UNIVERSIDAD NACIONAL DE CÓRDOBA

Facultad de Ciencias Exactas, Físicas y Naturales

Tesis Doctoral

Marcas de agua múltiples para autentificación y detección de adulteraciones en imágenes digitales médicas

Autor: Mag. Ing. Laura Mónica Vargas

Directora: Dra. Ing. Elizabeth Vera de Payer

Abril de 2015

Marcas de agua múltiples para autentificación y detección de adulteraciones en imágenes digitales médicas

por

Mag. Ing. Laura Mónica Vargas

Dra. Ing. Elizabeth Vera de Payer

Directora

COMISIÓN ASESORA:

Dra. Ing. Elizabeth Vera de Payer
FCEFyN - UNC
Dr. Ing. Eduardo Destefanis
FCEFyN - UNC
Dr. Gustavo Juri
FCEFyN - UNC

Esta Tesis fue enviada a la Facultad de Ciencias Exactas Físicas y Naturales de la Universidad Nacional de Córdoba para cumplimentar los requerimientos de obtención del grado académico de Doctor en Ciencias de la Ingeniería.

Córdoba, Argentina

Abril de 2015

UNIVERSIDAD NACIONAL DE CORDOBA
Facultad de Cs. Exactas, Fisicas y Naturales

ACTA DE EXAMENES

Libro: 00001 Acta: 02972 Hoja 01/01
LLAMADO: 1 27/04/2015
CATEDRA - MESA:

DI002 TESIS DOCTORADO EN CIENCIAS DE LA INGENIERIA

NUMERO	APELLIDO Y NOMBRE	DOCUMENTO	INGRESO	COND.	NOTA	FIRMA
10398771	VARGAS, Laura Mónica	DNI: 10398771	2009	T	APROBADO	

CORRAL BRIONES, Graciela - RIESCO, Daniel - VALENTE, Mauro - CANALI, Luis - BUSTOS, Oscar

Observaciones:

Córdoba, 27/04/2015-.
Certifico que la/s firma/s que ha/n sido puesta/s en la presente Acta pertenece/n a: Graciela Corral Briones

Inscriptos	Ausentes	Examinados	Reprobados (0-3)	Aprobados (4-10)
1	–	1	–	1

21/04/2015 13:37:28 Libro/Acta: 0000102972 Hoja: 01/ 01

A la Dra. Ing. Elizabeth Vera de Payer por iluminar mi vida profesional.

A Edgardo, Candela y Álvaro, en el orden en que llegaron a mi vida para iluminarla.

A mis padres, a Lea y a Olga.

ÍNDICE GENERAL

Página

LISTA de FIGURAS

LISTA de TABLAS

Lista de Acrónimos

CT Computer Tomography / Tomografía Computarizada

DCT Discrete Cosine Transform / Transformada Discreta de Coseno

DE Difference Expansion / Expansión de la Diferencia

DICOM Digital Imaging and Communications in Medicine

DWT Discrete Wavelet Transform / Transformada Discreta Wavelet

IDWT Inverse Discrete Wavelet Transform / Inversa de la Transformada Discreta Wavelet

GA Genetic Algorithms / Algoritmos Genéticos

LSB Least Significant Bit / Bit Menos Significativo

MD5 Message Digest Algorithm 5/ Algoritmo de Resumen de Mensaje 5

MIW Medical Image Watermarking / Marcado de Imágenes Médicas

MRI Magnetic Resonance Imaging / Imagen de Resonancia Magnética

MSE Mean Squared Error / Error Cuadrático Medio

MSSIM Modified Structural Similarity Index / Indice de Similaridad Estructural Modificado

PACS Picture Archiving and Communication System / Sistema de Archivo y Transmisión de Imágenes

PSNR Peak Signal Noise Ratio / Relación Pico Señal- Ruido

ROI Region of Interest / Región de Interés

RONI Region of Non-Interest / Región de No Interés

SHA Secure Hash Algorithm / Algoritmo de Control Seguro

SHV Sistema Humano Visual

SSIM Structural Similarity Index / Índice de Similaridad Estructural

TCP/IP Transmission Control Protocol/Internet Protocol- Protocolo de control de Internet/ Protocolo Internet

TW Transformada Wavelet

RESUMEN

En las últimas dos décadas el marcado de agua digital ganó un lugar como tema de investigación aplicada en la comunidad científica. Solo o con mayor frecuencia en combinación con técnicas criptográficas brinda solución a problemas de seguridad tales como la pérdida de integridad de los datos digitales o la necesidad de autenticar los archivos. Se aplica en contenidos digitales multimedia tales como textos, audio, imágenes o videos. Las técnicas de marcado de agua digital encuentran un amplio ámbito de aplicación en el área de la Medicina, donde las imágenes médicas se han convertido en un elemento de extrema importancia a la hora de efectuar un diagnóstico y se han desarrollado aplicaciones que envían dichas imágenes a distancia posibilitando la telediagnosis, por lo que es necesario tomar medidas de seguridad para preservar sus datos de alteraciones sean estas intencionales o no.

El desafío de los algoritmos de inserción de marcas de agua robustas es fundamentalmente lograr la imperceptibilidad de la marca, especialmente en las zonas de interés, y proteger la imagen contra recortes y compresiones, manteniendo al mismo tiempo la capacidad necesaria según el objetivo a alcanzar. La inserción de la marca con métodos irreversibles trae aparejada la degradación de la imagen, sin recuperación posible de la original. Las técnicas reversibles en cambio, permiten recuperar el objeto original y son de interés en imágenes legales, médicas y artísticas.

Esta tesis presenta como novedad la aplicación de técnicas de inteligencia artificial, en particular los algoritmos genéticos, en un algoritmo de marcado reversible adecuado para imágenes fijas en escala de grises. El empleo de algoritmos genéticos permite la automatización del método y la optimización del resultado. La reversibilidad determina que sea particularmente apto para su empleo en imágenes médicas, donde interesa la capacidad del método así como recuperar la imagen original a partir de una imagen marcada de calidad aceptable. En una etapa final, se compara el algoritmo introducido con otro que utiliza algoritmos genéticos y se determina que el presentado es de implementación más simple y eficaz.

ABSTRACT

In the last two decades the watermarking has gained a place as research topic in the scientific community. The watermarking by itself or more often combined with cryptographic techniques provides a solution to security problems such as loss of integrity of digital data or when authentication of files is required. It is applied in multimedia digital content such as text, audio, images or videos. The watermarking techniques can be widely applicable in the field of medicine, where medical images have become an extremely important element when making a diagnosis and applications send these images remotely what enables telediagnosis, making necessary to take security measures to preserve data from changes, whether they are intentional or not.

The challenge of robust watermarking algorithms in images is fundamentally achieve imperceptibility of the mark, specially in areas of interest and protect the images from trimming and compression, keeping the required capacity under the goal to reach. The mark insertion methods when irreversible bring image degradation and no return of the original. The reversible techniques allow to recover the original object, that's why they are of great interest in artistic, legal and medical images.

The novelty of this thesis lies in the application of artificial intelligence techniques, genetic algorithms, in a reversible watermarking algorithm suitable for still images in grayscale, which enables automation of the method and optimization of the result. Reversibility makes this algorithm suitable for use in medical imaging where the interests are the capacity and the recovery of the original image from a watermarked image of acceptable quality. Finally, another reversible algorithm using genetic algorithms is presented, both algorithms are compared and it is determined that the first one is simpler and more effective than the other.

RÉSUMÉ

Pendant les deux dernières décennies, le tatouage numérique a gagné une place en tant que sujet de recherche appliqué dans la communauté scientifique. Les algorithmes de tatouage numérique par eux-mêmes ou plus fréquemment associés aux techniques cryptographiques offrent une solution aux problèmes de sécurité tels que la perte de l'intégrité des données numériques ou la nécessité d'authentifier les fichiers. Ils sont appliqués dans le contenu numérique multimédia tels que textes, audio, images ou vidéos. Dans le domaine de la médecine des images médicales sont devenues un élément extrêmement important du diagnostic et des applications envoient ces images à distance permettant le télédiagnostique, par conséquent il faut prendre des mesures de sécurité pour empêcher les modifications de données, qu'elles soient intentionnelles ou non, donc ces techniques de tatouage numérique trouvent leur application.

Le défi des algorithmes de tatouage numérique robuste est fondamentalement atteindre l'imperceptibilité de la marque dans les zones d'intérêt et protéger l'image contre les coupures et la compression, en gardant la qualité requise par le but à atteindre. Les méthodes d'insertion irréversibles entraînent dégradation de l'image, et pas de retour de l'original. Les techniques de tatouage réversibles permettent de récupérer l'objet original, donc ils sont d'intérêt pour marquer les images juridiques, artistiques et médicaux.

Cette thèse présente un algorithme de marquage réversible appropié pour les images fixes en niveaux de gris et dont la nouveauté réside dans l'utilisation des techniques d'intelligence artificielle, les algorithmes génétiques, que permettent l'automatisation du méthode et l'obtention du meilleur résultat. La reversibilité du procédé le rend particulièrement adapté pour une utilisation dans l'imagerie médicale, parce qu'il permet de récupérer l'image originale et d'obtenir une image tatouée de qualité acceptable. Dans une dernière étape, on présente un autre algorithme réversible utilisant algorithms génétiques, on compare les deux algorithmes, on détermine que le premier algorithme est le plus simple et le plus efficace.

INTRODUCCIÓN

El watermarking o marcado de agua digital es la técnica de embeber información en un contenido digital conocido como “host” o “anfitrión” con el objetivo de protegerlo contra la manipulación o uso ilegal [1], que contrasta con los métodos criptográficos puesto que en este caso la información secreta es imperceptible. El nombre de marcas de agua lo toma de los procedimientos de marcado de billetes cuyo objetivo inicial era probar su autenticidad y esta fue la designación elegida por Emil Hembrooke quien ideó y patentó en 1954 un método de embebido de códigos inaudibles en señales de música para verificar propiedad [2]. El auge del watermarking digital comenzó unos años después, en la década de 1990, siendo uno de sus precursores más conspicuos Ingemar Cox con su técnica “spread spectrum” [3], luego se desarrollaron otras técnicas en el dominio DCT (Transformada Discreta de Coseno) y también en el de la transformadas wavelets, destacándose los trabajos de Barni, Kundur y otros [4] [5]. En el caso del marcado de papel, ideado en el Siglo XIII, en el fondo del molde empleado para formar el papel se colocaban hilos metálicos cuyo relieve provocaba un menor espesor en la pasta de celulosa. Una vez finalizado el proceso, era visible colocando al trasluz el papel la marca o filigrana que dejaban estos hilos. En el mundo de la información digital, la marca de agua es un mensaje que se oculta en un objeto principal y que contiene datos sobre este.

El propósito inicial de las marcas de agua digitales fue autenticar (es decir informar acerca del origen y propietario del objeto digital anfitrión), luego se fue ampliando para incluir registro de metadatos, protección contra copias y rastreo de las mismas [6]. En un comienzo, el marcado abarcó audio, imágenes fijas, video, textos y luego se extendió a circuitos integrados y bases de datos relacionales, entre otros [7] [8] [9].

Junto con la esteganografía el watermarking constituye lo que se conoce en general en la literatura como “information hiding” [10], se diferencian en que en

esta última la información embebida no está relacionada con el objeto contenedor. La esteganografía tiene como objetivo establecer una comunicación secreta o encubierta entre dos partes, el mensaje es independiente del host. En la criptografía se conoce que existe la comunicación, pero no se comprende el mensaje.

Watermarking y Criptografía

Si bien durante estas dos últimas décadas se han alzado algunas voces desde el ámbito de la criptografía clásica en contra del watermarking, en particular porque las primeras técnicas no eran reversibles y se somete a degradación el objeto anfitrión, la comunidad ha respondido desarrollando nuevos y mejores métodos y sigue siendo un tema abierto y activo [11]. Empresas como IBM, Digimarc, Philips, Cinea, Verimatrix, han incorporado tecnologías de watermarking y se ha conformado la DWA (Digital Watermarking Alliance) [12], conjunto de compañías entre las que se incluyen algunas de las empresas nombradas, y otras más que emplean y tienen interés en promover técnicas de watermarking, ofreciendo soluciones para seguridad de este tenor en el mercado multimedia.

Tanto la criptografía como el watermarking pueden autenticar y asegurar la integridad de los datos.

En cuanto a las diferencias, en las técnicas criptográficas clásicas se agrega un resumen para asegurar la integridad de los datos, este resumen es fácilmente separable del archivo original, el embeberlo le agrega un nivel más de seguridad. De igual forma aumenta la seguridad si se embebe la información necesaria para autenticar en lugar de mantenerla en una cabecera o cola separable. La imagen de una cámara de seguridad , por ejemplo, debe estar autenticada para servir como prueba en caso de litigio y dicha autenticación no puede estar separada de la imagen en sí.

En lo que respecta a la confidencialidad, el watermarking solo asegura la de los datos embebidos, el resto se mantiene igual, mientras que la criptografía puede asegurar toda la información.

De acuerdo a lo expresado, entre los puntos a favor de las marcas de agua se tiene que:

1) No necesita definir un formato especial tal como cabecera o cola para transportar los datos que agregan seguridad
2) Algunas técnicas pueden llegar a determinar en qué zonas se produjo la alteración o ataque.

Imágenes Médicas y Telemedicina

En el área de la medicina actualmente se generan grandes volúmenes de información en forma de imágenes digitales, como Tomografías Computarizadas (CT), Imágenes de Resonancia Magnética (MRI), entre otras. En Argentina, la Ley 26529 [13], referente a los derechos del paciente en su relación con los profesionales e instituciones de salud, sancionada el 21 de octubre de 2009 y publicada en el Boletín Oficial el 20 de noviembre del mismo año, en su artículo 18 prescribe la guarda por un tiempo de diez años de la historia clínica del paciente, contados a partir de la última intervención en la misma. Durante su tiempo de vida, los datos deben ser protegidos, ningún usuario no autorizado puede tener acceso a los mismos y se debe resguardar la privacidad de toda información sobre el paciente.

Por otra parte, las aplicaciones de la telemedicina: teleconsulta, telediagnosis, telecirugía están incrementando su importancia en el área de salud. El intercambio de información a través de las redes abiertas, obliga a incrementar la seguridad para asegurar la integridad de los datos médicos y del paciente. Así, tanto los datos almacenados como los transmitidos deben ser protegidos. La telemedicina que comenzó a desarrollarse con más fuerza en 1970, utilizó en sus comienzos telefonía convencional, y luego banda ancha, Internet, redes móviles inalámbricas. Estas últimas son menos seguras y exigen utilizar métodos criptográficos, para mantener la privacidad de los datos del paciente. En Estados Unidos, país pionero, la alta densidad demográfica y el elevado grado de especialización en los profesionales de la Medicina hacen que se utilice ampliamente, pero también se ha extendido en otros países, incluido el nuestro. El soporte físico actual son redes satelitales y en menor grado de fibra

óptica con enlaces domiciliarios inalámbricos. En la Argentina, hospitales como el Garrahan de Buenos Aires utilizan servicios de telemedicina. En la Patagonia, de baja densidad demográfica y grandes distancias entre centros especializados, ya se está aplicando exitosamente.

Ante el crecimiento del almacenamiento y transmisión de información digital en medicina, surgió como respuesta un estándar: el DICOM (Digital Information and Communication in Medicine), cuya primera versión importante es la 3.0, la que data de 1993. Para comunicaciones utiliza un protocolo de capa de aplicación que corre sobre el extendido estándar de facto TCP/IP (Transmission Control Protocol/Internet Protocol). DICOM ha sido ampliamente adoptado por centros médicos que usan equipos, servidores y estaciones de trabajo que tienen una *"conformance statement"* (declaración de conformidad) Dicom. La National Electrical Manufacturers Association (NEMA) almacena en sus páginas el estándar DICOM [14].

Este estándar DICOM que permite la transmisión a distancia de imágenes médicas fue pensado inicialmente para una transmisión punto a punto, con lo que no corría riesgos respecto a la privacidad de los datos. La particularidad de las imágenes en formato Dicom es que contienen en su cabecera la información del paciente, por lo que si un usuario malintencionado accede a la misma dispone de ella. En su parte 15, la norma establece perfiles de seguridad que se pueden considerar por separado, entre ellos considera la encriptación de los datos de la cabecera, la autenticación al establecerse una conexión y el aseguramiento de la integridad. La cabecera puede estar encriptada, pero puede ser separada por un usuario malintencionado del resto de la información. En el caso de usarse redes abiertas, no son suficientes las medidas de seguridad adoptadas.

Imágenes Médicas y Watermarking

Como alternativa o como refuerzo de los métodos de encriptado se comenzó a investigar la aplicación de las técnicas de marcas de agua o watermarking en imágenes médicas. Ya en el año 2000 se destacaba el aporte que podía hacer el watermarking al aseguramiento de la integridad, autenticación y

confidencialidad de los datos almacenados en imágenes médicas [15], se estudiaba en particular el almacenamiento de la información sobre el paciente en dichas imágenes [16], y se analizaba especialmente la degradación que producían las marcas de agua [17] [18]. Una de las primeras técnicas de watermarking aplicada a imágenes médicas fue desarrollada por Wakatani y contemplaba el embebido de la firma digital en la RONI (Región de No Interés) alrededor de la ROI (Región de Interés) [19], luego Zain, Baldwin y Clarke desarrollaron una técnica reversible con el objeto de autenticar la imagen [20], De Rosa, Cappellini, Bartolini y Piva, un método para insertar metadatos [21], Memon junto con Gilani y Ali formularon una aplicación especial que tiene como objetivo asegurar tomografías computadas del tórax [22] y, finalmente, son muchos los que proponen sistemas de marcado con múltiples aplicaciones: autenticación, aseguramiento de integridad e inserción de metadatos [23] [24] [25]. Este interés no ha cesado hasta ahora, tiene especialmente en cuenta la telemedicina [26] [27] siendo el tema objeto de investigación que cuenta con soporte de organismos oficiales en distintos países, incluidos los de la Unión Europea [28] [29] [30].

Objetivos de la Tesis

En esta tesis se plantean con respecto a las imágenes médicas los objetivos que se detallan a continuación.

Objetivo Principal

- Establecer una técnica de marcado de propósito múltiple aplicable a imágenes médicas que permita detectar adulteraciones, identificar origen e incluir datos identificatorios confidenciales, sin introducir distorsión visible.

Objetivos Específicos

- Analizar comparativamente técnicas de marcado frágil en imágenes médicas.

- Analizar técnicas criptográficas a aplicar en la información sobre el paciente que se debe embeber.
- Presentar variantes a técnicas en uso que puedan mejorar su desempeño.

Estructura del Informe

En el capítulo 1 se presentan fundamentos teóricos de marcas de agua, sus propiedades, aplicaciones y principales líneas de métodos reversibles, en el 2 se describen variantes de técnicas reversibles, por lo tanto especialmente aplicables a imágenes médicas, legales y artísticas, se proponen mejoras a las mismas, en el capítulo 3 se analizan los fundamentos teóricos de dichas mejoras, en el 4 se presentan resultados experimentales y en el capítulo 5 se enuncian conclusiones y se indican futuros trabajos.

CAPÍTULO 1

ASPECTOS TEÓRICOS Y ANTECEDENTES

1.1 Marcas de Agua

Una marca de agua es información que se embebe en un objeto anfitrión (texto, audio, imágenes, videos), la que siendo imperceptible (inaudible o invisible según corresponda) para el ser humano puede ser detectada por una computadora. La principal ventaja de este método en lo que hace a su contribución a la seguridad de la información es que no es fácilmente separable del objeto principal, como sucede cuando se almacenan en forma separada metadatos asociados a un archivo o firmas digitales. En el caso de imágenes o videos, también se consideran marcas los logos que se insertan para evitar reproducciones ilegales, en este caso visibles y obtenidas por medio de técnicas especiales de fusión.

