Proyecto de Final de Carrera

Desarrollo de nuevas técnicas de watermarking robusto y watermarking frágil

Miguel Colom Barco

Dirigido por
José Luis Lisani Roca

"Las cosas no son lo que parecen . . . ,
son como tú las quieras ver".

Índice

Capítulo 1

Datos del Proyecto de Final de Carrera

- **Título del proyecto** : Desarrollo de nuevas técnicas de watermarking robusto y watermarking frágil.
- **Autor** : Miguel Colom Barco.
- **Dirección** : José Luis Lisani Roca.
- **Estudios** : Ingeniería Técnica de Telecomunicaciones, especialidad Telemática.
- **Universitat de les Illes Balears.**

Capítulo 2

Objetivo del proyecto y estructura de la memoria

Los objetivos del presente proyecto son los siguientes :

- Hacer una revisión de algunas de las técnicas de watermarking más significativas que han sido desarrolladas con anterioridad, tanto de watermarking robusto, como de watermarking frágil.
- Estudiar con detalle una de las técnicas de watermarking robusto, implementarla y comprobar su funcionamiento mediante una serie de pruebas e introducir algunas mejoras en el método.
- Desarrollar una nueva técnica de watermarking frágil.

Para ello, la memoria del informe se ha dividido en varias partes. En la primera de ellas, titulada *"Introducción al watermarking digital"* (capítulo 3), se hace una introducción al tema, partiendo del desarrollo de las tecnologías de transmisión y almacenamiento digital, hasta llegar a la conclusión de que, en ciertos ámbitos, es necesaria la capacidad de poder asociar un determinado contenido con su autor legítimo, o asegurar que otro contenido multimedia (una imagen, sonido, vídeo, etc.) no ha sido manipulado intencionalmente. Se dan ejemplos de situaciones en las que se hace necesaria la aplicación de estas técnicas.

También en esta parte introductoria se define qué son en concreto las técnicas de "watermarking", cuáles son las características que las definen, qué tipos de métodos existen, cómo los podemos clasificar, y qué diferencias hay entre este tipo de técnicas y otras que podrían parecer equiparables como, por ejemplo, las de esteganografía.

En el capítulo 4, titulado *"Watermarking robusto"*, se profundiza en la descripción de qué es el watermarking, aunque referido en concreto a un tipo denominado "robusto".

Se explica en qué consisten este tipo de técnicas, por qué son utilizadas, y además se hace una revisión de algunas de las técnicas de watermarking robusto que han sido desarrolladas con anterioridad, para poder tener una visión global del estado en el que se encuentra la investigación actual, qué se ha conseguido desarrollar hasta el momento y a dónde parece dirigirse la investigación en nuevas técnicas.

En la sección 4.2, *"Estudio e implementación de un algoritmo de watermarking robusto"*, se analiza en profundidad uno de los métodos de watermarking robusto existentes en la literatura [1]. Se explica en qué consiste, se demuestra matemáticamente su viabilidad, se hacen una serie de pruebas para comprobar que realmente funciona tal y como afirman sus autores, y finalmente se describen algunas limitaciones observadas en el método, en la sección *"Limitaciones"* (4.2.19).

Dado que uno de los objetivos de este proyecto es desarrollar nuevas técnicas de watermarking, partiendo del análisis del método anterior se desarrolla una nueva técnica que mejora significativamente las prestaciones del método original.

El método original presenta algunos inconvenientes que lo hacen poco útil en aplicaciones prácticas, ya que algunos ataques consiguen destruir el watermark sin afectar la calidad de la imagen.

El método mejorado que se desarrolla a lo largo de este proyecto consigue que sea resistente a la práctica totalidad de los ataques conocidos, por lo que incluso podría ser aplicado de forma comercial. Este y otros aspectos aparecen en la sección de *"Conclusiones"* (4.4) de la parte de watermarking robusto.

Respecto a las técnicas de watermarking frágil, en el capítulo 5, *"Watermarking frágil"*, se realiza una revisión análoga a la que se hace con las de watermarking robusto, es decir, se profundiza en la descripción y definición del watermarking frágil, se acota su ámbito de aplicación, y se hace una revisión de las técnicas más significativas que han sido desarrolladas con anterioridad.

Esta revisión sirve como introducción a la sección 5.0.2, *"Método desarrollado de watermarking frágil"*, en la que se presenta un nuevo método de watermarking frágil totalmente diferente a los desarrollados con anterioridad, basado en la teoría de conjuntos de nivel, en la transformada FLST (Fast Level Sets Transform), y la generación de ruido pseudo–aleatorio invertible.

Hasta el momento no existe ningún método de watermarking que esté basado en la reciente técnica del análisis de conjuntos de nivel, y la que se va a presentar en este proyecto será la primera.

Se explicará en qué consiste la técnica desarrollada, qué aporta frente a las ya existentes, y se efectuarán una serie de pruebas para demostrar la validez del método propuesto.

Finalmente, se presentarán las conclusiones para este método, y se sugerirán algunas de las posibles líneas de investigación futura (sección 5.0.11).

Capítulo 3

Introducción al watermarking digital

Durante estos últimos años hemos podido ver cómo se ha producido un espectacular desarrollo de la tecnología que permite la transmisión y almacenamiento digital de la información. El máximo representante de esta revolución tecnológica es, sin lugar a dudas, la red Internet, aunque las mejoras en la transmisión de datos de dispositivos de telefonía móvil y todo tipo redes inalámbricas están creando nuevas redes capaces de transportar contenidos multimedia.

Internet no sólo ha permitido interconectar millones de ordenadores entre sí mediante una red que prácticamente ha conquistado casi todo el planeta sino que, además, su velocidad de transmisión ha ido aumentando cada vez más, hasta el punto de que el contenido multimedia se ha convertido en un elemento habitual, algo que era impensable en sus primeros años de vida. Aunque si bien era difícil de imaginar un desarrollo tan importante de Internet en sus inicios, más aún lo era pensar que los teléfonos móviles actuales serían capaces de no sólo incorporar una cámara digital, sino también grabar imágenes, o vídeo en formato MPEG durante varios minutos, y transmitir esa información a otros terminales.

Gracias a Internet, usuarios geográficamente distantes pueden comunicarse e intercambiar información, y muchas personas han podido tener acceso a una fuente de conocimiento de tal potencia en cantidad de información y facilidad de acceso, que tiempo atrás hubiera parecido ciencia-ficción.

Teniendo en cuenta que se puede considerar que estamos en el comienzo de esta revolución tecnológica de las comunicaciones, es de esperar que se produzca una gran convergencia de las redes fijas con las inalámbricas, y que las velocidades de transmisión, así como las de almacenamiento de la información se vean muy mejoradas a corto plazo.

Esta transformación tecnológica y sobre todo social, ha traído consigo una serie de cuestiones que hasta ahora no se habían planteado, como son las relacionadas con la protección y autentificación de los contenidos que viajan por la Red. Internet, y los medios de transmisión digital en general, permiten la transmisión de la información a alta velocidad, a lugares muy distantes físicamente, y sin pérdida de calidad, lo cual implica que es posible copiar el contenido indefinidamente, generando múltiples copias perfectamente iguales las unas de las otras, gracias al formato digital. Es aquí donde se nos plantea el problema de la protección y la autentificación de los contenidos.

El autor de un determinado contenido, como por ejemplo una imagen, un vídeo digital, música sobre soporte electrónico, etc., tiene derecho a que su creación sea distribuida, sin que esto implique que deje de ser reconocido como el legítimo propietario.

Las técnicas de watermarking que se van a explicar a continuación son una buena solución a este problema, y es preciso aclarar que, en sí mismas, no constituyen ninguna restricción a la libre distribución ni a la creación de nuevos contenidos.

Utilizado de forma adecuada, el watermarking fomenta la creación de nuevos contenidos, y protege al autor de posibles abusos, ya que es el autor quien decide qué utilización se puede hacer de su contenido, y cual no.

Como ejemplo, podríamos imaginar a un fotógrafo que decidiera crear una página *web* en Internet, con sus fotografías e información relacionada con su profesión, pero con la particularidad de introducir algún tipo de sistema de autentificación en las imágenes, tal como watermarking.

Evidentemente, el hecho de colocar el contenido en Internet permite que cualquiera que acceda a la web pueda copiar las fotografías, y hacer cualquier uso de ellas.

El dueño legítimo de las fotografías podría decidir que cualquier persona que hiciera un uso personal, o simplemente que no obtuviera un beneficio económico, estaría autorizada a copiar, distribuir o publicar su trabajo, pero que si una empresa pretende copiar las fotografías de su página web para colocarlas en la suya propia o en catálogos para su propio lucro y beneficio o, lo que es más grave, que afirmara ser la propietaria del contenido, entonces el autor legítimo estaría en su derecho de no dar el consentimiento para esa utilización, y gracias al watermarking podría demostrar que él es el verdadero propietario.

Respecto a la autentificación de los datos, podríamos poner el ejemplo de un satélite militar que tome fotografías de la posición de vehículos y armamento de sus enemigos.

Supongamos ahora que el país propietario de ese satélite tiene unos intereses

ocultos (por ejemplo, la posibilidad de hacerse con el control de la zona o de especular con sus riquezas), y está dispuesto a iniciar una guerra para llevar a término sus objetivos.

Dado que no le es posible justificar sus verdaderas motivaciones, necesita convencer a otros países de la necesidad del ataque, y una manera de conseguirlo puede consistir en manipular las imágenes de sus satélites para que muestren camiones transportando peligrosos misiles de largo alcance y elementos para la construcción de armas de destrucción masiva.

Por lo tanto, en este ejemplo totalmente ficticio, el secretario del país atacante podría presentarse ante una comisión de la ONU con sus pruebas falsificadas para tratar de convencer de la necesidad de la acción bélica.

Para evitar este tipo de fraudes de consecuencias tan serias, los países que acuerden formar un grupo de "confianza"podrían decidir introducir algún tipo de sistema de autentificación en las imágenes o vídeos de sus satélites, de manera que si alguien manipulara el contenido (desde que el satélite toma la imagen, hasta el momento en que es recibida) entonces el cambio sería detectado por el receptor.

En el ejemplo anterior del satélite militar, el país que pretende iniciar la guerra debería introducir un esquema de autentificación en sus imágenes como pre-requisito para ser aceptadas por el resto de países. Por ejemplo, podría introducir una marca invisible en la imagen, generada a partir de la clave pública del resto de los países a los cuales pretende convencer, de manera que sólo estos receptores podrían extraer los datos de la marca.

Evidentemente, no es posible modificar la imagen en tránsito, ya que esos cambios destruirían la marca, y tampoco es posible eliminar o modificar la marca sin que se detecte, ya que no se dispone de la clave privada de los receptores.

Las técnicas de watermarking digital son buenas candidatas a la hora de intentar cumplir con los objetivos de asociar un contenido a su verdadero dueño, y en la autentificación y detección de modificaciones en el contenido. Por supuesto, estas técnicas se pueden utilizar tanto para fines loables, como la detección de falsificaciones, como para otros que claramente lesionan los derechos de las personas, como el seguimiento de los contenidos generados por un determinado usuario, marcados con un watermark invisible, con la consiguiente pérdida de intimidad de ese usuario.

Precisamente, la Recording Industry Association of America Inc. (RIAA), está utilizando [1] en la actualidad el método de insertar marcas invisibles en

[1]En agosto de 2003, una mujer de California fue llevada a juicio por la RIAA acusada de poseer ilegalmente material protegido por copyright en su ordenador. Hay precedentes de actuaciones contra otras compañías, como el famoso caso Napster, aunque es el primero en el que la RIAA actúa contra un usuario en particular. El 28 de agosto de 2000, la RIAA llevó a

los contenidos de canciones con copyright codificadas en formato MP3, para hacer un seguimiento de los usuarios que copian y retransmiten a otros usuarios los contenidos en redes P2P (Peer-to-Peer). Hay quien cree que este tipo de técnicas son un abuso por parte de las compañías discográficas, ya que opinan que el usuario debe tener derecho a guardar copias de los contenidos siempre y cuando no obtenga un beneficio económico por ello, mientras que la otra parte argumenta que está en su derecho de identificar a los infractores del copyright y actuar legalmente contra ellos.

En este proyecto no se va a hacer ninguna valoración de las diferentes opiniones de defensores y detractores de este tipo de técnicas, sino que simplemente las vamos a analizar desde el punto de vista matemático, con la mayor objetividad e imparcialidad posible. Como se ha dicho antes, estas técnicas no son beneficiosas o lesivas en sí mismas, sino que dependen de la aplicación en concreto.

Una vez introducido el tema, podemos pasar a definir qué es el watermarking digital.

Definición de Watermarking Digital y clasificación

Podemos definir el watermarking digital como una serie de técnicas que permiten introducir ciertas modificaciones sobre un determinado contenido digital (una imagen, vídeo, sonido, etc.) de manera que puedan ser servir para cumplir uno de los siguientes dos objetivos fundamentales:

- Asociación del contenido con su legítimo autor.
- Detección de modificaciones sobre el contenido original.

Normalmente los métodos de watermarking se utilizan sólo para satisfacer uno de los dos objetivos, y generalmente se utiliza el watermarking *robusto* para identificar al autor del contenido, y el *frágil* para detectar falsificaciones o modificaciones del contenido original.

Cualquier técnica de watermarking (tanto robusto como frágil) debería cumplir una serie de requisitos:

Visualmente imperceptible: Esta es una de las características más importantes, ya que se pretende que el contenido marcado sea lo más pareci-

juicio a MP3.com. Por otra parte, en septiembre de 1999 un chico suizo de 17 años fue acusado formalmente de piratería por la International Federation of the Phonographic Industry (que representa a 53 compañías discográficas) por almacenar canciones con copyright en formato MP3 en su ordenador.

do posible al original, idealmente iguales [2]. Evidentemente, el contenido marcado ha sufrido una serie de modificaciones que lo hacen diferente del original, pero simplemente se pretende que un observador humano los perciba iguales, aunque en realidad no lo sean.

Robusto: Significa que ha de ser resistente a ciertos "ataques", que pueden ser tanto el resultado de un procesamiento digital del contenido (por ejemplo, compresión), como por la acción intencional de un atacante que modifica el contenido para intentar eliminar la firma de autentificación o los datos que permiten determinar si el contenido ha sido modificado o no. Es decir, ha de poder aceptar ciertas modificaciones (dentro de unos límites especificados por el método), y seguir funcionando correctamente [3].Los ataques intencionales clásicos en imágenes son:

1. Compresión JPEG (y en general, cualquier codificación con pérdidas).
2. Modificación del histograma (ecualización y estiramiento).
3. Filtrado paso bajo, y filtros medianos (*median filters*).
4. Adición de ruido gaussiano, y no gaussiano.
5. *Dithering*.
6. Cambios de escala.
7. Recortes en la imagen (*cropping*).
8. Rotaciones respecto del centro de la imagen.
9. Rotaciones respecto de un punto arbitrario.
10. Translaciones.
11. Deformaciones geométricas.
12. Conversión A/D y D/A.
13. Recuantización del valor de los pixels, o de coeficientes de transformadas frecuenciales (p. ej. : DTC, DFT, Wavelets, etc.)
14. Reducción de color (p. ej. : pasar de 256 niveles gris/pixel a 128 niveles/pixel).
15. Sumar un valor constante (*offset*) a todos los pixels de la imagen.
16. Intercambio de pixels o bloques de pixels de la imagen.
17. Inserción de múltiples marcas.

Inequívoco: Ha de identificar inequívocamente al autor del contenido, y descartar a posibles candidatos que no sean el legítimo autor cuando se utiliza como método de identificación de propietario.

[2]Existe una excepción a este punto, que es un tipo especial de técnicas de watermarking en las que la modificación es visible. Se las conoce como *watermarking visible*.

[3]No obstante, a veces es deseable que el sistema no sea robusto, ya que esto permite detectar modificaciones en la imagen. En este caso el cometido del watermark es verificar la autenticidad de la imagen, detectando las modificaciones, y es lo que se conoce como *watermarking frágil*.

Innumerable: También en el caso de ser utilizado para la identificación de un determinado propietario, el sistema ha de poder generar diferentes firmas para diferentes autores, de manera que sea posible distinguir contenidos de un autor de los de otro, mediante firmas diferentes.

Difícil de eliminar: Las modificaciones introducidas en el contenido han de modificarlo de manera que, a posteriori, analizando la imagen marcada, no sea posible (o sea muy difícil) saber en qué han consistido exactamente esas modificaciones [4], para evitar que un atacante las pueda eliminar sin afectar significativamente la calidad de la imagen.

Para este proyecto de final de carrera en concreto, el objeto de estudio será la inserción y detección de marcas tipo watermark en imágenes fijas en nivel de gris, por lo que a partir de ahora nos referiremos únicamente a imágenes.

Podemos dividir las posibles técnicas de watermarking según el dominio en el cual sea hace efectiva la modificación, y entonces nos quedan dos grandes categorías:

Dominio espacial: Las modificaciones se hacen directamente sobre los pixels de la imagen, cambiando alguna característica espacial, como por ejemplo modificando el valor del LSB[5] de la luminancia del pixel.

Dominio frecuencial: En este caso se transforma la imagen al dominio de las frecuencias, utilizando transformadas como por ejemplo la DFT[6], o la DCT[7]. Generalmente, cuando se utiliza la inserción de marcas en el dominio frecuencial, las marcas se colocan en las frecuencias medias, ya que las modificaciones en frecuencias bajas resultan muy visibles y las frecuencias altas son eliminadas por muchos algoritmos de compresión, como por ejemplo JPEG.

También es posible transformar la imagen mediante la DWT[8], que es una transformada que nos da información espacial y frecuencial al mismo tiempo, por lo que es una combinación de las dos anteriores.

Independientemente de la técnica utilizada, se puede hacer otra división más, esta vez teniendo en cuenta si el algoritmo de detección requiere o no la imagen original para su funcionamiento.

[4] Las técnicas de marcado basadas en espectro expandido (spread spectrum) son un buen ejemplo de marcas difíciles de eliminar.
[5] Less Significant Bit
[6] Discrete Fourier Transform.
[7] Discrete Cosine Transform.
[8] Discrete Wavelet Transform.

En este caso, la división es la siguiente:

Watermarking "ciego" (blind): El detector no requiere la imagen original para funcionar.

Watermarking "no-ciego"(non-blind): El detector compara la imagen marcada con la original.

Evidentemente, es deseable que el detector pueda funcionar únicamente con la imagen marcada, y de hecho, la mayoría de los algoritmos utilizados para verificar la autoría de los contenidos son de este tipo.

En el caso de utilizar el sistema de watermarking para la autentificación del contenido, es evidente que el módulo verificador no debería necesitar la imagen original, ya que entonces cualquier posible algoritmo se podría reducir a una simple comparación de los bits de la imagen original con la que estamos comprobando. No sólo eso, sino que además haría necesaria la existencia de una autoridad de verificación que mantuviera copias de todas aquellas imágenes que se quisieran verificar en el futuro. Es deseable que el verificador funcione únicamente con los datos de la imagen que se quiere comprobar, y en algunos casos, también el identificador del usuario que marca el contenido.

Para terminar con esta introducción general, vamos a ver qué diferencias hay entre las técnicas de watermarking digital, y las de esteganografía.

La **esteganografía** es el conjunto de métodos que permiten *ocultar* una información secreta, que es insertada en otro contenido (por ejemplo, una imagen) de manera que haga de portador de esa información, y que no sea detectada.

El receptor recibe el contenido, y utiliza el algoritmo adecuado que le permite extraer la información.

Si algún *espía* ha logrado interceptar el mensaje, en principio no tendría por qué sospechar que en esa imagen hay una información secreta, por lo que ni siquiera intentará aplicar métodos de criptoanálisis o de fuerza bruta para extraer o desencriptar la información, ya que no la habría detectado.

Podría parecer que el watermarking es una forma de esteganografía, pero en sentido estricto esto no es así, y los motivos son varios:

1. En el caso de la esteganografía, lo que se pretende es ocultar que hay un mensaje oculto en el medio portador. En el watermarking es todo lo contrario, ya que muchas veces se informa explícitamente del hecho de que las imágenes han sido marcadas, para evitar usos ilícitos.

2. En la mayoría de los algoritmos de watermarking es posible saber si una imagen marcada ha sido generada por una clave en concreto, *pero no son*

invertibles, en el sentido de que dada una imagen, no es posible (ni deseable que lo sea) obtener la clave generadora.

3. Con la esteganografía se pretende poder insertar un mensaje, que luego será recuperado en detección. Sin embargo, en las técnicas de watermarking el objetivo final no es extraer la información insertada, sino simplemente asociar una imagen (o cualquier otro medio) con un determinado autor. Esto es una consecuencia del punto anterior.

Capítulo 4

Watermarking robusto

Tal y como se explicó en la introducción, uno de los usos que se le puede dar a una técnica de watermarking es la de conseguir asociar un determinado contenido (por ejemplo, una imagen) con su verdadero autor.

Para este tipo de aplicaciones, es necesario que el método de watermarking utilizado sea *robusto*, es decir, que sea resistente tanto a ataques intencionales destinados a eliminar la marca de autentificación, como a los efectos colaterales provocados por el procesamiento digital de la señal, tales como son la compresión (por ejemplo, JPEG), el filtrado, la reimpresión, el reescaneado, etc.[1]

La necesidad de que el método sea robusto viene dada precisamente porque se quiere poder recuperar la firma de autentificación (generada por el propio usuario, o una autoridad de autentificación reconocida) a pesar de la manipulación del contenido (en nuestro caso, de la imagen).

Por ejemplo, supongamos que un periodista especializado en la prensa *del corazón* ha conseguido tomar ciertas imágenes que son del interés de algunas revistas, siempre y cuando les sean entregadas en exclusividad.

Supongamos que previamente el periodista se ha registrado en la base de datos de una autoridad de watermarking, que ha generado una firma asociada a esta persona.

Sólo la autoridad de watermarking conoce la firma, y se supone que no es posible extraerla de imágenes marcadas, dado que no se conocen otros parámetros importantes del algoritmo de inserción.

Entonces, si se diera el caso de que alguna otra agencia de noticias *capturase* una copia de la imagen marcada (por ejemplo, cuando era enviada mediante una red inalámbrica no segura), y la publicase en su propia revista (con el

[1] En el capítulo 3 se enumeran cuáles son los ataques clásicos que se suelen utilizar a la hora de intentar invalidar un método de watermarking.

consiguiente perjuicio económico para el verdadero autor), entonces el periodista podría demostrar ante un tribunal que la fotografía que aparece en la revista de la competencia fue tomada por él, dado que la autoridad de watermarking puede detectar su firma invisible en la fotografía, al mismo tiempo que la revista rival no puede dar ninguna firma tal que se pueda detectar en la imagen.

En este caso, está clara la necesidad de que el esquema de watermarking utilizado sea robusto, dado que la revista rival puede efectuar diversas clases de procesamiento a la imagen, como podrían ser cambios de escala, tomar sólo una parte de la imagen (*cropping*), compresión JPEG, reducción del número de colores, etc.

Aún a pesar de estas modificaciones, el método de watermarking robusto debería ser capaz de detectar la firma de autentificación. Evidentemente, existen ciertos límites que no deben ser sobrepasados, como por ejemplo, que si se toma una parte de la imagen (subimagen), ésta tenga unas dimensiones mínimas, o que la compresión JPEG no haya degradado la imagen tanto que no se pueda considerar que se trate de la imagen original.

En general, cualquier algoritmo de watermarking consiste de tres elementos claramente diferenciados:

- El watermark, o firma de autentificación.
- El insertor.
- El detector.

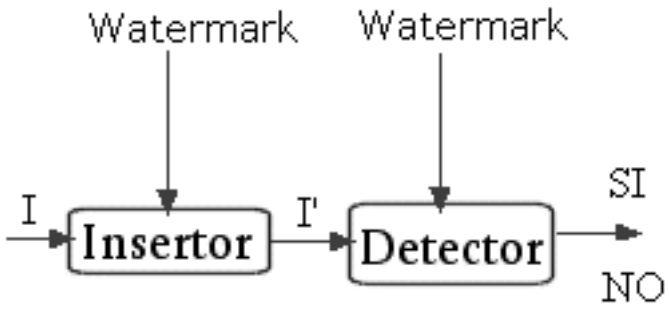

Figura 4.1: Esquema de un sistema de watermarking.

El *watermark* o firma de autentificación no es más que una secuencia (generalmente binaria, aunque esto depende del algoritmo en concreto) que se asocia con un determinado autor, y que se utiliza para generar la señal invisible que se

inserta en la imagen, que es utilizada por los módulos insertor y detector, para insertar la señal, y detectarla, respectivamente.

El *insertor* es el módulo que implementa el algoritmo que toma la imagen original y la firma de autentificación, y genera una nueva imagen en la que es posible detectar la marca, y que es muy parecida a la imagen original (idealmente, un observador humano no debería percibir diferencias).

Si llamamos I a la imagen original y S al watermark, podemos definir el insertor como una función matemática que toma I y S, y genera una nueva imagen marcada I':

$$E(I, S) = I' \tag{4.1}$$

El dominio en el que opera la función E de inserción depende del algoritmo utilizado, aunque se suelen dividir en técnicas espaciales, que modifican directamente el valor de algún parámetro de los pixels de la imagen (como por ejemplo, su luminancia), o métodos transformados, en los que se modifican variables de la transformada de la imagen (DCT, DFT, wavelet, etc.), como por ejemplo el módulo de los coeficientes, la fase o el nivel de cuantización, entre otros.

