Marca Original	Marca Extraída
	Número de Errores: 0 de 1000
Tabla 28: Inserción y extracción por el algoritmo DCT, $k = 100$	

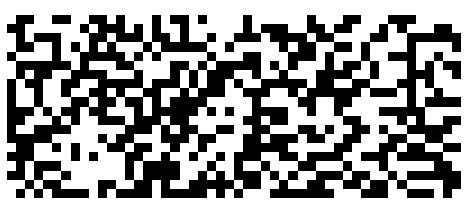
7.3.3.1.2. Pruebas de ataques

Se hará una prueba de cada ataque de los definidos, en ellas se comprobará la resistencia del algoritmo a los mismos, para estas pruebas se cogerá la imagen marcada que mejores resultados dio en las pruebas anteriores.

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

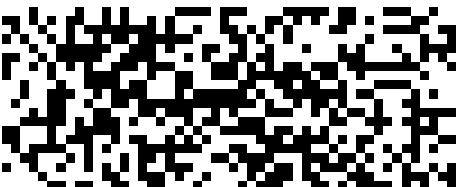
Nombre del Ataque:	Compresión JPEG, 60%
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 212 de 1000
Tabla 29: Ataque compresión JPEG 60% algoritmo DCT	

Nombre del Ataque:	Inserción de ruido gaussiano
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 353 de 1000
Tabla 30: Ataque inserción ruido gaussiano algoritmo DCT	

Nombre del Ataque:	Aplicación de un filtro de paso bajo basado en la media
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 504 de 1000
Tabla 31: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo DCT	

Nombre del Ataque:	Recortado de la imagen marcada
	
Imagen Marcada	Imagen Atacada
	Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben de tener al menos, las mismas dimensiones.
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: No Necesario
Tabla 32: Ataque recortado de la imagen marcada algoritmo DCT	

Nombre del Ataque:	Escalado de la imagen marcada, doble tamaño
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 484 de 1000
Tabla 33: Ataque escalado de la imagen marcada algoritmo DCT	

Nombre del Ataque:	Rotación de la imagen marcada, 90°
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 475 de 1000
Tabla 34: Ataque rotación de la imagen marcada algoritmo DCT	

7.3.3.1.3. Pruebas de tiempos de ejecución

Por último, se comprobará la eficiencia del algoritmo en tiempo de computación, para ello se cogerá una media de diez ejecuciones del mismo.

Algoritmo:	DCT	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	1.5600	0.7800
Ejecución 2	1.6224	0.8424
Ejecución 3	1.5756	0.8736
Ejecución 4	1.5912	0.8268
Ejecución 5	1.5600	0.8112
Ejecución 6	1.5600	0.7956
Ejecución 7	1.6380	0.8424
Ejecución 8	1.5600	0.8112
Ejecución 9	1.5756	0.8112
Ejecución 10	1.5132	0.8112
Media	1.5756 seg.	0.8205 seg.

Tabla 35: Tiempos de ejecución del algoritmo DCT

7.3.3.1.4. Conclusiones individuales

Después del estudio individual que se acaba de realizar se puede concluir que:

- En este algoritmo tiene una **extracción de la marca de agua muy buena**, en las pruebas ha conseguido extraerla a la perfección en todos los casos, dependiendo del valor de k se obtendrá una calidad de imagen u otra, un valor de **k bajo** dará una **calidad de imagen muy buena**, en cambio un valor de **k muy alto** dará una **calidad de imagen mala**.
- En cuanto a los ataques:
 - Es resistente a: .
 - Es vulnerable a: compresión JPEG al 60%, inserción de ruido gaussiano, filtro de paso bajo de la media, escalado, rotación.
 - No es aplicable a: recortado.

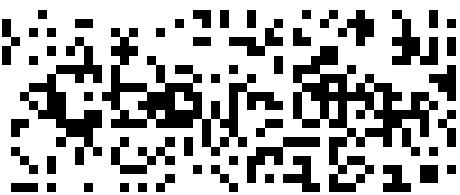
7.3.3.2. Segundo Método: DCT y correlación

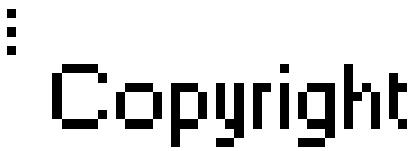
7.3.3.2.1. Pruebas de parámetros

Este algoritmo utilizará la imagen de Lena de dimensiones 512x512, para la marca de agua se usará el logotipo del copyright normal, como clave para inicializar la semilla se utilizará la imagen key y de tamaño de bloque 8x8. Las pruebas se harán para diferentes valores de k, que es el factor de ganancia.

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Como datos que se recogerán se encuentran la imagen marcada, la marca extraída, los valores de PSNR, el número de errores de la marca extraída y los tiempos de computación en la inserción y extracción.

Parámetros:	k = 1
	
Imagen Original	Imagen Marcada
	PSNR: 56.8158 dB
	Calidad: Muy Buena
	
Marca Original	Marca Extraída
	Número de Errores: 278 de 1000
Tabla 36: Inserción y extracción por el algoritmo DCT y correlación, k = 1	

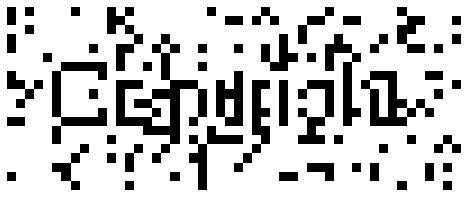
Parámetros:	k = 5
	
Imagen Original	Imagen Marcada
	PSNR: 42.8364 dB
	Calidad: Muy Buena
	
Marca Original	Marca Extraída
	Número de Errores: 27 de 1000
Tabla 37: Inserción y extracción por el algoritmo DCT y correlación, k = 5	

Parámetros:	$k = 10$
	
Imagen Original	Imagen Marcada
	PSNR: 36.8158 dB
	Calidad: Regular
	
Marca Original	Marca Extraída
	Número de Errores: 5 de 1000
Tabla 38: Inserción y extracción por el algoritmo DCT y correlación, $k = 10$	

7.3.3.2.2. Pruebas de ataques

Se hará una prueba de cada ataque de los definidos, en ellas se comprobará la resistencia del algoritmo a los mismos, para estas pruebas se cogerá la imagen marcada que mejores resultados dio en las pruebas anteriores.

Nombre del Ataque:	Compresión JPEG, 60%
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 12 de 1000
Tabla 39: Ataque compresión JPEG 60% algoritmo DCT y correlación	

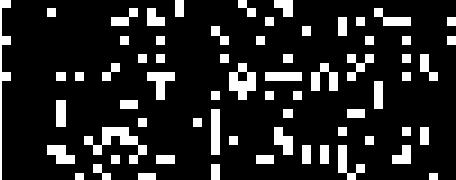
Nombre del Ataque:	Inserción de ruido gaussiano
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
Número de Errores: 169 de 1000	
Tabla 40: Ataque inserción ruido gaussiano algoritmo DCT y correlación	

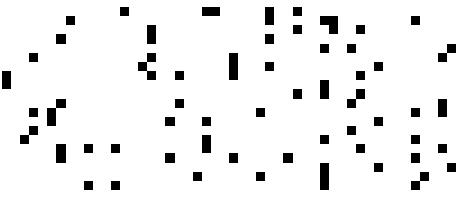
Nombre del Ataque:	Aplicación de un filtro de paso bajo basado en la media
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 550 de 1000

Tabla 41: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo DCT y correlación

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del Ataque:	Recortado de la imagen marcada
	
Imagen Marcada	Imagen Atacada
	Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben de tener al menos, las mismas dimensiones.
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: No Necesario
Tabla 42: Ataque recortado de la imagen marcada algoritmo DCT y correlación	

Nombre del Ataque:	Escalado de la imagen marcada, doble tamaño
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 778 de 1000
Tabla 43: Ataque escalado de la imagen marcada algoritmo DCT y correlación	

Nombre del Ataque:	Rotación de la imagen marcada, 90°
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
Número de Errores: 174 de 1000	
Tabla 44: Ataque rotación de la imagen marcada algoritmo DCT y correlación	

7.3.3.2.3. Pruebas de tiempos de ejecución

Por último, se comprobará la eficiencia del algoritmo en tiempo de computación, para ello se cogerá una media de diez ejecuciones del mismo.

Algoritmo:	DCT y correlación	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	1.5288	1.2948
Ejecución 2	1.5444	1.4040
Ejecución 3	1.4976	1.3572
Ejecución 4	1.4976	1.3260
Ejecución 5	1.4352	1.4040
Ejecución 6	1.5912	1.3260
Ejecución 7	1.4664	1.4352
Ejecución 8	1.5132	1.3416
Ejecución 9	1.5132	1.3416
Ejecución 10	1.5132	1.3260
Media	1.5101 seg.	1.3556 seg.

Tabla 45: Tiempos de ejecución del algoritmo DCT y correlación

7.3.3.2.4. Conclusiones individuales

Después del estudio individual que se acaba de realizar se puede concluir que:

- En este algoritmo la **extracción** de la **marca** de agua **no** llega a ser **muy buena**, en función del parámetro k variará la calidad de la imagen, un valor de **k bajo** hará que la imagen marcada tenga una **calidad** de imagen **muy buena** pero una **extracción** de la marca de agua **bastante mala**, con **muchos errores**, en cambio, un valor de **k alto** hará que la **calidad** de la imagen marcada sea **regular** pero la **extracción** de la marca será **mejor**.
- En cuanto a los ataques:
 - Es resistente a: compresión JPEG al 60%.
 - Es vulnerable a: inserción de ruido gaussiano, filtro de paso bajo de la media, escalado, rotación.
 - No es aplicable a: recortado.

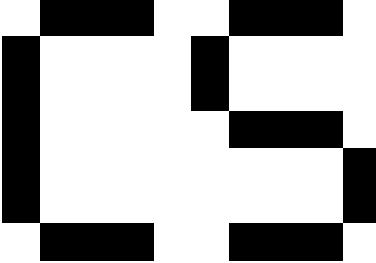
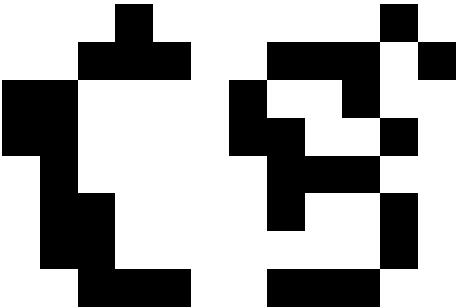
7.3.4. Algoritmo basado en CDMA

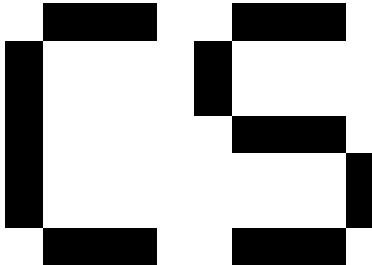
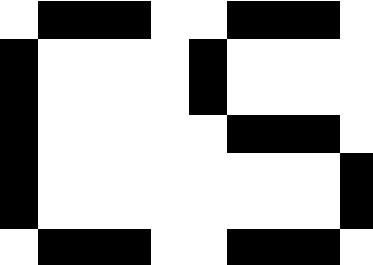
7.3.4.1. Pruebas de parámetros

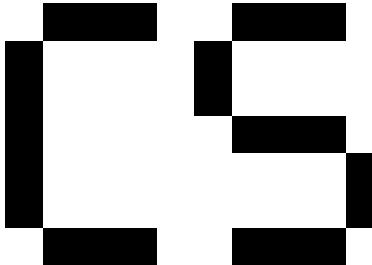
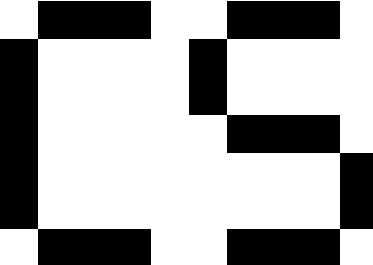
Este algoritmo utilizará la imagen de Lena de dimensiones 512x512, para la marca de agua se usará el logotipo del copyright pequeño y como clave para inicializar la semilla se utilizará la imagen key. Las pruebas se harán para diferentes valores de k, que es el factor de ganancia.

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Como datos que se recogerán se encuentran la imagen marcada, la marca extraída, los valores de PSNR, el número de errores de la marca extraída y los tiempos de computación en la inserción y extracción.

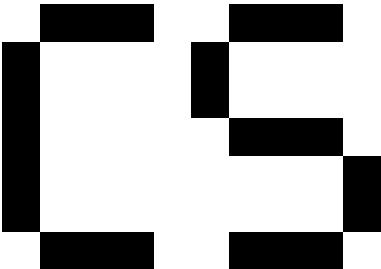
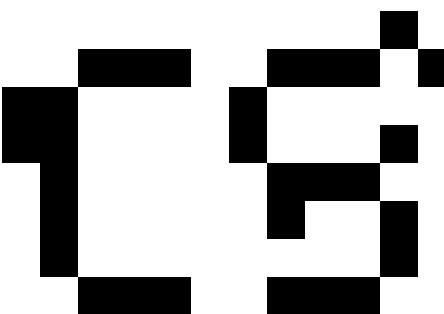
Parámetros:	$k = 0.5$
	
Imagen Original	Imagen Marcada
	PSNR: 43.0924 dB
	Calidad: Muy Buena
	
Marca Original	Marca Extraída
	Número de Errores: 11 de 108
Tabla 46: Inserción y extracción por el algoritmo CDMA, $k = 0.5$	

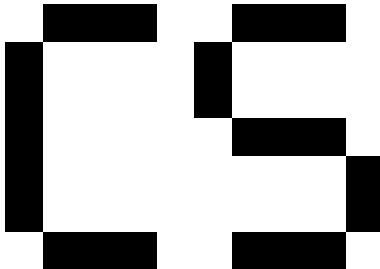
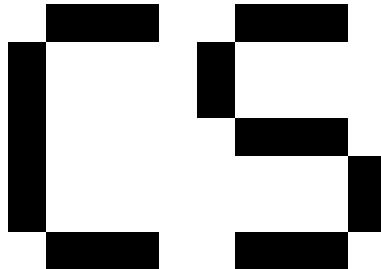
Parámetros:	$k = 2$
	
Imagen Original	Imagen Marcada
	PSNR: 31.0512 dB
	Calidad: Regular
	
Marca Original	Marca Extraída
	Número de Errores: 0 de 108
Tabla 47: Inserción y extracción por el algoritmo CDMA, $k = 2$	

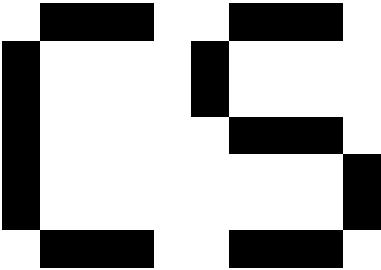
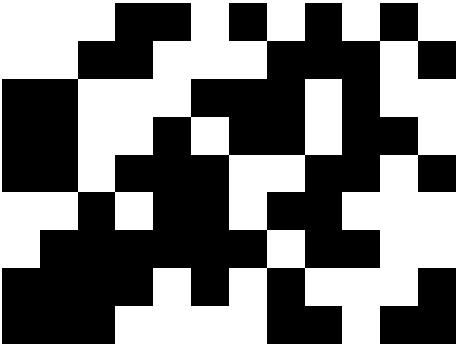
Parámetros:	$k = 5$
	
Imagen Original	Imagen Marcada
	PSNR: 23.0924 dB
	Calidad: Muy Mala
	
Marca Original	Marca Extraída
	Número de Errores: 0 de 108
Tabla 48: Inserción y extracción por el algoritmo CDMA, $k = 5$	

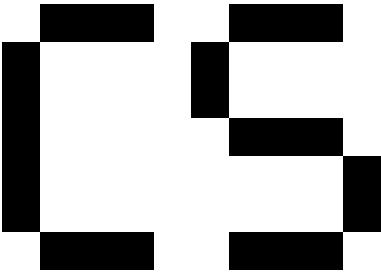
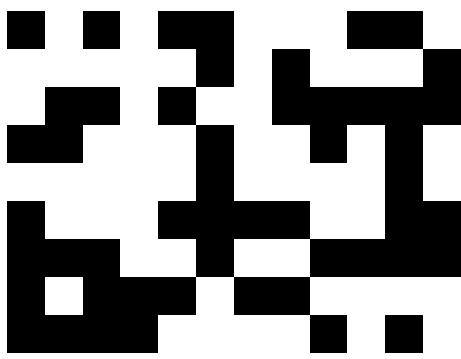
7.3.4.2. Pruebas de ataques

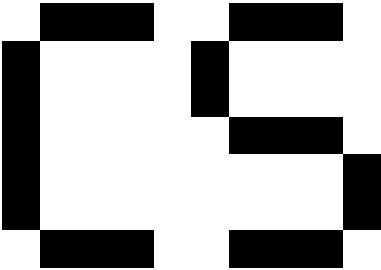
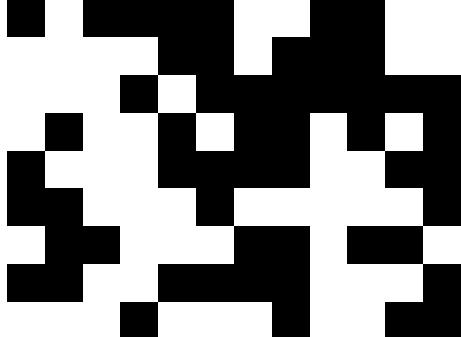
Se hará una prueba de cada ataque de los definidos, en ellas se comprobará la resistencia del algoritmo a los mismos, para estas pruebas se cogerá la imagen marcada que mejores resultados dio en las pruebas anteriores.

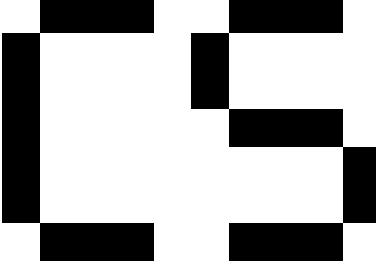
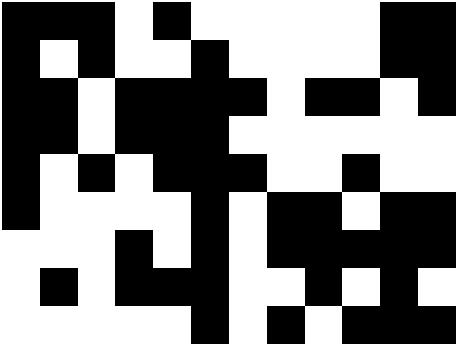
Nombre del Ataque:	Compresión JPEG, 60%
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 6 de 108
Tabla 49: Ataque compresión JPEG 60% algoritmo CDMA	

Nombre del Ataque:	Inserción de ruido gaussiano
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 0 de 108
Tabla 50: Ataque inserción ruido gaussiano algoritmo CDMA	

Nombre del Ataque:	Aplicación de un filtro de paso bajo basado en la media
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 51 de 108
Tabla 51: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo CDMA	

Nombre del Ataque:	Recortado de la imagen marcada
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
Número de Errores: 55 de 108	
Tabla 52: Ataque recortado de la imagen marcada algoritmo CDMA	

Nombre del Ataque:	Escalado de la imagen marcada, doble tamaño
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 52 de 108
Tabla 53: Ataque escalado de la imagen marcada algoritmo CDMA	

Nombre del Ataque:	Rotación de la imagen marcada, 90°
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
Número de Errores: 59 de 108	
Tabla 54: Ataque rotación de la imagen marcada algoritmo CDMA	

7.3.4.3. Pruebas de tiempos de ejecución

Por último, se comprobará la eficiencia del algoritmo en tiempo de computación, para ello se cogerá una media de diez ejecuciones del mismo.

Algoritmo:	CDMA	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	2.7300	4.7268
Ejecución 2	2.7924	4.7268
Ejecución 3	2.8080	4.6488
Ejecución 4	2.6364	4.7892
Ejecución 5	2.7300	4.7580
Ejecución 6	2.8236	4.6644
Ejecución 7	2.7456	4.5084
Ejecución 8	2.8080	4.7268
Ejecución 9	2.7768	4.6020
Ejecución 10	2.6832	4.6644
Media	2.7534 seg.	4.6816 seg.

Tabla 55: Tiempos de ejecución del algoritmo CDMA

7.3.4.4. Conclusiones individuales

Después del estudio individual que se acaba de realizar se puede concluir que:

- En este algoritmo la extracción de la marca de agua variará en función del parámetro k, es decir, que un valor de **k bajo** hará que la **calidad** de la imagen marcada sea **muy buena** pero la **extracción** de la marca de agua sea **regular**, en cambio con una valor de **k alto** la imagen marcada tendrá muy **mala calidad** y la **extracción** de la marca de agua será **perfecta**.
- En cuanto a los ataques:
 - Es resistente a: compresión JPEG al 60%, inserción de ruido gaussiano.
 - Es vulnerable a: filtro de paso bajo de la media, recortado, escalado, rotación.
 - No es aplicable a: .

7.3.5. Algoritmos basados en la DWT

7.3.5.1. Primer Método: CDMA en el dominio wavelet

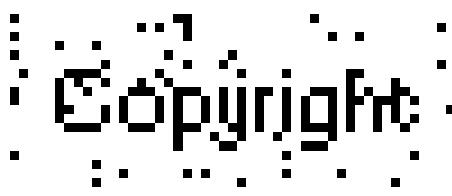
7.3.5.1.1. Pruebas de parámetros

Este algoritmo utilizará la imagen de Lena de dimensiones 512x512, para la marca de agua se usará el logotipo del copyright normal y como clave para inicializar la

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

semilla se utilizará la imagen key. Las pruebas se harán para diferentes valores de k, que es el factor de ganancia.

Como datos que se recogerán se encuentran la imagen marcada, la marca extraída, los valores de PSNR, el número de errores de la marca extraída y los tiempos de computación en la inserción y extracción.

Parámetros:	$k = 0.5$
	
Imagen Original	Imagen Marcada
	PSNR: 39.6745 dB
	Calidad: Buena
	
Marca Original	Marca Extraída
	Número de Errores: 49 de 1000
Tabla 56: Inserción y extracción por el algoritmo CDMA en el dominio wavelet, $k = 0.5$	

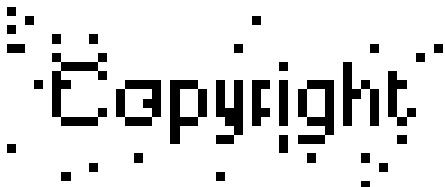
Parámetros:	k = 2
	
Imagen Original	Imagen Marcada
	PSNR: 27.6333 dB
	Calidad: Regular
: Copyright	: Copyright
Marca Original	Marca Extraída
	Número de Errores: 1 de 1000
Tabla 57: Inserción y extracción por el algoritmo CDMA en el dominio wavelet, k = 2	

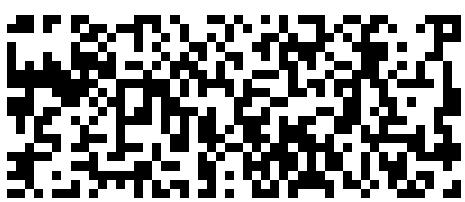
Parámetros:	$k = 5$
	
Imagen Original	Imagen Marcada
	PSNR: 19.6745 dB
	Calidad: Muy Mala
: Copyright	: Copyright
Marca Original	Marca Extraída
	Número de Errores: 1 de 1000
Tabla 58: Inserción y extracción por el algoritmo CDMA en el dominio wavelet, $k = 5$	

7.3.5.1.2. Pruebas de ataques

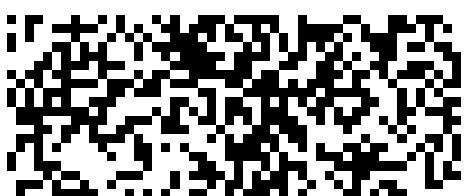
Se hará una prueba de cada ataque de los definidos, en ellas se comprobará la resistencia del algoritmo a los mismos, para estas pruebas se cogerá la imagen marcada que mejores resultados dio en las pruebas anteriores.

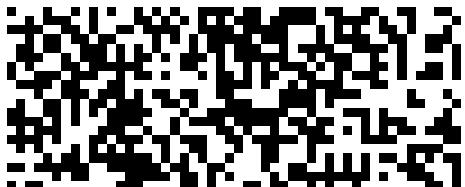
Nombre del Ataque:	Compresión JPEG, 60%
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 10 de 1000
Tabla 59: Ataque compresión JPEG 60% algoritmo CDMA en el dominio wavelet	

Nombre del Ataque:	Inserción de ruido gaussiano
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 30 de 1000
Tabla 60: Ataque inserción ruido gaussiano algoritmo CDMA en el dominio wavelet	

Nombre del Ataque:	Aplicación de un filtro de paso bajo basado en la media
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 507 de 1000
Tabla 61: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo CDMA en el dominio wavelet	

Nombre del Ataque:	Recortado de la imagen marcada
	
Imagen Marcada	Imagen Atacada
	Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben de tener al menos, las mismas dimensiones.
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: No Necesario
Tabla 62: Ataque recortado de la imagen marcada algoritmo CDMA en el dominio wavelet	

Nombre del Ataque:	Escalado de la imagen marcada, doble tamaño
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 509 de 1000
Tabla 63: Ataque escalado de la imagen marcada algoritmo CDMA en el dominio wavelet	

Nombre del Ataque:	Rotación de la imagen marcada, 90°
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 523 de 1000
Tabla 64: Ataque rotación de la imagen marcada algoritmo CDMA en el dominio wavelet	

7.3.5.1.3. Pruebas de tiempos de ejecución

Por último, se comprobará la eficiencia del algoritmo en tiempo de computación, para ello se cogerá una media de diez ejecuciones del mismo.

Algoritmo:	CDMA en el dominio wavelet	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	12.2773	19.7653
Ejecución 2	12.5113	21.6373
Ejecución 3	12.1213	21.6997
Ejecución 4	12.2617	21.5281
Ejecución 5	12.1369	21.6061
Ejecución 6	12.1993	21.4969
Ejecución 7	12.2929	21.7777
Ejecución 8	12.1213	21.5905
Ejecución 9	12.2929	21.2629
Ejecución 10	12.2773	21.3877
Media	12.2492 seg.	21.3752 seg.

Tabla 65: Tiempos de ejecución del algoritmo CDMA en el dominio wavelet

7.3.4.1.4. Conclusiones individuales

Después del estudio individual que se acaba de realizar se puede concluir que:

- En este algoritmo la extracción de la marca de agua variará con el valor de k, un valor de **k bajo** hará que la imagen marcada tenga una **calidad buena** pero una **extracción regular**, en cambio, con un valor de **k alto** la **calidad** de la imagen marcada será **muy mala** y la **extracción** de la marca de agua **casi perfecta**.
- En cuanto a los ataques:
 - Es resistente a: compresión JPEG al 60%, inserción de ruido gaussiano.
 - Es vulnerable a: filtro de paso bajo de la media, escalado, rotación.
 - No es aplicable a: recortado.

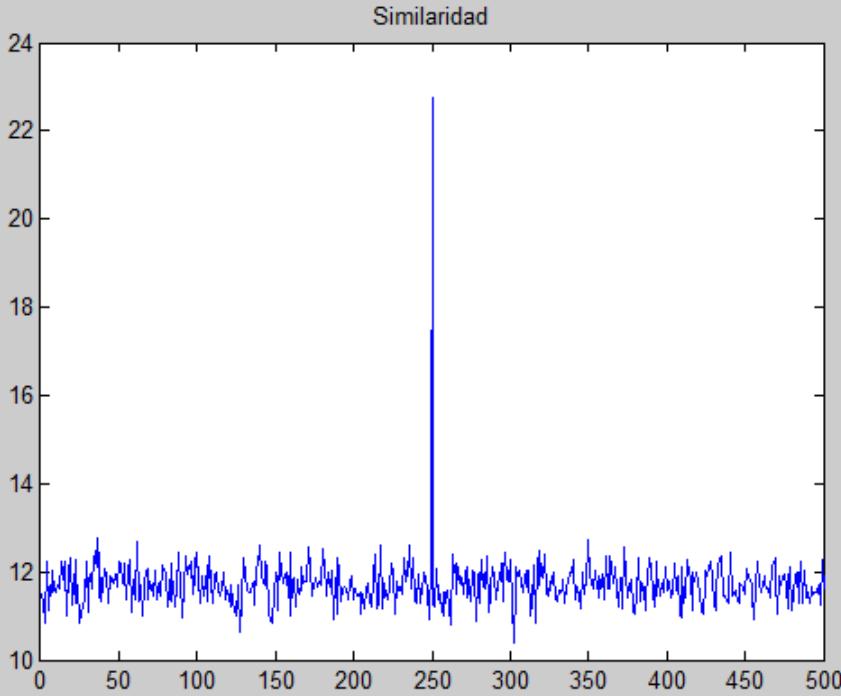
7.3.5.2. Segundo Método: DWT y la transformada de Haar

7.3.5.2.1. Pruebas de parámetros

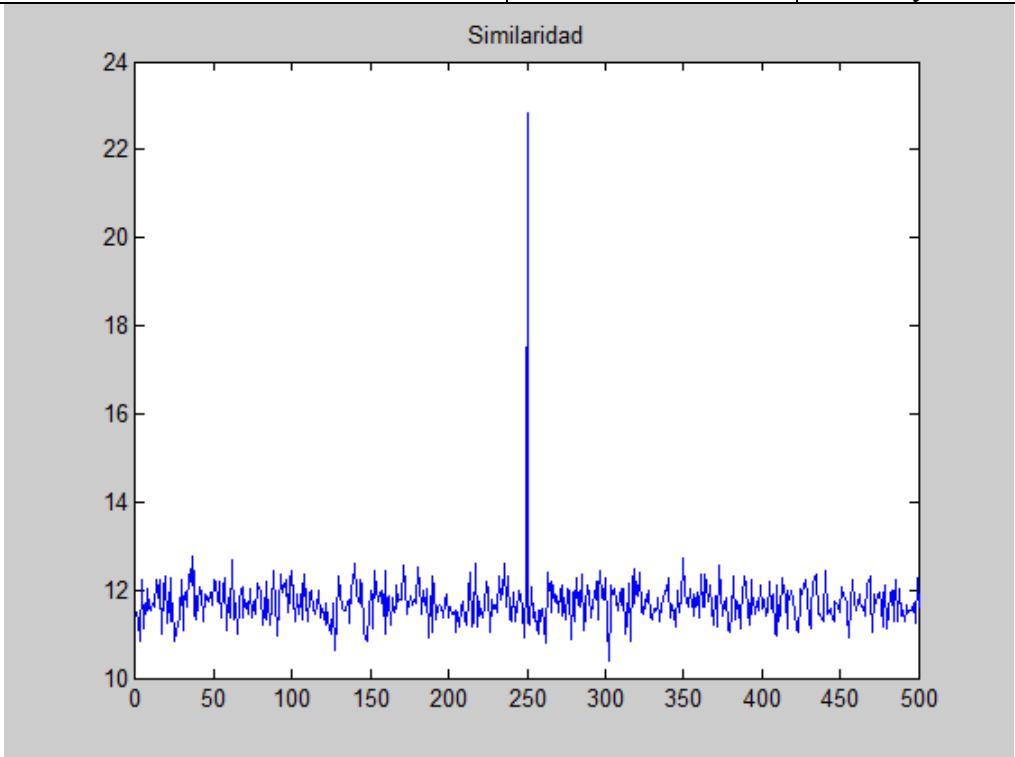
Este algoritmo utilizará la imagen de Lena de dimensiones 256x256, para la marca de agua se usará una secuencia pseudoaleatoria de 1000 elementos, como clave para inicializar la semilla se utilizará el valor 250 y como umbrales, s valdrá 2 y t valdrá 10. Las pruebas se harán para diferentes valores de s y t que son los que modifican el valor de alpha.

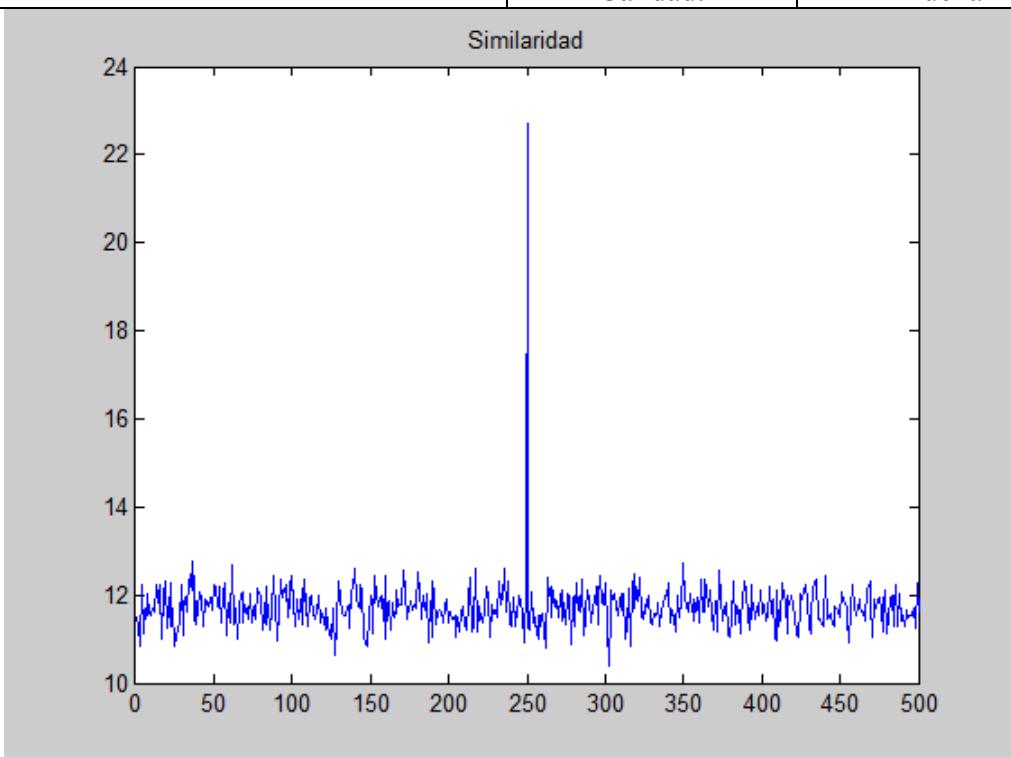
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Como datos que se recogerán se encuentran la imagen marcada, el gráfico de similaridad entre marcas, los valores de PSNR, el número de errores de la marca extraída y los tiempos de computación en la inserción y extracción.

Parámetros:	$s = 1, t = 5$
	
Imagen Original	Imagen Marcada
	PSNR: 54.0025 dB
	Calidad: Muy Buena
 <p>The graph plots 'Similaridad' (Similarity) on the y-axis (ranging from 10 to 24) against an unlabeled x-axis (ranging from 0 to 500). A blue line represents the similarity values, which are generally around 12, with significant noise. A single, extremely sharp vertical spike reaches a value of approximately 23 at the x-coordinate of 250.</p>	
Similaridad	
Número de Errores:	4 de 1000
Tabla 66: Inserción y extracción por el algoritmo DWT y la transformada de Haar, $s = 1, t = 5$	

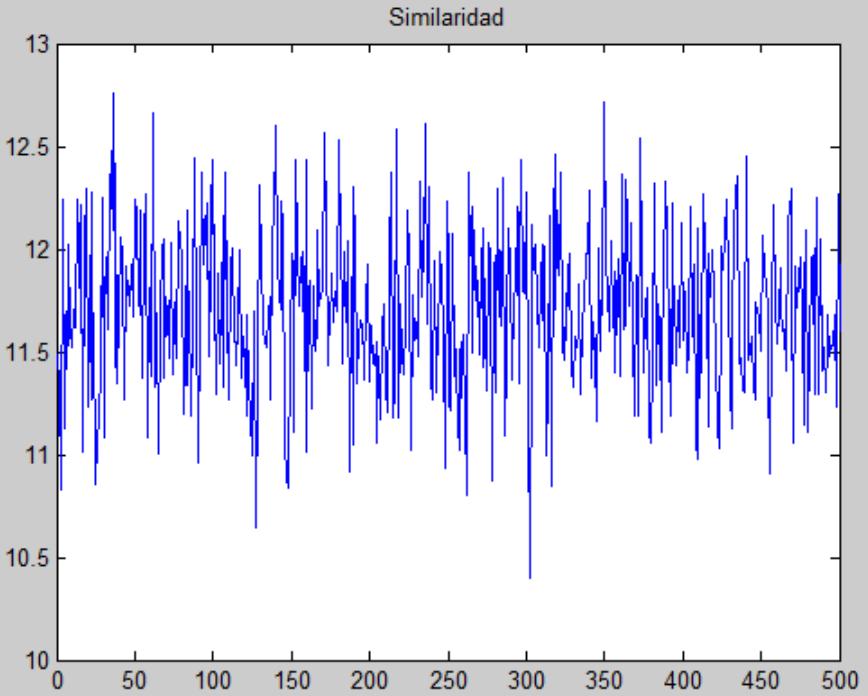
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

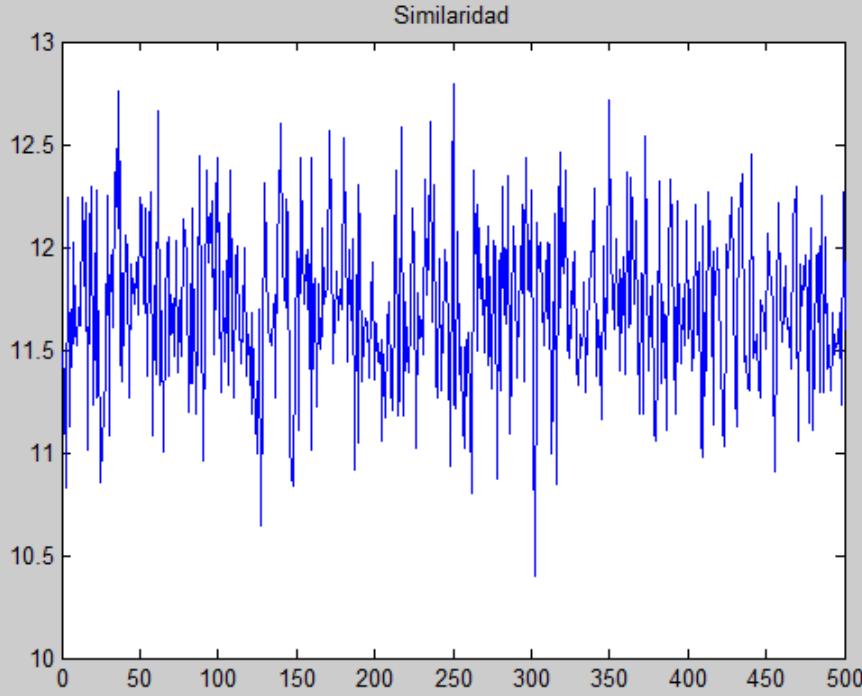
Parámetros:	s = 2, t = 10
	
Imagen Original	Imagen Marcada
	PSNR: 48.2316 dB
	Calidad: Muy Buena
	
Similaridad	
Número de Errores:	1 de 1000
Tabla 67: Inserción y extracción por el algoritmo DWT y la transformada de Haar, s = 2, t = 10	

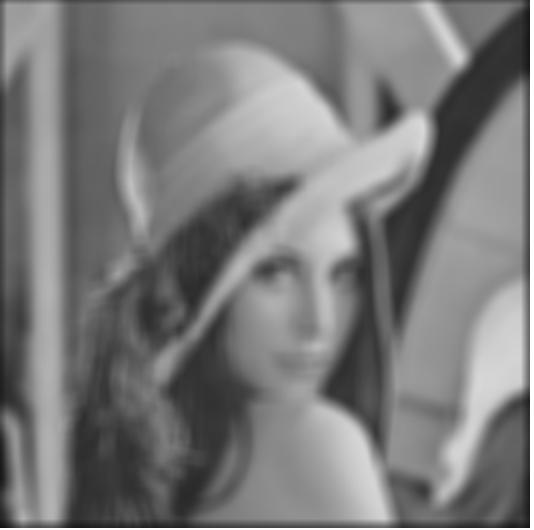
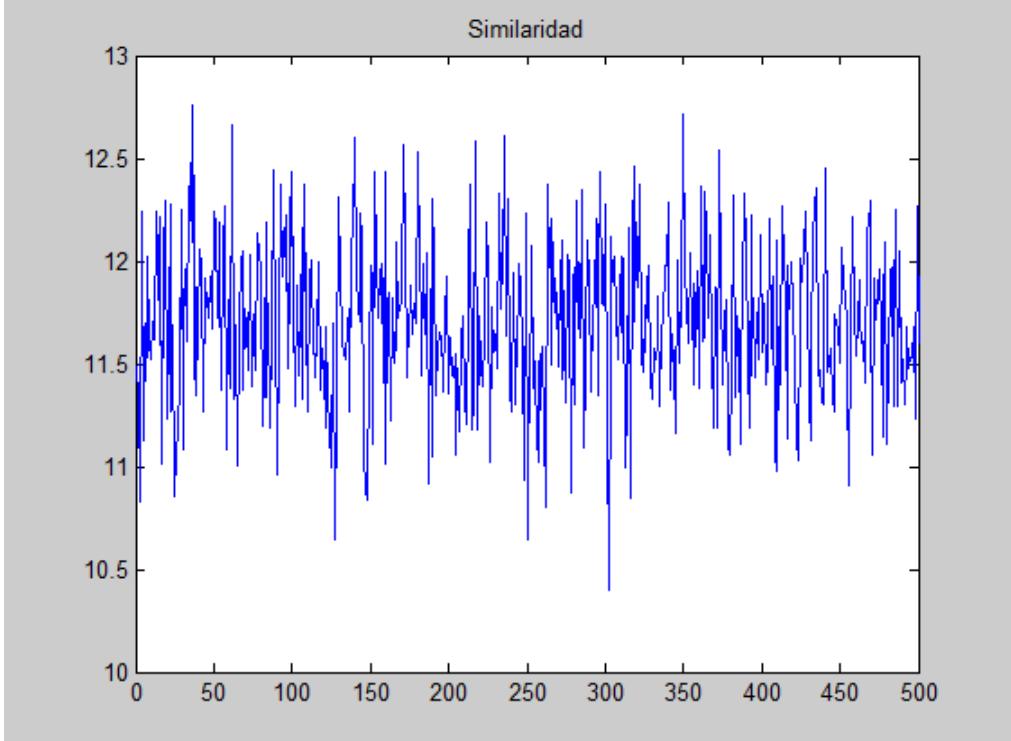
Parámetros:	$s = 4, t = 20$		
			
Imagen Original	Imagen Marcada		
	PSNR: 42.5173 dB		
	Calidad: Buena		
 <p>Similaridad</p> <p>Este gráfico muestra la similaridad entre los bloques de imágenes en función de su posición. La escala vertical (Y) va de 10 a 24, y la horizontal (X) de 0 a 500. La curva es generalmente estable en un rango entre 10 y 12, con una excepción significativa: un pico azul que alcanza una altura de aproximadamente 23.5 alrededor de la posición 250.</p>			
<p>Similaridad</p> <table> <tr> <td>Número de Errores:</td> <td>7 de 1000</td> </tr> </table>		Número de Errores:	7 de 1000
Número de Errores:	7 de 1000		
<p>Tabla 68: Inserción y extracción por el algoritmo DWT y la transformada de Haar, $s = 4, t = 20$</p>			

7.3.5.2.2. Pruebas de ataques

Se hará una prueba de cada ataque de los definidos, en ellas se comprobará la resistencia del algoritmo a los mismos, para estas pruebas se cogerá la imagen marcada que mejores resultados dio en las pruebas anteriores.

Nombre del ataque:	Compresión JPEG, 60%
	
Imagen Marcada	Imagen Atacada
	
Similaridad	
Número de Errores:	484 de 1000
Tabla 69: Ataque compresión JPEG 60% algoritmo DWT y la transformada de Haar	

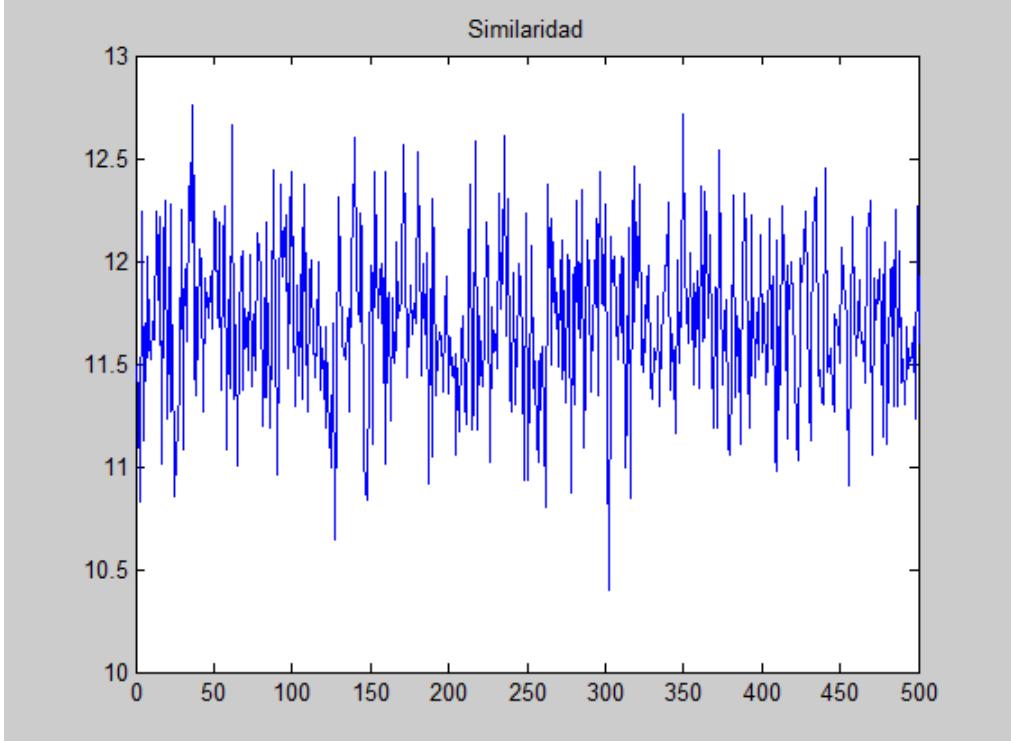
Nombre del ataque:	Inserción de ruido gaussiano
	
Imagen Marcada	Imagen Atacada
	
Número de Errores:	Similaridad 459 de 1000
Tabla 70: Ataque inserción ruido gaussiano algoritmo DWT y la transformada de Haar	

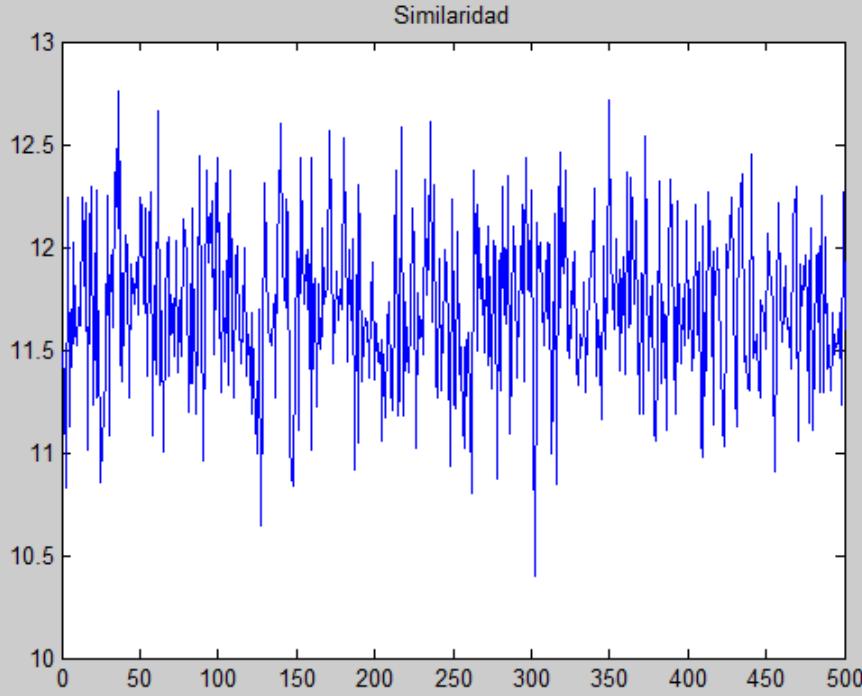
Nombre del ataque:	Aplicación de un filtro de paso bajo basado en la media
	
Imagen Marcada	Imagen Atacada
	
Número de Errores:	525 de 1000
Tabla 71: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo DWT y la transformada de Haar	

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del ataque:	Recortado de la imagen marcada
	
Imagen Marcada	Imagen Atacada
Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben tener al menos, las mismas dimensiones.	
Número de Errores:	Similaridad No Necesario

Tabla 72: Ataque recortado de la imagen marcada algoritmo DWT y la transformada de Haar

Nombre del ataque:	Escalado de la imagen marcada, doble tamaño
	
Imagen Marcada	Imagen Atacada
	
Similaridad	
Número de Errores:	506 de 1000
Tabla 73: Ataque escalado de la imagen marcada algoritmo DWT y la transformada de Haar	

Nombre del ataque:	Rotación de la imagen marcada, 90°
	
Imagen Marcada	Imagen Atacada
	
Número de Errores:	498 de 1000
Tabla 74: Ataque rotación de la imagen marcada algoritmo DWT y la transformada de Haar	

7.3.5.2.3. Pruebas de tiempos de ejecución

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Por último, se comprobará la eficiencia del algoritmo en tiempo de computación, para ello se cogerá una media de diez ejecuciones del mismo.

Algoritmo:	DWT y la transformada de Haar	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	3.0576	0.3120
Ejecución 2	3.1200	0.3120
Ejecución 3	3.1044	0.3432
Ejecución 4	3.0732	0.3900
Ejecución 5	3.0420	0.2964
Ejecución 6	3.0264	0.3432
Ejecución 7	3.0888	0.3276
Ejecución 8	3.0576	0.3120
Ejecución 9	3.0264	0.3588
Ejecución 10	3.0576	0.3744
Media	3.0654 seg.	0.3370 seg.

Tabla 75: Tiempos de ejecución del algoritmo DWT y la transformada de Haar

7.3.5.2.4. Conclusiones individuales

Después del estudio individual que se acaba de realizar se puede concluir que:

- En este algoritmo la extracción de la marca de agua no depende de un solo parámetro, sino de dos, de s y de t , en función de que se vayan **incrementando** los **valores** de estos la **calidad** de la imagen marcada **disminuirá** sin llegar a ser siquiera regular esta, la **extracción** de la marca de agua es **satisfactoria** en los casos estudiados.
- En cuanto a los ataques:
 - Es resistente a: .
 - Es vulnerable a: compresión JPEG al 60%, inserción de ruido gaussiano, filtro de paso bajo de la media, escalado, rotación.
 - No es aplicable a: recortado.