En las técnicas de ocultamiento de información se reconocen dos pasos:

1) Embebido de un mensaje en el host.
2) Detección o extracción del mensaje.

En el segundo paso, se puede recuperar archivo original (en el caso de las técnicas reversibles) y mensaje, o bien simplemente detectar la presencia del mensaje (en el caso de técnicas de embebido irreversibles).

En esteganografía, que se diferencia del watermarking exclusivamente en que la información embebida no está relacionada al objeto anfitrión, siempre interesa extraer el mensaje. En la Figura 1.1 se muestra un esquema reversible

de ocultamiento de información. Existe una clave que deben conocer emisor y destino que determina dónde embeber y luego cómo detectar la información embebida. El tercero o "guardián" no ve la información encubierta, solo puede sospechar que existe. Por el contrario, cuando se usan solamente métodos criptográficos un tercero sabe que hay un mensaje, pero no lo comprende.

En general, las técnicas de marcado de agua digital no se usan en forma aislada sino como complemento de técnicas criptográficas. Así, la información embebida para asegurar identidad o para insertar metadatos ha sido transformada por métodos criptográficos o bien se trata de un resumen obtenido mediante funciones hash cuando el objetivo es asegurar la integridad del archivo anfitrión.

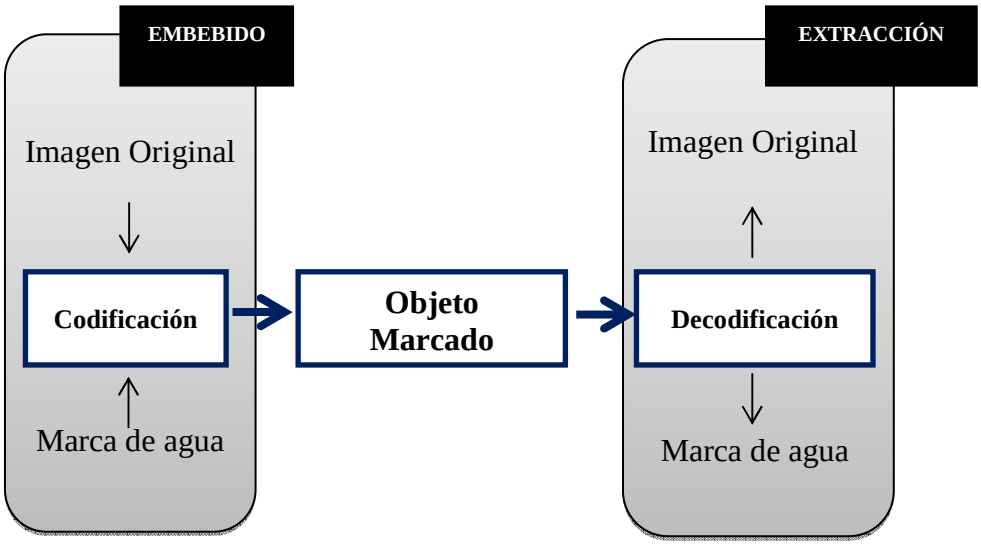

Figura 1.1 – Esquema de Ocultamiento de Información

Las Figuras 1.2.a y b muestran cómo se asegura un sistema PACS (Picture Archiving and Communication System / Sistema de Archivo y Transmisión de Imágenes), utilizado en almacenamiento y transmisión de imágenes médicas.

En la Figura 1.2.a se muestra el caso en que el encriptado de la cabecera se produce en el mismo momento de obtención de la imagen. En la Figura 1.2.b, este se produce posteriormente. El primer método es el más conveniente, ya que en el segundo caso la imagen se traslada por canales no seguros.

En los esquemas que se plantearán se considera la marcación de la imagen posteriormente al encriptado de la información.

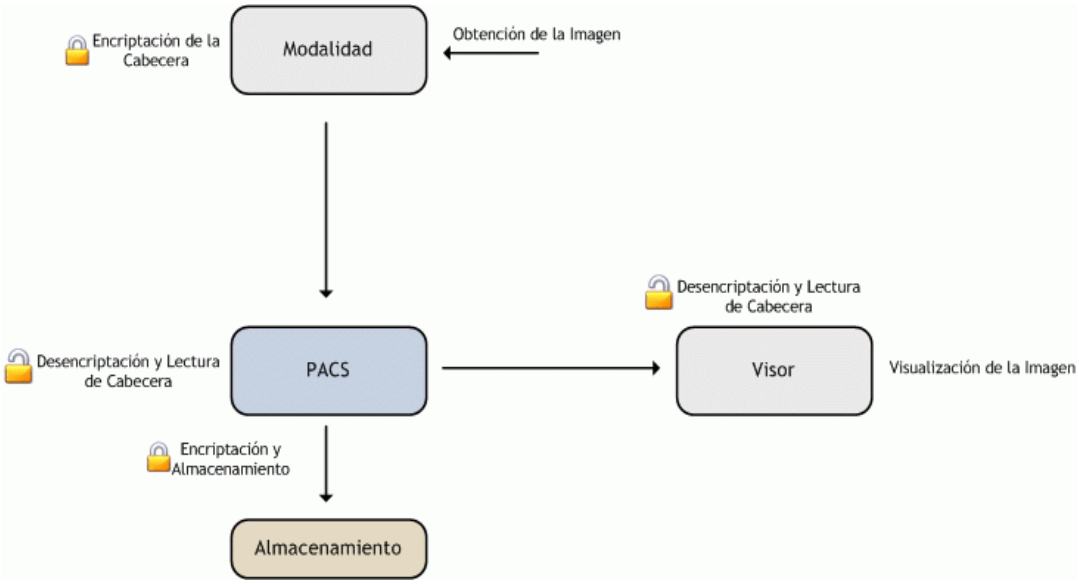

Figura 1.2. a- Seguridad en PACS – Canal con Imagen Segura

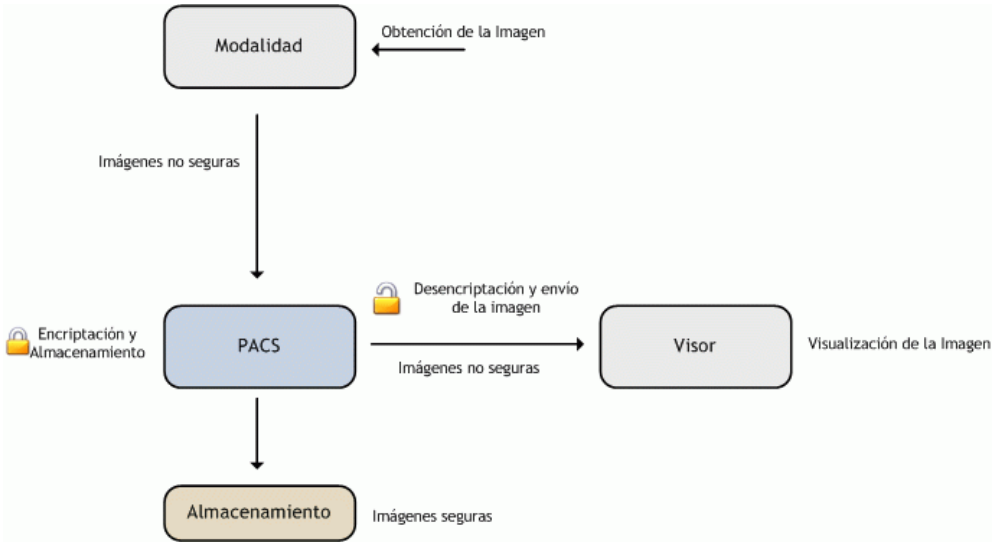

Figura 1.2. b – Seguridad en PACS- Canal con Imagen No Segura

1.2 Clasificación de las Marcas de Agua

Las marcas de agua se pueden clasificar según diferentes criterios.

1) Según la reacción ante los ataques,. sin importar si estos son intencionales o no, las marcas se clasifican en:

a) Robustas.

b) Frágiles

c) Semifrágiles

Las marcas de agua robustas deben resistir todo tipo de ataques, detectándose incluso después de los mismos. Sirven para proteger los derechos de autor. No se puede tolerar la eliminación de derechos de autor, por ejemplo, por deformaciones geométricas, rotación, escalado o compresión.

Las marcas de agua frágiles son aquellas que quedan eliminadas o modificadas y dejan de cumplir su función en caso de ataque. La incapacidad de recuperarlas, revela que se produjo algún ataque y ese es el objetivo buscado. No toleran ninguna transformación, ni siquiera las más comunes en procesamiento de imágenes. Se utilizan para asegurar integridad.

Las marcas de agua semifrágiles permiten sobrevivir a cierto tipo de alteraciones, como compresión sin pérdidas, pero deben destruirse ante cambios importantes, no reversibles. Una marca de este tipo puede consistir en extraer información de la Región de Interés (ROI) a la que se dividió en bloques y luego embeberla en la Región de No Interés (RONI). Si luego se prueba que algún bloque de la ROI fue modificado se puede recuperar la información con los datos embebidos en la RONI, e incluso determinar qué bloque fue alterado.

2) Según la necesidad de poseer la propia marca de agua para decidir si el archivo es auténtico o no, las técnicas de marcado se clasifican en:

a) Ciegas

b) No ciegas o informadas

c) Semiciegas

En un comienzo la marcación era informada, había que poseer la marca original, compararla con la detectada y decidir si coincidían, si era así quedaba probada la propiedad del objeto digital. En las ciegas, preferidas actualmente en la mayoría de los casos, no se precisa contar con la marca de agua para determinar la autenticidad del archivo. En las semiciegas se precisan algunos datos para recuperar la marca, por ejemplo en el caso de imágenes, su histograma.

3) Según el dominio en el que es insertada una marca, ya que pueden embeberse:

a) En el dominio espacial

b) En el dominio de la transformada, sea usando la DCT (Transformada Discreta de Coseno), la Transformada de Fourier, o bien algunas de las Transformadas Wavelet.

En el comienzo se utilizó el dominio espacial, embebiendo en el LSB (Least Significant Bit/ Bit Menos Significativo) según una clave. Esta técnica pronto se reveló ineficiente porque no era imperceptible y se comenzaron a emplear transformadas, insertando la marca en el dominio de la frecuencia.

4) Según el objetivo de la marca, se consideran tres grandes grupos:

a) Marcas para proteger derechos de autor o autenticar.

b) Marcas para insertar metadatos,

c) Marcas para asegurar la integridad del objeto, es decir que este no cambió.

1.3 Propiedades de las marcas de agua

Las cualidades que se persiguen en un marcado de agua digital son:

a) Capacidad.

b) Imperceptibilidad

c) Robustez

Capacidad: se refiere a la cantidad de información que se puede embeber. Se mide en bpp (bits por píxel). La capacidad necesaria depende de la aplicación en particular.

Imperceptibilidad: la marca no debe ser visible, salvo en el caso de los logos que constituyen un tipo especial de información embebida para autenticación y protección contra copias.

Robustez: es la cualidad de persistir pese a ataques intencionales o daños colaterales.

En general se debe buscar una solución de compromiso entre estos requisitos, ya que, por ejemplo, la marca será más imperceptible mientras menos bits contenga, si se quiere aumentar la capacidad necesariamente se tendrá que sacrificar imperceptibilidad. La robustez no es un requisito en el caso de marcas de agua frágiles.

El triángulo que se presenta en la Figura 1.3 refleja la interrelación entre propiedades en cada caso. Es un "triángulo mágico" ya que si se establece uno de los requisitos ya no hay libertad para elegir los otros dada la relación entre ellos. El caso del watermarking frágil es similar a la esteganografía segura. Si se aumenta la capacidad será a costa de sacrificar la imperceptibilidad.

Otra cualidad que se persigue en los algoritmos es tener una baja complejidad computacional en los procesos de embebido y extracción

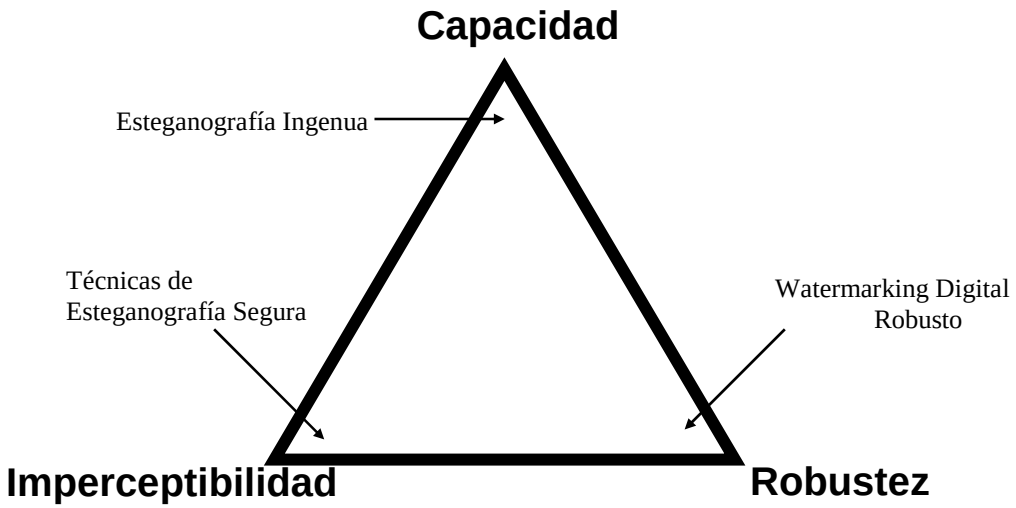

Figura 1.3- Relación entre Propiedades de Marcas- "Triángulo Mágico"

1.4 Evaluación de la Imperceptibilidad

La imperceptibilidad se mide de manera subjetiva (por parte de usuarios humanos) y objetivamente, a través del MSE (Mean Square Error/ Error Medio Cuadrático), el PSNR (Peak Signal Noise Relation/ Relación Pico Señal Ruido),

y el SSIM (Similarity Structural Index / Índice de Similaridad Estructural) indicados por las ecuaciones (1.1), (1.2) y (1.3), respectivamente. En estas fórmulas, M y N se refieren a la cantidad de filas y de columnas, respectivamente, de la matriz imagen original; I_w, I, e I_{pico} designan la intensidad de imagen marcada, la original y el valor pico, indicando i y j la posición del píxel en particular.

$$MSE = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} (I_w(i,j) - I(i,j))^2}{MN} \tag{1.1}$$

$$PSNR = 10\log \frac{(I_{Pico})^2}{(MN)^{-1}\sum_{i=1}^{M}\sum_{j=1}^{N}(I_w(i,j) - I(i,j))^2}\, dB \tag{1.2}$$

$$SSim(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_x + C_2)}{(2\mu_x^2\mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{1.3}$$

Se considera que una imagen cuya calidad está siendo evaluada es la suma de una señal de referencia sin distorsión y una señal de error. Un criterio ampliamente adoptado es que la calidad perceptual está dada por la visibilidad de la señal error. La interpretación más simple de este concepto es MSE que cuantifica la fuerza de la señal error. PSNR, por otro lado relaciona MSE con el valor pico de la señal.

En la métrica SSIM, reflejada en la ecuación (1.3), los autores buscaron tener en cuenta las características del SHV (Sistema Humano Visual) [31] y consideraron que la estructura de una imagen no depende de la iluminación la que debe ser aislada del resto. Dadas dos imágenes *x* e *y*, se calcula la media de la luminancia x (1.4), designada como μ, la desviación estándar (raíz cuadrada de la varianza) (1.5), y la covarianza (1.6) donde N es la cantidad de píxeles. Se comparan la luminancia, el contraste y la estructura de las ambas imágenes según indican (1.7), (1.8) y (1.9).

$$\mu_x = \frac{\sum_{1}^{N} x_i}{N} \tag{1.4}$$

$$\sigma_x = \sqrt{\frac{1}{N-1}\sum_{1}^{N}(x_i - \mu_x)^2} \tag{1.5}$$

$$\sigma_{xy} = \sum_{1}^{N}\frac{(x_i - \mu_x)(y_i - \mu_y)}{N-1} \tag{1.6}$$

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x{}^2 + \mu_y{}^2 + C_1} \tag{1.7}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x{}^2 + \sigma_y{}^2 + C_2} \tag{1.8}$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{1.9}$$

Finalmente, se considera SSIM (1.10) como el producto de (1.7), (1.8) y (1.9), donde los exponentes tienen en cuenta la importancia que se da a cada uno de los tres factores y se pueden considerar iguales a la unidad.

$$SSIM(x,y) = \left(l(x,y)\right)^{\alpha} * \left(c(x,y)\right)^{\beta} * \left(s(x,y)\right)^{\gamma} \tag{1.10}$$

Los valores no son obtenidos sobre toda la imagen sino localmente, sobre ventanas cuadradas que se desplazan sobre toda la imagen. Finalmente se calcula el valor medio, llamado en la literatura MSSIM (SSim Modificado).

Las constantes C_i incluidas en la fórmula se introducen para salvar la inestabilidad de la medida cuando los valores del denominador son débiles. Se relacionan con el rango dinámico de grises de la imagen designado como L por medio de constantes k_i. según se indica en 1.11, donde n es la cantidad de bits por píxel.

$$C_i = (K_i L)^2 \qquad L = 2^n - 1 \tag{1.11}$$

Si se toma una de las imágenes como patrón, los índices indican la distorsión de la segunda respecto a la primera. Dos imágenes pueden tener el mismo MSE pese a que la distribución de las distorsiones sea muy diferente. Cuando la distorsión se debe a los contornos de los objetos, el MSIMM es muy bajo, lo que está de acuerdo al Sistema Humano Visual.

1.5 Aplicaciones de las Marcas de Agua

Es importante recordar los dos tipos fundamentales de marcas de agua ya definidos, dado que estos tienen distintas aplicaciones:

1. las robustas
2. las frágiles

Aplicaciones de las Marcas de Agua Robustas

o Se usan con fines de acreditar derechos de autor, impedir copias, servir como huellas digitales en las transacciones (fingerprintings).

- Debe ser posible detectarlas incluso después de sufrir un ataque severo.
- El objetivo de un atacante es conseguir que el detector no pueda encontrar la marca de agua en el archivo anfitrión.

Aplicaciones de las Marcas de Agua Frágiles

o Se utilizan para asegurar el origen y la integridad del medio marcado en multimedia. En las mismas el objetivo de la marca es detectar si el archivo anfitrión (host) ha sido modificado.

- En un sistema de marcas de agua frágil después de realizarse modificaciones en el archivo la marca se destruye o cambia, y la falta o variación de la marca descubre la falsificación.

1.6 Ataques a las Marcas de Agua

Se considera ataque a una marca de agua a toda operación que tenga como fin impedir que se cumpla el objetivo para el que se introdujo dicha marca en la imagen. En general toda distorsión que reciba la marca de agua es considerada ataque [32] [33].

Los ataques pueden ser:

· ***Intencionales u hostiles***

· ***No intencionales o colaterales***

Los ataques no intencionales son aquellos que derivan de operaciones normales producto del procesamiento necesario de imágenes, tales como compresión, mientras que los intencionales tienen como único fin debilitar, quitar o al menos alterar la marca de agua. La compresión de imágenes con pérdida es la forma más común de ataque no intencional que debe soportar la marca. Esta compresión tenderá inevitablemente a descartar información perceptualmente irrelevante, tal como es la de la propia marca de agua. Otros ataques no intencionales son el recorte y el escalado.