El *detector* es el módulo que implementa el algoritmo que toma la imagen marcada y la firma de autentificación, y decide si la firma de autentificación S que se está verificando se encuentra, o no, en la imagen que se analiza.

Si llamamos I' a la imagen marcada y S al watermark, podemos definir el detector como una función matemática que toma I' y S, y genera una variable de salida mediante la cual decide si S se corresponde con el watermark que se detecta de la imagen I'.

$$D(I', S) = C \tag{4.2}$$

donde C puede tomar los valores *POSITIVO* (el watermark se corresponde con el de la imagen), o *NEGATIVO* (el watermark no se corresponde).

El método de decisión que genera la condición de detección positiva o negativa depende del algoritmo en concreto, aunque por lo general consiste en calcular la correlación entre la marca que se está comprobando y la que se detecta en la imagen.

Este valor de la correlación, se suele comparar con un umbral δ que se ajusta de manera que el detector sea más o menos estricto a la hora de decidir si la detección es positiva o negativa.

Si el detector es muy estricto, requerirá una alta correlación entre el watermark comprobado y el presente en la imagen, y por lo tanto la probabilidad de asignar la imagen a un autor que no le corresponde será baja, pero al mismo tiempo el método será muy sensible a los ataques, ya que éstos también afectan a la marca insertada, de manera que la correlación disminuye a medida que aumenta la intensidad de los ataques.

Por otra parte, si se relaja demasiado el umbral de decisión, el detector obtendrá una gran inmunidad a los ataques, pero también aumentará la probabilidad de que se produzca un *falso positivo*, es decir, que el detector decida erróneamente que la marca de un determinado autor está presente en la imagen, cuando en realidad esto no sea así, y el error se produzca porque el detector admite una correlación insuficiente entre las dos marcas para decidir el positivo.

Por lo tanto, se procura ajustar el parámetro δ de manera que el sistema tenga una probabilidad de error en falso positivo aceptable en su ámbito de uso, y que al mismo tiempo sea resistente a ciertos ataques, en los que previamente se ha decidido cuál es el umbral que no llega al límite de calidad exigible.

En muchos algoritmos, el umbral se selecciona de manera automática en función de las características de la imagen.

4.1. Revisión de técnicas previas

Hasta la fecha se han desarrollado muchas técnicas de watermarking robusto aplicadas a resolver el problema de asociar imágenes con sus legítimos autores, y en este apartado se mostrarán algunas de las más significativas, para dar una visión general de lo desarrollado hasta el momento. La revisión se centrará en los métodos de watermarking robusto e invisible.

Como se ha explicado anteriormente, una división posible de las técnicas de watermarking es según el ámbito de en el cual operan el insertor y el detector, y dentro de las técnicas de métodos transformados, existe un artículo que sin lugar a dudas es un referente de muchas técnicas de watermarking robusto: *"Secure Spread Spectrum Watermarking for Multimedia"* [3], de los autores I.J. Cox, J. Kilian, T. Leighton y T. Shamoon, publicado en el año 1995.

Muchas técnicas posteriores se han basado en este trabajo, que se ha visto modificado y mejorado posteriormente, pero al ser un referente tan importante, es conveniente incluirlo en esta revisión.

Para insertar el watermark, se dan los siguientes pasos:

1. Se calcula la transformada DCT-2D de la imagen.
2. Se crea un watermark $X = \{x_1, x_2, ..., x_n\}$, en el que cada componente x_i es una realización de una variable aleatoria normal N(0,1), es decir, una gaussiana de media cero y varianza unidad.
3. Se inserta el watermark según la regla $v_i' = v_i(1 + \alpha x_i)$, donde v_i' es el coeficiente i-ésimo de una lista de los n coeficientes de la DCT de valor absoluto más elevado (se asume orden en zig–zag), v_i es el coeficiente original, y α es un escalar que modula la intensidad de la inserción, y que generalmente toma el valor $\alpha = 0.1$.

La detección se hace de la siguiente manera:

1. Se calcula la transformada DCT-2D de la imagen original.
2. Se calcula la transformada DCT-2D de la imagen supuestamente marcada.
3. Se extrae el watermark insertado en las dos imágenes, y se crea el watermark X^*, que es la diferencia entre ambos : ${x_i}^* = \frac{{v_i}^* - v_i}{\alpha v_i}$
4. Luego se calcula la función de *similitud* entre el watermark X^* y X, definida como $sim(X, X^*) = \frac{XX^*}{\sqrt{XX^*}}$.
5. Finalmente, se compara el valor de la función de similitud con el de un umbral prefijado, y se decide si el watermark X está presente, o no, en la imagen que se comprueba.

La mayor aportación de este método es que por primera vez introduce un método basado en espectro expandido, es decir, que la información del watermark no se concentra en un pequeño conjunto de coeficientes (lo que aumentaría mucho su energía), sino que se reparte en un conjunto muy numeroso de ellos.

La ventaja es que si la energía del watermark queda repartida en muchos coeficientes, no es posible detectar qué conjunto de coeficientes son los que portan la información y eliminarlo, ya que son muchas las frecuencias las que se ven afectadas, y cualquier intento por eliminar la información sin conocer la imagen original,el watermark, o la posición exacta de los coeficientes no sería válida, ya que la imagen se degradaría demasiado visualmente.

Otra ventaja es que dado que la energía se reparte entre muchos coeficientes, la imagen resultante es muy parecida a la original, ya que tanto la energía como el valor de los coeficientes afectados apenas varía.

La desventaja más evidente de este método es que el detector requiere de la imagen original para verificar que el watermark que se comprueba está presente en la imagen marcada, lo cual fue corregido en otros métodos, como por ejemplo en [1], entre muchos otros.

Existen otros métodos transformados, como por ejemplo el descrito por Deepa Kundur y Dimitros Hatzinakos en su artículo *"Digital Watermarking using Multiresolution Wavelet Decomposition"* [9], en el que utilizan la transformada wavelet, y a diferencia del método anterior, no se requiere de la imagen original.

El método consiste, sin entrar en demasiados detalles, en utilizar un watermark binario cuyos elementos se seleccionan del conjunto $\{-1, +1\}$.

Cada usuario dispone de una clave compuesta por un número determinado de elementos binarios, que indican si un determinado coeficiente de la descomposición wavelet debe ser modificado, o no.

En el caso de que tenga que ser modificado se utiliza el valor del watermark para cuantizar el coeficiente, lo cual que se puede hacer de dos maneras, dado que cada elemento del watermark permite dos posibilidades.

El detector decide si el watermark comprobado se corresponde con el insertado en una imagen calculando la correlación del watermark extraído de la imagen con el que se comprueba, y comparando el coeficiente de correlación siguiente con un umbral T : $\rho(\omega, \widetilde{\omega}) = \frac{\sum \omega(n)\widetilde{\omega}(n)}{\sqrt{\sum \omega^2(n)}\sqrt{\sum \widetilde{\omega}^2(n)}} \geq T$

Otro ejemplo de técnica de watermarking robusto basada en un método transformado es la descrita en *"A DCT-domain System for Robust Image Watermarking"* [1], que será explicada detalladamente en la sección 4.2.

Esta técnica utiliza la transformada DCT, no requiere de la imagen original para su funcionamiento, y está basada en el cálculo de la correlación entre el watermark que se comprueba y el detectado en la imagen, que es comparado con un umbral que se ajusta automáticamente según las características de la imagen.

Respecto a las técnicas espaciales, es decir, en las que se modifica directamente algún parámetro de los puntos de la imagen (por ejemplo, la luminancia), existen algunos métodos que han combinado aspectos de la criptografía con el watermarking, aplicados a asociar una determinada imagen con su verdadero autor.

Se han desarrollado muchas otras técnicas de watermarking, muchas de las cuales han corregido deficiencias de métodos anteriores, y han añadido nuevas características. Los presentados en esta revisión son sólo unos pocos representantes.

En general, podemos concluir que los métodos de watermarking robusto más relevantes que se han desarrollado hasta la fecha consisten en generar un watermark, la mayoría de las veces consistente en una secuencia binaria de longitud determinada, y utilizar un algoritmo insertor que, dependiendo del método en concreto, inserte la información presente en el watermark en algún parámetro de la imagen, como por ejemplo, en la luminancia de los pixels (métodos espaciales), o modificando algún parámetro (fase, amplitud, ...) de los coeficientes transformados (DCT, DFT, DWT, etc.).

El detector, también hablando desde un punto de vista general, compara los valores de la secuencia watermark que se comprueba, con los que presenta la imagen, y calcula un coeficiente de correlación que es comparado con un umbral, que puede ser predeterminado, o ajustado de manera automática según las características de la imagen.

Este umbral se ajusta para conseguir un compromiso entre una baja probabilidad de error (falso positivo), y al mismo tiempo, conseguir una cierta inmunidad a los ataques que se puedan producir, dentro de unos límites establecidos por el ámbito de uso del sistema.

Si el coeficiente de correlación es igual o sobrepasa el umbral, se decide que el watermark comprobado está presente en la imagen, y en caso contrario, se decide que no está presente.

Por supuesto, existen métodos de watermarking robusto diferentes a lo explicado anteriormente pero, por lo general, se suelen ajustar a este esquema.

4.2. Estudio e implementación de un algoritmo de watermarking robusto

4.2.1. Introducción

En esta parte del informe del proyecto se hace un estudio del algoritmo de watermarking descrito en el artículo *"A DCT-domain System for Robust Image Watermarking"* [1], de Mauro Barni, Franco Bartolini, Vito Cappellini y Alessandro Piva, de la Universidad de Florencia (Italia), que está basado principalmente en el método de I.J. Cox, J. Kilian, T. Leighton y T. Shamoon [3], [4].

El algoritmo ha sido implementado mediante un programa en C++, por lo que ha sido posible evaluar y estudiar su comportamiento en diferentes condiciones.

Se explicará el algoritmo y los fundamentos matemáticos a partir de los cuales funciona el sistema, y se efectuarán una serie de pruebas para verificar que el algoritmo realmente funciona tal y como explican sus autores, y para los ataques a los cuales se le supone robusto.

También se comentarán algunas limitaciones que se han observado en este método, y en una sección posterior (4.3) se propondrán modificaciones significativas para hacerlo más robusto, incluso a ataques a los cuales antes no era resistente, tales como rotaciones de cualquier ángulo y translaciones, con lo cual el método queda claramente mejorado.

El método propuesto por los autores anteriormente citados consiste en un algoritmo de watermarking para imágenes en escala de gris, y "ciego", lo cual significa que en la fase de detección no es necesario contar con la imagen original.

Opera en el dominio de las frecuencias, e inserta una secuencia pseudo-aleatoria de números reales en un conjunto de coeficientes de la transformada DCT.

Para determinar si el watermark está o no presente en la imagen, se compara un valor de correlación con un determinado umbral, como veremos a continuación.

El sistema es resistente a varios ataques, en concreto, a ataques no intencionales, como el procesamiento de la imagen por parte de software de retoque fotográfico (por ejemplo, compresión JPEG), y a varios ataques intencionales, como filtros paso-bajo y medianos (*median filters*), ecualización y estiramiento (*stretching*) del histograma, *dithering*, adición de ruido gaussiano, cambios de escala e inserción de varios watermarks en una misma imagen.

4.2.2. Inserción del watermark

El watermark $X = \{x_1, x_2, ..., x_M\}$ consiste en una secuencia pseudo-aleatoria de longitud M, en la que cada x_i es un número pseudo–aleatorio real con distribución de Bernoulli, de media cero y varianza unidad.

Para la detección del watermark es importante que los números reales x_i que constituyen los diferentes watermarks sean estadísticamente independientes. Esta característica se consigue gracias a la naturaleza pseudo-aleatoria del generador del vector X.

Para insertar el watermark en la imagen de nivel de gris I de dimensiones $N \times N$, primero se calcula su DCT, y los coeficientes se consideran según el orden de un barrido en *zig–zag*.

La marca (el vector pseudo–aleatorio X) se insertará desde el coeficiente L+1 hasta el M+L, según el orden en zig–zag del espectro de la DCT, en el que los L primeros coeficientes no se tienen en consideración, para conseguir que la marca sea visualmente imperceptible.

Estos coeficientes originales constituyen el vector $T = \{t_{L+1}, t_{L+2}, ..., t_{L+M}\}$, y serán modificados de esta manera:

$$t'_{L+i} = t_{L+i} + \alpha |t_{L+i}| x_i \qquad , \qquad i = 1, 2, ..., M \tag{4.3}$$

Estos coeficientes serán los elementos del vector $T' = \{t'_{L+1}, t'_{L+2}, ..., t'_{L+M}\}$ de coeficientes modificados.

Finalmente, el vector T' es reinsertado en el espectro de la imagen siguiendo el orden en zig–zag, y se calcula la IDCT para obtener la imagen marcada I'.

Para implementar el generador de números aleatorios de media cero y varianza unidad, se programó en C++ una función que con probabilidad $p = \frac{1}{2}$ genera el valor +1 y con probabilidad $1 - p = \frac{1}{2}$ genera el valor -1 (variable aleatoria de Bernoulli), con lo cual se cumple que $\mu = 0 \quad y \quad \sigma^2 = 1$.

Por lo tanto, los pasos que dará el insertor serán los siguientes:

1. Lectura de la imagen original.
2. Lectura del watermark X que se va a insertar.
3. Transformar la imagen I mediante la DCT.
4. Formación del vector de coeficientes T.
5. Modificar el vector T según los valores de X, para obtener el vector $T^{'}$.
6. Cambiar los coeficientes originales de la imagen I por los coeficientes modificados del vector $T^{'}$.

7. Calcular la IDCT de la imagen I para obtener la imagen marcada.

8. Opcionalmente, se puede aplicar un enmascaramiento visual para mejorar la imperceptibilidad del watermark, tal y como se explica más adelante.

4.2.3. Detección del watermark

Dada una imagen I^*, posiblemente corrompida por la acción de diversos ataques, se calcula su transformada DCT $N \times N$.

Los coeficientes DCT de I^* se ordenan en zig–zag, y comenzando por el $L+1$ hasta el $L+M$ (inclusive), se seleccionan para formar el vector de coeficientes marcados y posiblemente corrompidos $T^* = \{t^*_{L+1}, t^*_{L+2}, ..., t^*_{L+M}\}$.

Dado que es imposible estimar la marca restando los coeficientes DCT sin marcar de T^*, se utiliza como indicador de la presencia de la marca la correlación entre los coeficientes marcados y posiblemente corrompidos T^* y la marca en sí misma.

Concretamente, utilizaremos la correlación z definida como:

$$z = \frac{YT^*}{M} = \frac{1}{M}\sum_{i=1}^{M} y_i t^*_{L+i}$$

En la que y_i es el watermark que estamos verificando, y t^*_{L+i} son los coeficientes DCT marcados, y posiblemente corrompidos por algún ataque.

Los coeficientes DCT que almacenamos en el vector T son siempre los coeficientes que hemos obtenido de aplicar el orden zig–zag, ignorando los L primeros y almacenando los M siguientes, por lo que a partir de ahora, y para no sobrecargar la notación, omitiremos el sumando L del índice de los coeficientes, dando por sobreentendido que al escribir t^*_k nos referimos en realidad al que ocupa la posición $L + k$.

Por lo tanto, siguiendo esta nueva notación, más clara, tenemos que

$$z = \frac{YT^*}{M} = \frac{1}{M}\sum_{i=1}^{M} y_i t^*_i \tag{4.4}$$

En el proceso de detección, el detector forma el vector de coeficientes T^* (siguiendo el orden en zig–zag, obviando los L primeros, y almacenando los M siguientes), y calcula la correlación z entre T^* y el watermark Y que estamos comprobando.

Suponiendo que la imagen con el watermark no ha sido corrompida, tenemos que

$$t^*_i = t'_i = t_i + \alpha|t_i|x_i \qquad , \qquad i = 1, 2, ..., M \tag{4.5}$$

y por lo tanto, sustituyendo el valor de t_i^* de la ecuación (4.5) en la ecuación (4.4) obtenemos

$$z = \frac{1}{M}\sum_{i=1}^{M} y_i t_i^* = \frac{1}{M}\sum_{i=1}^{M}(t_i y_i + \alpha|t_i|x_i y_i) \tag{4.6}$$

Veamos ahora qué valores toma el parámetro z dada una imagen I', y diferentes watermarks, según se hayan utilizado, o no, para marcar la imagen.

Supongamos que el watermark Y es igual al watermark X insertado en la imagen.

Entonces tenemos que

$$\begin{aligned} X = Y \Rightarrow \quad z &= \tfrac{1}{M} \quad \textstyle\sum_{i=1}^{M} y_i t_i^* && = \\ &= \quad \tfrac{1}{M}\textstyle\sum_{i=1}^{M}(t_i y_i + \alpha|t_i|x_i y_i) && = \\ &= \quad \tfrac{1}{M}\textstyle\sum_{i=1}^{M}(t_i x_i + \alpha|t_i|x_i^2) && = \\ &= \quad \tfrac{1}{M}\textstyle\sum_{i=1}^{M} t_i x_i + \tfrac{1}{M}\sum_{i=1}^{M}\alpha|t_i|x_i^2 && = \\ &= \quad \tfrac{1}{M}\textstyle\sum_{i=1}^{M} t_i x_i + \alpha\tfrac{1}{M}\sum_{i=1}^{M}|t_i|x_i^2 \end{aligned}$$

Por lo tanto, tenemos que:

$z = \mu(TY) + \alpha\mu(|T|1) = \mu(TY) + \alpha\mu(|T|) = \mu(T)\mu(Y) + \alpha\mu(|T|) =$

$= 0 + \alpha\mu(|T|) = \alpha\mu(|T|)$

(Ver esta nota[2]).

Esto es así, dado que:

- $x_i^2 = 1$.
- T, X son variables aleatorias independientes.
- La media de los valores de X es cero.
- $\mu(TX) = \mu(T)\mu(Y)$, si T y X son independientes[3].

[2]En general, dada una variable aleatoria discreta S para la cual se conocen M valores (M realizaciones), se puede aproximar el valor medio de estos M valores por la esperanza de S : $\frac{1}{M}\sum_{i=1}^{M} S_i \approx \mu(S)$. La aproximación será tanto mejor cuanto mayor sea M.

[3]Los autores afirman además que los valores t_i también tienen media cero, ya que son la suma de coeficientes de la DCT tomados en *zig–zag*.

Veamos ahora qué ocurre si $X \neq Y$:

$$\begin{aligned} X \neq Y \Rightarrow \quad z = \tfrac{1}{M} \quad & \textstyle\sum_{i=1}^{M} y_i t_i^* & = \\ = \quad & \tfrac{1}{M} \textstyle\sum_{i=1}^{M} (t_i y_i + \alpha |t_i| x_i y_i) & = \\ = \quad & \tfrac{1}{M} \textstyle\sum_{i=1}^{M} t_i x_i + \tfrac{1}{M} \textstyle\sum_{i=1}^{M} \alpha |t_i| x_i y_i & = \\ = \quad & \tfrac{1}{M} \textstyle\sum_{i=1}^{M} t_i x_i + \alpha \tfrac{1}{M} \textstyle\sum_{i=1}^{M} |t_i| x_i y_i & \end{aligned}$$

Tenemos que T,X,Y son variables independientes, y la media de X e Y es cero, por lo que ahora

$$z = \mu(TY) + \alpha\mu(|T|XY) = \mu(T)\mu(Y) + \alpha\mu(|T|)\mu(X)\mu(Y) =$$

$$= 0 + \alpha\mu(|T|)0 = 0$$

Por lo tanto, para saber si el watermark que estamos comprobando realmente está presente en la imagen, el detector dará estos pasos:

1. Lectura de la imagen marcada.
2. Lectura del watermark Y que se va a comprobar.
3. Transformar la imagen I mediante la DCT.
4. Formación del vector de coeficientes T^*.
5. Calcular la media z.
6. Comparar z con un umbral T_z para determinar si la imagen ha sido marcada con ese watermark Y, o no.

Normalmente las imágenes que llegan al detector habrán sufrido algún tipo de ataque, por lo que el valor de z calculado puede que no alcance el valor esperado en el caso de que el watermark introducido sea el utilizado para marcar la imagen, aunque siempre estará próximo a cero si el watermark que verificamos no coincide con el que está presente en la imagen.

Por lo tanto, utilizaremos un umbral T_z menor que la media calculada anteriormente, en concreto tres veces por debajo [4], por lo que finalmente tenemos que:

$$T_z = \frac{\alpha}{3M} \sum_{i=1}^{M} |t_i^*| \tag{4.7}$$

Si $z \geq T_z \Rightarrow$ Detección POSITIVA
Si $z < T_z \Rightarrow$ Detección NEGATIVA

[4] Es un valor experimental, que se ha comprobado que da buenos resultados.

4.2.4. Enmascaramiento visual

Al aplicar la transformada DCT, descomponemos la imagen en suma de componentes sinusoidales de diferentes frecuencias, cada una con una amplitud determinada, según el coeficiente por el que esté multiplicada.

Este método se basa precisamente en modificar estos coeficientes de la DCT, lo cual significa que al reconstruir la imagen mediante el cálculo de su IDCT, hemos introducido una señal sinusoidal en el dominio espacial que antes no estaba presente en la imagen, y que la afecta de manera global.

Si la amplitud de esta señal superpuesta no modifica el valor del pixel original en más de cinco niveles (suponiendo que el rango es 0-255), entonces no es percibida por un observador humano.

Evidentemente, cuanto mayor sea la amplitud de la señal superpuesta, más visible será. Pruebas experimentales demuestran que un valor de $\alpha = 0.2$ es un buen compromiso entre un buen funcionamiento del método y la imperceptibilidad del watermark.

Así mismo, se sabe que cuando una señal interferente se superpone a una zona de la imagen, un observador humano es capaz de percibirla en tanto en cuanto el nivel de texturización de esa zona de la imagen sea bajo.

Zonas de alta texturización enmascaran la percepción de la señal superpuesta, mientras que zonas con poca o ninguna texturización permiten que el observador perciba su presencia.

Para conseguir que la señal interferente debida al watermark sea imperceptible, se puede actuar de dos maneras.

La primera, es controlar la amplitud de la señal superpuesta, y este aspecto está directamente relacionado con el parámetro α que utiliza el insertor a la hora de modificar los coeficientes de la DCT. Un valor de $\alpha = 0.2$ es adecuado.

El otro aspecto que se puede controlar es la amplitud de la señal interferente, pero no de manera global, como antes, sino teniendo en cuenta la textura local sobre la que se asienta, por lo que la imagen original I y la imagen marcada I' se suman pixel a pixel según un factor de peso $\beta_{i,j}$, para producir una nueva imagen marcada I", en la que el watermark es aún menos visible, gracias a las características del HVS[5].

La suma ponderada con el parámetro $\beta_{i,j}$ se hace de la siguiente manera:

$$y''_{i,j} = y_{i,j}(1 - \beta_{i,j}) + \beta_{i,j} y'_{i,j} = y_{i,j} + \beta_{i,j}(y'_{i,j} - y_{i,j}) \tag{4.8}$$

[5]Siglas de Human Visual System (sistema visual humano).

El parámetro $\beta_{i,j}$ depende de la región de la imagen en la que estemos insertando el watermark, de manera que en regiones de alta texturización, en las que de manera natural el watermark es menos visible, $\beta_{i,j} \approx 1 \Rightarrow y''_{i,j} \approx y'_{i,j}$, por lo que el watermark se inserta fuertemente, mientras que en regiones de baja texturización, en las que el watermark es fácilmente visible, haremos que $\beta_{i,j} \approx 0 \Rightarrow y''_{i,j} \approx y_{i,j}$

Para calcular el grado de texturización $\beta_{i,j}$, para cada pixel $y_{i,j}$ se ha considerado un bloque de tamaño $R \times R$ centrado en (i,j) [6], y se ha calculado la varianza del bloque (σ^2).

Finalmente, esta varianza[7] se normaliza respecto de la máxima que se obtiene para cada uno de los demás bloques, y se aplica en la ecuación (4.8) para obtener la imagen marcada final.

[6] En este caso se ha considerado R=9.

[7] Experimentalmente se ha comprobado que es mejor utilizar la variable $\frac{\sigma^2}{1+\sigma^2}$ en lugar de σ^2.

4.2.5. Estimación de la probabilidad de error

Como se puede ver en la Figura 4.2, la media de la variable aleatoria que genera z es diferente en el caso de detección positiva (derecha), o detección negativa (izquierda).