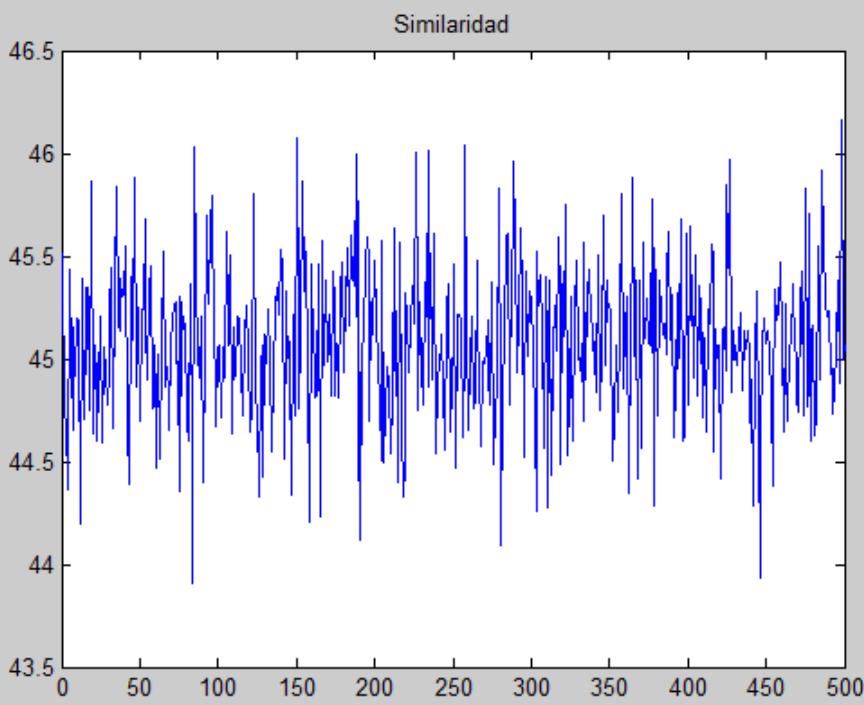
7.3.5.3. Tercer Método: DWT basado en la paridad

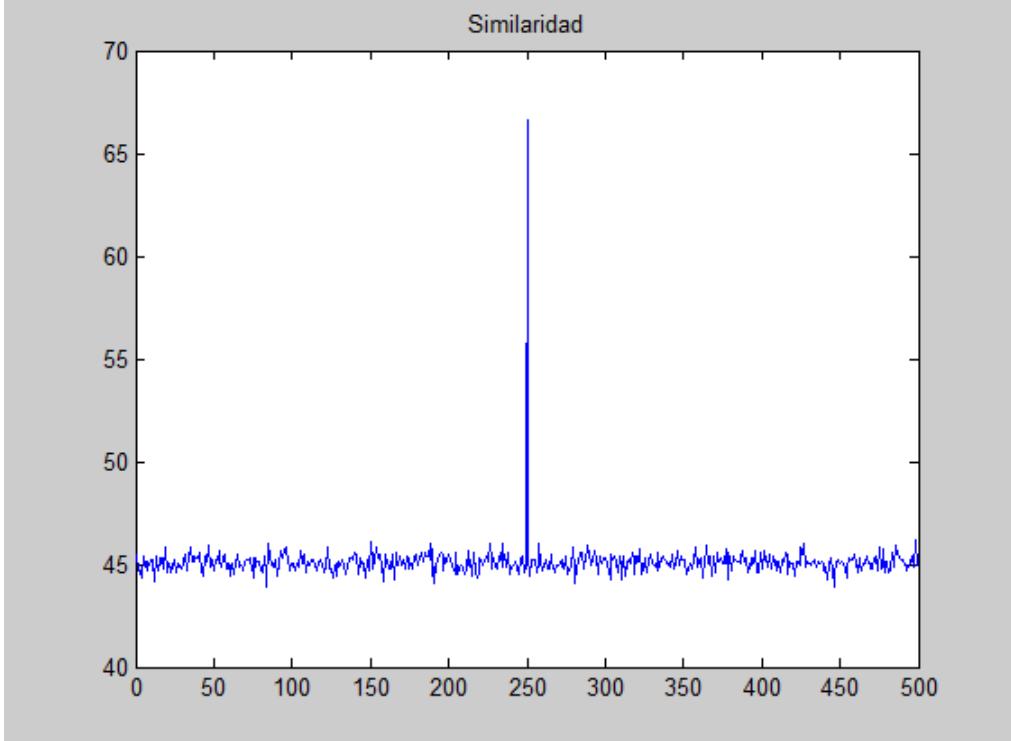
7.3.5.3.1. Pruebas de parámetros

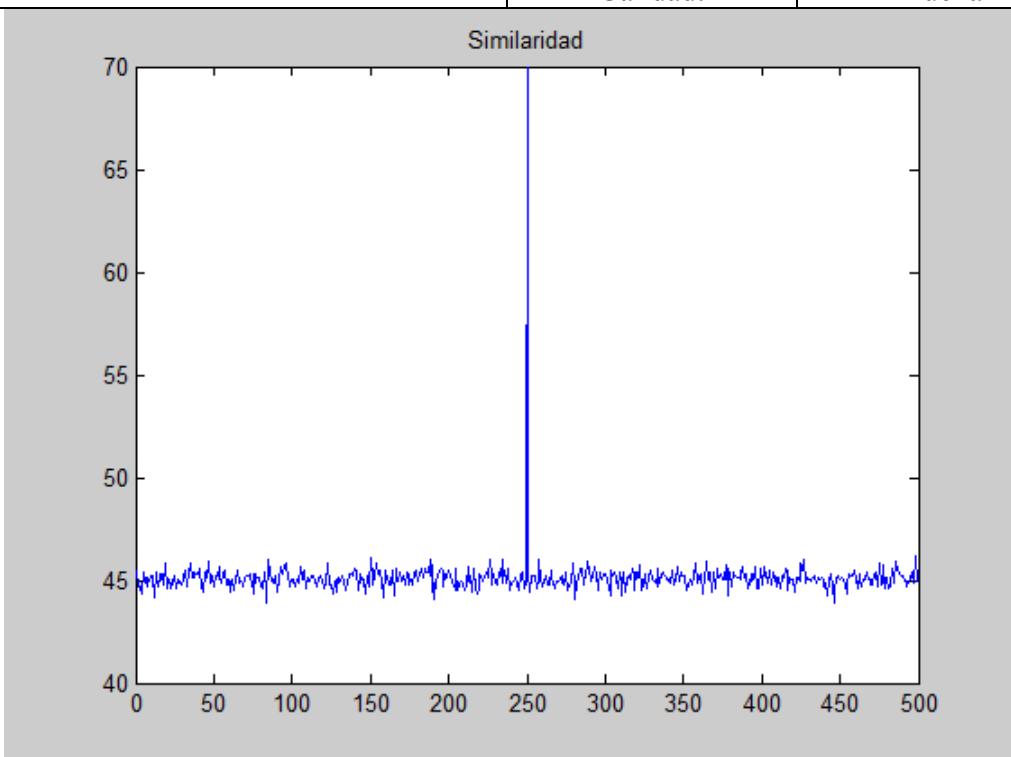
Este algoritmo utilizará la imagen de Lena de dimensiones 512x512, para la marca de agua se usará una secuencia pseudoaleatoria de dimensiones 128x128, como clave para inicializar la semilla se utilizará el valor de 250. El parámetro que sufrirá las variaciones es ‘constante’.

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

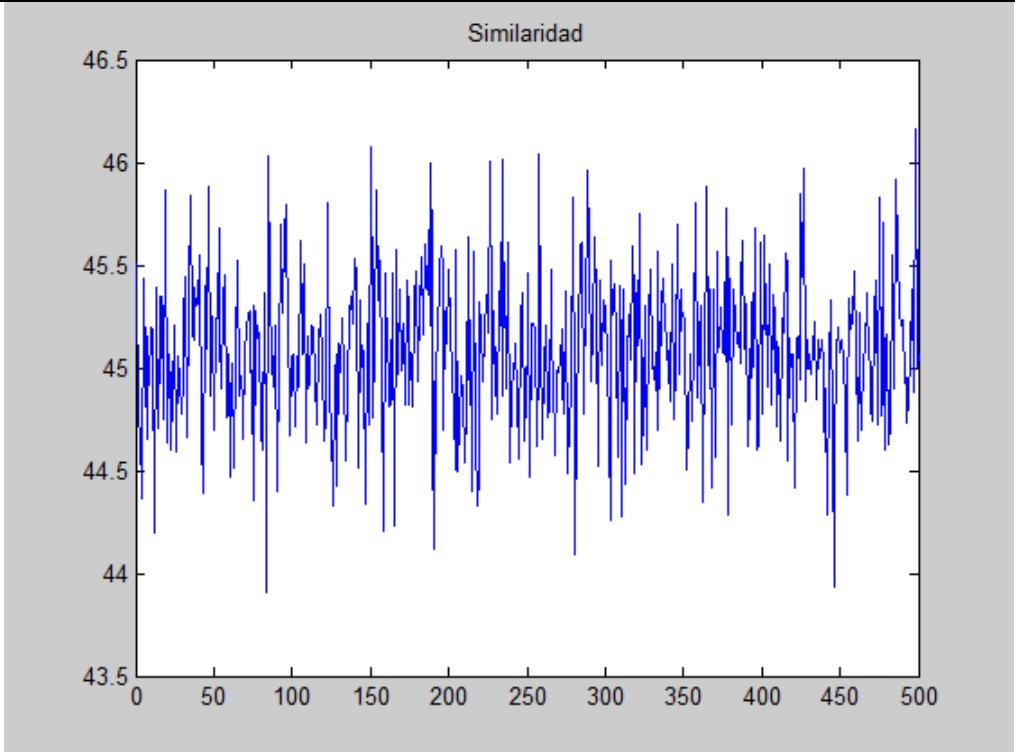
Como datos que se recogerán se encuentran la imagen marcada, la gráfica de similaridad, los valores de PSNR, el número de errores de la marca extraída y los tiempos de computación en la inserción y extracción.

Parámetros:	constante = 1		
			
Imagen Original	Imagen Marcada		
	PSNR: 66.2184 dB		
	Calidad: Muy Buena		
 <p>Similaridad</p> <p>Este gráfico muestra la "Similaridad" en el eje vertical (ranging de 43.5 a 46.5) contra un índice en el eje horizontal (ranging de 0 a 500). La curva es altamente fluctuante, con picos y valles que siguen una tendencia generalmente ascendente.</p>			
<p>Similaridad</p> <table> <tr> <td>Número de Errores:</td> <td>55 de 128</td> </tr> </table>		Número de Errores:	55 de 128
Número de Errores:	55 de 128		
<p>Tabla 76: Inserción y extracción por el algoritmo DWT basado en la paridad, constante = 1</p>			

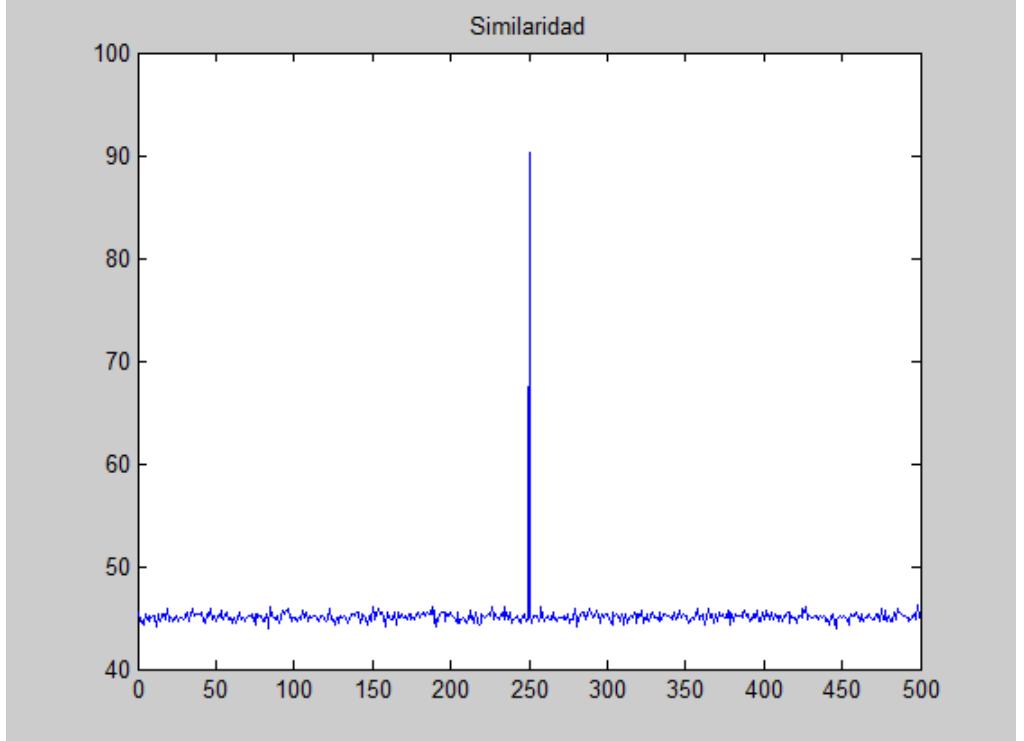
Parámetros:	constante = 10						
							
Imagen Original	Imagen Marcada						
	PSNR: 47.4737 dB						
	Calidad: Muy Buena						
 <p>Similaridad</p> <p>Este gráfico muestra la similaridad entre los bloques de imágenes en función de su posición. La escala vertical (Y) va de 40 a 70, y la horizontal (X) de 0 a 500. Una línea azul vertical se sitúa a una distancia constante de 250 unidades.</p> <table border="1"> <caption>Valores estimados del gráfico</caption> <thead> <tr> <th>Posición (X)</th> <th>Similaridad (Y)</th> </tr> </thead> <tbody> <tr><td>0 - 250</td><td>45 ± 5</td></tr> <tr><td>250 - 500</td><td>45 ± 5</td></tr> </tbody> </table>		Posición (X)	Similaridad (Y)	0 - 250	45 ± 5	250 - 500	45 ± 5
Posición (X)	Similaridad (Y)						
0 - 250	45 ± 5						
250 - 500	45 ± 5						
<p>Similaridad</p> <p>Número de Errores: 31 de 128</p>							
<p>Tabla 77: Inserción y extracción por el algoritmo DWT basado en la paridad, constante = 10</p>							

Parámetros:	constante = 20						
							
Imagen Original	Imagen Marcada						
	PSNR: 41.5258 dB						
	Calidad: Buena						
 <p>Similaridad</p> <p>Este gráfico muestra la similaridad entre los bloques de imágenes en función de su posición. La escala vertical (Y) va de 40 a 70, y la horizontal (X) de 0 a 500. Una línea azul vertical marca el punto de corte en x=250.</p> <table border="1"> <thead> <tr> <th>Número de Bloques</th> <th>Similaridad</th> </tr> </thead> <tbody> <tr><td>0 - 250</td><td>~45</td></tr> <tr><td>250 - 500</td><td>~45</td></tr> </tbody> </table>		Número de Bloques	Similaridad	0 - 250	~45	250 - 500	~45
Número de Bloques	Similaridad						
0 - 250	~45						
250 - 500	~45						
<p>Similaridad</p> <p>Número de Errores: 22 de 128</p>							
<p>Tabla 78: Inserción y extracción por el algoritmo DWT basado en la paridad, constante = 20</p>							

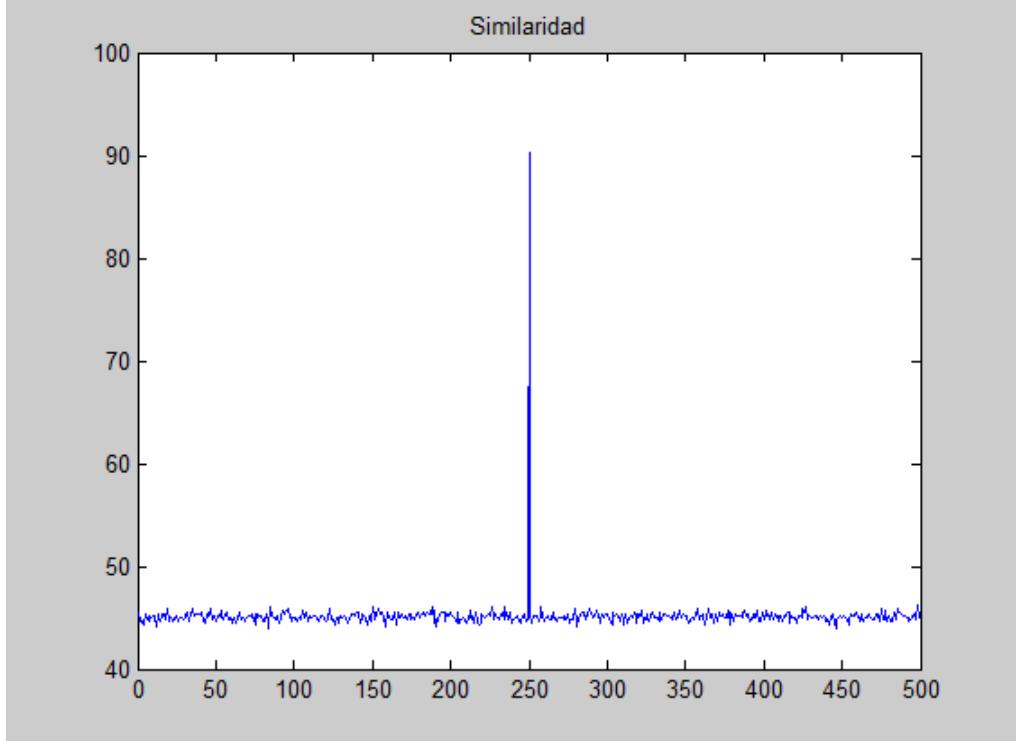
Prueba con el algoritmo de inserción modificado.

Parámetros:	constante = 1
	
Imagen Original	Imagen Marcada
	PSNR: 61.6405 dB
	Calidad: Muy Buena
	
Similaridad	
Número de Errores:	55 de 128
Tabla 79: Inserción modificada y extracción por el algoritmo DWT basado en la paridad, constante = 1	

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Parámetros:	constante = 10		
			
Imagen Original	Imagen Marcada		
	PSNR: 42.4344 dB		
	Calidad: Buena		
 <p>The graph plots 'Similaridad' (Similarity) on the y-axis (40 to 100) against an index on the x-axis (0 to 500). A vertical blue line marks the index 250, where the similarity value rises sharply from approximately 45 to about 90. The similarity values for most other indices are clustered around a baseline of 45.</p>			
<p>Similaridad</p> <table> <tr> <td>Número de Errores:</td> <td>0 de 128</td> </tr> </table>		Número de Errores:	0 de 128
Número de Errores:	0 de 128		
<p>Tabla 80: Inserción modificada y extracción por el algoritmo DWT basado en la paridad, constante = 10</p>			

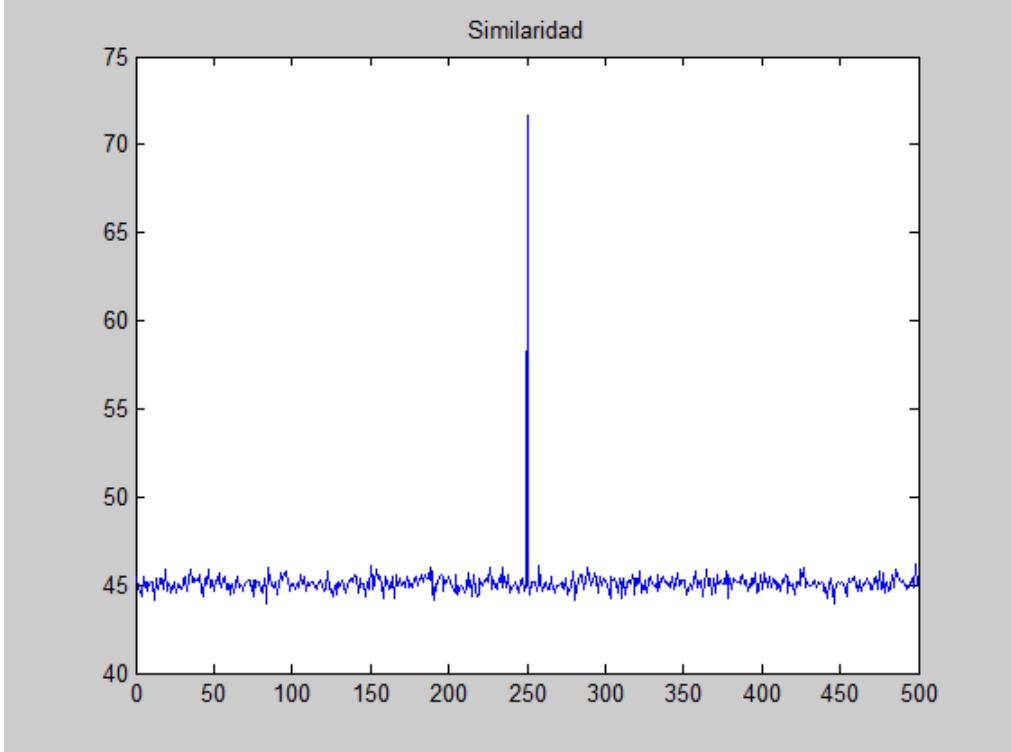
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Parámetros:	constante = 20		
			
Imagen Original	Imagen Marcada		
	PSNR: 36.4474 dB		
	Calidad: Regular		
 <p>The graph plots 'Similaridad' (Similarity) on the y-axis (40 to 100) against an index on the x-axis (0 to 500). A vertical blue line marks the index 250, where the similarity value is approximately 90. For indices less than 250 and greater than 250, the similarity values are consistently around 45, with minor fluctuations.</p>			
<p>Similaridad</p> <table> <tr> <td>Número de Errores:</td> <td>0 de 128</td> </tr> </table>		Número de Errores:	0 de 128
Número de Errores:	0 de 128		
Tabla 81: Inserción modificada y extracción por el algoritmo DWT basado en la paridad, constante = 20			

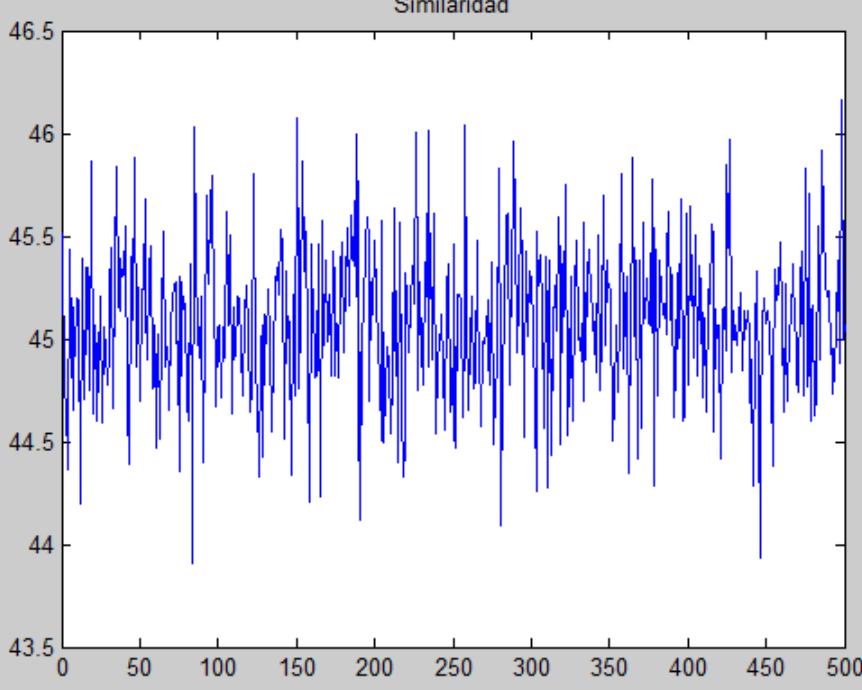
7.3.5.3.2. Pruebas de ataques

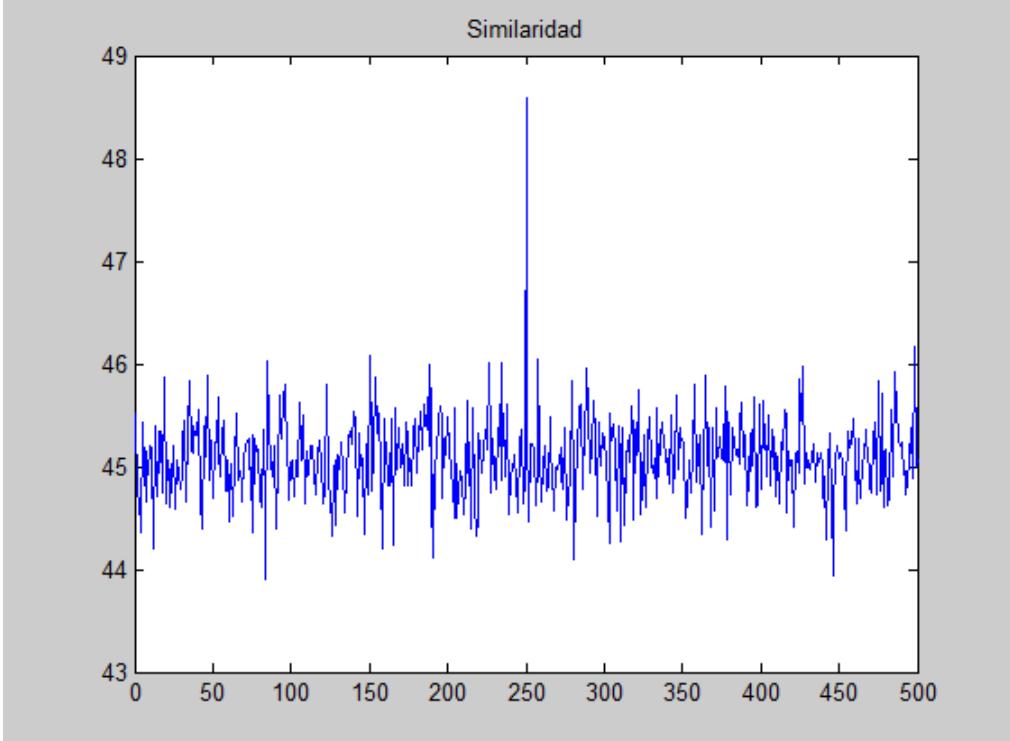
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Se hará una prueba de cada ataque de los definidos, en ellas se comprobará la resistencia del algoritmo a los mismos, para estas pruebas se cogerá la imagen marcada que mejores resultados dio en las pruebas anteriores. Para la inserción se ha utilizado la modificada ya que ofrece mejores resultados.

Nombre del ataque:	Compresión JPEG, 60%
	
Imagen Marcada	Imagen Atacada
	
Número de Errores:	28 de 128
Tabla 82: Ataque compresión JPEG 60% algoritmo DWT basado en la paridad	

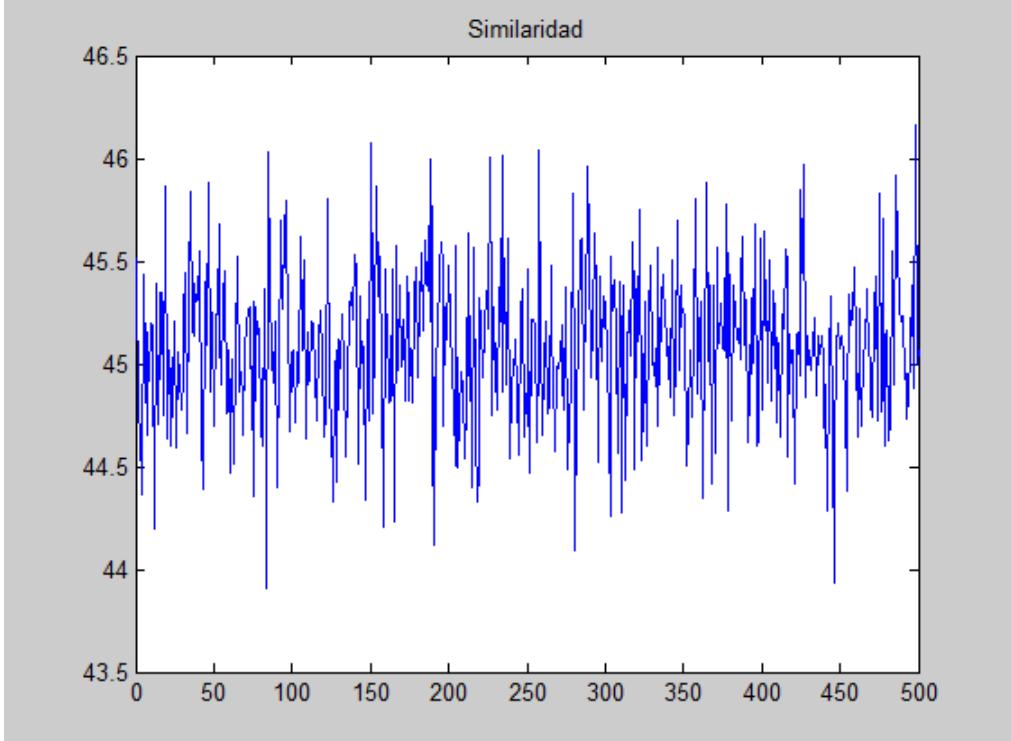
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

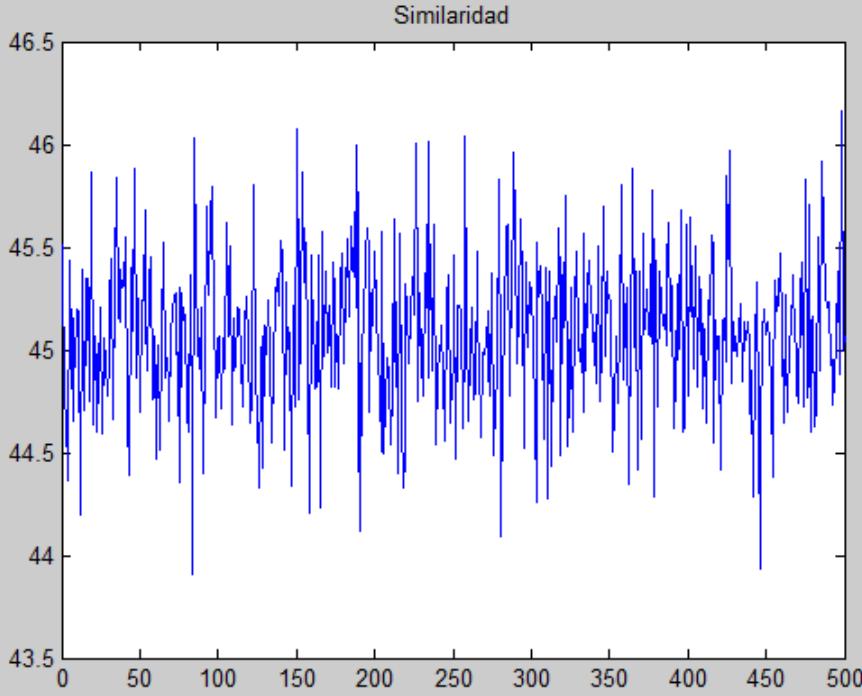
Nombre del ataque:	Inserción de ruido gaussiano
	
Imagen Marcada	Imagen Atacada
	
Similaridad	
Número de Errores:	56 de 128
Tabla 83: Ataque inserción ruido gaussiano algoritmo DWT basado en la paridad	

Nombre del ataque:	Aplicación de un filtro de paso bajo basado en la media
	
Imagen Marcada	Imagen Atacada
	
Número de Errores:	65 de 128
Tabla 84: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo DWT basado en la paridad	

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del ataque:	Recortado de la imagen marcada
	
Imagen Marcada	Imagen Atacada
Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben de tener al menos, las mismas dimensiones.	
Similaridad	
Número de Errores:	No Necesario
Tabla 85: Ataque recortado de la imagen marcada algoritmo DWT basado en la paridad	

Nombre del ataque:	Escalado de la imagen marcada, doble tamaño
	
Imagen Marcada	Imagen Atacada
	
Número de Errores:	56 de 128
Tabla 86: Ataque escalado de la imagen marcada algoritmo DWT basado en la paridad	

Nombre del ataque:	Rotación de la imagen marcada, 90°
	
Imagen Marcada	Imagen Atacada
	
Número de Errores:	60 de 128
Tabla 87: Ataque rotación de la imagen marcada algoritmo DWT basado en la paridad	

7.3.5.3.3. Pruebas de tiempos de ejecución

Por último, se comprobará la eficiencia del algoritmo en tiempo de computación, para ello se cogerá una media de diez ejecuciones del mismo.

Algoritmo:	DWT basado en la paridad	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	0.6240	0.3276
Ejecución 2	0.6240	0.4056
Ejecución 3	0.6864	0.3900
Ejecución 4	0.6708	0.3900
Ejecución 5	0.6552	0.4212
Ejecución 6	0.5928	0.3588
Ejecución 7	0.4992	0.4056
Ejecución 8	0.6240	0.4212
Ejecución 9	0.6240	0.3276
Ejecución 10	0.6552	0.4212
Media	0.6256 seg.	0.3869 seg.

Tabla 88: Tiempos de ejecución del algoritmo DWT basado en la paridad

Algoritmo:	DWT basado en la paridad (modificado)	
	Inserción de la marca de agua (modificado)	Extracción de la marca de agua
Ejecución 1	0.4992	0.3120
Ejecución 2	0.5616	0.3588
Ejecución 3	0.5148	0.3120
Ejecución 4	0.5616	0.3276
Ejecución 5	0.5148	0.3588
Ejecución 6	0.5616	0.3432
Ejecución 7	0.5460	0.3588
Ejecución 8	0.5616	0.3588
Ejecución 9	0.4992	0.3276
Ejecución 10	0.5928	0.3432
Media	0.5413 seg.	0.3401 seg.

Tabla 89: Tiempos de ejecución del algoritmo DWT basado en la paridad (modificado)

7.3.5.3.4. Conclusiones individuales

Después del estudio individual que se acaba de realizar se puede concluir que:

- En este algoritmo la extracción de la marca se ha realizado con dos posibles algoritmos de inserción, viendo que uno ofrece mejores resultados se comentará las conclusiones sobre él. El **algoritmo modificado de inserción** ofrece una **extracción** de la marca de agua **perfecta** con **valores** de la **constante altos** siendo la **calidad** de la imagen marcada **regular** como mucho, con un **valor bajo** de la **constante** la **calidad** de la imagen marcada es **muy buena** pero la **extracción** de la marca de agua es **mala**.
- En cuanto a los ataques:
 - Es resistente a: compresión JPEG al 60%, filtro de paso bajo de la media.
 - Es vulnerable a: inserción de ruido gaussiano, escalado, rotación.
 - No es aplicable a: recortado.

7.3.6. Algoritmos basados en la SVD

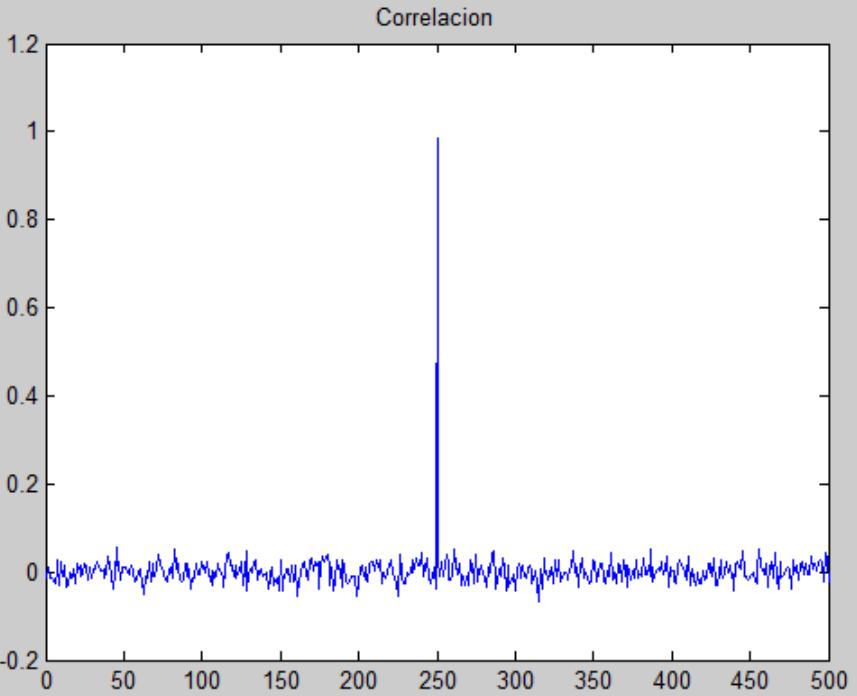
7.3.6.1. Primer Método: SVD

7.3.6.1.1. Pruebas de parámetros

Este algoritmo utilizará la imagen de Lena de dimensiones 256x256, para la marca de agua se usará una secuencia pseudoaleatoria de dimensiones 50x50 y como clave para inicializar la semilla se utilizará el valor 250.

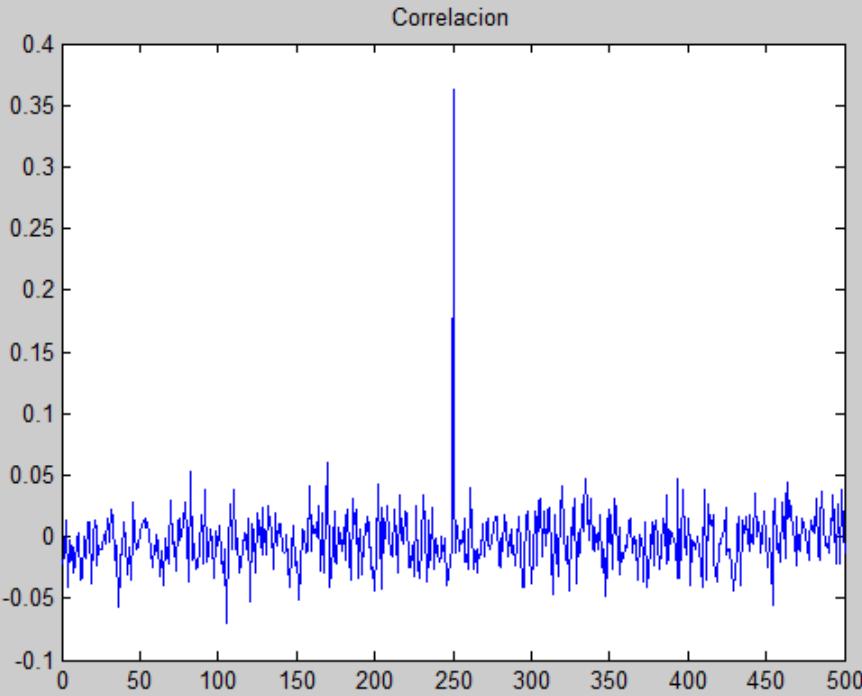
Como datos que se recogerán se encuentran la imagen marcada, la gráfica de correlación, los valores de PSNR y los tiempos de computación en la inserción y extracción.

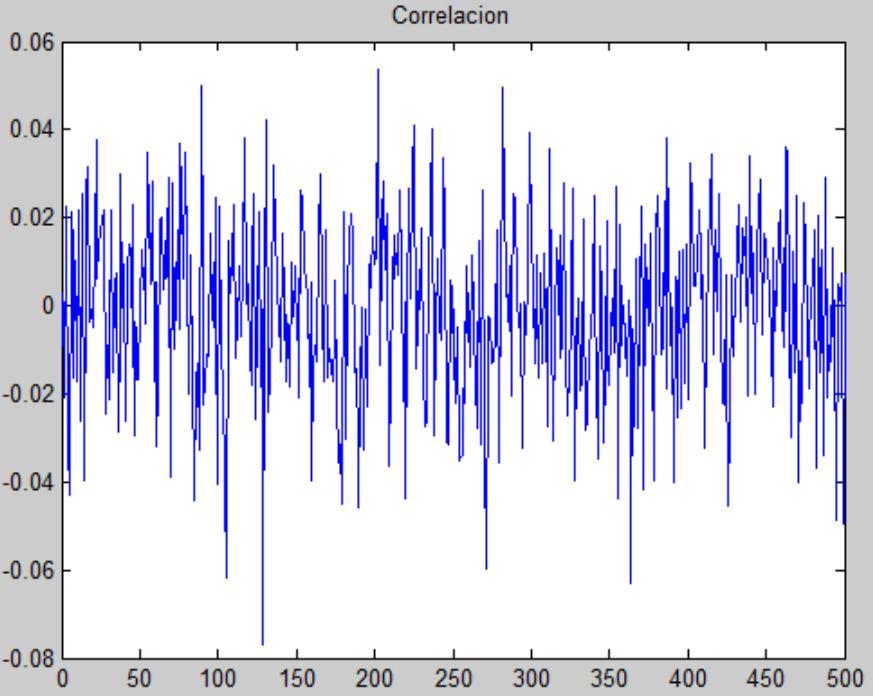
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

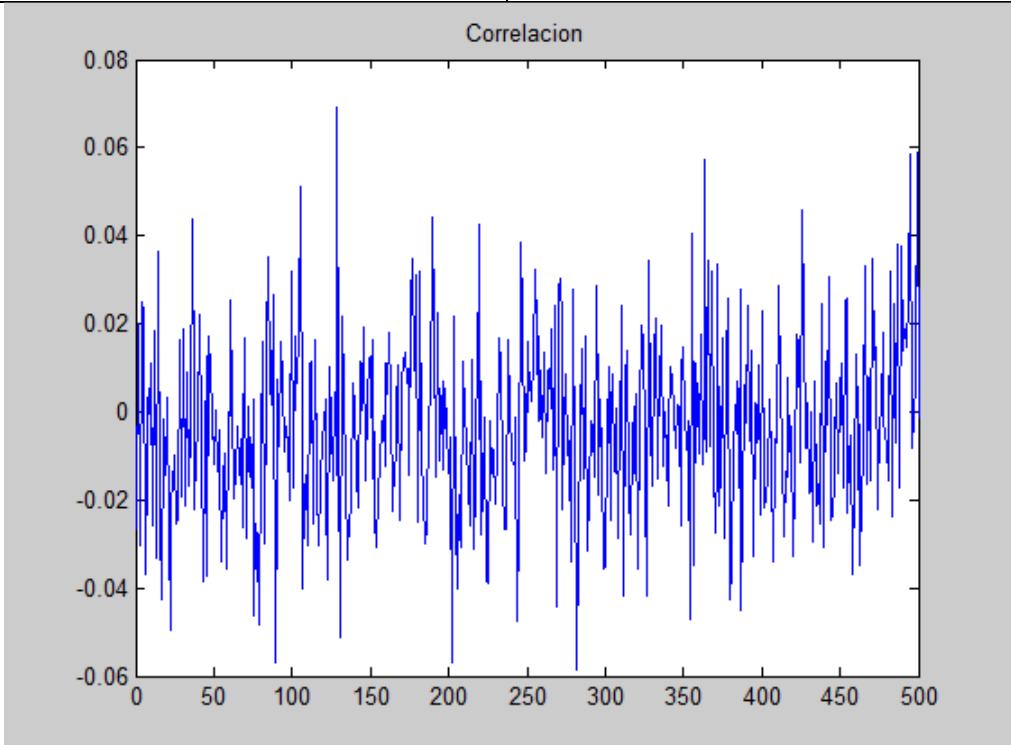
Parámetros:	Ninguno																			
																				
Imagen Original	Imagen Marcada																			
	PSNR: 91.2966 dB																			
	Calidad: Muy Buena																			
 <p>Correlacion</p> <table border="1"> <tr> <td>0</td> <td>50</td> <td>100</td> <td>150</td> <td>200</td> <td>250</td> <td>300</td> <td>350</td> <td>400</td> <td>450</td> <td>500</td> </tr> <tr> <td>-0.2</td> <td>0</td> <td>0.2</td> <td>0.4</td> <td>0.6</td> <td>0.8</td> <td>1</td> <td>1.2</td> </tr> </table>		0	50	100	150	200	250	300	350	400	450	500	-0.2	0	0.2	0.4	0.6	0.8	1	1.2
0	50	100	150	200	250	300	350	400	450	500										
-0.2	0	0.2	0.4	0.6	0.8	1	1.2													
Similaridad / Correlación																				
Número de Errores:	No Necesario																			
Tabla 90: Inserción y extracción por el algoritmo SVD																				

7.3.6.1.2. Pruebas de ataques

Se hará una prueba de cada ataque de los definidos, en ellas se comprobará la resistencia del algoritmo a los mismos, para estas pruebas se cogerá la imagen marcada que mejores resultados dio en las pruebas anteriores.

Nombre del ataque:	Compresión JPEG, 60%
	
Imagen Marcada	Imagen Atacada
	
Correlación	
Número de Errores:	No Necesario
Tabla 91: Ataque compresión JPEG 60% algoritmo SVD	

Nombre del ataque:	Inserción de ruido gaussiano
	
Imagen Marcada	Imagen Atacada
	
Correlación	
Número de Errores:	No Necesario
Tabla 92: Ataque inserción ruido gaussiano algoritmo SVD	

Nombre del ataque:	Aplicación de un filtro de paso bajo basado en la media
	
Imagen Marcada	Imagen Atacada
	
Correlación	
Número de Errores:	No Necesario
Tabla 93: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo SVD	

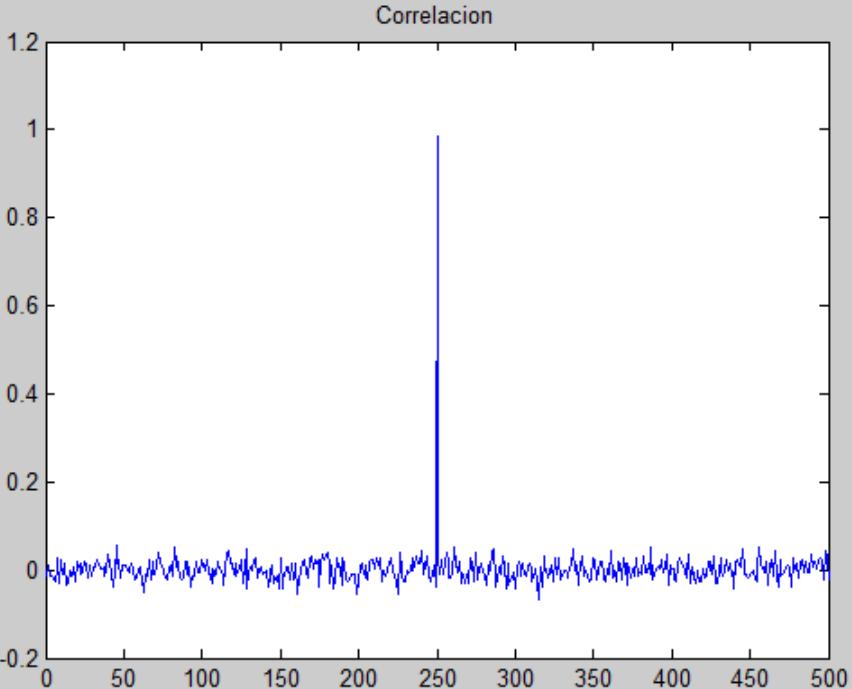
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del ataque:	Recortado de la imagen marcada
	
Imagen Marcada	Imagen Atacada
Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben tener las mismas dimensiones.	
Correlación	
Número de Errores:	No Necesario
Tabla 94: Ataque recortado de la imagen marcada algoritmo SVD	

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del ataque:	Escalado de la imagen marcada, doble tamaño
	
Imagen Marcada	Imagen Atacada
Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben tener las mismas dimensiones.	
Correlación	
Número de Errores:	No Necesario
Tabla 95: Ataque escalado de la imagen marcada algoritmo SVD	

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del ataque:	Rotación de la imagen marcada, 90°
	
Imagen Marcada	Imagen Atacada
	
Número de Errores:	Correlación No Necesario
Tabla 96: Ataque rotación de la imagen marcada algoritmo SVD	

7.3.6.1.3. Pruebas de tiempos de ejecución

Por último, se comprobará la eficiencia del algoritmo en tiempo de computación, para ello se cogerá una media de diez ejecuciones del mismo.

Algoritmo:	SVD	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	1.0452	0.5148
Ejecución 2	0.8268	0.4056
Ejecución 3	0.9048	0.5460
Ejecución 4	0.9672	0.4524
Ejecución 5	0.8892	0.4836
Ejecución 6	0.8736	0.5460
Ejecución 7	0.8424	0.5304
Ejecución 8	0.7956	0.4524
Ejecución 9	0.8268	0.5616
Ejecución 10	0.7800	0.5304
Media	0.8752 seg.	0.5023 seg.

Tabla 97: Tiempos de ejecución del algoritmo SVD

7.3.6.1.4. Conclusiones individuales

Después del estudio individual que se acaba de realizar se puede concluir que:

- En este algoritmo no hay **ningún parámetro** que modificar con lo que solo se puede concluir que la **extracción** que ofrece el algoritmo es **muy buena** y la **calidad de la imagen marcada** también es **muy buena**.
- En cuanto a los ataques:
 - Es resistente a: compresión JPEG al 60%, rotación.
 - Es vulnerable a: inserción de ruido gaussiano, filtro de paso bajo de la media.
 - No es aplicable a: recortado, escalado.

7.3.6.2. Segundo Método: SVD con transformada de Arnold

7.3.6.2.1. Pruebas de parámetros

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Este algoritmo utilizará la imagen de Lena de dimensiones 512x512, para la marca de agua se usará el logotipo del niño. Las pruebas se harán para diferentes valores de alpha.

Como datos que se recogerán se encuentran la imagen marcada, la marca extraída, los valores de PSNR, el número de errores de la marca extraída y los tiempos de computación en la inserción y extracción.

Parámetros:	alpha = 0.3
	
Imagen Original	Imagen Marcada
	PSNR: 32.1369 dB
	Calidad: Buena
	
Marca Original	Marca Extraída
	Número de Errores: 3371 de 4096
Tabla 98: Inserción y extracción por el algoritmo SVD con transformada de Arnold, alpha = 0.3	

Parámetros:	alpha = 1
	
Imagen Original	Imagen Marcada
	PSNR: 21.6794 dB
	Calidad: Muy Mala
	
Marca Original	Marca Extraída
	Número de Errores: 1269 de 4096
Tabla 99: Inserción y extracción por el algoritmo SVD con transformada de Arnold, alpha = 1	

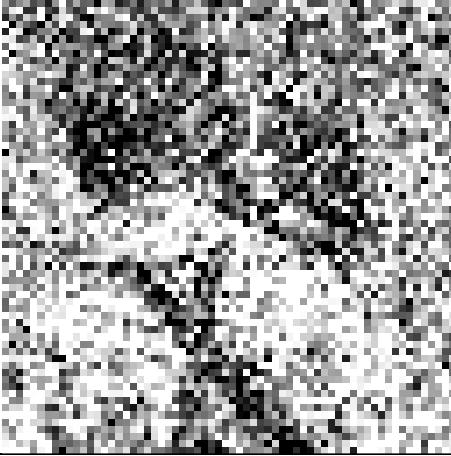
Parámetros:	alpha = 2
	
Imagen Original	Imagen Marcada
	PSNR: 15.6588 dB
	Calidad: Muy Mala
	
Marca Original	Marca Extraída
	Número de Errores: 549 de 4096
Tabla 100: Inserción y extracción por el algoritmo SVD con transformada de Arnold, alpha = 2	

7.3.6.2.2. Pruebas de ataques

Se hará una prueba de cada ataque de los definidos, en ellas se comprobará la resistencia del algoritmo a los mismos, para estas pruebas se cogerá la imagen marcada que mejores resultados dio en las pruebas anteriores.

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del Ataque:	Compresión JPEG, 60%
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 3993 de 4096
Tabla 101: Ataque compresión JPEG 60% algoritmo SVD con transformada de Arnold	

Nombre del Ataque:	Inserción de ruido gaussiano
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 4077 de 4096
Tabla 102: Ataque inserción ruido gaussiano algoritmo SVD con transformada de Arnold	

Nombre del Ataque:	Aplicación de un filtro de paso bajo basado en la media
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 4048 de 4096
Tabla 103: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo SVD con transformada de Arnold	

Nombre del Ataque:	Recortado de la imagen marcada
	
Imagen Marcada	Imagen Atacada
	Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben tener las mismas dimensiones.
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: No Necesario
Tabla 104: Ataque recortado de la imagen marcada algoritmo SVD con transformada de Arnold	

Nombre del Ataque:	Escalado de la imagen marcada, doble tamaño
	
Imagen Marcada	Imagen Atacada
	Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben tener las mismas dimensiones.
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: No Necesario
Tabla 105: Ataque escalado de la imagen marcada algoritmo SVD con transformada de Arnold	

Nombre del Ataque:	Rotación de la imagen marcada, 90°
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
Número de Errores: 4095 de 4096	
Tabla 106: Ataque rotación de la imagen marcada algoritmo SVD con transformada de Arnold	

7.3.6.2.3. Pruebas de tiempos de ejecución

Por último, se comprobará la eficiencia del algoritmo en tiempo de computación, para ello se cogerá una media de diez ejecuciones del mismo.