Los ataques a las marcas de agua admiten, entre otras, la siguiente clasificación:

- **Ataques Simples.** Son los que agregan distorsión a la imagen de forma tal que la marca se vuelve indetectable. No se intenta separar la marca. Es un ataque exitoso si la imagen es todavía inteligible y puede usarse para un propósito particular. Puede producirse por una compresión con pérdidas, agregado de ruido gaussiano, entre otros.
- **Ataques que inhabilitan la detección, sin atacar directamente la marca.** Estos ataques intentan romper la detección de correlación entre la marca extraída y la original. Una forma de lograrlo es dejar los píxeles iguales en la imagen original y la atacada, pero cambiarles la ubicación. Se pueden clasificar, a su vez, en:
    - **Ataques geométricos**. La imagen sufre translación, rotación o cambio de escala.
    - **Ataques de sincronismo o jitter**. Impiden encontrar la ubicación de la marca de agua, agregando filas o columnas de píxeles, por ejemplo.
    - **Ataques ambiguos.** Desacreditan la autoridad de la marca de agua, por ejemplo embebiendo por lo menos una marca adicional o posibilitando copiar la marca de una imagen a otra sin ningún control del dueño legítimo de la marca.
- **Ataques que remueven la marca**. Intentan separar y remover la marca. Ejemplos de estas técnicas son:

- Ataque de colusión (mezcla de varias copias que contienen marcas, las promedia y así desaparecen aumentando la calidad de la imagen).
- Denoising (quita el “ruido”).
- Filtrado no lineal.

Una marca de agua robusta debe resistir un ataque no permitiendo que se altere el valor de los datos que protege. En general, también se cumple que quien quiere remover la marca de agua no desea alterar el valor de los datos protegidos por la misma. En este sentido coinciden quienes colocan la marca de agua y quienes la desean extraer. Siempre hay que estudiar qué posibilidades tiene el atacante para, de acuerdo a esto, tomar contramedidas que protejan la marca.

1.7 Watermarking de Imágenes Médicas

El marcado de imágenes médicas está cobrando tanta importancia que en la literatura ya se lo distingue con una sigla: MIW (Medical Image Watermarking). Los métodos de diagnóstico tradicionales están siendo reemplazados por telediagnóstico, particularmente útil en lugares alejados de los grandes centros poblacionales. Un médico toma imágenes, hace su diagnóstico y la envía con información del paciente, del diagnóstico y suya personal a otro médico para tener una segunda opinión. El médico receptor hace su propio diagnóstico y lo informa en un archivo. Toda esta información se almacena en archivos que contienen los datos históricos del paciente. Si el intercambio de información se hace a través de un medio no seguro como Internet es necesario considerar como necesidades a satisfacer, las siguientes:

a) Autenticación: debe existir una prueba de que la información pertenece al paciente correcto y proviene de la fuente correspondiente.

b) Integridad: la información no debe haber sido modificada por usuarios no autorizados.

c) Confidencialidad: solo los usuarios autorizados deben tener acceso a la información.

Se debe considerar que en la actualidad, fácilmente se copian y modifican las imágenes. Esto también puede suceder con las imágenes médicas, y la copia o adulteración puede tener propósitos ilegales como cobrar un seguro, así como puede afectar tratamientos. En general, el marcado de imágenes médicas tiene dos objetivos fundamentales:

a) Insertar metadatos.

b) Insertar información que autentique la imagen y la proteja de adulteraciones.

El marcado de imágenes médicas encuentra más limitaciones que el de otro tipo de imágenes. El marcado puede esconder información que sea relevante para el médico. Para evitarlo, los métodos MIW siguen dos grandes lineamientos: unos tratan de preservar la zona de diagnóstico (ROI) y realizan el embebido en la RONI, que no es útil. Otros se inclinan por el marcado reversible, donde es posible recuperar la imagen original. También se pueden utilizar los dos caminos en forma simultánea.

En este trabajo en particular, el estudio se centra en marcas de agua reversibles o frágiles. Se supone que la imagen no será sometida a ninguna modificación posterior tal como compresión, escalado o recorte que involuntariamente no permitiera recuperar la marca. Cualquier modificación voluntaria o no, impedirá recuperar la imagen original, lo que es indeseable.

1.8 Marcas de Agua Reversibles

Las marcas de agua reversibles son aquellas en las que es posible recuperar la marca y el archivo original. Esto evita los inconvenientes de degradación de la imagen producida por la marca, ya que en caso de ser necesario se puede disponer del archivo original. Al mismo tiempo, se pretende que en la imagen

marcada se mantenga la calidad para que en la mayoría de las aplicaciones sea usable.

Los métodos reversibles son de interés para marcar imágenes artísticas y otras con valor legal como las de uso militar y médico. Su desarrollo adquirió un fuerte impulso a partir del año 2000.

Las técnicas reversibles modernas se pueden agrupar en dos grandes categorías:

1) Expansión de la diferencia (DE). Fue iniciada por Tian [34] en 2003 y sobre ella se han propuesto distintas mejoras que se tratarán en el siguiente capítulo. Es un método que, convenientemente modificado, brinda capacidad, pero a costa de calidad.
2) Corrimiento del histograma. Esta línea fue iniciada por Ni *et al.* [35] en 2006. Aporta poca capacidad, pero se obtienen imágenes marcadas de buena calidad.

Para obtener reversibilidad es necesario insertar información extra, lo que hace que, más allá de la capacidad nominal propia del método, sea necesario calcular la capacidad efectiva. Así, por ejemplo, los métodos primitivos de marcado de agua utilizaban el bit menos significativo para embeber la marca. Una técnica no reversible reemplazaba simplemente el bit por la marca. Una técnica reversible comprimía sin pérdidas el plano del bit menos significativo y en el espacio logrado insertaba la marca. Esto hace necesario almacenar la información necesaria para poder revertir la compresión, una vez extraída la marca.

1.8.1 Método de Expansión de la Diferencia (DE) de Tian

Es un método de inserción sustitutiva.

Consiste en la aplicación de un filtro pasabajo que calcula el promedio entre dos píxeles consecutivos (se toma la parte entera) y un filtro pasaalto que calcula la diferencia. Esta operación se conoce como transformada entera Haar y es inversible, según se observa en (1.12) y (1.13), donde x e y simbolizan la

intensidad de dos píxeles consecutivos, y *n* la cantidad de bits necesarias para indicar el valor de intensidad.

$$l = \left\lfloor \frac{x+y}{2} \right\rfloor \quad h = x - y \quad 0 \le x, y \le 2^n - 1 \tag{1.12}$$

$$x = l + \left\lfloor \frac{h+1}{2} \right\rfloor \quad y = l - \left\lfloor \frac{h}{2} \right\rfloor \tag{1.13}$$

A las diferencias obtenidas, se las puede clasificar en dos grupos: expansibles y cambiables. Sea *n*=8 y *b* un bit de la marca, si al desplazar un bit a la izquierda el valor de la intensidad del píxel y sumar el bit de la marca no se produce desbordamiento se dice que la diferencia es expansible (1.14), si esto no es posible se puede operar según (1.15) y se la designa como cambiable, pero en este caso no se puede recuperar el valor original de intensidad si no se guarda el último bit original.

Diferencia Expansible:

$$h' = \left|2h + b\right| \le \min(2(255 - l), 2l + 1) \tag{1.14}$$

Diferencia Cambiable

$$h' = \left|2 * \left\lfloor \frac{h}{2} \right\rfloor + b\right| \le \min(2(255 - l), 2l + 1) \tag{1.15}$$

En las diferencias expansibles se crea un nuevo LSB sin perder el original, en las cambiables se sustituye el LSB original, por el bit de la marca.

Hace falta guardar un mapa de las ubicaciones de las diferencias expansibles para recuperar el archivo original. Esto trae una apreciable sobrecarga la que dependerá de la imagen anfitriona. Para disminuirla se comprime el mapa con un método sin pérdidas.

Se cumplen algunas propiedades que se pueden aprovechar al implementar el algoritmo:

a) Todo *h* expansible es cambiable.

b) Un *h'* obtenido por expansión es cambiable.

c) Las diferencias *h*=0 y *h*=-1 son expansibles siempre.

Embebido

Se calculan *l* y *h*, eligiendo un sentido de recorrido.

Se forman cuatro conjuntos de elementos, según *h*: *EZ, EN, CN, y NC*

EZ: contiene los *h*=0 o 1.

EN: contiene los expansibles que no están en el conjunto anterior. Se considera un umbral U para evitar degradación visual, y se forman dos subconjuntos, *EN1* y *EN2* indicados en (1.16).

$$EN1 = \{h \in EN : |h| \leq U\}, EN2 = \{h \in EN : |h| > U\} \quad (1.16)$$

CN: contiene los cambiables que no están en los conjuntos anteriores, según (1.17),

$$h\ cambiable, h \notin EZ \cup EN \quad (1.17)$$

NC: contiene el resto, o sea los que no se puede cambiar.

Los cálculos se realizan según la que muestra la tabla 1.1.

Categoría	**Conjunto Original**	**Valor Original**	**Valor en el Mapa L (Location)**	**Nuevo valor**	**Nuevo conjunto**
Cambiable	*EZ* o *EN*1	*H*	*1*	2*h*+*b*	*CH*
	*EN*2 o *CN*	*H*	0	2xfloor(*h*/2)+*b*	*CH*
No cambiable	*NC*	*H*	0	*h*	*NC*

Tabla 1.1 - Algoritmo DE: Clasificación en Conjuntos

Los LSBs de los conjuntos *EN*2 y *CN* deben guardarse para obtener la reversibilidad. Estos forman un vector que se llamará *C*.

El mapa de ubicación *L* (conocido así en la literatura por su nombre en inglés Location Map) que es matricial se vectoriza y se comprime sin pérdida (por ejemplo con codificación aritmética). Al vector obtenido se lo designará

$L_{comprimido}$, También se comprimen sin pérdidas los bits LSBs que se deben almacenar, conjunto al que se llamará $C_{comprimido}$.

Se embebe el mapa comprimido, los LSBs comprimidos y la marca propiamente dicha o carga *P* (designación usual derivada de su nombre en inglés: Payload) .

Se calcula la inversa de la transformada entera, obteniéndose en este paso la imagen marcada.

Extracción

Se calcula la transformada entera, obteniéndose *l'* y *h'.*

Se buscan todos los *h'* cambiables (el *h'* obtenido de un *h* expansible es cambiable). Se extraen los LSBs que constituyen la marca (tanto *L* como *C* y *P*).

De la marca se separan $L_{comprimido}$ y $C_{comprimido}$ (para esto tuvo que guardarse información acerca de la longitud de los campos).

Se descomprimen $L_{comprimido}$ y $C_{comprimido}$.

Se recupera la matriz de ubicación *L* a partir del vector.

Según el mapa *L* se ubican los valores expansibles y se recupera el *h* original mediante la operación *h*=floor(*h*'/2). (floor es la función que indica la parte entera inferior).

Con los LSBs originales se recuperan los *h* cambiables y así queda reconstruida la transformada primitiva.

Se antitransforma, recuperándose la imagen original.

Discusión

La capacidad total del método está dada por el conjunto de las diferencias que pueden almacenar información. La capacidad máxima teórica corresponde a la mitad del total de píxeles o sea que es 0,5 bpp. Se puede aumentar recorriendo la imagen por fila y luego por columna, pero a costa de mayor degradación. Se debe cumplir (1.18).

$$EZ + EN1 \geq L_{comprimido} + C_{comprimido} + P \qquad (1.18)$$

Es común que la carga sea baja, como la de una función hash que puede ser un SHA (Secure Hash Algorithm), el que a lo sumo tiene 512 bits o un MD5 (Message Digest Algorithm 5) que pesa 128 bits.

El tiempo de procesamiento para un mismo hardware y software es más elevado que el que precisa el método de Ni que se describirá en la siguiente sección.

Experimentación

En la Figura 1.4.a se muestra una imagen original en formato dicom y la marcada correspondiente, para la que se obtienen los valores presentados en la Tabla 1.2, realizando dos iteraciones (una por fila y otra por columna), lo que permite llegar a una capacidad mayor a 0,5 bpp, a costa de mayor tiempo de procesamiento.

Análogamente, para la imagen 'lena.pgm' que se muestra en la Figura 1.5, se obtienen los valores de la Tabla 1.3.

Figura 1.4 - Imagen Médica Original y Marcada según Tian

Como regla general, solo se marcan las diferencias que están por debajo de un valor lo que contribuye a mantener la calidad visual de la imagen, aunque repercute negativamente sobre la capacidad del método. En el caso de 'lena' solo se han marcado las diferencias que están por debajo de un valor de 20, fijado empíricamente. El valor del umbral conveniente y la relación de compromiso entre este valor, la capacidad y la calidad visual es un problema que el autor del algoritmo no resuelve.

Figura 1.5 - Imagen 'lena.pgm' - Original y Marcada según Tian

Iteración # 1

Longitud Mapa L = 131072 ------ Longitud del mapa L después de la compresión = 19835
Bits originales = 2842 ------ Bits luego de la compresión = 2798
Capacidad total = 131072 ------ **Longitud de la marca útil** = 108327
Carga o Payload útil = 0.413235 bpp
PSNR = 44.217539 dB

Iteración # 2

Longitud Mapa L = 131072 ------ Longitud del mapa L después de la compresión= 75892
Bits originales = 14570 ------ Bits luego de la compresión = 5518
Capacidad total = 127565 ------ **Longitud de la marca útil**= 46043
Carga o Payload útil = 0.175640 bpp
PSNR = 40.953138 dB

Payload = 0.588875 bpp ----- **Total de bits insertados útiles** = 154370
PSNR Final = 39.678795 dB

Tabla 1.2 - Marcado de Tian - Imagen "cerebro.dcm"

Iteración # 1

Longitud Mapa L= 131072 ------ Longitud Mapa L comprimido = 51603
Bits originales= 10177 ------ Bits luego de la compresión = 10182
Capacidad total = 131072 ------ **Longitud de la marca útil**= 69175
Carga o Payload útil= 0.263882 bpp
PSNR = 36.992557 dB

Iteración # 2

Longitud Mapa L = 131072 ------ Longitud Mapa L comprimido = 49535
Bits originales = 9574 ------ Bits originales luego de la compresión = 9552
Capacidad total= 131061 ------ **Longitud de la marca útil**= 71862
Carga o Payload útil = 0.274132 bpp
PSNR = 35.801350 dB

Carga o Payload total = 0.538013 bpp----- **Total de bits insertados útiles** = 141037
PSNR Final = 32.091080 dB

Tabla 1.3 - Marcado de Tian - Imagen 'lena.pgm'

La Tabla 1.4 presenta los resultados obtenidos al aplicar el algoritmo en un solo recorrido de la imagen en 'lena' (Figura 1.5), 'barbara', 'baboon' y 'boats' (Figura 1.6), mientras que la Tabla 1.5 se refiere a las imágenes médicas que se muestran en la Figura 1.7.

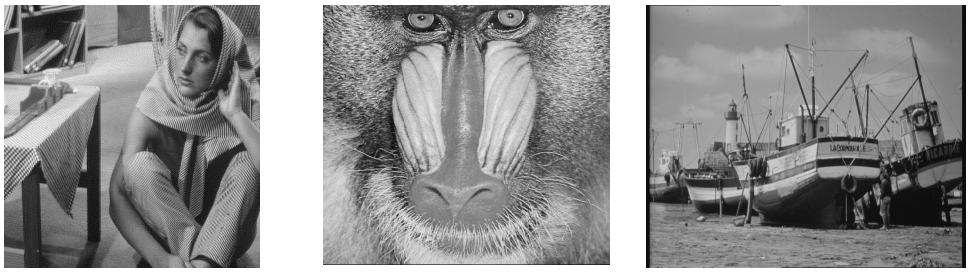

Figura 1.6 - Imágenes 'barbara', 'baboon' y 'boats'

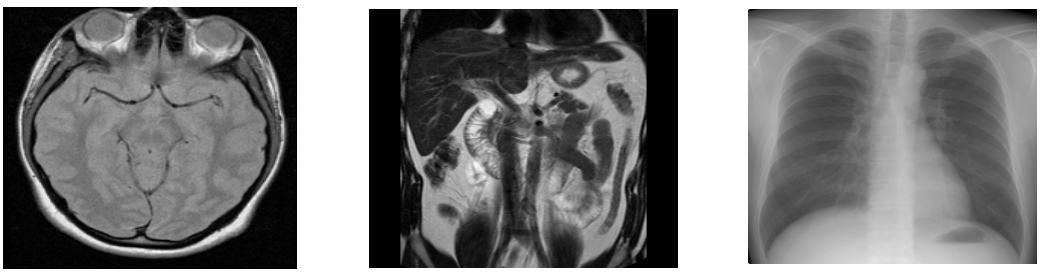

Figura 1.7 - Imágenes 'mr-cerebro', 'mr-abdo' y 'cr-torax'

Imagen	**Umbral**	**Capacidad (bpp)**	**PSNR(dB)**	**MSE**
lena (512x512x8)	20	0,30	37,99	10,31
	50	0,46	34,33	23,95
barbara (512x512x8)	20	0,28	30,88	53,12
	50	0,35	30,75	54,72
baboon (512x512x8)	20	sin capacidad de insertar carga útil		
	50	0,30	29,17	78,75
boats (576x720x8)	20	0,28	38,72	8,73
	50	0,43	34,58	22,65

Tabla 1.4 - Capacidad (bpp) vs. Calidad – Algoritmo Original de Tian

Imagen	**Umbral**	**Capacidad (bpp)**	**PSNR(dB)**	**MSE**
mr-cerebro (256x256x16)	100	0,20	75,33	125,82
	200	0,28	71,48	304,95
mr-abdo (256x256x16)	100	0,20	72,34	250,35
	200	0,37	68,55	598,68
cr- tórax (440x440x16)	20	0,37	85,36	12,48
	50	0,47	83,68	18,38

Tabla 1.5 - Capacidad (bpp) vs. Calidad - Algoritmo Original de Tian – Imágenes Médicas

Se observa que la calidad y capacidad dependen del umbral por encima del cual no se marca. A mayor valor del umbral, mayor capacidad en desmedro de la calidad. El SSIM evaluado en todos los casos supera el valor de 0,9990.

1.8.2- Método de Corrimiento de Histograma (Ni *et al.*)

Este método usa inserción aditiva.

Embebido

Se determina un pico P_p y un cero V_p en el histograma. Si $P_p<V_p$ se suma una unidad a las intensidades de los grises pertenecen al intervalo (P_p V_p), si fuera $P_p>V_p$ se restaría un uno a todas las intensidades pertenecientes al intervalo (V_p P_p). Sin perder generalidad se supone que se cumple el primer caso: entonces queda vacío el nivel de gris inmediatamente superior al pico. Se supondrá también que se inserta la marca recorriendo la imagen secuencialmente, pero puede hacerse en cualquier otro orden, acordado por las partes.

i=0. Se recorre la imagen y cuando se encuentra la intensidad del pico se verifica el bit b_i de la marca y se procede de la siguiente manera:

a) Si es 0 no se hace nada.

b) Si es 1, se suma una unidad a la intensidad de gris pico (por lo que va a ocupar el nivel que se había vaciado P_p+1).

Se incrementa en 1 el valor de *i* que indica la posición en la marca.

Se continúa hasta cubrir todos los bits de la marca.

Extracción

Se recorre la imagen con el objetivo de recuperar la marca. Cuando se encuentra un valor igual al valor de intensidad P_p se coloca un 0 en el vector de la marca recuperada. Cuando se encuentra el valor P_p+1, se coloca un 1 en el vector de la marca.

Se resta un 1 a todas las intensidades que pertenecen al intervalo (P_p V_p).

Discusión del método

Este método necesita poca información extra para lograr reversibilidad: las intensidades P_p y V_p, también la longitud de la marca. Puede ser necesaria más información, por ejemplo en el caso de usar un valle distinto de cero.

Su capacidad máxima está dada por la frecuencia de la intensidad P_p. Si se quiere aumentar la capacidad se deben buscar más pares pico-valle en el histograma de la imagen, en este caso deben almacenarse todas las intensidades picos-valles y longitud de la marca.

La calidad es alta, ya que solamente se modifican en una unidad los valores de intensidad. Esto se ve claramente en la expresión de MSE que tiene un valor máximo posible de 1 (1.19). La complejidad computacional es baja.

Experimentación

En la Figura 1.8 se muestra el histograma de la imagen médica original de la Figura 1.4. En este caso el pico corresponde a la intensidad 1095 y la frecuencia es 5027. Por lo tanto la capacidad máxima es de 5027 bits. Al insertar una marca generada aleatoriamente aprovechando toda la capacidad se obtuvo un PSNR de 70,17 dB.