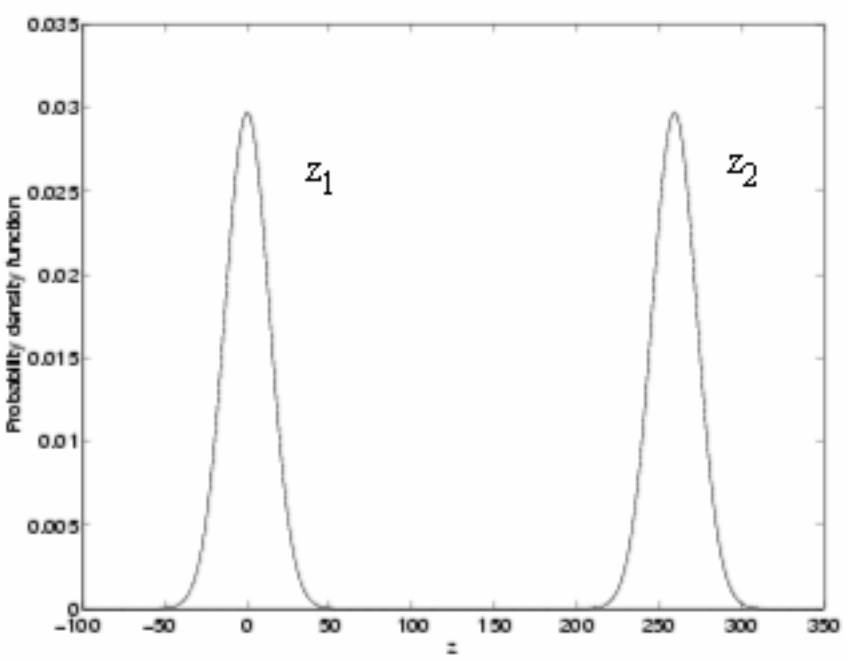

Figura 4.2: Funciones de densidad probabilidad de la variable aleatoria z, cuando el watermark detectado no coincide con el insertado (izquierda), y cuando sí coincide (derecha). La gráfica ha sido tomada del artículo [1], pág. 22.

La gráfica de la izquierda corresponde a la función de probabilidad en el caso de que el watermark que estamos comprobando no sea el que está presente en la imagen, y como vemos, el valor más probable es cero. La otra corresponde a la situación en la que el watermark realmente coincide con el insertado en la imagen, y entonces z ya no es cero, sino un valor mucho mayor.

Vamos a llamar z_1 a la curva de detección negativa (izquierda), y z_2 a la de detección positiva (derecha), y por lo tanto, tendremos que μ_1 y σ_1^2 serán respectivamente la media y la varianza de z_1, y que μ_2 y σ_2^2 serán las de z_2. Entonces, $\mu_1 = 0$, y $\mu_2 = \alpha\mu_{|t|}$, en ausencia de ataques.

Queremos calcular la probabilidad de que dado un watermark aleatorio que no se corresponde con el que está en la imagen, el detector decida que sí está pre-

sente, que es lo que se conoce como *falso positivo*.

Para ello, integraremos la función de densidad gaussiana de media cero desde T_z hasta ∞, con lo que estaremos calculando la probabilidad de que dado un watermark que no está presente en la imagen[8], el parámetro z esté por encima del umbral de detección que fijamos para el caso de que el watermark que comprobamos esté presente en la imagen. Tomaremos $T_z = \mu_2/3$ (ver ecuación (4.7)).

Así pues,

$$P_e = \frac{1}{\sqrt{2\pi\sigma_1^2}} \int_{T_z}^{\infty} e^{-\frac{x^2}{2\sigma_1^2}} dx = \frac{1}{2} erfc\left(\frac{T_z}{2\sigma_1^2}\right) \qquad , \tag{4.9}$$

donde *erfc(x)* es la función de error complementaria. También se ha supuesto que $\sigma_1^2 = \sigma_2^2$. Su relación con la varianza de los coeficientes de la DCT ($\sigma_t{}^2$) es la siguiente (ver el artículo [1], pág 10):

$\sigma_z^2 = \frac{1+2\alpha^2}{M}\sigma_t^2 + \frac{\alpha^2}{M}\sigma_{|t|}^2$, si $X = Y$

$\sigma_z^2 = \frac{1+\alpha^2}{M}\sigma_t^2$, si $X \neq Y$

$\sigma_z^2 = \frac{1}{M}\sigma_t^2$, si la imagen no está marcada

Además, dado que $\sigma_{|t|}^2 \ll \sigma_t^2$ y que $\alpha \ll 1$, entonces se puede escribir que $\sigma_z^2 \simeq \frac{\sigma_t^2}{M}$, para todos los casos.

Por lo tanto, a partir de (4.9) obtenemos:

$$P_e \approx \frac{1}{2} erfc\left(\frac{T_z}{2\sigma_t^2/M}\right) \tag{4.10}$$

Para conocer el valor numérico de P_e, necesitamos estimar el valor de la media y la varianza de la variable aleatoria que genera el parámetro z para un gran número de imágenes.

Los autores del artículo [1] utilizaron una base de datos con 170 imágenes monocromas, y concluyeron que una buena aproximación es considerar que la media $\mu_{|t|} = 0.7$, y que $\sigma_t^2 = 1$, cuando L y M están entre 10000 y 20000.

[8] Este es el motivo por el que se integra la gaussiana de media cero.

Si suponemos que el umbral se ha escogido de manera que $T_z = \mu_2/3 \quad ; \quad \alpha = 0.2$ (ver ecuación (4.7)), $L = M = 16000 \quad$, entonces sustituyendo en la ecuación (4.9) tenemos que:

$$P_e = \frac{1}{2} erfc\left(\frac{T_z}{\sqrt{2\sigma_z^2}}\right) \approx \frac{1}{2} erfc\left(\frac{T_z}{\sqrt{\frac{2}{M}\sigma_t^2}}\right) =$$

$$= \frac{1}{2} erfc\left(\frac{T_z}{\sqrt{\frac{2}{M}}}\right) = \frac{1}{2} erfc\left(T_z\sqrt{\frac{M}{2}}\right) =$$

$$= \frac{1}{2} erfc\left(\frac{\mu_2}{3}\sqrt{\frac{M}{2}}\right) = \frac{1}{2} erfc\left(\frac{\alpha\mu_{|t|}}{3}\sqrt{\frac{M}{2}}\right) =$$

$$= \frac{1}{2} erfc\left(\frac{0.2*0.7}{3}\sqrt{\frac{16000}{2}}\right) = 3.5712 * 10^{-9}$$

En promedio, el detector decidirá erróneamente (falso positivo) unas 3.6 veces por cada mil millones de imágenes sin marcar que analice.

En la Figura 4.3 vemos la evolución de la probabilidad de error según el valor del parámetro α, con $L = M = 16000$. Podemos observar cómo aproximadamente a partir de $\alpha = 0.15$, la probabilidad de error comienza a ser pequeña, y en concreto para $\alpha = 0.15$ la probabilidad de error es $P_e = 4.77346 * 10^{-6}$ (menos de 4.8 errores por cada millón de imágenes).

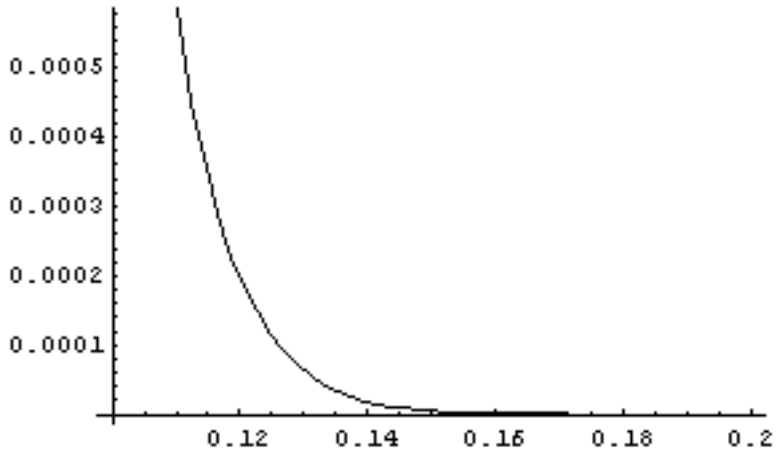

Figura 4.3: Probabilidad de error vs. parámetro α.

Valores por debajo de $\alpha = 0.15$ nos dan probabilidades de error cada vez más altas, y a un ritmo cada vez mayor, aunque con la ventaja de que el impacto

visual del watermark es prácticamente inexistente.

Al mismo tiempo, valores por encima de $\alpha = 0.15$ dan probabilidades de error muy bajas, aunque con el inconveniente de que el impacto visual cada vez es mayor.

Basándonos en estos datos y en las pruebas prácticas, la conclusión es que es conveniente elegir valores de α a partir de 0.1, según la robustez que requiera la aplicación en concreto.

4.2.6. Pruebas realizadas

Una vez explicado el algoritmo de watermarking, el paso siguiente es realizar una serie de pruebas para verificar que efectivamente cumple con los requisitos fijados.

Los autores del artículo utilizaron la imagen estándar *"Fishing Boat"* para su serie de tests. Para las pruebas de este proyecto se utilizará una imagen estándar diferente, de manera que podamos verificar que el método funciona no sólo en los tests efectuados por autores con esa imagen en concreto, sino que también funciona en otros imágenes diferentes. Utilizar una imagen diferente también nos permite obtener nuestros propios resultados y confrontarlos con los obtenidos por los autores.

La imagen que utilizaremos (Figura 4.4) será *"Lena"*, obtenida de la base de datos de la University of Southern California [2], de 512x512 pixels, y en escala de grises.

Figura 4.4: Imagen original *"Lena"*.

Marcaremos la imagen con $\alpha = 0.2; L = M = 16000; R = 9$, con un watermark (enmascarado visualmente) al que llamaremos *"correcto"*, y crearemos gráficas de la salida del detector en presencia del watermark correcto, y tam-

bién cuando el watermark de entrada no se corresponde con el insertado en la imagen.

Para evaluar el comportamiento del detector utilizaremos un conjunto de 1000 watermarks, uno de los cuales será el correcto, y los restantes se generarán de forma aleatoria.

La línea horizontal de puntos y rayas que aparece en las figuras de la salida del detector representa el umbral T_z.

En concreto, evaluaremos el funcionamiento del método para los siguientes ataques:

- Compresión JPEG.
- Filtrado con filtros medianos (*median filters.*)
- Ecualización y "estiramiento"(*stretching*) del histograma.
- Adición de ruido gaussiano.
- *Dithering.*
- Cambio de escala.
- Recortes (*cropping.*)
- Inserción de múltiples marcas.

También veremos algunas de las limitaciones del método (que en la versión mejorada dejan de serlo), como son:

- No robusto frente a rotaciones.
- No robusto frente a translaciones.

Enmascaramiento visual

Para conseguir el enmascaramiento visual utilizaremos el método consistente en calcular una función que estima el grado de texturización de diferentes zonas de la imagen.

Para conocer el grado de texturización en cada pixel de la imagen, se considera un bloque de $R \times R$ pixels (en esta serie de pruebas: 9×9), y se calcula la función $\frac{\sigma^2}{1+\sigma^2}$ sobre un entorno del punto central, que es precisamente el bloque de $R \times R$ pixels.

Este cálculo se efectúa para todos los puntos de la imagen original, de manera que obtenemos una serie de valores que nos indican el grado de texturización en cada uno de los puntos de la imagen original.

Si reescalamos estos valores entre 0 y 255 (suponiendo que este sea el rango de la imagen), y los asociamos a valores de luminancia, entonces generamos una imagen que nos informa del grado de texturización en las diferentes zonas de la imagen original, de tal manera que las zonas con mayor brillo son las que tienen una mayor texturización (y por lo tanto enmascaran mucho el watermark), y cuanto más oscuras, menos textura (y más visible el watermark). En la Figura 4.5 se puede ver la máscara generada a partir de la imagen de la Figura 4.4.

Este es el tipo de enmascaramiento visual del watermark que se ha aplicado en todas las pruebas del método, tanto en el original, como en el mejorado.

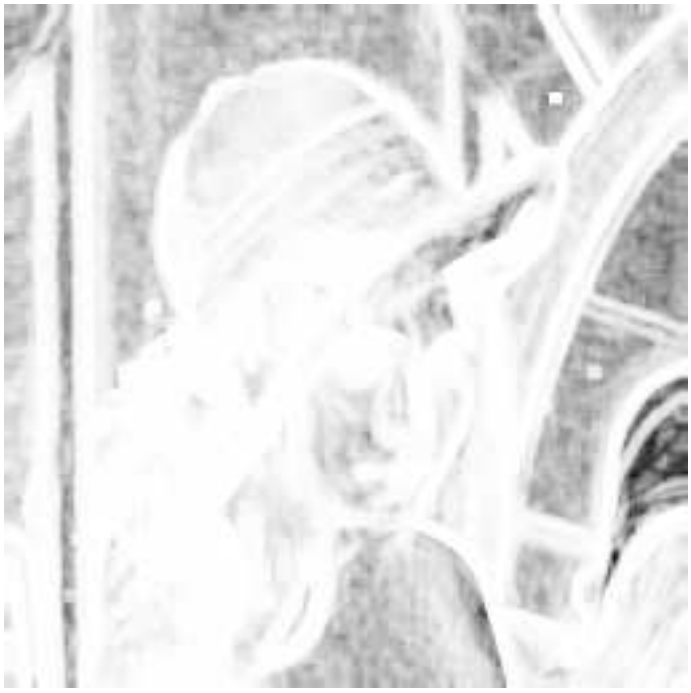

Figura 4.5: Máscara de la imagen original *"Lena"*.

4.2.7. Prueba de robustez : *Sin ataques*

Como primera prueba de evaluación del detector, vamos a marcar la imagen original *Lena* con los parámetros $\alpha = 0.2; L = M = 16000; R = 9$ (al igual que en los demás tests), y veremos cuál es la respuesta del detector frente a posibles watermarks que no se corresponden con el insertado en la imagen, y frente al único que sí se corresponde, de un total de 1000 candidatos.

En todas las pruebas que vamos a efectuar, el candidato número 500 (el que ocupa la posición central en la gráfica) será el que se ha insertado en la imagen, y los demás se generarán de forma aleatoria.

En este caso, el detector fijó el umbral de detección en $T_z = 1.1951$, y su respuesta fue la que se puede ver en la Figura 4.6.

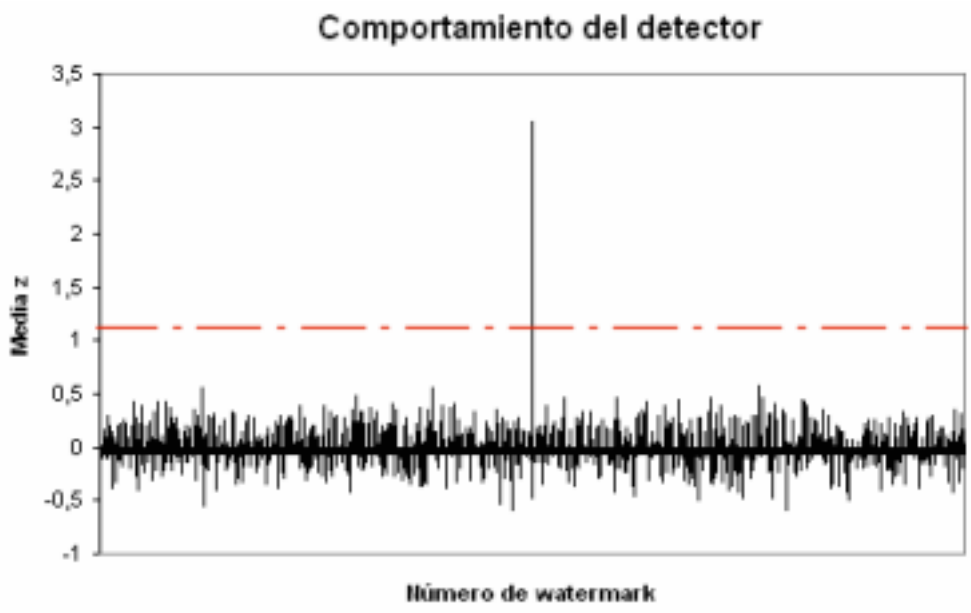

Figura 4.6: Respuesta del detector a la imagen marcada sin ataques.

Como se puede ver, el detector funcionó de manera correcta, ya que la media de z que calculó para los watermarks candidatos que no se correspondían con el de la imagen permanece por debajo del umbral T_z, mientras que únicamente el correcto nos da una z por encima, en concreto da un valor de $z = 3.05742$.

4.2.8. Prueba de robustez : *Compresión JPEG*

El algoritmo de compresión con pérdidas JPEG implica que la imagen comprimida se ve muy afectada en las frecuencias altas, ya que son eliminadas como parte del proceso de compresión. Además, la cuantificación de los coeficientes de la transformada DCT que lleva acabo el algoritmo JPEG, implica que *todas* las componentes frecuenciales se ven afectadas, y el efecto es mayor cuanto más elevadas sean las frecuencias.

Dado que modificaciones en las frecuencias bajas son muy visibles, y en frecuencias altas son eliminadas por el algoritmo JPEG, el watermark se inserta en las frecuencias medias, por lo que es resistente a este tipo de compresión.

También hay que tener en cuenta otro aspecto, y es el factor de compresión, dado que al ser un algoritmo de compresión con pérdidas, cuanto más se comprime la imagen, más información perdemos, y esto lógicamente también afecta a la información de la marca, ya que es parte de la imagen.

Los autores del artículo afirman que el método es resistente siempre y cuando el factor de compresión[9] se mantenga por debajo de 69.

Por lo tanto, evaluaremos el detector en presencia de una imagen marcada comprimida con el algoritmo JPEG con un factor de compresión de 69.

En este caso, el detector fijó el umbral de detección en $T_z = 1.20446$, y su respuesta fue la que se puede ver en la Figura 4.7.

El detector funcionó de manera correcta. Media del watermark que se corresponde con el de la imagen: $z = 1.87946$.

[9] A mayor factor de compresión, menor calidad. El rango del factor de compresión comprende desde 1 (no compresión) hasta 100 (imagen muy degradada visualmente).

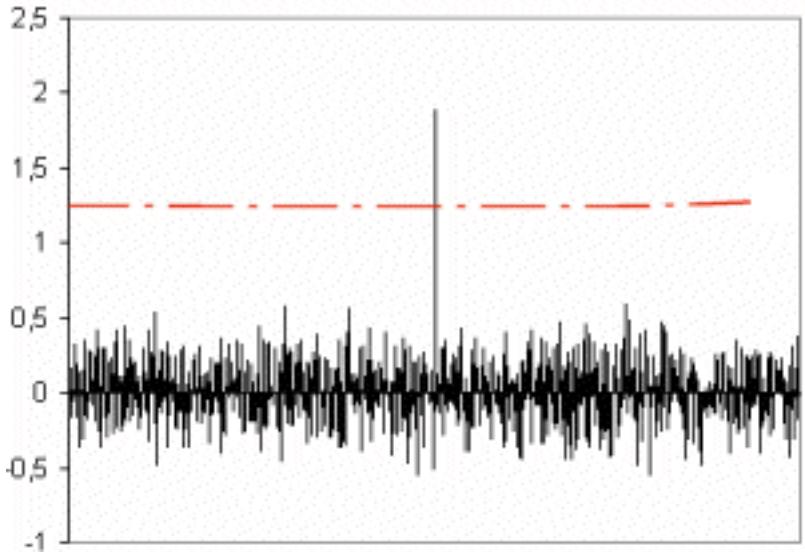

Figura 4.7: Respuesta del detector en presencia de compresión JPEG.

4.2.9. Prueba de robustez : *Filtro mediano de apertura 5*

Los autores del artículo afirman que el método es resistente a filtros medianos cuya apertura sea igual o menor que cinco pixels.

Evaluaremos el detector en presencia de una imagen marcada y filtrada con un filtro de apertura cinco.

En este caso, el detector fijó el umbral de detección en $T_z = 0.303019$, y su respuesta fue la que se puede ver en la Figura 4.8.

El detector funcionó de manera correcta. Media del watermark que se corresponde con el de la imagen: $z = 0.440917$.

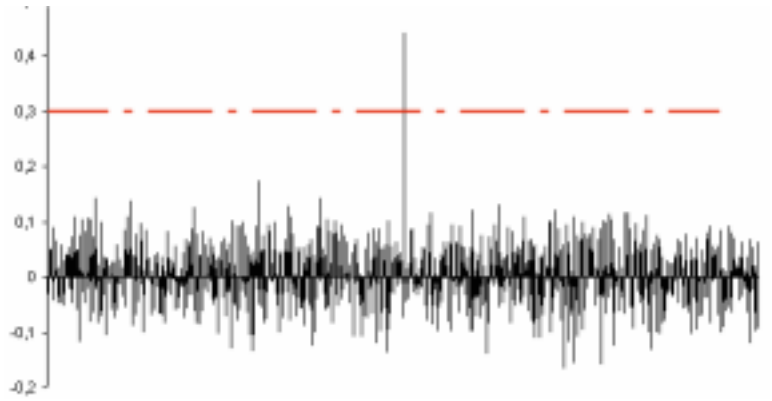

Figura 4.8: Respuesta del detector en presencia del filtro mediano.

4.2.10. Prueba de robustez : *Filtrado paso-bajo*

La técnica del filtrado paso–bajo consiste en poner a cero aquellos coeficientes a frecuencias superiores a la *frecuencia de corte* del filtro, y mantener el valor de los restantes coeficientes cuya frecuencia esté por debajo.

Por lo tanto, es evidente que el detector será capaz de detectar el watermark insertado en la imagen sólo en aquellos casos en los que la frecuencia de corte del filtro sea tal que los coeficientes del watermark no se vean afectados.

A medida que se vayan poniendo a cero los coeficientes que contienen información del watermark, entonces la detección será cada vez peor, hasta llegar a un punto en el que la media z esté por debajo del umbral T_z, y entonces el detector pasará a decir que la imagen no está marcada con ese watermark.

Para evaluar el comportamiento del sistema frente al filtrado paso–bajo, se programó un filtro que siguiendo el orden en zig–zag no consideró los 16000 (L) primeros coeficientes ni los 8000 siguientes ($M/2$), para luego poner a cero los 8000 ($M/2$) siguientes. Esta operación dañó el 50 % del watermark.

En la Figura 4.9 se puede ver la degradación visual que ha sufrido la imagen.

En este caso, el detector fijó el umbral de detección en $T_z = 0.702498$, y su respuesta fue la que se puede ver en la Figura 4.10.

El detector funcionó de manera correcta. Media del watermark que se corresponde con el de la imagen: $z = 1.80124$.

Figura 4.9: Imagen marcada, ataque : filtrado paso-bajo.

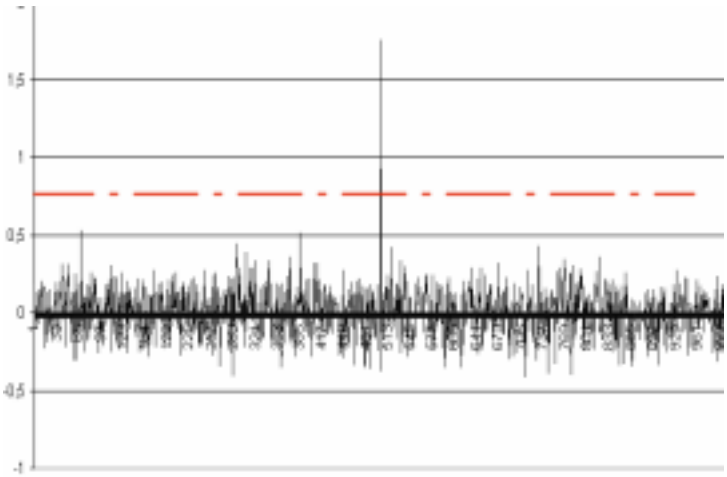

Figura 4.10: Respuesta del detector después del filtrado paso-bajo.

4.2.11. Prueba de robustez : *Ecualización del histograma*

La ecualización del histograma, aún siendo un posible ataque, no degrada el funcionamiento del método, sino que por el contrario, los resultados experimentales muestran que el pico que se produce con el watermark correcto es aún mayor cuando se aplica este procesamiento.

En la Figura 4.11 se puede ver la imagen después de la ecualización del histograma.

Figura 4.11: Imagen marcada, ataque : ecualización del histograma.

En este caso, el detector fijó el umbral de detección en $T_z = 1.74284$, y su respuesta fue la que se puede ver en Figura 4.12.

El detector funcionó de manera correcta. Media del watermark que se corresponde con el de la imagen: $z = 4.64467$.

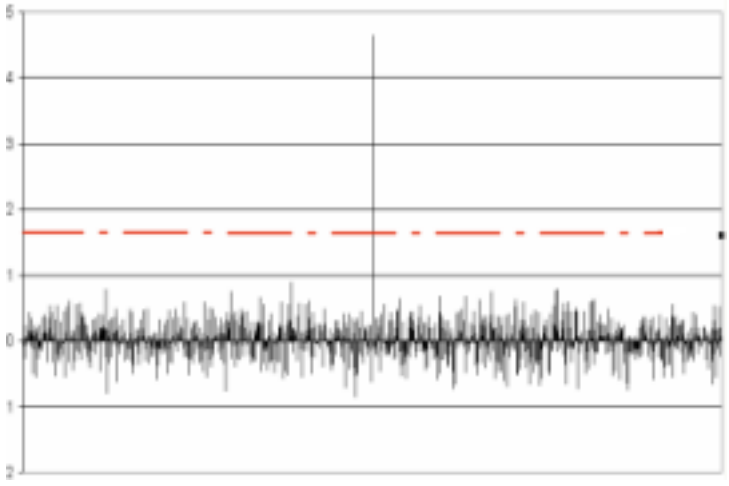

Figura 4.12: Respuesta del detector en presencia de la ecualización del histograma.

4.2.12. Prueba de robustez : *Stretching del histograma*

El *stretching*[10] del histograma es el proceso mediante el cual se modifica el histograma original para que las frecuencias de aparición de cada nivel de gris queden repartidas por todo el rango de valores que se permite para cada pixel, por ejemplo, entre 0 y 255.