Algoritmo:	SVD con transformada de Arnold	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	17.5189	16.0525
Ejecución 2	17.0821	16.2241
Ejecución 3	17.6749	17.0197
Ejecución 4	17.3785	16.8481
Ejecución 5	17.7373	16.5673
Ejecución 6	17.0977	16.6297
Ejecución 7	17.8153	16.6141
Ejecución 8	17.7997	15.9745
Ejecución 9	17.7217	16.7545
Ejecución 10	17.6125	16.0057
Media	17.5439 seg.	16.4690 seg.

Tabla 107: Tiempos de ejecución del algoritmo SVD con transformada de Arnold

8.3.6.2.4. Conclusiones individuales

Después del estudio individual que se acaba de realizar se puede concluir que:

- En este algoritmo la **extracción** es **bastante mala**, aunque depende del parámetro alpha, ya que un valor de **alpha bajo** hace que la imagen marcada tenga una **calidad buena**, la **extracción** es **mala**; con un valor de **alpha alto** la **calidad** de la imagen marcada es **muy mala** y la **extracción** de la marca de agua es **buena**.
- En cuanto a los ataques:
 - Es resistente a: .
 - Es vulnerable a: compresión JPEG al 60%, inserción de ruido gaussiano, filtro de paso bajo de la media, rotación.
 - No es aplicable a: recortado, escalado.

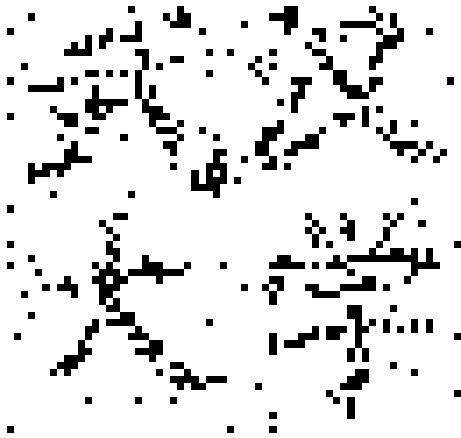
7.3.6.3. Tercer Método: SVD basado en el intercambio de valores

7.3.6.3.1. Pruebas de parámetros

Este algoritmo utilizará la imagen de Lena de dimensiones 512x512, para la marca de agua se usará el logotipo del niño. Las pruebas se harán para diferentes valores de alpha.

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Como datos que se recogerán se encuentran la imagen marcada, la marca extraída, los valores de PSNR, el número de errores de la marca extraída y los tiempos de computación en la inserción y extracción.

Parámetros:	alpha = 1
	
Imagen Original	Imagen Marcada
	PSNR: 38.8793 dB
	Calidad: Buena
	
Marca Original	Marca Extraída
	Número de Errores: 811 de 4096
Tabla 108: Inserción y extracción por el algoritmo SVD basado en el intercambio de valores, alpha = 1	

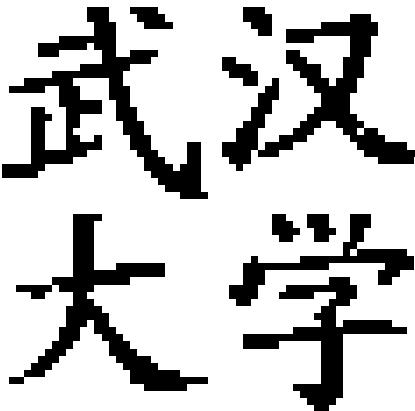
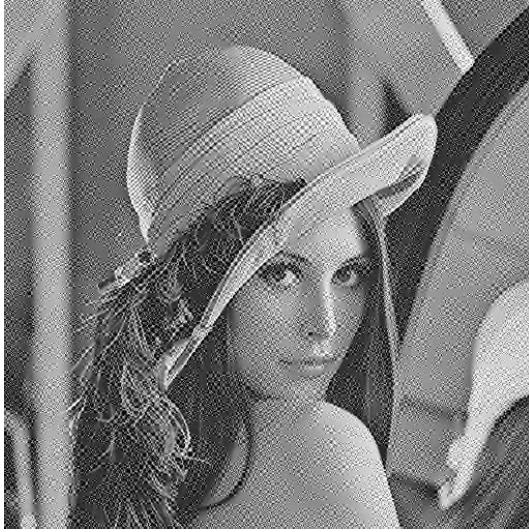
Parámetros:	alpha = 2
	
Imagen Original	Imagen Marcada
	PSNR: 38.7997 dB
	Calidad: Buena
	
Marca Original	Marca Extraída
	Número de Errores: 475 de 4096

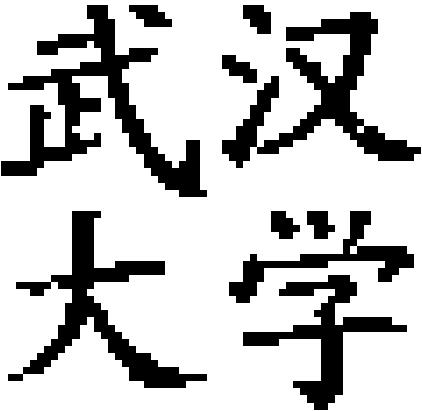
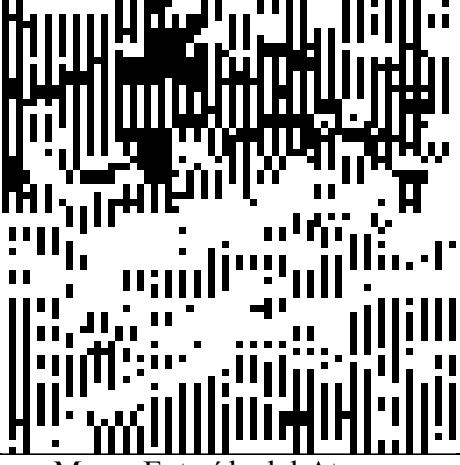
Tabla 109: Inserción y extracción por el algoritmo SVD basado en el intercambio de valores, alpha = 2

Parámetros:	alpha = 20
	
Imagen Original	Imagen Marcada
	PSNR: 33.4876 dB
	Calidad: Regular
	
Marca Original	Marca Extraída
	Número de Errores: 470 de 4096
Tabla 110: Inserción y extracción por el algoritmo SVD basado en el intercambio, alpha = 20	

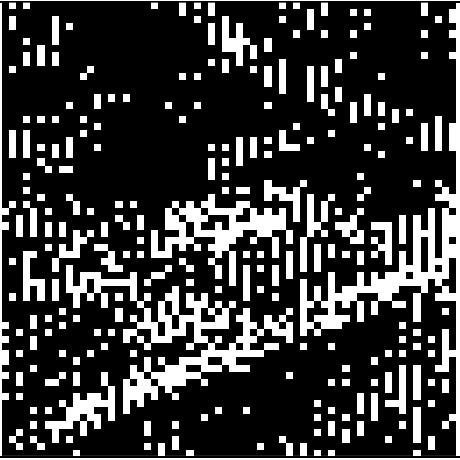
Parámetros:	alpha = 200
	
Imagen Original	Imagen Marcada
	PSNR: 14.9396 dB
	Calidad: Muy Mala
	
Marca Original	Marca Extraída
	Número de Errores: 470 de 4096
Tabla 111: Inserción y extracción por el algoritmo SVD basado en el intercambio de valores, alpha = 200	

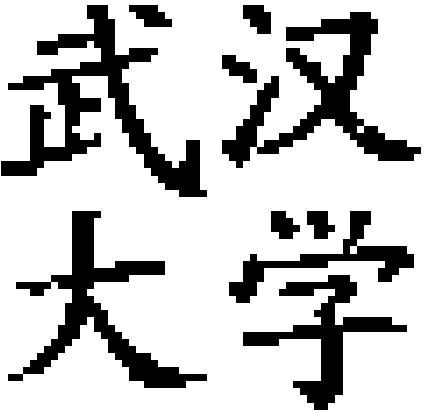
7.3.6.3.2. Pruebas de ataques

Se hará una prueba de cada ataque de los definidos, en ellas se comprobará la resistencia del algoritmo a los mismos, para estas pruebas se cogerá la imagen marcada que mejores resultados dio en las pruebas anteriores.

Nombre del Ataque:	Compresión JPEG, 60%
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 1923 de 4096
Tabla 112: Ataque compresión JPEG 60% algoritmo SVD basado en el intercambio de valores	

Nombre del Ataque:	Inserción de ruido gaussiano
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 1071 de 4096
Tabla 113: Ataque inserción ruido gaussiano algoritmo SVD en el intercambio de valores	

Nombre del Ataque:	Aplicación de un filtro de paso bajo basado en la media
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 2896 de 4096
Tabla 114: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo SVD basado en el intercambio de valores	

Nombre del Ataque:	Recortado de la imagen marcada
	
Imagen Marcada	Imagen Atacada
	Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben tener al menos, las mismas dimensiones.
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: No Necesario
Tabla 115: Ataque recortado de la imagen marcada algoritmo SVD basado en el intercambio de valores	

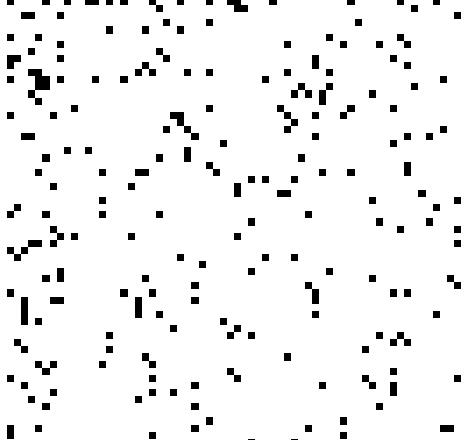
Nombre del Ataque:	Escalado de la imagen marcada, doble tamaño
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 1224 de 4096

Tabla 116: Ataque escalado de la imagen marcada algoritmo SVD basado en el intercambio de valores

Nombre del Ataque:	Rotación de la imagen marcada, 90°
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 1366 de 4096
Tabla 117: Ataque rotación de la imagen marcada algoritmo SVD basado en el intercambio de valores	

7.3.6.3.3. Pruebas de tiempos de ejecución

Por último, se comprobará la eficiencia del algoritmo en tiempo de computación, para ello se cogerá una media de diez ejecuciones del mismo.

Algoritmo:	SVD basado en el intercambio de valores	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	1.0608	0.8892
Ejecución 2	0.9204	0.9204
Ejecución 3	0.9672	0.8580
Ejecución 4	0.9984	0.9360
Ejecución 5	0.9828	0.9204
Ejecución 6	0.9828	0.8580
Ejecución 7	0.9672	0.9672
Ejecución 8	0.9048	0.8736
Ejecución 9	0.9516	0.9360
Ejecución 10	0.9516	0.9204
Media	0.9688 seg.	0.9079 seg.

Tabla 118: Tiempos de ejecución del algoritmo SVD basado en el intercambio de valores

7.3.6.3.4. Conclusiones individuales

Después del estudio individual que se acaba de realizar se puede concluir que:

- En este algoritmo la extracción depende del valor del parámetro alpha, un valor de **alpha bajo** hace que la imagen marcada tenga una **calidad buena** pero la marca de agua no se extrae hasta un valor de 2 aproximadamente, en cambio, un valor de **alpha alto** hace que la imagen marcada tenga una **calidad muy mala** y la **extracción** de la marca de agua sea **buena**, aunque ocurre algo curioso que es que siendo alpha muy grande la marca de agua tiene el mismo número de errores.
- En cuanto a los ataques:
 - Es resistente a: .
 - Es vulnerable a: compresión JPEG al 60%, inserción de ruido gaussiano, filtro de paso bajo de la media, escalado, rotación.
 - No es aplicable a: recortado.

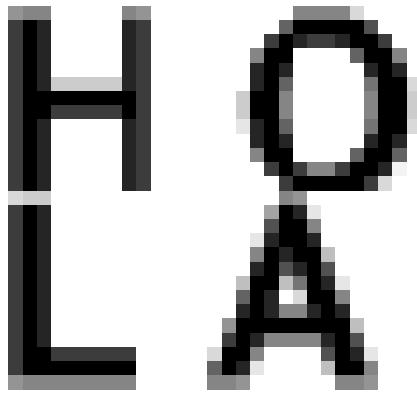
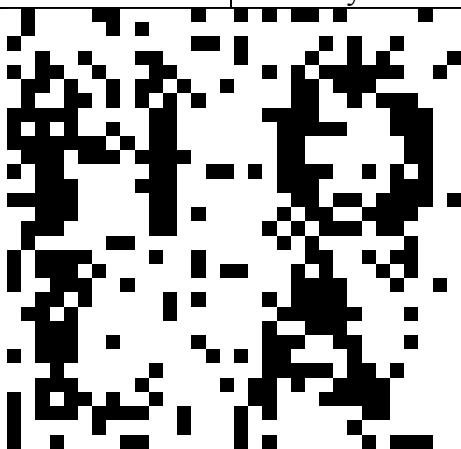
7.3.6.4. Cuarto Método: SVD basado en el orden de los coeficientes

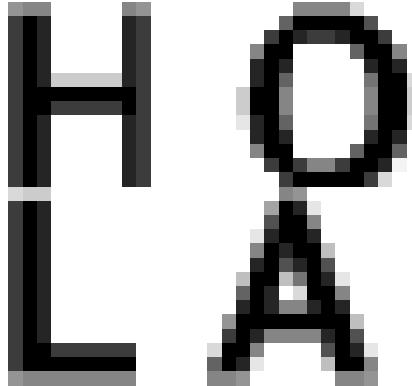
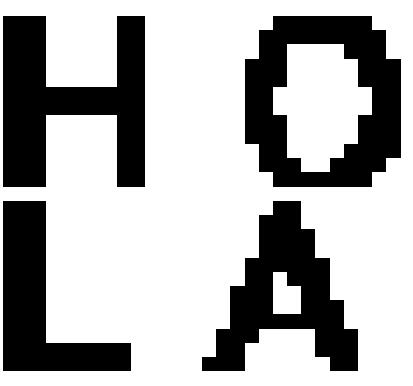
7.3.6.4.1. Pruebas de parámetros

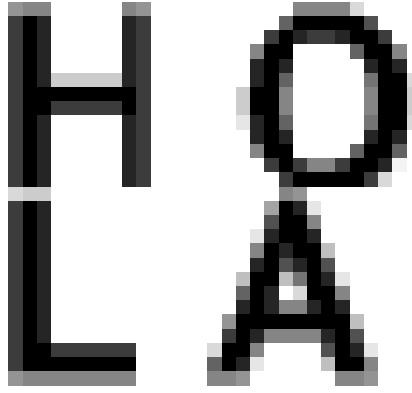
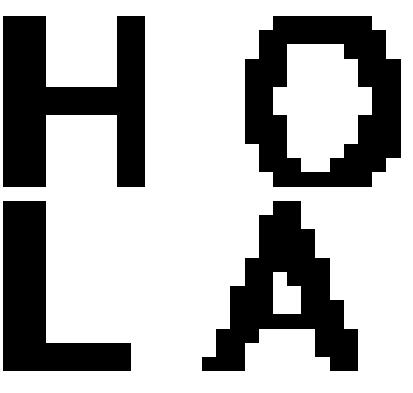
Este algoritmo utilizará la imagen de Lena de dimensiones 256x256, para la marca de agua se usará el logotipo de hola. Las pruebas se harán para diferentes valores de T, que es el umbral.

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Como datos que se recogerán se encuentran la imagen marcada, la marca extraída, los valores de PSNR, el número de errores de la marca extraída y los tiempos de computación en la inserción y extracción.

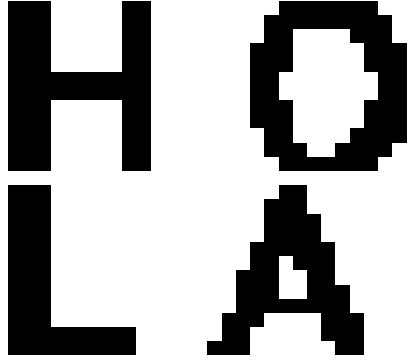
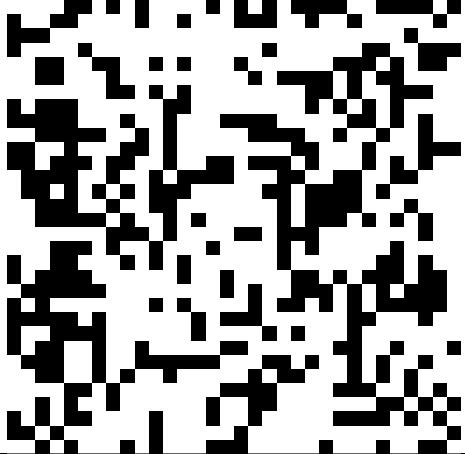
Parámetros:	T = 0.001
	
Imagen Original	Imagen Marcada
	PSNR: 46.7815 dB Calidad: Muy Buena
	
Marca Original	Marca Extraída
	Número de Errores: 363 de 1024
Tabla 119: Inserción y extracción por el algoritmo SVD basado en el orden de los coeficientes, T = 0.001	

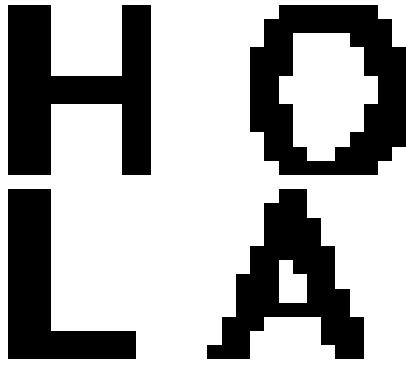
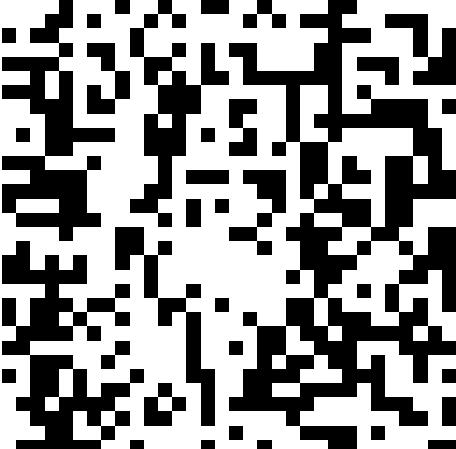
Parámetros:	T = 0.012
	
Imagen Original	Imagen Marcada
	PSNR: 43.1241 dB
	Calidad: Buena
	
Marca Original	Marca Extraída
	Número de Errores: 211 de 1024
Tabla 120: Inserción y extracción por el algoritmo SVD basado en el orden de los coeficientes, T = 0.012	

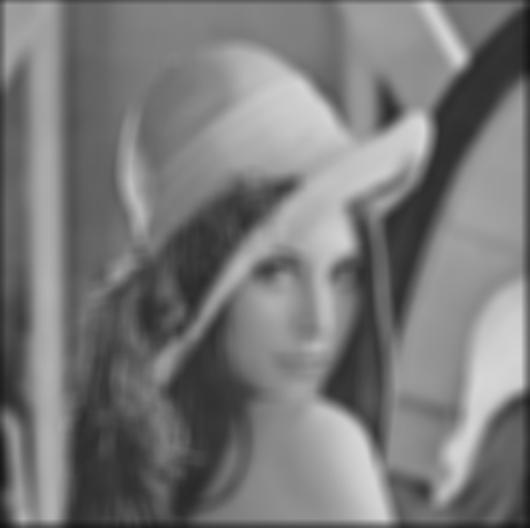
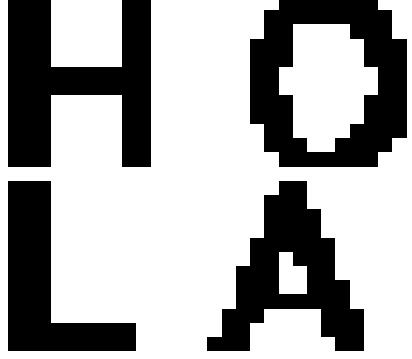
Parámetros:	T = 1
	
Imagen Original	Imagen Marcada
	PSNR: 8.3462 dB
	Calidad: Muy Mala
	
Marca Original	Marca Extraída
	Número de Errores: 211 de 1024
Tabla 121: Inserción y extracción por el algoritmo SVD basado en el orden de los coeficientes, T = 1	

7.3.6.4.2. Pruebas de ataques

Se hará una prueba de cada ataque de los definidos, en ellas se comprobará la resistencia del algoritmo a los mismos, para estas pruebas se cogerá la imagen marcada que mejores resultados dio en las pruebas anteriores.

Nombre del Ataque:	Compresión JPEG, 60%
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 478 de 1024
Tabla 122: Ataque compresión JPEG 60% algoritmo SVD basado en el orden de los coeficientes	

Nombre del Ataque:	Inserción de ruido gaussiano
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 499 de 1024
Tabla 123: Ataque inserción ruido gaussiano algoritmo SVD basado en el orden de los coeficientes	

Nombre del Ataque:	Aplicación de un filtro de paso bajo basado en la media
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 588 de 1024
Tabla 124: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo SVD basado en el orden de los coeficientes	

Nombre del Ataque:	Recortado de la imagen marcada
	
Imagen Marcada	Imagen Atacada
	Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben tener las mismas dimensiones.
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: No Necesario

Tabla 125: Ataque recortado de la imagen marcada algoritmo SVD basado en el orden de los coeficientes

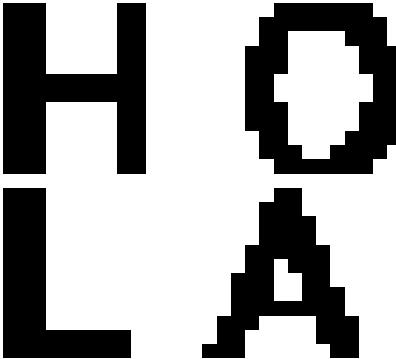
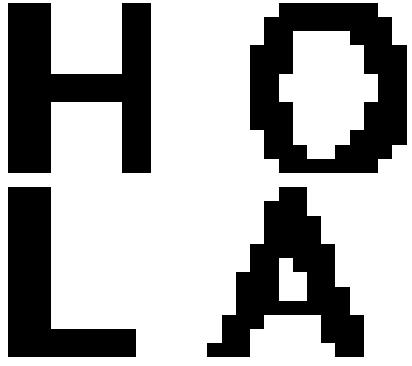
Nombre del Ataque:	Escalado de la imagen marcada, doble tamaño
	
Imagen Marcada	Imagen Atacada
	Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben tener las mismas dimensiones.
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: No Necesario

Tabla 126: Ataque escalado de la imagen marcada algoritmo SVD basado en el orden de los coeficientes

Nombre del Ataque:	Rotación de la imagen marcada, 90°
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 664 de 1024
Tabla 127: Ataque rotación de la imagen marcada algoritmo SVD basado en el orden de los coeficientes	

7.3.6.4.3. Pruebas de tiempos de ejecución

Por último, se comprobará la eficiencia del algoritmo en tiempo de computación, para ello se cogerá una media de diez ejecuciones del mismo.

Algoritmo:	SVD basado en el orden de los coeficientes	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	0.3276	0.1872
Ejecución 2	0.2340	0.2028
Ejecución 3	0.2340	0.2028
Ejecución 4	0.2964	0.2184
Ejecución 5	0.2340	0.2808
Ejecución 6	0.3120	0.2496
Ejecución 7	0.2652	0.1716
Ejecución 8	0.2340	0.2028
Ejecución 9	0.2496	0.2808
Ejecución 10	0.2808	0.2028
Media	0.2668 seg.	0.2200 seg.

Tabla 128: SVD basado en el orden de los coeficientes

7.3.6.4.4. Conclusiones individuales

Después del estudio individual que se acaba de realizar se puede concluir que:

- En este algoritmo la extracción es diferente en función del parámetro T, un valor de **T bajo** hace que la imagen marcada tenga una **calidad muy buena** pero no extrae bien la marca, en cambio, un valor de **T alto** hace que la imagen marcada tenga una **calidad muy mala**. Una cosa curiosa que ocurre con este algoritmo es que desde que se extrae la marcada de agua sin afectar mucho la imagen marcada hasta que T tiene un valor elevado el número de errores no parece cambiar.
- En cuanto a los ataques:
 - Es resistente a: .
 - Es vulnerable a: compresión JPEG al 60%, inserción de ruido gaussiano, filtro de paso bajo de la media, rotación.
 - No es aplicable a: recortado, escalado.

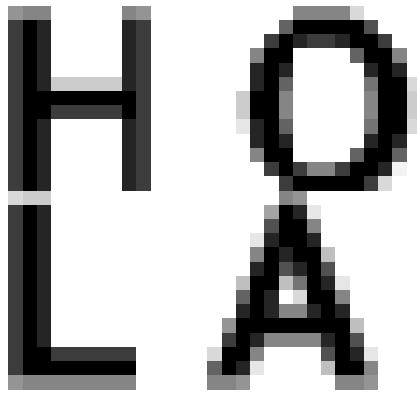
7.3.6.5. Quinto Método: SVD basado en la proximidad a un intervalo

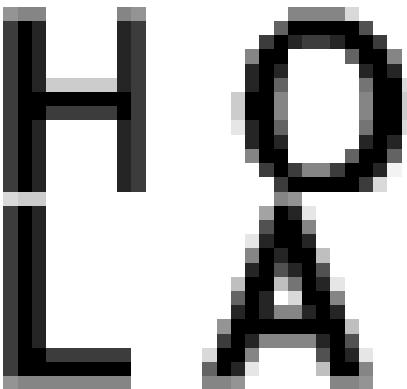
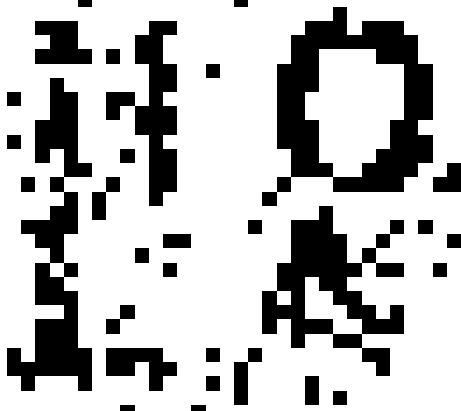
7.3.6.5.1. Pruebas de parámetros

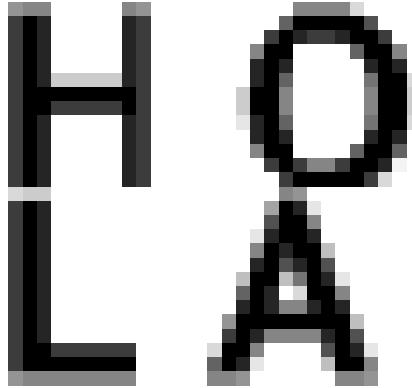
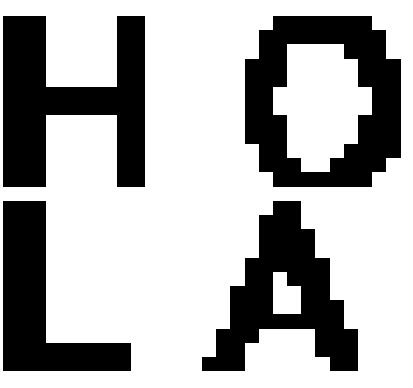
Este algoritmo utilizará la imagen de Lena de dimensiones 256x256, para la marca de agua se usará el logotipo de hola. Las pruebas se harán para diferentes valores de T, que es el umbral.

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Como datos que se recogerán se encuentran la imagen marcada, la marca extraída, los valores de PSNR, el número de errores de la marca extraída y los tiempos de computación en la inserción y extracción.

Parámetros:	T = 5
	
Imagen Original	Imagen Marcada
	PSNR: 50.4282 dB
	Calidad: Muy Buena
	
Marca Original	Marca Extraída
	Número de Errores: 615 de 1024
Tabla 129: Inserción y extracción por el algoritmo SVD basado en la proximidad a un intervalo, T = 5	

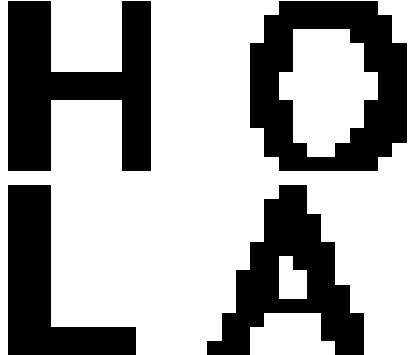
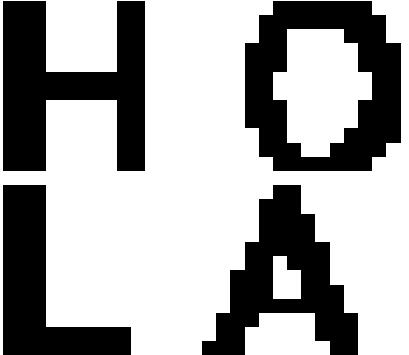
Parámetros:	T = 10
	
Imagen Original	Imagen Marcada
	PSNR: 44.4020 dB
	Calidad: Muy buena
	
Marca Original	Marca Extraída
	Número de Errores: 293 de 1024
Tabla 130: Inserción y extracción por el algoritmo SVD basado en la proximidad a un intervalo, T = 10	

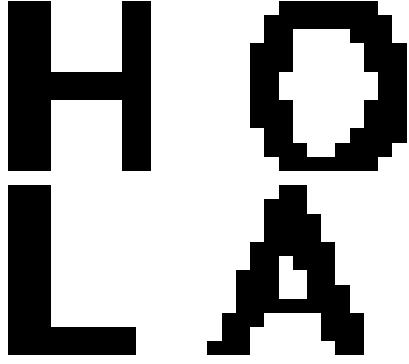
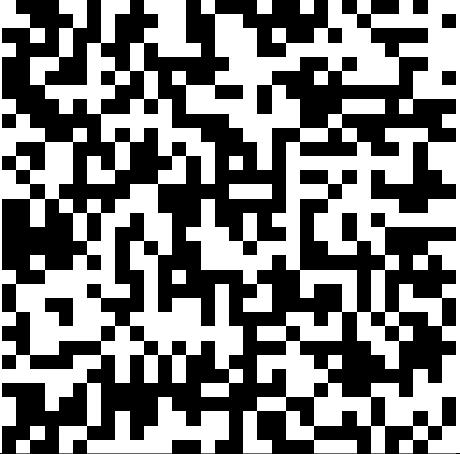
Parámetros:	T = 60
	
Imagen Original	Imagen Marcada
	PSNR: 28.7645 dB
	Calidad: Regular
	
Marca Original	Marca Extraída
	Número de Errores: 211 de 1024
Tabla 131: Inserción y extracción por el algoritmo SVD basado en la proximidad a un intervalo, T = 60	

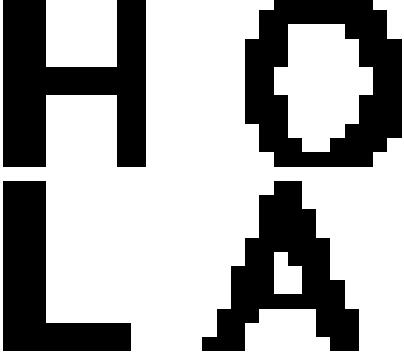
Parámetros:	T = 100
Imagen Original	Imagen Marcada
	PSNR: 24.2885 dB
	Calidad: Muy Mala
Marca Original	Marca Extraída
	Número de Errores: 211 de 1024
Tabla 132: Inserción y extracción por el algoritmo SVD basado en la proximidad a un intervalo, T = 100	

7.3.6.5.2. Pruebas de ataques

Se hará una prueba de cada ataque de los definidos, en ellas se comprobará la resistencia del algoritmo a los mismos, para estas pruebas se cogerá la imagen marcada que mejores resultados dio en las pruebas anteriores.

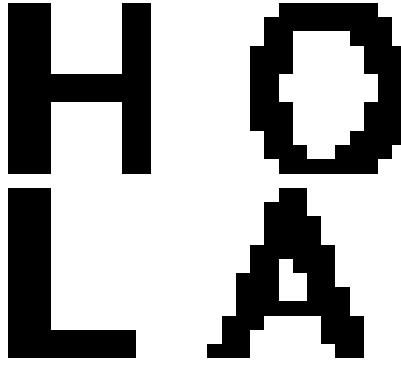
Nombre del Ataque:	Compresión JPEG, 60%
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 211 de 1024
Tabla 133: Ataque compresión JPEG 60% algoritmo SVD basado en la proximidad a un intervalo	

Nombre del Ataque:	Inserción de ruido gaussiano
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 620 de 1024
Tabla 134: Ataque inserción ruido gaussiano algoritmo SVD en la proximidad a un intervalo	

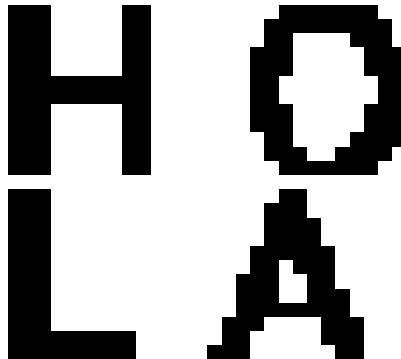
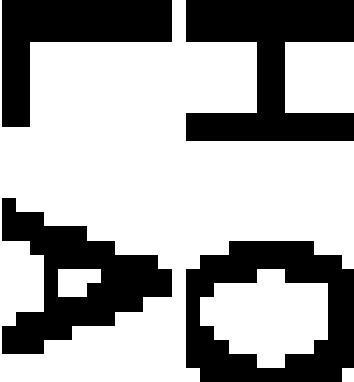
Nombre del Ataque:	Aplicación de un filtro de paso bajo basado en la media
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 540 de 1024
Tabla 135: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo SVD basado en la proximidad a un intervalo	

Nombre del Ataque:	Recortado de la imagen marcada
	
Imagen Marcada	Imagen Atacada
	Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben tener las mismas dimensiones.
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: No Necesario

Tabla 136: Ataque recortado de la imagen marcada algoritmo SVD basado en la proximidad a un intervalo

Nombre del Ataque:	Escalado de la imagen marcada, doble tamaño
	
Imagen Marcada	Imagen Atacada
	Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben tener las mismas dimensiones.
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: No Necesario

[Tabla 137: Ataque escalado de la imagen marcada algoritmo SVD basado en el proximidad a un intervalo](#)

Nombre del Ataque:	Rotación de la imagen marcada, 90°
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
Número de Errores: 446 de 1024	
Tabla 138: Ataque rotación de la imagen marcada algoritmo SVD basado en la proximidad a un intervalo	

7.3.6.5.3. Pruebas de tiempos de ejecución

Por último, se comprobará la eficiencia del algoritmo en tiempo de computación, para ello se cogerá una media de diez ejecuciones del mismo.

Algoritmo:	SVD basado en la proximidad a un intervalo	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	0.3900	0.2652
Ejecución 2	0.3900	0.2340
Ejecución 3	0.3432	0.3588
Ejecución 4	0.3900	0.2964
Ejecución 5	0.3744	0.2808
Ejecución 6	0.4056	0.3120
Ejecución 7	0.4056	0.2496
Ejecución 8	0.3588	0.1716
Ejecución 9	0.4212	0.2496
Ejecución 10	0.3432	0.2496
Media	0.3822 seg.	0.2668 seg.

Tabla 139: Tiempos de ejecución del algoritmo SVD basado en la proximidad a un intervalo

7.3.6.5.4. Conclusiones individuales

Después del estudio individual que se acaba de realizar se puede concluir que:

- En este algoritmo la extracción está en función del parámetro T, un valor de **T bajo** hace que la imagen marcada tenga una **calidad muy buena** pero la **extracción** de la marca de agua es **mala**, en cambio, un valor de **T alto** hace que la imagen marcada tenga una **calidad regular o mala** y una **extracción medio buena**, aquí como en los casos anteriores pasa también que el número de errores parece quedarse fijo.
- En cuanto a los ataques:
 - Es resistente a: compresión JPEG al 60%, rotación.
 - Es vulnerable a: inserción de ruido gaussiano, filtro de paso bajo de la media.
 - No es aplicable a: recortado, escalado.

7.3.7. Algoritmo basado en LSB

7.3.7.1. Pruebas de parámetros

Este algoritmo utilizará la imagen de Lena de dimensiones 512x512, para la marca de agua se usará el logotipo del copyright normal.

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Como datos que se recogerán se encuentran las imágenes marcadas, las marcas extraídas, los valores de PSNR, el número de errores de la marca extraída y los tiempos de computación en la inserción y extracción.

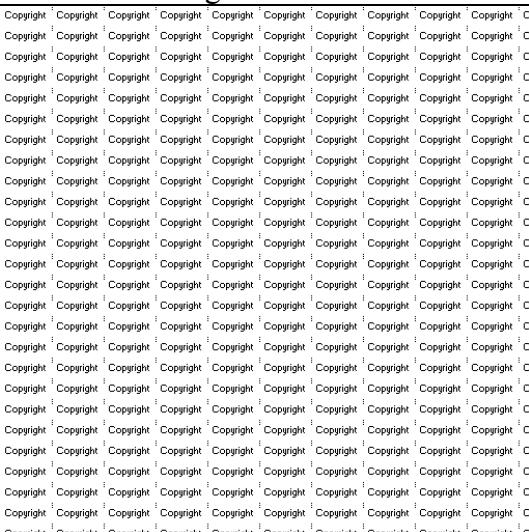
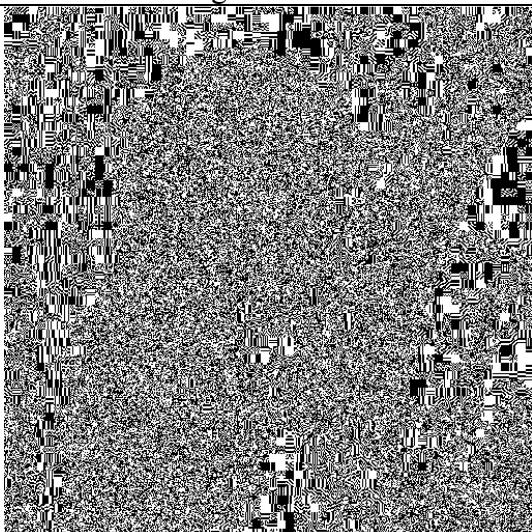
Parámetros:	Ninguno
	
Imagen Original	Imagen Marcada
PSNR:	50.8680 dB
Calidad:	Muy Buena
Marca Original	Marca Extraída (repetición de la marca)
	Número de Errores: 0 de 262144
Copyright	

Tabla 140: Inserción y extracción por el algoritmo LSB

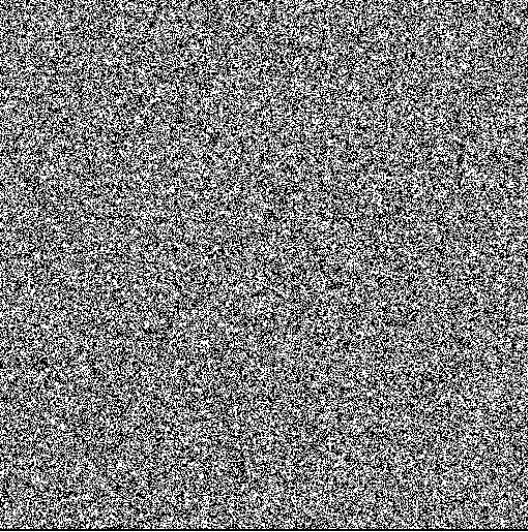
7.3.7.2. Pruebas de ataques

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Se hará una prueba de cada ataque de los definidos, en ellas se comprobará la resistencia del algoritmo a los mismos, para estas pruebas se cogerá la imagen marcada que mejores resultados dio en las pruebas anteriores.

Nombre del Ataque:	Compresión JPEG, 60%
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 131481 de 262144
Tabla 141: Ataque compresión JPEG 60% algoritmo LSB	

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del Ataque:	Inserción de ruido gaussiano
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 130814 de 262144
Tabla 142: Ataque inserción ruido gaussiano algoritmo LSB	

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

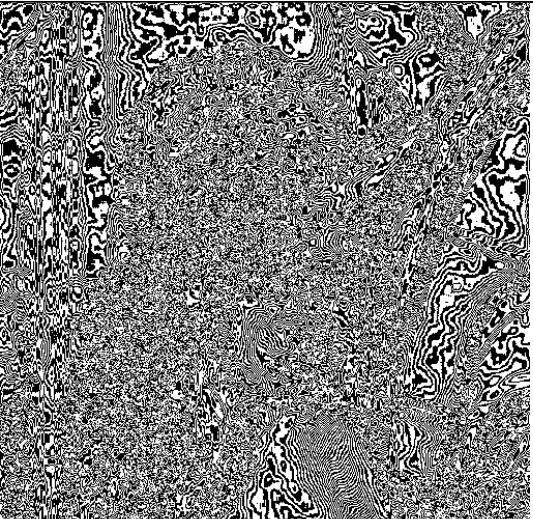
Nombre del Ataque:	Aplicación de un filtro de paso bajo basado en la media
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 133433 de 262144

Tabla 143: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo LSB

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

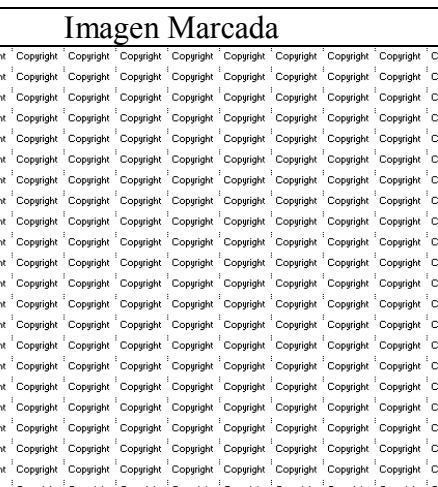
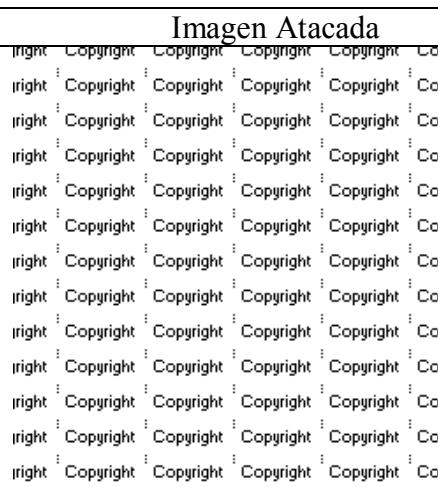
Nombre del Ataque:	Recortado de la imagen marcada
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 18925 de 90601

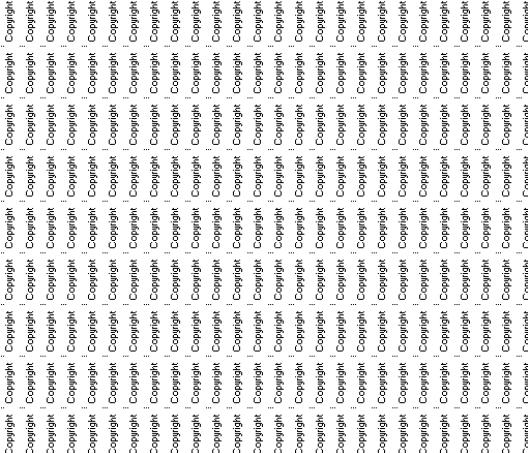
Tabla 144: Ataque recortado de la imagen marcada algoritmo LSB

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del Ataque:	Escalado de la imagen marcada, doble tamaño
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 128090 de 262144

Tabla 145: Ataque escalado de la imagen marcada algoritmo LSB

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del Ataque:	Rotación de la imagen marcada, 90°
	
Imagen Marcada	Imagen Atacada
	
Marca Extraída	Marca Extraída del Ataque
	Número de Errores: 49954 de 262144
Tabla 146: Ataque rotación de la imagen marcada algoritmo LSB	

7.3.7.3. Pruebas de tiempos de ejecución

Por último, se comprobará la eficiencia del algoritmo en tiempo de computación, para ello se cogerá una media de diez ejecuciones del mismo.

Algoritmo:	LSB	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	0.6084	0.4992
Ejecución 2	0.7020	0.5304
Ejecución 3	0.7332	0.5148
Ejecución 4	0.6864	0.4680
Ejecución 5	0.6084	0.5148
Ejecución 6	0.7020	0.4524
Ejecución 7	0.7332	0.4680
Ejecución 8	0.7176	0.4836
Ejecución 9	0.7176	0.4992
Ejecución 10	0.6864	0.4836
Media	0.6895 seg.	0.4914 seg.

Tabla 147: Tiempos de ejecución del algoritmo LSB

7.3.7.4. Conclusiones individuales

Después del estudio individual que se acaba de realizar se puede concluir que:

- En este algoritmo la extracción se realiza sin influir **ningún parámetro**, la imagen marcada tiene una **calidad muy buena** y la **extracción** de la marca de agua es **bueno** ya que no comete **ningún error**.
- En cuanto a los ataques:
 - Es resistente a: recortado, rotación.
 - Es vulnerable a: compresión JPEG al 60%, inserción de ruido gaussiano, filtro de paso bajo de la media.
 - No es aplicable a: escalado.

7.3.8. Algoritmo basado en las Secuencias Caóticas

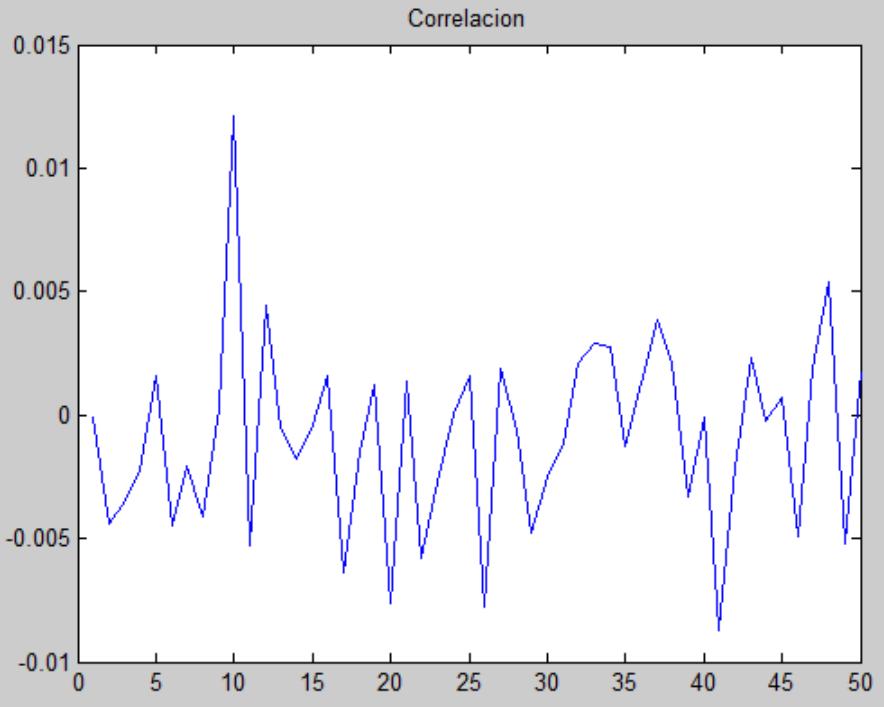
7.3.8.1. Primer Método: Secuencias Caóticas

7.3.8.1.1. Pruebas de parámetros

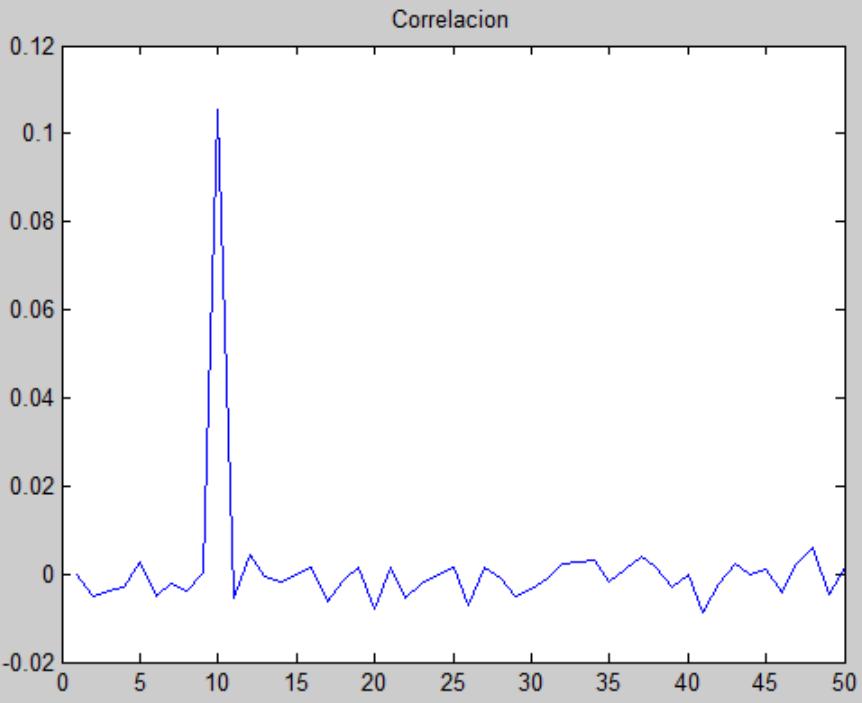
Este algoritmo utilizará la imagen de Lena de dimensiones 256x256, para la marca de agua se usará una secuencia pseudoaleatoria del mismo tamaño que la imagen original. Las pruebas se harán para diferentes valores de alpha.

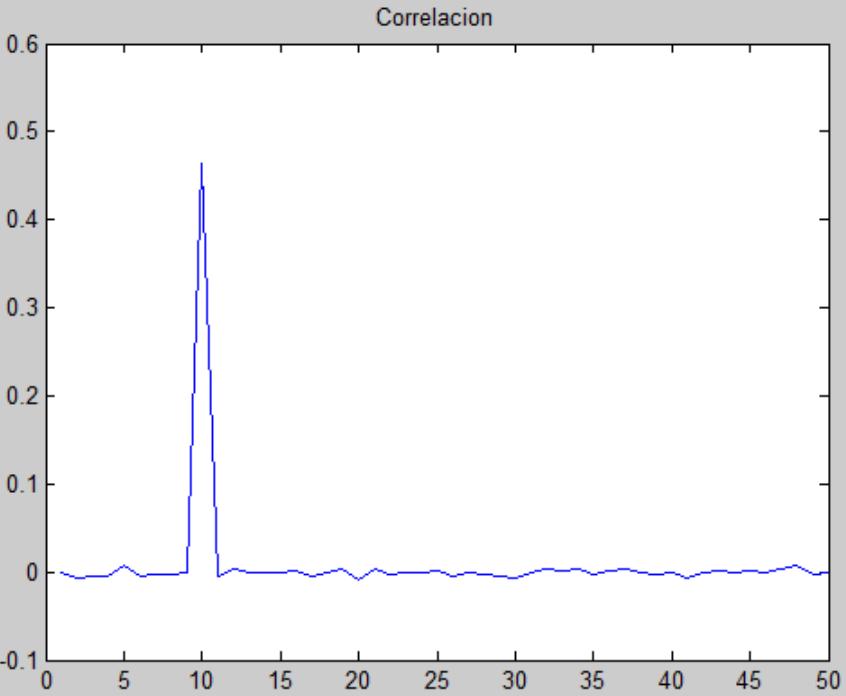
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Como datos que se recogerán se encuentran la imagen marcada, la gráfica de correlación, los valores de PSNR y los tiempos de computación en la inserción y extracción.