Para imágenes comunes de 256 niveles de gris, el PSNR mínimo es de 48 dB.

$$MSE_{\max} = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[I_W(x,y) - I(x,y)\right]^2 = 1$$

$$PSNR(dB)_{\min} = 10\log_{10}\frac{\left[I_{pico}(x,y)\right]^2}{MSE} = 10\log_{10}\left(255\right)^2 = 48dB \qquad (1.19)$$

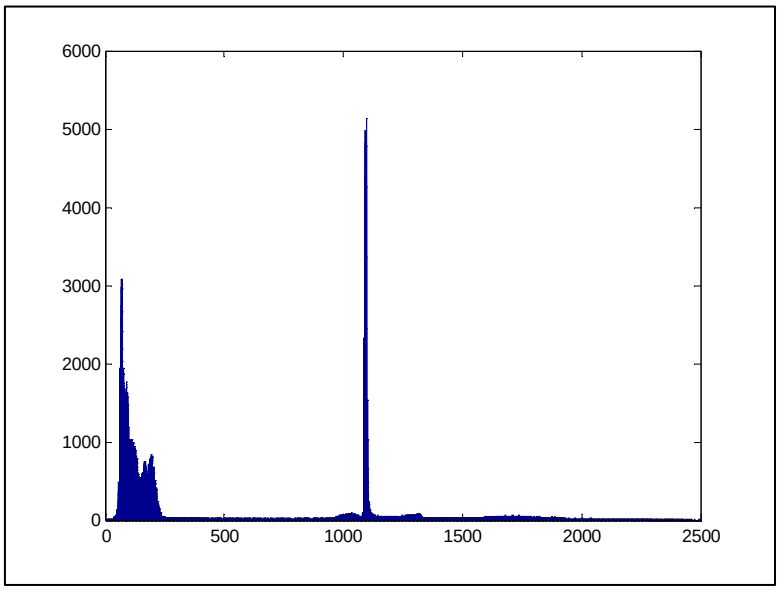

Figura 1.8- Histograma Imagen Médica Original (Figura 1.4)

En la tabla 1.6, se muestran valores de capacidad y calidad, obtenidos para las imágenes presentadas en las Figuras 1.5, 1.6 y 1.7. Se considera un solo recorrido de la imagen y la utilización de un único pico.

Imagen	Bits Embebibles	Capacidad (bpp)	PSNR(dB)	MSE
lena	2723	0,010	53,31	0,28
barbara	2217	0,008	53,78	0,25
baboon	2708	0,010	50,45	0,58
boats	7480	0,018	53,29	0,23
mr-cerebro	64512	0,25	96,89	0,87
mr-abdo	13544	0,21	96,8	0,89
cr-tórax	679	0,003	104,21	0,16

Tabla 1.6 - Capacidad vs. Calidad - Algoritmo de Ni

El autor indica, por ejemplo, para la imagen lena la posibilidad de alcanzar una capacidad de 5460 bits (0,0208 bpp) con un PSNR de 48,2 dB y para la imagen baboon 5421 bits (0,0206 bpp) con un PSNR también de 48,2 dB.

La calidad del método es excelente. Los valores de PSNR siempre superan los 48 dB. Sin embargo, la capacidad es variable, depende de la imagen, y solo se puede aumentar recurriendo a la utilización de otros picos-valles, lo que no siempre es posible.

1.8.3 Algoritmo de Xuan: Compresión de Plano de Bits

Entre la calidad excelente de Ni y la capacidad de Tian, se ubican otros algoritmos reversibles. Uno de ellos es el desarrollado por Xuan *et al.* [36] que opera en el dominio de la Transformada Wavelet, en particular de la cdf(2,2) que se analizará en el siguiente capítulo. Comprime aritméticamente un plano de bits de los coeficientes de las bandas de detalle para colocar en el espacio ganado los bits de la marca. La sobrecarga es la que precisa el método de

compresión para recuperar el plano de bits original luego de extraída la marca. El plano de bits propuesto para embeber es el segundo. Una compresión del plano de LSBs, será poco efectiva. Una compresión en planos más altos aporta más capacidad, pero a costa de mayor degradación de la imagen. Como el método no puede predecir el desbordamiento al realizar la antitransformada, propone un preprocesamiento que consiste en comprimir el histograma. Esta compresión trae aparejada la necesidad de embeber más carga extra para recuperar la imagen original y degrada la imagen.

La tabla 1.7 muestra los valores máximos de carga obtenidos cuando se utiliza el cuarto plano de bits en las imágenes ‘lena’, ‘baboon’ y ‘barbara’. El SSIM no supera los 0,97, lo que indica menor adaptación al SHV que el método de Tian.

Imagen	**Capacidad (bpp)**	**PSNR(dB)**	**MSE**
lena (512x512x8)	0,1	35,71	17,44
	0,6	33,57	28,54
	0,7	No alcanza esta capacidad	
barbara (512x512x8)	0,1	31,34	47,74
	0,4	30,52	57,64
	0,5	No alcanza esta capacidad	
baboon (512x512x8)	0,1	29,51	72,74
	0,2	29,35	75,49
	0,3	No alcanza esta capacidad	

Tabla 1.7 - Capacidad vs. Calidad - Algoritmo de Xuan - Compresión Plano 4

La Figura 1.9 compara calidad (utiliza los valores de PSNR) y capacidad (medida en bpp) de este método, comprimiendo el cuarto plano de coeficientes de las subbandas de detalle con los que arroja Tian, ambos aplicados a la imagen ‘lena’.

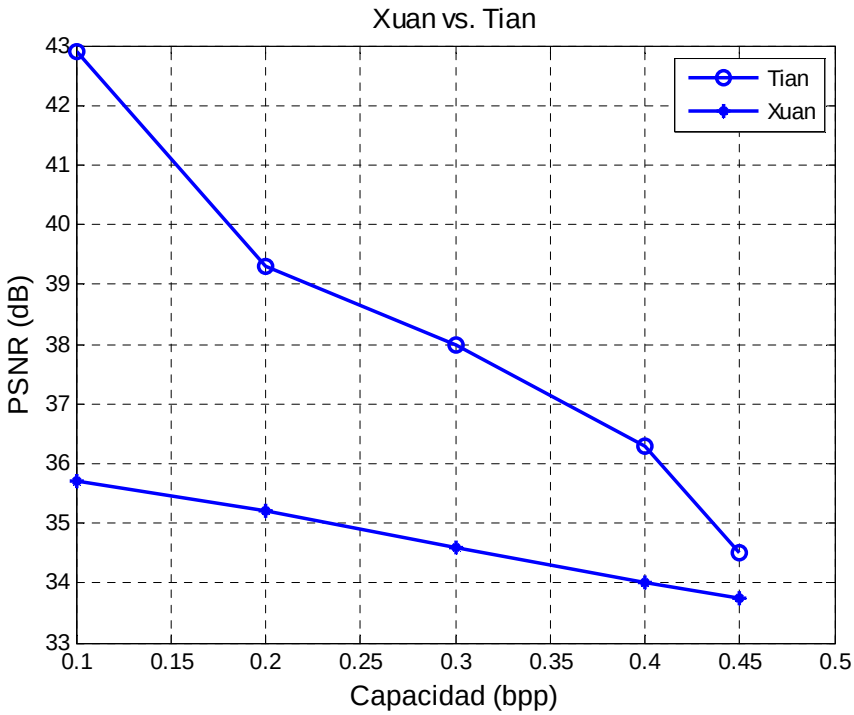

Figura 1.9 - Comparación: Xuan vs. Tian – Imagen 'lena'

1.9 Discusión de Métodos de Marcado Reversibles

Los métodos de marcado reversibles son los indicados para escenarios donde se requiere asegurar autenticación e integridad de los datos, pero también se persigue recuperar el contenido original del archivo para que este sea el manipulado por usuarios finales, tales como profesionales médicos. Un estudio detallado de técnicas reversibles realizado por Caldelli *et al.* se puede consultar en [37].

Se han presentado dos de las principales técnicas que reconoce la literatura: Ni y Tian. La calidad que produce el algoritmo de Ni en la imagen marcada es superior. Sin embargo el de Tian tiene, en general, mayor capacidad para una buena calidad de imagen. En las técnicas reversibles como se espera que la imagen marcada tenga utilidad, pero se recuperará el archivo original que será el manipulado por profesionales, se le otorgará mayor importancia a la capacidad que a la calidad de la imagen marcada.

También se ha explicado el algoritmo de Xuan de compresión de planos de bits, cuya capacidad es limitada, depende de la imagen y no se puede modificar.

El capítulo siguiente mostrará cómo se puede superar la capacidad del método de Tian, sin necesidad de hacer nuevos recorridos de la imagen.

CAPÍTULO 2

MÉTODOS REVERSIBLES CON COMPANSIÓN

El algoritmo de Tian es un método reversible de calidad apropiado para imágenes legales y médicas cuya capacidad máxima teórica es de 0,5 bpp si se hace un único recorrido de la imagen, valor que se puede aumentar con dos recorridos: uno vertical y otro horizontal, tal como se demostró en el capítulo anterior. Alattar [38] propuso posteriormente una extensión del método a *n* píxeles, tomando *n*-1 diferencias (en lugar de un par de píxeles y una única diferencia), lo que aumenta la capacidad del método que para un único recorrido de la imagen pasa a ser (*n-1*)/*n* bpp. Weng [39] restringió la propuesta de Alattar a cuaternas, tomando por lo tanto 3 diferencias, lo que arroja un valor de capacidad de 0,75 bpp, y además recogió una idea planteada en algunos trabajos de investigación previos: la compansión. Es posible aumentar la capacidad del método manteniendo la calidad visual, efectuando un proceso de compresión de las diferencias *h*, cuyo valor se encuentre por encima de un cierto umbral, de este modo se produce un mapeo de enteros a enteros. En la extracción hay que revertir el proceso realizando una expansión de los coeficientes. Este proceso de compresión y expansión es el que se conoce en procesamiento de señales como compansión.

La compansión registra antecedentes en particular en el procesamiento de señales de audio, donde se utiliza para realizar una cuantización no uniforme,

aplicar compansión y cuantización uniforme es equivalente a implementar cuantización no uniforme [40].

En watermarking este método fue propuesto inicialmente por Yang *et al.* [41], para ser aplicado en coeficientes DCT. Posteriormente fue utilizado por Xuan *et al.* [42] quienes desarrollaron un método de marcado que inserta la carga en los coeficientes wavelets de las subbandas de detalle de la transformada wavelet Cohen- Daubechies- Feauveau cdf(2,2). El estudio del SHV indica que ligeras modificaciones en los coeficientes de las subbandas de detalle no son percibidas por el ojo humano y, en particular esta transformada posee alta capacidad de embebido y también presenta calidad visual de la imagen marcada siendo la adoptada por la norma de compresión JPEG2000. Estos autores previeron la compresión de los coeficientes a partir de un umbral que se determina experimentalmente y que es función de la carga que se desea embeber. El método de Xuan *et al.* tiene una complicación ya que puede producirse overflow o underflow al realizar la transformada inversa para obtener la imagen marcada (en una imagen de 8 bpp esto significa obtener valores de intensidad por debajo del 0 o por encima de 255). Para solucionarlo Xuan *et al.* proponen realizar un preprocesamiento de la imagen consistente en una compresión del histograma antes de practicar la transformada y guardar la información necesaria para recuperar la imagen original [42]. Como la modificación del histograma desmerece la calidad de la imagen obtenida se debe efectuar solo en aquellos casos en los que sea estrictamente necesario.

En compansión, si se trata de calcular un único umbral globalmente, es posible encontrar el mejor valor por iteración en imágenes de 8 bpp o 16 bpp, por ejemplo, con mayor tiempo de búsqueda en las segundas, aunque es laborioso Si se pretende dividir la imagen en bloques y asignarles a cada uno un umbral conveniente, se vuelve muy costoso computacionalmente hacerlo por repetición.

La contribución original de este trabajo consiste en automatizar la búsqueda del mejor umbral mediante el empleo de algoritmos genéticos, considerando como función objetivo la minimización de MSE para una capacidad determinada menor que el límite impuesto por el método. Esta optimización se aplicará sobre una variante del algoritmo de Tian, en la que para mejorar la capacidad

del método se consideran cuaternas [44] y no pares de píxeles, y también sobre el algoritmo citado de Xuan, en este caso en forma similar a la planteada por Arsalan, Malik y Khan [45], para luego efectuar una comparación entre ambos. Los métodos se desarrollarán en este capítulo, en el siguiente se presentarán los algoritmos genéticos para finalizar mostrando los resultados de la experimentación.

2.1 Técnica de Compansión

En la mayoría de las imágenes, los coeficientes de alta frecuencia de las transformadas son pequeños y no produce desbordamiento sumarles un bit *b* que puede ser 0 o 1. Pero no siempre ocurre así por lo que conviene comprimirlos, según una función *C(x)* indicada en la ecuación (2.1). Se puede observar en la Figura 2.1 el gráfico de la función compresión, suponiendo un umbral U=20.

$$C(x) = \begin{cases} x & |x| < U \\ signo(x) \times \left(\dfrac{|x| - U}{2} + U \right) & |x| \geq U \end{cases} \qquad (2.1)$$

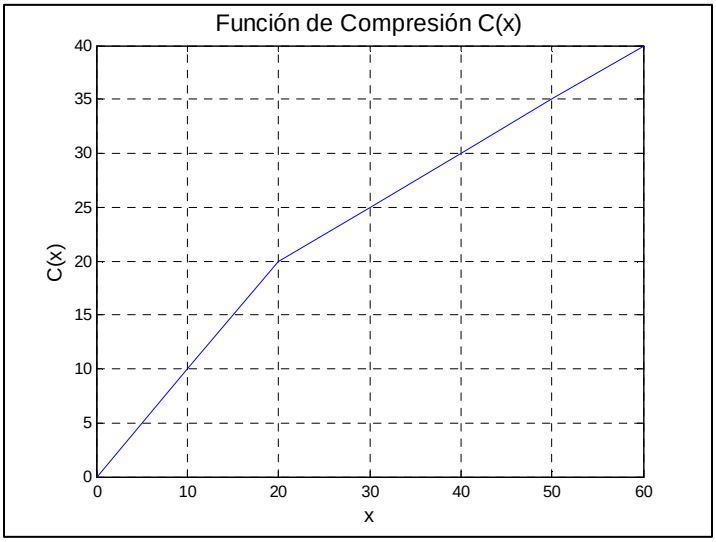

Figura 2.1- Función de Compresión- U=20

Por la naturaleza digital de la señal tratada se debe cuantificar la función de compresión, tomando la parte entera de la división, según indica (2.2).

$$C(x) = \begin{cases} x & |x| < U \\ signo(x) \times \left(\left\lfloor \frac{|x| - U}{2} \right\rfloor + U \right) & |x| \geq U \end{cases} \quad (2.2)$$

De esta forma, a los valores *x, x*+1 y *x* -1, les corresponde una misma imagen. Por lo tanto para recuperar el valor original se precisa embeber información extra, lo que se provoca una sobrecarga y hace bajar la perfomance del método. El error (0, 1 o -1) que se debe almacenar se indica en la (2.3) y se reduce a dos posibilidades (0 o 1).

$$Error = x - |E(C(x))| \quad (2.3)$$

Con este método se aumenta por un lado la capacidad porque hay más coeficientes embebibles, pero por otro lado se produce una sobrecarga por la necesidad de embeber los errores para lograr la reversibilidad. Sin embargo como los valores pueden comprimirse mediante algún método como la codificación aritmética y, en general, esta compresión es particularmente efectiva, el saldo es positivo y conviene aplicar la compansión.

2.2 Algoritmo de Tian Modificado

El algoritmo presentado por Tian en 2003 utiliza fundamentalmente la expansión de la diferencia entre los valores de dos píxeles vecinos. Para realizarla practica la sencilla operación de multiplicar por 2, lo que equivale a un desplazamiento de un bit a izquierda del valor original de la misma. Luego suma el valor de la marca. La restricción que se aplica es que no se produzca desbordamiento, es decir valores por encima de 255 o por debajo del 0, si se tratara de una imagen de 256 niveles de gris y, en general, teniendo en cuenta los valores extremos de intensidad. Mientras Tian trabajó exclusivamente con la diferencia entre dos píxeles vecinos sin solaparlos, en una variante posterior Alattar en lugar de tomar un par de vecinos consideró un vector formado por píxeles *n*-vecinos, dejando uno de ellos como pivote y expandiendo las diferencias de intensidad entre este píxel y los *n*-1 restantes. Luego, la variante desarrollada por Weng *et al.* utiliza submatrices 2x2 obtenidas a partir de la imagen y aplica además la técnica de compresión que luego revierte con una

expansión, según se trató en el punto anterior. Esta compresión de las diferencias aumenta la cantidad de las mismas en las que se puede incorporar marca sin producir desbordamiento. Para esto se usa un valor umbral, que definirá si la diferencia es o no comprimida. Este umbral se puede variar de forma tal de conseguir mejores resultados, manteniendo una relación de compromiso entre calidad (medida por PSNR, MSE o SSIM) y capacidad. Todas estas técnicas van agregando una sobrecarga, es decir nueva información que debe incorporarse a la imagen para poder recuperarla, lo que se concreta haciéndola formar parte de la marca con lo que se preserva el tamaño original del archivo. Es necesario que un algoritmo de marcado exhiba capacidad, por lo que se deben adoptar otras estrategias en este sentido, tales como la compresión de la información extra que se debe embeber para lograr la reversibilidad. En las subsecciones siguientes, se explicará en detalle el algoritmo y la composición de esta información extra que se debe almacenar.

2.2.1 La Transformada Entera

Se considera una matriz 2x2, que recibe el nombre de *quad*:

$$q = \begin{bmatrix} u_0 & u_1 \\ u_2 & u_3 \end{bmatrix}$$

donde $u_0, u_1, u_2, u_3 \in Z$.

La transformada entera directa indicada en (2.4) se define según (2.5):

$$\begin{bmatrix} u_0 & u_1 \\ u_2 & u_3 \end{bmatrix} \xrightarrow{T} \begin{bmatrix} v_0 & v_1 \\ v_2 & v_3 \end{bmatrix} \tag{2.4}$$

$$\begin{aligned} v_0 &= \left\lfloor \frac{u_0 + u_1 + u_2 + u_3}{4} \right\rfloor \\ v_1 &= u_0 - u_1 \\ v_2 &= u_0 - u_2 \\ v_3 &= u_0 - u_3 \end{aligned} \tag{2.5}$$

2.2.2 Inversibilidad de la Transformada Entera

Esta transformada es inversible, tal como se indica en (2.6), aplicando las ecuaciones (2.7):

$$\begin{bmatrix} v_0 & v_1 \\ v_2 & v_3 \end{bmatrix} \overset{T^{-1}}{\rightarrow} \begin{bmatrix} u_0 & u_1 \\ u_2 & u_3 \end{bmatrix} \tag{2.6}$$

$$\begin{aligned} u_0 &= v_0 + \left\lfloor \frac{v_1 + v_2 + v_3}{4} \right\rfloor \\ u_1 &= u_0 - v_1 \\ u_2 &= u_0 - v_2 \\ u_3 &= u_0 - v_3 \end{aligned} \tag{2.7}$$

La recuperación de los valores de los píxeles del quad original es fácilmente comprobable.

2.2.3 Obtención y Clasificación de los Quads

El algoritmo de embebido comprende los siguientes pasos:

a) Se divide la imagen a marcar *I* en bloques de píxeles adyacentes 2x2, los que no se solapan.

$$q = \begin{bmatrix} u_0 & u_1 \\ u_2 & u_3 \end{bmatrix} \; donde \; u_i \in Z \wedge Min \le u_i \le Max$$

y *Min* y *Max* son los valores extremos de intensidad del píxel, correspondientes al caso particular.

b) Se calcula la transformada entera, según (2.4). El valor v_0, promedio de los 4 píxeles vecinos, es un valor de gris que se mantendrá inalterable. Los otros valores son diferencias y podrán ser positivos o negativos.