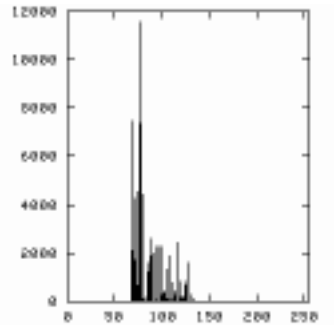

Figura 4.13: Histograma original.

En la Figura 4.13 se puede ver el histograma de una imagen, en la que los valores del histograma se reparten la mayoría entre 75 y 125.

[10]Lo podríamos traducir como "alargamiento"del histograma.

Después de aplicar el histograma, estos valores se reescalan entre 0 y 255, de manera que el valor 75 anterior ahora pasa a estar en la posición 0, y el 125 pasa a ser el máximo, es decir, se mueve hasta 255, y los valores intermedios se reescalan para que ocupen sus nuevas posiciones. En la Figura 4.14 se puede ver el histograma después de aplicar el stretching, y como se ve, es como si se hubiera "alargado"(stretching) para ocupar todo el rango $[0, 255]$, y de ahí su nombre.

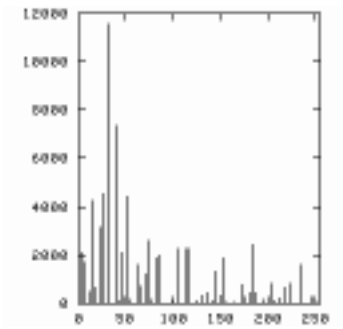

Figura 4.14: Histograma después del stretching.

Con esto se consigue mejorar el contraste de la imagen, ya que la luminancia no se concentra en una determinada zona del histograma, sino que queda repartida.

Al igual que ocurría en el caso anterior con la ecualización, el stretching del histograma no degrada el funcionamiento del método, sino que los resultados son mejores cuando se aplica este procesamiento.

En la Figura 4.15 se puede ver la imagen después del stretching.

En este caso, el detector fijó el umbral de detección en $T_z = 1.1951$, y su respuesta fue la que se puede ver en la Figura 4.16.

El detector funcionó de manera correcta. Media del watermark que se corresponde con el de la imagen: $z = 3.05742$.

Figura 4.15: Imagen marcada, ataque : stretching del histograma.

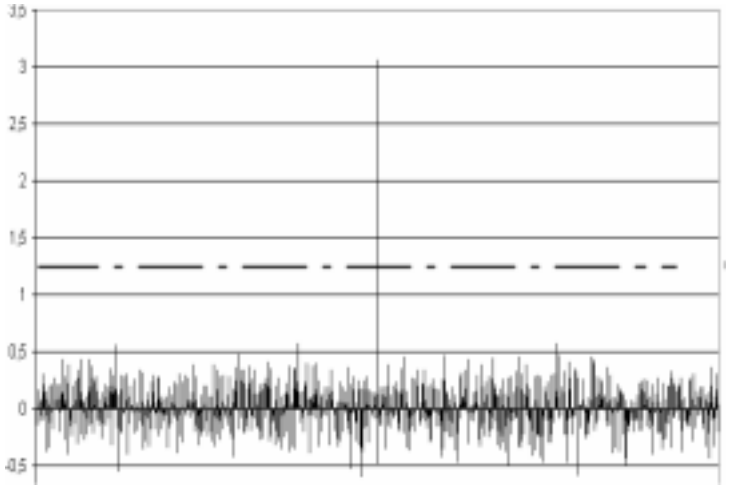

Figura 4.16: Respuesta del detector en presencia de stretching en el histograma.

4.2.13. Prueba de robustez : *Adición de ruido gaussiano*

Para generar las muestras de ruido gaussiano, se programó el algoritmo de Muller-Box [6] , que nos da muestras de una variable aleatoria gaussiana normal (media cero y varianza unidad).

Dado que queremos sumar ruido gaussiano de cualquier media y varianza, tenemos que transformar la variable $N(0,1)$ en una gaussiana $Z(\mu,\sigma^2)$.

Existe una propiedad mediante la cual se puede llevar a cabo esta transformación : $Z = \mu + \sigma N(0,1)$.

Por lo tanto, lo que se ha hecho es, para cada pixel de la imagen original, se ha creado una muestra de ruido gaussiano mediante el algoritmo de Muller-Box. Esta muestra se ha convertido en el resultado de una realización del proceso que genera la variable $Z(\mu,\sigma^2)$ mediante la fórmula anterior, y se ha sumado al valor del pixel.

El rango de posibles valores generados por una variable aleatoria gaussiana abarca todos los números reales, por lo que, evidentemente, al sumarlos al valor del pixel, éste puede quedar fuera de su rango 0–255.

Existen dos maneras de evitarlo. Una consiste en truncar aquellos valores que estén fuera del rango, y la otra consiste en reescalar todos los valores de nuevo dentro del rango 0–255.

Se ha decidido utilizar la primera técnica, es decir, truncar los valores.

El motivo, es que consideramos que truncar los valores es más cercano a lo que ocurre en la realidad cuando una imagen queda sometida a ruido, es decir, que cuando el ruido es negativo, el pixel perderá brillo hasta llegar a un valor de cero, ya que valores negativos no tienen sentido físico referido a la luminancia del pixel.

De la misma forma, si el ruido hace aumentar el brillo del pixel, lo hará de forma que su valor de luminancia alcance el máximo que permite su rango de valores digitales (por ejemplo, rango 0–255), pero una vez alcanzado el máximo, ya no se observarán cambios.

La alternativa consistente en reescalar todos los valores de la imagen, implica que todos los valores de la imagen quedan afectados por el hecho de que un solo pixel afectado por el ruido haya quedado fuera del rango 0–255, lo cual no es realista.

Por ejemplo, supongamos que una imagen está compuesta por pixels de valor 128, con un rango 0–255.

Si el ruido afecta a uno de los pixels, y hace que su valor pase a ser 500, entonces después de reescalar la imagen nos encontraríamos con que el pixel

afectado por el ruido pasa a tener valor 255 (el máximo), pero al mismo tiempo los demás pixels, que no se habían visto afectados por el ruido, pasan a valer 65.

Si no se hace el reescalado, sino que se truncan los valores, entonces el pixel que se vio afectado pasaría a valer 255, y los demás mantendrían su valor 128, lo cual es más cercano a la realidad.

Para este test, hemos insertado ruido gaussiano de media $\mu = 0$ en la imagen marcada, con varianza desde $\sigma^2 = 0$ (ausencia de ruido) hasta $\sigma^2 = 2000$.

En la Figura 4.17 se ha representado la evolución de las variables z (la media calculada por el detector) y T_z (el umbral fijado por el detector), para diferentes varianzas de ruido.

La curva superior corresponde a la media z, y la inferior al umbral T_z.

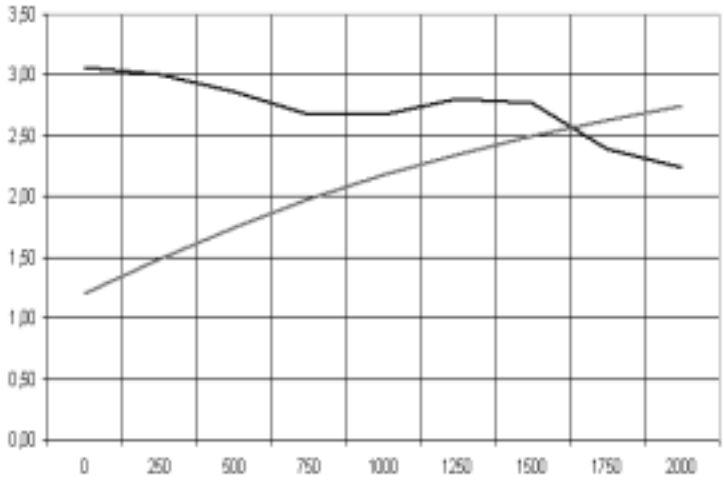

Figura 4.17: Respuesta del detector en presencia de ruido gaussiano.

Como se puede observar, el detector es capaz de reconocer la firma de autentificación mientras la varianza del ruido esté por debajo de aproximadamente $\sigma^2 = 1500$.

Es un buen resultado, porque un ruido gaussiano con $\sigma^2 = 1500$ degrada mucho la imagen (ver Figura 4.18), por lo cual tenemos que concluir que el comportamiento del detector frente al ruido es excelente.

En este caso, el detector fijó el umbral de detección en $T_z = 2.50199$, y la media z fue $z = 2.77176$.

Figura 4.18: Imagen marcada, ataque : adición de ruido gaussiano con $\mu = 0$ y $\sigma^2 = 1500$.

4.2.14. Prueba de robustez : *Dithering*

El dithering es una técnica que permite simular colores que no están presentes en la paleta de la imagen, de manera que un observador tenga la sensación de ver los colores, cuando en realidad la imagen no los tiene.

Esto se consigue de diversas formas, y una de las más conocidas consiste en utilizar los colores de la paleta de la cual se dispone, de manera que colocando cerca pixels de diferentes colores, al observar la imagen se tenga la sensación de estar viendo un color nuevo.

Un parámetro que se suele tener muy en cuenta en este proceso es el error cometido entre los pixels utilizados, y los de la imagen original, ya que si este error sobrepasa ciertos límites, entonces la calidad de la imagen se empieza a ver afectada.

Para esta prueba se redujo la paleta de la imagen original, que utiliza 225 colores, hasta obtener una paleta de únicamente dos colores (totalmente negro -valor 0- y totalmente blanco -valor 255-), utilizando difusión de error mediante el algoritmo de Floyd-Steinberg.

En la Figura 4.19 se puede ver la imagen después del dithering:

En este caso, el detector fijó el umbral de detección en $T_z = 2.39589$, y su respuesta fue la que se puede ver en la Figura 4.20.

El detector funcionó de manera correcta. Media del watermark que se corresponde con el de la imagen: $z = 5.14927$.

Figura 4.19: Imagen marcada, ataque : dithering.

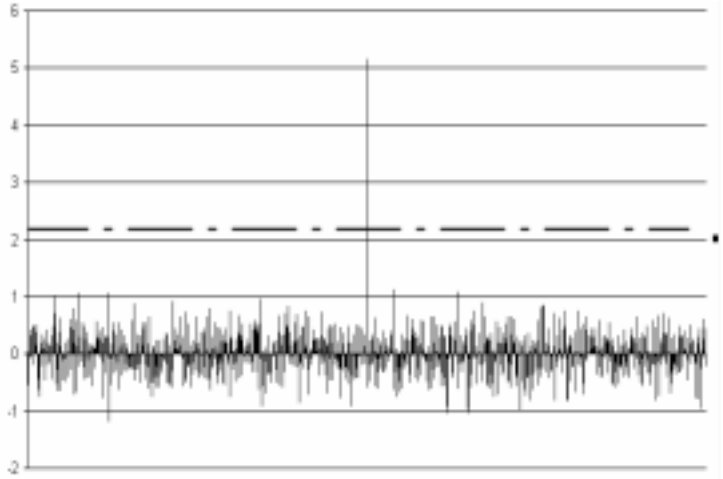

Figura 4.20: Respuesta del detector después del dithering.

4.2.15. Prueba de robustez : *Cambio de escala*

Los cambios de escala no afectan a la posición relativa de los coeficientes del espectro de la DCT, por lo que en principio, el método no se ve afectado por el reescalado.

Cuando la imagen se amplía, el tamaño de la imagen aumenta, lo cual se traduce en que el espectro de la imagen es mayor, es decir, que los coeficientes que estaban presentes siguen en la misma posición, pero además, la imagen ahora puede contener frecuencias mayores, que quedan puestas a cero en la nueva imagen, ya que la original no las contenía.

Dado que al aplicar el método a la imagen ampliada se van a buscar los coeficientes marcados en la posición correcta (ya que su posición no ha cambiado), se detectará el watermark.

Así pues, siempre que ampliemos la imagen, el detector funcionará correctamente.

En el caso de zoom negativo, es decir, que en vez de ampliar la imagen, la estemos reduciendo, entonces se produce una fenómeno que antes no ocurría, y es el de *aliasing*, que consiste en que por efecto del zoom negativo, las frecuencias altas comienzan a ocupar el lugar de las frecuencias bajas, tanto más cuanto mayor sea el zoom negativo.

Dado que el watermark se inserta en frecuencias medias, un zoom negativo que no sobrepase aproximadamente el 50 % del tamaño original no afectará a la detección por el motivo anterior, es decir, que los coeficientes marcados ocupan la misma posición, por lo cual el detector seguirá funcionando correctamente.

Si se llega a producir el aliasing, entonces los coeficientes marcados se verán afectados por las altas frecuencias, y entonces la detección será tanto peor cuando más aliasing se produzca, es decir, cuanto mayor sea el zoom negativo aplicado.

Para esta prueba hemos reducido al 50 % la imagen marcada, y el detector fijó el umbral de detección en $T_z = 0.528574$, y su respuesta fue la que ver en la Figura 4.21.

El detector funcionó de manera correcta. Media del watermark que se corresponde con el de la imagen: $z = 1.30826$.

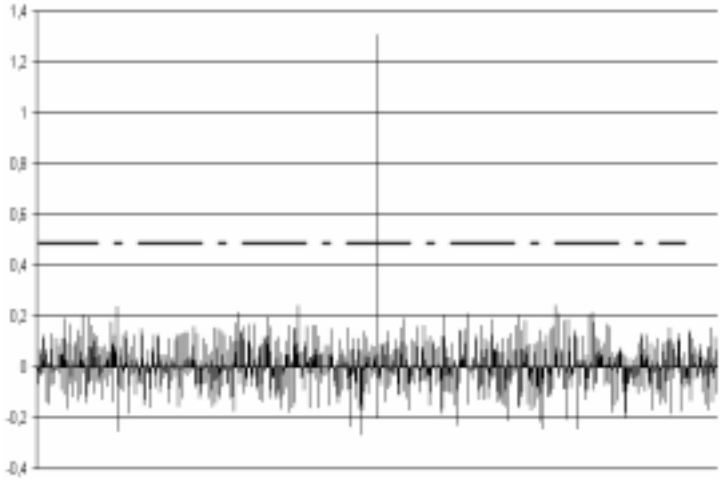

Figura 4.21: Respuesta del detector después del zoom al 50 %.

4.2.16. Prueba de robustez : *Recortes (cropping)*

Aunque el recorte en la imagen afecte a zonas de la imagen en las que se ha insertado el watermark (recordemos que la inserción de frecuencias en la imagen la afecta de manera global), si la secuencia watermark tiene la longitud suficiente, la subimagen puede tener la suficiente información como para que el detector pueda determinar que el watermark que se está comprobando sea el que está contenido en la imagen.

Esto es así porque a la hora de decidir el umbral de detección no se utilizó el valor que se obtendría en ausencia de ataques, sino que se utilizó un valor tres veces por debajo del máximo, por lo que el método es resistente a este tipo de ataques, aunque evidentemente, el valor de z que calcula el detector será menor que en el caso de que no se hubiera hecho ningún recorte.

El recorte en la imagen se hace de manera que la subimagen que queda siga ocupando la misma posición que en la imagen original, ya que el método en principio no es resistente a translaciones (aunque en la sección 4.3 el método se mejora para que sea robusto a las mismas).

Para esta prueba se han puesto a cero algunos pixels de la imagen, de manera arbitraria, tal y como se puede ver en la Figura 4.22.

En este caso, el detector fijó el umbral de detección en $T_z = 1.38445$, y su respuesta fue la que se puede ver en la Figura 4.23.

El detector funcionó de manera correcta. Media del watermark que se co-

Figura 4.22: Imagen marcada, ataque : cropping.

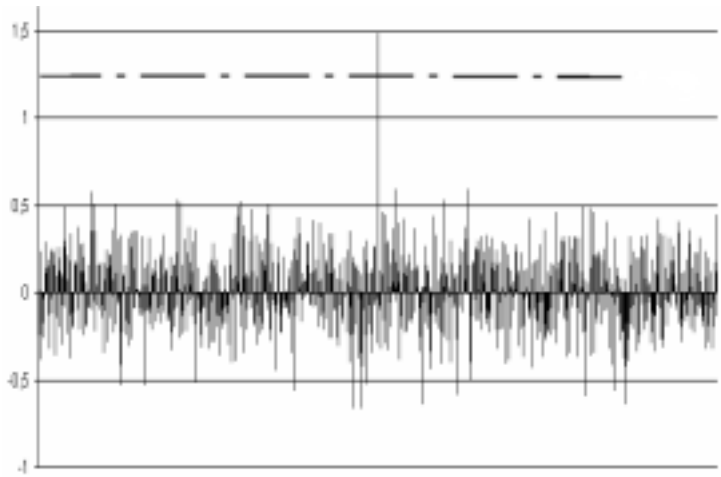

Figura 4.23: Respuesta del detector después del cropping.

rresponde con el de la imagen: $z = 1.48073$.

Como se puede observar, el detector ha determinado correctamente que el watermark que se comprueba es que el que está presente en la imagen, aunque esta vez el parámetro z ha quedado muy cerca del umbral, lo que indica que si hubiéramos continuado eliminando información de la imagen, el detector hubiera pasado a decidir que el watermark comprobado no está presente.

4.2.17. Prueba de robustez : *Inserción de múltiples marcas*

La inserción de múltiples marcas es uno de los procedimientos habituales que se suelen intentar a la hora de romper un sistema de watermarking.

El ataque consiste tomar en una imagen marcada, y aplicar el algoritmo de marcado múltiples veces, y con diferentes marcas, con la esperanza de que al terminar el proceso, la imagen no haya quedado demasiado afectada visualmente, y sobre todo, que el detector no sea capaz de detectar el watermark inicial sobre la imagen resultante.

Para este sistema en concreto, la inserción de múltiples marcas sobre una imagen ya marcada no provoca que el detector deje de reconocer el watermark inicial, sino que el resultado es que el sistema los detecta todos, es decir, que genera un positivo siempre que se esté comprobando uno de los posibles watermarks con los que fue marcada la imagen, y genera una condición de detección negativa cuando el watermark que se comprueba no se corresponde con ninguno de los presentes en la imagen.

Los watermarks pueden compartir la misma posición (parámetros L y M) en el espectro de la DCT, aunque el estudio de cuántos watermarks pueden insertarse de manera que el método sea resistente a este ataque queda pendiente, y no es el objetivo de este proyecto.

Para esta prueba, se ha tomado la imagen marcada (Figura 4.24) que hemos utilizado en todos los tests anteriores, y la hemos marcado con cinco watermarks adicionales diferentes, que quedan insertados en la imagen simultáneamente.

El primer y más evidente efecto que se puede observar, es que las marcas de parámetro $\alpha = 0.2$ que en principio eran invisibles, por el hecho de insertar varios watermarks, ahora afectan visualmente a la imagen, tanto más cuantos más watermarks albergue la imagen.

Esto que podría a priori parecer un inconveniente, en realidad es una ventaja, porque un atacante que intente eliminar la marca de autentificación se encontrará con que para conseguir quitar la marca debe insertar muchos watermarks, pero esto implica que la imagen queda muy afectada visualmente.

Si aplicamos el detector a la imagen con las seis marcas simultáneas, el umbral de detección queda fijado en $T_z = 1.19743$.

La medias obtenidas al comprobar los diferentes watermarks fueron respectivamente $z_{inicial} = 4.45144$, $z_1 = 4.70437$, $z_2 = 3.49854$, $z_3 = 3.27109$, $z_4 = 3.31867$, $z_5 = 3.68932$, es decir, todos dieron como resultado detección positiva.

La respuesta del detector a la prueba de los mil watermarks con el central correcto es la que se puede ver en la Figura 4.25.

Figura 4.24: Imagen marcada, ataque : inserción de varias marcas.

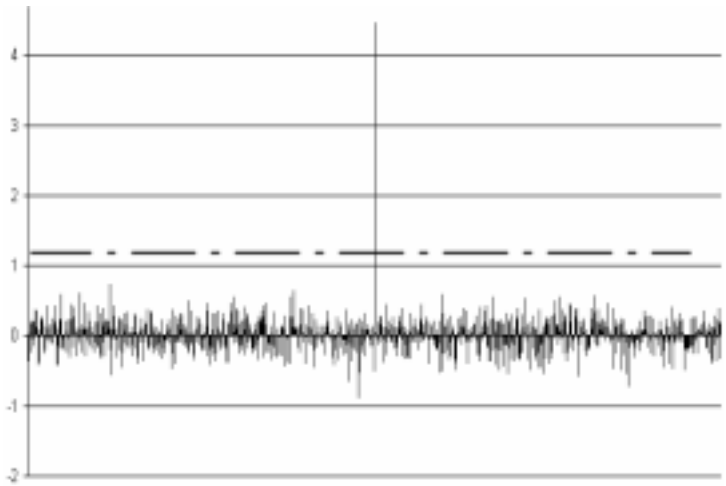

Figura 4.25: Respuesta del detector después de insertar 5 nuevos watermarks.

Los watermarks correctos dan una media z muy por encima del umbral, y al mismo tiempo, la imagen se degrada rápidamente a medida que vamos insertando nuevas marcas, por lo que la conclusión es que el sistema es robusto frente a este ataque.

4.2.18. Prueba de robustez : *Varios ataques simultáneos*

Las pruebas que se han hecho hasta ahora han servido para comprobar la robustez del método a una serie de ataques aplicados por separado, aunque en la práctica es de esperar que las imágenes analizadas hayan sufrido no uno, sino una combinación de varios de los ataques presentados anteriormente.

Para esta prueba vamos a proceder a aplicar a la imagen marcada los siguientes ataques, de forma simultánea:

1. Compresión JPEG con factor de compresión igual a 69.
2. Ecualización del histograma.
3. Stretching del histograma.
4. Filtrado paso bajo.
5. Filtrado mediante filtro mediano de apertura 5.

En la Figura 4.26 se puede ver la imagen resultante de aplicar estos ataques.

Figura 4.26: Imagen marcada, ataque : combinación de varios ataques.

En este caso, el detector fijó el umbral de detección en $T_z = 0.742525$, y su respuesta fue la que se puede ver en la Figura 4.27.

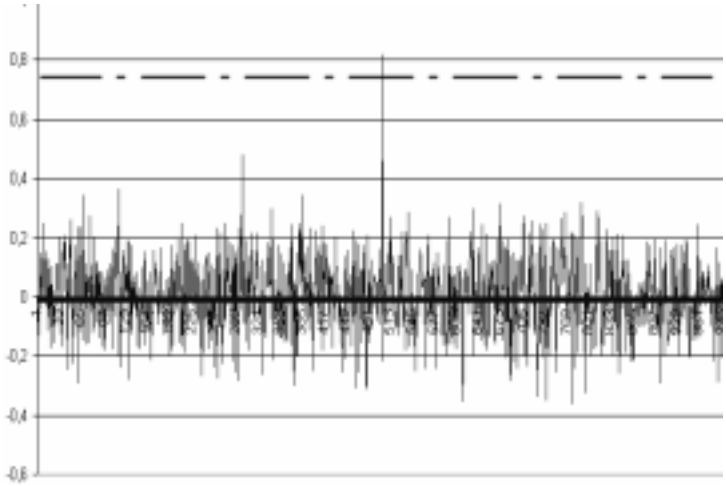

Figura 4.27: Respuesta del detector al ataque combinado.

El detector funcionó de manera correcta. Media del watermark que se corresponde con el de la imagen: $z = 0.81839$.

Como se puede observar, el detector ha determinado correctamente que el watermark comprobado es que el que está presente en la imagen, aunque esta vez el parámetro z ha quedado muy cerca del umbral, lo que indica que si hubiéramos continuado combinando ataques, además de conseguir degradar visualmente la imagen, también hubiéramos conseguido que el detector pasara a decidir que el watermark correcto no está presente en la imagen, aunque a costa de degradar la calidad por debajo del límite admisible.

4.2.19. Limitaciones del método

En las pruebas efectuadas al método se encontraron dos formas diferentes de romperlo fácilmente, es decir, un atacante puede generar una imagen muy parecida visualmente a la marcada, pero en la cual el detector no es capaz de reconocer la marca de autentificación.

Uno de estos ataques consiste en hacer una translación de la imagen de más de un pixel.

En la Figura 4.28 se puede observar la imagen marcada desplazada dos pixels en el sentido positivo del eje horizontal, con lo que las los pixels de las dos primeras columnas se han sustituido por los de la tercera columna.

Figura 4.28: Imagen marcada desplazada dos pixels.

Como se observa, la imagen es visualmente igual a la marcada, y si no fuera por el desplazamiento, sería exactamente la misma imagen, pero en este caso el detector genera una salida $z = 0.4532$, con un umbral $T_z = 1.22095$, con lo cual decide que la imagen no se corresponde con el watermark correcto que lleva insertado.

Esto se debe a que el método está basado en la transformada DCT, que

no es invariante frente a translaciones ni rotaciones, lo cual significa que estas operaciones modifican lo suficiente los coeficientes como para que, al calcular la media z, el resultado sea una condición de detección negativa.

Si rotamos la imagen medio grado a la izquierda obtenemos la Figura 4.29:

Figura 4.29: Imagen rotada 0.5 grados a la izquierda.

La imagen es muy similar a la marcada, pero sin embargo la media calculada por el detector para el watermark correcto fue de $z = 0.272515$ y el umbral fue fijado en $T_z = 1.13394$, con lo que el detector decidió que la marca no estaba presente en la imagen.

El motivo es que la transformada DCT no es invariante frente a la rotación, por lo que al aplicar uno de los dos efectos anteriores a la imagen marcada, el detector ya no es capaz de encontrar la marca.