Parámetros:	alpha = 1
	
Imagen Original	Imagen Marcada
	PSNR: 51.1719 dB
	Calidad: Muy Buena
	
Correlación	
Número de Errores:	No Necesario
Tabla 148: Inserción y extracción por el algoritmo basado en secuencias caóticas, alpha = 1	

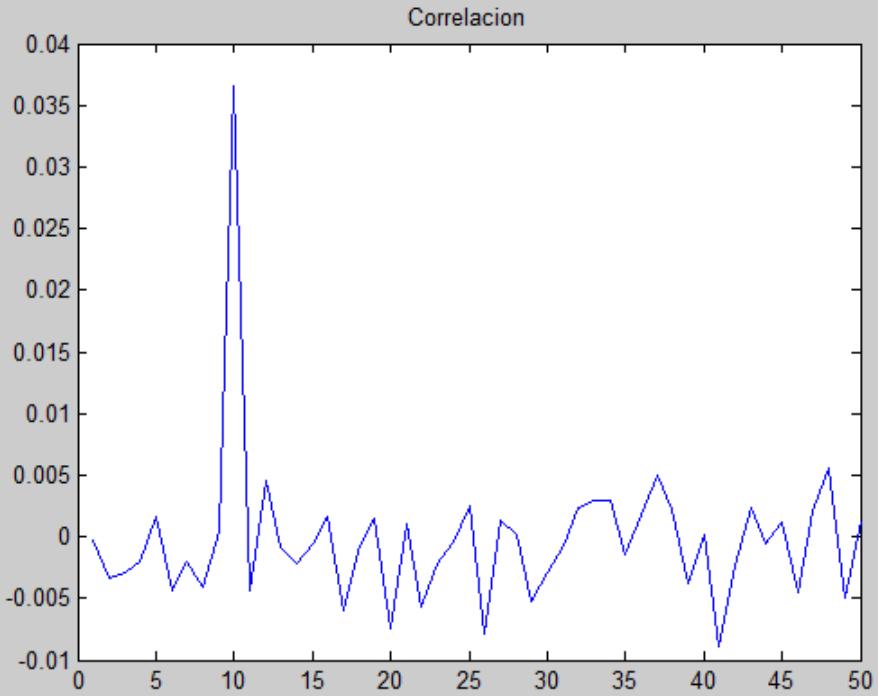
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Parámetros:	alpha = 10
	
Imagen Original	Imagen Marcada
	PSNR: 31.1719 dB
	Calidad: Regular
	
Correlación	
Número de Errores:	No Necesario
Tabla 149: Inserción y extracción por el algoritmo basado en secuencias caóticas, alpha = 10	

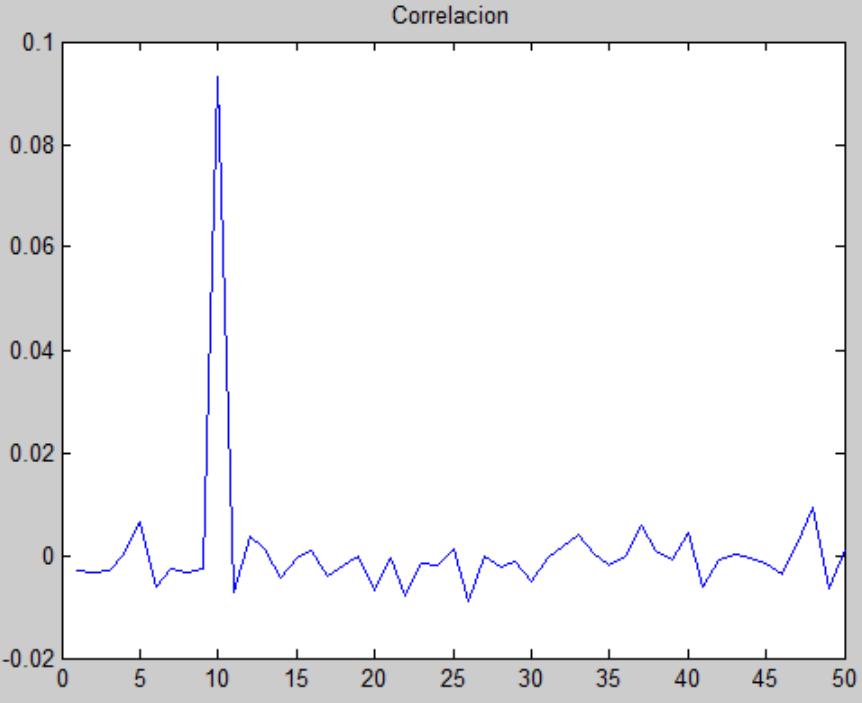
Parámetros:	alpha = 50																						
																							
Imagen Original	Imagen Marcada																						
	PSNR: 17.1925 dB																						
	Calidad: Muy Mala																						
 <p>Correlacion</p> <table border="1"> <tr> <td>0</td><td>5</td><td>10</td><td>15</td><td>20</td><td>25</td><td>30</td><td>35</td><td>40</td><td>45</td><td>50</td> </tr> <tr> <td>0.6</td><td>0.5</td><td>0.4</td><td>0.3</td><td>0.2</td><td>0.1</td><td>0</td><td>-0.1</td><td></td><td></td><td></td> </tr> </table>		0	5	10	15	20	25	30	35	40	45	50	0.6	0.5	0.4	0.3	0.2	0.1	0	-0.1			
0	5	10	15	20	25	30	35	40	45	50													
0.6	0.5	0.4	0.3	0.2	0.1	0	-0.1																
<p>Correlación</p> <table border="1"> <tr> <td>Número de Errores:</td><td>No Necesario</td></tr> </table>		Número de Errores:	No Necesario																				
Número de Errores:	No Necesario																						
<p>Tabla 150: Inserción y extracción por el algoritmo basado en secuencias caóticas, alpha = 50</p>																							

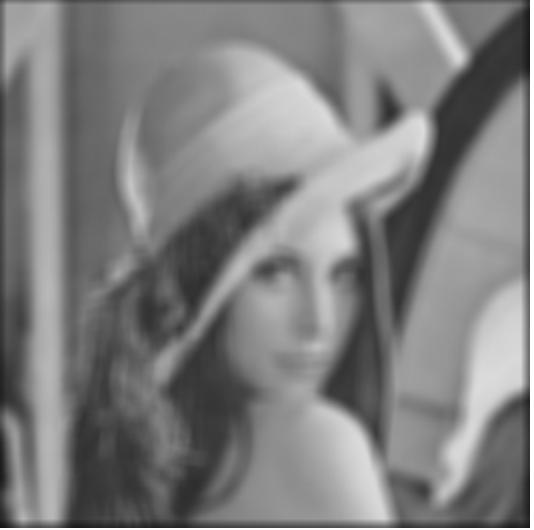
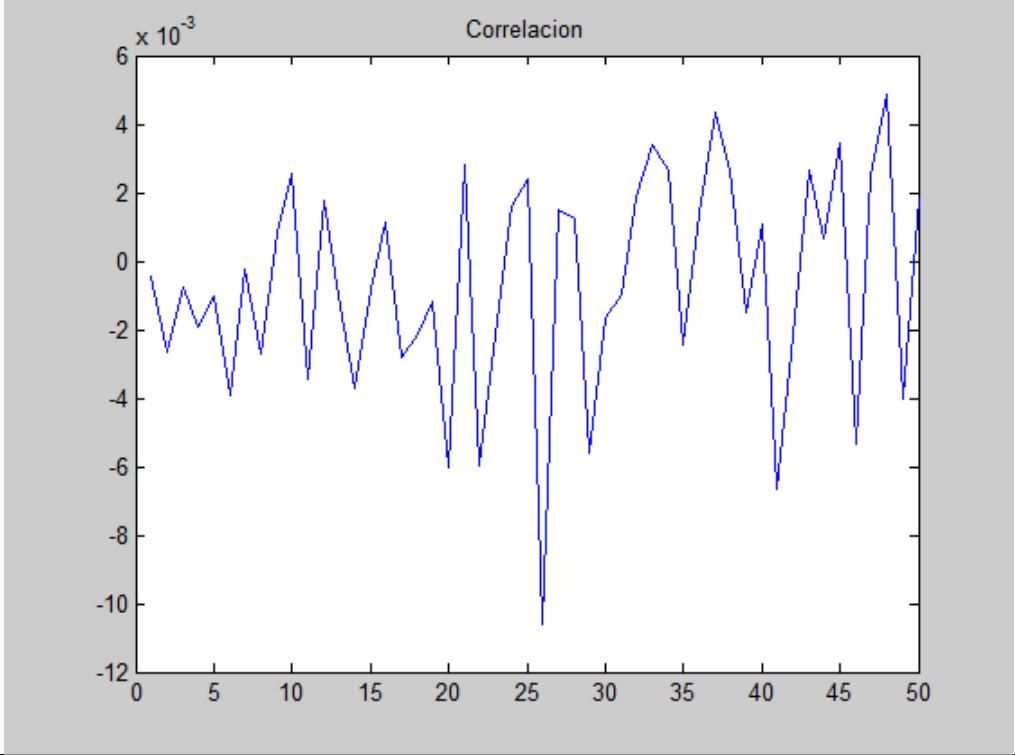
7.3.8.1.2. Pruebas de ataques

Se hará una prueba de cada ataque de los definidos, en ellas se comprobará la resistencia del algoritmo a los mismos, para estas pruebas se cogerá la imagen marcada que mejores resultados dio en las pruebas anteriores.

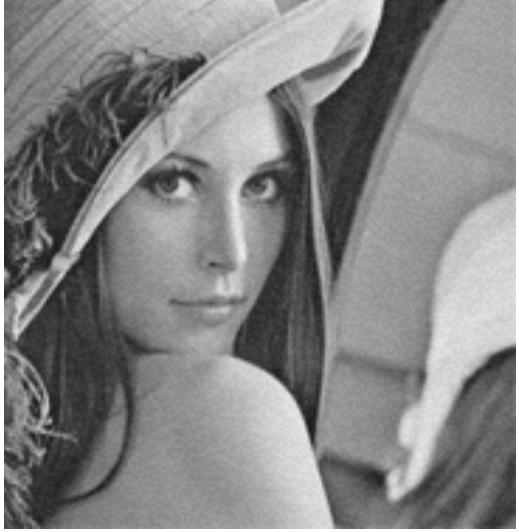
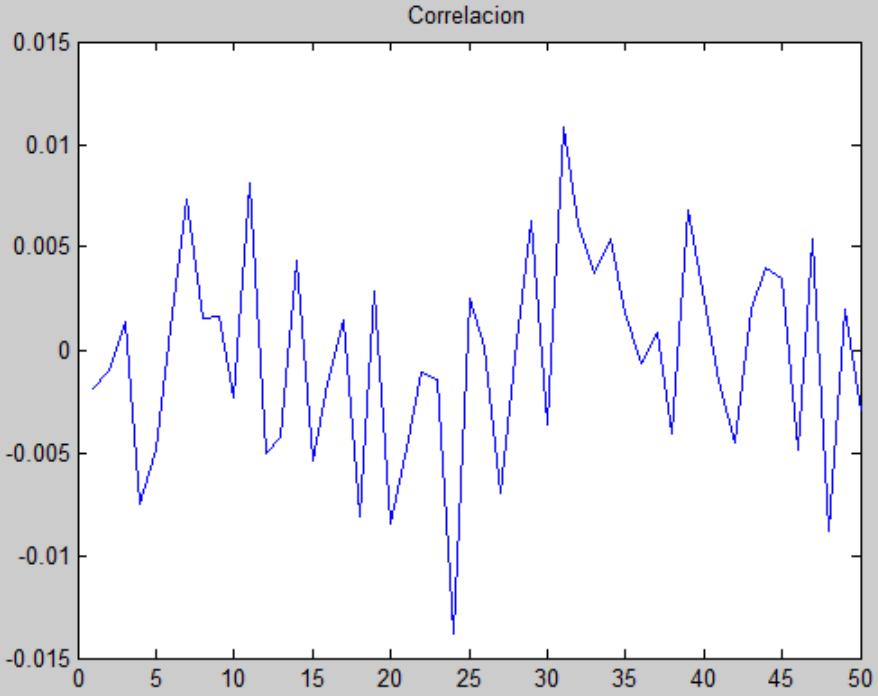
Nombre del ataque:	Compresión JPEG, 60%
	
Imagen Marcada	Imagen Atacada
	
Correlación	
Número de Errores:	No Necesario
Tabla 151: Ataque compresión JPEG 60% algoritmo basado en secuencias caóticas	

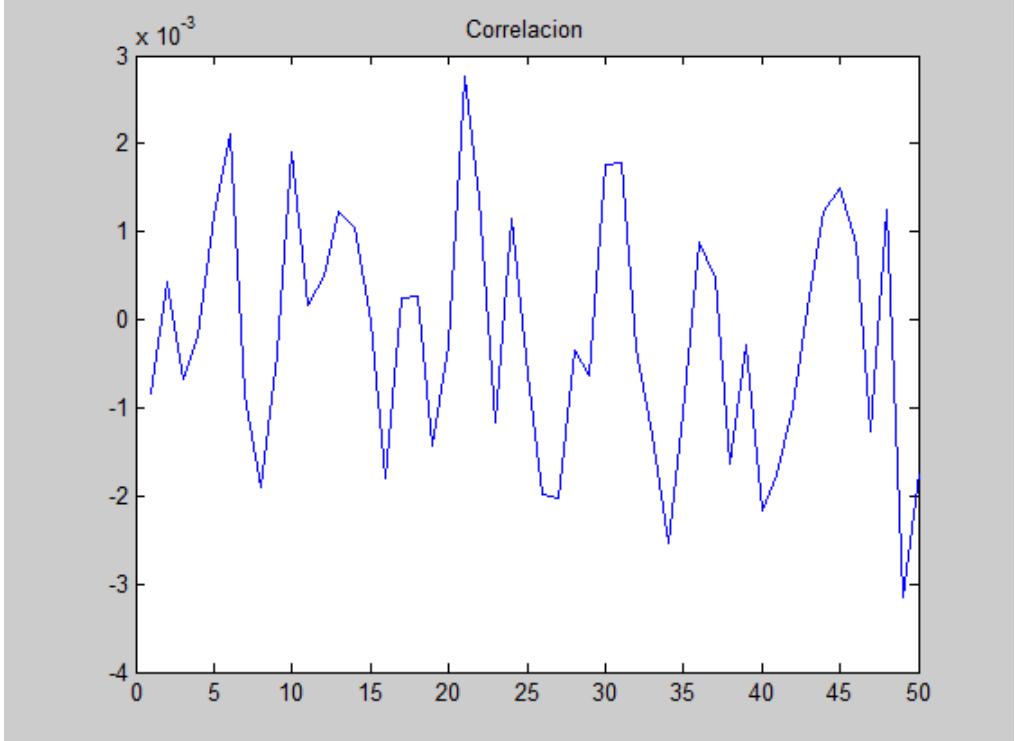
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

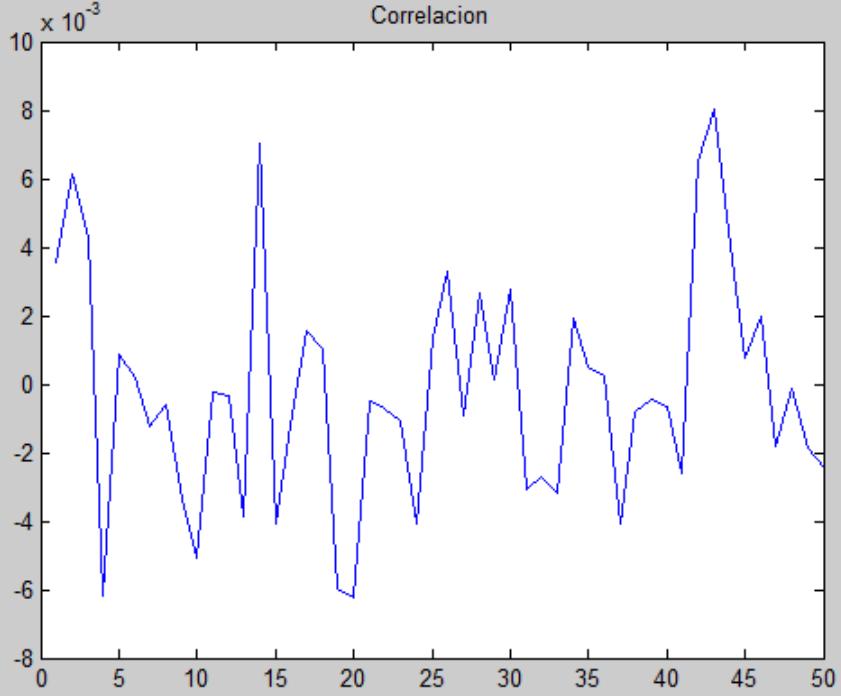
Nombre del ataque:	Inserción de ruido gaussiano
	
Imagen Marcada	Imagen Atacada
	
Correlación	
Número de Errores:	No Necesario
Tabla 152: Ataque inserción ruido gaussiano algoritmo basado en secuencias caóticas	

Nombre del ataque:	Aplicación de un filtro de paso bajo basado en la media
	
Imagen Marcada	Imagen Atacada
	
Correlación	
Número de Errores:	No Necesario
Tabla 153: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo basado en secuencias caóticas	

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del ataque:	Recortado de la imagen marcada
	
Imagen Marcada	Imagen Atacada
	
Correlación	
Número de Errores:	No Necesario
Tabla 154: Ataque recortado de la imagen marcada algoritmo basado en secuencias caóticas	

Nombre del ataque:	Escalado de la imagen marcada, doble tamaño
	
Imagen Marcada	Imagen Atacada
	
Correlación	
Número de Errores:	No Necesario
Tabla 155: Ataque escalado de la imagen marcada algoritmo basado en secuencias caóticas	

Nombre del ataque:	Rotación de la imagen marcada, 90°
	
Imagen Marcada	Imagen Atacada
	
Correlación	
Número de Errores:	No Necesario
Tabla 156: Ataque rotación de la imagen marcada algoritmo basado en secuencias caóticas	

7.3.8.1.3. Pruebas de tiempos de ejecución

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Por último, se comprobará la eficiencia del algoritmo en tiempo de computación, para ello se cogerá una media de diez ejecuciones del mismo.

Algoritmo:	Secuencias Caóticas	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	0.4680	1.6536
Ejecución 2	0.3900	1.6692
Ejecución 3	0.3744	1.7160
Ejecución 4	0.3900	1.6848
Ejecución 5	0.3744	1.6848
Ejecución 6	0.3900	1.7004
Ejecución 7	0.3432	1.7004
Ejecución 8	0.3900	1.7004
Ejecución 9	0.3900	1.6848
Ejecución 10	0.4524	1.6692
Media	0.3962 seg.	1.6864 seg.

Tabla 157: Tiempos de ejecución del algoritmo basado en secuencias caóticas

7.3.8.1.4. Conclusiones individuales

Después del estudio individual que se acaba de realizar se puede concluir que:

- En este algoritmo la extracción dependerá del parámetro alpha, un valor de **alpha bajo** hará que la imagen marcada tenga una **calidad muy buena** pero la **extracción** de la marca de agua **no** será **satisfactoria**, en cambio, un valor de **alpha alto** hará que la imagen marcada tenga una **calidad muy mala** pero la **extracción** de la marca de agua será **muy buena**.
- En cuanto a los ataques:
 - Es resistente a: compresión JPEG al 60%, inserción de ruido gaussiano.
 - Es vulnerable a: filtro de paso bajo de la media, recortado, escalado, rotación.
 - No es aplicable a: .

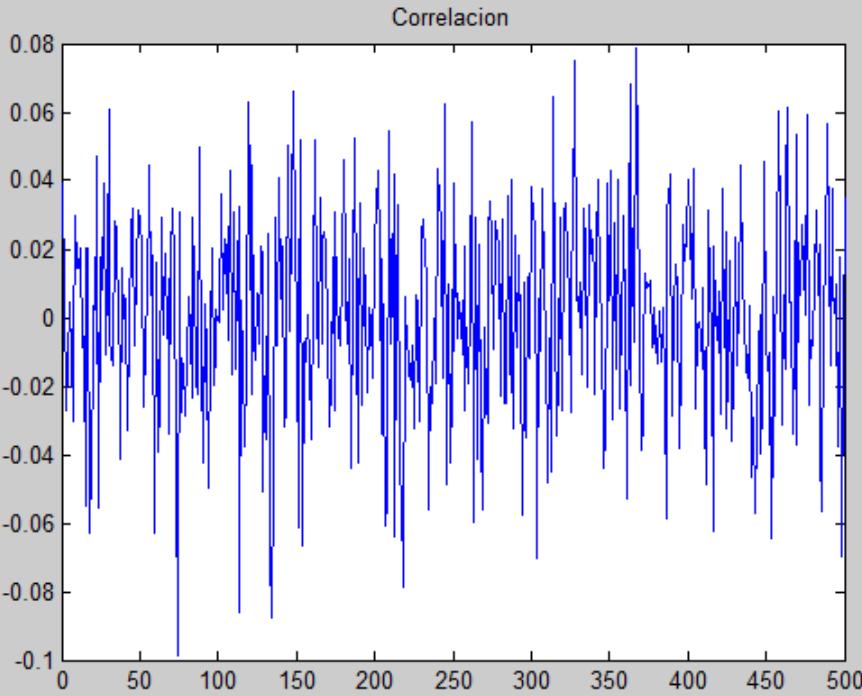
7.3.8.2. Segundo Método: Secuencias Caóticas y DCT

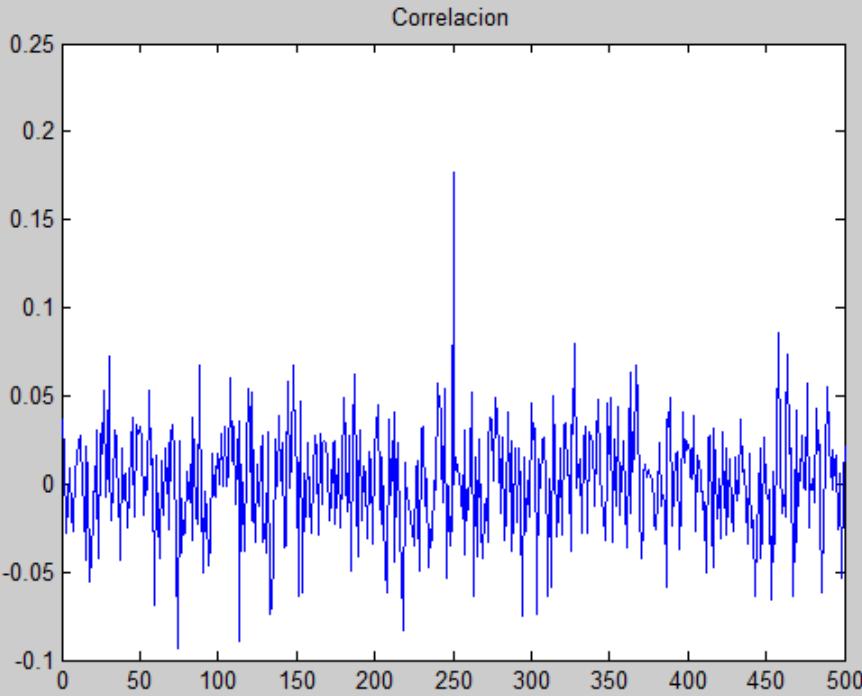
7.3.8.2.1. Pruebas de parámetros

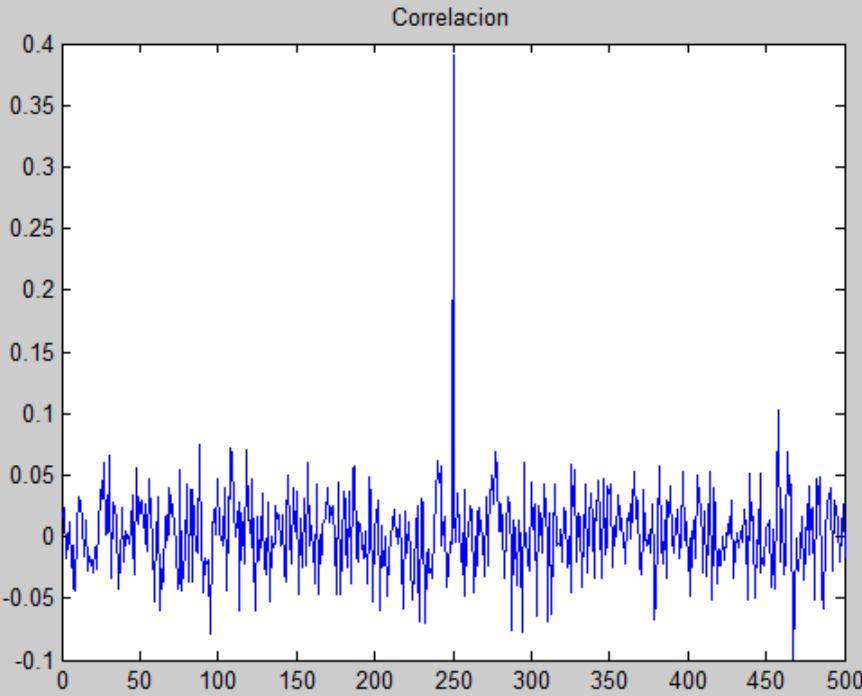
Este algoritmo utilizará la imagen de Lena de dimensiones 256x256, para la marca de agua se usará una secuencia pseudoaleatoria de 1000 elementos, como clave para inicializar la semilla se utilizará el valor 250. Las pruebas se harán para diferentes valores de alpha, que es el factor de ganancia.

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

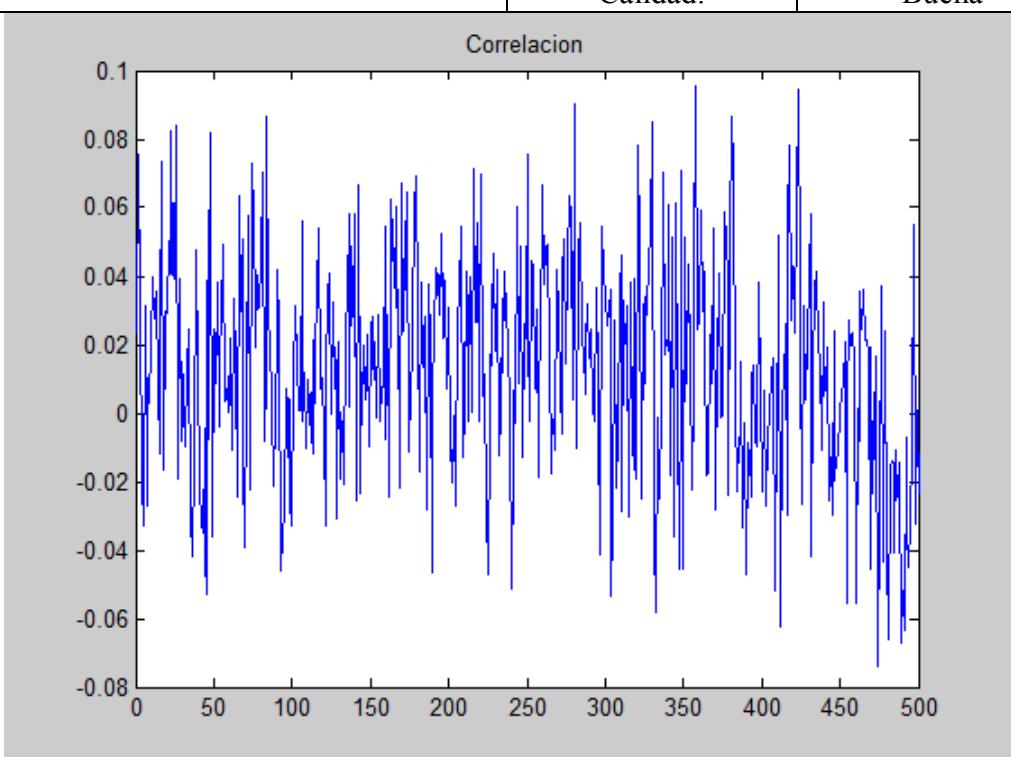
Como datos que se recogerán se encuentran las imágenes marcadas, las marcas extraídas, los valores de PSNR y los tiempos de computación en la inserción y extracción.

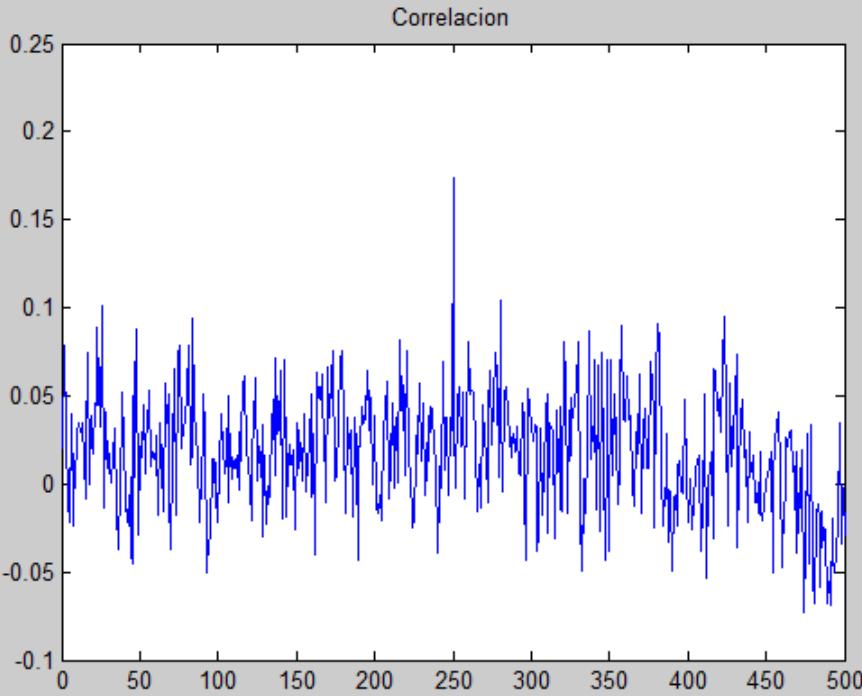
Parámetros:	alpha = 0.1
	
Imagen Original	Imagen Marcada
	PSNR: 47.7156 dB
	Calidad: Buena
	
Correlación	
Número de Errores:	No Necesario
Tabla 158: Inserción y extracción por el algoritmo basado en secuencias caóticas y DCT, alpha = 0.1	

Parámetros:	alpha = 0.3
	
Imagen Original	Imagen Marcada
	PSNR: 38.1732 dB
	Calidad: Regular
 <p>Correlacion</p> <p>Este gráfico muestra la correlación entre los bloques de 8x8 de la imagen marcada. El eje horizontal (X) va de 0 a 500, y el eje vertical (Y) va de -0.1 a 0.25. La correlación es generalmente negativa, con una gran señal positiva en el punto 250.</p>	
Correlación	
Número de Errores:	No Necesario
Tabla 159: Inserción y extracción por el algoritmo basado secuencias caóticas y DCT, alpha = 0.3	

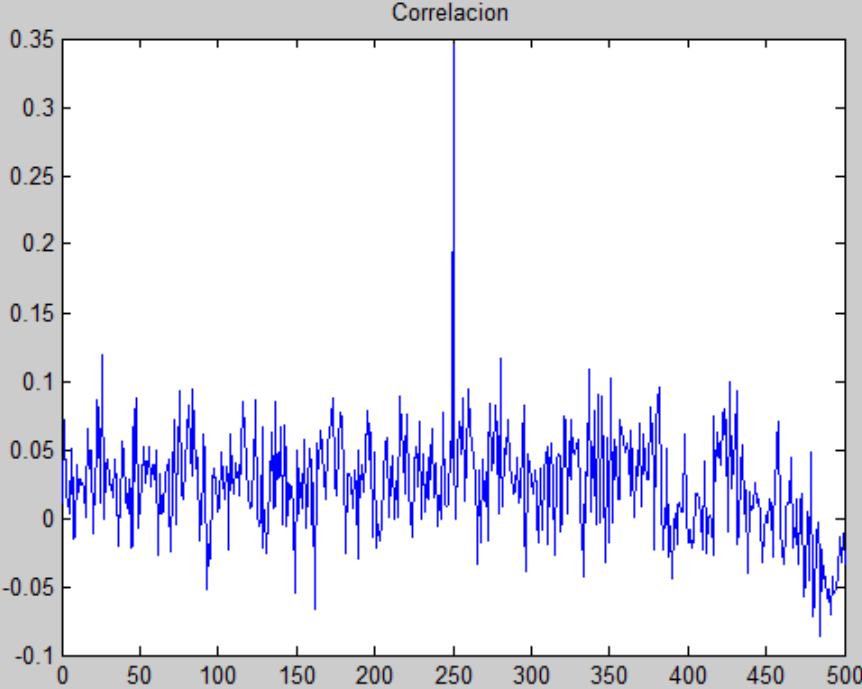
Parámetros:	alpha = 1
	
Imagen Original	Imagen Marcada
	PSNR: 27.7156 dB
	Calidad: Muy Mala
 <p>Correlacion</p> <p>Este gráfico muestra la correlación entre los bloques de 8x8 de la imagen marcada. El eje vertical (y) va de -0.1 a 0.4 con pasos de 0.05. El eje horizontal (x) va de 0 a 500 con pasos de 50. La curva es predominantemente negativa (entre -0.05 y 0.05), con una gran señal de sincronización que alcanza un pico de aproximadamente 0.4 alrededor de x=250.</p>	
Correlación	
Número de Errores:	No Necesario
Tabla 160: Inserción y extracción por el algoritmo basado en secuencias caóticas y DCT, alpha = 1	

Prueba con los algoritmos sin permutación.

Parámetros:	alpha = 0.1
	
Imagen Original	Imagen Marcada
	PSNR: 38.0289 dB
	Calidad: Buena
	
Correlación	
Número de Errores:	No Necesario
Tabla 161: Inserción y extracción por el algoritmo basado en secuencias caóticas y DCT sin permutación, alpha = 0.1	

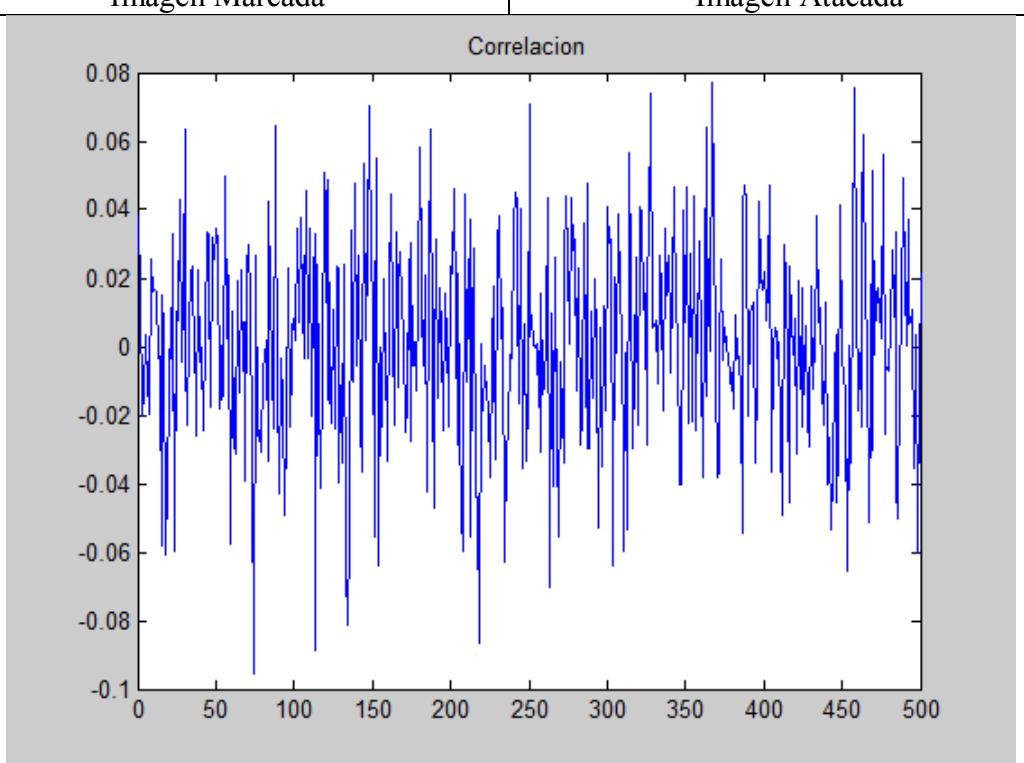
Parámetros:	alpha = 0.3		
			
Imagen Original	Imagen Marcada		
	PSNR: 28.4864 dB		
	Calidad: Regular		
 <p>Correlacion</p> <p>Este gráfico muestra la correlación entre los datos de la imagen original y los marcados. El eje x va de 0 a 500, y el eje y de -0.1 a 0.25. Se observan picos significativos en las posiciones 250 y 300.</p>			
<p>Correlación</p> <table> <tr> <td>Número de Errores:</td> <td>No Necesario</td> </tr> </table>		Número de Errores:	No Necesario
Número de Errores:	No Necesario		
<p>Tabla 162: Inserción y extracción por el algoritmo basado en secuencias caóticas y DCT sin permutación, alpha = 0.3</p>			

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Parámetros:	alpha = 1
	
Imagen Original	Imagen Marcada
	PSNR: 18.0289 dB
	Calidad: Muy Mala
	
Correlación	
Número de Errores:	No Necesario
Tabla 163: Inserción y extracción por el algoritmo basado en secuencias caóticas y DCT sin permutación, alpha = 1	

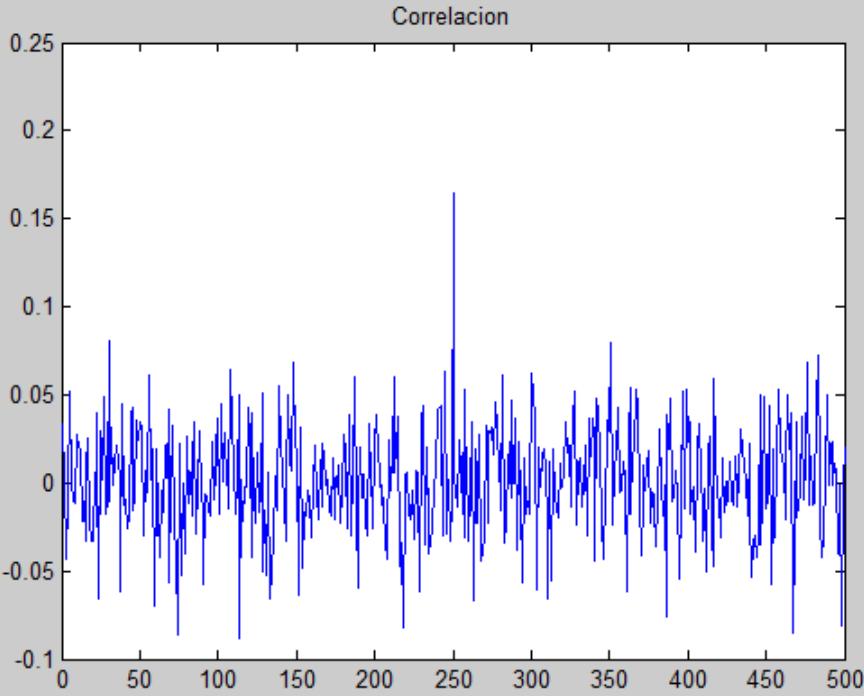
7.3.8.2.2. Pruebas de ataques

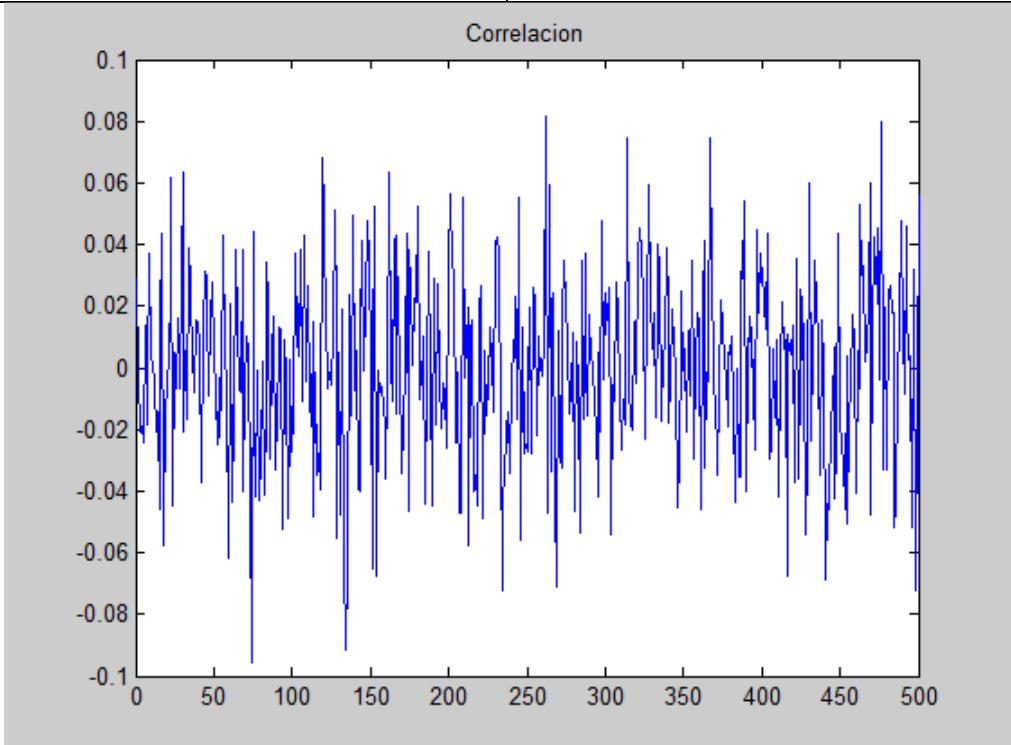
Se hará una prueba de cada ataque de los definidos, en ellas se comprobará la resistencia del algoritmo a los mismos, para estas pruebas se cogerá la imagen marcada que mejores resultados dio en las pruebas anteriores. Se ha utilizado los algoritmos con permutación ya que ofrecen mejores resultados.

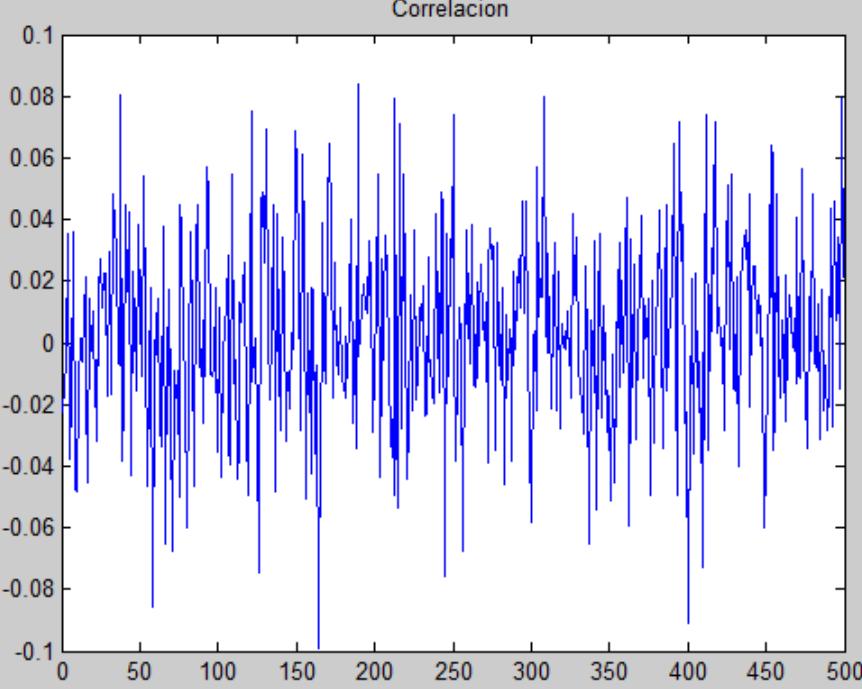
Nombre del ataque:	Compresión JPEG, 60%
	
Imagen Marcada	Imagen Atacada
	
Correlación	
Número de Errores:	No Necesario

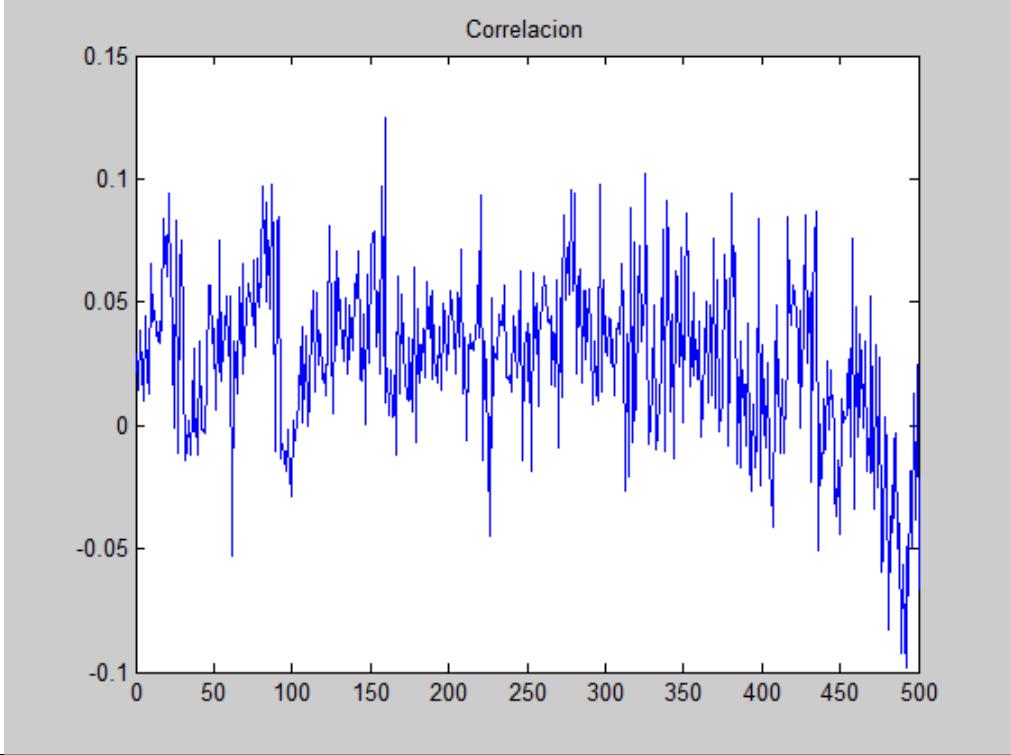
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

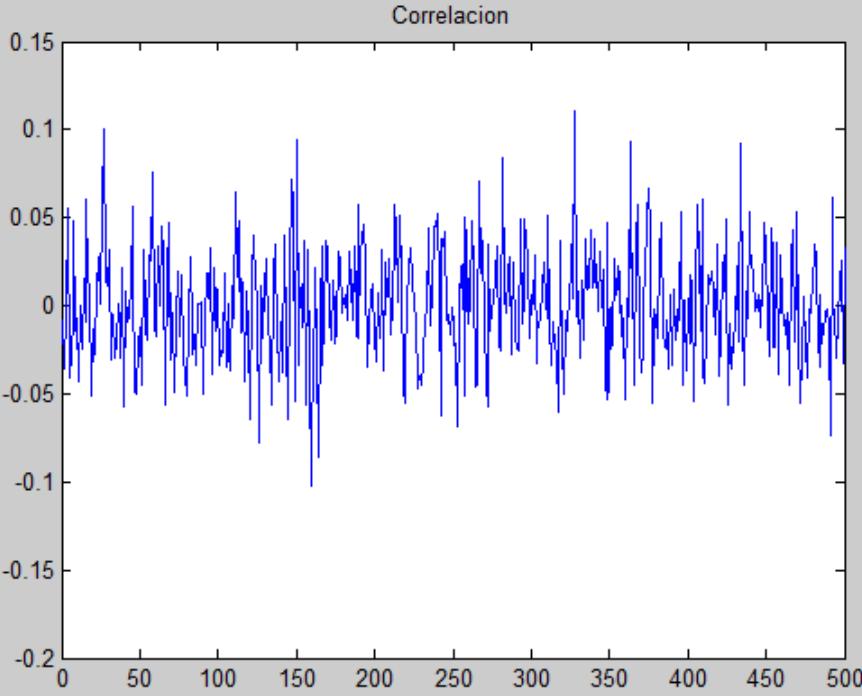
Tabla 164: Ataque compresión JPEG 60% algoritmo basado en secuencias caóticas y DCT

Nombre del ataque:	Inserción de ruido gaussiano
	
Imagen Marcada	Imagen Atacada
	
Correlación	
Número de Errores:	No Necesario
Tabla 165: Ataque inserción ruido gaussiano algoritmo basado en secuencias caóticas y DCT	

Nombre del ataque:	Aplicación de un filtro de paso bajo basado en la media
	
Imagen Marcada	Imagen Atacada
	
Correlación	
Número de Errores:	No Necesario
Tabla 166: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo basado en secuencias caóticas y DCT	

Nombre del ataque:	Recortado de la imagen marcada
	
Imagen Marcada	Imagen Atacada
	
Correlación	
Número de Errores:	No Necesario
Tabla 167: Ataque recortado de la imagen marcada algoritmo basado en secuencias caóticas y DCT	

Nombre del ataque:	Escalado de la imagen marcada, doble tamaño
	
Imagen Marcada	Imagen Atacada
	
Correlación	
Número de Errores:	No Necesario
Tabla 168: Ataque escalado de la imagen marcada algoritmo basado en secuencias caóticas y DCT	

Nombre del ataque:	Rotación de la imagen marcada, 90°
	
Imagen Marcada	Imagen Atacada
	
Correlación	
Número de Errores:	No Necesario
Tabla 169: Ataque rotación de la imagen marcada algoritmo basado en secuencias caóticas y DCT	

7.3.8.2.3. Pruebas de tiempos de ejecución

Por último, se comprobará la eficiencia del algoritmo en tiempo de computación, para ello se cogerá una media de diez ejecuciones del mismo.

Algoritmo:	Secuencias Caóticas y DCT	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	1.3572	0.6864
Ejecución 2	1.3416	0.6708
Ejecución 3	1.3572	0.6864
Ejecución 4	1.3572	0.6864
Ejecución 5	1.3416	0.7020
Ejecución 6	1.3572	0.7020
Ejecución 7	1.3416	0.6864
Ejecución 8	1.3572	0.6864
Ejecución 9	1.3728	0.7020
Ejecución 10	1.3572	0.6864
Media	1.3541 seg.	0.6895 seg.

Tabla 170: Tiempos de ejecución del algoritmo basado en secuencias caóticas y DCT

Algoritmo:	Secuencias Caóticas y DCT (sin permutación)	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	0.5616	0.2964
Ejecución 2	0.5616	0.3120
Ejecución 3	0.5616	0.2964
Ejecución 4	0.5616	0.2964
Ejecución 5	0.5616	0.2964
Ejecución 6	0.5304	0.2808
Ejecución 7	0.5616	0.2964
Ejecución 8	0.5616	0.2964
Ejecución 9	0.5616	0.3120
Ejecución 10	0.5460	0.2808
Media	0.5569 seg.	0.2964 seg.