De acuerdo a la idea de Tian de expandir la diferencia, estas se multiplican por dos lo que equivale a un desplazamiento a izquierda que deja un bit libre, que

se utilizará para colocar la marca. Esto es posible siempre que esta operación produzca un valor que al invertir la transformada no salga del rango permitido (entre 0 y 255, en el caso de tratarse de 256 niveles de gris). La expansión de la diferencia *h*, se indica en la ecuación (2.9), donde *h'* es la diferencia expandida y *b* el bit embebido.

$$h' = 2h + b \tag{2.9}$$

Luego:

$$\tilde{v}_i = 2v_i + b_i \quad i = 1,2,3 \tag{2.10}$$

Los valores $\tilde{u}_i$ obtenidos al antitransformar según las ecuaciones (2.7), a partir de los nuevos valores $\tilde{v}_i$ de la matriz transformada, deben cumplir las condiciones (2.11).

$$Min \le \tilde{u}_i \le Max \quad i = 0,1,2,3 \tag{2.11}$$

Dado que b_i puede tomar el valor 0 o 1, se debe hacer la verificación para ambos casos, es decir que

$$b_i \in \{0,1\} \tag{2.12}$$

Hay que ser cuidadoso al verificar los extremos, considerando siempre el menor y mayor valor posible.

Si el quad satisface las ecuaciones (2,11) para todas las diferencias se clasifica en un conjunto *C*1, y se dice que dichas diferencias son *expansibles*.

Se aplica compresión a las diferencias, de forma de obtener más valores pertenecientes a la clase C_1. Esta compresión se revertirá posteriormente con una expansión. En este caso la función utiliza un valor umbral y no va a ser reversible a menos que se inserte durante la expansión el error que se produce cuando el valor diferencia y el umbral son de distinta paridad, debido a que los valores de intensidad son enteros. En el algoritmo presentado, luego de comprimir se procesa ese valor obtenido, se embebe la marca y recién luego de realizar la extracción de la marca se procederá a la expansión y corrección mediante la suma del error almacenado para lograr reversibilidad.

Para un valor v, la función compresión se define en (2.13), donde *U* es el valor

del umbral.

$$C(v) = v_C = \begin{cases} v & |v| < U \\ signo(v) \times \left(\left\lfloor \frac{|v| - U}{2} \right\rfloor + U \right) & |v| \geq U \end{cases} \quad (2.13)$$

La elección del valor del umbral es importante. Por ahora se supone que es el mismo para todos los quads.

La función expansión se define en (2.14)

$$E(C(v)) = E(v_C) = \begin{cases} v_c & |v_c| < U \\ signo(v_C) \times \left(2 \times |v_C| - U \right) & |v_c| \geq U \end{cases} \quad (2.14)$$

Así, se observa que para valores mayores o iguales al del umbral se debe adicionar un error de compansión para recuperar el valor original, el que se calcula según (2.15), mientras que para valores menores al umbral el error es siempre nulo.

$$Error = e = v - |E(C(v))| \quad (2.15)$$

En definitiva, las ecuaciones (2.10) se aplicarán sobre las diferencias comprimidas, aumentando así la capacidad de embebido.

Para los quads que no pertenecen a la clase C_1, Tian previó una segunda clase, a la que llamó C_2, en esta las diferencias se conocen como *cambiables*. En este caso, al no poder expandir la diferencia porque se sale del rango de intensidad permitido, se divide por dos, perdiendo el LSB (bit menos significativo) y recién después se multiplica por dos y se embebe el bit de la marca. Esto se realiza según indica la ecuación (2.16), en aquellos quads que no pertenecen a la clase C_1 considerando de la misma manera que antes el valor de la diferencia comprimido. Una vez calculado el valor modificado por el embebido, se antitransforma y se verifica aplicando (2.11).

$$\tilde{v}_{i-2} = 2 \times \left\lfloor \frac{v_{iC}}{2} \right\rfloor + b \quad i = 1,2,3 \quad (2.16)$$

Estas verificaciones deben hacerse para $b \in \{0,1\}$. Si se cumplen las condiciones indicadas, el quad se puede marcar, pero para volver a obtener la imagen original es necesario almacenar el bit perdido y reponerlo.

Por último, todos los quads que no pertenecen a la clase C_1, ni a la clase C_2, no son embebibles y se asignan a una tercera clase designada C_3.

Para averiguar al extraer la marca a qué clase pertenece el quad, se hace necesario almacenar otra información más: un mapa de ubicación de los quads que llevará un 1 en aquellos que pertenecen a la clase C_1 y un 0 en los otros. Se debe considerar que dentro de la clase C_1 se distinguen dos subconjuntos, uno al que pertenecen los valores menores al umbral y otro para el resto. Un umbral muy bajo implicará muchos errores distintos de 0 que hay que guardar.

2.2.4 Construcción y Embebido de la Marca

Paso 1: Construcción de la marca

La información necesaria para lograr la reversibilidad forma parte de la marca y como tal será embebida, por lo tanto no formará parte de una cabecera de archivo, fácilmente desprendible, y no acarreará la necesidad de mayor capacidad de almacenamiento.

La marca estará formada por: a) El mapa de ubicación que lleva un 1 en las posiciones de los quads que pertenecen a C_1 y un 0 en el resto, al que se llamará *M*. b) Los bits originales menos significativos que deben almacenarse en el caso de las diferencias que pertenecen a los quads de clase C_2, conjunto al que se designará *B*. c) Los errores que produce la compresión, necesarios para que al expandir se logre recuperar la imagen original, conjunto al que se llamará E. d) La marca propiamente dicha o carga *P*. Esta puede ser un resumen o hash del archivo original, por ejemplo MD5. También puede tratarse de información para autenticar el archivo o datos encriptados relacionados con el archivo, es decir metadatos. A la marca total se la conocerá como *W*. Es decir que W= [M(mapa ubicación) B(LSBs perdidos) E(Errores por la compansión) P(marca propiamente dicha)] tal como indica la Figura 2.2. También es necesario en este caso que ambas partes conozcan el umbral, o bien debe incluírselo en el vector a embeber.

M (mapa de ubicación)	B (LSBs perdidos en cambiables)	E (Errores por compansión)	P (carga o marca propiamente dicha)

Figura 2.2 - Composición de la Marca en Tian con Compansión

En la Figura 2.3 se muestra la composición de la marca según Tian original sin el proceso de compresión- expansión.

M (mapa de ubicación)	B (LSBs perdidos en cambiables	P (carga o marca propiamente dicha)

Figura 2.3- Composición de la Marca en Tian sin Compansión

Los componentes de la marca tales como el mapa, los bits perdidos al marcar los cambiables y los errores se comprimen para lograr mayor efectividad del algoritmo, por ejemplo utilizando codificación aritmética o JBIG (Joint Bilevel Image Experts Group) [45]. Deben formar parte de la marca los valores necesarios para decodificar cuando se quiere recuperar la imagen. Por esta razón, es necesario agregar a la marca propiamente dicha una cabecera con información de la longitud de los distintos componentes de la marca, incluida la posición donde comienzan y terminan los datos propiamente dichos a recuperar.

Paso 2: Embebido de la marca

Para los quads que pertenecen a la clase C_1 se embeben los bits en los valores de la diferencia según indica la ecuación (2.10) y para los que pertenecen a la clase C_2 según la ecuación (2.16), sin olvidar guardar previamente los LSBs originales. Los quads que pertenecen a la clase C_3 quedan sin marcar. El embebido se hace según una clave, para aumentar la seguridad.

Paso 3: Transformada inversa

Insertada la marca se realiza la transformada entera inversa obteniendo la imagen marcada I_w.

2.2.5 Extracción de la Marca

Se realiza la transformada entera directa de la imagen marcada.

Para extraer la marca se deben clasificar los quads en dos grandes grupos según pertenezcan a C_3 y no hayan sido marcados, o bien formen parte de su complemento, en este último caso pueden haber pertenecido a la clase C_1 o C_2, lo que terminará de definirse cuando se extraiga e interprete el mapa de ubicación, de aquellos que pertenecen al complemento de $C3$ los que llevan un

1 en el mapa de ubicación pertenecen a la clase *C*1, el resto a *C*2.

Se prueba fácilmente que un valor que sido expandido es siempre cambiable y que los cambiables siguen siéndolo, por lo que se estudia en la imagen marcada I_w cuáles son los quads cambiables y cuáles no lo son. Estos últimos son asignados a la clase C_3, el resto al complemento de la misma. En dicho complemento se recupera la marca lo que se efectúa tomando el bit menos significativo. Es claro que no tiene por qué haberse seguido un orden secuencial en la inserción, pero se puede suponer así en una primera instancia. Una vez recuperada la marca hay que estudiar la cabecera a la que se asigna longitud fija considerando el peor caso para definirla. Otra posibilidad mejor es usar marcadores de EOS (end of stream) para separarla del resto y también para dividir los campos de la misma. De acuerdo a los valores de longitudes leídos en la cabecera se separa de la marca el mapa de ubicación de los quads que pertenecen a C_1, los bits perdidos en los quads que pertenecen a C_2, los errores que se deben considerar para poder revertir la compresión y, finalmente, la marca propiamente dicha, por ejemplo el resumen MD5 o metadatos encriptados. Como M, B y E están comprimidos, es necesario realizar la descompresión.

Una vez separados los datos embebidos, se buscar recuperar la imagen original. Para hacerlo hay que considerar si se aplicó la ecuación (2.10) o la (2.16), tener en cuenta los bits perdidos en el caso de haber habido un cambio y no una expansión de la diferencia y los errores en el caso de pertenecer los quads al subconjunto de C_1 de diferencias mayores al umbral, todo esto antes de aplicar la transformada inversa.

Para recuperar el valor original, se aplica en primer lugar la ecuación (2.17), siendo $\tilde{v}_{iw}$ el valor de las diferencias que corresponden a la imagen marcada para i=1,2,3.

$$v_i^{'} = \begin{cases} \tilde{v}_{iw} & \tilde{v}_{iw} \in C_3 \vee \tilde{v}_{iw} = \tilde{v}_i \\ \left\lfloor \frac{\tilde{v}_{iw}}{2} \right\rfloor & \tilde{v}_{iw} \in C_1 \wedge \ \tilde{v}_{iw} \neq \tilde{v}_i \\ 2 \times \left\lfloor \frac{\tilde{v}_{iw}}{2} \right\rfloor + b & \tilde{v}_{iw} \in C_2, \ b \in B \wedge \tilde{v}_{iw} \neq \tilde{v}_i \end{cases} \qquad (2.17)$$

Al decir que $\tilde{v}_{iw} = \tilde{v}_i$, se está afirmando que esta diferencia no se ha visto afectada por la marca, o sea que el quad correspondiente no fue marcado

Por último se debe aplicar la expansión según indica la ecuación (2.18), considerando especialmente aquellos casos en que hubo compresión y el valor original de la diferencia no fue inferior al umbral *U*.

$$v_i^{''} = \begin{cases} v_i^{'} & \left|v_i^{'}\right| < U \\ signo(v_i^{'}) \times \left(2 \times \left|v_i^{'}\right| - U + e\right) & \left|v_i^{'}\right| \geq U, e \in E \end{cases} \qquad (2.18)$$

Con este último paso, se logra la reversibilidad de la imagen. Luego se calcula la transformada entera inversa según la ecuación (2.7). Recuperada la imagen original, si la marca propiamente dicha es una función del tipo MD5 o SHA, se recalcula sobre el archivo recuperado y se compara con la marca recuperada. Ambas deben ser iguales, ya que el algoritmo es reversible. Si no son iguales significa que la imagen fue alterada. Si se hubiera calculado por bloques se puede llegar a saber además en qué zonas de la imagen se produjo alteración.

2.2.6 Elección del Umbral

La elección del umbral es de fundamental importancia para lograr la mayor capacidad posible manteniendo la imperceptibilidad de la marca, esta última se evaluará por medio de MSE o PSNR.

Aunque se puede lograr un buen umbral utilizando iteraciones, no es una buena solución y puede no llevar a ninguna solución. La propuesta es que para un capacidad dada se logre la mejor solución minimizando MSE, o bien maximizando PSNR. en forma automática mediante el empleo de algoritmos genéticos.

2.3 Algoritmo con Compresión/Expansión en el Dominio Wavelet

El algoritmo de Xuan que también usa compansión fue presentado por Xuan *et al.* en el año 2005. Este realiza el embebido en el domino de la transformada

wavelet Cohen-Daubechies-Feauveau 2,2 (cdf 2,2), previa compresión a partir de un umbral para poder aumentar la capacidad del método. Xuan había participado en el 2002 en el desarrollo de un método en el dominio de la transformada wavelet cdf(2,2) más sencillo [46]. Este comprimía un plano medio de bits de los coeficientes de las subbandas de detalle de la transformada con una técnica reversible tal como codificación aritmética e insertaba la marca en forma aditiva en el espacio conseguido. Además se debía insertar junto con la marca propiamente dicha información extra requerida por el método de codificación tal como la frecuencia de ceros y unos para lograr la decodificación y, por lo tanto, la reversibilidad. Preveía preprocesamiento de la imagen, consistente en compresión del histograma, para evitar overflow o underflow. La capacidad no superaba los 0,3 bpp con una calidad indicada por el PSNR en los mejores casos apenas superior a los 30 dB.

En el nuevo método también realiza el embebido solamente en las subbandas de detalle, horizontal, vertical y en diagonal de la transformada wavelet cdf(2,2), en forma aditiva, esta vez utilizando compansión. Para evitar que al antitransformar se produzca un desbordamiento por exceso o por defecto, una vez insertada la marca, propone realizar antes de calcular la transformada directa una compresión del histograma que sea reversible. En las subsecciones siguientes se detallan los pasos.

2.3.1 Compresión del Histograma

La compresión del histograma solo se efectuará si luego de realizar por primera vez el proceso de transformación wavelet, embebido y antitransformación se verifica que hay desbordamiento. En general, se debe tratar de evitar ya que introduce distorsión extra.

Sea, por ejemplo, una imagen de L niveles de gris, o sea que su rango está dado por el intervalo cerrado [0, L-1]. Se decide, por ejemplo, comprimir 2 niveles, en ambos extremos del rango. El nivel 0 pasa a ser 1 y el 1 pasa a ser 2. Entonces es necesario almacenar información en forma vectorial que nos diga al recorrer la imagen si el 2 encontrado corresponde a un nivel original

(por ejemplo almacenando un 0 en un vector) o no (almacenando un 1 en el mismo vector). Así se procede también en el otro extremo del rango, L-1 pasa a ser L-2 y L-2 pasa a ser L-3, con lo que se almacena en un segundo vector un 0 si la intensidad original era L-3 o un 1 si no lo era. Este procedimiento se puede generalizar, desplazando m y n niveles en ambos extremos del histograma, respectivamente. La desventaja, además de la distorsión introducida, es que se aumenta la sobrecarga pues se deben almacenar estos vectores, llamados de escaneo, para recuperar el histograma original, por lo que no conviene ejecutar esta operación a menos que sea estrictamente necesario.

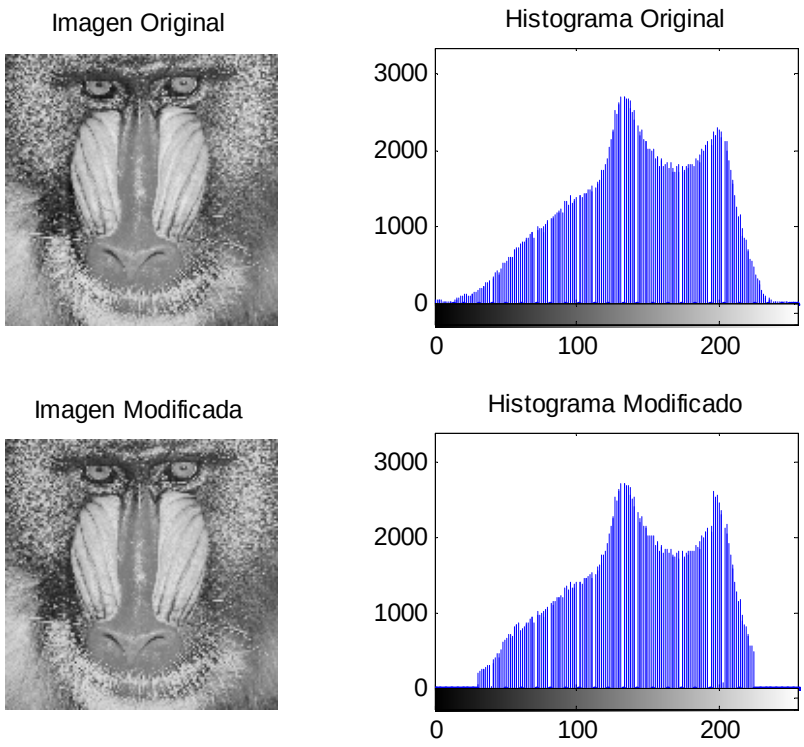

Figura 2.4 – Imagen Original y con Histograma Comprimido

En la Figura 2.4 se muestra la imagen ‘baboon.pgm’ cuyos valores extremos son las intensidades 0 y 255 con su histograma original y la modificada con el histograma comprimido al intervalo [30, 225], arrojando un valor PSNR entre la imagen original y la modificada de 37,30 dB, antes de efectuar la inserción de la marca. Esto indica una importante degradación de la imagen previa al marcado.

2.3.2 Transformada Wavelet cdf(2,2)

La transformada entera Wavelet cdf(2,2), que es también la que se emplea en el método de compresión de imágenes JPEG2000, fue la elegida en razón de que sus coeficientes son particularmente pequeños, poseen mayor capacidad de incrustación con una buena calidad visual, hecho confirmado mediante la experimentación [47]. La Figura 2.5 muestra a modo ejemplificador las subbandas obtenidas aplicando cdf(2,2) a la imagen 'baboon.pgm'.