Esto hace que para un atacante sea muy fácil romper el método, ya que puede generar nuevas imágenes muy similares a la marcada en las que la marca no se puede detectar, por lo que para este proyecto se propuso encontrar una variación que permitiera mejorarlo, y es lo que se presenta en la sección 4.3.

4.3. Método mejorado

4.3.1. Introducción

El método que hemos estudiado ha demostrado ser resistente a una gran variedad de ataques, pero aún así, es necesario mejorarlo para que sea robusto frente a dos ataques más, en concreto, las translaciones y las rotaciones.

Como se ha comprobado en las pruebas anteriores, para un atacante es sencillo romper el sistema, ya que conociendo a priori sus dos debilidades puede generar imágenes muy parecidas a las marcadas, pero en las que el detector no es capaz de reconocer la marca de autentificación.

El motivo de esta debilidad en el método se explica por la utilización de la transformada DCT, que no es invariante frente a rotaciones ni translaciones.

Sin embargo, la DFT sí es invariante frente a estas dos operaciones, dado que frente a una rotación, los módulos del espectro de la DFT también rotan, y frente a una translación, los módulos del espectro se conservan.

Matemáticamente [11]:

- Si se aplica a $f(x, y)$ una rotación con un ángulo θ en el plano XY:

  $$f_\theta(x, y) = f(x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta)$$

  entonces su transformada de Fourier también queda sometida a una rotación con el mismo ángulo:

  $$\hat{f}_\theta(\xi_1, \xi_2) = \hat{f}(\xi_1\cos\theta - \xi_2\sin\theta, \xi_1\sin\theta + \xi_2\cos\theta)$$

- Si aplicamos una translación a una señal en el dominio espacial, en el dominio frecuencial la función transformada queda multiplicada por una exponencial (módulo unidad), por lo que no se observan cambios en el módulo:

  $$f(t-u) \longleftrightarrow e^{-i2\pi\xi u}\hat{f}(\xi)$$

Aún así, el cambiar de transformada implica varios cambios en el método, que serán los que se van a explicar a continuación.

Finalmente, se efectuará una serie de tests con diferentes imágenes para comprobar que el método es resistente a los ataques, y que por lo tanto es válido.

[11] Fuente : *A. Bruce Carlson, Communication Systems, McGraw–Hill, 1986. [16]*

4.3.2. Información en los módulos

La primera diferencia significativa respecto del método anterior, en el que se utilizaba la DCT, es que la DFT de una señal real no es necesariamente real, lo cual significa que los coeficientes ya no serán números reales, como en el caso de la DCT, sino que ahora serán números complejos.

Esto implica que a la hora de modificar los coeficientes, ahora se puede elegir entre varios parámetros, como el módulo, su fase, o combinaciones de los dos anteriores.

Como se ha explicado en la introducción, utilizar el módulo permite la invariancia frente a rotaciones y translaciones, por lo que será el parámetro que modificaremos.

Existe un resultado clásico en la teoría del procesamiento de imágenes, que nos dice que a la hora de reconstruir una imagen mediante la IDFT, visualmente tiene más importancia la información contenida en la fase que la que está en los módulos del espectro.

Así pues, lo que se hará es insertar la información de watermarking en los módulos, y sin cambiar la fase del coeficiente que se está marcando, y para ello se seguirán los siguientes pasos:

1. Calcular el módulo M del coeficiente C_i.
2. Normalizar el número complejo C_i dividiéndolo por su módulo M.
3. Multiplicar C_i por el nuevo módulo M'.
4. Llegados a este punto, el coeficiente C_i mantiene la fase inicial, pero pasa a tener módulo M'.

De ahora en adelante, cuando hablemos de *cambiar el módulo* de un coeficiente, nos estaremos refiriendo a este algoritmo.

4.3.3. Inserción del watermark

El watermark $X = \{x_1, x_2, ..., x_M\}$ consiste en una secuencia pseudo-aleatoria de longitud M, en la que cada x_i es un número pseudo–aleatorio real con distribución de Bernoulli, de media cero y varianza unidad. La definición del watermark es exactamente la misma que la que se dio en el método original, y todo lo explicado sobre la secuencia watermark es aplicable aquí.

Para insertar el watermark en la imagen I de dimensiones $N \times N$, y en nivel de gris, primero se calcula su DCT, y los coeficientes se reordenan según el orden de un barrido en *zig–zag*.

En realidad no es estrictamente necesario seguir el orden en zig–zag, sino que lo verdaderamente importante es que los coeficientes que se marquen estén situados en frecuencias medias, y que el orden en el que se marcan se pueda reproducir a la hora de recuperarlos. En nuestro caso, hemos marcado únicamente las frecuencias positivas, aunque evidentemente también se podrían haber marcado coeficientes que tengan una componente frecuencial negativa, u otra positiva.

Hay muchas maneras de marcar los coeficientes, y quizá una buena manera de elegir un método para generar el orden en el que se marcan sea a partir de la secuencia watermark, es decir, utilizar una parte de los bits de información del watermark para determinar un determinado orden de lectura de los coeficientes, o un método en concreto.

Para este proyecto en particular, se ha tomado el orden en zig–zag en las frecuencias positivas, aunque futuras mejoras de esta propuesta podrían ir encaminadas a generar parámetros de configuración de los algoritmos de inserción/detección a partir de la secuencia de watermark, pero esto es algo que excede los objetivos de este proyecto.

La marca (el vector pseudo–aleatorio X) se insertará desde el coeficiente $L+1$ hasta el $M+L$, según el orden en zig–zag del espectro de la DFT, en el que los L primeros coeficientes no se tienen en consideración, para conseguir que la marca sea visualmente imperceptible.

Los módulos de estos coeficientes originales constituyen el vector

$$T = \{t_{L+1}, t_{L+2}, ..., t_{L+M}\}.$$

La primera diferencia respecto del método original, es que el vector T ya no almacena los coeficientes, sino que ahora almacenamos los módulos.

En el método original, los elementos del vector T se modificaban de la siguiente manera (ecuación (4.3)) para formar el vector T' :

$$t'_{L+i} = t_{L+i} + \alpha |t_{L+i}| x_i \qquad , \qquad i = 1, 2, ..., M \tag{4.11}$$

Pero esta ecuación no la vamos a poder aplicar directamente en el nuevo método, sino que habrá que introducir algunos cambios.

A continuación, se va a explicar por qué esta ecuación ha dejado de ser válida.

En el método anterior, en el caso de detección negativa teníamos que T,X,Y eran variables independientes, por lo que

$$z = \mu(TY) + \alpha\mu(|T|XY) = \mu(T)\mu(Y) + \alpha\mu(|T|)\mu(X)\mu(Y) =$$

$$= 0 + \alpha\mu(|T|)0 = 0$$

La media de las variables X e Y es cero, aunque en la práctica no toman exactamente ese valor, sino que uno muy cercano a cero, por lo que $\mu(X)\mu(Y) \approx 0$, pero dado que la media del vector T es un valor muy elevado, el producto $\mu(T)\mu(Y)$ en la práctica es diferente de cero.

El resultado es que en detección negativa ya no obtendríamos un valor de z cercano a cero, sino que dependería fuertemente del valor del módulo de los coeficientes de la imagen original en concreto, con lo cual el detector no podría diferenciar entre detección positiva y negativa.

Así pues, el motivo por el cual el detector del método original no funciona si utilizamos los módulos, es precisamente que los módulos no tienen media cero, y por lo tanto no se puede distinguir si hubo, o no, detección.

La solución a este problema consiste en utilizar una variable aleatoria auxiliar, que esté directamente relacionado con los módulos, pero que tenga media cero. Esta nueva variable será la que nos genere las componentes t_i del vector T, y que nos permitirá aplicar el método, con algunas variaciones adicionales.

De esta manera $\mu(T) = 0$ y entonces $z = 0$ cuando no hay detección.

Proposición : utilizar una variable pseudoaleatoria auxiliar.

Dada una variable M de media diferente a cero, se puede utilizar una variable auxiliar T que sea el producto $T = MR$, donde R es una variable pseudoaleatoria que puede tomar los valores $+1$ y -1 con probabilidad $\frac{1}{2}$. Dado que M y R son variables independientes, y R es de media cero, entonces T es de media cero.

Aprovecharemos este hecho para poder aplicar el método original utilizando los módulos, dado que ahora la variable T que marcamos sí que tiene media cero, y por lo tanto, el detector funcionará correctamente.

El que la variable R sea pseudoaleatoria significa que se puede reproducir la secuencia de valores que genera si proveemos al generador de números aleatorios la misma semilla (seed) tanto en el insertor como en el detector.

El detector puede reproducir la secuencia pseudoaleatoria, es decir, conoce la variable R, y por lo tanto, puede conocer el valor de la variable marcada T, dado que conoce que $T = MR$.

En nuestro caso en concreto, para generar la variable R se programó en C++ una función que con probabilidad $p = \frac{1}{2}$ genera el valor +1 y con probabilidad $1 - p = \frac{1}{2}$ genera el valor -1 (variable aleatoria de Bernoulli).

Los detalles de cómo se inserta y se detecta el watermark se explicarán en esta, y en la siguiente sección (4.3.4), respectivamente.

Como se ha explicado, ahora ya es posible aplicar la misma ecuación (4.5) del método anterior :

$$t'_i = t_i + \alpha |t_i| x_i \qquad , \qquad i = 1, 2, ..., M$$

pero con la diferencia de que ahora los t_i no son coeficientes de la DCT, sino que se forman a partir de los módulos del espectro, así:

$$t_i = r_i M_i \tag{4.12}$$

donde r_i son los valores pseudo–aleatorios que va generando la variable R, y M_i son los módulos del espectro de la imagen, leídos según el orden en zig–zag.

Si sustituimos la expresión de t_i de la ecuación (4.12) en la ecuación (4.5), entonces tenemos que:

$$\begin{aligned} t'_i = &\quad t_i + \alpha |t_i| x_i &= \\ = &\quad r_i M_i + \alpha |r_i M_i| x_i &= \\ = &\quad r_i M_i + \alpha |r_i| |M_i| x_i &= \\ = &\quad r_i M_i + \alpha M_i x_i &= \\ = &\quad M_i (r_i + \alpha x_i) \end{aligned}$$

Se ha aplicado que $|r_i| = 1$, y que $|M_i| = M_i$, para finalmente obtener la ecuación:

$$t'_i = M_i (r_i + \alpha x_i) \tag{4.13}$$

Esta ecuación nos da los valores de las componentes del vector T', pero lo que realmente nos interesa conocer es cuál es el valor de los módulos modificados que hay que utilizar para marcar la imagen.

Si aplicamos la ecuación (4.12) a los vectores T' y M', tenemos que:

$$t'_i = r_i M'_i \tag{4.14}$$

Por lo tanto, si despejamos M_i' de la ecuación (4.14), entonces obtenemos:

$$M_i' = \frac{t_i'}{r_i} \tag{4.15}$$

Esta última ecuación (4.15) nos relaciona los valores de la variable T', que es la que utilizamos de forma auxiliar para aplicar el método, con los módulos que tenemos que establecer en el espectro DFT de la imagen para hacer efectiva la marca.

Si desarrollamos esta última ecuación (4.15), sustituyendo por la expresión de t_i' que obtuvimos en la ecuación (4.14), entonces tendremos que:

$$M_i' = \frac{t_i'}{r_i} = \frac{M_i(r_i + \alpha x_i)}{r_i} = M_i\left(1 + \alpha\frac{x_i}{r_i}\right) = M_i\left(1 + \alpha x_i r_i\right) \tag{4.16}$$

ya que $\frac{1}{r_i} = r_i$ cuando $r_i \in \{-1, +1\}$.

Esta ecuación (4.16) es la que nos da las componentes del vector M' que se utiliza para marcar las imágenes. Dado el módulo original del coeficiente en cuestión, el watermark X y la variable pseudoaleatoria R, el resultado es el nuevo módulo que hay que insertar en el espectro de la imagen que se quiere marcar.

Una vez cambiados los módulos, siguiendo el orden en zig–zag, se calcula la IDFT para obtener finalmente la imagen marcada I'.

Por lo tanto, los pasos que dará el insertor serán los siguientes:

1. Lectura de la imagen original.
2. Lectura del watermark X que se va a insertar.
3. Transformar la imagen I mediante la DFT.
4. Formación del vector de módulos originales M.
5. A partir del vector M, generar el vector M', utilizando para ello la ecuación (4.16).
6. Cambiar los módulos originales de la imagen I por los módulos modificados del vector M'.
7. Calcular la IDFT de la imagen I para obtener la imagen marcada.
8. Opcionalmente, se puede aplicar un enmascaramiento visual para mejorar la imperceptibilidad del watermark. El procedimiento para enmascarar la imagen es idéntico al que se explicó para el caso del método original.

4.3.4. Detección del watermark

Dada una imagen I*, posiblemente corrompida por la acción de diversos ataques, se calcula su transformada DFT $N \times N$.

Los módulos de los coeficientes DFT de I* se ordenan en orden zig–zag y, comenzando por el $L+1$ hasta el $L+M$, se seleccionan para formar el vector de módulos marcados y posiblemente corrompidos M^*.

Dado que el método no ha insertado la información directamente en los módulos (variable M'), sino en la variable auxiliar T', lo primero que se debe hacer es determinar los valores de t_i^* a partir de los valores M_i^* leídos del espectro DFT de la imagen.

Si aplicamos la ecuación (4.14) directamente se obtiene que

$$t_i^* = r_i M_i^* \tag{4.17}$$

El detector obtiene los valores de r_i a partir de la ejecución del mismo algoritmo de generación de números pseudo–aleatorios que utilizó el insertor, dado que utilizan la misma semilla (*seed*).

Como indicador de la presencia de la marca se utiliza la correlación entre el vector T^*, y el watermark Y.

La expresión de la correlación z es idéntica a la del método original:

$$z = \frac{YT^*}{M} = \frac{1}{M}\sum_{i=1}^{M} y_i t_i^* \tag{4.18}$$

En la que y_i es el watermark que estamos verificando, y t_i^* son los módulos marcados y posiblemente corrompidos por algún ataque.

Suponiendo que la imagen con el watermark no ha sido corrompida, tenemos que

$$t_i^* = t_i' = t_i + \alpha|t_i|x_i \qquad , \qquad i = 1, 2, ..., M \tag{4.19}$$

Veamos ahora qué valores toma el parámetro z dada una imagen I' y diferentes watermarks, según se hayan utilizado, o no, para marcar la imagen. Supondremos ausencia de ataques ($M_i' = M_i^*$).

Supongamos que el watermark Y es igual al watermark X insertado en la imagen.

Entonces, haciendo las mismas consideraciones que antes, tenemos que

$$\begin{aligned}
X = Y \Rightarrow \quad z = \tfrac{1}{M} \quad & \textstyle\sum_{i=1}^{M} y_i t_i^* & = \\
= \quad & \tfrac{1}{M} \textstyle\sum_{i=1}^{M} (r_i M_i^* y_i + \alpha M_i^* x_i y_i) & = \\
= \quad & \tfrac{1}{M} \textstyle\sum_{i=1}^{M} r_i x_i M_i^* + \tfrac{1}{M} \textstyle\sum_{i=1}^{M} \alpha M_i^* & = \\
= \quad & \tfrac{1}{M} \textstyle\sum_{i=1}^{M} r_i x_i M_i^* + \tfrac{\alpha}{M} \textstyle\sum_{i=1}^{M} M_i^* & = \\
= \quad & \mu(RXM^*) + \alpha\mu(M^*) & = \\
= \quad & \mu(R)\mu(X)\mu(M^*) + \alpha\mu(M^*) & = \\
= \quad & 0 + \alpha\mu(M^*) & = \\
= \quad & \alpha\mu(M^*)
\end{aligned}$$

Hemos aplicado que X y R son variables aleatorias independientes de media cero, y por lo tanto la media de $x_i r_i$ es cero.

Veamos ahora qué ocurre si $X \neq Y$:

$$\begin{aligned}
X \neq Y \Rightarrow \quad z = \tfrac{1}{M} \quad & \textstyle\sum_{i=1}^{M} y_i t_i^* & = \\
= \quad & \tfrac{1}{M} \textstyle\sum_{i=1}^{M} (r_i M_i^* y_i + \alpha M_i^* x_i y_i) & = \\
= \quad & \mu(RM^*Y) + \alpha\mu(M^*XY) & = \\
= \quad & \mu(R)\mu(M^*)\mu(Y) + \alpha\mu(M^*XY) & = \\
= \quad & 0
\end{aligned}$$

Aquí se ha aplicado que X y R son variables aleatorias independientes de media cero.

Las translaciones no modifican el módulo de los coeficientes DFT de la imagen transformada, por lo que no afectan de ninguna manera al detector.

En cambio, las rotaciones en la imagen implican la rotación del espectro de la DFT, por lo que el detector ha de encontrar el ángulo para el cual la media z es máxima, y verificar si supera, o no, el umbral T_z establecido.

Por lo tanto, para saber si el watermark que estamos comprobando realmente está presente en la imagen, el detector seguirá los siguientes pasos:

1. Lectura de la imagen marcada.
2. Lectura del watermark Y que se va a comprobar.
3. Transformar la imagen I mediante la DFT.
4. Inicializar el ángulo inicial $\theta = 0$, la variable `maximo` a cero, y fijar un incremento angular δ.
5. Formación del vector T^* a partir de los módulos situados en las posiciones del orden en zig–zag rotadas un ángulo θ respecto de la frecuencia (0,0).
6. Calcular el parámetro de correlación z entre T^* y Y.
7. Recalcular el valor del umbral T_z.
8. Comparar z con el umbral T_z.
9. Si z es mayor que la variable `maximo` asignar a la variable `maximo` el valor de z.
10. Incrementar el ángulo $\theta = \theta + \delta$.
11. Si $\theta < \pi$ (rad), volver al paso 5.
12. Comparar el valor de la variable `maximo` con T_z:
    Si `maximo` $\geq T_z \Rightarrow$Detección POSITIVA
    Si `maximo` $< T_z \Rightarrow$ Detección NEGATIVA

Normalmente las imágenes que llegan al detector habrán sufrido algún tipo de ataque, por lo que el valor de `maximo` calculado puede que no alcance el valor que cabría esperar en el caso de que el watermark verificado coincida con el que fue utilizado para marcar la imagen. Aunque siempre estará próximo a cero si el watermark que verificamos no coincide con el que está presente en la imagen.

Por lo tanto, utilizaremos un umbral T_z menor que la media calculada de los $\alpha|t_i^*|$, en concreto tres veces por debajo, por lo que finalmente se tiene que

$$T_z = \frac{\alpha}{3M} \sum_{i=1}^{M} |t_i^*| \tag{4.20}$$

Hay que tener en cuenta que el valor de T_z va cambiando según el ángulo de rotación que se esté inspeccionando, ya que los valores de $|t_i^*|$ son diferentes según el ángulo en el que se hayan leído y, por lo tanto, hay que recalcular el valor de este umbral en cada rotación que se analice, como se ve en el punto 7 del algoritmo.

4.3.5. Pruebas realizadas

Para la evaluación del método propuesto para este proyecto, utilizaremos tres imágenes diferentes (*Lena*, *Fishing Boat* y *Man*), y mostraremos la respuesta del detector a un ataque del tipo *combinado*, es decir, aplicaremos a cada imagen varios ataques simultáneos, y veremos cuál es el comportamiento del detector una vez que ha determinado y corregido el ángulo de rotación que se aplicó como parte del ataque.

También se mostrarán gráficas del comportamiento del detector cuando el watermark que comprueba coincide con el insertado en la imagen, e intenta determinar el ángulo de rotación, calculando la media z para todos los posibles ángulos analizados, y escogiendo el ángulo que corresponde a la mayor media z.

Las tres imágenes se obtuvieron de la base de datos de la University of Southern California [2], y al igual que en los tests anteriores, su resolución es de 512 x 512 pixels, y con 256 posibles tonos de gris (8 bits/pixel).

Marcaremos las imágenes con estos parámetros:

- $\alpha = 0.4$.
- $L = M = 16000$.
- $R = 9$.
- $\delta = 0.1$ grados.

Hay que tener en cuenta que aunque los nombres de los parámetros sean los mismos, estamos utilizando un método diferente al original, dado que utiliza una transformada distinta (DFT), y también se han introducido modificaciones en los algoritmos de inserción y detección y, por lo tanto, no se pueden evaluar los métodos únicamente comparando entre ellos los valores de los parámetros, porque pertenecen a métodos diferentes.

Como en los anteriores tests, también se aplicará el enmascaramiento visual.

En concreto, evaluaremos el funcionamiento del método para un ataque combinado que consistirá en:

- *Stretching* del histograma.
- Ecualización del histograma.
- Compresión JPEG con factor de compresión 50.
- Rotación[12] de 65 grados (sentido contrario al de las agujas del reloj).

[12]El detector implementado utiliza el sentido de giro contrario, por lo que el ángulo que devuelve está desfasado 180 grados.

- *Cropping* (intencional y debido también al efecto de la rotación y la translación).
- Translación.
- Filtro mediano de apertura 3.
- Adición de ruido gaussiano de $\mu = 0$ y $\sigma^2 = 10$.

4.3.6. Prueba de robustez con la imagen *"Lena"*

La imagen original, antes de efectuar ningún ataque es la que se puede ver en la Figura 4.30.

Figura 4.30: Imagen original *"Lena"*.

Tras aplicar los ataques, la imagen resultante que analizó el detector es la de la Figura 4.31.

La máxima media z se dio para $z = 0.639157$, con un umbral $T_z = 0.582582$.

En la Figura 4.32, se puede ver una gráfica con la evolución de la media z y de su umbral T_z asociado, para los posibles ángulos de rotación de la imagen, y suponiendo que al detector se le ha pasado el watermark correcto.

Figura 4.31: Imagen *"Lena"* después de la combinación de ataques.

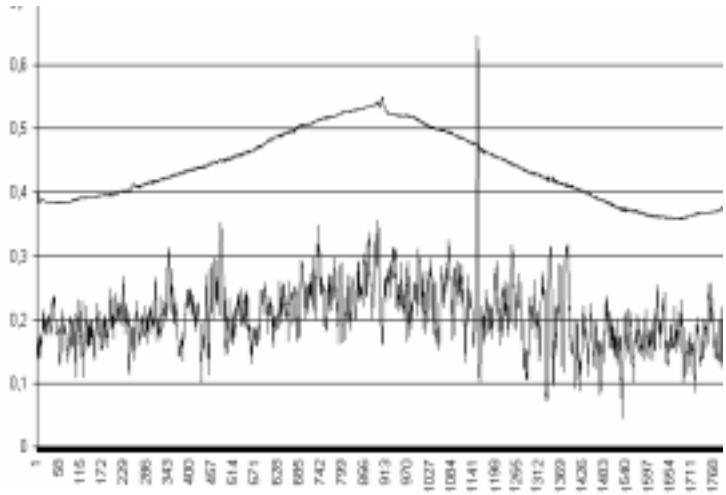

Figura 4.32: Comportamiento del detector según el ángulo de rotación (décimas de grado) para la imagen *"Lena"*.

4.3.7. Prueba de robustez con la imagen *"Fishing Boat"*

La imagen original, antes de efectuar ningún ataque es la que se puede ver en la Figura 4.33.

Figura 4.33: Imagen original *"Fishing Boat"*.

Tras aplicar los ataques, la imagen resultante que analizó el detector es de la Figura 4.34.

La máxima media z se dio para $z = 0.784823$, con un umbral $T_z = 0.666268$.

En la Figura 4.35, se puede ver una gráfica con la evolución de la media z y de su umbral T_z asociado, para los posibles ángulos de rotación de la imagen, y suponiendo que al detector se le ha pasado el watermark correcto.

Figura 4.34: Imagen *"Fishing Boat"* después de la combinación de ataques.

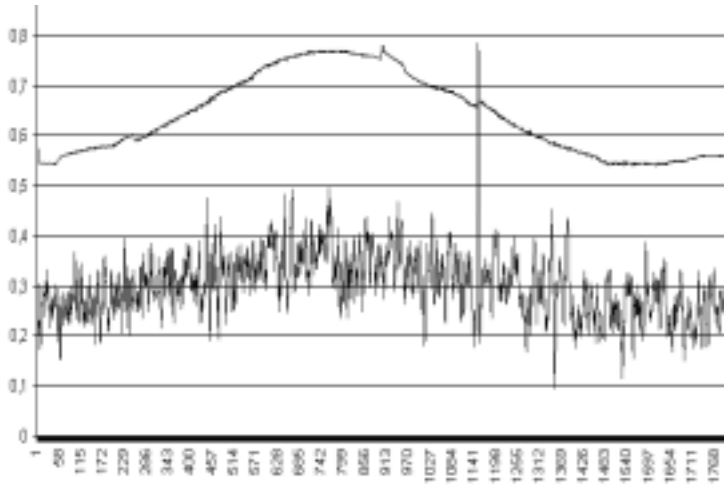

Figura 4.35: Comportamiento del detector según el ángulo de rotación (décimas de grado) para la imagen *"Fishing Boat"*.

4.3.8. Prueba de robustez con la imagen *"Man"*

La imagen original, antes de efectuar ningún ataque es la que se puede ver en la Figura 4.36.

Figura 4.36: Imagen original *"Man"*.

Tras aplicar los ataques, la imagen resultante que analizó el detector es la de la Figura 4.37.

La máxima media z se dio para $z = 0.677474$, con un umbral $T_z = 0.514420$.