Tabla 171: Tiempos de ejecución del algoritmo basado en secuencias caóticas y DCT (sin permutación)

7.3.8.2.4. Conclusiones individuales

Después del estudio individual que se acaba de realizar se puede concluir que:

- En este algoritmo la extracción dependerá del parámetro alpha, un valor de **alpha bajo** hará que la imagen marcada tenga una **calidad buena** pero la **extracción no** tendrá **éxito**, en cambio, un valor de **alpha alto** hará que la imagen marcada tenga una **calidad muy mala** pero que la **extracción** de la marca de agua sea **buena**. Estos resultados se han tomado a partir de elegir el algoritmo que ofrece mejores resultados, en este caso, el algoritmo con permutación.
- En cuanto a los ataques:
 - Es resistente a: inserción de ruido gaussiano.
 - Es vulnerable a: compresión JPEG al 60%, filtro de paso bajo de la media, recortado, escalado, rotación.
 - No es aplicable a: .

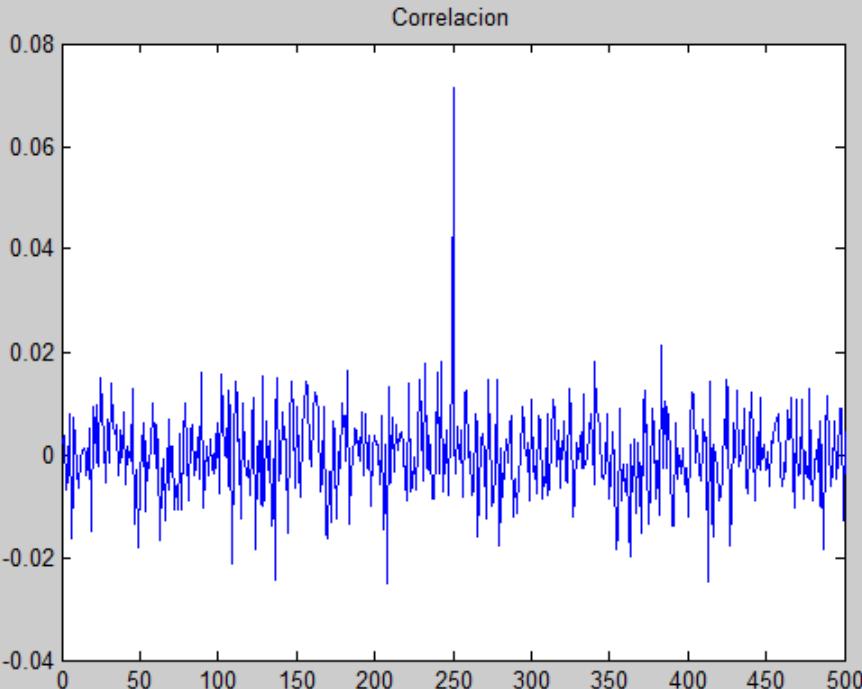
7.3.9. Algoritmo basado en PCA

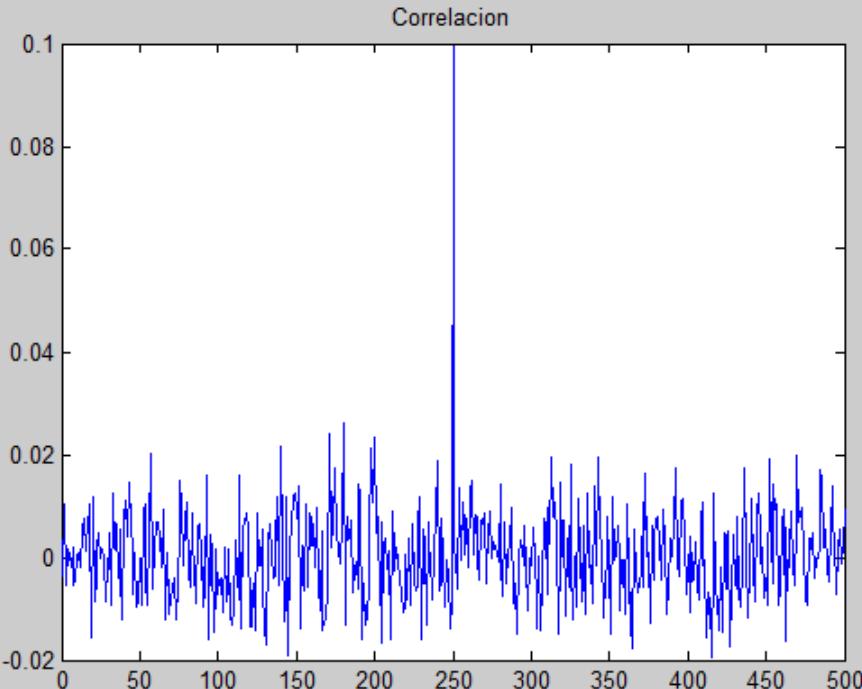
7.3.9.1. Primer Método: PCA

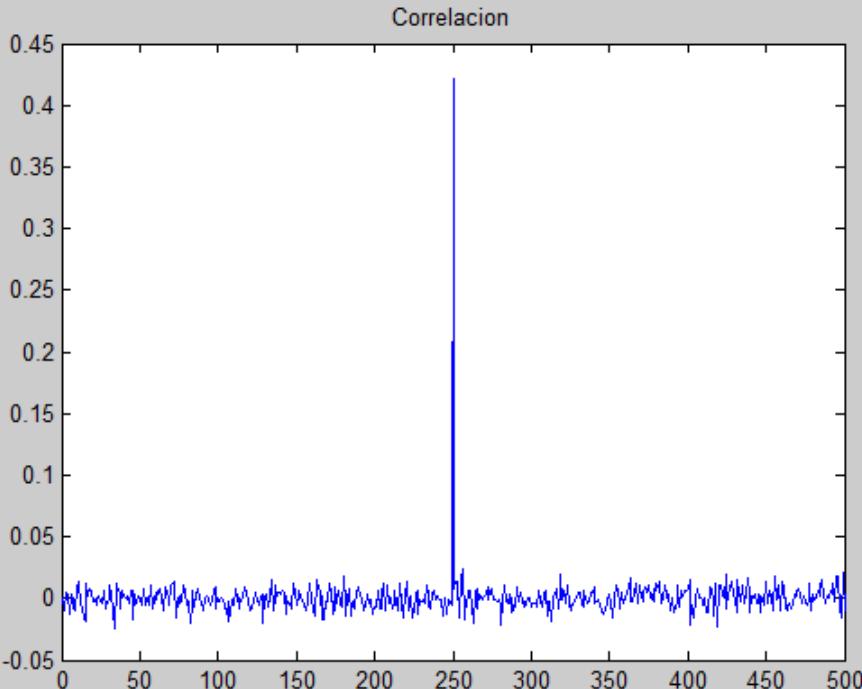
7.3.9.1.1. Pruebas de parámetros

Este algoritmo utilizará la imagen de Lena de dimensiones 256x256, para la marca de agua se usará una secuencia pseudoaleatoria, como clave para inicializar la semilla se utilizará el valor 250. Las pruebas se harán para diferentes valores de alpha.

Como datos que se recogerán se encuentran la imagen marcada, la gráfica de correlación, los valores de PSNR y los tiempos de computación en la inserción y extracción.

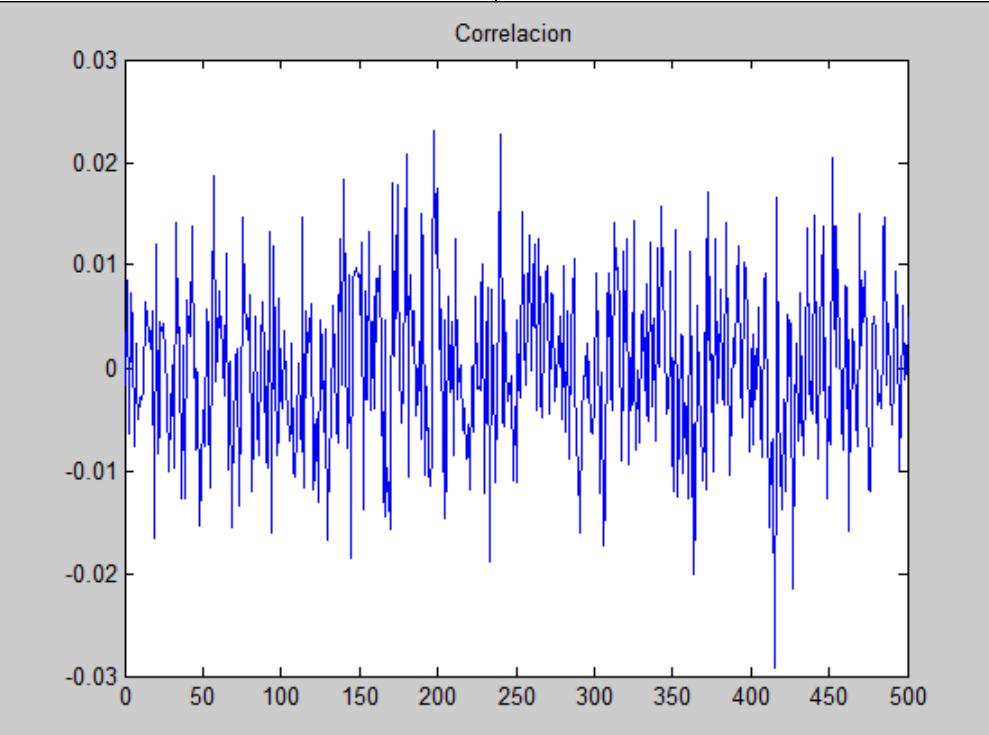
Parámetros:	alpha = 1
	
Imagen Original	Imagen Marcada
	PSNR: 44.8295 dB
	Calidad: Muy Buena
	
Correlación	
Número de Errores:	No Necesario
Tabla 172: Inserción y extracción por el algoritmo PCA, alpha = 1	

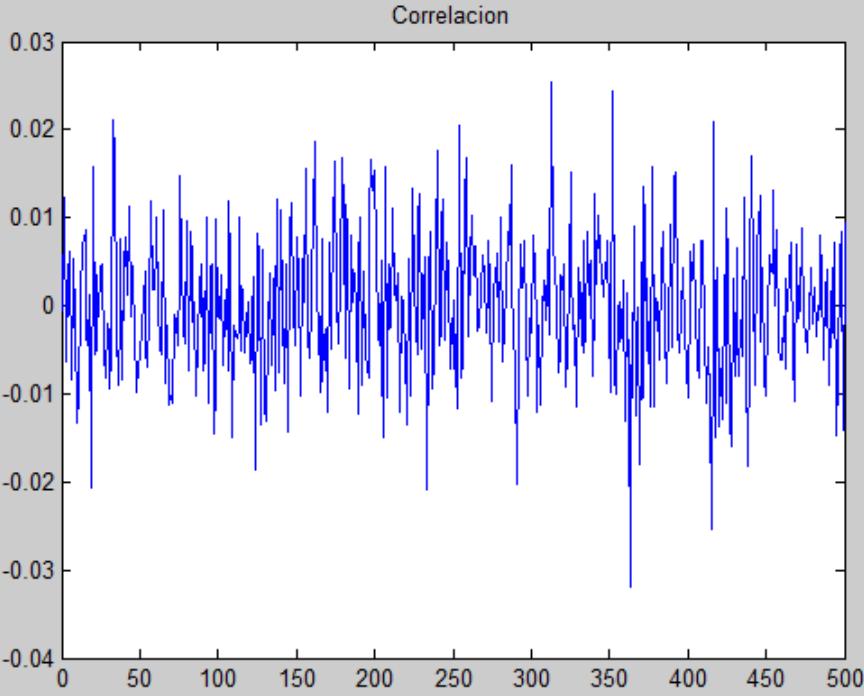
Parámetros:	alpha = 4				
					
Imagen Original	Imagen Marcada				
	PSNR: 32.7883 dB				
	Calidad: Regular				
 <p>Correlacion</p> <table border="1"> <thead> <tr> <th>Número de Errores:</th> <th>Correlación</th> </tr> </thead> <tbody> <tr> <td>No Necesario</td> <td></td> </tr> </tbody> </table>		Número de Errores:	Correlación	No Necesario	
Número de Errores:	Correlación				
No Necesario					
<p>Tabla 173: Inserción y extracción por el algoritmo PCA, alpha = 4</p>					

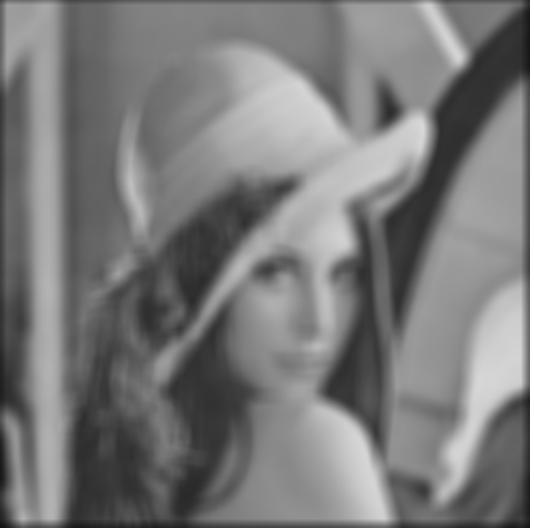
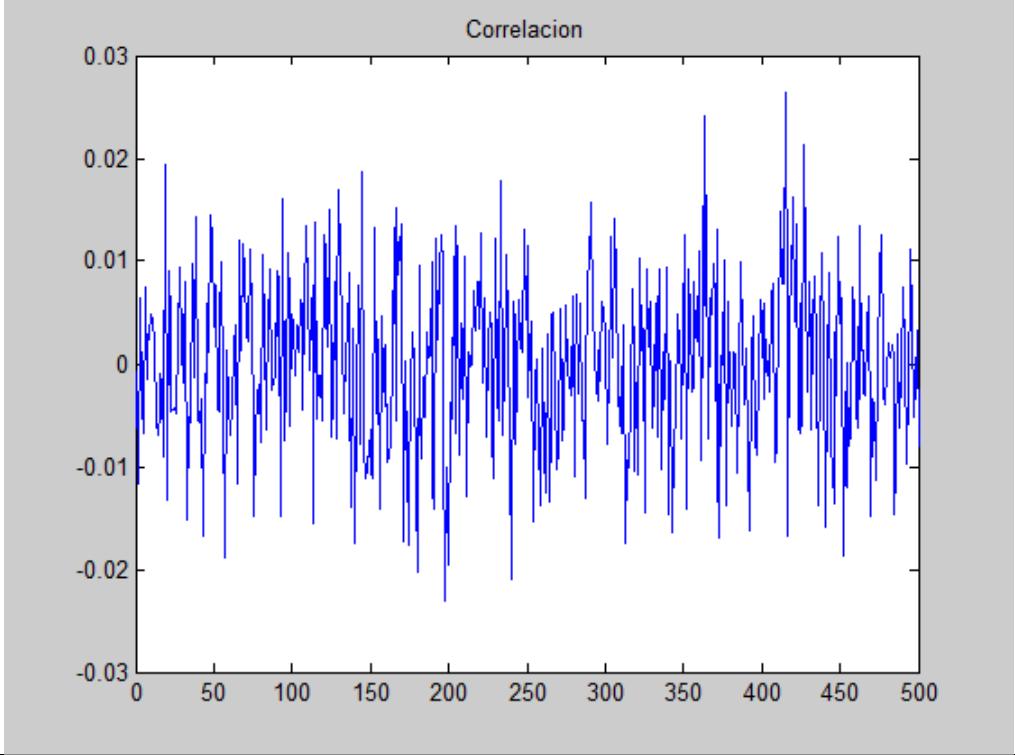
Parámetros:	alpha = 8.5
	
Imagen Original	Imagen Marcada
	PSNR: 26.2411 dB
	Calidad: Mala
 <p>Correlacion</p> <p>Este gráfico muestra la correlación entre los pixeles de una imagen. El eje vertical (y) va de -0.05 a 0.45, y el eje horizontal (x) va de 0 a 500. La mayoría de los valores están alrededor de cero, con una gran señal de sincronización vertical que alcanza un valor de aproximadamente 0.42 en el pixel 250.</p>	
<p>Correlación</p>	
Número de Errores:	No Necesario
Tabla 174: Inserción y extracción por el algoritmo PCA, alpha = 8.5	

7.3.9.1.2. Pruebas de ataques

Se hará una prueba de cada ataque de los definidos, en ellas se comprobará la resistencia del algoritmo a los mismos, para estas pruebas se cogerá la imagen marcada que mejores resultados dio en las pruebas anteriores.

Nombre del ataque:	Compresión JPEG, 60%
	
Imagen Marcada	Imagen Atacada
	
Correlación	
Número de Errores:	No Necesario
Tabla 175: Ataque compresión JPEG 60% algoritmo PCA	

Nombre del ataque:	Inserción de ruido gaussiano
	
Imagen Marcada	Imagen Atacada
	
Correlación	
Número de Errores:	No Necesario
Tabla 176: Ataque inserción ruido gaussiano algoritmo PCA	

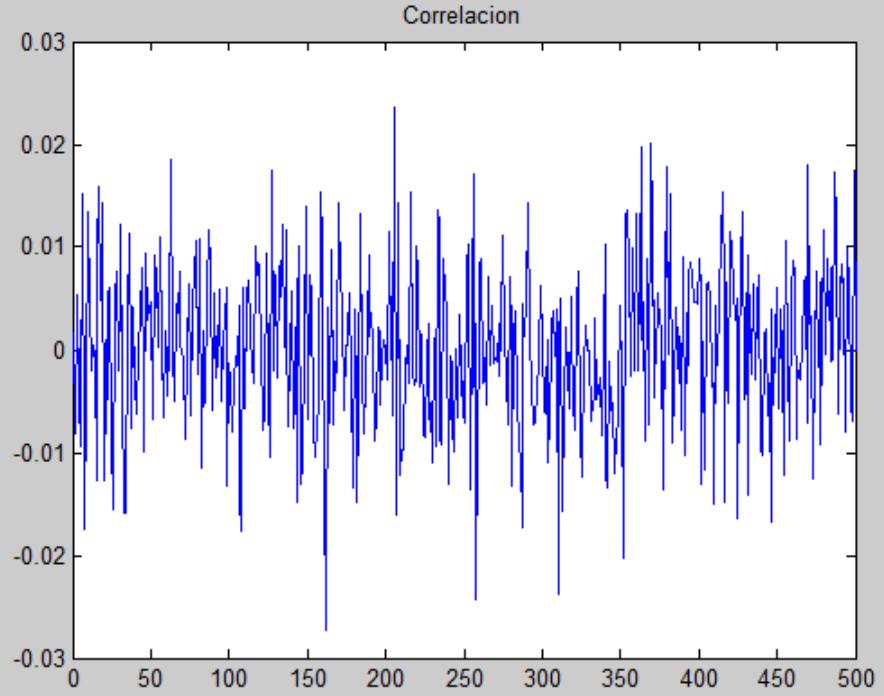
Nombre del ataque:	Aplicación de un filtro de paso bajo basado en la media
	
Imagen Marcada	Imagen Atacada
	
Correlación	
Número de Errores:	No Necesario
Tabla 177: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo PCA	

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del ataque:	Recortado de la imagen marcada
	
Imagen Marcada	Imagen Atacada
Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben tener las mismas dimensiones.	
Correlación	
Número de Errores:	No Necesario
Tabla 178: Ataque recortado de la imagen marcada algoritmo PCA	

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del ataque:	Escalado de la imagen marcada, doble tamaño
	
Imagen Marcada	Imagen Atacada
Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben tener las mismas dimensiones.	
Correlación	
Número de Errores:	No Necesario
Tabla 179: Ataque escalado de la imagen marcada algoritmo PCA	

Nombre del ataque:	Rotación de la imagen marcada, 90°
	
Imagen Marcada	Imagen Atacada
	
Número de Errores:	Correlación No Necesario
Tabla 180: Ataque rotación de la imagen marcada algoritmo PCA	

7.3.9.1.3. Pruebas de tiempos de ejecución

Por último, se comprobará la eficiencia del algoritmo en tiempo de computación, para ello se cogerá una media de diez ejecuciones del mismo.

Algoritmo:	PCA	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	1.9500	1.7472
Ejecución 2	1.8096	1.5756
Ejecución 3	1.9344	1.7472
Ejecución 4	1.8096	1.4352
Ejecución 5	1.7316	1.7004
Ejecución 6	1.7628	1.5288
Ejecución 7	1.8252	1.6224
Ejecución 8	1.9500	1.6224
Ejecución 9	1.6068	1.7316
Ejecución 10	1.7628	1.4040
Media	1.8143 seg.	1.6115 seg.

Tabla 181: Tiempos de ejecución del algoritmo PCA

7.3.9.1.4. Conclusiones individuales

Después del estudio individual que se acaba de realizar se puede concluir que:

- En este algoritmo la extracción dependerá del parámetro alpha, un valor de **alpha bajo** hará que la imagen marcada tenga una **calidad muy buena** y la **extracción** de la marca será **bueno**, en cambio, un valor de **alpha bajo** hará que la imagen marcada tenga una **calidad mala** y la **extracción** de la marca será **muy buena**.
- En cuanto a los ataques:
 - Es resistente a: .
 - Es vulnerable a: compresión JPEG al 60%, inserción de ruido gaussiano, filtro de paso bajo de la media, rotación.
 - No es aplicable a: recortado, escalado.

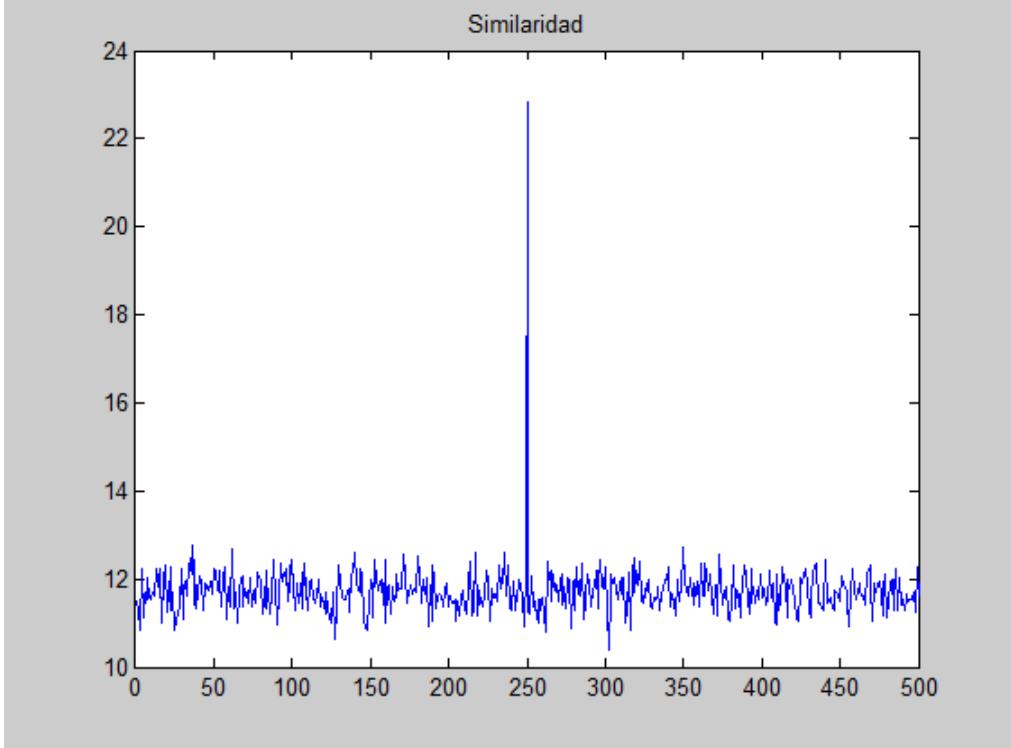
7.3.9.2. Segundo Método: PCA para construir la imagen de referencia

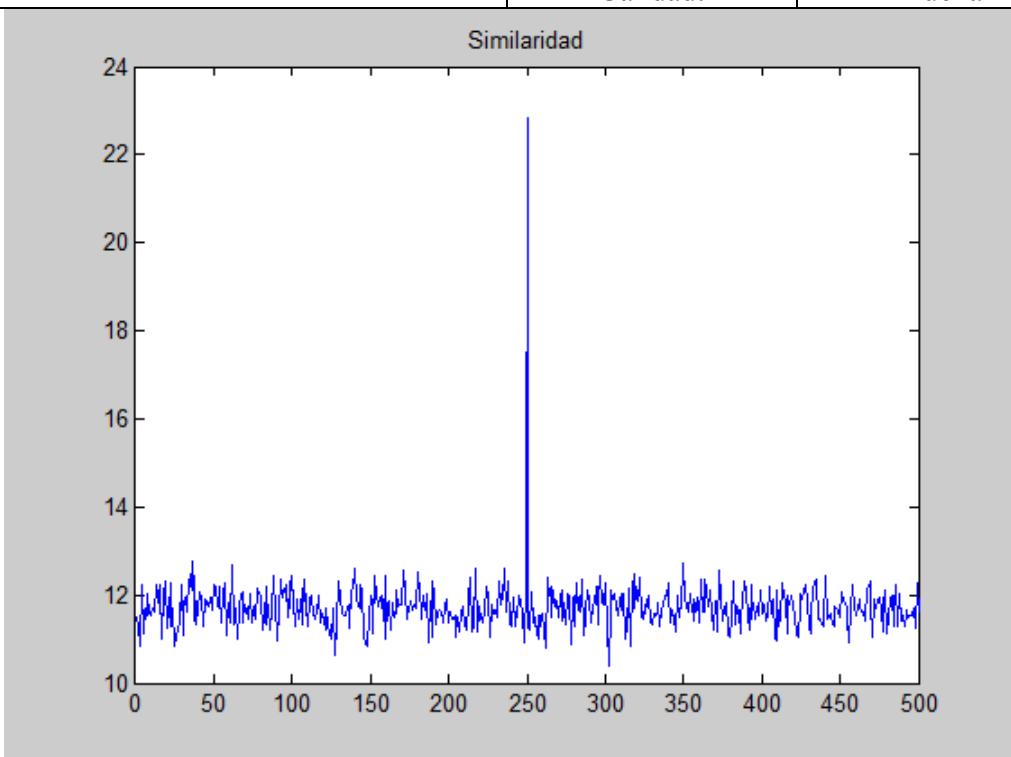
7.3.9.2.1. Pruebas de parámetros

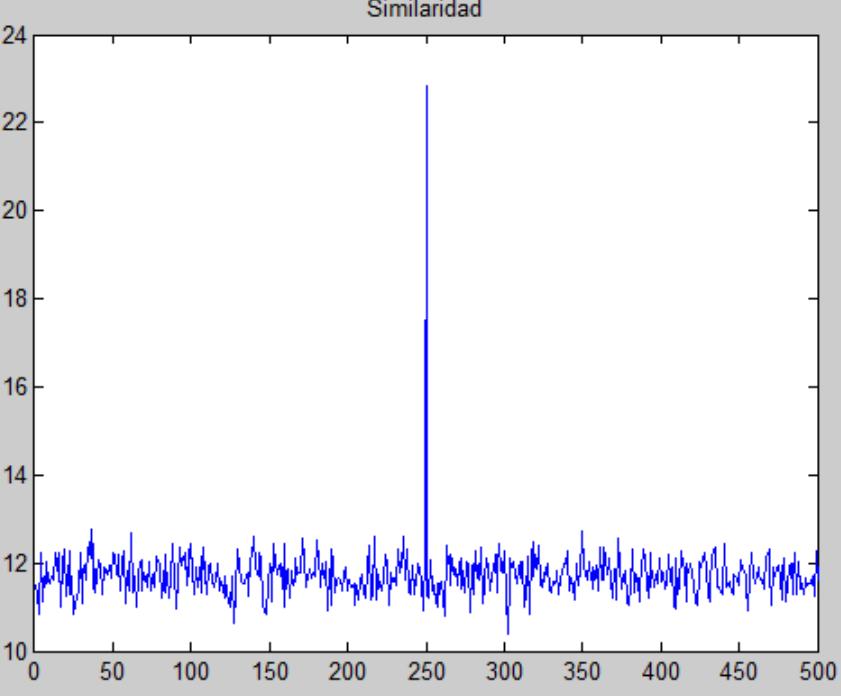
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Este algoritmo utilizará la imagen de Lena de dimensiones 256x256, para la marca de agua se usará una secuencia pseudoaleatoria de 1000 elementos, como clave para inicializar la semilla se utilizará el valor 250. Las pruebas se harán para diferentes valores de los umbrales s y t.

Como datos que se recogerán se encuentran la imagen marcada, la gráfica de similaridad, los valores de PSNR, el número de errores de la marca extraída y los tiempos de computación en la inserción y extracción.

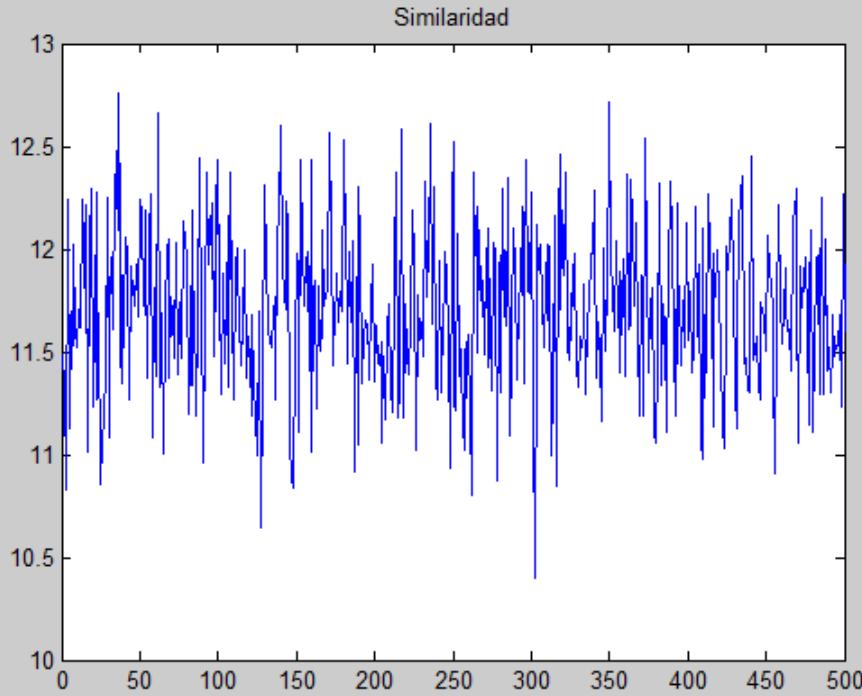
Parámetros:	$s = 1, t = 3$		
			
Imagen Original	Imagen Marcada		
	PSNR: 56.7039 dB		
	Calidad: Muy buena		
 <p>Similaridad</p> <p>Este gráfico muestra la similaridad entre los datos en la posición 250 y los datos en las posiciones cercanas. La escala vertical (Y) va de 10 a 24, y la escala horizontal (X) va de 0 a 500. Una línea azul vertical se extiende desde el punto 250 hasta el valor 23, lo cual indica una alta similaridad en esa posición.</p>			
<p>Similaridad</p> <table> <tr> <td>Número de Errores:</td> <td>0 de 1000</td> </tr> </table>		Número de Errores:	0 de 1000
Número de Errores:	0 de 1000		
<p>Tabla 182: Inserción y extracción por el algoritmo PCA para construir la imagen de referencia, $s=1, t=3$</p>			

Parámetros:	s = 6, t = 8		
			
Imagen Original	Imagen Marcada		
	PSNR: 45.9685 dB		
	Calidad: Buena		
 <p>Similaridad</p> <p>Este gráfico muestra la similaridad entre los datos en función del índice. El eje vertical (Y) va de 10 a 24, y el eje horizontal (X) va de 0 a 500. La curva es generalmente estable en un rango entre 10 y 12, con una excepción significativa: un punto que se eleva abruptamente a un valor de alrededor de 23 a la mitad del eje X (aprox. índice 250).</p>			
<p>Similaridad</p> <table> <tr> <td>Número de Errores:</td> <td>0 de 1000</td> </tr> </table>		Número de Errores:	0 de 1000
Número de Errores:	0 de 1000		
<p>Tabla 183: Inserción y extracción por el algoritmo PCA para construir la imagen de referencia, s = 6, t = 8</p>			

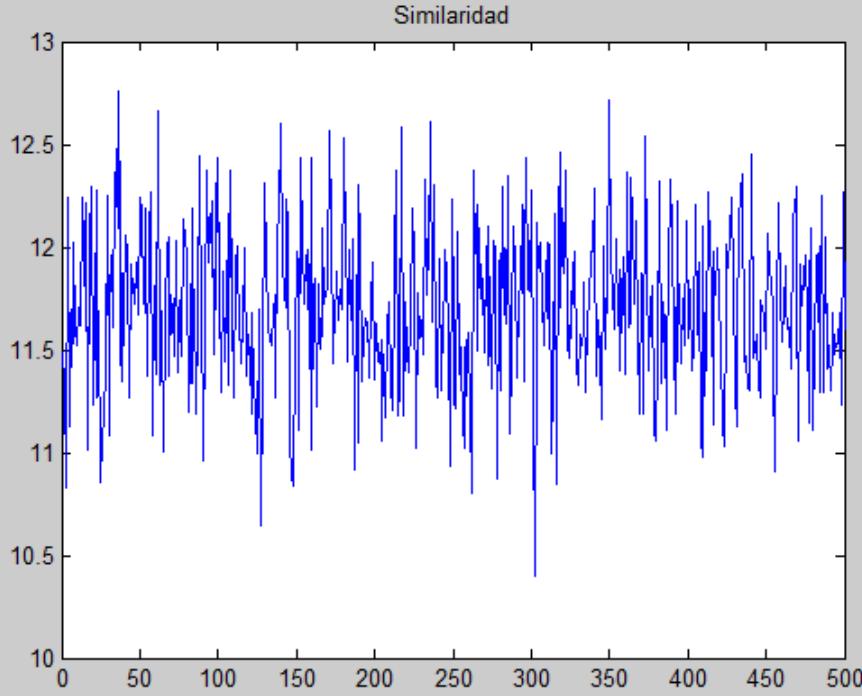
Parámetros:	$s = 20, t = 22$		
			
Imagen Original	Imagen Marcada		
	PSNR: 36.4950 dB		
	Calidad: Mala		
 <p>Similaridad</p> <p>Este gráfico muestra la similaridad entre los datos en función del índice. El eje vertical (Y) va de 10 a 24, y el eje horizontal (X) va de 0 a 500. La curva es generalmente estable en un rango entre 10 y 12, con una excepción significativa: un punto que se eleva abruptamente a un valor de alrededor de 23 a la mitad del eje X (aprox. índice 250).</p>			
<p>Similaridad</p> <table border="1"> <tr> <td>Número de Errores:</td> <td>0 de 1000</td> </tr> </table>		Número de Errores:	0 de 1000
Número de Errores:	0 de 1000		
<p>Tabla 184: Inserción y extracción por el algoritmo PCA para construir la imagen de referencia, $s = 20, t = 22$</p>			

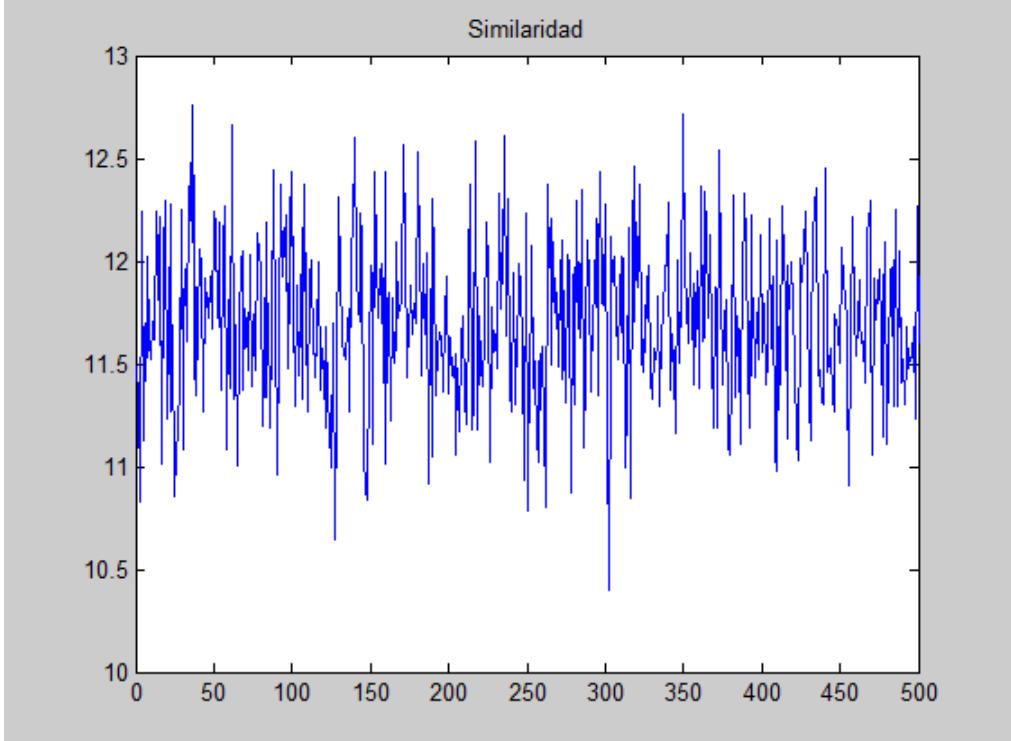
7.3.9.2.2. Pruebas de ataques

Se hará una prueba de cada ataque de los definidos, en ellas se comprobará la resistencia del algoritmo a los mismos, para estas pruebas se cogerá la imagen marcada que mejores resultados dio en las pruebas anteriores.

Nombre del ataque:	Compresión JPEG, 60%
	
Imagen Marcada	Imagen Atacada
	
Similaridad	
Número de Errores:	454 de 1000
Tabla 185: Ataque compresión JPEG 60% algoritmo PCA para construir la imagen de referencia	

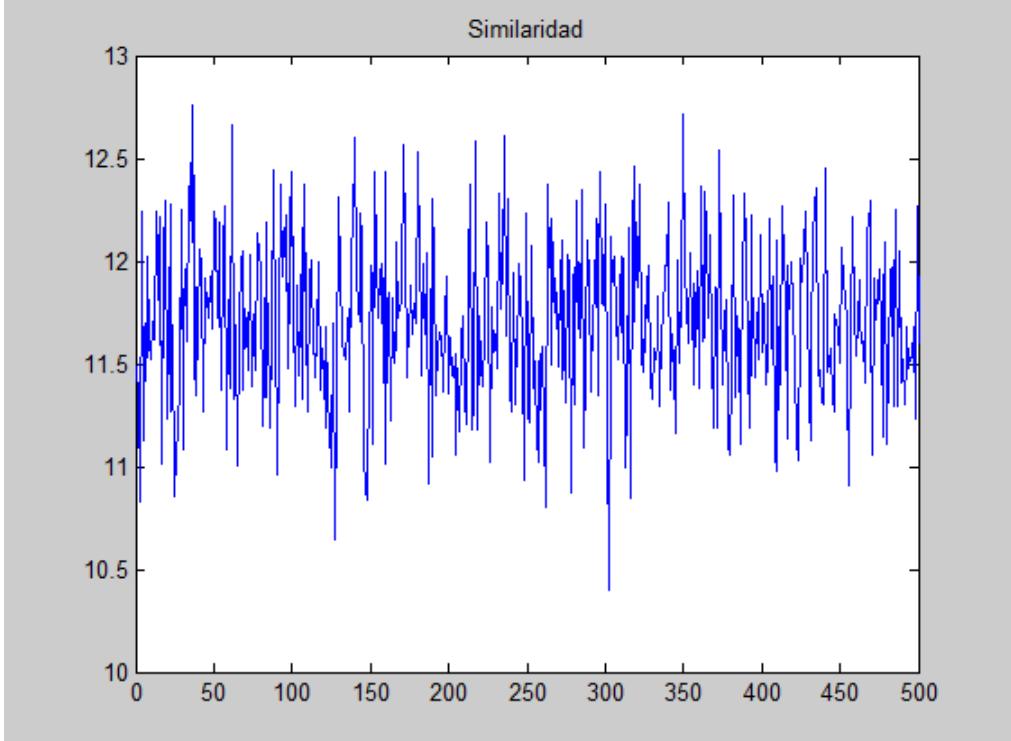
CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del ataque:	Inserción de ruido gaussiano
	
Imagen Marcada	Imagen Atacada
	
Número de Errores:	Similaridad 481 de 1000
Tabla 186: Ataque inserción ruido gaussiano algoritmo PCA para construir la imagen de referencia	

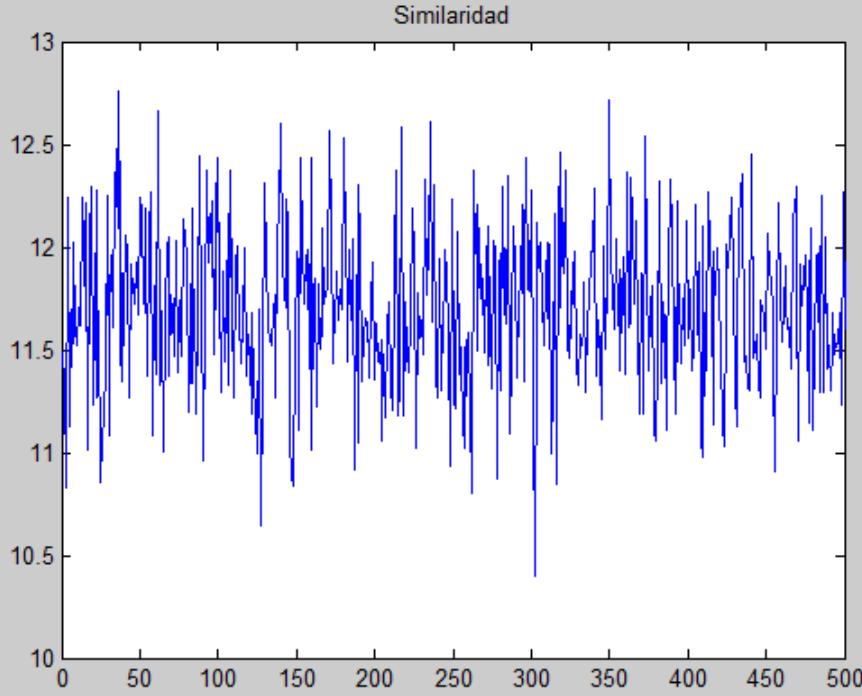
Nombre del ataque:	Aplicación de un filtro de paso bajo basado en la media
	
Imagen Marcada	Imagen Atacada
	
Número de Errores:	525 de 1000
Tabla 187: Ataque aplicación de un filtro de paso bajo basado en la media algoritmo PCA para construir la imagen de referencia	

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del ataque:	Recortado de la imagen marcada
	
Imagen Marcada	Imagen Atacada
Los métodos de extracción que existen no permiten este tipo de ataque ya que la imagen marcada y la atacada deben tener al menos, las mismas dimensiones.	
Similaridad	
Número de Errores:	No Necesario
Tabla 188: Ataque recortado de la imagen marcada algoritmo PCA para construir la imagen de referencia	

Nombre del ataque:	Escalado de la imagen marcada, doble tamaño
	
Imagen Marcada	Imagen Atacada
	
Similaridad	
Número de Errores:	514 de 1000
Tabla 189: Ataque escalado de la imagen marcada algoritmo PCA para construir la imagen de referencia	

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Nombre del ataque:	Rotación de la imagen marcada, 90°
	
Imagen Marcada	Imagen Atacada
	
Número de Errores:	Similaridad 511 de 1000
Tabla 190: Ataque rotación de la imagen marcada algoritmo PCA para construir la imagen de referencia	

7.3.9.2.3. Pruebas de tiempos de ejecución

Por último, se comprobará la eficiencia del algoritmo en tiempo de computación, para ello se cogerá una media de diez ejecuciones del mismo.

Algoritmo:	PCA para construir la imagen de referencia	
	Inserción de la marca de agua	Extracción de la marca de agua
Ejecución 1	1.6068	1.0140
Ejecución 2	1.7004	0.9048
Ejecución 3	1.4664	0.9048
Ejecución 4	1.5756	0.9672
Ejecución 5	1.4820	0.9984
Ejecución 6	1.6692	1.0140
Ejecución 7	1.5132	1.0296
Ejecución 8	1.4820	0.9048
Ejecución 9	1.6848	1.0140
Ejecución 10	1.6224	1.0452
Media	1.5803 seg.	0.9797 seg.

Tabla 191: Tiempos de ejecución del algoritmo PCA para construir la imagen de referencia

7.3.9.2.4. Conclusiones individuales

Después del estudio individual que se acaba de realizar se puede concluir que:

- En este algoritmo la extracción dependerá de los parámetros s y t, en este caso con unos valores de **s y t bajos** se consigue una imagen marcada con una **calidad muy buena** y una **extracción de la marca de agua perfecta**.
- En cuanto a los ataques:
 - Es resistente a: .
 - Es vulnerable a: compresión JPEG al 60%, inserción de ruido gaussiano, filtro de paso bajo de la media, escalado, rotación.
 - No es aplicable a: recortado.

7.4. Pruebas comunes

7.4.1. Comparativa de tiempos de computación

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Se va a presentar una tabla comparativa de los algoritmos basados a estudio, en ella se mostrará los tiempos de ejecución de las inserciones y extracciones. Para estos cálculos se han utilizado los algoritmos predefinidos por los autores en sus artículos, es decir, que se han utilizado los parámetros que los autores proponían como los mejores. Para diferenciar cuales son los parámetros predefinidos por los autores, en las tablas de las pruebas de parámetros, estos se han resaltado poniéndolos en negrita.

	Tiempo Inserción	Tiempo Extracción
Técnicas basadas en correlación	0.2433 seg.	0.3182 seg.
	0.2433 seg.	0.1653 seg.
Cox	0.5241 seg.	0.3775 seg.
DCT	1.5756 seg.	0.8205 seg.
DCT y Correlación	1.5101 seg.	1.3556 seg.
CDMA	2.7534 seg.	4.6816 seg.
CDMA en el dominio wavelet	12.2492 seg.	21.3752 seg.
DWT y la transformada de Haar	3.0654 seg.	0.3370 seg.
DWT basado en la paridad	0.6256 seg.	0.3869 seg.
SVD	0.8752 seg.	0.5023 seg.
SVD con la transformada de Arnold	17.5439 seg.	16.4690 seg.
SVD basado en el intercambio de valores	0.9688 seg.	0.9079 seg.
SVD basado en el orden de los coeficientes	0.2668 seg.	0.2200 seg.
SVD basado en la proximidad a un intervalo	0.3822 seg.	0.2668 seg.
LSB	0.6895 seg.	0.4914 seg.
Secuencias Caóticas	0.3962 seg.	1.6864 seg.
Secuencias Caóticas y DCT	1.3541 seg.	0.6895 seg.
	0.5569 seg.	0.2964 seg.
PCA	1.8143 seg.	1.6115 seg.
PCA para construir la imagen de referencia	1.5803 seg.	0.9797 seg.
Tabla 192: Comparativa de los tiempos de ejecución		

Con esto podemos concluir que por lo general la mayoría de los algoritmos rondan entre el medio segundo y los dos segundos en ejecución. Las inserciones

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

suelen ser más lentas que las extracciones, predominando en las extracciones los tiempos de ejecución por debajo del segundo.

Cabe comentar que el algoritmo más rápido en tiempo de ejecución, ya sea de inserción o extracción, es el algoritmo de técnicas basadas en correlación, en concreto el algoritmo modificado. El algoritmo que más tarda en computar los resultados en la inserción es el algoritmo SVD con la transformada de Arnold y en la extracción es el algoritmo CDMA en el dominio wavelet.

7.4.2. Comparativa de tipos de algoritmos

Una de las comparaciones más importantes que se tienen que hacer, es el tipo de algoritmo con el que se ha trabajado, para esto se hará una clasificación en cuanto a, si el algoritmo de extracción ha utilizado la imagen o la marca de agua original, si ha utilizado solo algunas variables o parámetros en la inserción que harán falta para la extracción (semiblind), o si no utilizan nada (blind, comúnmente llamados a ciegas).

	Blind	Semiblind	Requiere		
			Imagen original	Marca original	Parámetros
Técnicas basadas en correlación	X		NO	NO	NO
Cox		X	SI	NO	NO
DCT	X		NO	NO	NO
DCT y Correlación	X		NO	NO	NO
CDMA	X		NO	NO	NO

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

CDMA en el dominio wavelet	X		NO	NO	NO
DWT y la transformada de Haar		X	NO	NO	SI
DWT basado en la paridad	X		NO	NO	NO
SVD		X	NO	NO	SI
SVD con la transformada de Arnold		X	NO	NO	SI
SVD basado en el intercambio de valores	X		NO	NO	NO
SVD basado en el orden de los coeficientes	X		NO	NO	NO
SVD basado en la proximidad a un intervalo		X	NO	NO	SI
LSB			NO	SI	NO
Secuencias Caóticas	X		NO	NO	NO
Secuencias Caóticas y DCT	X		NO	NO	NO

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

PCA			SI	NO	NO
PCA para construir la imagen de referencia		X	NO	NO	SI
Tabla 193: Comparativa de los tipos de algoritmos					

Una vez realizada la tabla comparativa se puede observar que la mayoría de los algoritmos realizan la extracción sin utilizar nada para ella, es decir, son blind o a ciegas. También comentar que no hay ninguno que utilice la imagen y la marca de agua originales para la extracción, con lo que esto dice mucho de la calidad de los algoritmos.

Por último, decir que solamente tres necesitan de la imagen o la marca de agua originales para la extracción, dos necesitan de la imagen original y uno de la marca de agua original y cinco hacen uso de alguna variable o parámetro adicional.

7.4.3. Comparativa de resistencia a ataques

Se va a realizar una tabla comparativa que mostrará los resultados en cuanto a los ataques, se mostrará si un algoritmo ha resistido, no ha resistido o no se ha podido realizar dicho ataque.

	Resistencia a ataque					
	JPEG	Ruido	Filtro	Recortado	Escalado	Rotación
Técnicas basadas en correlación	NO	SI	NO	NO PERMITE	NO PERMITE	SI
Cox	SI	SI	SI	NO PERMITE	NO	NO
DCT	NO	NO	NO	NO PERMITE	NO	NO

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

DCT y Correlación	SI	NO	NO	NO PERMITE	NO	NO
CDMA	SI	SI	NO	NO	NO	NO
CDMA en el dominio wavelet	SI	SI	NO	NO PERMITE	NO	NO
DWT y la transformada de Haar	NO	NO	NO	NO PERMITE	NO	NO
DWT basado en la paridad	SI	NO	SI	NO PERMITE	NO	NO
SVD	SI	NO	NO	NO PERMITE	NO PERMITE	SI
SVD con la transformada de Arnold	NO	NO	NO	NO PERMITE	NO PERMITE	NO
SVD basado en el intercambio de valores	NO	NO	NO	NO PERMITE	NO	NO
SVD basado en el orden de los coeficientes	NO	NO	NO	NO PERMITE	NO PERMITE	NO
SVD basado en la proximidad a un intervalo	SI	NO	NO	NO PERMITE	NO PERMITE	SI
LSB	NO	NO	NO	SI	NO	SI
Secuencias Caóticas	SI	NO	NO	NO	NO	NO

CAPÍTULO 7: PRUEBAS Y CONCLUSIONES

Secuencias Caóticas y DCT	NO	SI	NO	NO	NO	NO
PCA	NO	NO	NO	NO PERMITE	NO PERMITE	NO
PCA para construir la imagen de referencia	NO	NO	NO	NO PERMITE	NO	NO
Total	SI	8	5	2	1	0
	NO	10	13	16	3	12
Tabla 194: Comparativa de la resistencia a los ataques						

Con lo que se puede concluir que el ataque al cual un algoritmo es más vulnerable es el **filtro de paso bajo basado en la media**, el ataque más resistente para los algoritmos es la **compresión JPEG** y el ataque que no se puede aplicar en la mayoría de los casos por culpa del algoritmo es el **recortado**.