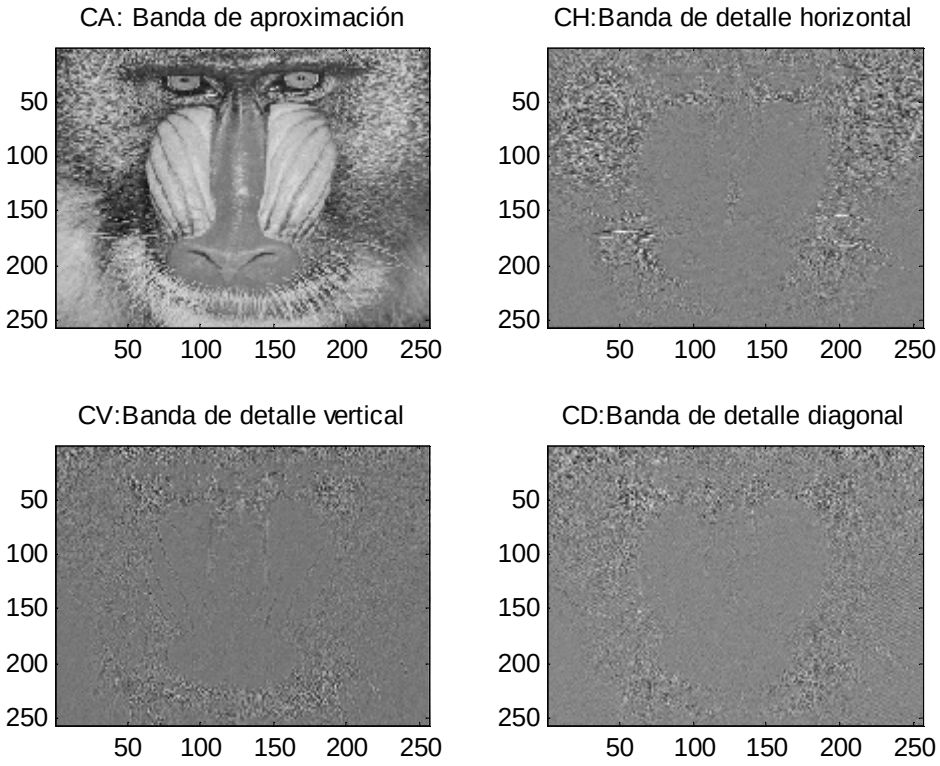

Figura 2.5- Subbandas Obtenidas mediante la Transformada Wavelet cdf (2,2)

La Transformada Entera Wavelet Biortogonal Cohen-Daubechies-Feauveau (2,2) utiliza dos filtros pasa bajos ***h'*** (para la transformada) y ***h*** (para la transformada inversa) y dos filtros pasa altos ***g'*** (para la transformada) y ***g*** (para la transformada inversa). Se ha usado el esquema lifting para el cálculo, en este caso se hacen predicciones y actualizaciones, considerando valores en posición par *s* y los que están en posición impar *d*, aplicando (2.19) y (2.20). Así se tiene:

Transformada Directa

$s_j \leftarrow x_{2j}; d_j \leftarrow x_{2j+1}$

$$\text{Predice} \quad d_j \leftarrow d_j - \left(\frac{s_j + s_{j+1}}{2} \right) \quad j = 1, 2, ..., N/2 \tag{2.19}$$

$$\text{Actualiza} \quad s_j \leftarrow s_j + \left(\frac{d_j + d_{j-1}}{4} \right) \quad j = 1, 2, ..., N/2$$

Transformada Inversa

$$s_j \leftarrow s_j - \left(\frac{d_j + d_{j-1}}{4} \right) \quad j = 1, 2, ..., N/2 \tag{2.20}$$

$$d_j \leftarrow d_j - \left(\frac{s_j + s_{j+1}}{2} \right) \quad j = 1, 2, ..., N/2$$

Para el caso en que se usan en el cálculo de la cdf(2,2) los tradicionales filtros pasaalto (indicados como g) y pasabajo (indicados como h) para el cálculo de esta transformada, los coeficientes son los que se indican en la tabla 2.1.

k	z^2	z^1	z^0	z^{-1}	z^{-1}	$z^{.3}$
h’	$\sqrt{2}\frac{-1}{8}$	$\sqrt{2}\frac{1}{4}$	$\sqrt{2}\frac{3}{4}$	$\sqrt{2}\frac{1}{4}$	$\sqrt{2}\frac{-1}{8}$	
g			$\sqrt{2}\frac{-1}{4}$	$\sqrt{2}\frac{1}{2}$	$\sqrt{2}\frac{-1}{4}$	
h		$\sqrt{2}\frac{1}{4}$	$\sqrt{2}\frac{1}{2}$	$\sqrt{2}\frac{1}{4}$		
g’		$\sqrt{2}\frac{-1}{8}$	$\sqrt{2}\frac{-1}{4}$	$\sqrt{2}\frac{3}{4}$	$\sqrt{2}\frac{-1}{4}$	$\sqrt{2}\frac{-1}{8}$

Tabla 2.1- Coeficientes cdf(2,2)

2.3.3 – Compresión de los Coeficientes

La marca de agua propiamente dicha compuesta por los datos útiles y la sobrecarga necesaria para lograr reversibilidad se inserta en los coeficientes de las bandas de detalle.

Para ampliar la capacidad, se recurre en este método a la compresión de los coeficientes a partir de un cierto umbral U, tal como se indicó en el algoritmo anterior. También es este caso se consideró originalmente un único umbral para toda la imagen. El proceso de compresión se indica en (2.21) y el de expansión que tendrá lugar en el proceso de extracción de la marca se indica en (2.22), designándose como v_w a los coeficientes de las subbandas de detalle obtenidas mediante la transformada wavelet.

Para generalizar no se considera un umbral constante, sino uno que depende de la posición del coeficiente considerado.

$$v_{cw} = C(v_w) = \begin{cases} v_w & |v_w| < U(i,j) \\ signo(v_w) \times \left(\left\lfloor \frac{|v_w| - U(i,j)}{2} \right\rfloor + U(i,j) \right) & |v_w| \geq U(i,j) \end{cases} \quad (2.21)$$

$$E(C(v_w)) = E(v_{cw}) = \begin{cases} v_{cw} & |v_{cw}| < U(i,j) \\ signo(v_{cw}) \times \left(2 \times |v_c| - U(i,j) \right) & |v_{cw}| \geq U(i,j) \end{cases} \quad (2.22)$$

Debido al redondeo, la recuperación no es exacta, cometiéndose un error en aquellos valores superiores al umbral. Por lo tanto estos errores, dados por (2.23), deben almacenarse para lograr reversibilidad.

$$Error = e = v_w - |E(C(v_w))| \quad (2.23)$$

Para ganar capacidad, la matriz de errores debe comprimirse mediante algún método reversible, como el de codificación aritmética. Si bien hay tres valores posibles de error: 0, 1 y -1, se pueden reducir solamente al 0 y el 1 y después considerar el signo mediante cálculo. También debe guardarse la información necesaria para realizar en la recuperación de la imagen original, la decodificación aritmética.

2.3.4 Elección del Umbral

Se debe considerar que el valor del umbral elegido debe ser tal que no se produzca desbordamiento al calcular la transformada inversa, que se obtenga una buena calidad y también la capacidad deseada, recordando que estas están interrelacionadas.

La elección del valor umbral es crítica. La propuesta de este trabajo es aplicar algoritmos genéticos en el cálculo, los que se detallarán en el capítulo siguiente, y dividir las subbandas en ventanas de modo tal de que cada una de ellas pueda tener un valor de umbral distinto, validando que no solo se obtenga una buena calidad sino que no se produzca desbordamiento.

Este caso se revela más complicado que el anterior donde se puede prevenir el desbordamiento sin consumir muchos cálculos. Con este método es necesario realizar los cálculos primero y si hay truncamiento se debe comprimir el histograma y recalcular. Esto se repite hasta tanto no haya desbordamiento, recordando que la compresión del histograma no es deseable ya que degrada la imagen.

2.3.5 Construcción y Embebido de la Marca

La marca a insertar se compone del vector que guarda los datos necesarios para recuperar la imagen y la marca propiamente dicha o útil.

Una posibilidad es que el vector contenga los siguientes campos:

a) Una cabecera

1) Longitud del vector de errores comprimidos. (24 bits)
2) Frecuencia de los ceros en la matriz de errores. (24 bits). La frecuencia de los unos se obtendrá por cálculo.
3) Valor del umbral, supuesto constante (8 bits).
4) Valor mínimo de intensidad en el histograma comprimido (8 bits) .
5) Valor máximo de intensidad en el histograma comprimido (8 bits).
6) Longitud del vector de scaneo histograma a izquierda (16 bits).
7) Longitud del vector de scaneo histograma a derecha (16 bits).
8) Longitud de la marca de agua propiamente dicha (32 bits).

b) Un cuerpo que contiene:

1) El vector de errores
2) La marca útil.

3) El vector de escaneo a izquierda.

4) El vector de escaneo a derecha.

Para el cuerpo. los bits usados son los que se indican en los campos de la cabecera. La longitud de los campos de la cabecera se toma como 8 bits o múltiplo de 8 bits.

La marca, construida como se indicó, es embebida en los coeficientes de detalle, ya sea en forma secuencial o siguiendo una secuencia pseudoaleatoria previamente convenida, para agregar seguridad, mediante la ecuación (2.24).

$$v_{cw}^{'}(i,j) = 2 * v_{cw}(i,j) + w(k) \quad k \in [0, L_M - 1] \tag{2.24}$$

siendo L_M la longitud de la marca

Se calcula la transformada inversa wavelet. Así se obtiene la imagen marcada. Se puede determinar la calidad de la misma, mediante PSNR, MSE o SSIM.

2.3.6 Extracción de la Marca

El primer paso es calcular la transformada wavelet cdf(2,2) de la imagen marcada.

Luego se extraen los bits menos significativos de los coeficientes de la subbandas de detalle según la ecuación (2.25) realizada en el dominio de la transformada sobre los coeficientes de las subbandas de detalle.

$$b = LSB(v_{CW}) = \mathrm{mod}(v_{CW}, 2) \tag{2.25}$$

En estos bits está embebida la marca y los datos necesarios para lograr reversibilidad. La ecuación (2.26) permite recuperar el valor original comprimido.

$$v_c = \frac{(v_{cw} - b)}{2} \tag{2.26}$$

Luego se realiza la expansión y se suman los errores.

Finalmente se antitransforma y se recupera el histograma original.

2.4 Discusión de la Capacidad de los Métodos

La capacidad máxima del método descripto en la sección anterior es de 0,75 bpp, ya que solo se embeben tres de las cuatro subbandas y no se utiliza más de un bit por píxel. Para aumentar la capacidad, la única posibilidad es continuar el embebido de la marca en un segundo bit, pero esto degrada excesivamente la calidad de la imagen en las comunes, aunque se puede emplear en las médicas que tienen mayor cantidad de bits por píxel.

De este modo la capacidad teórica máxima de este segundo método (Xuan) coincide con la capacidad teórica máxima del método anteriormente descripto utilizando quads (Tian modificado).

CAPÍTULO 3

ALGORITMOS GENÉTICOS y MEJORAS PROPUESTAS

La aplicación más común de los Algoritmos Genéticos (en adelante GA) ha sido resolver problemas de optimización donde han demostrado ser eficientes y confiables. En el caso en estudio los GA permitirán encontrar el umbral óptimo con lo que se podrá alcanzar mayor capacidad en el método con la mayor calidad posible en la imagen marcada, es decir mayor imperceptibiliad de la marca.

3.1 Introducción

John Holland ya en 1962 [48] [49] sentó las bases de posteriores desarrollos que hoy se conocen como algoritmos genéticos.

Un algoritmo genético es un método de búsqueda que imita la teoría de la evolución biológica de Darwin en la resolución de problemas. Se parte de una población inicial de la cual se seleccionan los individuos más capaces, luego estos se reproducen, y también mutan, para obtener la siguiente generación de la que se espera tenga individuos más aptos que la anterior.

Los objetivos que perseguían John Holland y sus colegas de la Universidad de Michigan cuando concibieron los algoritmos genéticos, fueron dos:

- abstraer y explicar rigurosamente el proceso adaptativo de los sistemas naturales.

- diseñar sistemas artificiales que reprodujeran los mecanismos más importantes de los sistemas naturales.

Un tema central en las investigaciones sobre algoritmos genéticos ha sido la robustez, la capacidad de sobrevivir en entornos diferentes. Las implicaciones que tiene la robustez en los sistemas artificiales son variadas. Si se puede conseguir que un sistema artificial sea más robusto, se podrán reducir, e incluso eliminar, los costos por rediseños. Y si se es capaz de lograr niveles altos de adaptación, los sistemas podrán desarrollar sus funciones mejor y durante más tiempo.

Cuando la optimización es el principal objetivo del uso del algoritmo genético, deben tenerse en cuenta las siguientes consideraciones:

- Si la función a optimizar tiene muchos máximos/mínimos locales se requerirán más iteraciones del algoritmo para "asegurar" el máximo/mínimo global.
- Si la función a optimizar contiene varios puntos muy cercanos en valor al óptimo, solamente se puede "asegurar" que se encontrará uno de ellos que, muchas veces, no es necesariamente el óptimo.

3.2 Algunas Definiciones

Gen es el nombre que recibe cada uno de los *parámetros* que forma parte de la solución a un problema.

Cromosoma es el nombre que recibe el *conjunto de todos los parámetros* (*genes*) codificados en una cadena de valores.

Genotipo comprende toda la información genética que tiene un individuo en particular, información que puede o no manifestarse en el individuo. El genotipo junto con factores ambientales determinan el fenotipo.

Fenotipo son las características que se manifiestan, que se observan y que corresponden a una determinada clase. Por ejemplo, el grupo sanguíneo particular que tiene una persona.

Alelo es el nombre que recibe cada uno de los bits que pertenecen a un gen (parámetro). Su ubicación se conoce como locus.

Para implementar un GA se usa codificación binaria. Por lo tanto a cada parámetro (gen) se le asigna un determinado número de bits. Las variables son discretizadas.

La Figura 3.1 muestra un ejemplo de cromosoma que codifica 3 parámetros con 12 alelos.

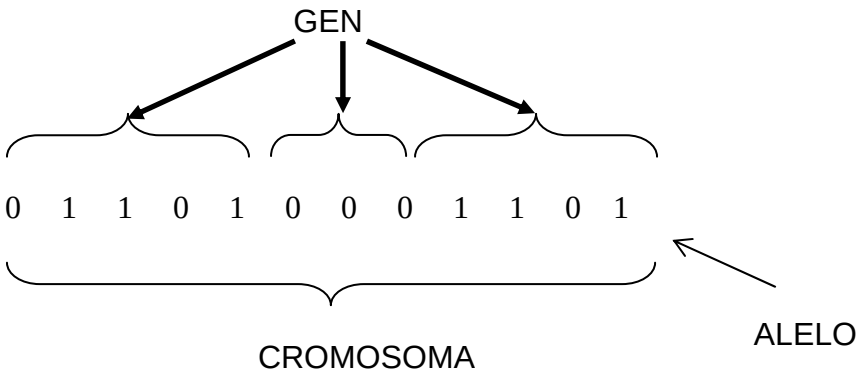

Figura 3.1 - Ejemplo de Cromosoma que Codifica 3 Parámetros – 12 Alelos

3.3 Esquema Básico

Dado un problema computacional, la aplicación de GA consiste en la realización de los siguientes pasos:

1. **Inicialización**: se crea una variedad de soluciones aleatorias válidas, cada una constituida por los genes. Dicha variedad recibe el nombre de población inicial.
2. **Selección**: se eligen las mejores soluciones de la población inicial en función de un criterio. En cambio, las "peores" soluciones serán reemplazadas. En esto consiste “la supervivencia del más apto” (survival of the fittest) de la que hablaba Darwin.
3. **Reproducción**: se genera la siguiente generación. Esto se hace con ayuda de dos operaciones inspirados en la evolución natural: cruza y mutación.

   En la cruza se mezclan partes de soluciones (los genes).

En la mutación se hacen modificaciones ligeras, tal como sucede en la naturaleza.

4. **Terminación**: Se repiten los pasos 2 a 3 hasta que se alcance una condición predeterminada.

3.4 Algoritmo Genético

En la Figura 3.2 se muestra el diagrama de flujo correspondiente a la implementación del algoritmo genético y en la Figura 3.3 el algoritmo en pseudocódigo donde P se refiere a la Población y P(Generación) a la que corresponde a una determinada Generación. P(0) se refiere a la población de la generación inicial.

En el momento de definir la población individual se fijan los criterios por los cuales se elegirán los individuos más aptos.

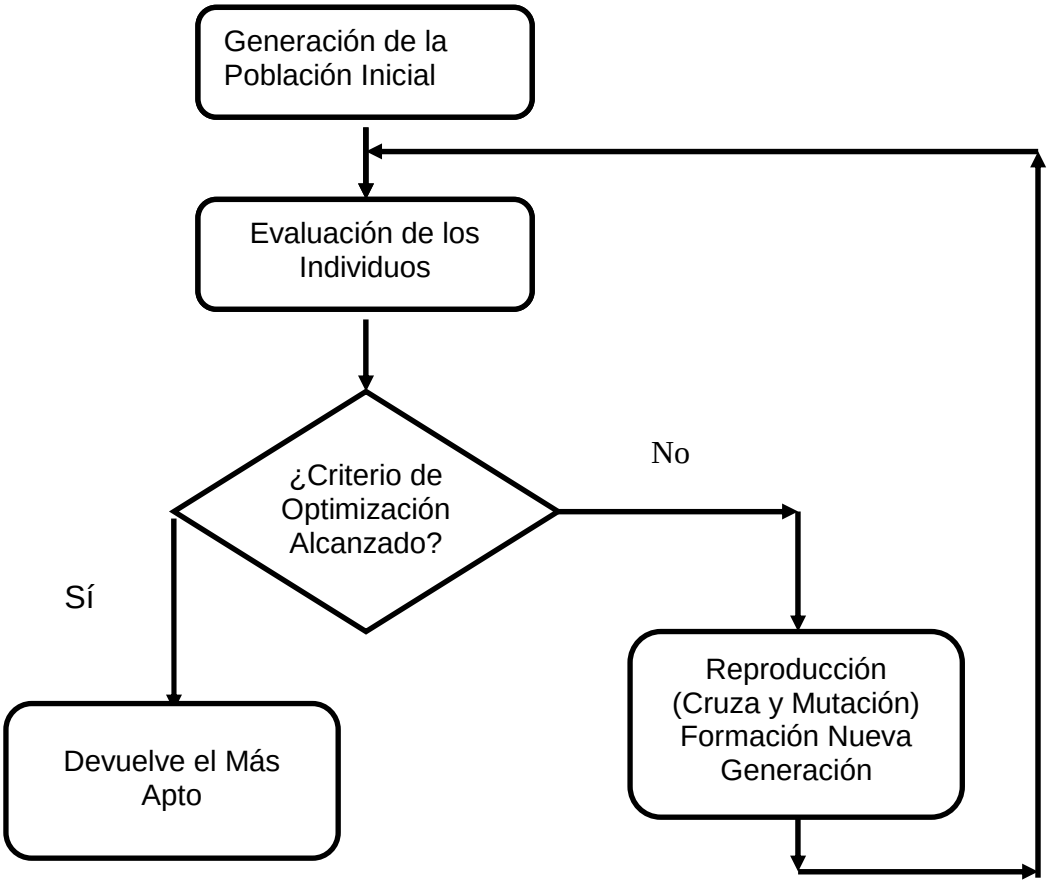

Figura 3.2- Diagrama de Flujo de Algoritmos Genéticos

```
Generación=0
Inicializar(P(Generación))
Evaluar(P(Generación))
mientras (no CriterioParada) hacer
   Padres = Seleccionar(P(Generación))
   Hijos = Aplicar_Cruza(Padres)
   Hijos = Aplicar_Mutación(Hijos)
   NuevaPob = Reemplazar(Hijos, P(generación))
   Generación ++
   P(Generación) = NuevaPob
   Evaluar(P(Generación))
fin
retornar Mejor Solución Encontrada
```

Figura 3.3: Esquema Algorítmico de un GA

3.5 Parámetros de los Algoritmos Genéticos

Para el estudio de los algoritmos genéticos [50] [51] hay que tener en cuenta una serie de parámetros, los que se detallan en las próximas subsecciones:

3.5.1 Tamaño de la Población

El tamaño de la población indica el número de cromosomas que se tiene en una población de una generación determinada.

En caso de que esta medida sea insuficiente, el algoritmo genético tiene pocas posibilidades de realizar buenas reproducciones, se realizaría una búsqueda de soluciones escasa y no óptima. Por otro lado, si la población es excesiva, el algoritmo genético será excesivamente lento sin aportar más beneficios. Hay un límite a partir del cual es ineficiente elevar el tamaño de la población puesto que no se consigue una mejor resolución del problema.

3.5.2 Probabilidad de Cruce

La probabilidad de cruce indica la frecuencia con la que se producen cruces entre los cromosomas padre, es decir qué probabilidad de reproducción existe

entre ellos. En caso de que esta sea nula, o sea que no existe probabilidad de reproducción, los hijos serán copias exactas de los padres. En caso de haber probabilidad de reproducción, los hijos tendrán partes de los cromosomas de los padres. Si la probabilidad de cruce es del 100%, todos los individuos de la nueva generación se generaron por cruza. Se recomienda que pasen algunos individuos copiados (los mejores).