En la Figura 4.38, se puede ver una gráfica con la evolución de la media z y de su umbral T_z asociado, para los posibles ángulos de rotación de la imagen, y suponiendo que al detector se le ha pasado el watermark correcto.

Figura 4.37: Imagen *"Man"* después de la combinación de ataques.

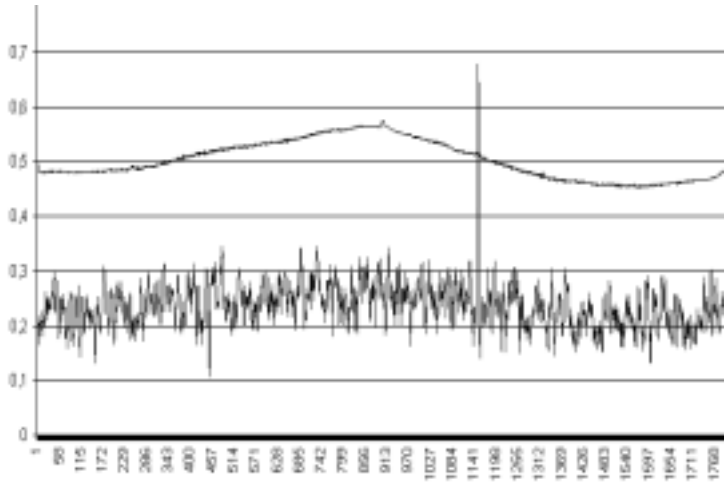

Figura 4.38: Comportamiento del detector según el ángulo de rotación (décimas de grado) para la imagen *"Man"*.

4.4. Conclusiones y líneas de investigación futura

En vista de los resultados de las pruebas efectuadas al método [1], y presentadas en este proyecto, la conclusión es que el método ha dado resultados positivos en aquellos casos en los que estaba previsto su buen funcionamiento, y entre todos los ataques, cabe destacar su inmunidad al ruido.

No obstante, se observaron algunas deficiencias, tales como la incapacidad de detectar la marca de autentificación en aquellos casos en los que la imagen sufría un desplazamiento o una rotación, por lo que se decidió mejorarlo.

El método mejorado se basó esencialmente en el método [1], pero su introdujeron variaciones significativas, como el pasar a usar la transformada DFT en lugar de la DCT, y las modificaciones en los algoritmos de inserción y detección que implica este cambio.

También se introdujeron algunas técnicas nuevas que no habían sido consideradas en el método [1], que permiten aplicar el método original para insertar el watermark sobre una variable que no tenga media nula, lo cual permite insertar la información del watermark en los módulos de la transformada DFT.

Los resultados de este nuevo método también han sido expuestos en este proyecto, y la conclusión es que los resultados han sido satisfactorios, ya que el detector consigue reconocer la firma de autentificación en el caso de ataques simples, y combinaciones de ellos, además de con imágenes rotadas y desplazadas.

Aún así, se pueden sugerir algunas líneas de investigación futura sobre este método, y aspectos que se podrán mejorar aún más, pero que quedan fuera del ámbito de este proyecto.

Uno de los aspectos que requieren más investigación, es determinar cuántos watermarks se pueden insertar dentro de una misma imagen antes de que la calidad se vuelva inadmisible para los propósitos de la aplicación en concreto.

Las pruebas efectuadas en este proyecto dieron como resultado que se podían insertar hasta cinco watermarks antes de que la calidad se viera seriamente afectada, aunque sería conveniente un estudio más riguroso.

Relacionado con lo anterior, hay otro aspecto que sería interesante analizar en profundidad, y es el estudio de cuáles son los parámetros con los que habría que configurar el insertor para un funcionamiento óptimo.

En concreto, a partir de la imagen que se quiera marcar, se tendría que poder determinar cuál es el parámetro α adecuado para esa imagen y aplicación en concreto, y los parámetros L y M óptimos.

Otra variación del método presentado, podría consistir en cambiar el algoritmo de inserción y detección, de manera que el watermark no se inserte siempre según el orden en zig–zag (lo cual da a posibles atacantes una idea de dónde pueden estar los coeficientes marcados), sino que la posición exacta de los coeficientes dependa de la firma digital del autor, por ejemplo, de la información extraída de algunos de sus bits. El desarrollo de un algoritmo que lleve a cabo esta tarea queda pendiente y fuera del ámbito concreto de este proyecto.

Finalmente, uno de los inconvenientes de este nuevo método frente al original, es el tiempo de procesamiento, dado que se han de verificar uno por uno todos los posibles ángulos, y calcular para cada uno la media z y su umbral T_z asociado.

Este problema se soluciona si se recurre a la programación concurrente, es decir, a modificar los programas de manera que se calculen *al mismo tiempo* las medias z y umbral T_z correspondientes a varios ángulos, aprovechando el hecho de que el análisis de cada ángulo es independiente de los demás, y se puede efectuar de manera simultánea.

Incluso se podría aprovechar la capacidad de procesamiento de varios ordenadores conectados un red para que ejecutaran simultáneamente los cálculos correspondientes a cada ángulo, y que enviaran sus resultados al programa coordinador, que se encargaría de enviar las peticiones, y recoger los resultados.

Otra posible mejorar de los algoritmos es extensión a imágenes en color.

Esta extensión se puede hacer de forma sencilla, sin introducir cambios significativos en los algoritmos, simplemente cambiando la representación del color de cada pixel.

Si en vez de representar la crominancia en el espacio RGB se hace en un sistema equivalente en el que una de las variables sea la luminancia, entonces es posible insertar la información del watermark en esa componente.

Un posible espacio al cual transformar es el YUV, en el que Y, U y V son combinaciones lineales de las componentes R, G y B, y la variable Y representa la luminancia del pixel. Esta representación es la misma que utiliza JPEG internamente.

Así pues, el insertor en color daría los siguientes pasos:

1. Cargar la imagen en color original.
2. Convertir la representación RGB de cada pixel a YUV.
3. Utilizar la componente Y para marcar la imagen, aplicando el mismo algoritmo que en el caso monocromático.
4. Convertir la representación $Y'UV$ (en la que Y' es la luminancia marcada) a RGB.

5. Guardar la imagen en color marcada.

Por su parte, el detector haría lo siguiente:

1. Cargar la imagen en color original.
2. Convertir la representación RGB de cada pixel a YUV.
3. Utilizar la componente Y para detectar la presencia del watermark, aplicando el mismo algoritmo que en el caso monocromático.

La forma de pasar de RGB a YUV es la siguiente:

$$Y = 0.299R + 0.587G + 0.114B \tag{4.21}$$

$$U = -0.169R - 0.332G + 0.5B + 128 \tag{4.22}$$

$$V = 0.500R - 0.419G - 0.0813B + 128 \tag{4.23}$$

Y para transformar YUV en RGB:

$$R = Y + [1.4075(V - 128)] \tag{4.24}$$

$$G = Y - 0.3455(U - 128) - 0.7169(V - 128) \tag{4.25}$$

$$B = Y + 1.7790(U - 128) \tag{4.26}$$

El método mejorado que se ha presentado es totalmente funcional, y ha corregido algunas deficiencias del original aunque, por supuesto, quedan muchos aspectos en los cuales investigar y tratar de mejorar.

Capítulo 5

Watermarking frágil

Al igual que se hizo con las técnicas de watermarking robusto, en esta sección se va a hacer una revisión del concepto de watermarking frágil, y se mostrarán algunas de las técnicas más representativas que se han desarrollado con anterioridad, sin entrar en detalles, y únicamente con la intención de dar una visión general.

Las técnicas de watermarking frágil se diferencian de las de watermarking robusto en que el objetivo principal no es que la marca insertada pueda ser recuperada tras la aplicación de los ataques, sino que precisamente se intenta conseguir que los ataques modifiquen el watermark insertado de manera que sea posible extraer información de interés.

La aplicación más evidente es la detección de modificaciones en una imagen original, y el procedimiento consiste en insertar una marca invisible al observador humano, de manera que al modificar la imagen, el watermark quede también modificado, y el método permita decidir si la imagen ha sido modificado, o no, respecto de la original, y en la mayoría de los casos, localizar espacialmente en qué posición se han producido las modificaciones.

Respecto de las propiedades que debería tener un buen método de watermarking frágil, las siguientes son las más importantes:

- Debe ser capaz de localizar espacialmente las modificaciones.
- Debe ser imperceptible para un observador humano, o conseguir un gran parecido entre la imagen original y la marcada.
- No debería ser necesaria la imagen original para el funcionamiento del sistema. En caso contrario, el algoritmo se podría reducir a una simple comparación pixel a pixel entre la imagen original y la marcada.
- Debe existir una autoridad oficialmente reconocida que sea la que inserte y verifique las firmas de autentificación.

- Es deseable que el método de watermarking pueda resistir cierto grado de compresión, aunque esto no es estrictamente necesario, y depende del ámbito de aplicación en concreto del método.

La mayoría de las técnicas se basan en insertar una firma de autentificación (el watermark), que consiste en una secuencia generalmente binaria asociada a cada posible usuario.

Al igual que en el caso de watermarking robusto, un método de watermarking frágil se compone de tres elementos:

- El watermark, o firma de autentificación.
- El insertor.
- El detector.

Su funcionamiento es el mismo que se explicó en el caso de watermarking robusto, y la única diferencia entre este tipo de métodos y los robustos es precisamente que éstos últimos son resistentes a los ataques, mientras que los frágiles no.

La mayoría de las técnicas existentes segmentan la imagen en bloques contiguos, de manera que cubran la mayor área posible de imagen, y se inserta la marca bloque a bloque.

La forma de insertarla depende del método en concreto, y puede insertar todo el watermark en un mismo bloque, repartir la información entre varios bloques, concatenar varios bloques, etc.

También depende del método en concreto el dominio de inserción de la marca, que puede ser:

- **Espacial** (por ejemplo, modificar el LSB de los pixels de cada bloque).
- **Frecuencial** (por ejemplo, cuantificar determinados coeficientes de la DCT).
- **Combinaciones de ambos**, que se dan cuando se utiliza la transformada wavelet, ya que combina los dos aspectos anteriores.

Al igual que sucede con las técnicas de watermarking robusto, el detector suele basar sus decisiones en la comparación del watermark que se comprueba con el detectado en la imagen, y generalmente se calcula un coeficiente de correlación y se compara con un cierto umbral de decisión.

Un posible ataque destinado a romper un método de autentificación puede consistir en, una vez conocido el algoritmo de marcado, intentar averiguar cuál

es el watermark que ha sido insertado en la imagen, para luego modificarla y reinsertar el watermark.

De esta manera, el detector no sería capaz de encontrar diferencias entre el watermark insertado y el comprobado. Si este ataque tiene éxito, entonces sería posible modificar la imagen, sin que el detector detectase ni localizara los cambios, y su decisión sería siempre negativa (imagen no modificada).

Para evitar la posibilidad de que se den este tipo de ataques es conveniente que las secuencias watermark no sean del dominio público, sino que sólo tenga acceso a ellas una autoridad de watermarking debidamente reconocida.

Es conveniente que la seguridad de un método de watermarking (al igual que ocurre con cualquier método de criptografía) no dependa de la necesidad de mantener en secreto el método o los algoritmos de marcado y detección, sino que debería depender únicamente del secreto de las claves, en este caso, de las secuencias watermark.

En el caso de que las secuencias watermark sean públicas, la autoridad de watermarking que inserta y detecta las modificaciones, debería como mínimo utilizar ciertos parámetros secretos que no permitieran que un atacante pudiera prever la localización de la marca, y en ningún caso detectarla o extraerla.

Si el atacante logra extraer la marca, esto significaría que ha logrado averiguar los parámetros secretos con los que la autoridad de watermarking configura los algoritmos de marcado y detección, y en ese caso, podría romper el método. El método propuesto de watermarking frágil en este proyecto es un ejemplo de método en el cual un atacante no puede acceder directamente a los datos del watermark sin el conocimiento previo de una clave secreta que sólo conoce la autoridad de watermarking.

Por supuesto, el secreto de las secuencias watermark y los parámetros de configuración de la autoridad de watermarking se pueden proteger mediante el uso de la criptografía, y existen varios esquemas de criptografía asimétrica que podrían solucionar de forma efectiva estos problemas, pero no es el objetivo de este proyecto el análisis de métodos criptográficos, y nos centraremos únicamente en los algoritmos de marcado y detección de watermarking frágil.

5.0.1. Revisión de técnicas previas

Existe una gran variedad de técnicas de watermarking frágil que se han desarrollado hasta ahora, y el objetivo de este apartado es mostrar de forma general sólo algunas de las más representativas.

La técnica más sencilla a la hora de insertar el watermark frágil para detectar modificaciones consiste en modificar el LSB [1] del nivel de intensidad codificado

[1] Siglas de Less Significant Bit (bit menos significativo).

para cada pixel de la imagen. Luego, en la fase de detección, se verifica si los LSB siguen cumpliendo el criterio que se siguió cuando fueron eliminados.

Una forma muy sencilla de hacerlo consiste en poner todos los LSB a cero (o a uno), de manera que la modificación hará que algunos pixels pasen a tener un LSB igual a uno (o a cero), y por lo tanto se detectará la modificación.

El hecho de cambiar el valor del LSB no tiene efectos perceptibles para un observador humano, ya que implica el cambio en un nivel de cuantificación, y un observador humano sólo puede percibir diferencias de color cuando el cambio es de cinco o más niveles (suponiendo ocho bits por pixel de nivel de gris).

Aunque sea una técnica sencilla, es al mismo tiempo muy potente y ampliamente utilizada, ya sea de forma directa, como en el ejemplo anterior, o introduciendo algunas variantes.

Dado que sólo se modificada el bit menos significativo de la codificación de cada pixel, esto significa que la modificación tiene probabilidad $p = \frac{1}{2}$ de afectar al LSB, es decir, que sólo en la mitad de las ocasiones se detectará la modificación.

Existen técnicas para evitar este problema, como por ejemplo dividir la imagen en diferentes bloques (agrupaciones de pixels contiguos dispuestos en forma rectangular), que recubran la imagen, y en vez de marcar el LSB de cada pixel de forma individual, se establece un criterio para marcar todos los LSB dentro del bloque, de forma que la posibilidad de detectar un error ya no dependa únicamente de un pixel individual, sino de todos los pixels del bloque.

En el artículo *"Information Authentication for a Slippery New Age"* [11] Watson propone una técnica basada en dividir la imagen en bloques, combinada con la técnica LSB explicada anteriormente. Utiliza un generador de números pseudo–aleatorios generados a partir de una clave determinada para obtener una suma de control construida a partir de la suma de los siete bits más significativos de los pixels y tomando el resto de la división de la suma de control entre un entero N muy grande. En la comprobación de integridad de la imagen, se analiza el resto obtenido, y se determina si la imagen fue modificada, o no.

No es preciso explicar en detalle el método de Watson, pero esta técnica y otras son un referente que demuestra que es posible reducir la probabilidad de no detectar una modificación si en vez de considerar pixels individuales, se consideran bloques de pixels. En el caso del método de Watson, la probabilidad obtenida es $p = \frac{1}{N}$, frente a $p = \frac{1}{2}$, que se obtiene al marcar los pixels de forma individual.

En general, la contrapartida que presentan los métodos basados en dividir la imagen en bloques, es precisamente que no es posible localizar espacialmente con tanta exactitud la modificación (como ocurre en el caso de modificar el LSB de los pixels de forma individual), sino que la detección se hace por bloques, de

manera que el detector determina que hubo modificaciones en un determinado bloque, pero no puede saber qué o cuántos pixels en concreto han participado en la modificación.

R. G. Wolfgang y E. J. Delp en su artículo *"A Watermark for Digital Images"* [12] proponen una técnica también basada en la modificación del LSB, pero con el añadido de que es resistente a compresión JPEG, siempre y cuando el factor de compresión no esté por encima de un valor prefijado.

El método se basa en calcular la correlación entre el watermark que se comprueba, y el detectado en la imagen, pero se permite cierta relajación en la decisión, ya que el valor del coeficiente de correlación calculado se compara con un umbral de decisión.

Ajustado adecuadamente el umbral de decisión, se consigue que el método sea resistente a compresión JPEG, al tiempo que puede detectar muchas posibles combinaciones de modificaciones en la imagen. Por supuesto, la resistencia a JPEG se produce al relajar el umbral de decisión, por al mismo tiempo también se aumenta la probabilidad de no detectar algunas modificaciones. Al aumentar el umbral de decisión la resistencia a JPEG disminuye, pero también lo hace la probabilidad de no detectar una modificación.

Un método también basado en el cómputo de la correlación entre el watermark comprobado y el insertado, pero utilizando un método transformado, es por ejemplo el artículo *"Image Watermarking for Tamper Detection"* [13] de Jiri Fridrich.

A grandes rasgos, el algoritmo es similar a [12], pero en vez de ser un método espacial, se basa en la modificación de ciertos bits de algunos coeficientes de la transformada DCT, y la detección se realiza mediante el cálculo de la correlación entre el watermark extraído y el comprobado. También permite relajar la detección mediante un umbral de detección, para adquirir cierta resistencia a la compresión JPEG.

Existen otros métodos transformados que no está basados en la modificación directa de bits de coeficientes transformados, como por ejemplo el método propuesto por Min Wu y Bede Liu en el artículo *"Watermarking for Image Authentication"* [14], en el que la modificación consiste [2] en cuantizar algunos coeficientes de la DCT, y la decisión de marcar o no un determinado coeficiente viene dada por los valores binarios presentes en el watermark. La detección y posterior decisión se hace, como es habitual, mediante el cálculo de la correlación entre el watermark comprobado y el detectado en la imagen.

[2] Existen otras diferencias, como que no se marcan todos los bloques, sino que se seleccionan aquellos en los que se prevé que la cuantización del coeficiente DCT no afectará de manera significativa a la calidad de la imagen.

Algunas técnicas de watermarking frágil se han basado en la criptografía y en las funciones de hash, como por ejemplo P.W.Wong, que en su artículo *"A Public Key Watermark for Image Verification and Authentication"* [15] propone utilizar un esquema de watermarking en el que la integridad y autoría de la imagen puede ser comprobada utilizando una clave pública.

El propietario de la imagen inserta un watermark (de manera espacial, por ejemplo, en el LSB de la luminancia de los pixels) utilizando su clave privada K'. En la extracción del watermark, cualquier persona puede utilizar la clave pública K (correspondiente a la clave privada K') para extraer el watermark.

Ping Wah Wong y Nasir Memon, en su artículo *"Secret and Public Key Image Watermarking Schemes for Image Authetication and Ownership Verification"* [8] proponen un método espacial para conseguir asociar una imagen con su verdadero propietario [3].

La imagen es segmentada en bloques, y para cada uno de ellos se computa una función de hash.

La función de hash (en concreto, se utiliza la MD5 [10]), toma como entrada un vector formado por estos elementos:

- La clave K del usuario.
- El identificador de imagen I_Y.
- El ancho de la imagen M_Y.
- El alto de la imagen N_Y.
- Un identificador r del bloque que se está procesando.
- El valor (por ejemplo, luminancia) de los pixels del bloque.

Para insertar el watermark, simplemente se calcula la operación or-exclusiva (XOR) entre el LSB (bit menos significativo) del valor de cada pixel con cada uno de los bits que ha generado la función de hash, para cada pixel del bloque. Dado que el cambio en el LSB únicamente modifica un nivel el valor del pixel, y suponiendo que como mínimo se codifican 256 niveles por pixel, entonces los cambios son invisibles[4].

La operación se repite para todos y cada uno de los bloques de la imagen.

La detección consiste en tomar la imagen original, crear un vector de datos análogo al anterior, y comparar los bits que genera la función de hash con los que

[3]Este método además también sirve para detectar modificaciones en la imagen.

[4]Los cambios en la luminancia son invisibles siempre que la variación sea igual o menor que cinco niveles, cuando el rango es [0, 255].

codifica el LSB de los pixels de la imagen marcada. Si estos valores coinciden, entonces se decide detección positiva, y negativa en caso contrario.

Las ventajas que presenta de este método son, por una parte, que permite localizar espacialmente en qué bloque se han producido modificaciones, en el caso de que se utilice el método para detectarlas, y que no es necesaria la imagen original para su funcionamiento.

La mayor desventaja es que no es robusto a ningún ataque, por lo que en principio, no sería válido como esquema de watermarking robusto destinado a asociar la imagen a su verdadero autor, ya que es muy fácil eliminar la firma de autentificación.

Este método, y en general muchos que están basados en el uso de funciones de hash, no son resistentes prácticamente a ningún ataque (generalmente, no resisten ninguno), y por lo tanto, tampoco son aplicables en ámbitos en los que se aplique la compresión como un procesamiento más de la señal.

La ventaja es que no es posible extraer el watermark de la imagen, ya que queda protegido por la encriptación y el secreto de la clave privada del usuario, y por lo tanto, es un método muy seguro.

El método propuesto en este proyecto también se basa en ocultar al atacante la información a partir de la cual se hace la detección de las modificaciones (es decir, el watermark), pero está basado en un método totalmente nuevo.

5.0.2. Método desarrollado de watermarking frágil

5.0.3. Introducción

La aplicación del método de watermarking frágil que se ha desarrollado para este proyecto es la detección modificaciones en la imagen marcada por el insertor, sin necesidad de la imagen original.

Este método está basado en un concepto nuevo que hasta ahora no había sido aplicado al campo del watermarking, y se trata del análisis de la distribución de conjuntos de nivel de una imagen de nivel de gris.

Aprovechando las propiedades de la transformada de conjuntos de nivel, se ha desarrollado un método que además es **seguro**, en el sentido de que la seguridad del método está basada en que para un atacante no es posible acceder a la estructura de conjuntos de nivel, ya que queda enmascarada por un ruido pseudo–aleatorio.

El atacante no sólo no puede modificar la imagen sin que los cambios sean detectados a posteriori, sino que dada una imagen marcada, ni siquiera puede obtener información que le permita saber si ha sido previamente marcada, o no.

El método no sólo tiene la aplicación de detectar modificaciones, sino que también puede ser utilizado para la verificación del propietario de la imagen, ya que es posible utilizar la información del identificador del usuario como clave privada a la hora de insertar la marca de comprobación y de extraerla.

Se hará una introducción a la teoría de conjuntos de nivel en la que está basado el método, para posteriormente presentar los algoritmos de marcado y verificación, y finalmente se mostrarán resultados prácticos de la aplicación del sistema.

5.0.4. Representación de las imágenes mediante sus conjuntos de nivel

La parte de la Visión por Computador dedicada al estudio de las características básicas que definen las imágenes recibe el nombre de *Análisis de Imágenes*.

Generalmente se acepta que la información básica de una imagen está contenida en los contornos de los objetos presentes en la misma, en el sentido de que somos capaces de reconocer la imagen a partir de estos contornos (ver la Figura 5.1).

Figura 5.1: Imagen original (izquierda) y algunos de los contornos de los objetos que aparecen en ella (derecha). Somos capaces de reconocer la imagen a partir de estos contornos.

Lo que no está tan claro es la manera de definir y calcular estos contornos. Los primeros trabajos en este campo, debidos a Marr y su equipo ([22], 1980), definen los contornos (o *edges*) como la colección de puntos en la imagen con máxima variación de intensidad. Estos puntos se calculan como puntos de inflexión de la función de intensidad (ver Figura 5.2) y en la práctica se calculan tras un filtraje paso-bajo previo de la imagen. De esta manera se obtienen una serie de contornos para cada escala de filtraje (representación multi-escala de la imagen [23], Figura 5.3). La extracción de los *edges* depende de la escala de filtrado y de algunos umbrales para determinar la presencia, o no, de los puntos de inflexión.

Otra manera de definir los contornos es a partir de una **segmentación** de la imagen [17]. Los pixels con propiedades parecidas (p.ej. nivel de gris similar) se agrupan en regiones; las fronteras entre estas regiones determinan los contornos buscados (ver Figura 5.4). En general, las técnicas de segmentación dependen de un parámetro que regula el número final de regiones en las que se puede dividir la imagen.

Recientemente ([21]) se ha propuesto una nueva descripción de las imágenes basada en sus conjuntos de nivel. Estos conjuntos de nivel incluyen los contornos de la imagen, no requieren de ningún parámetro para su cálculo y además poseen una serie de propiedades particularmente útiles para el tratamiento de las imágenes.

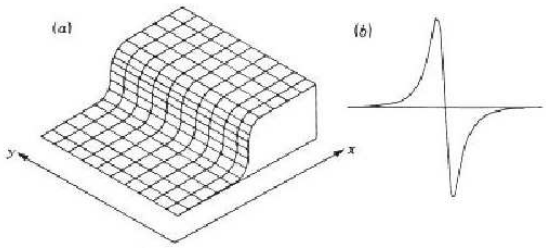

Figura 5.2: [20] Los puntos de inflexión de la función intensidad (derivada primera máxima, derivada segunda igual a cero) marcan los puntos de máxima variación de la intensidad. (a) muestra un cambio de intensidad $u(x, y)$ y (b) muestra los valores de la segunda derivada en la dirección del eje x.

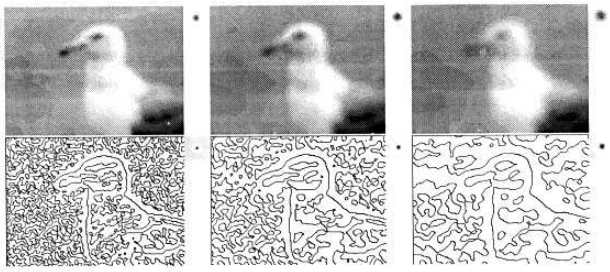

Figura 5.3: [18] *Edges* obtenidos mediante el método de Marr para diferentes escalas de filtraje. Arriba, imágenes filtradas; abajo, *edges*.