Esto quiere decir que, si un atacante desea romper los derechos de autor de una imagen la cual haya sido marcada previamente, éste podrá atacar dicha imagen aplicando un filtro de paso bajo basado en la media y así destruir en la mayoría de los casos la marca de agua que el autor había insertado para protegerse de las posibles copias a su producto.

También hay que decir que el algoritmo que más ataques ha superado ha sido el algoritmo de Cox, esto quiere decir que este método tiene todas las posibilidades para poder ser mejorado en el caso de una futura aplicación real, ya que ofrece una resistencia que el resto de algoritmos no son capaces de llegar a proporcionar.

Por último, comentar que en esto del manejo de las marcas de agua todavía queda mucho por investigar, ya que en la mayoría de los casos los resultados podrían mejorarse.

CAPÍTULO 8: FUTURAS MEJORAS

8.1. Introducción

Como todo proyecto de investigación, y más en informática, todo va avanzando y se van descubriendo cosas nuevas e innovadoras. Por ello, es normal que todo pueda ser mejorable, ya sea en este caso, porque se descubran nuevos algoritmos o porque se desarrolle alguna aplicación donde se utilicen los métodos en los cuales se ha basado al estudio para un uso comercial. A continuación se destacarán las posibles mejoras que se esperan se puedan realizar.

8.2. Algoritmos

A día de hoy no se ha encontrado la solución definitiva al problema del marcado digital de imágenes. Para ello basta con hacer una búsqueda sobre el tema en congresos y revistas científicas donde se publica mucho sobre este asunto. Una mejora del proyecto sería incorporar nuevos métodos de marcado de imágenes digitales.

En este proyecto se ha abordado el marcado digital de una imagen en escala de grises. No se ha abordado el marcado de imágenes digitales a color, ni el marcado de video ni el marcado de ficheros digitales de audio. Por esto, una posible mejora de este proyecto podría consistir en incorporar técnicas de marcado de agua para este otro tipo de ficheros digitales.

8.3. Programa

CAPÍTULO 8: FUTURAS MEJORAS

Obviamente, este proyecto está basado en un estudio sobre los algoritmos existentes y no se ha centrado la atención en realizar un programa como tal para que un usuario pueda manejar, descubrir e interesarse por el mundo de las marcas de agua. Por esto, una posible mejora en este estudio sería poder realizar un programa como tal que pueda ofrecer todos los algoritmos estudiados, con ello se haría el tema del marcado digital de una imagen más fácil y atractivo para un usuario no experimentado.

8.4. Interfaz

Ya que una posible mejora sería realizar una aplicación para que un usuario sin conocimientos de lenguajes de programación pudiese manejar una aplicación basada en el manejo de las marcas de agua, lo más intuitivo es hacer una buena interfaz gráfica para que sea lo más simple. Esta interfaz gráfica llevará un trabajo específico para que sea lo más intuitiva posible y detallada en cuanto al manejo de las marcas de agua, ya sea para la ocultación de ellas o para la extracción, siempre teniendo en cuenta que se pueda observar todos los posibles casos que, al menos, se han estudiado en este proyecto.

8.5. Documentación

Siendo este el primer proyecto que he realizado en mi carrera, ni que decir tiene que habrá partes que deberán estar expuestas a revisión y mejora de los contenidos. Por ello, una posible mejora, y quizás la más importante, es la revisión de la documentación del proyecto. Tanto aclaración y adición de contenidos como una documentación más extensa en el caso de que finalmente se realizase una aplicación informática. La realización de una aplicación para el manejo de las marcas de agua y de la interfaz gráfica conllevaría volver a realizar documentación de la parte específica de Ingeniería del Software, sobre todo de los puntos de análisis de requisitos y modelado del sistema.

BIBLIOGRAFÍA

- [1] Shoemaker, C. Hidden Bits: A Survey of Techniques for Digital Watermarking, Union College, Schenectady, NY, 2002, última visita 23-Julio-2009, <http://www.vu.union.edu/~shoemakc/watermarking/>.
- [2] Godoy, M.; Mignola, C. Marcas de Agua (Watermarking) en Imágenes, Aplicadas en el Dominio Espacial Basadas en Correlación, 2004, última visita 23-Julio-2009, http://cpdsi-fich.wdfiles.com/local--files/tpsapplacion/2004_GodoyMignola-WatermarkingCorrelacion.pdf
- [3] Mandhani, Navneet K. Watermarking Using Decimal Sequences, 2004, última visita 23-Julio-2009, http://etd.lsu.edu/docs/available/etd-07072004-112736/unrestricted/Mandhani_thesis.pdf
- [4] Cox, I.J.; Kilian, J.; Leighton, F.T.; Shamoon, T. Secure Spread Spectrum Watermarking for Multimedia, IEEE, Volume: 6, Issue: 12-Dec-1997, Page(s): 1673-1687.
- [5] Stanescu, D.; Stratulat, M.; Ciubotaru, B.; et al. Digital Watermarking using Karhunen-Loeve transform, Yearly 17-May-2007, Page(s): 187-190.
- [6] Maity, Santi P.; Kundu, Malay K. A Blind CDMA Image Watermarking Scheme in Wavelet Domain, 2004, última visita 24-Julio-2009, http://www.isical.ac.in/~malay/Papers/Conf/ICIP%2704_1595.pdf
- [7] Jiang-Lung Liu; Der-Chyuan Lou; Ming-Chang Chang; et al. A Robust Watermarking Scheme Using Self-reference Image, ScienceDirect, Computer Standards & Interfaces, Volume: 28, Issue: 3-Jan-2006, Page(s): 356-367.

BIBLIOGRAFÍA

- [8] Ruizhen Liu; Tieniu Tan. A SVD-Based Watermarking Scheme for Protecting Rightful Ownership, IEEE, IEEE Transactions on Multimedia, Volume: 4, Issue: 1-Mar-2002, Page(s): 121-128.
- [9] Roman Rykaczewski. Comments on “An SVD-Based Watermarking Scheme for Protecting Rightful Ownership”, IEEE, IEEE Transactions on Multimedia, Volume: 9, Issue: 2, Page(s): 421-423.
- [10] Xiao-Ping Zhang; Kan Li. Comments on “An SVD-Based Watermarking Scheme for Protecting Rightful Ownership”, IEEE, IEEE Transactions on Multimedia, Volume: 7, Issue: 3-Jun-2005, Page(s): 593 – 594.
- [11] Deyun Peng; Jiazen Wang; Sumin Yang; et al. CDMA Based Multiple-User Digital Watermarking, IEEE, International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2006. IIH-MSP '06. Dec-2006, Page(s): 75 – 78.
- [12] Yan Xing; Jieqing Tan. A Color Watermarking Scheme Based on Block-SVD and Arnold Transformation, IEEE, Second Workshop on Digital Media and its Application in Museum & Heritages, 10-12 Dec-2007, Page(s): 3 – 8.
- [13] Zude Zhou; Bing Tang; Xinhua Liu. A Block-SVD Based Image Watermarking Method, IEEE, The Sixth World Congress on Intelligent Control and Automation, 2006. WCICA 2006, Volume: 2, Page(s): 10347 – 10351.
- [14] Yong-Gang Fu; Hui-Rong Wang. A Novel Discrete Wavelet Transform Based Digital Watermarking Scheme, 2nd International Conference on Anti-counterfeiting, Security and Identification, 2008. ASID 2008, 20-23 Aug-2008, Page(s): 55 – 58.
- [15] Belkacem, S.; Dibi, Z.; Bouridane, A. Color Image Watermarking based on Chaotic Map, IEEE, 14th IEEE International Conference on Electronics, Circuits and Systems, 2007. ICECS 2007, 11-14 Dec-2007, Page(s): 343 – 346.
- [16] Chin-Chen Chang; Yih-Shin Hu; Chia-Chen Lin. A Digital Watermarking Scheme Based on Singular Value Decomposition, SpringerLink, Combinatorics, Algorithms, Probabilistic and Experimental Methodologies, Volume: 4614/2007, Issue: 2007, Page(s): 82-93.
- [17] Kuo-Liang Chung; Wei-Ning Yang; Yong-Huai Huang; et al. On SVD-based watermarking algorithm, ScienceDirect, Applied Mathematics and Computation, Volume: 188, Issue: 1-May-2007, Page(s): 54-57.
- [18] Chin-Chen Chang; Piyu Tsai; Chia-Chen Lin. SVD-based digital image watermarking scheme, ScienceDirect, Pattern Recognition Letters, Volume: 26, Issue: 10-15 Jul-2005, Page(s): 1577-1586.

BIBLIOGRAFÍA

- [19] B.Chandra Mohan, S. Srinivas Kumar. A Robust Image Watermarking Scheme using Singular Value Decomposition, Journal Of Multimedia, Vol. 3, No. 1, May-2008.
- [20] Ganic, E; Eskicioglu, Ahmet M. Robust DWT-SVD Domain Image Watermarking: Embedding Data in All Frequencies, 2004, última visita 04-Agosto-2009, <http://www.sci.brooklyn.cuny.edu/~eskicioglu/papers/ACMworkshop2004.pdf>
- [21] Grace C.; W. Ting. Ambiguity Attacks on the Ganic-Eskicioglu Robust DWT-SVD Image Watermarking Scheme, SpringerLink, Information Security and Cryptology - ICISC 2005, Volume: 3935/2006, 2006, Page(s): 378-388.
- [22] Huo, Chong Ling; Raphael C.; W. Phan; et al. Attacks on SVD-Based Watermarking Schemes, SpringerLink, Intelligence and Security Informatics, Volume 5075/2008, 2008, Page(s): 83-91.
- [23] Chrysochos, E.; Fotopoulos, V.; Skodras, A. N. Robust Watermarking of Digital Images Based on Chaotic Mapping and DCT, 2008, última visita 04-Agosto-2009, <http://www.eurasip.org/Proceedings/Eusipco/Eusipco2008/papers/1569104383.pdf>
- [24] Thai D Hien; Yen-Wei Chen; Zensho Nakao. PCA Based Digital Watermarking, SpringerLink, Knowledge-Based Intelligent Information and Engineering Systems, Volume: 2773/2003, 2003, Page(s): 1427-1434.
- [25] Mirza, H.H.; Thai, H.D.; Nagata, Y.; et al. Digital VideoWatermarking Based on Principal Component Analysis, IEEE, Second International Conference on Innovative Computing, Information and Control, 2007. ICICIC '07. 5-7 Sep-2007, Page(s): 290 – 290.
- [26] Thai Duy Hien; Zensho Nakao; Kazuyoshi Miyara; et al. A New Chromatic Color Image Watermarking and Its PCA-Based Implementation, SpringerLink, Artificial Intelligence and Soft Computing – ICAISC 2006, Volume: 4029/2006, 2006, Page(s): 787-795.
- [27] Erkan Yavuz; Ziya Telatar. Digital Watermarking with PCA Based Reference Images, SpringerLink, Advanced Concepts for Intelligent Vision Systems, Volume: 4678/2007, 2007, Page(s): 1014-1023.
- [28] The Math Works, System Requirements - Release 2009b, última visita 09-Septiembre-2009, <http://www.mathworks.com/support/sysreq/sv-r2009a/>

BIBLIOGRAFÍA

APÉNDICE A: MANUAL DE USUARIO

A.1. Instalación

Para poder manejar todos los scripts que se proporcionan en el proyecto, antes de nada, es imprescindible tener el programa Matlab, en al menos la versión 7.6 y con los toolbox:

- Image Acquisition
- Image Processing
- Optimization
- Signal Processing
- Statistics
- Symbolic Math
- Wavelet

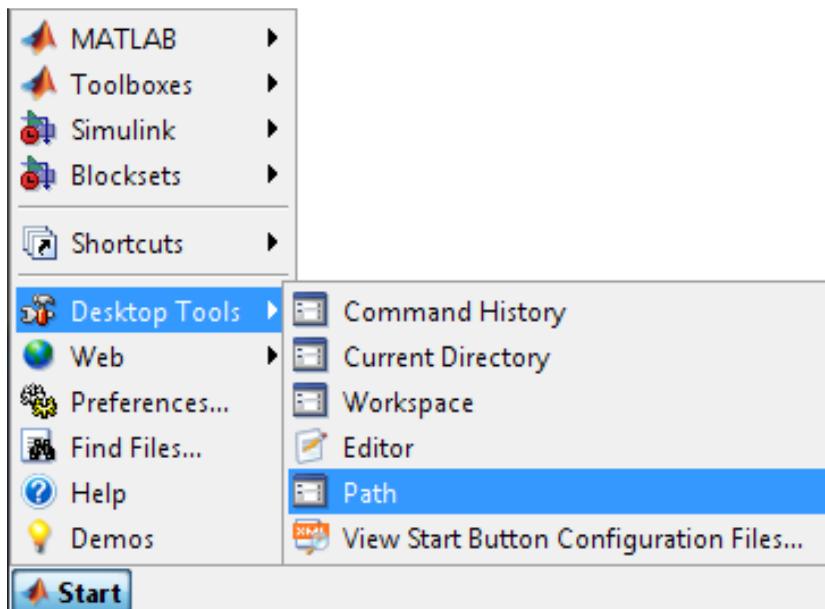
Con esto ya se pueden ejecutar todos los scripts sin ningún problema, sin embargo, no está de más hacer lo que ahora se comenta, ya que proporciona una mayor facilidad en el manejo de los scripts.

Para poder trabajar con los scripts casi como si fuese un toolbox desarrollado, lo que se debe de hacer es seguir los siguientes pasos (pasos para la versión 7.6):

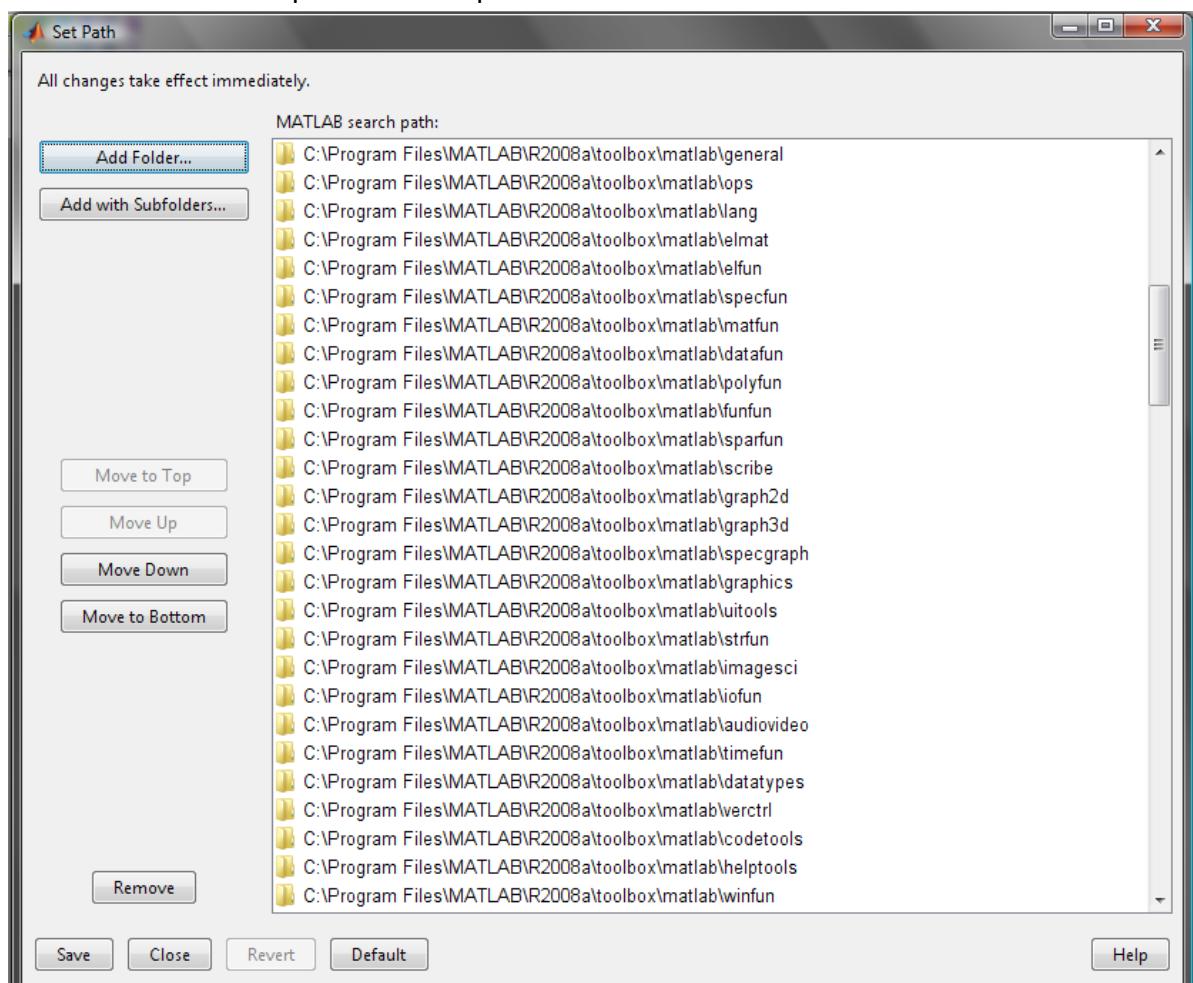
1. Abrir Matlab desde el archivo ejecutable correspondiente a la aplicación. 
2. En la esquina inferior izquierda habrá un icono donde pone Start, púlselo. 
3. Una vez pulsado habrá distintas opciones, sitúese en la opción Desktop Tools.

APÉNDICE A: MANUAL DE USUARIO

4. Saldrán más opciones de las cuales deberá elegir Path. Pulse ahí.
5. La figura muestra cómo ha de hacerse:

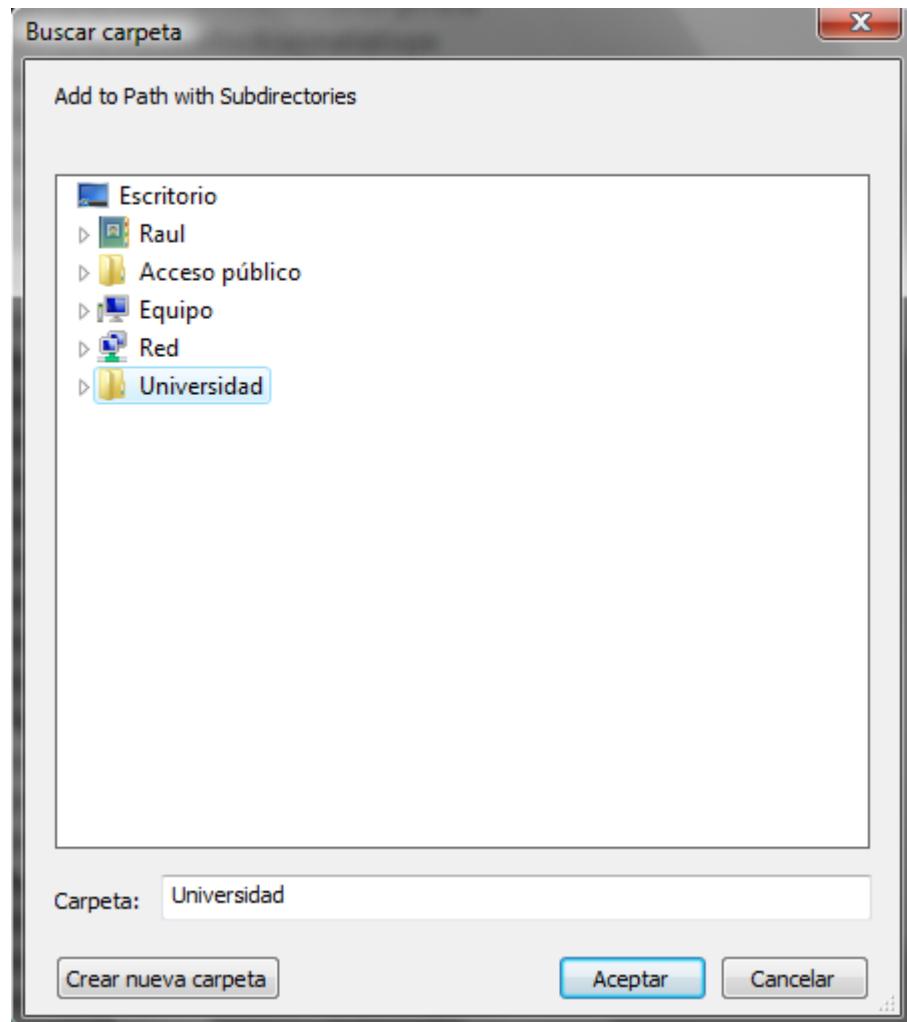


6. Cuando le dé aparecerá esta pantalla:



7. Pulse sobre Add with Subfolders. Aparecerá esta pantalla:

APÉNDICE A: MANUAL DE USUARIO



8. Ahora seleccione la carpeta donde se encuentra el proyecto y dele a Aceptar. Una vez realizado esto se añadirán todas las carpetas y subcarpetas que contiene el proyecto al Path, con lo que podrá invocar a los scripts directamente desde el intérprete de Matlab o desde los scripts que se proporcionan.
9. Por último, pulse sobre Save y seguidamente sobre Close.

Por otro lado se va a explicar las subcarpetas que contiene el proyecto y las cuales tienen su función específica para que todo quedase bien ordenado, existen 4 subcarpetas dentro del proyecto llamadas:

- Datos: almacenará los datos auxiliares de algunos algoritmos que necesiten estos para la extracción de la marca de agua.
- ImagenesAtaques: almacenará las seis imágenes atacadas de cada algoritmo de los sometidos a estudio.
- ImagenesOriginales: almacenará las imágenes y los logotipos que se utilizarán como marcas de agua originales.

APÉNDICE A: MANUAL DE USUARIO

- ImagenesPruebas: almacenará las imágenes resultantes a la inserción de la marca de agua, las marcas de agua extraídas, las gráficas de similaridad o correlación.

A.2. Desinstalación

No existe como tal, lo único que se podría tratar en este apartado es la desinstalación del programa Matlab, que no es del interés de este proyecto sino de la propia aplicación, o la eliminación de las carpetas del proyecto que hayan sido añadidas al Path, en cuyo caso habrá que seguir los siguientes pasos:

1. Sitúese en la ventana del paso 6 del apartado anterior (A.1).
2. Seleccione las carpetas que desea eliminar pulsando la tecla Ctrl + carpeta a eliminar.
3. Una vez seleccionadas estas pulse sobre la tecla Remove y procederá a la completa eliminación de las carpetas del Path.
4. Si desease volver a incluirlas solo deberá volver a seguir los pasos que se definen en el apartado anterior (A.1).

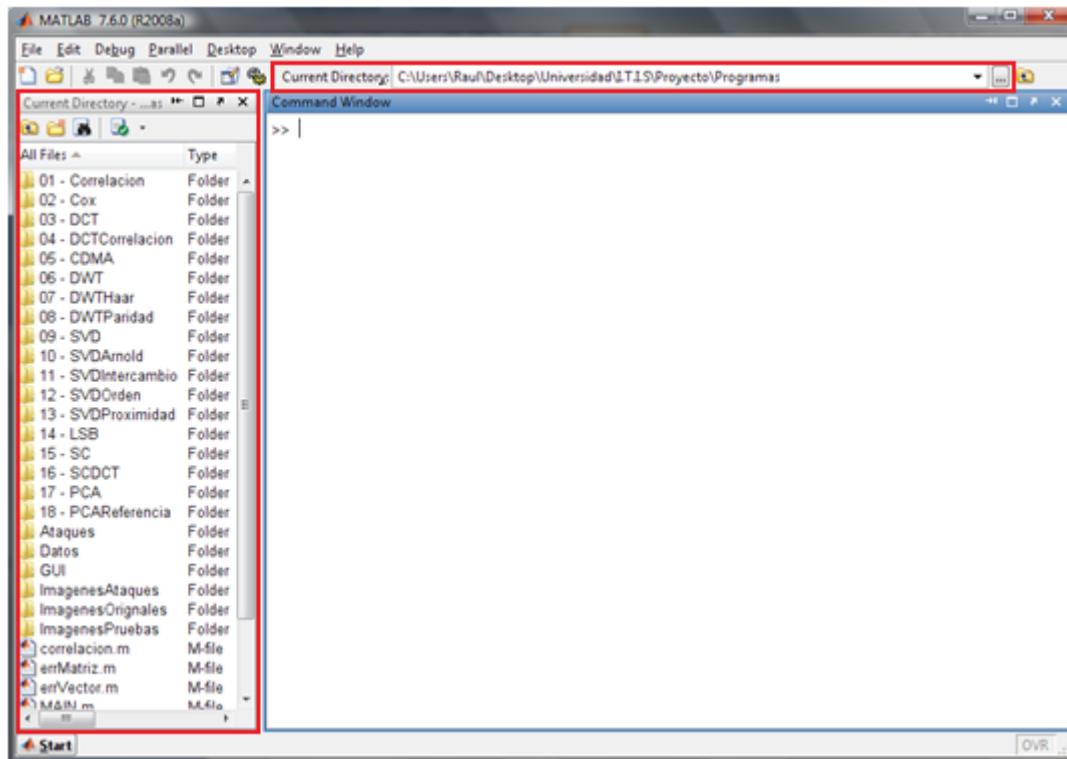
A.3. Ejemplo de un caso de uso

Para poder comprender de qué manera se han realizado las pruebas y ver un ejemplo de utilización de uno de los algoritmos que se han llevado a cabo para el estudio se muestra el siguiente ejemplo, en el que se utilizará el algoritmo CDMA, en este caso en la extracción de la marca de agua será un logotipo y no una gráfica de similaridad o correlación.

Para un seguimiento del ejemplo claro se dividirá en pasos, estos son:

1. Abrir Matlab.
2. Comprobar que se encuentra en el directorio adecuado, si no se está, situarse en él. Para comprobarlo se puede mirar en la parte superior del entorno gráfico de Matlab, donde pone Current Directory, o en la parte izquierda que tiene el mismo nombre. Se muestra en un recuadro rojo las zonas de comprobación.

APÉNDICE A: MANUAL DE USUARIO



3. Una vez comprobado, se hace la invocación en el interprete del algoritmo de inserción de la marca de agua, se haría de este modo:

```
>> cdma_insertar
```

Donde las salidas que se proporcionan por pantalla son el tiempo de ejecución y el PSNR:

```
elapsed_time =
```

```
2.9328
```

```
psnr =
```

```
31.0512
```

Y la salida externa es la imagen marcada que se encontrará en la subcarpeta ImagenesPruebas, estas imágenes tendrán la sintaxis de nombre de la manera: <nombreAlgoritmo>_watermarked.bmp, donde <nombreAlgoritmo> es el nombre del algoritmo correspondiente.

4. En este momento se hace la llamada al algoritmo de extracción, para ello se hará de este modo:

```
>> cdma_recuperar
```

Donde las salidas que se proporcionan por pantalla son el tiempo de ejecución y el número de errores que se han cometido entre la marca de agua original y la extraída:

APÉNDICE A: MANUAL DE USUARIO

elapsed_time =

4.7424

Bits mal recuperados de la marca: 0 de 108

Y la salida externa es la marca de agua extraída que se encontrará en la subcarpeta ImagenesPruebas, estas marcas tendrán la sintaxis de nombre de la manera: <nombreAlgoritmo>_watermark.bmp.

En los algoritmos en los que no se puede extraer la marca de agua de manera externa se hará la gráfica de similaridad o correlación que se sacará en lugar de la marca extraída.

Con esto queda mostrado un simple ejemplo de manipulación de los algoritmos, el manejo del resto de los scripts se hará de manera análoga. Para la simulación de los ataques solo hará falta ejecutar el script de simAtaques.m, el cual generará todas las imágenes de los seis ataques posibles en cada uno de los algoritmos que son objeto de estudio.

APÉNDICE B: MANUAL DE CÓDIGO

B.1. Algoritmos

B.1.1. Algoritmos de técnicas basadas en correlación

Inserción (cor_insertar.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: cor_insertar.m |%
%| Función: inserción de una marca de agua por el método |%
%| de correlación. |%
%-----%

% se limpia todo
clear all;

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se inicializan los parámetros necesarios
k = 15; % se da un valor inicial al factor de ganancia
blocksize = 16; % se pone el tamaño de bloque

% se guardan las variables que harán falta para la extracción
cd 'Datos'
save 'datosCor.mat' blocksize k;
cd ..

% se lee la imagen de cobertura
file_name = 'lena_512.bmp';
cover_object = double(imread(file_name));
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se determinan las dimensiones de la imagen de cobertura
[Mc,Nc] = size(cover_object);

% se determina el tamaño máximo de la marca basado en las
dimensiones de la
% imagen y el tamaño de bloque
max_message = Mc*Nc/(blocksize^2);

% se lee la marca de agua y se convierte a tipo double
file_name = 'copyright_50x20.bmp';
message = double(imread(file_name));
[Mm,Nm] = size(message);

% se redimensiona la marca poniendola como vector
message = round(reshape(message,Mm*Nm,1)./256);

% se comprueba que la marca de agua no sea demasiado grande
if(length(message) > max_message)
    error('Marca de agua demasiado grande.')
end

% se crea la nueva marca con el tamaño máximo establecido con todo
a unos
message_pad = ones(1,max_message);
message_pad(1:length(message)) = message;

% se lee una clave para el generador PN
file_name = 'key.bmp';
key = double(imread(file_name))./256;

% se reinicia el generador PN de MATLAB con "key"
rand('state',key(1));

% se genera la marca de agua igual que el tamaño de un bloque
pn_sequence = round(2*(rand(blocksize,blocksize)-0.5));

% se procesa la imagen en bloques, primero se construye la máscara
de marca
% global
x = 1;
y = 1;
for kk = 1:length(message_pad)
    % si el bit de la máscara contiene un cero, se añade una
    secuencia PN
    if(message_pad(kk) == 0)
        watermark_mask(y:y+blocksize-1,x:x+blocksize-1) = pn_sequence;
    % de otra manera, la máscara es rellenada de ceros
    else
        watermark_mask(y:y+blocksize-1,x:x+blocksize-1) =
zeros(blocksize,blocksize);
    end

    % se mueve al siguiente bloque
    if((x+blocksize) >= Nc)
        x = 1;
        y = y+blocksize;
    else
        x = x+blocksize;
    end
end
```

APÉNDICE B: MANUAL DE CÓDIGO

```
    end
end

% se añade la máscara de marca a la imagen de cobertura usando el
factor de
% ganancia k
watermarked_image dbl = cover_object+k*watermark_mask;
watermarked_image int = uint8(cover_object+k*watermark_mask);

% se guarda la imagen marcada en un fichero
cd 'ImagenesPruebas'
imwrite(watermarked_image_int,'cor_watermarked.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el valor del PSNR para la imagen marcada
psnr = psnr(cover_object,watermarked_image dbl,Mc,Nc)

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermarked_image_int,[])
% title('Imagen Marcada')
```

Extracción (cor_recuperar.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: cor_recuperar.m |%
%| Función: extracción de la marca de agua por el método |%
%| de correlación. |%
%-----%

% se limpia todo
clear all;

% se cargan los parámetros necesarios para la extracción
cd 'Datos'
load 'datosCor.mat'
cd ..

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se lee la imagen marcada y se le cambia el tipo a double
file_name = 'cor_watermarked.bmp';
watermarked_image = double(imread(file_name));

% se determinan las dimensiones de la imagen marcada
[Mw,Nw] = size(watermarked_image);
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se determina el máximo tamaño posible de la marca de agua
max_message = Mw*Nw/(blocksize^2);

% se lee la marca de agua original y se le cambia el tipo a double
file_name = 'copyright_50x20.bmp';
orig_watermark = double(imread(file_name));

% se determinan las dimensiones de la marca original
[Mo,No] = size(orig_watermark);

% se lee la clave para el generador PN
file_name = 'key.bmp';
key = double(imread(file_name))./256;

% se reinicia el generador PN de MATLAB con "key"
rand('state',key(1));

% se genera una marca igual al tamaño del bloque
pn_sequence = round(2*(rand(blocksize,blocksize)-0.5));

% se inicializa la marca de agua con todo a unos
message_vector = ones(Mo*No,1);

% se procesa la imagen en bloques, para cada bloque se determina su
% correlación con la secuencia pn base pn
x = 1;
y = 1;
for kk = 1:length(message_vector)
    % se calcula la correlación
    correlation(kk) = corr2(watermarked_image(y:y+blocksize-
1,x:x+blocksize-1),pn_sequence);

    % se mueve al siguiente bloque
    if((x+blocksize) >= Nw)
        x = 1;
        y = y+blocksize;
    else
        x = x+blocksize;
    end
end

% si la correlación excede la correlación media
for kk = 1:length(correlation)
    if(correlation(kk) > mean(correlation(1:Mo*No)))
        message_vector(kk) = 0;
    end
end

% se redimensiona la marca para ponerlo como matriz
watermark = reshape(message_vector(1:Mo*No),Mo,No)*255;

% se guarda la marca de agua en un fichero
cd 'ImagenesPruebas'
imwrite(watermark,'cor_watermark.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se calcula el número de errores que contiene la marca extraída  
errMatriz(orig_watermark,watermark,Mo,No);  
  
% se muestra la imagen marcada por pantalla  
% figure  
% imshow(watermark,[])  
% title('Marca Recuperada')
```

Extracción modificada (cor_recuperarmod.m)

```
%-----%  
%| Nombre proyecto: Marcas de agua en imágenes digitales |%  
%| Autor: Raúl Péruela Martínez |%  
%| Año: 2009 |%  
%-----%  
%| Nombre código: cor_recuperarmod.m |%  
%| Función: extracción de la marca de agua por el método |%  
%| de correlación modificado. |%  
%-----%  
  
% se limpia todo  
clear all;  
  
% se cargan los parámetros necesarios para la extracción  
cd 'Datos'  
load 'datosCor.mat'  
cd ..  
  
% se guarda el tiempo de comienzo de computación  
start_time = cputime;  
  
% se lee la imagen marcada y se le cambia el tipo a double  
file_name = 'cor_watermarked.bmp';  
watermarked_image = double(imread(file_name));  
  
% carga de las imágenes atacadas  
% file_name = 'cor_JPEGattack.jpg';  
% file_name = 'cor_NOISEattack.bmp';  
% file_name = 'cor_MEANattack.bmp';  
% file_name = 'cor_CROPPINGattack.bmp';  
% file_name = 'cor_SCALEDattack.bmp';  
% file_name = 'cor_ROTATEattack.bmp';  
% cd 'ImagenesAtaques'  
% watermarked_image = double(imread(file_name));  
% cd ..  
  
% se determinan las dimensiones de la imagen marcada  
[Mw,Nw] = size(watermarked_image);  
  
% se lee la imagen original y se le cambia el tipo a double  
file_name = 'lena_512.bmp';  
cover_object = double(imread(file_name));  
  
% se determina el máximo tamaño posible de la marca de agua
```

APÉNDICE B: MANUAL DE CÓDIGO

```
max_message = Mw*Nw/(blocksize^2);

% se lee la marca de agua original y se le cambia el tipo a double
file_name = 'copyright_50x20.bmp';
orig_watermark = double(imread(file_name));

% se determinan las dimensiones de la marca original
[Mo, No] = size(orig_watermark);

% se lee la clave para el generador PN
file_name = 'key.bmp';
key = double(imread(file_name))./256;

% se reinicia el generador PN de MATLAB con "key"
rand('state',key(1));

% se genera una marca igual al tamaño del bloque
pn_sequence = round(2*(rand(blocksize,blocksize)-0.5));

% se inicializa la marca de agua con todo a unos
message_vector = ones(No*Mo,1);

% se calcula la marca
W = (watermarked_image-cover_object)./k;

% se procesa la imagen en bloques, para cada bloque se determina su
% correlación con la secuencia pn base pn
x = 1;
y = 1;
ceros = zeros(blocksize,blocksize);
for kk = 1:length(message_vector)
    bloque = W(y:y+blocksize-1,x:x+blocksize-1);
    error1 = sum(sum((bloque-pn_sequence).^2));
    error2 = sum(sum((bloque-ceros).^2));
    if(error1 < error2)
        message_vector(kk) = 0;
    else
        message_vector(kk) = 1;
    end

    % se mueve al siguiente bloque
    if ((x+blocksize) >= Nw)
        x = 1;
        y = y+blocksize;
    else
        x = x+blocksize;
    end
end

% se redimensiona la marca para ponerla como matriz
watermark = reshape(message_vector(1:Mo*No),Mo,No)*255;

% se guarda la marca de agua en un fichero
cd 'ImagenesPruebas'
imwrite(watermark,'cor_watermarkMod.bmp');
cd ..

% se muestra el tiempo de computación
```

APÉNDICE B: MANUAL DE CÓDIGO

```
elapsed_time = cputime-start_time

% se calcula el número de errores que contiene la marca extraída
errMatriz(orig_watermark,watermark,Mo,No);

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermark,[])
% title('Marca Recuperada')
```

B.1.2. Algoritmo de Cox

Inserción (cox_insertar.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: cox_insertar.m |%
%| Función: inserción de una marca de agua por el método |%
%| de Cox. |%
%-----%

% se limpia todo
clear all;

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se inicializan los parámetros necesarios
alpha = 0.05;
tam = 2500;
key = 10;

% se guardan las variables que harán falta para la extracción
cd 'Datos'
save 'datosCox.mat' alpha tam key;
cd ..

% se lee la imagen de cobertura
file_name = 'lena_512.bmp';
cover_object = imread(file_name);

% se determinan las dimensiones de la imagen de cobertura
[Mc,Nc] = size(cover_object);

% se realiza la DCT de la imagen de cobertura
dctF1 = dct2(double(cover_object));

% se reinicia el generador PN de MATLAB con "key"
randn('state',key);

% se crea la marca con 'tam' números pseudoaleatorios
W = randn(tam,1);
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se calculan los coeficientes de mayor valor absoluto, se excluye el
% coeficiente DC, es decir, el coeficiente (1,1)
A = dctF1(:);
B = A;
B(1) = 0;

% Y1 contiene los valores ordenados, I1 contiene los índices en que se
% encuentran los valores del vector original
[Y1,I1] = sort(abs(B),'descend');

% se toman los coeficientes sin valor absoluto
Y1 = A(I1);

k = tam;
M = zeros(Mc*Nc,1);
l = 1;
for i = 1:Mc*Nc
    if(k >= 1)
        M(i) = Y1(i)*(1+alpha*W(l));
        l = l+1;
        k = k-1;
    else
        M(i) = Y1(i);
    end
end

% se cogen los valores en el orden de los índices ordenados
N = zeros(Mc*Nc,1);
for i = 1:Mc*Nc
    N(I1(i)) = M(i);
end

% se redimensiona la marca en forma de matriz
dctF2 = reshape(N,[Mc Nc]);

% se aplica la DCT inversa para recuperar la imagen
idctF1 = idct2(dctF2);

% se convierte la marca de agua a tipo uint8
watermarked_image_int = uint8(idctF1);

% se guarda la imagen marcada en un fichero
cd 'ImagenesPruebas'
imwrite(watermarked_image_int,'cox_watermarked.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el valor del PSNR para la imagen marcada
psnr = psnr(cover_object,watermarked_image_int,Mc,Nc)

% se muestra la imagen marcada
% figure;
% imshow(watermarked_image_int,[]);
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% title('Imagen Marcada');
```

Extracción (cox_recuperar.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: cox_recuperar.m |%
%| Función: extracción de la marca de agua por el método |%
%| Cox. |%
%-----%

% se limpia todo
clear all;

% se cargan los datos necesarios para la extracción
cd 'Datos'
load 'datosCox.mat'
cd ..

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se lee la imagen original y se le cambia el tipo a double
file_name = 'lena_512.bmp';
cover_object = double(imread(file_name));

% se lee la imagen marcada y se le cambia el tipo a double
file_name = 'cox_watermarked.bmp';
watermarked_image = double(imread(file_name));

% carga de las imágenes atacadas
% file_name = 'cox_JPEGattack.jpg';
% file_name = 'cox_NOISEattack.bmp';
% file_name = 'cox_MEANattack.bmp';
% file_name = 'cox_CROPPINGattack.bmp';
% file_name = 'cox_SCALEDattack.bmp';
% file_name = 'cox_ROTATEattack.bmp';
% cd 'ImagenesAtaques'
% watermarked_image = double(imread(file_name));
% cd ..

% se reinicia el generador PN de MATLAB con "key", produce números
% aleatorios de N(0,1)
randn('state',key);

% se crea la marca de tamaño 'tam' inicializado todo a cero
orig_watermark = zeros(tam,1);
for i = 1:tam
    orig_watermark(i) = randn(1);
end

% se calcula la DCT de la imagen original
```

APÉNDICE B: MANUAL DE CÓDIGO

```
dct = dct2(cover_object);

% se busca los coeficientes de mayor magnitud quitando el
% coeficiente DC,
% posicion (1,1)
A = dct(:);
B = A;
B(1) = 0;

% Y1 contiene el vector ordenado de forma descendiente, I1 contiene
el
% valor de los índices
[Y1,I1] = sort(abs(B), 'descend');

% se toma los coeficientes sin valor absoluto
Y1 = A(I1);

% se obtienen los índices correctos
V = Y1(1:tam);

% se calcula la DCT de la imagen marcada
dctmarcada = dct2(watermarked_image);
dctmarc = dctmarcada(:);
V1 = dctmarc(I1);
V1 = V1(1:tam);

% se extrae la marca
watermark = (V1-V)./(alpha*V);

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula la similaridad
graf = similaridad(orig_watermark, watermark, tam);
title('Similaridad');

% se guarda la gráfica en un fichero
cd 'ImagenesPruebas'
I = getframe(gcf);
imwrite(I.cdata, 'cox_similaridad.bmp');
cd ..
```

B.1.3. Algoritmos basados en la DCT

B.1.3.1. Primer Método: DCT

Inserción (*dct_insertar.m*)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: dct_insertar.m |%
```

APÉNDICE B: MANUAL DE CÓDIGO

```
%| Función: inserción de una marca de agua por el método |%
%| DCT. |%
%-----%  
  
% se limpia todo  
clear all;  
  
% se guarda el tiempo de comienzo de computación  
start_time = cputime;  
  
% se inicializan los parámetros necesarios  
k = 10; % factor de ganancia  
blocksize = 8; % tamaño del bloque  
  
% se guardan las variables que harán falta para la extracción  
cd 'Datos'  
save 'datosDCT.mat' blocksize;  
cd ..  
  
% se lee la imagen de cobertura y se convierte a tipo double  
file_name = 'lena_512.bmp';  
cover_object = double(imread(file_name));  
  
% se determinan las dimensiones de la imagen de cobertura  
[Mc,Nc] = size(cover_object);  
  
% se determina el tamaño maximo de la marca de agua basado en la  
imagen de  
% cobertura y el tamaño de bloque  
max_message = Mc*Nc/(blocksize^2);  
  
% se lee la marca de agua y se convierte a tipo double  
file_name = 'copyright_50x20.bmp';  
message = double(imread(file_name));  
  
% se determinan las dimensiones de la marca de agua  
[Mm,Nm] = size(message);  
  
% se redimensiona la marca de agua en forma de vector  
message = round(reshape(message,Mm*Nm,1)./256);  
  
% se comprueba que la marca de agua no sea demasiado grande  
if(length(message) > max_message)  
    error('Marca de agua demasiado grande.')  
end  
  
% se crea la nueva marca con el tamaño máximo establecido con todo  
a unos  
message_pad = ones(1,max_message);  
message_pad(1:length(message)) = message;  
  
% se genera una copia de la imagen marcada  
watermarked_image = cover_object;  
  
% se procesa la imagen en bloques codificados tal que (5,2) > (4,3)  
cuando  
% message(kk) = 0 y que (5,2) < (4,3) cuando message(kk) = 1  
x = 1;
```

APÉNDICE B: MANUAL DE CÓDIGO

```
y = 1;
for i = 1:length(message_pad)
    % bloque transformado usando DCT
    dct_block = dct2(cover_object(y:y+blocksize-1,x:x+blocksize-1));

    % si el bit de la marca es cero, (5,2) > (4,3)
    if(message_pad(i) == 0)
        % si (5,2) < (4,3) entonces se necesita intercambiar las
        posiciones
        if (dct_block(5,2) < dct_block(4,3))
            temp = dct_block(4,3);
            dct_block(4,3) = dct_block(5,2);
            dct_block(5,2) = temp;
        end
        % si el bit de la marca es uno, (5,2) < (4,3)
        elseif(message_pad(i) == 1)
            % si (5,2) > (4,3) entonces se necesita intercambiar las
            posiciones
            if(dct_block(5,2) >= dct_block(4,3))
                temp = dct_block(4,3);
                dct_block(4,3) = dct_block(5,2);
                dct_block(5,2) = temp;
            end
        end
    end

    % ahora se ajustan los dos valores tal que su diferencia sea >= k
    if(dct_block(5,2) > dct_block(4,3))
        if(dct_block(5,2) - dct_block(4,3) < k)
            dct_block(5,2) = dct_block(5,2)+(k/2);
            dct_block(4,3) = dct_block(4,3)-(k/2);
        end
    else
        if(dct_block(4,3) - dct_block(5,2) < k)
            dct_block(4,3) = dct_block(4,3)+(k/2);
            dct_block(5,2) = dct_block(5,2)-(k/2);
        end
    end
end

% se transforma el bloque al dominio espacial
watermarked_image(y:y+blocksize-1,x:x+blocksize-1) =
idct2(dct_block);

% se mueve al siguiente bloque
if((x+blocksize) >= Nc)
    x = 1;
    y = y+blocksize;
else
    x = x+blocksize;
end
end

% se convierte la imagen marcada a tipo uint8 y se guarda en un
fichero
watermarked_image_int = uint8(watermarked_image);
cd 'ImagenesPruebas'
imwrite(watermarked_image_int,'dct_watermarked.bmp');
cd ..
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se muestra el tiempo de computación  
elapsed_time = cputime-start_time  
  
% se calcula el valor del PSNR para la imagen marcada  
psnr = psnr(cover_object,watermarked_image,Mc,Nc)  
  
% se muestra la imagen marcada por pantalla  
% figure  
% imshow(watermarked_image_int,[])  
% title('Imagen Marcada')
```

Extracción (dct_recuperar.m)

```
%-----%  
%| Nombre proyecto: Marcas de agua en imágenes digitales |%  
%| Autor: Raúl Péruela Martínez |%  
%| Año: 2009 |%  
%-----%  
%| Nombre código: dct_recuperar.m |%  
%| Función: extracción de la marca de agua por el método |%  
%| DCT. |%  
%-----%  
  
% se limpia todo  
clear all;  
  
% se cargan los datos necesarios para la extracción  
cd 'Datos'  
load 'datosDCT.mat'  
cd ..  
  
% se guarda el tiempo de comienzo de computación  
start_time = cputime;  
  
% se lee la imagen marcada y se convierte a tipo double  
% file_name = 'dct_watermarked.bmp';  
% watermarkedImage = double(imread(file_name));  
  
% carga de las imágenes atacadas  
% file_name = 'dct_JPEGatack.jpg';  
% file_name = 'dct_NOISEatack.bmp';  
% file_name = 'dct_MEANatack.bmp';  
% file_name = 'dct_CROPPINGatack.bmp';  
% file_name = 'dct_SCALEDatack.bmp';  
file_name = 'dct_ROTATEatack.bmp';  
cd 'ImagenesAtaques'  
watermarkedImage = double(imread(file_name));  
cd ..  
  
% se determinan las dimensiones de la imagen marcada  
[Mw,Nw] = size(watermarkedImage);  
  
% se determina el tamaño maximo de la marca de agua basado en la  
% imagen de  
% cobertura y el tamaño de bloque
```

APÉNDICE B: MANUAL DE CÓDIGO

```
max_message = Mw*Nw/(blocksize^2);

% se lee la marca da agua original y se convierte a tipo double
file_name = 'copyright_50x20.bmp';
orig_watermark = double(imread(file_name));

% se determinan las dimensiones de la marca de agua original
[Mo,No] = size(orig_watermark);

% se procesa la imagen en bloques
x = 1;
y = 1;
for i = 1:max_message
    % bloque transformado usando DCT
    dct_block = dct2(watermarked_image(y:y+blocksize-1,x:x+blocksize-1));

    % si dct_block(5,2) > dct_block(4,3) entonces message(i) = 0
    % de otro modo message(i) = 1
    if(dct_block(5,2) > dct_block(4,3))
        message_vector(i) = 0;
    else
        message_vector(i) = 1;
    end

    % se mueve al bloque siguiente
    if((x+blocksize) >= Nw)
        x = 1;
        y = y+blocksize;
    else
        x = x+blocksize;
    end
end

% se redimensiona la marca de agua en forma de matriz
watermark = reshape(message_vector(1:Mo*No),Mo,No)*255;

% se guarda la marca de agua en un fichero
cd 'ImagenesPruebas'
imwrite(watermark,'dct_watermark.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el número de errores que contiene la marca extraída
errMatriz(orig_watermark,watermark,Mo,No);

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermark,[])
% title('Marca Recuperada')
```

B.1.3.2. Segundo Método: DCT y correlación

Inserción (dctcor_insertar.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: dctcor_insertar.m |%
%| Función: inserción de una marca de agua por el método |%
%| DCT haciendo uso de la correlacion. |%
%-----%

% se limpia todo
clear all;

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se inicializan los parámetros necesarios
k = 10; % factor de ganancia
blocksize = 8; % tamaño de bloque
pn_sequence_search = 'T'; % búsqueda para encontrar baja
correlación entre pn % sequences {T,F}

% se guardan las variables que harán falta para la extracción
cd 'Datos'
save 'datosDCTCor.mat' blocksize pn_sequence_search;
cd ..

% se definen las frecuencias de banda media del dct 8x8
midband = [ 0,0,0,1,1,1,1,0;
             0,0,1,1,1,1,0,0;
             0,1,1,1,1,0,0,0;
             1,1,1,1,0,0,0,0;
             1,1,1,0,0,0,0,0;
             1,1,0,0,0,0,0,0;
             1,0,0,0,0,0,0,0;
             0,0,0,0,0,0,0,0 ];

% se lee la imagen de cobertura y se convierte a tipo double
file_name = 'lena_512.bmp';
cover_object = double(imread(file_name));

% se determinan las dimensiones de la imagen de cobertura
[Mc,Nc] = size(cover_object);

% se determina el tamaño maximo de la marca de agua basado en la
imagen de
% cobertura y el tamaño de bloque
max_message = Mc*Nc/(blocksize^2);

% se lee la marca de agua y se convierte a tipo double
file_name = 'copyright_50x20.bmp';
message = double(imread(file_name));
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se determinan las dimensiones de la marca de agua
[Mm,Nm] = size(message);

% reshape the message to a vector
message = round(reshape(message,Mm*Nm,1)./256);

% se comprueba que la marca de agua no sea demasiado grande
if(length(message) > max_message)
    error('Marca de agua demasiado grande.')
end

% se crea la nueva marca con el tamaño máximo establecido con todo
% a unos
message_vector = ones(1,max_message);
message_vector(1:length(message)) = message;

% se genera una copia de la imagen marcada
watermarked_image = cover_object;

% se lee una clave para el generador PN
file_name = 'key.bmp';
key = double(imread(file_name))./256;

% se reinicia el generador PN de MATLAB con "key"
rand('state',key(1));

% se generan PN secuencias de "1" y "0"
pn_sequence_one = round(2*(rand(1,sum(sum(midband)))-0.5));
pn_sequence_zero = round(2*(rand(1,sum(sum(midband)))-0.5));

% find two highly un-correlated PN sequences
if(pn_sequence_search == 'T')
    while(corr2(pn_sequence_one,pn_sequence_zero) > -0.55)
        pn_sequence_one = round(2*(rand(1,sum(sum(midband)))-0.5));
        pn_sequence_zero = round(2*(rand(1,sum(sum(midband)))-0.5));
    end
end

% se procesa la imagen en bloques
x = 1;
y = 1;
for kk = 1:length(message_vector)
    % bloque transformado usando DCT
    dct_block = dct2(cover_object(y:y+blocksize-1,x:x+blocksize-1));

    % si el bit de la marca de agua contiene un cero entonces se
    % oculta en
    % pn_sequence_zero en los componentes de banda media del
    % dct_block
    ll = 1;
    if(message_vector(kk) == 0)
        for ii = 1:blocksize
            for jj = 1:blocksize
                if(midband(jj,ii) == 1)
                    dct_block(jj,ii) =
dct_block(jj,ii)+k*pn_sequence_zero(ll);
                ll = ll+1;
            end
        end
    end
end
```

APÉNDICE B: MANUAL DE CÓDIGO

```
    end
end
end

% en otro caso, se oculta pn_sequence_one en los componentes de
banda
% media de dct_block
else
for ii = 1:blocksize
    for jj = 1:blocksize
        if(midband(jj,ii) == 1)
            dct_block(jj,ii) =
dct_block(jj,ii)+k*pn_sequence_one(ll);
            ll = ll+1;
        end
    end
end
end

% se transforma el bloque al dominio espacial
watermarked_image(y:y+blocksize-1,x:x+blocksize-
1)=idct2(dct_block);

% se mueve al siguiente bloque
if((x+blocksize) >= Nc)
    x = 1;
    y = y+blocksize;
else
    x = x+blocksize;
end
end

% se convierte la imagen marcada a tipo uint8
watermarked_image_int = uint8(watermarked_image);

% se guarda la imagen marcada en un fichero
cd 'ImagenesPruebas'
imwrite(watermarked_image_int,'dctcor_watermarked.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el valor del PSNR para la imagen marcada
psnr = psnr(cover_object,watermarked_image,Mc,Nc)

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermarked_image_int,[])
% title('Imagen Marcada')
```

Extracción (dctcor_recuperar.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
```

APÉNDICE B: MANUAL DE CÓDIGO

```
%| Autor: Raúl Pérrula Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: dctcor_recuperar.m |%
%| Función: extracción de la marca de agua por el método |%
%| DCT basado en la correlacion. |%
%-----%

% se limpia todo
clear all;

% se cargan los datos necesarios para la extracción
cd 'Datos'
load 'datosDCTCor.mat'
cd ..