3.5.3 Probabilidad de Mutación

La mutación altera un porcentaje de los bits de los cromosomas. Los bits a mutar suelen elegirse en forma aleatoria. Si se decide que no haya mutación, los descendientes son los mismos que había tras la reproducción. Puede mutarse el 100% de los bits, en ese caso la totalidad del cromosoma cambia, lo que significa que se produce en realidad su inversión y la población degenera muy rápidamente. La mutación permite extender el espacio de búsqueda, pero si es excesiva impide que el algoritmo converja a la mejor solución. La búsqueda deja de ser inteligente para ser aleatoria.

3.6 Función Evaluación

Para el correcto funcionamiento de GA se debe poseer un método que indique si los individuos de la población representan o no, buenas soluciones al problema planteado. Por lo tanto, para cada tipo de problema que se desee resolver hay que diseñar un método, otro tanto ocurre con la propia codificación de los individuos, es una para cada problema.

La función de evaluación que se diseña para cada caso en particular establece una medida numérica de la bondad de una solución. Esta medida recibe el nombre de ajuste, en la literatura comúnmente llamada *fitness*.

En la naturaleza, se puede considerar el ajuste de un individuo como la probabilidad de que ese individuo sobreviva hasta la edad de reproducción y se reproduzca.

En la ejecución de los GAs, se empleará el valor del ajuste para controlar la ejecución de los operadores genéticos, la cantidad de generaciones, es decir el número de selecciones, cruzas, copias y mutaciones que se lleven a cabo.

En el entorno en que se aplicará el GA, el ajuste puede ser el PSNR, MSE o SSIM. Puede intervenir también la capacidad del método o ser una combinación de todos o algunos de estos valores. Es común que en los programas se utilice encontrar un mínimo, este sería el caso si se utiliza MSE. Si se opta por el PSNR; por ejemplo, la mejor solución está dada por el máximo valor hallado, En ese caso se puede programar tomando el recíproco como ajuste o el complemento respecto a una constante elegida según el caso. A aquellos individuos que no cumplan con las condiciones se les asignará un valor de ajuste arbitrariamente alto.

3.7 Criterio de Selección

Los algoritmos de selección [52] serán los encargados de escoger qué individuos van a disponer de oportunidades de reproducirse y cuáles no. Pueden dividirse en dos grandes grupos: probabilísticos y determinísticos.

El primer tipo, los criterios probabilísticos, adjudica las posibilidades de selección mediante un componente importante basado en el azar.

El segundo grupo, los criterios determinísticos, engloba una serie de algoritmos que, dado el ajuste o bondad conocido de cada individuo, permite asignar a cada uno el número de veces que será escogido para reproducirse.

Los algoritmos probabilísticos para seleccionar son los que se utilizan en la práctica con mayor frecuencia.

La presión de selección de un método indica cuánto favorece este a los mejores individuos para que formen parte de la población padre. Si bien cuando la presión es alta, la convergencia es rápida, el riesgo es que se converja a un valor subóptimo, sin haber explorado todo el espacio de búsqueda. Si, en cambio, la presión de selección es nula, todos los individuos tienen igual posibilidad de sobrevivir, entonces la búsqueda recorre todo el espacio, pero se vuelve aleatoria.

3.7.1 Selección por Ruleta

Este método fue propuesto por De Jong en 1975 y es, posiblemente, el más utilizado desde los orígenes de los GA.

A cada uno de los individuos de la población se le asigna una parte proporcional a su ajuste en una ruleta, de tal forma que la suma de todos los porcentajes sea la unidad. Los mejores individuos recibirán una porción de la ruleta mayor que la recibida por los peores.

Generalmente, la población está ordenada en base al ajuste, por lo que las porciones más grandes se encuentran al inicio de la ruleta. Para seleccionar un individuo basta con generar un número aleatorio que pertenezca al intervalo [0,1] y devolver el individuo situado en esa posición de la ruleta. Esta posición se suele obtener también recorriendo los individuos de la población y acumulando sus proporciones de ruleta hasta que la suma exceda el valor obtenido, lo que se muestra en la Figura 3.4 donde p(I) es la porción de la ruleta asignada al individuo I.

```
Población
Suma=0
Repetir N veces (N= cantidad individuos a reemplazar)
   Generar un número aleatorio entre [0, 1]
   Recorrer los valores de los individuos
   Suma=Suma+p(I)
Hasta Suma>Nro Aleatorio Generado
Individuo_Seleccionado= I
```

Figura 3.4- Algoritmo de Selección por Ruleta

Es un método muy sencillo pero ineficiente a medida que aumenta el tamaño de la población. Una desventaja es que que necesita pesar los individuos. Presenta además el inconveniente de que el peor individuo puede ser seleccionado más de una vez.

3.7.2 Selección por Torneo

La idea principal de este método de selección consiste en escoger a los individuos genéticos en base a comparaciones (torneos) directas entre sus genotipos (conjunto de parámetros del cromosoma).

Existen dos versiones de selecciones mediante torneo, el torneo determinístico y el torneo probabilístico.

En la versión determinística se selecciona al azar un número P de individuos (generalmente se escoge P=2) y entre estos individuos se toma el más apto para pasarlo a la siguiente generación tal como se muestra en la Figura 3.5.

```
Población
Repetir
   Selección aleatoria de P Individuos de la Población
         (En general P=2)
   Elección del individuo de mayor aptitud
   Designarlo Ganador
Hasta Generar N Ganadores (Padres de la Siguiente Generación)
```

Figura 3.5- Algoritmo Determinístico de Selección por Torneo

En cambio, en la versión probabilística, se eligen aleatoriamente P individuos, pero en vez de escoger siempre el mejor individuo para pasarlo a la próxima generación, se genera un número aleatorio que pertenece al intervalo [0 1]. Si el valor generado aleatoriamente es mayor que un parámetro *p* (que se fija una sola vez para todo el proceso evolutivo) se escoge el más apto de los individuos seleccionados, en caso contrario, el menos apto. Generalmente *p* toma valores en el rango $0.5 < p \leq 1$. Se muestra el algoritmo en la Figura 3.6.

Variando el número de individuos que participan en cada torneo se puede modificar la presión de selección. Cuando participan muchos individuos en cada torneo, la presión de selección es elevada y así los peores individuos apenas tienen oportunidades de reproducción. Cuando el tamaño de los participantes del torneo es reducido, la presión de selección disminuye y así los

peores individuos tienen más oportunidades de ser seleccionados. Un caso particular es el *elitismo global*. Se trata de un torneo en el que participan todos los individuos de la población, con lo cual la selección se vuelve totalmente determinística.

```
Población
Determinar p
Repetir
   Selección aleatoria de P Individuos (En general P=2)
   Generar Número Aleatorio NA
   Si NA>p
         Se elige el individuo más apto de Población
   de otra manera
         Se elige el individuo menos apto de Población
   Fin Si
Hasta generar N ganadores (Padres de la Siguiente Generación)
```

Figura 3.6- Algoritmo Probabilístico de Selección por Torneo

La selección por torneo es una técnica eficiente y fácil de implementar. No requiera ordenar los Individuos según la función de aptitud, ya que las comparaciones son directas. Puede manejar la presión de selección según la cantidad de individuos,

3.7.3 Elección del Método de Selección

Elegir uno u otro método de selección determinará la estrategia de búsqueda del GA. Conviene una amplia zona de búsqueda inicial y luego más restringida. Si se opta por un método con alta presión de selección se centra la búsqueda de las soluciones en un entorno próximo a las mejores soluciones del momento en cuanto a ajuste. Por el contrario, optando por una presión de selección menor se deja el camino abierto para la exploración de nuevas regiones del espacio de búsqueda. Los métodos de ruleta y torneo son los más empleados.

Existen muchos otros algoritmos de selección. Unos buscan mejorar la eficiencia computacional, otros el número de veces que los mejores o peores individuos pueden ser seleccionados.

3.8 Cruce

Una vez seleccionados los individuos, éstos son recombinados para producir la descendencia que se insertará en la siguiente generación. El cruce es una estrategia de reproducción. El operador cruce o reproducción (conocido también como *crossover*) es el más importante dentro de los GA, esto se debe a que las tasas de cruce con las que se suele trabajar para la transición entre generaciones rondan el 90%

Los diferentes métodos de cruce podrán operar de dos formas diferentes. Si se opta por una estrategia destructiva, los descendientes se insertarán en la población temporal aunque sus padres tengan mejor ajuste (si se trabaja con una única población esta comparación se realizará con los individuos a reemplazar). Por el contrario, si se utiliza una estrategia no destructiva, la descendencia pasará a la siguiente generación únicamente si supera la bondad del ajuste de los padres (o de los individuos a reemplazar).

La idea principal del cruce se basa en que, si se toman dos individuos correctamente adaptados al medio y se obtiene una descendencia que comparta genes de ambos, existe la posibilidad de que los genes heredados sean precisamente los causantes de la bondad de los padres. Al compartir características buenas de dos individuos, la descendencia, o al menos parte de ella debería tener una bondad mayor que cada uno de los padres por separado. Si el cruce no agrupa las mejores características en uno de los hijos, y la descendencia tiene un peor ajuste que los padres, no significa que se esté dando un paso atrás. Si se opta por una estrategia de cruce no destructiva se garantiza que pasen a la siguiente generación los mejores individuos. Si aún con un ajuste peor, se opta por insertar a la descendencia, descartando a los padres, puesto que los genes de los padres continuarán en la población, aunque dispersos y posiblemente levemente modificados por la mutación, en

posteriores cruces se podrán volver a obtener estos padres, recuperando así la bondad previamente perdida.

Existen numerosos algoritmos de cruce. Los más empleados se detallan en las próximas subsecciones.

3.8.1 Cruce de Un Punto

Es la más sencilla de las técnicas de cruce. Una vez seleccionados dos individuos se cortan sus cromosomas en un punto seleccionado aleatoriamente para generar dos segmentos diferenciados en cada uno de ellos: la cabeza y la cola. Se intercambian las colas entre los dos individuos para generar los nuevos descendientes. De esta manera ambos descendientes heredan información genética de los padres como se observa en la Figura 3.7.

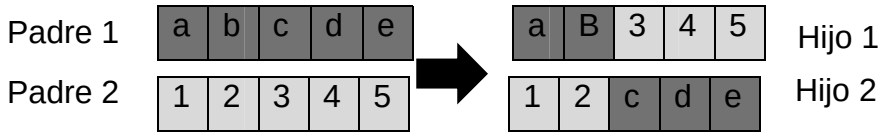

Figura 3.7- Cruce de Un Punto

3.8.2 Cruce de Dos Puntos

Se trata de una generalización del cruce de un punto.

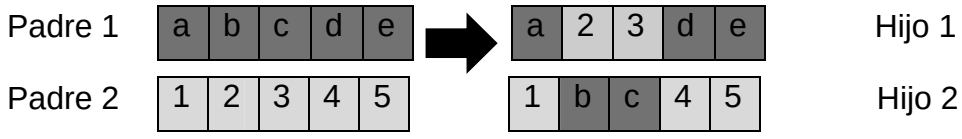

Figura 3.8- Cruce de Dos Puntos

En vez de cortar por un único punto los cromosomas de los padres, como en el caso anterior, se realizan dos cortes. Se deberá tener en cuenta que ninguno de estos puntos de corte coincida con el extremo de los cromosomas para

garantizar que se originen tres segmentos. Para generar la descendencia se escoge el segmento central de unos de los padres y los segmentos laterales del otro padre, según se observa en la Figura 3.8.

3.8.3 Cruces Multipunto

Se puede generalizar lo ya expuesto, añadiendo más puntos de cruce dando lugar a algoritmos de cruce multipunto. Existen estudios que desaprueban esta técnica (DeJong & Spears, 1999). Aunque se admite que el cruce de 2 puntos aporta una mejora sustancial con respecto al cruce de un solo punto, el hecho de añadir un mayor número de puntos de cruce reduce el rendimiento del Algoritmo Genético.

El problema principal de añadir nuevos puntos de cruce radica en que es más fácil que los segmentos originados sean corrompibles, es decir, que separando el original se pierdan las características de bondad que poseían conjuntamente. Hay que evitar, por lo tanto, romper dichos segmentos, denominados bloques constructivos. Sin embargo, añadiendo más puntos de cruce se consigue que el espacio de búsqueda del problema sea explorado con más intensidad, de modo que se puede obtener alguna ventaja de la aplicación del cruce multipunto.

3.8.4 Cruce Uniforme

El cruce uniforme es una técnica completamente diferente de las vistas hasta el momento. Cada gen de la descendencia tiene las mismas probabilidades de pertenecer a uno u otro padre.

Aunque se puede implementar de muy diversas formas, la técnica implica la generación de una máscara de cruce con valores binarios. Si en una de las posiciones de la máscara hay un 1, el gen situado en esa posición en uno de los descendientes se copia del primer padre. Si por el contrario hay un 0, el gen se copia del segundo padre.

Para producir el segundo descendiente se intercambian los papeles de los padres, o bien se intercambia la interpretación de los unos y los ceros de la máscara de cruce.

Como se puede apreciar en la 3.9 la descendencia contiene una mezcla de genes de cada uno de los padres. El número efectivo de puntos de cruce es fijo pero será en general del orden L/2, siendo L la longitud del cromosoma (es decir el número de alelos en las representaciones binarias)

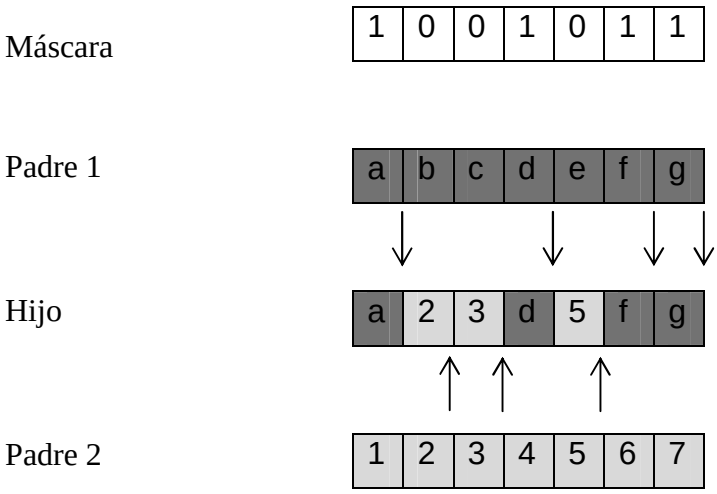

Figura 3.9- Cruce Uniforme

Una observación importante es que la máscara de cruce puede no permanecer fija durante todo el proceso evolutivo, generándose de forma aleatoria para cada cruce.

3.8.5 Copia

La copia es otra estrategia de reproducción. Consiste en utilizar el mismo individuo en la próxima generación. El porcentaje de copias debe ser relativamente reducido, ya que en caso contrario se corre el riesgo de no explorar en forma eficiente el espacio de búsqueda.

3.9 Mutación

La mutación es un operador binario que agrega aleatoridad al esquema

La mutación de un individuo consiste en que alguno de sus genes, generalmente uno solo, varíe su valor de forma aleatoria.

Aunque se pueden seleccionar los individuos directamente de la población actual y mutarlos antes de introducirlos a la nueva población, la mutación se suele utilizar de manera conjunta con el operador de cruce.

En una primera etapa se seleccionan dos individuos de la población para realizar el cruce. Si el cruce tiene éxito entonces uno de los descendientes, o ambos, se muta con cierta probabilidad P_m. Se imita de esta manera el comportamiento que se da en la naturaleza, pues, cuando se genera la descendencia siempre se produce algún tipo de error, por lo general sin mayor trascendencia, en el paso de la carga genética de padres a hijos, esto es lo que representa la mutación.

La probabilidad de mutación es muy baja, generalmente menor al 1%. Esto se debe sobre todo a que los individuos suelen tener un ajuste menor después de mutados. Sin embargo se realizan mutaciones para garantizar que ningún punto del espacio de búsqueda tenga una probabilidad nula de ser examinado.

La mutación más usual consiste en el reemplazo aleatorio, es decir variar aleatoriamente un gen de un cromosoma. Si se trabaja con codificaciones binarias, se tratará simplemente de negar un bit. También es posible realizar la mutación intercambiando los valores de dos alelos del cromosoma.

3.10 Criterio de Finalización de la Búsqueda

La búsqueda puede finalizar cuando se mejora un ajuste prefijado o bien por haber alcanzado determinada cantidad de generaciones fijadas previamente. En este segundo caso, se elige luego el mejor ajuste de todos los obtenidos. Puede ser necesario limitar la búsqueda en tiempo, o en cantidad de generaciones durante las cuales no se ha producido mejora en el ajuste.

3.11 Antecedentes de Aplicaciones de GA en Watermarking

Son numerosos los investigadores en watermarking que han aplicado GA en sus técnicas, la mayoría con el objetivo de obtener marcas robustas.

Uno de los primeros trabajos de aplicación de GA en watermarking es firmado por Shih y Wu [53] quienes desarrollaron en el año 2004 un algoritmo robusto para proteger imágenes en el dominio de la frecuencia basado en GA. Utilizaban el embebido en el dominio de la DCT, transformada que es particularmente útil para proteger la imagen marcada contra la compresión JPEG que también usa DCT. En particular, mediante el empleo de GA procuraron minimizar los errores de redondeo que se producen al convertir números reales en enteros durante la transformación DCT. El cromosoma inicial es la diferencia entre el valor de intensidad de la imagen original y el de la imagen marcada redondeada.

La DCT, transformada que permite obtener marcas robustas, tiene la particularidad de que si se embebe la marca en bajas frecuencias es resistente a ataques, pero en desmedro de la pérdida de imperceptibilidad. Si se inserta la marca en altas frecuencias, arroja una imagen marcada de mayor calidad, pero es débil frente a ataques, una compresión JPEG en este caso elimina la marca. Además el dominio DCT no brinda tanta seguridad como el de las transformadas wavelets. Las bandas medias son la solución de compromiso en DCT para embeber la marca. Huang *et al.* [54] presentaron una técnica progresiva de marcado que puede usarse en video, realizada mediante GA, en el dominio de la DCT en la búsqueda de mayor seguridad, robustez y calidad. Embeben la marca en la imagen dividida en bloques 8x8, en los que aplican DCT. El algoritmo modifica el orden de embebido mediante GA, probando la resistencia de la imagen marcada frente a ataques y tomando PSNR como función de ajuste.

Sikander *et al.* [55] propusieron un método similar basado en GA para aumentar la fortaleza de una marca en los valores medios de la DCT. Probaron su resistencia frente a ataques tales como filtro pasa-alto, de mediana y recortado de la imagen. El algoritmo va variando los coeficientes DCT en los

que inserta la marca mediante GA según la evaluación de la resistencia de la imagen marcada frente a ataques.

Goyal y Gupta [56] estudiaron la aplicación de mecanismos de selección por medio de torneo en una técnica GA de marcado que tiene como objetivo obtener un alto PSNR para cierta capacidad de marcado, para lo cual varía la posición donde se insertan los bits de la marca. En este caso solamente se busca imperceptibilidad y no se verifica robustez.

Finalmente, en el ámbito de las marcas reversibles, Khan *et al.* están trabajando fuertemente en la aplicación de GA tanto en el marcado de imágenes [57] como en el aseguramiento de la información de otros objetos tales como bases de datos relacionales [58]. En [57] desarrollan un algoritmo reversible aplicable a imágenes, la información es embebida en los LSBs de los coeficientes de una transformada wavelet. Buscan mejorar la calidad mediante la aplicación de GA que varía por mecanismo de selección y mutación la posición de los coeficientes donde se embebe la marca. Puesto que seleccionan los coeficientes, no aprovechan toda la capacidad posible. Arsalan *et al.* [45], en cambio, no eligen las posiciones de embebido, sino que utilizan compansión de los coeficientes de detalle de cdf(2,2) buscando aumentar la capacidad sin perder calidad, modificando el umbral para lo que usan GA.