Figura 5.4: Imagen original (izquierda), imagen segmentada con 300 regiones (centro) y contornos de la segmentación (derecha).

Dada una imagen $u(x, y) : \Omega \subseteq \mathbb{R}^2 \longrightarrow \mathbb{R}$ definimos el **conjunto de nivel (superior)** λ de u como:

$$\mathcal{X}_\lambda u = \mathcal{X}_\lambda^{\sup} u = \{(x, y) \in \Omega : u(x, y) \geq \lambda\} \tag{5.1}$$

De manera análoga podemos definir el **conjunto de nivel inferior** λ como:

$$\mathcal{X}_\lambda^{\inf} u = \{(x, y) \in \Omega : u(x, y) \leq \lambda\} \tag{5.2}$$

Observación. Un conjunto de nivel (superior o inferior) puede estar formado por una o varias componentes conexas[5]. Cada una de estas componentes conexas recibe el nombre de **shape** o **forma** del conjunto. (ver la Figura 5.5).

La Figura 5.5 muestra un ejemplo de conjunto de nivel superior de una imagen.

Algunas de las propiedades más relevantes de los conjuntos de nivel son las siguientes:

1) **Representación sin parámetros**. En una imagen[6] digital existen un máximo de 255 conjuntos de nivel superiores (resp. inferiores) diferentes ($\mathcal{X}_0 u$, $\mathcal{X}_1 u$, $\cdots$, $\mathcal{X}_2 55u$, $\mathcal{X}_0^{\inf} u$, $\cdots$, $\mathcal{X}_2 55^{\inf} u$). Es relativamente sencillo calcular **todos** los conjuntos de nivel de la imagen, de manera que no es necesario ningún parámetro que determine qué conjuntos deben calcularse.

2) **Propiedades de inclusión**.

   I) $\mathcal{X}_\lambda u \subseteq \mathcal{X}_\mu u$, para todo $\lambda > \mu$

   II) $\mathcal{X}_\lambda u = \cap_{\mu<\lambda} \mathcal{X}_\mu$ u, para todo $\lambda \in \mathbb{R}$

   La Figura 5.6 muestra (marcados en blanco) algunos ejemplos de conjuntos de nivel para una imagen. Se puede observar la propiedad de inclusión de estos conjuntos a medida que aumenta λ. También se observa que cada conjunto de nivel está formado por una o varias *shapes*.

   Estas propiedades de inclusión permiten el diseño de un algoritmo rápido para el cálculo de las *shapes* presentes en una imagen y su almacenamiento en una estructura de datos con forma de árbol (ver el ejemplo de la Figura 5.7). P. Monasse y F. Guichard desarrollaron este algoritmo en 1999 y lo denominaron FLST (Fast Level Sets Transform) [19].

[5] Se dice que dos pixels son conexos si tienen una arista en común (noción de 4-conexidad: el pixel (i, j) es conexo a los pixels $(i + 1, j)$, $(i - 1, j)$, $(i, j - 1)$ y $(i, j + 1)$). Otra definición posible de conexidad (8-conexidad) es la que dice que dos pixels son conexos si tienen algún vértice en común. Para cualquiera de las dos definiciones de conexidad anteriores se dice que un conjunto de pixels es conexo (4 u 8 conexo) si dentro del conjunto existe un camino formado por pixels conexos (4 u 8 conexidad) que une dos pixels cualesquiera del conjunto.

[6] Suponiendo una codificación de 8 bits/pixel.

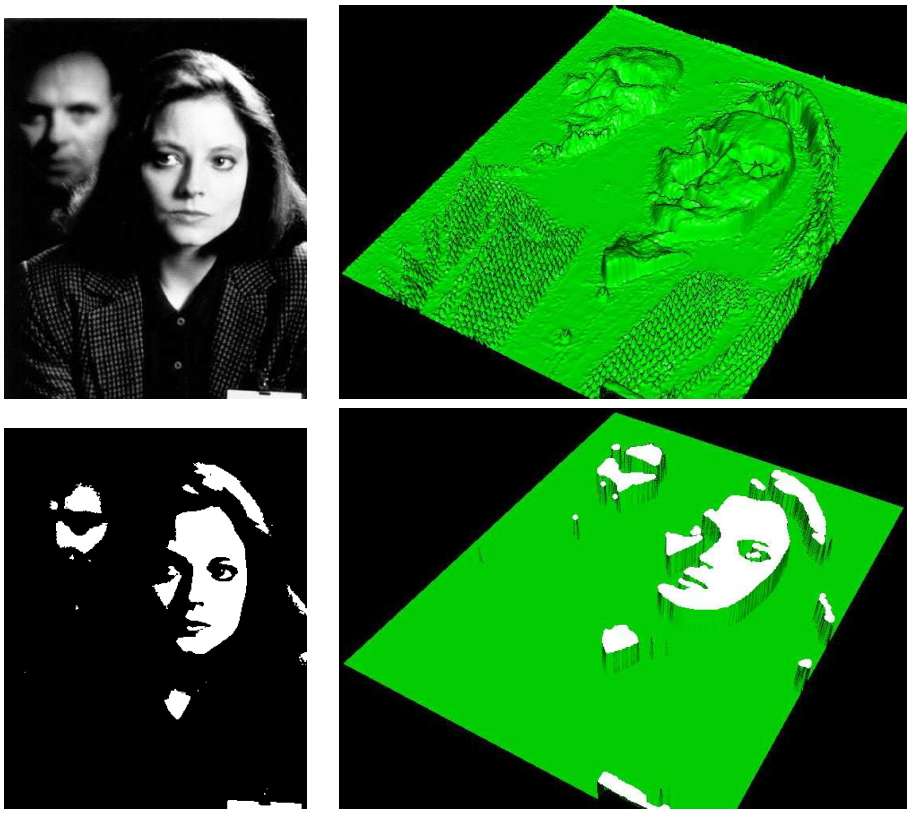

Figura 5.5: Representación tridimensional de una imagen y de uno de sus conjuntos de nivel. En la parte superior, imagen original y representación 3D. En la parte inferior, un conjunto de nivel de la imagen y su representación tridimensional. Observamos como el conjunto de nivel está compuesto por varias componentes conexas.

Figura 5.6: Imagen original y conjuntos de nivel superiores para $\lambda = 100$, 150 y 200. Los píxeles que pertenecen al conjunto de nivel se han marcado en color blanco. Se observa la propiedad de inclusión entre los conjuntos y la existencia de varias componentes conexas en cada uno de ellos.

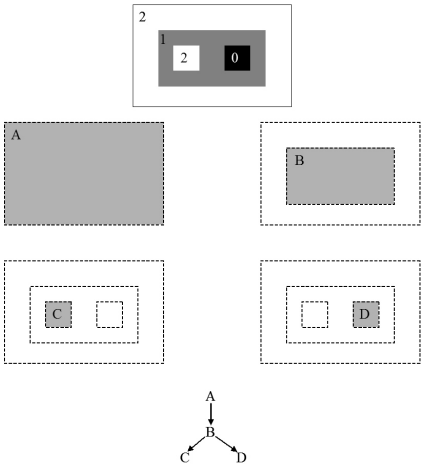

Figura 5.7: Árbol de *shapes* proporcionado por la FLST para una imagen sencilla.

3) La descripción de la imagen proporcionada por los conjuntos de nivel es una **descripción completa**, en el sentido de que es posible recuperar la imagen original a partir de sus conjuntos de nivel mediante la fórmula de reconstrucción siguiente:

$$u(x,y) = \sup\{\lambda \in \mathbb{R} : (x,y) \in \mathcal{X}_\lambda u\} \tag{5.3}$$

4) **Invariancia ante cambios de contraste**. Dadas dos imágenes de la misma escena u y v relacionadas por un cambio de contraste ($v = g(u)$, donde g es una función continua estrictamente creciente), u y v tienen los mismos conjuntos de nivel. Es decir, todo conjunto de nivel de u es un conjunto de nivel de v, y viceversa. Las Figuras 5.8 y 5.9 ilustran esta propiedad.

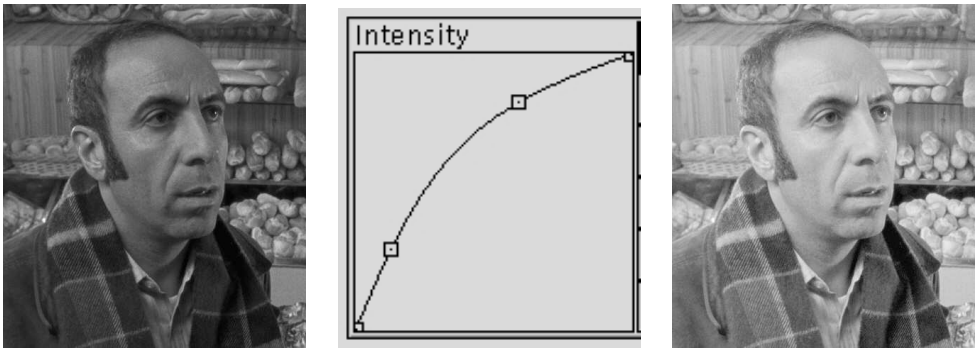

Figura 5.8: Imagen original, cambio de contraste e imagen resultante de aplicar el cambio de contraste a la primera imagen.

Esta propiedad es especialmente importante y distingue la representación basada en conjuntos de nivel de las representaciones basadas en contornos (*edges*) o en segmentación. Dadas las imágenes u y v descritas más arriba, el conjunto de *edges* o de contornos de segmentación obtenidos para cada imagen serían distintos, por lo que un algoritmo de análisis de imágenes basado en estas técnicas concluiría que ambas imágenes son distintas.

Sin embargo, desde el punto de vista de la percepción humana, ambas imágenes representan la misma escena y contienen los mismos objetos o formas, por lo que deben considerarse como equivalentes. En este sentido, el análisis proporcionado por los conjuntos de nivel está mejor adaptado a nuestra percepción visual. Se dice que un análisis de la imagen basado en sus conjuntos de nivel es un análisis **morfológico**.

Finalmente, se ha de comentar que las *shapes* de pequeño tamaño (de área

Figura 5.9: La imagen muestra un conjunto de nivel común a las imágenes mostradas en la figura anterior. El conjunto de nivel corresponde al nivel 128 para la primera imagen y al nivel 192 para la segunda.

inferior a 10 pixels) son, en general, debidas al ruido, por lo que pueden ser eliminadas sin afectar a la calidad de la imagen. El filtraje consistente en eliminar estas pequñas *shapes* recibe el nombre de **eliminación de extremos** o **filtro de grano**. La estructura de árbol proporcionada por la FLST es especialmente útil para realizar este filtraje.

La manera de eliminar una *shape* de la imagen consiste en asignar a los pixels que la componen el nivel de gris de la *shape* 'padre' en la estructura de árbol de la FLST. La Figura 5.10 muestra un ejemplo aplicado a la sencilla imagen de la Figura 5.7. La Figura 5.11 muestra algunos ejemplos de la aplicación del filtro de grano para la eliminación de *shapes* de distintas tamaños.

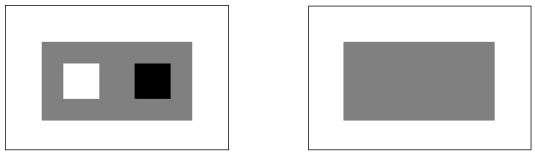

Figura 5.10: Ejemplo de la eliminación de dos pequeñas shapes en una imagen sencilla. De izquierda a derecha, la imagen original y la imagen simplificada. A partir del árbol de shapes calculado en la Figura 5.7, las shapes C y D son eliminadas y se asigna un nuevo nivel de gris a los pixels que pertenecían a las mismas. El nuevo nivel de gris es el de la shape 'padre' de ambas (shape B).

Figura 5.11: De izquierda a derecha, imagen original y el resultado de eliminar las formas de tamaño 10 y 40. Como se puede observar, en el primer caso la imagen filtrada es prácticamente idéntica a la original, mientras que en el segundo caso algunos detalles empiezan a perderse (esto se nota, por ejemplo, en la textura de tejados de las casas).

5.0.5. Algoritmo de marcado

Acabamos de ver que la FLST proporciona una descripción completa de la imagen y, en párticular, de las formas o *shapes* que aparecen en la misma. Cualquier modificación de la imagen consistente en añadir una nueva forma será fácilmente detectada al comparar la FLST de la nueva imagen con la FLST original. En cambio, si la modificación consiste en un cambio de contraste (por ejemplo, una ecualización de histograma) el cambio no será detectado puesto que las FLSTs serán idénticas.

En esta sección se describe un método de watermarking basado en la información proporcionada por la FLST, combinada con la introducción de un ruido pseudo–aleatorio, que permite detectar y localizar prácticamente cualquier modificación de la imagen, incluso aquellas debidas a cambios de contraste (ver la sección 5.0.7 para una discusión sobre la probabilidad de no detección de alguna modificación).

Si se quieren detectar las modificaciones sin necesidad de recurrir a la comparación con la imagen original, entonces tenemos que conseguir cambiar la distribución de los conjuntos de nivel en la imagen original de tal manera que cumplan con unos requisitos determinados que se puedan verificar en la fase de detección, y por supuesto, sin afectar a la calidad de la imagen de forma significativa.

Nos referiremos a este proceso como *"marcado"*.

En concreto, la técnica que se utilizará para que la distribución de conjuntos de nivel cumpla unos ciertos requisitos detectables a posteriori será eliminar las *shapes* interiores [7] que cumplan que su área está dentro de un intervalo determinado.

[7] Las shapes interiores son aquellas que no tienen ningún pixel sobre el borde delimitador del área de la imagen.

Si se eliminan las shapes cuyas áreas sean pequeñas en comparación con las del resto de shapes de la imagen, entonces la calidad de la imagen apenas se ve afectada, y para un observador humano es difícil apreciar la diferencia entre la imagen marcada y la original (ver Figura 5.11).

Dado que queremos que las modificaciones puedan ser localizadas espacialmente dentro de la imagen, se ha optado por dividir la imagen en bloques, y eliminar las shapes de forma local para cada bloque.

Dentro de cada bloque se eliminan todas aquellas shapes que cumplen que ningún punto del conjunto de pixels que la componen esté en contacto con el borde del bloque, independientemente de su área, y nos referiremos a este tipo de shapes como *interiores*.

Por lo tanto, se eliminarán este tipo de shapes de la imagen, y en la fase de detección se verificará si la imagen analizada contiene shapes cuyas áreas coincidan con las que fueron previamente eliminadas. Si esas áreas aparecen, entonces el detector decidirá que la imagen fue modificada, y si no, entonces se concluirá que la imagen se corresponde con la original.

El motivo por el que sólo se eliminan shapes interiores es porque si se quitan shapes que están en contacto con el borde de un bloque, se pueden producir *efectos de frontera* que tienen un impacto visual elevado.

Cuando se eliminan shapes interiores, se están eliminando shapes dentro del bloque que son las que están presentes en la imagen de forma natural, pero si por el contrario eliminamos una shape de forma local de un bloque, entonces puede darse la situación de que la shape que estamos eliminando no sea una shape de la imagen propiamente dicha, sino que sea el resultado de dividir una shape más grande en varias shapes de áreas más pequeñas, que debido al proceso de división en bloques, ha quedado repartida y dividida en diferentes bloques.

Cada uno de estos bloques contiene una parte de la shape de la imagen, pero si tratamos este hecho de forma local, nos encontramos con que cada bloque contiene una shape de un área determinada, que no es interior, sino que está en el borde del bloque.

Si se eliminan las shapes de los bloques siguiendo el criterio de áreas, entonces puede darse el caso de que la shape que está presente en dos o más bloques ahora sea eliminada de algunos bloques, pero de otros bloques contiguos no, dependiendo del área interceptada por cada bloque. El resultado es que la shape que estaba presente de forma natural en la imagen, después del proceso de eliminación, da la impresión visual de haber quedado "recortada", según la forma de los bloques que la contienen.

Por lo tanto, se decidió que, para evitar este tipo de efectos con alto impacto visual, sólo se eliminarían aquellas shapes que son interiores en cada bloque de la imagen.

Se han efectuado diversas pruebas, y quedó demostrado experimentalmente que un tamaño de 10×10 pixels consigue un buen compromiso entre un bloque lo suficientemente pequeño como para localizar las modificaciones con cierta precisión, y la suficiente fiabilidad, en el sentido de que bloques con un área menor puede que no contengan shapes interiores, y por lo tanto las modificaciones no serían detectadas en algunos casos.

Los pixels que han quedado desprotegidos son aquellos que están en la frontera, formando un cuadrado o rectángulo que rodea el área interior.

Para detectar modificaciones en estos pixels de frontera, se añade un paso adicional al algoritmo de marcado, que consiste en eliminar previamente todas las shapes cuyo área esté comprendida entre 1 y 10, de forma global[8].

Con esto se consigue que si se produce el caso de que las modificaciones afecten únicamente a la frontera de los bloques, se puedan detectar y localizar espacialmente por la aparición en el árbol FLST global de la imagen de shapes de áreas comprendidas entre 1 y 10.

Por otra parte, dado que el algoritmo de marcado previsiblemente será público [9], cualquier atacante podría efectuar el siguiente ataque:

1. Cargar la imagen marcada.
2. Efectuar modificaciones.
3. Eliminar de forma global shapes cuyas áreas estén comprendidas entre 1 y 10.
4. Eliminar las shapes interiores de cada bloque.
5. Grabar la imagen resultante.

Si el atacante puede acceder a la distribución de conjuntos de nivel, entonces puede romper el método fácilmente.

Es necesaria una protección adicional que enmascare la información requerida por el atacante.

Para impedir que este ataque se pueda llevar a la práctica, el proceso de marcado se completa insertando en la imagen un ruido pseudo–aleatorio que consiste en sumar a cada valor de luminancia codificado en cada pixel de la imagen (suponiendo que están entre 0 y 255), un valor n_i tal que $n_i \in \{0, 1, 2, 3\}$.

[8] Es decir, ya no se consideran los pequeños bloques contiguos en los que se divide la imagen, sino que se trata la imagen de forma unitaria.

[9] Y aunque no lo fuera, la seguridad del sistema debería recaer en el secreto de las claves, y nunca en el secreto del método.

Los cuatro valores posibles se seleccionan de forma pseudoaleatoria a partir de una clave secreta conocida por la autoridad de watermarking, y su selección es equiprobable con $p = \frac{1}{4}$. Evidentemente, la autoridad de watermarking **genera** y **conoce** la clave secreta que debe utilizar tanto en el insertor como en el detector.

Dado que no se trata de un ruido aleatorio propiamente dicho, sino que es una señal determinista y repetible, el detector puede eliminar el ruido simplemente efectuando una resta, siempre y cuando conozca la clave secreta que permite generar la mima secuencia de valores de n_i.

A priori, se presentan dos dificultades derivadas de utilizar el ruido pseudo–aleatorio, aunque como veremos, no impiden aplicar el método.

La primera, es la posibilidad de que después de sumar el ruido a un pixel, el valor de luminancia quede por debajo de cero. Esta primera posibilidad no se dará nunca, ya que el valor de la luminancia siempre es positivo, en el rango $[0, 255]$, y las muestras de ruido $n_i \in \{0, 1, 2, 3\}$ también lo son, lo cual implica que la suma siempre será positiva.

La segunda, es la posibilidad de que después de sumar el ruido a un pixel, el valor de luminancia quede por encima de 255, es decir, que se produzca el desbordamiento del byte.

Si el valor resultante queda fuera del rango [0, 255], y dado que el valor que almacena el pixel realmente es la suma en módulo 256, entonces el impacto visual es notable.

Por ejemplo, si el pixel tuviera el valor 255, y se le suma 2, entonces tendríamos que el valor almacenado sería $255 + 2(mod256) = 257(mod256) = 1$. Es decir, se ha pasado de un valor de luminancia máxima, a uno muy oscuro.

Para evitar el desbordamiento de la capacidad del byte, antes de insertar el ruido se fuerza a que aquellos pixels cuyo valor de luminancia esté por encima o sea igual a 252 pasen a tener valor el valor 252.

El nuevo valor (252) asegura que la suma del valor de luminancia con cualquier muestra de ruido siempre queda dentro del rango [0, 255], evitando el desbordamiento de byte y el consiguiente efecto muy visible.

En resumen, el atacante no conoce la secuencia de valores que se ha utilizado para generar el ruido y, por lo tanto, no lo puede eliminar. Si no puede eliminar el ruido, no puede determinar la distribución de los conjuntos de nivel de la imagen dado que ha quedado totalmente corrompida por el ruido.

En la fase de detección, el detector sí que puede determinar qué secuencia fue la que se superpuso a la imagen, dado que es una secuencia pseudoaleatoria

construida a partir de una clave conocida por la autoridad de watermarking, y por lo tanto puede eliminar la secuencia superpuesta, simplemente restando las muestras de ruido del valor de luminancia de los pixels de la imagen.

De esta forma, el atacante nunca accede a la distribución de conjuntos de nivel con los cuales opera el algoritmo, sino sobre una versión de la imagen en la que esta información queda enmascarada.

En la sección 5.0.7 se hacen algunos comentarios adicionales sobre la probabilidad de no detectar alguna modificación en los pixels de la imagen marcada.

El algoritmo de marcado consistirá en estos pasos:

1. Cargar la imagen original.
2. Para cada valor de luminancia c de cada pixel de la imagen, comprobar que si está por encima o es igual al valor 252.
   Si $c \geq 252$ entonces asignar $c \leftarrow 252$.
3. Eliminar de *forma global* todas aquellas shapes que cumplan que su área esté dentro del rango [1, 10].
4. Determinar el número (entero) de bloques horizontales y verticales que componen la imagen. Se calcula así:

   Cx = (unsigned)(ColumnasImagen / T_{cx});

   Cy =(unsigned)(FilasImagen / T_{cy});

   donde T_{cx} y T_{cy} son respectivamente el número de columnas y el número de filas que componen el bloque. En nuestro caso, $T_{cx} = T_{cy} = 10$.
5. Leer la clave secreta, y utilizarla como *seed* inicial para el generador de números pseudo–aleatorios.
6. Inicializar un bucle que recorrerá todos los bloques de la imagen.
7. Seleccionar el siguiente bloque.
8. Eliminar de las shapes *interiores* del bloque seleccionado, de forma *local*.
9. Si se han procesado todos los bloques, salir del bucle (paso siguiente). Si no, continuar con el bucle.
10. Insertar el ruido pseudo–aleatorio en la imagen, sumando cada muestra generada de ruido al valor de luminancia de cada pixel.
11. Guardar la imagen marcada.

La inserción del ruido pseudo–aleatorio consiste en lo siguiente:

1. Inicializar un bucle que recorra todos los pixels de la imagen de entrada.
2. Seleccionar de forma equiprobable un número pseudo–aleatorio r del conjunto $\{0, 1, 2, 3\}$.
3. Calcular $c = c + r$.
4. Sustituir el valor de luminancia del pixel seleccionado por el nuevo valor de c.
5. Si se han procesado todos los pixels, fin del algoritmo. Si no, continuar con el bucle.

5.0.6. Algoritmo de detección

El algoritmo de detección toma como entrada la imagen marcada en la que se quieren localizar las modificaciones, y genera una nueva imagen en la que quedan señalados aquellos bloques en los que se ha detectado una modificación respecto de la imagen original.

Evidentemente, el algoritmo no necesita comparar la imagen de entrada con la original, sino que lo único que necesita el detector es conocer la clave secreta que se utilizó como parámetro del proceso de marcado.

La clave es necesaria, ya que a partir de ella se inicializa el generador de números pseudo–aleatorios, y mediante este generador se reconstruye la secuencia que utilizó el algoritmo de marcado para insertar el ruido pseudo–aleatorio.

La superposición de esta secuencia pseudo–aleatorio de muestras de ruido hace posible que un atacante no pueda averiguar la distribución de áreas de los conjuntos de nivel de la imagen original.

Por lo tanto, el primer paso que da el detector es precisamente eliminar el ruido, ya que es capaz de reconstruir la secuencia de ruido pseudo–aleatorio.

Una vez que se ha eliminado el ruido, procede a la detección de shapes cuyo área esté dentro del área [1, 10] y a localizarlas espacialmente.

Hay que recordar que el proceso de marcado eliminó esta clase de shapes, y por lo tanto su aparición sólo puede ser debida a modificaciones en la imagen marcada, y por lo tanto el detector considera como zonas no–válidas aquellas en las que aparecen este tipo de formas, y procede a marcarlas.

Posteriormente, el detector divide la imagen en bloques idénticos a los que utilizó el algoritmo de marcado, de la misma forma. Como se ha explicado anteriormente, se considera que un tamaño de 10×10 pixels es óptimo, por lo

que se utiliza este tamaño por defecto, aunque si por otros motivos se decide utilizar un tamaño diferente, entonces sería otro parámetro a tener en cuenta, además de la clave secreta.

Para cada uno de estos bloques se cuenta el número de shapes interiores, y si cumplen con el criterio establecido (en nuestro caso, que no haya ninguna shape interior), entonces se decide que el bloque en cuestión no ha sido modificado, mientras que si por el contrario no cumple con el criterio, se decide que el bloque sí que ha sufrido alguna modificación.