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se definen las frecuencias de banda media del dct 8x8
midband = [ 0,0,0,1,1,1,1,0;
             0,0,1,1,1,1,0,0;
             0,1,1,1,1,0,0,0;
             1,1,1,1,0,0,0,0;
             1,1,1,0,0,0,0,0;
             1,1,0,0,0,0,0,0;
             1,0,0,0,0,0,0,0;
             0,0,0,0,0,0,0,0 ];

% se lee la imagen marcada y se convierte a tipo double
% file_name = 'dctcor_watermarked.bmp';
% watermarkedImage = double(imread(file_name));

% carga de las imágenes atacadas
% file_name = 'dctcor_JPEGatack.jpg';
% file_name = 'dctcor_NOISEatack.bmp';
% file_name = 'dctcor_MEANatack.bmp';
% file_name = 'dctcor_CROPPINGatack.bmp';
% file_name = 'dctcor_SCALEDatack.bmp';
file_name = 'dctcor_ROTATEatack.bmp';
cd 'ImagenesAtaques'
watermarkedImage = double(imread(file_name));
cd ..

% se determinan las dimensiones de la imagen marcada
[Mw,Nw] = size(watermarkedImage);

% se determina el tamaño maximo de la marca de agua basado en la
% imagen de
% cobertura y el tamaño de bloque
max_message = Mw*Nw/(blocksize^2);

% se lee la marca de agua original y se convierte a tipo double
file_name = 'copyright_50x20.bmp';
orig_watermark = double(imread(file_name));

% se determinan las dimensiones de la marca de agua original
[Mo,No] = size(orig_watermark);
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se lee una clave para el generador PN
file_name = 'key.bmp';
key = double(imread(file_name))./256;

% se reinicia el generador PN de MATLAB con "key"
rand('state',key(1));

% se generan PN secuencias de "1" y "0"
pn_sequence_one = round(2*(rand(1,sum(sum(midband)))-0.5));
pn_sequence_zero = round(2*(rand(1,sum(sum(midband)))-0.5));

% se encuentras dos PN secuencias altamente no correlacionadas
if(pn_sequence_search == 'T')
    while(corr2(pn_sequence_one,pn_sequence_zero) > -0.55)
        pn_sequence_one = round(2*(rand(1,sum(sum(midband)))-0.5));
        pn_sequence_zero = round(2*(rand(1,sum(sum(midband)))-0.5));
    end
end

% se procesa la imagen en bloques
x = 1;
y = 1;
for kk = 1:max_message
    % bloque transformado usando DCT
    dct_block = dct2(watermarked_image(y:y+blocksize-1,x:x+blocksize-1));

    % se extrae los coeficientes de la manda media
    ll = 1;
    for ii = 1:blocksize
        for jj = 1:blocksize
            if(midband(jj,ii) == 1)
                sequence(ll) = dct_block(jj,ii);
                ll = ll+1;
            end
        end
    end

    % se calcula la correlación de la secuencia de banda media con
    pn_sequences
    % y se elige el valor con la mayor correlación de la marca
    cor_one(kk) = corr2(pn_sequence_one,sequence);
    cor_zero(kk) = corr2(pn_sequence_zero,sequence);
    if(cor_zero(kk) > cor_one(kk))
        message_vector(kk) = 0;
    else
        message_vector(kk) = 1;
    end

    % se mueve al siguiente bloque
    if((x+blocksize) >= Nw)
        x = 1;
        y = y+blocksize;
    else
        x = x+blocksize;
    end
end
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se redimensiona la marca de agua en forma de matriz  
watermark = reshape(message_vector(1:Mo*No), Mo, No)*255;  
  
% se guarda la marca de agua en un fichero  
cd 'ImagenesPruebas'  
imwrite(watermark, 'dctcor_watermark.bmp');  
cd ..  
  
% se muestra el tiempo de computación  
elapsed_time = cputime-start_time  
  
% se calcula el número de errores que contiene la marca extraída  
errMatriz(orig_watermark, watermark, Mo, No);  
  
% se muestra la imagen marcada por pantalla  
% figure  
% imshow(watermark, [])  
% title('Marca Recuperada')
```

B.1.4. Algoritmo basado en CDMA

Inserción (cdma_insertar.m)

```
%-----%  
%| Nombre proyecto: Marcas de agua en imágenes digitales |%  
%| Autor: Raúl Péruela Martínez |%  
%| Año: 2009 |%  
%-----%  
%| Nombre código: cdma_insertar.m |%  
%| Función: inserción de una marca de agua por el método |%  
%| CDMA. |%  
%-----%  
  
% se limpia todo  
clear all;  
  
% se guarda el tiempo de comienzo de computación  
start_time = cputime;  
  
% se da un valor inicial al factor de ganancia  
k = 2;  
  
% se lee la imagen de cobertura  
file_name = 'lena_512.bmp';  
cover_object = double(imread(file_name));  
  
% se determinan las dimensiones de la imagen de cobertura  
[Mc, Nc] = size(cover_object);  
  
% se lee la marca de agua y se redimensiona poniendolo como un  
vector  
file_name = 'copyright_12x9.bmp';  
message = double(imread(file_name));
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se determinan las dimensiones de la marca de agua  
[Mm,Nm] = size(message);  
  
% se redimensiona la marca en forma de vector binario  
message_vector = round(reshape(message,Mm*Nm,1)./256);  
  
% se lee una clave para el generador PN  
file_name = 'key.bmp';  
key = double(imread(file_name))./256;  
  
% se reinicia el generador PN de MATLAB con "key"  
rand('state',key(1));  
  
% se crea una copia de la imagen  
watermarked_image = cover_object;  
  
% cuando la marca contiene un '0', se añade una secuencia pn con un  
factor  
% de ganancia k para la imagen de cobertura  
for i = 1:length(message_vector)  
    pn_sequence = round(2*(rand(Mc,Nc)-0.5));  
    if(message(i) == 0)  
        watermarked_image = watermarked_image+k*pn_sequence;  
    end  
end  
  
% se convierte la marca de agua a tipo uint8  
watermarked_image_int = uint8(watermarked_image);  
  
% se guarda la imagen marcada en un fichero  
cd 'ImagenesPruebas'  
imwrite(watermarked_image_int,'cdma_watermarked.bmp');  
cd ..  
  
% se muestra el tiempo de computación  
elapsed_time = cputime-start_time  
  
% se calcula el valor del PSNR para la imagen marcada  
psnr = psnr(cover_object,watermarked_image,Mc,Nc)  
  
% se muestra la imagen marcada por pantalla  
% figure  
% imshow(watermarked_image_int,[])  
% title('Imagen Marcada')
```

Extracción (cdma_recuperar.m)

```
%-----%  
%| Nombre proyecto: Marcas de agua en imágenes digitales |%  
%| Autor: Raúl Péruela Martínez |%  
%| Año: 2009 |%  
%-----%  
%| Nombre código: cdma_recuperar.m |%  
%| Función: extracción de la marca de agua por el método |%  
%| CDMA. |%
```

APÉNDICE B: MANUAL DE CÓDIGO

```
%-----%
% se limpia todo
clear all;

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se lee la imagen marcada y se le cambia el tipo a double
file_name = 'cdma_watermarked.bmp';
watermarked_image = double(imread(file_name));

% carga de las imágenes atacadas
% file_name = 'cdma_JPEGattack.jpg';
% file_name = 'cdma_NOISEattack.bmp';
% file_name = 'cdma_MEANattack.bmp';
% file_name = 'cdma_CROPPINGattack.bmp';
% file_name = 'cdma_SCALEDATAattack.bmp';
% file_name = 'cdma_ROTATEattack.bmp';
% cd 'ImagenesAtaques'
% watermarked_image = double(imread(file_name));
% cd ..

% se determinan las dimensiones de la imagen marcada
[Mw,Nw] = size(watermarked_image);

% se lee la marca de agua original y se le cambia el tipo a double
file_name = 'copyright_12x9.bmp';
orig_watermark = double(imread(file_name));

% se determinan las dimensiones de la marca original
[Mo,No] = size(orig_watermark);

% se lee la clave para el generador PN
file_name = 'key.bmp';
key = double(imread(file_name))./256;

% se reinicia el generador PN de MATLAB con "key"
rand('state',key(1));

% se inicializa la marca de agua con todo a unos
message_vector = ones(1,Mo*No);
for i = 1:length(message_vector)
    % se genera secuencias PN de valores {-1,0,1}
    pn_sequence = round(2*(rand(Mw,Nw)-0.5));

    % se calcula la correlación
    correlation(i) = corr2(watermarked_image,pn_sequence);
end

% se usa como el valor medio de correlación como umbral
threshold = mean(correlation);

% si la correlación excede el umbral, se pone el bit de la marca a
cero
for i = 1:length(message_vector)
    if(correlation(i) > threshold)
        message_vector(i) = 0;
```

APÉNDICE B: MANUAL DE CÓDIGO

```
end  
end  
  
% se redimensiona la marca para ponerlo como matriz  
watermark = reshape(message_vector,Mo,No)*255;  
  
% se guarda la marca de agua en un fichero  
cd 'ImagenesPruebas'  
imwrite(watermark,'cdma_watermark.bmp');  
cd ..  
  
% se muestra el tiempo de computación  
elapsed_time = cputime-start_time  
  
% se calcula el número de errores que contiene la marca extraída  
errMatriz(orig_watermark,watermark,Mo,No);  
  
% se muestra la imagen marcada por pantalla  
% figure  
% imshow(watermark,[])  
% title('Marca Recuperada')
```

B.1.5. Algoritmos basados en la DWT

B.1.5.1. Primer Método: CDMA en el dominio wavelet

Inserción (dwt_insertar.m)

```
%-----%  
%| Nombre proyecto: Marcas de agua en imágenes digitales |%  
%| Autor: Raúl Péruela Martínez |%  
%| Año: 2009 |%  
%-----%  
%| Nombre código: dwt_insertar.m |%  
%| Función: inserción de una marca de agua por el método |%  
%| DWT. |%  
%-----%  
  
% se limpia todo  
clear all;  
  
% se guarda el tiempo de comienzo de computación  
start_time = cputime;  
  
% se inicializan los parámetros necesarios  
k = 2;  
  
% se lee la imagen de cobertura y se convierte a tipo double  
file_name = 'lena_512.bmp';  
cover_object = double(imread(file_name));  
  
% se determinan las dimensiones de la imagen de cobertura  
[Mc,Nc] = size(cover_object);  
  
% se lee la marca de agua y se redimensiona en un vector
```

APÉNDICE B: MANUAL DE CÓDIGO

```
file_name = 'copyright_50x20.bmp';
message = double(imread(file_name));

% se determinan las dimensiones de la marca de agua
[Mm,Nm] = size(message);

% se redimensiona la marca en forma de vector binario
message_vector = round(reshape(message,Mm*Nm,1)./256);

% se lee una clave para el generador PN
file_name = 'key.bmp';
key = double(imread(file_name))./256;

% se reinicia el generador PN de MATLAB con "key"
rand('state',key(1));

% se realiza la transformada de Haar
[cA1,cH1,cV1,cD1] = dwt2(cover_object,'haar');

% se añaden secuencias pn a los componentes H1 y V1 cuando message
= 0
for i = 1:length(message_vector)
    pn_sequence_h = round(2*(rand(Mc/2,Nc/2)-0.5));
    pn_sequence_v = round(2*(rand(Mc/2,Nc/2)-0.5));
    if(message(i) == 0)
        cH1 = cH1+k*pn_sequence_h;
        cV1 = cV1+k*pn_sequence_v;
    end
end

% se realiza la transformada inversa
watermarked_image = idwt2(cA1,cH1,cV1,cD1,'haar',[Mc,Nc]);

% se convierte la imagen marcada a tipo uint8
watermarked_image_int = uint8(watermarked_image);

% se guarda la imagen marcada en un fichero
cd 'ImagenesPruebas'
imwrite(watermarked_image_int,'dwt_watermarked.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el valor del PSNR para la imagen marcada
psnr = psnr(cover_object,watermarked_image,Mc,Nc)

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermarked_image_int,[])
% title('Imagen Marcada')
```

Extracción (dwt_recuperar.m)

%-----%

APÉNDICE B: MANUAL DE CÓDIGO

```
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: dwt_recuperar.m |%
%| Función: extracción de la marca de agua por el método |%
%| DWT. |%
%-----%

% se limpia todo
clear all;

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se lee la imagen marcada y se convierte a tipo double
file_name = 'dwt_watermarked.bmp';
watermarked_image = double(imread(file_name));

% carga de las imágenes atacadas
% file_name = 'dwt_JPEGattack.jpg';
% file_name = 'dwt_NOISEattack.bmp';
% file_name = 'dwt_MEANattack.bmp';
% file_name = 'dwt_CROPPINGattack.bmp';
% file_name = 'dwt_SCALEDattack.bmp';
% file_name = 'dwt_ROTATEattack.bmp';
% cd 'ImagenesAtaques'
% watermarked_image = double(imread(file_name));
% cd ..

% se determinan las dimensiones de la imagen marcada
[Mw,Nw] = size(watermarked_image);

% se lee la marca de agua original y se convierte a tipo double
file_name = 'copyright_50x20.bmp';
orig_watermark = double(imread(file_name));

% se determinan las dimensiones de la marca de agua original
[Mo,No] = size(orig_watermark);

% se lee una clave para el generador PN
file_name = 'key.bmp';
key = double(imread(file_name))./256;

% se reinicia el generador PN de MATLAB con "key"
rand('state',key(1));

% se inicializa la marca con todo a unos
message_vector = ones(1,Mo*No);

% se realiza la transformada de Haar
[cA1,cH1,cV1,cD1] = dwt2(watermarked_image,'haar');

% se añaden secuencias pn a los componentes H1 y V1 cuando message
= 0
for i = 1:length(message_vector)
    pn_sequence_h = round(2*(rand(Mw/2,Nw/2)-0.5));
    pn_sequence_v = round(2*(rand(Mw/2,Nw/2)-0.5));
```

APÉNDICE B: MANUAL DE CÓDIGO

```
correlation_h(i) = corr2(cH1,pn_sequence_h);
correlation_v(i) = corr2(cV1,pn_sequence_v);
correlation(i) = (correlation_h(i)+correlation_v(i))/2;
end

% si la correlación excede el umbral, se pone el bit a '0'
for i = 1:length(message_vector)
    if(correlation(i) > mean(correlation))
        message_vector(i) = 0;
    end
end

% se redimensiona la marca de agua en forma de matriz
watermark = reshape(message_vector,Mo,No)*255;

% se guarda la marca de agua en un fichero
cd 'ImagenesPruebas'
imwrite(watermark,'dwt_watermark.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el número de errores que contiene la marca extraída
errMatriz(orig_watermark,watermark,Mo,No);

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermark,[])
% title('Marca Recuperada')
```

B.1.5.2. Segundo Método: DWT y la transformada de Haar

Inserción (*dwtHaar_insertar.m*)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: dwtHaar_insertar.m |%
%| Función: inserción de una marca de agua por el método |%
%| de la DWT con la transformada de Haar. |%
%-----%

% se limpia todo
clear all;

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se inicializan los parámetros necesarios
key = 250;
tam = 1000;
```

APÉNDICE B: MANUAL DE CÓDIGO

```
s = 1;
t = 5;
alpha = (s+t)/2;

% se lee la imagen de cobertura
file_name = 'lena_256.bmp';
cover_object = double(imread(file_name));

% se determinan las dimensiones de la imagen de cobertura
[Mc,Nc] = size(cover_object);

% se realiza la DWT con la transformada de Haar
[LL,H,V,D] = dwt2(cover_object,'haar');

% se crea el bloque que contiene todo ceros
ceros = zeros(Mc/2,Nc/2);

% se hace la DWT inversa para crear la imagen de referencia la cual
solo
% contendrá las frecuencias bajas, LL, y el resto a ceros
referencia = idwt2(LL,ceros,ceros,ceros,'haar');

%se calcula cuántos elementos de la imagen sirven y su índice
contador = 0;
idx = [];
for i = 1:Mc
    for j = 1:Nc
        if((abs(cover_object(i,j))-referencia(i,j)) > s) &&
        (abs(cover_object(i,j))-referencia(i,j)) < t))
            idx = [idx;[i j]];
            contador = contador+1;
        end
    end
end

% se guarda lo calculado ya que hará falta para la extracción
cd 'Datos'
save 'datosDWTHaar.mat' contador idx key tam;
cd ..

% se reinicia el generador PN de MATLAB con "key"
rand('state',key);

% se crea la marca con 1000 números aleatorios
marca = ceil(2*rand(tam,1)-1);

% se hace una permutación aleatoria para escoger aquellos
coeficientes
% donde se va a ocultar la marca
permutable = randperm(contador);
permuta = permutable(1:tam);

% se crea la imagen marcada
watermarked_image = cover_object;
for k = 1:tam
    l = permuta(k);
    vector = idx(l,:);
    i = vector(1);
```

APÉNDICE B: MANUAL DE CÓDIGO

```
j = vector(2);
bit = marca(k);
if((cover_object(i,j) >= referencia(i,j)) && (bit == 0))
    watermarked_image(i,j) = referencia(i,j)-alpha;
end
if((cover_object(i,j) < referencia(i,j)) && (bit == 1))
    watermarked_image(i,j) = referencia(i,j)+alpha;
end
end

% se convierte la marca de agua a tipo uint8
watermarked_image_int = uint8(watermarked_image);

% se guarda la imagen marcada en un fichero
cd 'ImagenesPruebas'
imwrite(watermarked_image_int,'dwtHaar_watermarked.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el valor del PSNR para la imagen marcada
psnr = psnr(cover_object,watermarked_image,Mc,Nc)

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermarked_image_int,[])
% title('Imagen Marcada')
```

Extracción (dwtHaar_recuperar.m)

```
%-----
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----
%| Nombre código: dwtHaar_recuperar.m |%
%| Función: extracción de la marca de agua por el método |%
%| de la DWT con la transformada de Haar. |%
%-----

% se limpia todo
clear all;

% carga de los datos necesarios para la extracción
cd 'Datos'
load datosDWTHaar.mat;
cd ..

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se lee la imagen marcada y se le cambia el tipo a double
file_name = 'dwtHaar_watermarked.bmp';
```

APÉNDICE B: MANUAL DE CÓDIGO

```
watermarked_image = double(imread(file_name));

% carga de las imágenes atacadas
% file_name = 'dwtHaar_JPEGattack.jpg';
% file_name = 'dwtHaar_NOISEattack.bmp';
% file_name = 'dwtHaar_MEANattack.bmp';
% file_name = 'dwtHaar_CROPPINGattack.bmp';
% file_name = 'dwtHaar_SCALEDattack.bmp';
% file_name = 'dwtHaar_ROTATEattack.bmp';
% cd 'ImagenesAtaques'
% watermarked_image = double(imread(file_name));
% cd ..

% se determinan las dimensiones de la imagen marcada
[Mw,Nw] = size(watermarked_image);

% se calcula la imagen de referencia a partir de la marcada
[LL H V D] = dwt2(watermarked_image,'haar');
ceros = zeros(Mw/2,Nw/2);
referenciamarcada = idwt2(LL,ceros,ceros,ceros,'haar');

% se reinicia el generador PN de MATLAB con "key"
rand('state',key);

% se crea la marca con 'tam' números aleatorios
orig_watermark = ceil(2*rand(tam,1)-1);

% se realiza la permutación
permuta1 = randperm(contador);
permuta = permuta1(1:tam);

% se extrae la marca
watermark = zeros(tam,1);
for k = 1:tam
    l = permuta(k);
    vector = idx(l,:);
    i = vector(1);
    j = vector(2);
    if(watermarked_image(i,j) >= referenciamarcada(i,j))
        bit = 1;
    else
        bit = 0;
    end
    watermark(k) = bit;
end

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el número de errores que contiene la marca extraída
errVector(orig_watermark,watermark,tam);

% se calcula la similaridad
graf = similaridad(orig_watermark,watermark,tam);
title('Similaridad');

% se guarda la gráfica en un fichero
cd 'ImagenesPruebas'
```

APÉNDICE B: MANUAL DE CÓDIGO

```
I = getframe(gcf);
imwrite(I.cdata,'dwtHaar_similaridad.bmp');
cd ..
```

B.1.5.3. Tercer Método: DWT basado en la paridad

Inserción (*dwtParidad_insertar.m*)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Pérula Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: dwtParidad_insertar.m |%
%| Función: inserción de una marca de agua por el método |%
%| DWT basado en la paridad. |%
%-----%

% se limpia todo
clear all;

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se inicializan los parámetros necesarios
constante = 20;
tam = 128;
key = 250.;

% se guardan los datos que harán falta para la extracción
cd 'Datos'
save datosDWTParidad.mat constante tam key;
cd ..

% se lee la imagen de cobertura
file_name = 'lena_512.bmp';
cover_object = double(imread(file_name));

% se determinan las dimensiones de la imagen de cobertura
[Mc,Nc] = size(cover_object);

% se crea la marca de agua que será una matriz de 128 bits
pseudoaleatorios
rand('state',key);
marca = ceil(2*rand(tam*tam,1)-1);

% se realiza la DWT bidimensional en dos niveles
[cA1,cH1,cV1,cD1] = dwt2(cover_object,'haar');
[cA2,cH2,cV2,cD2] = dwt2(cA1,'haar');

coefmod = cA2;
k = 1;
for i = 1:tam
    for j = 1:tam
```

APÉNDICE B: MANUAL DE CÓDIGO

```
bit = marca(k);
k = k+1;
coef = floor(cA2(i,j)/constante);
if(bit ~= mod(coef,2))
    coefmod(i,j) = (coef+1)*constante;
end
end

% se reconstruye la imagen marcada
X2 = idwt2(coefmod,cH2,cV2,cD2,'haar');
watermarked_image dbl = idwt2(X2,cH1,cV1,cD1,'haar');
watermarked_image_int = uint8(watermarked_image dbl);

% se guarda la imagen marcada en un fichero
cd 'ImagenesPruebas'
imwrite(watermarked_image_int,'dwtParidad_watermarked.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el valor del PSNR para la imagen marcada
psnr = psnr(cover_object,watermarked_image dbl,Mc,Nc)

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermarked_image_int,[])
% title('Imagen Marcada')
```

Inserción modificada (dwtParidad_insertarmod.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: dwtParidad_insertarmod.m |%
%| Función: inserción de una marca de agua por el método |%
%| DWT basado en la paridad, modificado. |%
%-----%

% se limpia todo
clear all;

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se inicializan los parámetros necesarios
constante = 10;
tam = 128;
key = 250.;

% se guardan los datos que harán falta para la extracción
cd 'Datos'
```

APÉNDICE B: MANUAL DE CÓDIGO

```
save datosDWTParidad.mat constante tam key;
cd ..

% se lee la imagen de cobertura
file_name = 'lena_512.bmp';
cover_object = double(imread(file_name));

% se determinan las dimensiones de la imagen de cobertura
[Mc,Nc] = size(cover_object);

% se crea la marca de agua que será una matriz de 128 bits
pseudoaleatorios
rand('state',key);
marca = ceil(2*rand(tam*tam,1)-1);

% se realiza la DWT bidimensional en dos niveles
[cA1,cH1,cV1,cD1] = dwt2(cover_object,'haar');
[cA2,cH2,cV2,cD2] = dwt2(cA1,'haar');

coefmod = cA2;
k = 1;
for i = 1:tam
    for j = 1:tam
        bit = marca(k);
        k = k+1;
        coef = floor(cA2(i,j)/constante);
        if(bit ~= mod(coef,2))
            coefmod(i,j) = (coef+1)*constante+constante/2;
        end
    end
end

% se reconstruye la imagen marcada
X2 = idwt2(coefmod,cH2,cV2,cD2,'haar');
watermarked_image dbl = idwt2(X2,cH1,cV1,cD1,'haar');
watermarked_image_int = uint8(watermarked_image dbl);

% se guarda la imagen marcada en un fichero
cd 'ImagenesPruebas'
imwrite(watermarked_image_int,'dwtParidad_watermarkedMod.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el valor del PSNR para la imagen marcada
psnr = psnr(cover_object,watermarked_image dbl,Mc,Nc)

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermarked_image_int,[])
% title('Imagen Marcada')
```

Extracción (dwtParidad_recuperar.m)

APÉNDICE B: MANUAL DE CÓDIGO

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: dwtParidad_recuperar.m |%
%| Función: extracción de la marca de agua por el método |%
%| DWT basado en la paridad. |%
%-----%

% se limpia todo
clear all;

% se cargan los datos necesarios para la extracción
cd 'Datos'
load datosDWTParidad.mat;
cd ..

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se lee la imagen marcada y se le cambia el tipo a double
% file_name = 'dwtParidad_watermarked.bmp';
file_name = 'dwtParidad_watermarkedMod.bmp';
watermarked_image = double(imread(file_name));

% carga de las imágenes atacadas
% file_name = 'dwtParidad_JPEGattack.jpg';
% file_name = 'dwtParidad_NOISEattack.bmp';
% file_name = 'dwtParidad_MEANattack.bmp';
% file_name = 'dwtParidad_CROPPINGattack.bmp';
% file_name = 'dwtParidad_SCALEDattack.bmp';
% file_name = 'dwtParidad_ROTATEattack.bmp';
% cd 'ImagenesAtaques'
% watermark_image = double(imread(file_name));
% cd ..

% se crea la marca de agua que será un vector de 128 bits
pseudoaleatorios
rand('state',key);
orig_watermark = ceil(2*rand(tam*tam,1)-1);

[cA1,cH1,cV1,cD1] = dwt2(watermarked_image,'haar');
[cA2,cH2,cV2,cD2] = dwt2(cA1,'haar');

% se extrae la marca de agua
watermark = zeros(1,tam*tam);
k = 1;
for i = 1:tam
    for j = 1:tam
        coef = floor(cA2(i,j)/constante);
        if(mod(coef,2) == 1)
            watermark(k) = 1;
        end
        k = k+1;
    end
end
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se muestra el tiempo de computación  
elapsed_time = cputime-start_time  
  
% se calcula el número de errores que contiene la marca extraída  
errVector(orig_watermark,watermark,tam);  
  
% se calcula la similaridad  
graf = similaridad(orig_watermark,watermark,tam,tam);  
title('Similaridad');  
  
% se guarda la gráfica en un fichero  
cd 'ImagenesPruebas'  
I = getframe(gcf);  
% imwrite(I.cdata,'dwtParidad_similaridad.bmp');  
imwrite(I.cdata,'dwtParidad_similaridadMod.bmp');  
cd ..
```

B.1.6. Algoritmos basados en la SVD

B.1.6.1. Primer Método: SVD

Inserción (svd_insertar.m)

```
%-----%  
%| Nombre proyecto: Marcas de agua en imágenes digitales |%  
%| Autor: Raúl Péruela Martínez |%  
%| Año: 2009 |%  
%-----%  
%| Nombre código: svd_insertar.m |%  
%| Función: inserción de una marca de agua por el método |%  
%| SVD. |%  
%-----%  
  
% se limpia todo  
clear all;  
  
% se guarda el tiempo de comienzo de computación  
start_time = cputime;  
  
% se inicializan los parámetros necesarios  
key = 250;  
tam = 50;  
  
% se lee la imagen de cobertura  
file_name = 'lena_256.bmp';  
cover_object = double(imread(file_name));  
  
% se determinan las dimensiones de la imagen de cobertura  
[Mc,Nc] = size(cover_object);  
  
% se reinicia el generador PN de MATLAB con "key"  
randn('state',key);  
  
% se crea la marca con números aleatorios de N(0,1)  
watermark = randn(tam,tam);
```

APÉNDICE B: MANUAL DE CÓDIGO

```
marcagrande = zeros(Mc,Nc);
marcagrande(1:tam,1:tam) = watermark;

% se inserta la marca de agua
[U,S,V] = svd(cover_object);
D = S+0.2*marcagrande;
[UW,SW,VW] = svd(D);

% se guardan las matrices que harán falta para la extracción
cd 'Datos'
save datosSVD.mat UW VW S key tam;
cd ..

% se hace la operación inversa para recuperar la imagen marcada
watermarked_image = U*SW*(V');

% se convierte la imagen marcada a tipo uint8
watermarked_image_int = uint8(watermarked_image);

% se guarda la imagen marcada en un fichero
cd 'ImagenesPruebas'
imwrite(watermarked_image_int,'svd_watermarked.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el valor del PSNR para la imagen marcada
psnr = psnr(cover_object,watermarked_image,Mc,Nc)

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermarked_image_int,[])
% title('Imagen Marcada')
```

Extracción (svd_recuperar.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: svd_recuperar.m |%
%| Función: extracción de la marca de agua por el método |%
%| SVD. |%
%-----%

% se limpia todo
clear all;

% se cargan los datos necesarios para la extracción
cd 'Datos'
load datosSVD.mat;
cd ..
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se lee la imagen marcada y se le cambia el tipo a double
file_name = 'svd_watermarked.bmp';
watermarked_image = double(imread(file_name));

% carga de las imágenes atacadas
% file_name = 'svd_JPEGattack.jpg';
% file_name = 'svd_NOISEattack.bmp';
% file_name = 'svd_MEANattack.bmp';
% file_name = 'svd_CROPPINGattack.bmp';
% file_name = 'svd_SCALEDattack.bmp';
% file_name = 'svd_ROTATEattack.bmp';
% cd 'ImagenesAtaques'
% watermark = double(imread(file_name));
% cd ..

% se reinicia el generador PN de MATLAB con "key"
randn('state',key);

% se obtiene la marca original
orig_watermark = randn(tam,tam);

% se calcula los SVD de la imagen marcada
[U,SW,V] = svd(watermarked_image);

% se calcula la marca
D = UW*SW*(VW');
watermark = (1/0.2)*(D-S);
watermark = watermark(1:tam,1:tam);

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula la similaridad
graf = correlacion(orig_watermark,watermark,tam);
title('Correlacion');

% se guarda la gráfica en un fichero
cd 'ImagenesPruebas'
I = getframe(gcf);
imwrite(I.cdata,'svd_correlacion.bmp');
cd ..
```

B.1.6.2. Segundo Método: SVD con transformada de Arnold

Inserción (*svdArnold_insertar.m*)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
```

APÉNDICE B: MANUAL DE CÓDIGO

```
%| Nombre código: svdArnold_insertar.m |%
%| Función: inserción de una marca de agua por el método |%
%| SVD con la transformada de Arnold. |%
%-----%  
  
% se limpia todo  
clear all;  
  
% se guarda el tiempo de comienzo de computación  
start_time = cputime;  
  
% se lee la imagen de cobertura  
file_name = 'lena_512.bmp';  
cover_object = double(imread(file_name));  
  
% se determinan las dimensiones de la imagen de cobertura  
[Mc,Nc] = size(cover_object);  
  
% se lee la marca de agua y se redimensiona poniendolo como un  
vector  
file_name = 'nino_64.bmp';  
message = double(imread(file_name));  
  
% se determinan las dimensiones de la marca de agua  
[Mm,Nm] = size(message);  
  
% se redimensiona la marca en forma de vector  
message = message(:);  
  
% se crea la imagen marcada  
watermarked_image = zeros(Mc,Nc);  
alpha = 0.3;  
cont = 1;  
listaS = zeros(1,Mm*Nm);  
for i = 1:8:Mc  
    for j = 1:8:Nc  
        inter = cover_object(i:i+7,j:j+7);  
        [U,S,V] = svds(inter,8);  
        listaS(cont) = S(1,1);  
        S(1,1) = S(1,1)+alpha*message(cont);  
        cont = cont+1;  
        watermarked_image(i:i+7,j:j+7) = U*S*(V');  
    end  
end  
  
% se guardan los parámetros que harán falta para la extracción  
cd 'Datos'  
save datosSVDArnold.mat listaS alpha;  
cd ..  
  
% se convierte la marca de agua a tipo uint8  
watermarked_image_int = uint8(watermarked_image);  
  
% se guarda la imagen marcada en un fichero  
cd 'ImagenesPruebas'  
imwrite(watermarked_image_int,'svdArnold_watermarked.bmp');  
cd ..
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se muestra el tiempo de computación  
elapsed_time = cputime-start_time  
  
% se calcula el valor del PSNR para la imagen marcada  
psnr = psnr(cover_object,watermarked_image,Mc,Nc)  
  
% se muestra la imagen marcada por pantalla  
% figure  
% imshow(watermarked_image_int,[])  
% title('Imagen Marcada')
```

Extracción (svdArnold_recuperar.m)

```
%-----%  
%| Nombre proyecto: Marcas de agua en imágenes digitales |%  
%| Autor: Raúl Péruela Martínez |%  
%| Año: 2009 |%  
%-----%  
%| Nombre código: svdArnold_recuperar.m |%  
%| Función: extracción de la marca de agua por el método |%  
%| SVD con la transformada de Arnold. |%  
%-----%  
  
% se limpia todo  
clear all;  
  
% carga de los datos necesarios para la extracción  
cd 'Datos'  
load datosSVDArnold.mat;  
cd ..  
  
% se guarda el tiempo de comienzo de computación  
start_time = cputime;  
  
% se lee la imagen marcada y se le cambia el tipo a double  
file_name = 'svdArnold_watermarked.bmp';  
watermarked_image = double(imread(file_name));  
  
% carga de las imágenes atacadas  
% file_name = 'svdArnold_JPEGattack.jpg';  
% file_name = 'svdArnold_NOISEattack.bmp';  
% file_name = 'svdArnold_MEANattack.bmp';  
% file_name = 'svdArnold_CROPPINGattack.bmp';  
% file_name = 'svdArnold_SCALEDattack.bmp';  
% file_name = 'svdArnold_ROTATEattack.bmp';  
% cd 'ImagenesAtaques'  
% watermarkedImage = double(imread(file_name));  
% cd ..  
  
% se determinan las dimensiones de la imagen marcada  
[Mw,Nw] = size(watermarked_image);  
  
% se lee la marca original y se le cambia el tipo a double  
file_name = 'nino_64.bmp';  
orig_watermark = double(imread(file_name));
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se determinan las dimensiones de la marca original  
[Mo, No] = size(orig_watermark);  
  
% se extrae la marca de la imagen  
message_vector = zeros(1, Mo*No);  
cont = 1;  
for i = 1:8:Mw  
    for j = 1:8:Nw  
        inter = watermarked_image(i:i+7, j:j+7);  
        [U, S, V] = svds(inter, 8);  
        message_vector(cont) = (S(1,1)-listaS(cont))/alpha;  
        cont = cont+1;  
    end  
end  
  
% se redimensiona la marca para ponerlo como matriz y a tipo uint8  
watermark = uint8(reshape(message_vector, [Mo No]));  
  
% se guarda la marca de agua en un fichero  
cd 'ImagenesPruebas'  
imwrite(watermark, 'svdArnold_watermark.bmp');  
cd ..  
  
% se muestra el tiempo de computación  
elapsed_time = cputime-start_time  
  
% se calcula el número de errores que contiene la marca extraída  
errMatriz(orig_watermark, watermark, Mo, No);  
  
% se muestra la imagen marcada por pantalla  
% figure  
% imshow(watermark, [])  
% title('Marca Recuperada')
```

B.1.6.3. Tercer Método: SVD basado en el intercambio de valores

Inserción (*svdIntercambio_insertar.m*)

```
%-----%  
%| Nombre proyecto: Marcas de agua en imágenes digitales |%  
%| Autor: Raúl Péruela Martínez |%  
%| Año: 2009 |%  
%-----%  
%| Nombre código: svdIntercambio_insertar.m |%  
%| Función: inserción de una marca de agua por el método |%  
%| SVD basado en el intercambio de valores. |%  
%-----%  
  
% se limpia todo  
clear all;  
  
% se guarda el tiempo de comienzo de computación  
start_time = cputime;
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se inicializa el parámetro necesario
alpha = 2;

% se guarda el parametro necesario para la extracción
cd 'Datos'
save datosSVDIntercambio.mat alpha;
cd ..

% se lee la imagen de cobertura
file_name = 'lena_512.bmp';
cover_object = double(imread(file_name));

% se determinan las dimensiones de la imagen de cobertura
[Mc,Nc] = size(cover_object);

% se lee la marca de agua y se convierte a tipo double, la marca
que se
% insertara sera la repeticion de la original
file_name = 'mark_64.bmp';
marca = double(imread(file_name));
marca = marca(:)/255.;
marca = [marca marca marca marca];

% se crea la imagen marcada
watermarked_image dbl = zeros(512,512);
k = 1;
for i = 1:4:512
    for j = 1:4:512
        A = cover_object(i:i+3,j:j+3);
        bit = marca(k);
        [U,S,V] = svd(A);
        S(3,3) = S(2,2);
        S(2,2) = S(2,2)+alpha*bit;
        if(S(1,1) < S(2,2))
            S(1,1) = S(2,2);
        end
        Amod = U*S*(V');
        watermarked_image dbl(i:i+3,j:j+3) = Amod;
        k = k+1;
    end
end

% se convierte la imagen marcada a tipo uint8
watermarked_image_int = uint8(watermarked_image dbl);

% se guarda la imagen marcada en un fichero
cd 'ImagenesPruebas'
imwrite(watermarked_image_int,'svdIntercambio_watermarked.bmp');
cd .. 

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el valor del PSNR para la imagen marcada
psnr = psnr(cover_object,watermarked_image dbl,Mc,Nc)

% se muestra la imagen marcada por pantalla
% figure
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% imshow(watermarked_image_int,[])
% title('Imagen Marcada')
```

Extracción (svdIntercambio_recuperar.m)

```
%-----
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----
%| Nombre código: svdIntercambio_recuperar.m |%
%| Función: extracción de la marca de agua por el método |%
%| de SVD basado en el intercambio de valores. |%
%-----

% se limpia todo
clear all;

% se cargan los datos necesarios para la extracción
cd 'Datos'
load datosSVDIntercambio.mat;
cd ..

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se lee la imagen marcada y se le cambia el tipo a double
file_name = 'svdIntercambio_watermarked.bmp';
watermarked_image = double(imread(file_name));

% carga de las imágenes atacadas
% file_name = 'svdIntercambio_JPEGatack.jpg';
% file_name = 'svdIntercambio_NOISEatack.bmp';
% file_name = 'svdIntercambio_MEANatack.bmp';
% file_name = 'svdIntercambio_CROPPINGatack.bmp';
% file_name = 'svdIntercambio_SCALEDatack.bmp';
% file_name = 'svdIntercambio_ROTATEatack.bmp';
% cd 'ImagenesAtaques'
% watermark = double(imread(file_name));
% cd ..

% se determinan las dimensiones de la imagen marcada
[Mw,Nw] = size(watermarked_image);

% se lee la marca de agua original y se le cambia el tipo a double
file_name = 'mark_64.bmp';
orig_watermark = imread(file_name);

% se determinan las dimensiones de la marca original
[Mo,No] = size(orig_watermark);

% se extrae la marca de agua
watermark = zeros(1,Mo*No*3);
k = 1;
for i = 1:4:Mw
```

APÉNDICE B: MANUAL DE CÓDIGO

```
for j = 1:4:Nw
    A = watermarked_image(i:i+3,j:j+3);
    [U,S,V] = svd(A);

    if((S(2,2)-S(3,3)) > alpha/2)
        watermark(k) = 1;
    else
        watermark(k) = 0;
    end
    k = k+1;
end
end

marcarecu = zeros(1,Mo*No);
for i = 1:Mo*No
    marcarecu(i) =
watermark(i)+watermark(i+Mo*No)+watermark(i+2*Mo*No);
    if(marcarecu(i) >= 2)
        marcarecu(i) = 1;
    else
        marcarecu(i) = 0;
    end
end

% se redimensiona y cambia el tipo de la marca de agua extraída
marcarecu = reshape(marcarecu, [Mo, No]);
marcarecu = uint8(marcarecu)*255;

% se guarda la marca de agua en un fichero
cd 'ImagenesPruebas'
imwrite(marcarecu,'svdIntercambio_watermark.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el número de errores que contiene la marca extraída
errMatriz(orig_watermark,marcarecu,Mo,No);

% se muestra la imagen marcada por pantalla
% figure
% imshow(marcarecu,[])
% title('Marca Recuperada')
```

Comentarios

Ataque 1 (svdIntercambio_ataque.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: svdIntercambio_ataque.m |%
%| Función: ataque del método SVD basado en el |%
```

APÉNDICE B: MANUAL DE CÓDIGO

```
%|           intercambio de valores. |%
%-----%
% se limpia todo
clear all;

% se lee la imagen marcada y se le cambia el tipo a double
file_name = 'svdIntercambio_watermarked.bmp';
watermarked_image = double(imread(file_name));

% se determinan las dimensiones de la imagen marcada
[Mw,Nw] = size(watermarked_image);

% se realiza el ataque
atacada = watermarked_image;
for i = 1:4:Mw
    for j = 1:4:Nw
        A = watermarked_image(i:i+3,j:j+3);
        [U,S,V] = svd(A);
        S(2,2) = S(3,3);
        A = U*S*(V');
        atacada(i:i+3,j:j+3) = A;
    end
end

% se cambia el tipo a uint8
atacada = uint8(atacada);

% se guarda la imagen atacada en un fichero
cd 'ImagenesPruebas'
imwrite(atacada,'svdIntercambio_ataque.bmp');
cd ..

% se muestra la imagen atacada por pantalla
figure
imshow(atacada,[])
title('Imagen Atacada')
```

Ataque 2(svdIntercambio_ataque2.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor:          Raúl Péruela Martínez |%
%| Año:            2009 |%
%-----%
%| Nombre código: svdIntercambio_ataque.m |%
%| Función:       ataque del método SVD basado en el |%
%|               intercambio de valores. |%
%-----%
```

```
% se limpia todo
clear all;

% se inicializa el parámetro necesario
alpha = 20;
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se lee la imagen marcada y se le cambia el tipo a double
file_name = 'svdIntercambio_watermarked.bmp';
image = double(imread(file_name));

% se determinan las dimensiones de la imagen marcada
[Mw,Nw] = size(image);

% se lee la marca de agua original y se le cambia el tipo a double
file_name = 'EPS_64.bmp';
marca = double(imread(file_name));
marca = marca(:)/255.;
marca = [marca marca marca marca];

% se realiza el ataque
watermarked_image = zeros(Mw,Nw);
k = 1;
for i = 1:4:Mw
    for j = 1:4:Nw
        A = image(i:i+3,j:j+3);
        bit = marca(k);
        [U,S,V] = svd(A);
        S(3,3) = S(2,2);
        S(2,2) = S(2,2)+alpha*bit;
        if(S(1,1) < S(2,2))
            S(1,1) = S(2,2);
        end
        Amod = U*S*(V');
        watermarked_image(i:i+3,j:j+3) = Amod;
        k = k+1;
    end
end

% se cambia el tipo a uint8
watermarked_image = uint8(watermarked_image);

% se guarda la imagen atacada en un fichero
cd 'ImagenesPruebas'
imwrite(watermarked_image,'svdIntercambio_ataque2.bmp');
cd ..

% se muestra la imagen atacada por pantalla
figure
imshow(watermarked_image,[])
title('Imagen Atacada')
```

B.1.6.4. Cuarto Método: SVD basado en el orden en los coeficientes

Inserción (svdOrden_insertar.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
```

APÉNDICE B: MANUAL DE CÓDIGO

```
%| Nombre código: svdOrden_insertar.m |%
%| Función: inserción de una marca de agua por el método |%
%| SVD basado en el orden de los coeficientes. |%
%-----%
% se limpia todo
clear all;

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se lee la imagen de cobertura
file_name = 'lena_256.bmp';
cover_object = double(imread(file_name));

% se determinan las dimensiones de la imagen de cobertura
[Mc,Nc] = size(cover_object);

% se lee la marca de agua
file_name = 'hola_32.bmp';
marca = imread(file_name)/255;
marca = marca(:);

% se establece el valor del umbral
T = 0.012;

% se crea la imagen marcada
watermarked_image = cover_object;
l = 1;
for i = 1:8:Mc
    for j = 1:8:Nc
        bit = marca(l);
        A = cover_object(i:i+7,j:j+7);
        [U,S,V] = svd(A);
        udif = U(2,1)-U(3,1);
        ptomedio = (U(2,1)+U(3,1))/2;
        if((bit == 1) && (udif < T))
            U(2,1) = ptomedio+T/2;
            U(3,1) = ptomedio-T/2;
        end
        if((bit == 0) && (udif > -T))
            U(2,1) = ptomedio-T/2;
            U(3,1) = ptomedio+T/2;
        end
        l = l+1;
        B = U*S*(V');
        watermarked_image(i:i+7,j:j+7) = B;
    end
end

% se convierte la imagen marcada a tipo uint8
watermarked_image_int = uint8(watermarked_image);

% se guarda la imagen marcada en un fichero
cd 'ImagenesPruebas'
imwrite(watermarked_image_int,'svdOrden_watermarked.bmp');
cd ..
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se muestra el tiempo de computación  
elapsed_time = cputime-start_time  
  
% se calcula el valor del PSNR para la imagen marcada  
psnr = psnr(cover_object,watermarked_image,Mc,Nc)  
  
% se muestra la imagen marcada por pantalla  
% figure  
% imshow(watermarked_image_int,[])  
% title('Imagen Marcada')
```

Extracción (*svdOrden_recuperar.m*)

```
%-----%  
%| Nombre proyecto: Marcas de agua en imágenes digitales |%  
%| Autor: Raúl Péruela Martínez |%  
%| Año: 2009 |%  
%-----%  
%| Nombre código: svOrden_recuperar.m |%  
%| Función: extracción de la marca de agua por el método |%  
%| SVD basado en el orden de los coeficientes. |%  
%-----%  
  
% se limpia todo  
clear all;  
  
% se guarda el tiempo de comienzo de computación  
start_time = cputime;  
  
% se lee la imagen marcada y se le cambia el tipo a double  
file_name = 'svdOrden_watermarked.bmp';  
watermarked_image = double(imread(file_name));  
  
% carga de las imágenes atacadas  
% file_name = 'svdOrden_JPEGattack.jpg';  
% file_name = 'svdOrden_NOISEattack.bmp';  
% file_name = 'svdOrden_MEANattack.bmp';  
% file_name = 'svdOrden_CROPPINGattack.bmp';  
% file_name = 'svdOrden_SCALEDattack.bmp';  
% file_name = 'svdOrden_ROTATEattack.bmp';  
% cd 'ImagenesAtaques'  
% watermarked_image = double(imread(file_name));  
% cd ..  
  
% se determinan las dimensiones de la imagen marcada  
[Mw,Nw] = size(watermarked_image);  
  
% se lee la marca da agua original y se convierte a tipo double  
file_name = 'hola_32.bmp';  
orig_watermark = imread(file_name);  
  
% se determinan las dimensiones de la marca de agua original  
[Mo,No] = size(orig_watermark);  
  
% se extrae la marca de agua de la imagen
```

APÉNDICE B: MANUAL DE CÓDIGO

```
lista = zeros(1,Mo*No);
k = 1;
for i = 1:8:Mw
    for j = 1:8:Nw
        A = watermarked_image(i:i+7,j:j+7);
        [U,S,V] = svd(A);
        if((U(2,1)-U(3,1)) >= 0)
            lista(k) = 1;
        end
        k = k+1;
    end
end

% se redimensiona y cambia el tipo de la marca de agua extraída
watermark = reshape(lista,[Mo, No])*255;
watermark = uint8(watermark);

% se guarda la marca de agua en un fichero
cd 'ImagenesPruebas'
imwrite(watermark,'svdOrden_watermark.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el número de errores que contiene la marca extraída
errMatriz(orig_watermark,watermark,Mo,No);

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermark,[])
% title('Marca Recuperada')
```

B.1.6.5. Quinto Método: SVD basado en la proximidad a un intervalo

Inserción (*svdProximidad_insertar.m*)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: svdProximidad_insertar.m |%
%| Función: inserción de una marca de agua por el método |%
%| SVD basado en la proximidad a un intervalo. |%
%-----%

% se limpia todo
clear all;

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se lee la imagen de cobertura
file_name = 'lena_256.bmp';
```

APÉNDICE B: MANUAL DE CÓDIGO

```
cover_object = double(imread(file_name));

% se determinan las dimensiones de la imagen de cobertura
[Mc,Nc] = size(cover_object);

% se lee la marca de agua y se convierte a tipo double
file_name = 'holo_32.bmp';
watermark = imread(file_name)/255;

% se determinan las dimensiones de la marca de agua
[Mm,Nm] = size(watermark);
watermark = watermark(:);

% se extraen los valores singulares
k = 1;
lista = zeros(1,Mm*Nm);
for i = 1:8:Mc
    for j = 1:8:Nc
        A = cover_object(i:i+7,j:j+7);
        [U,S,V] = svd(A);
        lista(k) = S(1,1);
        k = k+1;
    end
end

T = 60;
dmin = min(lista);
dmax = max(lista);
n = floor((dmax-dmin+2*T)/T)+1;

bin = zeros(n,2);
bin(1,1) = dmin-T;
bin(1,2) = dmin;
for i = 2:n
    bin(i,1) = bin(i-1,2);
    bin(i,2) = bin(i,1)+T;
end

% se crea la imagen marcada
watermarked_image = cover_object;
k = 1;
limiteinferior = dmin-T;
for i = 1:8:Mc
    for j = 1:8:Nc
        A = cover_object(i:i+7,j:j+7);
        [U,S,V] = svd(A);
        % se averigua en que subintervalo se encuentra
        h = floor((S(1,1)-limiteinferior)/T);
        bit = watermark(k);
        if(bit == 1)
            S(1,1) = (bin(h,1)+(bin(h,2)+bin(h,1))/2)/2;
        else
            S(1,1) = (bin(h,2)+(bin(h,2)+bin(h,1))/2)/2;
        end
        Amod = U*S*(V');
        watermarked_image(i:i+7,j:j+7) = Amod;
        k = k+1;
    end
end
```

APÉNDICE B: MANUAL DE CÓDIGO

```
end

% se guardan los parámetros que harán falta para la extracción
cd 'Datos'
save 'datosSVDProximidad.mat' bin limiteinferior T;
cd ..

% se convierte la imagen marcada a tipo uint8
watermarked_image_int = uint8(watermarked_image);

% se guarda la imagen marcada en un fichero
cd 'ImagenesPruebas'
imwrite(watermarked_image_int,'svdProximidad_watermarked.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el valor del PSNR para la imagen marcada
psnr = psnr(cover_object,watermarked_image,Mc,Nc)

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermarked_image_int,[])
% title('Imagen Marcada')
```

Extracción (svdProximidad_recuperar.m)

```
%-----
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----
%| Nombre código: svdProximidad_recuperar.m |%
%| Función: extraccion de la marca de agua por el método |%
%| SVD basado en la proximidad a un intervalo. |%
%-----

% se limpia todo
clear all;

% se cargan los datos necesarios para la extracción
cd 'Datos'
load 'datosSVDProximidad.mat'
cd ..

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se lee la imagen marcada y se le cambia el tipo a double
file_name = 'svdProximidad_watermarked.bmp';
watermarked_image = double(imread(file_name));

% carga de las imágenes atacadas
% file_name = 'svdProximidad_JPEGatack.jpg';
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% file_name = 'svdProximidad_NOISEattack.bmp';
% file_name = 'svdProximidad_MEANattack.bmp';
% file_name = 'svdProximidad_CROPPINGattack.bmp';
% file_name = 'svdProximidad_SCALEDattack.bmp';
% file_name = 'svdProximidad_ROTATEattack.bmp';
% cd 'ImagenesAtaques'
% watermarked_image = double(imread(file_name));
% cd ..

% se determinan las dimensiones de la imagen marcada
[Mw,Nw] = size(watermarked_image);

% se lee la marca da agua original y se convierte a tipo double
file_name = 'hola_32.bmp';
orig_watermark = imread(file_name);

% se determinan las dimensiones de la marca de agua original
[Mo,No] = size(orig_watermark);

% se extrae la marca de agua
extraida = zeros(1,Mo*No);
k = 1;
for i = 1:8:Mw
    for j = 1:8:Nw
        A = watermarked_image(i:i+7,j:j+7);
        [U,S,V] = svd(A);
        % se averigua en que subintervalo se encuentra
        h = floor((S(1,1)-limiteinferior)/T)+1;
        if(h == 0)
            h = 1;
        end
        ptomedio = (bin(h,2)+bin(h,1))/2;
        if(S(1,1) < ptomedio)
            extraida(k) = 1;
        end
        k = k+1;
    end
end

% se redimensiona la marca de agua en forma de matriz
watermark = reshape(extraida, [Mo, No])*255;

% se guarda la marca de agua en un fichero
cd 'ImagenesPruebas'
imwrite(watermark,'svdProximidad_watermark.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el número de errores que contiene la marca extraida
errMatriz(orig_watermark,watermark,Mo,No);

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermark,[])
% title('Marca Recuperada')
```

B.1.7. Algoritmo basado en LSB

Inserción (lsb_insertar.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: lsb_insertar.m |%
%| Función: inserción de una marca de agua por el método |%
%| LSB. |%
%-----%

% se limpia todo
clear all;

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se lee la imagen de cobertura
file_name = 'lena_512.bmp';
cover_object = imread(file_name);

% se determinan las dimensiones de la imagen de cobertura
[Mc,Nc] = size(cover_object);

% se lee la marca de agua
file_name = 'copyright_50x20.bmp';
message = imread(file_name);

% se convierte la marca a tipo double para normalizarla, después se
vuelve
% a convertir a tipo uint8
message = double(message);
message = round(message./256);
message = uint8(message);

% se determinan las dimensiones de la marca
[Mm,Nm] = size(message);

% se crea la nueva marca a partir de la marca anterior, se repite
esta
% hasta que tenga el mismo tamaño que la imagen original
for i = 1:Mc
    for j = 1:Nc
        watermark(i,j) = message(mod(i,Mm)+1, mod(j,Nm)+1);
    end
end

% ahora se ponen los lsb de cover_object(i,j) para los valores de
% watermark(i,j)
watermarked_image = cover_object;
for i = 1:Mc
    for j = 1:Nc
```

APÉNDICE B: MANUAL DE CÓDIGO

```
    watermarkedImage(i,j) =
bitset(watermarkedImage(i,j),1,watermark(i,j));
end
end

% se guarda la imagen marcada en un fichero
cd 'ImagenesPruebas'
imwrite(watermarkedImage,'lsb_watermarked.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el valor del PSNR para la imagen marcada
psnr = psnr(cover_object,watermarkedImage,Mc,Nc)

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermarkedImage,[])
% title('Imagen Marcada')
```

Extracción (lsb_recuperar.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: lsb_recuperar.m |%
%| Función: extracción de la marca de agua por el método |%
%| LSB. |%
%-----%

% se limpia todo
clear all;

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se lee la imagen marcada
file_name = 'lsb_watermarked.bmp';
watermarkedImage = imread(file_name);

% carga de las imágenes atacadas
% file_name = 'lsb_JPEGattack.jpg';
% file_name = 'lsb_NOISEattack.bmp';
% file_name = 'lsb_MEANattack.bmp';
% file_name = 'lsb_CROPPINGattack.bmp';
% file_name = 'lsb_SCALEDattack.bmp';
% file_name = 'lsb_ROTATEattack.bmp';
% cd 'ImagenesAtaques'
% watermarkedImage = double(imread(file_name));
% cd ..

% se determinan las dimensiones de la imagen marcada
```

APÉNDICE B: MANUAL DE CÓDIGO

```
[Mw,Nw] = size(watermarked_image);

% se lee la marca de agua original y se convierte a tipo double
file_name = 'copyright_512.bmp';
orig_watermark = double(imread(file_name));

% se determinan las dimensiones de la marca original
[Mo,No] = size(orig_watermark);

% se usan los lsb de la imagen marcada para recuperar la marca de
agua
for i = 1:Mw
    for j = 1:Nw
        watermark(i,j) = bitget(watermarked_image(i,j),1);
    end
end

% se escala la marca de agua recuperada
watermark = double(watermark)*255;

% se guarda la marca de agua en un fichero
cd 'ImagenesPruebas'
imwrite(watermark,'lsb_watermark.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el número de errores que contiene la marca extraída
errMatriz(orig_watermark,watermark,Mo,No);

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermark,[])
% title('Marca Recuperada')
```

B.1.8. Algoritmo basado en las Secuencias Caóticas

B.1.8.1. Primer Método: Secuencias Caóticas

Inserción (sc_insertar.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: SC_insertar.m |%
%| Función: inserción de una marca de agua por el método |%
%| de las secuencias caóticas. |%
%-----%

% se limpia todo
clear all;
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se inicializan los parámetros necesarios
lambda = 3.98;
alpha = 10;
umbral = 0.5;

% se guardan los datos que harán falta para la extracción
cd 'Datos'
save datosSC.mat lambda umbral;
cd ..

% se lee la imagen de cobertura
file_name = 'lena_256.bmp';
cover_object = double(imread(file_name));

% se determinan las dimensiones de la imagen de cobertura
[Mc,Nc] = size(cover_object);

% se genera la marca pseudoaleatoria
x = zeros(1,Mc*Nc);
x(1) = 0.1;
for i = 2:Mc*Nc
    x(i) = lambda*x(i-1)*(1-x(i-1));
end

w = zeros(1,Mc*Nc);
for i = 1:Mc*Nc
    if(x(i) < umbral)
        w(i) = 1;
    end
end

% se redimensiona la marca en forma de matriz
W = reshape(w,[Mc,Nc]);

% se añade la máscara de marca a la imagen de cobertura usando el
% factor de
% ganancia k
watermarked_image dbl = cover_object+alpha*W;
watermarked_image_int = uint8(watermarked_image dbl);

% se guarda la imagen marcada en un fichero
cd 'ImagenesPruebas'
imwrite(watermarked_image_int,'sc_watermarked.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el valor del PSNR para la imagen marcada
psnr = psnr(cover_object,watermarked_image dbl,Mc,Nc)

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermarked_image_int,[])
% title('Imagen Marcada')
```

Extracción (*sc_recuperar.m*)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Pérula Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: SC_recuperar.m |%
%| Función: extracción de la marca de agua por el método |%
%| de las secuencias caóticas. |%
%-----%

% se limpia todo
clear all;

% se cargan los datos necesarios para la extracción
cd 'Datos'
load datosSC.mat;
cd ..

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se lee la imagen marcada y se le cambia el tipo a double
file_name = 'sc_watermarked.bmp';
watermarked_image = double(imread(file_name));

% carga de las imágenes atacadas
% file_name = 'sc_JPEGatack.jpg';
% file_name = 'sc_NOISEatack.bmp';
% file_name = 'sc_MEANatack.bmp';
% file_name = 'sc_CROPPINGatack.bmp';
% file_name = 'sc_SCALEDatack.bmp';
% file_name = 'sc_ROTATEatack.bmp';
% cd 'ImagenesAtaques'
% watermarked_image = double(imread(file_name));
% cd ..

% se determinan las dimensiones de la imagen marcada
[Mw,Nw] = size(watermarked_image);

% se hace la simulación utilizando el vector de correlaciones entre
% las
% marcas creadas y la original
corre = zeros(1,50);
for k = 1:50
    x = zeros(1,Mw*Nw);
    x(1) = double(k);
    x(1) = x(1)/100;
    for i = 2:Mw*Nw
        x(i) = lambda*x(i-1)*(1-x(i-1));
    end
end

w = zeros(1,Mw*Nw);
```

APÉNDICE B: MANUAL DE CÓDIGO

```
for i = 1:Mw*Nw
    if(x(i) < umbral)
        w(i) = 1;
    end
end

W = reshape(w, [Mw,Nw]);

% se crea el vector de correlaciones de la imagen marcada con las
marcas
corre(k) = corr2(W,watermarked_image);
end

x = 1:50;
graf = plot(x,corre);
title('Correlacion');

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se guarda la gráfica en un fichero
cd 'ImagenesPruebas'
I = getframe(gcf);
imwrite(I.cdata,'sc_correlacion.bmp');
cd ..
```

B.1.8.2. Segundo Método: Secuencias Caóticas y DCT

Inserción (scdct_insertar.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: scdct_insertar.m |%
%| Función: inserción de una marca de agua por el método |%
%| de las secuencias caóticas y DCT. |%
%-----%

% se limpia todo
clear all;

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se inicializan los parámetros necesarios
tam = 1000;
alpha = 0.3;
key = 250.;

% se guardan las variables que harán falta para la extracción
cd 'Datos'
save 'datosSCDCT.mat' tam key;
cd ..
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se lee la imagen de cobertura
file_name = 'lena_256.bmp';
cover_object = double(imread(file_name));

% se determinan las dimensiones de la imagen de cobertura
[Mc,Nc] = size(cover_object);

ma = [1 1; 2 3];      % matriz inicial
K = [115 84;168 27]; % matriz para realizar la permutación
directamente

% se realizan las 42 iteraciones de Arnold
permutada = zeros(Mc,Nc);
for i = 1:Mc
    for j = 1:Nc
        ic = i-1;
        jc = j-1;
        nuevos = mod(K*[ic;jc],Mc);
        permutada(nuevos(1)+1,nuevos(2)+1) = cover_object(i,j);
    end
end

% se calcula la DCT
trans = dct2(permutada);

% se genera la marca con números pseudoaleatorios de una N(0,1)
randn('state',key);
marca = randn(tam,1);

% se ponen los coeficientes en zig-zag
transmod = trans;
k = 1;
for i = 2:Mc
    for j = 1:i
        if(k <= tam)
            transmod(i,j) = trans(i,j)+alpha*marca(k)*abs(trans(i,j));
            k = k+1;
        end
    end
end

%se realiza la transformada inversa
inversa = idct2(transmod);

% se deshace la permutación
K = [27 172;88 115]; % matriz para deshacer la permutación
directamente
watermarked_image_dbl = zeros(Mc,Nc);
for i = 1:Mc
    for j = 1:Nc
        ic = i-1;
        jc = j-1;
        nuevos = mod(K*[ic;jc],Mc);
        watermarked_image_dbl(nuevos(1)+1,nuevos(2)+1) = inversa(i,j);
    end
end
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se convierte la imagen marcada a tipo uint8
watermarked_image_int = uint8(watermarked_image dbl);

% se guarda la imagen marcada en un fichero
cd 'ImagenesPruebas'
imwrite(watermarked_image_int,'scdct_watermarked.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el valor del PSNR para la imagen marcada
psnr = psnr(cover_object,watermarked_image dbl,Mc,Nc)

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermarked_image_int,[])
% title('Imagen Marcada')
```

Extracción (scdct_recuperar.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: scdct_recuperar.m |%
%| Función: extracción de la marca de agua por el método |%
%| de las secuencias caóticas y la DCT. |%
%-----%

% se limpia todo
clear all;

% se cargan los datos necesarios para la extracción
cd 'Datos'
load 'datosSCDCT.mat'
cd ..

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se lee la imagen marcada y se le cambia el tipo a double
file_name = 'scdct_watermarked.bmp';
watermarked_image = double(imread(file_name));

% carga de las imágenes atacadas
% file_name = 'scdct_JPEGattack.jpg';
% file_name = 'scdct_NOISEattack.bmp';
% file_name = 'scdct_MEANattack.bmp';
% file_name = 'scdct_CROPPINGattack.bmp';
% file_name = 'scdct_SCALEDatack.bmp';
% file_name = 'scdct_ROTATEattack.bmp';
% cd 'ImagenesAtaques'
% watermarked_image = double(imread(file_name));
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% cd ..

% se determinan las dimensiones de la imagen marcada
[Mw,Nw] = size(watermarked_image);

% se genera la marca con números pseudoaleatorios de una N(0,1)
randn('state',key);
orig_watermark = randn(tam,1);

ma = [1 1; 2 3];      % matriz inicial
K = [115 84;168 27]; % matriz para realizar la permutación
directamente

% se realizan las 42 iteraciones de Arnold
permutada = zeros(Mw,Nw);
for i = 1:Mw
    for j = 1:Nw
        ic = i-1;
        jc = j-1;
        nuevos = mod(K*[ic;jc],256);
        permutada(nuevos(1)+1,nuevos(2)+1) = watermarked_image(i,j);
    end
end

% se calcula la DCT
trans = dct2(permutada);

% se extraen los coeficientes en zig-zag, se extraen los
coeficientes DCT
% que se usaron para ocultar la marca
watermark = zeros(tam,1);
k = 1;
for i = 2:Mw
    for j = 1:i
        if(k <= tam)
            watermark(k) = trans(i,j);
            k = k+1;
        end
    end
end

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula la similaridad
graf = correlacion(orig_watermark,watermark,tam,1);
title('Correlacion');

% se guarda la gráfica en un fichero
cd 'ImagenesPruebas'
I = getframe(gcf);
imwrite(I.cdata,'scdct_correlacion.bmp');
cd ..
```

Inserción (scdct_insertar_sin_permutacion.m)

APÉNDICE B: MANUAL DE CÓDIGO

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: scdct_insertar_sin_permutacion.m |%
%| Función: inserción de una marca de agua por el método |%
%| de las secuencias caóticas y la DCT, sin |%
%| permutación. |%
%-----%

% se limpia todo
clear all;

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se inicializan los parámetros necesarios
tam = 1000;
alpha = 1;
key = 250.;

% se guardan las variables que harán falta para la extracción
cd 'Datos'
save 'datosSCDCT_noper.mat' tam key;
cd ..

% se lee la imagen de cobertura
file_name = 'lena_256.bmp';
cover_object = double(imread(file_name));

% se determinan las dimensiones de la imagen de cobertura
[Mc,Nc] = size(cover_object);

% se calcula la DCT
trans = dct2(cover_object);

% se genera la marca con números pseudoaleatorios de una N(0,1)
randn('state',key);
marca = randn(tam,1);

% se ponen los coeficientes en zig-zag
transmod = trans;
k = 1;
for i = 2:Mc
    for j = 1:i
        if(k <= tam)
            transmod(i,j) = trans(i,j)+alpha*marca(k)*abs(trans(i,j));
            k = k+1;
        end
    end
end

% se realiza la transformada inversa
watermarked_image_dbl = idct2(transmod);

% se convierte la imagen marcada a tipo uint8
watermarked_image_int = uint8(watermarked_image_dbl);
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se guarda la imagen marcada en un fichero  
cd 'ImagenesPruebas'  
imwrite(watermarked_image_int,'scdct_noper_watermarked.bmp');  
cd ..  
  
% se muestra el tiempo de computación  
elapsed_time = cputime-start_time  
  
% se calcula el valor del PSNR para la imagen marcada  
psnr = psnr(cover_object,watermarked_image dbl,Mc,Nc)  
  
% se muestra la imagen marcada por pantalla  
% figure  
% imshow(watermarked_image_int,[])  
% title('Imagen Marcada')
```

Extracción (scdct_recuperar_sin_permutacion.m)

```
%-----%  
%| Nombre proyecto: Marcas de agua en imágenes digitales |%  
%| Autor: Raúl Péruela Martínez |%  
%| Año: 2009 |%  
%-----%  
%| Nombre código: scdct_recuperar_sin_permutacion.m |%  
%| Función: extracción de la marca de agua por el método |%  
%| de las secuencias caóticas y la DCT, sin |%  
%| permutación. |%  
%-----%  
  
% se limpia todo  
clear all;  
  
% se cargan los datos necesarios para la extracción  
cd 'Datos'  
load 'datosSCDCT_noper.mat'  
cd ..  
  
% se guarda el tiempo de comienzo de computación  
start_time = cputime;  
  
% se lee la imagen marcada y se le cambia el tipo a double  
file_name = 'scdct_noper_watermarked.bmp';  
watermarked_image = double(imread(file_name));  
  
% se determinan las dimensiones de la imagen marcada  
[Mw,Nw] = size(watermarked_image);  
  
% se genera la marca con números pseudoaleatorios de una N(0,1)  
randn('state',key);  
orig_watermark = randn(tam,1);  
  
% se calcula la DCT  
trans = dct2(watermarked_image);
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se extraen los coeficientes en zig-zag, se extraen los
coefcientes DCT
% que se usaron para ocultar la marca
watermark = zeros(tam,1);
k = 1;
for i = 2:Mw
    for j = 1:i
        if(k <= tam)
            watermark(k) = trans(i,j);
            k = k+1;
        end
    end
end

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula la similaridad
graf = correlacion(orig_watermark,watermark,tam,1);
title('Correlacion');

% se guarda la gráfica en un fichero
cd 'ImagenesPruebas'
I = getframe(gcf);
imwrite(I.cdata,'scdct_noper_correlacion.bmp');
cd ..
```

B.1.9. Algoritmo basado en PCA

B.1.9.1. Primer Método: PCA

Inserción (pca_insertar.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: PCA_insertar.m |%
%| Función: inserción de una marca de agua por el método |%
%| PCA. |%
%-----%

% se limpia todo
clear all;

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se inicializan los parámetros necesarios
p = 64; % número de componentes principales
alpha = 1;
tam = 1024*16; % longitud de la marca
key = 250;
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se guardan las variables que harán falta para la extracción
cd 'Datos'
save 'datosPCA.mat' p alpha tam key;
cd ..

% se lee la imagen de cobertura
file_name = 'lena_256.bmp';
cover_object = double(imread(file_name));

% se determinan las dimensiones de la imagen de cobertura
[Mc,Nc] = size(cover_object);

% la marca sera una secuencia pseudoaleatoria de números
procedentes de una
% distibucion nomal N(0,1)
randn('state',key);
marca = randn(tam,1);

% se divide la imagen en bloques 8x8, se convierten en vectores 1D
X = zeros(64,1024);
k = 1;
for i = 1:8:Mc
    for j = 1:8:Nc
        A = cover_object(i:i+7,j:j+7);
        A = A(:);
        X(:,k) = A;
        k = k+1;
    end
end

% se calcula la media de las filas
media = mean(X,2);

% a cada columna se le resta la media
X2 = X-repmat(media,[1 1024]);

% se calcula la matriz de proyección: U
[U,S,V] = svds((1/sqrt(1024.))*X2,p);

% se realiza la proyección
proy = U'*X2;

% se oculta 16 elementos de la marca en los ultimos 16 elementos
k = 1;
for i = 1:1024
    num = marca(k:k+15);
    proy(49:64,i) = proy(49:64,i)+alpha*proy(49:64,i).*num;
    k = k+16;
end

% se hace la reconstrucción
X3 = U*proy;
X3 = X3+repmat(media,[1 1024]);

% se crea la imagen marcada
imagenmarcada = zeros(Mc,Nc);
k = 1;
for i = 1:8:Mc
```

APÉNDICE B: MANUAL DE CÓDIGO

```
for j = 1:8:Nc
    columna = X3(:,k);
    watermarked_image(i:i+7,j:j+7) = reshape(columna,[8,8]);
    k = k+1;
end
end

% se convierte la imagen marcada a tipo uint8
watermarked_image_int = uint8(watermarked_image);

% se guarda la imagen marcada en un fichero
cd 'ImagenesPruebas'
imwrite(watermarked_image_int,'pca_watermarked.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el valor del PSNR para la imagen marcada
psnr = psnr(cover_object,watermarked_image,Mc,Nc)

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermarked_image_int,[])
% title('Imagen Marcada')
```

Extracción (pca_recuperar.m)

```
%-----
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----
%| Nombre código: PCA_recuperar.m |%
%| Función: extracción de la marca de agua por el método |%
%| PCA. |%
%-----

% se limpia todo
clear all;

% se cargan los datos necesarios para la extracción
cd 'Datos'
load 'datosPCA.mat'
cd ..

% se guarda el tiempo de comienzo de computación
start_time = cputime;

% se lee la imagen marcada y se le cambia el tipo a double
file_name = 'pca_watermarked.bmp';
watermarked_image = double(imread(file_name));

% carga de las imágenes atacadas
% file_name = 'pca_JPEGattack.jpg';
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% file_name = 'pca_NOISEattack.bmp';
% file_name = 'pca_MEANattack.bmp';
% file_name = 'pca_CROPPINGattack.bmp';
% file_name = 'pca_SCALEDattack.bmp';
% file_name = 'pca_ROTATEattack.bmp';
% cd 'ImagenesAtaques'
% watermarked_image = double(imread(file_name));
% cd ..

% se determinan las dimensiones de la imagen marcada
[Mw,Nw] = size(watermarked_image);

% se lee la imagen de cobertura
file_name = 'lena_256.bmp';
cover_object = double(imread(file_name));

% se obtiene la marca original
randn('state',key);
orig_watermark = randn(tam,1);

% se averigua la matriz de proyeccion a partir de la matriz
% original
% se divide la imagen en bloques 8x8, se convierten en vectores 1D
X = zeros(64,1024);
k = 1;
for i = 1:8:Mw
    for j = 1:8:Nw
        A = cover_object(i:i+7,j:j+7);
        A = A(:);
        X(:,k) = A;
        k = k+1;
    end
end

% se calcula la media de las filas
media = mean(X,2);

% a cada fila se le resta su media
X2 = X-repmat(media,[1 1024]);

% se calcula la matriz de proyección: U
[U,S,V] = svds((1/sqrt(1024.))*X2,p);

% se proyecta la imagen original
proy = U'*X2;

% se extraen las 16 ultimas coordenadas de cada bloque de la imagen
% original para formar el vector y(i)
y = zeros(tam,1);
k = 1;
for i = 1:1024
    y(k:k+15,1) = proy(49:64,i);
    k = k+16;
end

% se proyecta la imagen marcada con la matriz de proyección de la
% imagen
% original
```

APÉNDICE B: MANUAL DE CÓDIGO

```
X = zeros(64,1024);
k = 1;
for i = 1:8:Mw
    for j = 1:8:Nw
        A = watermarked_image(i:i+7,j:j+7);
        A = A(:);
        X(:,k) = A;
        k = k+1;
    end
end
X2 = X-repmat(media,[1 1024]);
proymarcada = U'*X2;

% se extraen las 16 ultimas coordenadas proyectadas
ypri = zeros(tam,1);
k = 1;
for i = 1:1024
    ypri(k:k+15,1) = proymarcada(49:64,i);
    k = k+16;
end
unos = ones(tam,1);
watermark = (ypri./y-unos)/alpha;

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula la similaridad
graf = correlacion(orig_watermark,watermark,tam,1);
title('Correlacion');

% se guarda la gráfica en un fichero
cd 'ImagenesPruebas'
I = getframe(gcf);
imwrite(I.cdata,'pca_correlacion.bmp');
cd ..
```

B.1.9.2. Segundo Método: PCA para construir la imagen de referencia

Inserción (*pcaReferencia_insertar.m*)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: PCAReferencia_insertar.m |%
%| Función: inserción de una marca de agua por el método |%
%|           PCA para construir la imagen de referencia. |%
%-----%

% se limpia todo
clear all;

% se guarda el tiempo de comienzo de computación
start_time = cputime;
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se inicializan los parámetros necesarios
b = 10;
s = 1;
t = 3;
tam = 1000;
alpha = (s+t)/2;
key = 250.;

% se lee la imagen de cobertura
file_name = 'lena_256.bmp';
cover_object = double(imread(file_name));

% se determinan las dimensiones de la imagen de cobertura
[Mc,Nc] = size(cover_object);

% se calcula de la imagen de referencia usando PCA con la imagen completa
% si b = 256, la imagen obtenida en recons2 es la original
media = mean(cover_object,2);
X = cover_object-repmat(media,[1 Nc]);
H = (1/sqrt(Nc))*X;
[U,S,V] = svds(H,b);
proy = (U')*X;
recons = U*proy+repmat(media,[1 Nc]);

% se calcula cuantos elementos de la imagen sirven
referencia = recons;
contador = 0;
idx = [];
for i = 1:Mc
    for j = 1:Nc
        if((abs(cover_object(i,j))-referencia(i,j)) > s) &&
        (abs(cover_object(i,j))-referencia(i,j)) < t)
            idx = [idx;[i j]];
            contador = contador+1;
        end
    end
end

% se guardan las variables que harán falta para la extracción
cd 'Datos'
save datosPCARefencia.mat b s t tam alpha idx key;
cd ..

% se crea la marca, siendo esta 'tam' bits pseudoaleatorios
rand('state',key);
marca = ceil(2*rand(tam,1)-1);

% se realiza una permutación aleatoria para escoger aquellos
% coeficientes donde
% se va a ocultar la marca
permuta1 = randperm(contador);
permuta = permuta1(1:tam);

% se crea la imagen marcada
watermarked_image = cover_object;
for k = 1:tam
```

APÉNDICE B: MANUAL DE CÓDIGO

```
pos = permuta(k);
vector = idx(pos,:);
i = vector(1);
j = vector(2);
bit = marca(k);
if((cover_object(i,j) <= referencia(i,j)) && (bit == 1))
    watermarked_image(i,j) = referencia(i,j)+alpha;
end
if((cover_object(i,j) > referencia(i,j)) && (bit == 0))
    watermarked_image(i,j) = referencia(i,j)-alpha;
end
end

% se convierte la imagen marcada a tipo uint8
watermarked_image_int = uint8(watermarked_image);

% se guarda la imagen marcada en un fichero
cd 'ImagenesPruebas'
imwrite(watermarked_image_int,'pcaReferencia_watermarked.bmp');
cd ..

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el valor del PSNR para la imagen marcada
psnr = psnr(cover_object,watermarked_image,Mc,Nc)

% se muestra la imagen marcada por pantalla
% figure
% imshow(watermarked_image_int,[])
% title('Imagen Marcada')
```

Extracción (pcaReferencia_recuperar.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: PCAReferencia_recuperar.m |%
%| Función: extracción de la marca de agua por el método |%
%|           PCA para construir la imagen de referencia. |%
%-----%

% se limpia todo
clear all;

% se cargan los datos necesarios para la extracción
cd 'Datos'
load 'datosPCAReferencia.mat'
cd ..

% se guarda el tiempo de comienzo de computación
start_time = cputime;
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% se lee la imagen marcada y se le cambia el tipo a double
file_name = 'pcaReferencia_watermarked.bmp';
watermarked_image = double(imread(file_name));

% carga de las imágenes atacadas
% file_name = 'pcaReferencia_JPEGattack.jpg';
% file_name = 'pcaReferencia_NOISEattack.bmp';
% file_name = 'pcaReferencia_MEANattack.bmp';
% file_name = 'pcaReferencia_CROPPINGattack.bmp';
% file_name = 'pcaReferencia_SCALEdattack.bmp';
% file_name = 'pcaReferencia_ROTATEattack.bmp';
% cd 'ImagenesAtaques'
% watermarked_image = double(imread(file_name));
% cd ..

% se determinan las dimensiones de la imagen marcada
[Mw,Nw] = size(watermarked_image);

% se calcula la imagen de referencia usando PCA con la imagen
% marcada
media = mean(watermarked_image,2);
X = watermarked_image-repmat(media,[1 Nw]);
H = (1/sqrt(Nw))*X;
[U,S,V] = svds(H,b);
proy = (U')*X;
recons = U*proy+repmat(media,[1 Nw]);
referencia = recons;

% se crea la marca que son 'tam' bits pseudoaleatorios
rand('state',key);
orig_watermark = ceil(2*rand(tam,1)-1);

% se hace una permutación aleatoria para escoger aquellos
% coeficientes donde
% se va a ocultar la marca
contador = size(idx,1);
permutal = randperm(contador);
permuta = permutal(1:tam);

% se extrae la marca
watermark = zeros(tam,1);
for k = 1:tam
    pos = permuta(k);
    vector = idx(pos,:);
    i = vector(1);
    j = vector(2);
    if(watermarked_image(i,j) > referencia(i,j))
        watermark(k) = 1;
    end
end

% se muestra el tiempo de computación
elapsed_time = cputime-start_time

% se calcula el número de errores que contiene la marca extraída
errVector(orig_watermark,watermark,tam);

% se calcula la similaridad
```

APÉNDICE B: MANUAL DE CÓDIGO

```
graf = similaridad(orig_watermark,watermark,tam);
title('Similaridad');

% se guarda la gráfica en un fichero
cd 'ImagenesPruebas'
I = getframe(gcf);
imwrite(I.cdata,'pcaReferencia_similaridad.bmp');
cd ..
```

B.2. Ataques

B.2.1. Compresión JPEG (JPEG.m)

```
%-----
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Pérula Martínez |%
%| Año: 2009 |%
%-----
%| Nombre código: JPEG.m |%
%| Función: guarda una imagen con compresión jpeg. |%
%| Parámetros entrada: |%
%|   - I: imagen a tratar. |%
%|   - file_name: nombre del archivo a guardar. |%
%|   - quality: calidad de la compresión. |%
%| Parámetros salida: |%
%|   - No tiene. |%
%-----

function [] = JPEG(I,file_name,quality)
cd 'ImagenesAtaques'
if nargin == 2
    % la compresión se realiza con 75 de calidad
    imwrite(I,file_name);
else
    % se especifica la calidad que se desea
    imwrite(I,file_name,'Quality',quality);
end
cd ..
return
```

B.2.2. Inserción de ruido Gaussiano (noise.m)

```
%-----
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Pérula Martínez |%
%| Año: 2009 |%
%-----
%| Nombre código: noise.m |%
%| Función: añade ruido gausiano a una imagen. |%
%| Parámetros entrada: |%
%|   - I: imagen a tratar. |%
```

APÉNDICE B: MANUAL DE CÓDIGO

```
%|      - M: media especificada. |%
%|      - V: varianza especificada. |%
%| Parámetros salida: |%
%|      - I: imagen con el ruido gaussiano añadido. |%
%-----|%

function [] = noise(I,file_name,M,V)
if(nargin == 2)
    % se añade ruido gaussiano con media cero y varianza 0.01
    I = imnoise(I,'gaussian');
    cd 'ImagenesAtaques'
    imwrite(I,file_name);
    cd ..
else
    % se añade ruido gaussiano con media y varianza especificados
    I = imnoise(I,'gaussian',M,V);
end
return
```

B.2.3. Aplicación de un filtro de paso bajo basado en la media (meanFilter.m)

```
%-----
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----

%| Nombre código: meanFilter.m |%
%| Función: aplica un filtro de paso bajo a una imagen. |%
%| Parámetros entrada: |%
%|      - I: imagen a tratar. |%
%| Parámetros salida: |%
%|      - I: imagen con el filtro de paso bajo aplicado. |%
%-----|%

function [] = meanFilter(I,file_name)
    % se crea el filtro de la media
    h = fspecial('average',8);
    % se aplica el filtro
    I = imfilter(I,h);
    cd 'ImagenesAtaques'
    imwrite(I,file_name);
    cd ..
return
```

B.2.4. Recortado de la imagen (cropping.m)

```
%-----
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----|%
```

APÉNDICE B: MANUAL DE CÓDIGO

```
%| Nombre código: cropping.m |%
%| Función: recortado de una imagen. |%
%| Parámetros entrada: |%
%|   - I: imagen a tratar. |%
%|   - XMIN: coordenada x. |%
%|   - YMIN: coordenada y. |%
%|   - WIDTH: ancho que se desea. |%
%|   - HEIGHT: alto que se desea. |%
%| Parámetros salida: |%
%|   - I: imagen con el recortado realizado. |%
%-----%
```

```
function [] = cropping(I,XMIN,YMIN,WIDTH,HEIGHT,file_name)
I = imcrop(I, [XMIN YMIN WIDTH HEIGHT]);
cd 'ImagenesAtaques'
imwrite(I,file_name);
cd ..
return
```

B.2.5. Escalado de la imagen (scaled.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: scaled.m |%
%| Función: escalado de una imagen. |%
%| Parámetros entrada: |%
%|   - I: imagen a tratar. |%
%|   - scale: tamaño de escalado. |%
%| Parámetros salida: |%
%|   - I: imagen con el escalado realizado. |%
%-----%
```

```
function [] = scaled(I,scale,file_name)
I = imresize(I, scale);
cd 'ImagenesAtaques'
imwrite(I,file_name);
cd ..
return
```

B.2.6. Rotación de la imagen (rotate.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: rotate.m |%
%| Función: rotación de una imagen en el plano. |%
%| Parámetros entrada: |%
```

APÉNDICE B: MANUAL DE CÓDIGO

```
%|      - I: imagen a tratar. |%
%|      - angle: ángulo en grados. |%
%| Parámetros salida: |%
%|      - I: imagen rotada. |%
%-----%  
  
function [] = rotate(I,angle,file_name)
    I = imrotate(I,angle,'crop');
    cd 'ImagenesAtaques'
    imwrite(I,file_name);
    cd ..
return
```

B.2.7. Simulación de los ataques (simAtaques.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: simAtaques.m |%
%| Función: realiza la simulación de todos los ataques |%
%| posibles a todos los algoritmos estudiados. |%
%-----%  
  
% Ataques sobre algoritmo de técnicas basadas en correlación
cd 'ImagenesPruebas'
I = imread('cor_watermarked.bmp');
cd ..  
  
JPEG(I,'cor_JPEGattack.jpg',60);
noise(I,'cor_NOISEattack.bmp');
meanFilter(I,'cor_MEANattack.bmp');
cropping(I,75,68,300,300,'cor_CROPPINGattack.bmp');
scaled(I,2,'cor_SCALEDattack.bmp');
rotate(I,90,'cor_ROTATEattack.bmp');  
  
% Ataques sobre algoritmo de Cox
cd 'ImagenesPruebas'
I = imread('cox_watermarked.bmp');
cd ..  
  
JPEG(I,'cox_JPEGattack.jpg',60);
noise(I,'cox_NOISEattack.bmp');
meanFilter(I,'cox_MEANattack.bmp');
cropping(I,75,68,300,300,'cox_CROPPINGattack.bmp');
scaled(I,2,'cox_SCALEDattack.bmp');
rotate(I,90,'cox_ROTATEattack.bmp');  
  
% Ataques sobre algoritmo DCT
cd 'ImagenesPruebas'
I = imread('dct_watermarked.bmp');
cd ..  
  
JPEG(I,'dct_JPEGattack.jpg',60);
```

APÉNDICE B: MANUAL DE CÓDIGO

```
noise(I,'dct_NOISEattack.bmp');
meanFilter(I,'dct_MEANattack.bmp');
cropping(I,75,68,300,300,'dct_CROPPINGattack.bmp');
scaled(I,2,'dct_SCALEDattack.bmp');
rotate(I,90,'dct_ROTATEattack.bmp');

% Ataques sobre algoritmo DCTCorrelacion
cd 'ImagenesPruebas'
I = imread('dctcor_watermarked.bmp');
cd ..

JPEG(I,'dctcor_JPEGattack.jpg',60);
noise(I,'dctcor_NOISEattack.bmp');
meanFilter(I,'dctcor_MEANattack.bmp');
cropping(I,75,68,300,300,'dctcor_CROPPINGattack.bmp');
scaled(I,2,'dctcor_SCALEDattack.bmp');
rotate(I,90,'dctcor_ROTATEattack.bmp');

% Ataques sobre algoritmo CDMA
cd 'ImagenesPruebas'
I = imread('cdma_watermarked.bmp');
cd ..

JPEG(I,'cdma_JPEGattack.jpg',60);
noise(I,'cdma_NOISEattack.bmp');
meanFilter(I,'cdma_MEANattack.bmp');
cropping(I,75,68,300,300,'cdma_CROPPINGattack.bmp');
scaled(I,2,'cdma_SCALEDattack.bmp');
rotate(I,90,'cdma_ROTATEattack.bmp');

% Ataques sobre algoritmo DWT
cd 'ImagenesPruebas'
I = imread('dwt_watermarked.bmp');
cd ..

JPEG(I,'dwt_JPEGattack.jpg',60);
noise(I,'dwt_NOISEattack.bmp');
meanFilter(I,'dwt_MEANattack.bmp');
cropping(I,75,68,300,300,'dwt_CROPPINGattack.bmp');
scaled(I,2,'dwt_SCALEDattack.bmp');
rotate(I,90,'dwt_ROTATEattack.bmp');

% Ataques sobre algoritmo DWTHaar
cd 'ImagenesPruebas'
I = imread('dwtHaar_watermarked.bmp');
cd ..

JPEG(I,'dwtHaar_JPEGattack.jpg',60);
noise(I,'dwtHaar_NOISEattack.bmp');
meanFilter(I,'dwtHaar_MEANattack.bmp');
cropping(I,75,68,300,300,'dwtHaar_CROPPINGattack.bmp');
scaled(I,2,'dwtHaar_SCALEDattack.bmp');
rotate(I,90,'dwtHaar_ROTATEattack.bmp');

% Ataques sobre algoritmo DWTParidad
cd 'ImagenesPruebas'
I = imread('dwtParidad_watermarkedMod.bmp');
cd ..
```

APÉNDICE B: MANUAL DE CÓDIGO

```
JPEG(I,'dwtParidad_JPEGattack.jpg',60);
noise(I,'dwtParidad_NOISEattack.bmp');
meanFilter(I,'dwtParidad_MEANattack.bmp');
cropping(I,75,68,300,300,'dwtParidad_CROPPINGattack.bmp');
scaled(I,2,'dwtParidad_SCALEDattack.bmp');
rotate(I,90,'dwtParidad_ROTATEattack.bmp');

% Ataques sobre algoritmo SVD
cd 'ImagenesPruebas'
I = imread('svd_watermarked.bmp');
cd ..

JPEG(I,'svd_JPEGattack.jpg',60);
noise(I,'svd_NOISEattack.bmp');
meanFilter(I,'svd_MEANattack.bmp');
cropping(I,75,68,300,300,'svd_CROPPINGattack.bmp');
scaled(I,2,'svd_SCALEDattack.bmp');
rotate(I,90,'svd_ROTATEattack.bmp');

% Ataques sobre algoritmo SVDArnold
cd 'ImagenesPruebas'
I = imread('svdArnold_watermarked.bmp');
cd ..

JPEG(I,'svdArnold_JPEGattack.jpg',60);
noise(I,'svdArnold_NOISEattack.bmp');
meanFilter(I,'svdArnold_MEANattack.bmp');
cropping(I,75,68,300,300,'svdArnold_CROPPINGattack.bmp');
scaled(I,2,'svdArnold_SCALEDattack.bmp');
rotate(I,90,'svdArnold_ROTATEattack.bmp');

% Ataques sobre algoritmo SVDIntercambio
cd 'ImagenesPruebas'
I = imread('svdIntercambio_watermarked.bmp');
cd ..

JPEG(I,'svdIntercambio_JPEGattack.jpg',60);
noise(I,'svdIntercambio_NOISEattack.bmp');
meanFilter(I,'svdIntercambio_MEANattack.bmp');
cropping(I,75,68,300,300,'svdIntercambio_CROPPINGattack.bmp');
scaled(I,2,'svdIntercambio_SCALEDattack.bmp');
rotate(I,90,'svdIntercambio_ROTATEattack.bmp');

% Ataques sobre algoritmo SVDOrden
cd 'ImagenesPruebas'
I = imread('svdOrden_watermarked.bmp');
cd ..

JPEG(I,'svdOrden_JPEGattack.jpg',60);
noise(I,'svdOrden_NOISEattack.bmp');
meanFilter(I,'svdOrden_MEANattack.bmp');
cropping(I,75,68,300,300,'svdOrden_CROPPINGattack.bmp');
scaled(I,2,'svdOrden_SCALEDattack.bmp');
rotate(I,90,'svdOrden_ROTATEattack.bmp');

% Ataques sobre algoritmo SVDProximidad
cd 'ImagenesPruebas'
```

APÉNDICE B: MANUAL DE CÓDIGO

```
I = imread('svdProximidad_watermarked.bmp');
cd ..

JPEG(I,'svdProximidad_JPEGatack.jpg',60);
noise(I,'svdProximidad_NOISEatack.bmp');
meanFilter(I,'svdProximidad_MEANatack.bmp');
cropping(I,75,68,300,300,'svdProximidad_CROPPINGatack.bmp');
scaled(I,2,'svdProximidad_SCALEDatack.bmp');
rotate(I,90,'svdProximidad_ROTATEatack.bmp');

% Ataques sobre algoritmo LSB
cd 'ImagenesPruebas'
I = imread('lsb_watermarked.bmp');
cd ..

JPEG(I,'lsb_JPEGatack.jpg',60);
noise(I,'lsb_NOISEatack.bmp');
meanFilter(I,'lsb_MEANatack.bmp');
cropping(I,75,68,300,300,'lsb_CROPPINGatack.bmp');
scaled(I,2,'lsb_SCALEDatack.bmp');
rotate(I,90,'lsb_ROTATEatack.bmp');

% Ataques sobre algoritmo SC
cd 'ImagenesPruebas'
I = imread('sc_watermarked.bmp');
cd ..

JPEG(I,'sc_JPEGatack.jpg',60);
noise(I,'sc_NOISEatack.bmp');
meanFilter(I,'sc_MEANatack.bmp');
cropping(I,75,68,300,300,'sc_CROPPINGatack.bmp');
scaled(I,2,'sc_SCALEDatack.bmp');
rotate(I,90,'sc_ROTATEatack.bmp');

% Ataques sobre algoritmo SCDCT
cd 'ImagenesPruebas'
I = imread('scdct_watermarked.bmp');
cd ..

JPEG(I,'scdct_JPEGatack.jpg',60);
noise(I,'scdct_NOISEatack.bmp');
meanFilter(I,'scdct_MEANatack.bmp');
cropping(I,75,68,300,300,'scdct_CROPPINGatack.bmp');
scaled(I,2,'scdct_SCALEDatack.bmp');
rotate(I,90,'scdct_ROTATEatack.bmp');

% Ataques sobre algoritmo PCA
cd 'ImagenesPruebas'
I = imread('pca_watermarked.bmp');
cd ..

JPEG(I,'pca_JPEGatack.jpg',60);
noise(I,'pca_NOISEatack.bmp');
meanFilter(I,'pca_MEANatack.bmp');
cropping(I,75,68,300,300,'pca_CROPPINGatack.bmp');
scaled(I,2,'pca_SCALEDatack.bmp');
rotate(I,90,'pca_ROTATEatack.bmp');
```

APÉNDICE B: MANUAL DE CÓDIGO

```
% Ataques sobre algoritmo PCAReferencia
cd 'ImagenesPruebas'
I = imread('pcaReferencia_watermarked.bmp');
cd ..

JPEG(I,'pcaReferencia_JPEGattack.jpg',60);
noise(I,'pcaReferencia_NOISEattack.bmp');
meanFilter(I,'pcaReferencia_MEANattack.bmp');
cropping(I,75,68,300,300,'pcaReferencia_CROPPINGattack.bmp');
scaled(I,2,'pcaReferencia_SCALEDattack.bmp');
rotate(I,90,'pcaReferencia_ROTATEattack.bmp');
```

B.3. Funciones Auxiliares

B.3.1. Correlación (correlacion.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales |%
%| Autor: Raúl Péruela Martínez |%
%| Año: 2009 |%
%-----%
%| Nombre código: correlacion.m |%
%| Función: calcula la correlación entre dos marcas de |%
%| agua. |%
%| Parámetros entrada: |%
%| - orig_watermark: marca original. |%
%| - watermark: marca extraída. |%
%| - tam: número de filas de la marca. |%
%| - tam1: número de columnas de la marca. |%
%| Parámetros salida: |%
%| - grafico: grafica de correlaciones. |%
%-----%

function grafico = correlacion(orig_watermark,watermark,tam,tam1)
% se inicializa a ceros el vector
correlacion = zeros(1,500);

if nargin == 3
    % se crean las marcas aleatorias
    for i = 1:500
        randn('state',double(i));
        ale_watermark = randn(tam,tam);
        correlacion(i) = corr2(watermark,ale_watermark);
    end
else
    % se crean las marcas aleatorias
    for i = 1:500
        randn('state',double(i));
        ale_watermark = randn(tam,tam1);
        correlacion(i) = corr2(watermark,ale_watermark);
    end
end

% se pone el resultado con la marca original y la extraida
```

APÉNDICE B: MANUAL DE CÓDIGO

```
correlacion(250) = corr2(orig_watermark,watermark);  
  
% se crea el gráfico con los 500 resultados  
x = 1:500;  
grafico = plot(x,correlacion);  
end
```

B.3.2. Similaridad (similaridad.m)

```
%-----%  
%| Nombre proyecto: Marcas de agua en imágenes digitales |%  
%| Autor: Raúl Péruela Martínez |%  
%| Año: 2009 |%  
%-----%  
%| Nombre código: similaridad.m |%  
%| Función: calcula la similaridad entre dos marcas de |%  
%| agua. |%  
%| Parámetros entrada:  
%| - orig_watermark: marca original.  
%| - watermark: marca extraída.  
%| - tam: número de filas de la marca.  
%| - tam1: número de columnas de la marca.  
%| Parámetros salida:  
%| - grafico: grafica de similaridades. |%  
%-----%  
  
function grafico = similaridad(orig_watermark,watermark,tam,tam1)  
if (size(orig_watermark) ~= size(watermark))  
    watermark = reshape(watermark,size(orig_watermark));  
end  
  
% se inicializa a ceros el vector  
simi = zeros(1,500);  
  
% según el número de parámetros de entrada se crea unas marcas u  
otras  
if nargin == 3  
    % se crean las marcas aleatorias  
    for i = 1:500  
        rand('seed',double(i));  
        ale_watermark = ceil(2*rand(tam,1)-1);  
        simi(i) =  
(ale_watermark'*orig_watermark)/sqrt(ale_watermark'*ale_watermark);  
    end  
else  
    % se crean las marcas aleatorias  
    for i = 1:500  
        rand('seed',double(i));  
        ale_watermark = ceil(2*rand(tam*tam1,1)-1);  
        simi(i) =  
(ale_watermark'*orig_watermark)/sqrt(ale_watermark'*ale_watermark);  
    end  
end  
  
% se pone el resultado con la marca original y la extraída
```

APÉNDICE B: MANUAL DE CÓDIGO

```
simi(250) =  
(watermark'*orig_watermark)/sqrt(watermark'*watermark);  
  
% se crea el gráfico con los 500 resultados  
x = 1:500;  
grafico = plot(x,simi);  
end
```

B.3.3. Cálculo de los errores

B.3.3.1. En una matriz (errMatriz.m)

```
%-----%  
%| Nombre proyecto: Marcas de agua en imágenes digitales |%  
%| Autor: Raúl Péruela Martínez |%  
%| Año: 2009 |%  
%-----%  
%| Nombre código: errMatriz.m |%  
%| Función: comprueba cuantos bits son erróneos entre la |%  
%| marca original y la extraida. |%  
%| Parámetros entrada:  
%| - orig_watermark: marca original. |%  
%| - watermark: marca extraída. |%  
%| - Mw: número de filas de la marca. |%  
%| - Nw: número de columnas de la marca. |%  
%| Parámetros salida:  
%| - Ninguno. |%  
%-----%  
  
function errMatriz(orig_watermark,watermark,Mw,Nw)  
malos = 0;  
  
for i = 1:Mw  
    for j = 1:Nw  
        if(orig_watermark(i,j) ~= watermark(i,j))  
            malos = malos+1;  
        end  
    end  
end  
  
fprintf('Bits mal recuperados de la marca: %d de  
%d\n',malos,Mw*Nw);  
end
```

B.3.3.2. En un vector (errVector.m)

```
%-----%  
%| Nombre proyecto: Marcas de agua en imágenes digitales |%  
%| Autor: Raúl Péruela Martínez |%  
%| Año: 2009 |%  
%-----%  
%| Nombre código: errVector.m |%
```

APÉNDICE B: MANUAL DE CÓDIGO

```
%| Función: comprueba cuantos bits son erróneos entre la marca original y la extraida.
%| Parámetros entrada:
%|   - orig_watermark: marca original.
%|   - watermark: marca extraída.
%|   - tam: dimensión de la marca.
%| Parámetros salida:
%|   - Ninguno.

function errVector(orig_watermark,watermark,tam)
malos = 0;

for i = 1:tam
    if(orig_watermark(i) ~= watermark(i))
        malos = malos+1;
    end
end

fprintf('Bits mal recuperados de la marca: %d de %d\n',malos,tam);
end
```

B.3.4. PSNR (psnr.m)

```
%-----%
%| Nombre proyecto: Marcas de agua en imágenes digitales %
%| Autor: Raúl Péruela Martínez %
%| Año: 2009 %
%-----%
%| Nombre código: psnr.m %
%| Función: calcula el psnr de una imagen. %
%| Parámetros entrada:
%|   - image: imagen original. %
%|   - image_prime: imagen marcada. %
%|   - M: ancho de la imagen. %
%|   - N: alto de la imagen. %
%| Parámetros salida:
%|   - A: valor del psnr en decibelios. %

function [A] = psnr(image,image_prime,M,N)
% convierte las imágenes a tipo double para poder tratarlas
image = double(image);
image_prime = double(image_prime);

% comprobacion de que no se divida por 0
if((sum(sum(image-image_prime))) == 0)
    error('Las imágenes de entrada no deben de ser identicas')
else
    psnr_num = M*N*max(max(image.^2)); % calcula el numerador
    psnr_den = sum(sum((image-image_prime).^2)); % calcula el denominador
```

APÉNDICE B: MANUAL DE CÓDIGO

```
A = psnr_num/psnr_den; % calcula el  
PSNR  
end  
  
A = 10*log10(A); % convierte el PSNR en decibelios  
return
```