Por lo tanto, el detector ejecutará el siguiente algoritmo:

1. Cargar la imagen marcada.
2. Detectar *de forma global* todas aquellas shapes de la imagen que cumplan que su área está dentro del área $[1, 10]$, y marcarlas como zonas *modificadas* de la imagen marcada.
3. Leer la clave secreta, y utilizarla como *seed* inicial para el generador de números pseudo–aleatorios.
4. Eliminar el ruido pseudo–aleatorio de la imagen.
5. Determinar el número (entero) de bloques horizontales y verticales que componen la imagen. Se calcula así:

   Cx = (unsigned)(ColumnasImagen / T_{cx});

   Cy =(unsigned)(FilasImagen / T_{cy});

   donde T_{cx} y T_{cy} son respectivamente el número de columnas y el número de filas que componen el bloque. En nuestro caso, $T_{cx} = T_{cy} = 10$.
6. Inicializar un bucle que recorrerá todos los bloques de la imagen.
7. Seleccionar el siguiente bloque.
8. Contar el número de shapes (n) interiores del bloque seleccionado.
9. Si $n \neq 0$ entonces marcar el bloque como **modificado**.
10. Si se han procesado todos los bloques, fin del algoritmo. Si no, continuar con el bucle.

El proceso de eliminación del ruido pseudo–aleatorio es muy similar al de adición, con la diferencia de que ahora no es necesario verificar el posible desbordamiento de los bytes, ya que durante el proceso de marcado se dio valores a la luminancia de los pixels de manera que no se pudiera dar nunca esa situación.

Otra diferencia evidente, es que ahora la secuencia que se utilizó para generar el ruido se *resta* del valor de luminancia de los pixels, para eliminarlo.

Por lo tanto, el proceso de eliminación del ruido consiste en los siguientes pasos:

1. Inicializar un bucle que recorra todos los pixels de la imagen de entrada.
2. Leer el valor de luminancia del pixel, y asignarlo a la variable c (8 bits sin signo).
3. Seleccionar de forma equiprobable un número pseudo–aleatorio r del conjunto $\{0, 1, 2, 3\}$.
4. Calcular $c = c - r$.
5. Sustituir el valor de luminancia del pixel seleccionado por el nuevo valor de c.
6. Si se han procesado todos los pixels, fin del algoritmo. Si no, continuar con el bucle.

5.0.7. Probabilidad de fallo en la detección de modificaciones

Existe un caso especial de modificación de la imagen marcada que puede pasar inadvertido al proceso de detección. Vamos a demostrar en esta sección que la probabilidad de que esto ocurra es muy baja.

Supongamos que alguien modifica la imagen marcada tal como se muestra en la Figura 5.12. Se trata de hacer una modificación que afecte a varios bloques de la imagen y que cree alguna shape no interior a ningún bloque. Si no se efectuara la eliminación del ruido pseudo–aleatorio introducido en la fase de marcado, la nueva shape creada no sería detectada, ya que el método tan solo detecta shapes interiores.

El hecho de eliminar el ruido pseudo–aleatorio modifica los valores de la shape introducida, haciendo que una parte de los pixels que la componen pasen a ser shapes interiores de algún bloque y por tanto ser detectados.

Existe, sin embargo, una pequeña probabilidad de que los pixels de la shape sean afectados de tal manera por el ruido pseudo–aleatorio que continúen formando una shape no interior.

En particular, esto ocurrirá cuando a todos los pixels de la shape se les sume la misma cantidad de ruido pseudo–aleatorio (es decir, a todos los pixels se les añade el valor 0, 1, 2 o 3). Dado que la probabilidad de que el ruido tome cualquiera de estos valores es $\frac{1}{4}$ y cada pixel de la shape es afectado de forma independiente, para una shape de área A, la probabilidad de que ocurra este suceso es $4\left(\frac{1}{4}\right)^A$. El factor 4 que aparece en esta expresión se debe a que consideramos cuatro posibles variaciones de los niveles de los pixels de la shape (valor 0, 1, 2 o 3).

Debido a que en el paso 2 de la detección se decide que la imagen ha sido modificada si existen shapes (tanto interiores como exteriores) de tamaño inferior o igual a 10, la única manera de que una nueva shape no interior no sea detectada es que tenga un área superior a 10 pixels. Para estas shapes, la probabilidad de no detección es igual a $4\left(\frac{1}{4}\right)^{10} \simeq 3.81 * 10^{-6}$. Por lo tanto, se puede concluir que la probabilidad de fallo en la detección es muy baja.

Cabría añadir que existe un caso en el que una modificación no sería detectada, que ocurre cuando, tras modificar un conjunto conexo de pixels, la introducción del ruido psedo-aleatorio hace que todos los pixels modificados pasen a tener el mismo valor y formar una shape interior de tamaño superior a 10 pixels. La probabilidad de que esto ocurra es de nuevo del orden de $\left(\frac{1}{4}\right)^A$, donde A es el tamaño de la shape formada.

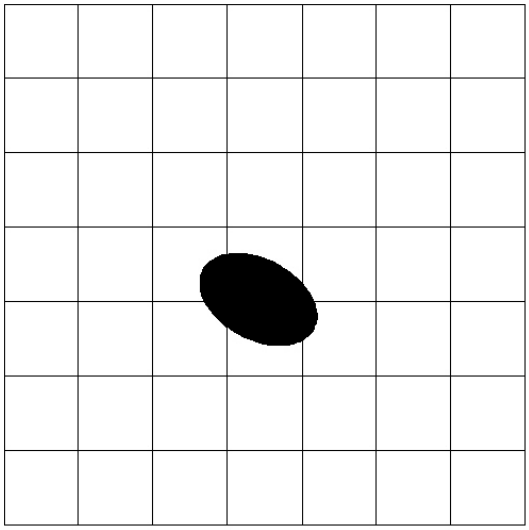

Figura 5.12: Ejemplo de modificación intencionada de la imagen que podría no ser detectada con el método desarrollado. Se trata de una modificación que afecta a una región conexa de pixels que forman una shape no interior a ningún bloque de la imagen. La probabilidad de no detección de tales modificaciones es muy baja.

5.0.8. Pruebas realizadas

Para mostrar el funcionamiento del método presentado, se han efectuado algunas pruebas con las imágenes estándar *Lena*, *House*, *Truck* y *Sailboat*, obtenidas de [2].

5.0.9. Pruebas de detección : *modificaciones con patrón conocido*

En esta prueba se evalúa el comportamiento del detector cuando analiza una imagen previamente marcada, en la que se han efectuado algunas modificaciones.

Se va a utilizar siempre el mismo patrón de modificaciones en cada imagen, para poder comparar el funcionamiento del detector en presencia de las mismas modificaciones, pero en diferentes imágenes.

El patrón de modificaciones es el que podemos ver en la Figura 5.13.

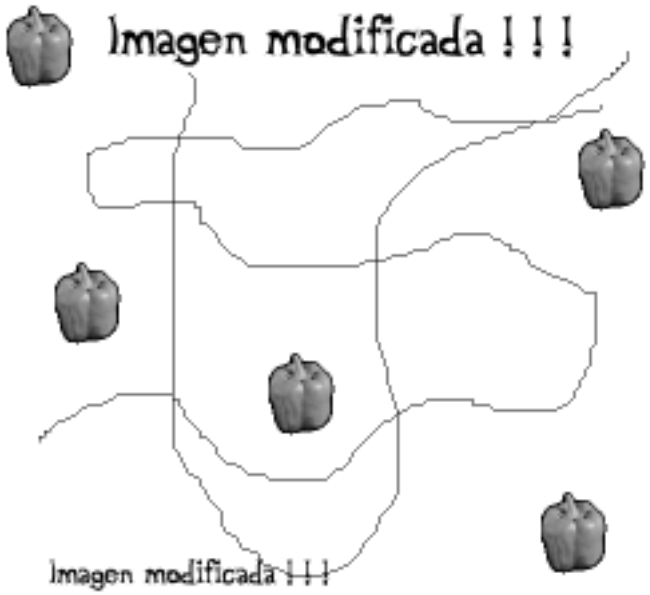

Figura 5.13: Patrón de modificaciones.

La modificación consiste en sustituir aquellos pixels de la imagen original por aquellos situados en las mismas coordenadas en el patrón de modificaciones,

siempre y cuando el punto considerado en el patrón no tenga el valor 255, que será considerado como "transparente", y no modificará la imagen original.

A continuación se mostrarán las imágenes originales, sus versiones marcadas, y las imágenes marcadas modificadas.

El detector genera una nueva imagen en la que marca las zonas modificadas, poniéndolas de color blanco, tanto cuando detecta la aparición global de shapes cuyo área está en el área [1, 10], como cuando detecta localmente shapes interiores en alguno de los bloques.

Por ejemplo, algunas secciones de las líneas del patrón de modificaciones pasan exactamente sobre el borde de algunos bloques, sin afectar el interior, y se puede observar que estas modificaciones se detectan por la detección de pequeñas shapes de áreas entre 1 y 10. El hecho de que las líneas negras del patrón se muestren blancas demuestra que el detector las ha reconocido como modificaciones.

Imagen "Lena"

Figura 5.14: Imagen original "Lena"sin marcar.

Figura 5.15: Imagen "Lena" marcada.

Figura 5.16: Imagen "Lena" modificada.

Figura 5.17: Salida del detector para la imagen “Lena”.

Imagen "House"

Figura 5.18: Imagen original "House"sin marcar.

Figura 5.19: Imagen "House" marcada.

Figura 5.20: Imagen "House" modificada.

Figura 5.21: Salida del detector para la imagen “House”.

Imagen "Truck"

Figura 5.22: Imagen original "Truck"sin marcar.

Figura 5.23: Imagen “Truck”marcada.

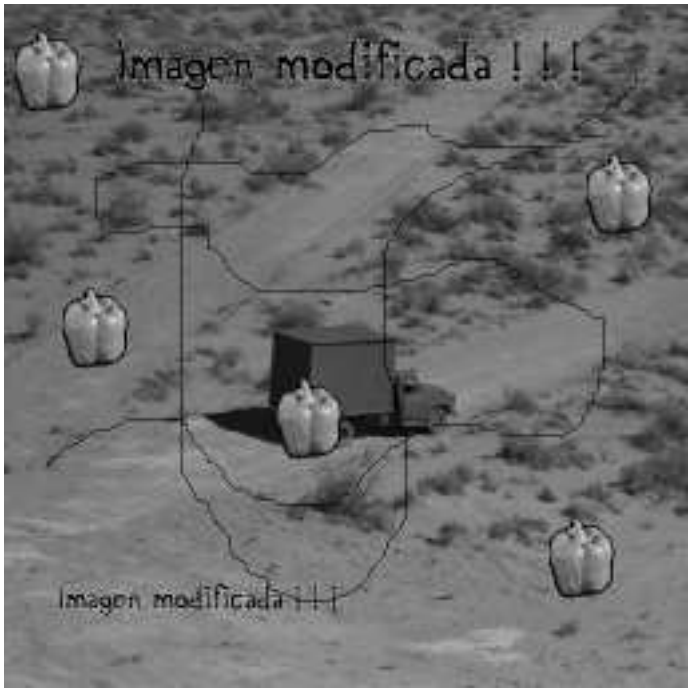

Figura 5.24: Imagen "Truck"modificada.

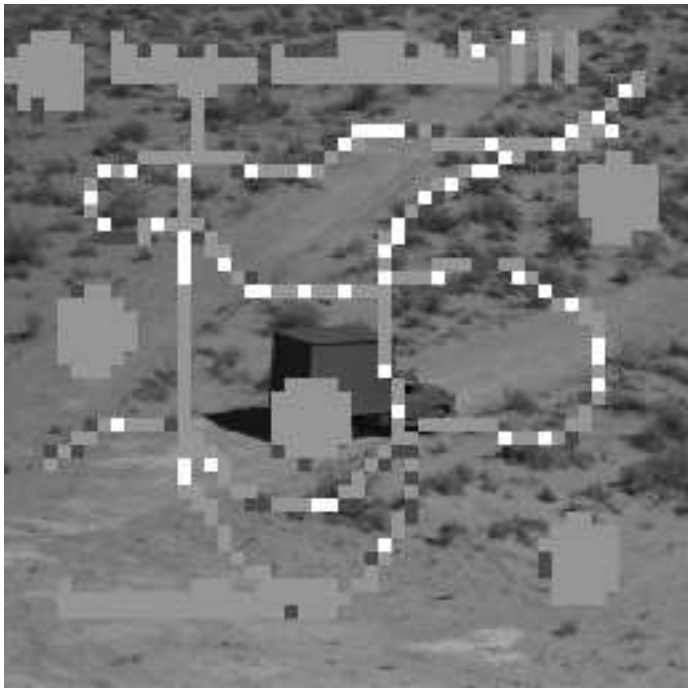

Figura 5.25: Salida del detector para la imagen “Truck”.

5.0.10. Pruebas de detección : *modificaciones arbitrarias*

En esta prueba se evalúa el comportamiento del detector cuando analiza una imagen previamente marcada, en la que se han efectuado algunas modificaciones que no siguen un patrón determinado, sino que se han efectuado simulando posibles ataques, en los que se pretende que un observador humano acepte erróneamente las modificaciones como parte de la imagen original.

Imagen "Truck"

Figura 5.26: Imagen "Truck" modificada.

Las modificaciones efectuadas han consistido en añadir varias réplicas del camión que aparece en el centro de la imagen.

Se puede imaginar una situación en la que la imagen original ha sido tomada por un satélite, y posteriormente ha sido modificada de esta manera, para dar la impresión de que en una determinada zona existe un convoy de vehículos enemigos que avanza hacia una determinada posición.

Si la imagen ha sido marcada (como efectivamente se ha hecho), entonces en destino se podría verificar si la imagen es verdadera o, como es el caso, se trata de un fraude.

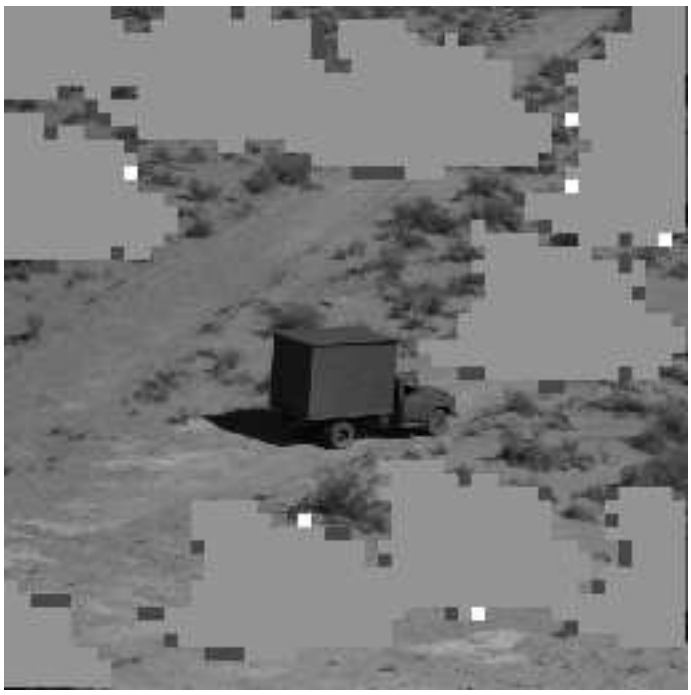

Figura 5.27: Salida del detector para la imagen “Truck”.

En esta imagen se puede ver la salida del detector después de analizar la imagen anterior, en la que se han detectado como modificaciones todos los nuevos vehículos que fueron insertados de forma fraudulenta, mientras que el vehículo central se considera que es auténtico.

También se puede observar que los bloques en los que se ha dividido la imagen no cubren la totalidad del área de la imagen (ver borde derecho), pero esto no impide la detección de las modificaciones, ya que el detector informa de la presencia de shapes de área dentro del rango [1, 10] en esa zona.

Imagen "Sailboat"

Figura 5.28: Imagen "Sailboat" modificada.

Las modificaciones efectuadas han consistido en añadir varias réplicas del velero original, mediante el mismo procedimiento que se utilizó con la imagen "Truck".

Figura 5.29: Salida del detector para la imagen “Sailboat”.

En esta imagen podemos ver la salida del detector después de analizar la imagen anterior, en la que se han detectado como modificaciones todos los nuevos veleros añadidos, mientras que sólo uno de ellos, el que estaba presente en la imagen original, se considera verdadero.

5.0.11. Conclusiones y líneas de investigación futura

Se ha expuesto un nuevo método basado en una técnica que hasta el momento nunca no había sido aplicada al campo del watermarking, basada en el análisis de la distribución de conjuntos de nivel de gris de la imagen.

También se introdujo una técnica que no había sido utilizada anteriormente, que es la inserción/eliminación de un ruido pseudo–aleatorio determinista y repetible para enmascarar la distribución de los conjuntos de nivel.

Los resultados fueron satisfactorios, ya que se suelen detectar todas las modificaciones, y como se demostró en la sección 5.0.7, la probabilidad de no detectar una modificación es muy baja.

Una desventaja del método presentado es que no es resistente a ningún ataque, por lo que ni siquiera es posible aplicar compresión JPEG, ya que el resultado del detector sería una imagen en la que la gran mayoría de los bloques aparecen modificados.

Un aspecto a investigar en el futuro podría consistir en estudiar en qué consiste exactamente la variación de los conjuntos de nivel al aplicar compresión JPEG, y tenerlo en cuenta en nuevas implementaciones del detector, para que pueda distinguir entre la aparición de shapes debidas a modificaciones intencionales y las provocadas por la compresión con pérdidas JPEG.

Otra mejora que se puede aplicar es la extensión a color, de la manera que se explicó en la sección de conclusiones de watermarking robusto (4.4).

La conclusión final es que los métodos presentados cumplen satisfactoriamente con los requisitos que se fijaron y son totalmente funcionales aunque, por supuesto, quedan varios aspectos a investigar y mejorar en el futuro.

Capítulo 6

Glosario

- **Bit** : Concepto matemático que representa la unidad de medida de la información. También se aplica el término bit *(BInary digiT)* a aquellos dígitos binarios que codifican uno (y sólo uno) de dos estados posibles.
- **Compresión** : Proceso mediante el cual se elimina información redundante (sin pérdidas) o poco relevante visualmente (con pérdidas), con el objetivo de reducir el tamaño de la imagen almacenada.
- **Conexidad** : Se dice que dos pixels son conexos si tienen una arista en común (noción de 4-conexidad: el pixel (i, j) es conexo a los pixels $(i+1, j)$, $(i-1, j)$, $(i, j-1)$ y $(i, j+1)$). Otra definición posible de conexidad (8-conexidad) es la que dice que dos pixels son conexos si tienen algún vértice en común. Para cualquiera de las dos definiciones de conexidad anteriores se dice que un conjunto de pixels es conexo (4 u 8 conexo) si dentro del conjunto existe un camino formado por pixels conexos (4 u 8 conexidad) que une dos pixels cualesquiera del conjunto.
- **Contraste** : Diferencia entre las zonas claras y oscuras de la imagen.
- **Cropping** : Proceso mediante el cual se eliminan espacialmente ciertas partes de la imagen.
- **DCT** : Discrete Cosine Transform (transformada discreta del coseno).
- **DFT** : Discrete Fourier Transform (transformada discreta de Fourier).
- **DWT** : Discrete Wavelet Transform (transformada wavelet discreta).
- **FC** : Factor de compresión.
- **FFT** : Fast Fourier Transform (transformada rápida de Fourier). Ver referencia [7].
- **FLST** : Fast Level Sets Transform (transformada rápida de conjuntos de nivel). Ver referencia [19].

- **HVS** : Human Visual System (sistema visual humano).
- **JPEG** : Joint Photographics Expert Group – Es una técnica de compresión de imágenes basado en compresión con pérdidas, que permite ajustar el factor de compresión.
- **Luminancia** : Intensidad de la emisión de luz de un pixel.
- **LSB** : Less Significant Bit (bit menos significativo).
- **LST** : Level Sets Transform (transformada de conjuntos de nivel). Ver referencia [19].
- **P2P** : Peer to peer. Tipo de comunicación entre iguales, generalmente aplicado al caso de usuarios, redes y programas que comparten ficheros en Internet.
- **Pixel** : PICture ELement (elemento de imagen).Es la mínima unidad de área de la imagen a la que se puede asignar un color o luminancia individual.
- **Resolución** : Pixels por unidad de área.
- **RGB** : Red-Green-Blue. Forma de representar el color de un pixel, separando los valores de luminancia de los canales rojo, verde y azul, respectivamente. Para más información sobre los espacios de color, consultar [20].
- **Seed** : Número que se utiliza para inicializar un generados de números pseudo–aleatorios. La secuencia que se genera es totalmente determinista y repetible, dado que siempre se genera la misma secuencia cuando se provee el mismo seed.
- **Semilla** : Ver *Seed*.
- **Shape** : Componente conexa que forma parte de un determinado conjunto de nivel. Ver término *conexidad*.
- **Shape interior** : Aquellas que no tienen ningún pixel sobre el borde delimitador del área de la imagen que la contiene.
- **XOR** : eXclusive-OR – Operación binaria or-exclusiva, equivalente a la suma en módulo dos.
- **YUV** : Sistema de representación cromática en la que Y, U y V son combinaciones lineales de R, G y B. La variable Y coincide con la luminancia. Para más información sobre los espacios de color, consultar [20].

Índice de figuras

Bibliografía

[1] Mauro Barni, Franco Bartolini, Vito Cappellini y Alessandro Piva : "A DCT-domain System for Robust Image Watermarking"(16 de noviembre de 2001).

[2] The USC-SIPI Image Database :
`http://sipi.usc.edu/services/database/Database.html`
Colección de imágenes digitalizadas para su uso en procesamiento digital, análisis de imágenes y visión por ordenador, de la University of Southern California.

[3] I.J. Cox, J. Kilian, T. Leighton y T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia", NEC Research Institute Technical Report 95 10, 1995.

[4] I.J. Cox, J. Kilian, T. Leighton y T. Shamoon, "Secure Spread Spectrum Watermarking for Images, Audio and Video", Proc. IEEE International Conference on Image Processing (ICIP'96), Lausanne, Switzerland, 16-19 Septiembre 1996, vol. III, pp. 243–246.

[5] Watermarking World :
`http://www.watermarkingworld.org`
La web de Watermarking World ofrece algunos enlaces interesantes a sitios relacionados con técnicas de watermarking, así como una lista de correo (`http://www.watermarkingworld.org/ml.html`) en la que se tratan nuevas técnicas, algoritmos, ideas, sugerencias, etc.

[6] Box, G.E.P, M.E. Muller 1958; A note on the generation of random normal deviates, Annals Math. Stat, V. 29, pp. 610-611.

[7] Cooley, J. W. y J. W. Tukey, "An Algorithm for the Machine Computation of the Complex Fourier Series,"Mathematics of Computation, Vol. 19, Abril 1965, pp. 297-301.

[8] Ping Wah Wong y Nasir Memon, "Secret and Public Key Image Watermarking Schemes for Image Authetication and Ownership Verification"IEEE Transactions on Image Processing, vol. 10, núm. 10, págs 1593-1601.

[9] D. Kundur y D. Hatzinakos, "Digital Watermarking Using Multiresolution Wavelet Decomposition"Proc. IEEE Conf. Acoust., Speech, Signal Processing, vol. 5, 199, págs 2969-2972.

[10] R. L. Rivest, "The MD5 message digest algorithm", Tech. Tep., 1992.

[11] S. Walton, "Information Authentication for a Slippery New Age", Dr. Dobbs Journal, vol. 20, núm. 4, pp. 18-26, Abril 1995.

[12] R. G. Wolfgang y E. J. Delp, "A Watermark for Digital Images", Proc. IEEE Intl. Conf. on Image Processing, ICIP-96, Vol.3, págs. 219-222.

[13] Jiri Fridrich, "Image Watermarking for Tamper Detection", Proc. IEEE International Conf. on Image Processing, ICIP-98, Vol.2, págs. 404-408.

[14] Min Wu y Bede Liu, "Watermarking for Image Authentication", Proc. IEEE International Conf. on Image Processing, ICIP-98, Vol.2, págs. 437-441.

[15] P.W.Wong, "A Public Key Watermark for Image Verification and Authentication", Proc. IEEE International Conf. on Image Processing, ICIP-98, Vol.1, págs 455-459.

[16] A. Bruce Carlson, "Communication Systems", McGraw–Hill, 1986.

[17] D. Mumford y J. Shah, "Optimal approximations by piecewise smooth functions and variational problems", Communications on Pure and Applied Mathematics, Vol. 42 núm.5, págs. 577-685, 1988.

[18] F. Guichard y J.M. Morel, "Iterative smoothing and PDE's", libro en preparación, 2002.

[19] F. Guichard y P. Monasse, "Fast computation of a contrast-invariant image representation", IEEE Transactions on Image Processing, Vol. 9 núm. 5, págs. 860-872, 2000.

[20] R. González y R.E. Woods, "Digital Image Processing", Addison-Wesley, 1992.

[21] V. Caselles, B. Coll y J.M. Morel, "Topographic maps and local contrast changes in natural images", International Journal on Image Processing, vol. 8 núm. 2, págs 5-27, Septiembre 1999.

[22] D. Marr y E. Hildreth, "Theory of edge detection", Proc. of the Royal Society, London, vol. 207, págs. 187-217, 1980.

[23] A.P. Witkin, "Scale-space filtering", International Joint Conference on Artificial Intelligence, págs. 1019-1022, Karlsruhe, 1983.