La innovación que se presenta en este trabajo es la utilización de GA para el cálculo de umbral en forma automática en el algoritmo de Tian aplicado a cuaternas de modo de obtener imperceptibilidad de la marca para una capacidad dada. Se aplicará también el mismo método al algoritmo de Xuan que embebe la marca en los coeficientes de detalle de la transformada cdf(2,2), en forma similar a la desarrollada por Arsalan, se diferencia de este en que se usarán distintos umbrales para cada subbandas de detalle. Se demostrará comparando estos algoritmos que es preferible el modificado de Tian por sobre el de Xuan ya que el primero previene el desbordamiento y no precisa recálculo ni modificación del histograma, aunque reconociendo que en aquellos casos en que no es necesario esta última modificación la transformada wavelet cdf(2,2) produce una imagen de muy buena calidad. Se utilizarán cuaternas al aplicar Tian para lograr una capacidad aceptable en un solo recorrido de la imagen.

Esta automatización conseguida permite independizarse del usuario para determinar el umbral óptimo.

En este trabajo también se presenta la división de la imagen en bloques, aplicada por primera vez al algoritmo de Tian, cada uno con su umbral óptimo calculado automáticamente, lo que posibilita localizar la zona donde se produjo alguna modificación indeseada con similar complejidad computacional.

CAPÍTULO 4

PARTE EXPERIMENTAL

En una primera etapa se comparó Tian original (2003) con Tian modificado utilizando quads con umbrales fijos y únicos determinados por experimentación, lo que se reflejó en el capítulo 2.

En una segunda etapa se utilizaron algoritmos genéticos para determinar el umbral en forma automática para distintas capacidades, siempre con la restricción de que estas no pueden superar 0,75 bpp en ninguno de los métodos estudiados, ya que este es el límite teórico en ambos.

En la aplicación de algoritmos genéticos se eligió MSE como función de ajuste, aunque también se calculan y muestran el PSNR y el SSIM para evaluar la calidad de la imagen marcada. Experimentalmente se determina que valores de PSNR por encima de 30 dB, reflejan una calidad aceptable. Esto varía según el tipo de imagen. Se observa que imágenes como ‘baboon’ con menor PSNR mantienen un alto SSIM.

Se repitió el proceso anterior para el algoritmo de Xuan presentado en el capítulo 2.

Finalmente se dividió la imagen en bloques, asignando a cada uno de ellos un umbral, mediante GA. Esta división aumenta la carga debido a la necesidad de embeber también los umbrales de todos los bloques, pero puede ser útil para detectar la zona donde se produce una alteración. Se esperaba que el resultado mejore dada la correlación entre píxeles vecinos, lo que suavizaría en parte el efecto de la mayor carga. El aprovechamiento de esta correlación es posible puesto que las transformadas utilizadas tienen información de la posición espacial.

Como herramienta se usó el MatLab 13b.

Para realizar comparaciones con el algoritmo de Tian se partió de la función embebido suministrada por el investigador Asad Alí [59], adaptada para el caso de imágenes médicas y se programó la extracción a fin de tener todos los pasos del proceso. El resto de los programas son personales y se encuentran listados en el Anexo I.

4.1 – Tian en Cuaternas

4.1.1 Tian con pares versus Tian con cuaternas

Se marcaron las imágenes ‘lena.pgm’, ‘baboon.pgm’, una radiografía de tórax y una tomografía computada de cerebro, estas dos últimas en formato dicom y a las que se designará ‘cr-tórax’ y ‘ct-cerebro’, respectivamente. Las figuras 4.1, 4.2, 4.3 y 4.4 muestran las imágenes originales y las marcadas. En la reversión se obtiene que la imagen restaurada coincide con la original, es decir SSIM=1, MSE=0, PSNR=Inf.

Posteriormente, se varió el umbral. Luego de diversas pruebas, se determinó que un buen valor es en general, igual a la mediana o la media de los valores absolutos del bloque (la matriz de las diferencias en el caso de la variante de Tian, y las matrices de las subbandas de detalle en el caso de Xuan).

En otro paso se varió la carga en bits por píxel (bpp). Los resultados obtenidos se muestran en las tablas 4.1 y 4.2. La compansión persigue incrementar la cantidad de quads expandibles (clase C1) y favorece la compresión del mapa; por ejemplo, en el caso de un umbral de 20 en la imagen ‘lena’ el mapa comprimido ocupa solamente 35 bytes. Esta mejora siempre se ve empañada por la necesidad de guardar los errores producidos por la compansión para los valores de diferencia mayores que el umbral. No obstante, si se consigue efectivamente aumentar la cardinalidad de C1, la desventaja desaparece, ya que con la clase C2 también se debe guardar un bit, el LSB, y esto produce una disminución de la capacidad total. Con valores bajos de umbral se aprecia la ventaja de la compansión.

Luego se insertó como marca un resumen calculado con el algoritmo MD5,

para asegurar integridad. En este caso, la carga útil (bpp) es muy baja (128 bits) y los resultados que se muestran en la tabla 4.3 reflejan una muy buena calidad de la imagen marcada. Como la carga es la misma para una imagen dada, se facilita la comparación de los valores de PSNR obtenidos. En otros casos, para una misma carga útil se tiene distinta carga total, según la imagen o el umbral, porque varía la sobrecarga. En la tabla se puede observar que una imagen como 'baboon', muy texturada, necesita mucha carga extra.

Fig. 4.1 – Imagen Original y Marcada 'lena' – Tian Modificado

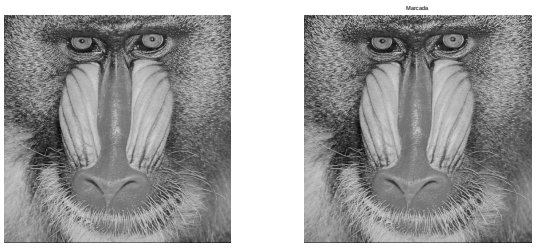

Fig. 4.2- Imagen Original y Marcada 'baboon' – Tian Modificado

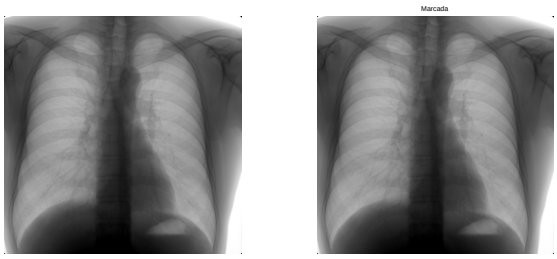

Fig. 4.3- Imagen Original y Marcada 'cr-tórax' – Tian Modificado

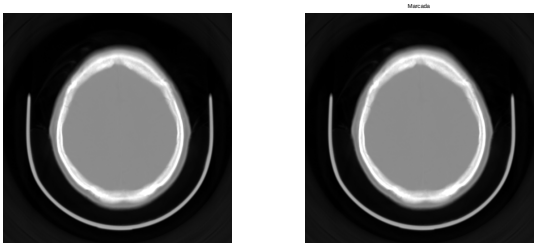

Fig. 4.4- Imagen Original y Marcada ‘ct-cerebro’ – Tian Modificado

Imagen	Umbral	Carga útil(bpp)	Carga total(bpp)	PSNR (dB)
lena (512x512x8)	20	0,346	0,404	37,52
	30	0,361	0,389	36,89
	40	0,368	0,382	36,61
	50	0,371	0,379	36,39
	100	0,372	0,378	36,08
	120	0,372	0,378	36,12
	20	0,693	0,750	34,80
	30	0,722	0,750	33,36
	40	0,735	0,750	32,71
	50	0,742	0,750	32,28
	100	0,745	0,750	31,79
baboon (512x512x8)	10	0,024	0,533	32,78
	10	0,121	0,630	32,99
	50	0,322	0,428	26,71
	120	0,349	0,401	25,73
	10	0,375	0,75	30,41

Tabla 4.1- Variación de Carga y Umbral - Tian sobre Quads

Imagen	Umbral	Carga útil (bpp)	Carga total (bpp)	PSNR (dB)
cr-tórax (440x440x16)	50	0,644	0,75	25,20
	20	0,353	0,396	48,04
	50	0,371	0,379	47,07
	100	0,371	0,379	47,07
	20	0,707	0,750	45,83
	50	0,741	0,750	44,51
	100	0,742	0,750	44,43

Tabla 4.2 Variación de Carga y Umbral - Tian sobre Quads – 'cr-tórax'

Imagen	Umbral	Capacidad requerida (bpp)	Long.Mapa Comprimido(B)	PSNR (dB)
lena (512x512x8)	20	0,058	35	42,10
	50	0,009	171	49,13
	100	0,006	907	57,93
baboon (512x512x8)	20	0,31	1757	30,81
	50	0,11	4407	30,87
	100	0,05	8393	32,56
cr-tórax (440x440x16)	20	0,046	595	53,73
	50	0,011	881	55,54
	100	0,010	1027	56,34
ct-cerebro (512x512x16)	100	0,004	171	44,88
	300	0,009	513	56,38
	500	0,006	544	76,00

Tabla 4.3 – Calidad de la Imagen Marcada con MD5 (Carga útil= 128 bits)

Finalmente se calculó PSNR versus la carga (en bits) aplicando el algoritmo original de Tian y el modificado en la imagen 'lena', lo que se presenta en las Figuras 4.5 y 4.6, respectivamente.

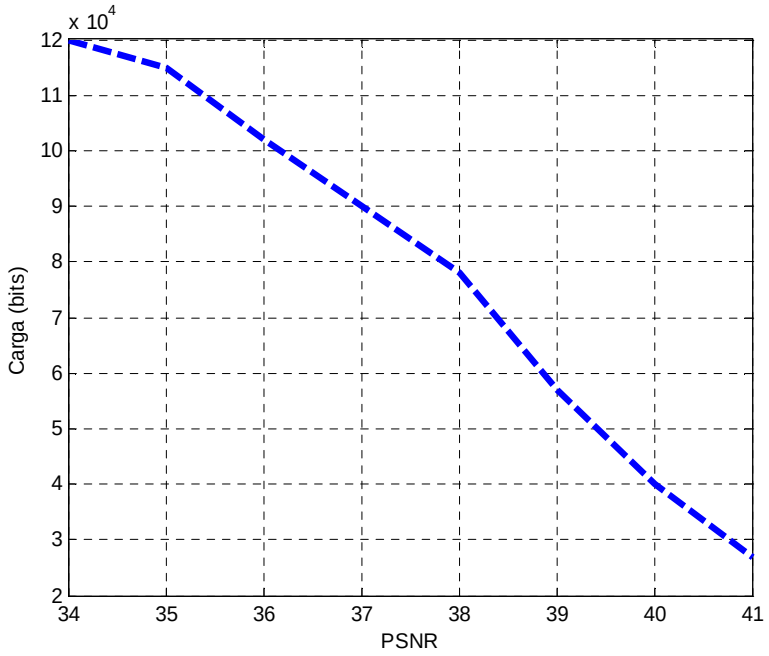

Figura 4.5- Carga (bits) vs. PSNR (dB)- Tian Original – 'lena'

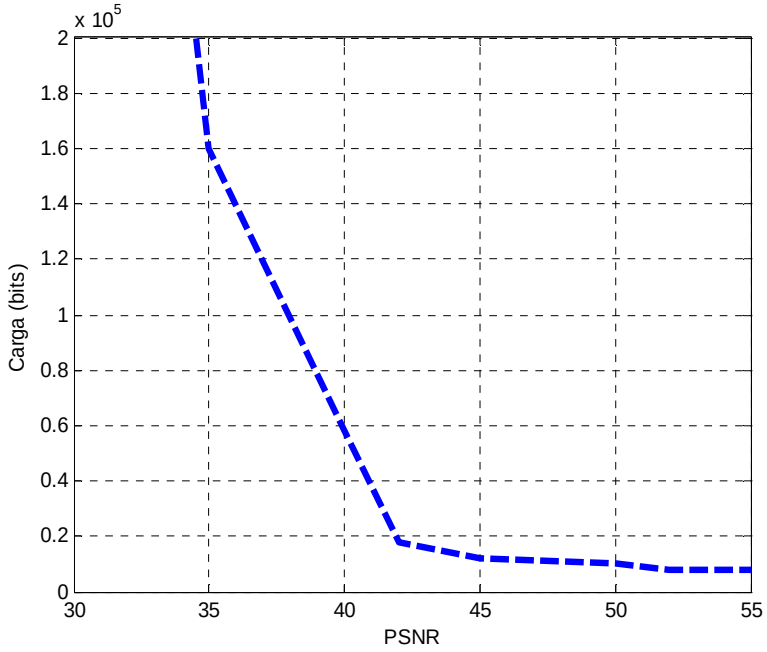

Figura 4.6 – Carga (bits) vs. PSNR (dB)- Tian sobre Quads – 'lena'

Si se compara la capacidad para valores similares de calidad, se observa que Tian ejecutado sobre pares de píxeles es superado en capacidad y en calidad

cuando se usan quads, especialmente para bajas cargas, cuando se utilizan cuaternas de píxeles. En la Figura 4.5 la escala de cargas es de $2x10^4$ y en la Figura 4.6 es de $2x10^5$.

4.1.2 Determinación del Mejor Umbral con GA

Se trabajó con distintos tamaños de bloques, determinando en cada caso un umbral. Esto se hizo con un mismo programa, trabajando con tamaño variable de bloque. Como función de ajuste del GA para determinar el mejor umbral se eligió la minimización de MSE, aunque también se calculan y muestran PSNR y SSIM.

Se hicieron numerosas pruebas para elegir la cantidad de población, las generaciones, el método de cruza y el de mutación. Se fijó un mínimo de MSE bajo, de modo que el resultado final es determinado después de cumplida una cantidad de generaciones o por tiempo, tomando de todos los valores obtenidos el mejor. Finalmente los valores elegidos fueron:

Población: 30. Tipo de población: vector, tipo de dato: double.

Generaciones: 50.

Método de selección: ruleta. Elitismo= 2 individuos

Método de cruza: aleatorio

Proporción de la población que es cruzada: 0,.8

Método de mutación: que se mantenga dentro del rango permitido.

La cantidad de variables depende de la cantidad de bloques del caso.

Los valores de la población deben pertenecer al intervalo dado por el valor absoluto mínimo y el máximo de los comprimibles.

En la Figura 4.7 se muestra como se divide la imagen en bloques

En las tablas 4.4, 4.5, 4.6 y 4.7 se muestran los resultados obtenidos para 1, 4 y 64 bloques variando la carga para imágenes comunes. En la tabla 4.8 se presentan los resultados obtenidos en imágenes médicas.

En la Figura 4.8 se muestran las imágenes ‘barbara.pgm’ y ‘boats.pgm’ y en la Figura 4.9 las imágenes médicas marcadas.

En la Figura 4.10 se comparan, para una misma imagen ‘lena’, los valores de PSNR obtenidos variando la cantidad de bloques.

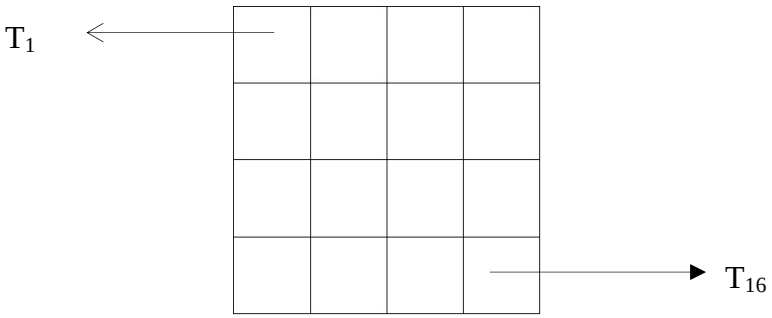

Figura 4.7- Imagen Dividida en 16 Bloques.

Figura 4.8 - Imágenes ‘barbara’ y ‘boats’

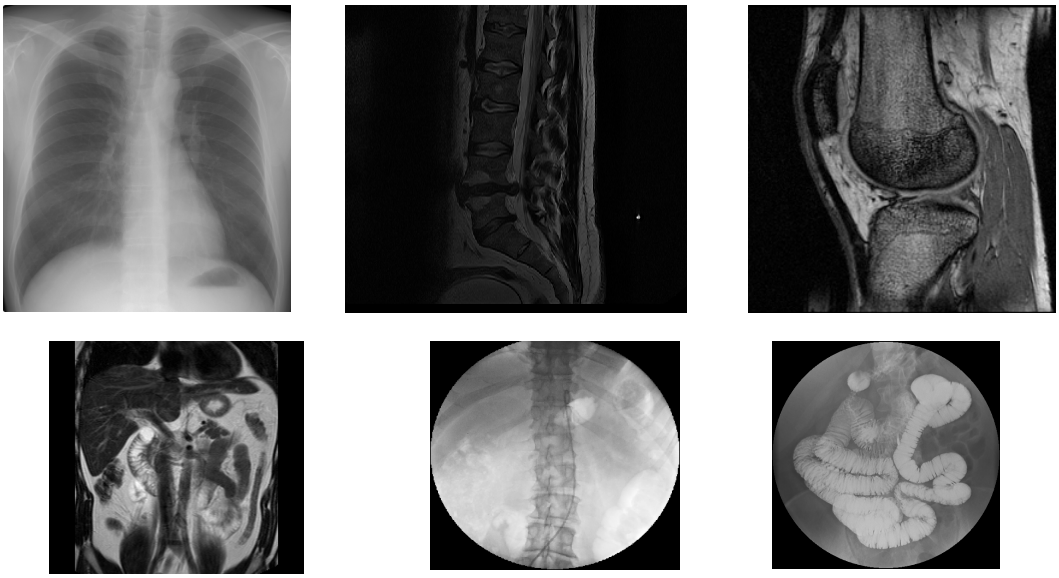

Figura 4.9- Imágenes ‘cr-tórax’, ‘mr-columna’, ‘mr-rodilla’,
’mr-abdo’, ‘cr-riñón’, ‘cr-intestino’

Carga Útil (bpp)	**Calidad**	**Imagen**			
		lena (512x512)	**baboon (512x512)**	**barbara (512x512)**	**boats (576x720)**
0,1	**PSNR(dB)**	45,40	32,61	39,53	47,66
	MSE	1,88	35,61	7,23	1,15
	SSIM	0,9997	0,9996	0,9992	0,9996
0,2	**PSNR(dB)**	41,22	30,01	36,35	41,58
	MSE	4,90	64,55	15,08	4,50
	SSIM	0,9989	0,9996	0,9999	0,9993
0,3	**PSNR(dB)**	39,00	27,48	32,06	39,01
	MSE	8,19	64,55	40,68	8,18
	SSIM	0,9989	0,9996	0,9993	0,9987
0,5	**PSNR(dB)**	35,82	24,68	28,34	33,80
	MSE	17,02	223,85	95,18	27,14
	SSIM	0,9989	0,9997	0,9993	0,9987
0,6	**PSNR(dB)**	33,69	24,11	27,08	31,91
	MSE	27,82	252.40	127,39	41,91
	SSIM	0,9988	0,9996	0,9992	0,9987
0,7	**PSNR(dB)**	32,18	No se logra esta capacidad	No se logra esta capacidad	31,32
	MSE	39,37			47.90
	SSIM	0,9988			0.9987

Tabla 4.4 - Tian sobre Quads- Capacidad (bpp) vs. PSNR (dB)- Cantidad de Bloques = 1

Carga Útil (bpp)	**Calidad**	**Imagen**			
		lena (512x512)	**baboon (512x512)**	**barbara (512x512)**	**boats (576x720)**
0,1	**PSNR(dB)**	47,42	31,84	38,61	47,43
	MSE	1,18	42,63	8,95	1,17
	SSIM	0,9998	0,9996	0,9993	0,9994
0,2	**PSNR(dB)**	42,64	28,03	36,17	41,30
	MSE	3,54	102,28	15,69	4,82
	SSIM	0,9996	0,9997	0,9998	0,9993
0,3	**PSNR(dB)**	37,64	26,63	31,64	37,80
	MSE	11,19	141,16	45,28	12,98
	SSIM	0,9994	0,9996	0,9996	0,9991
0,4	**PSNR(dB)**	35,44	25,10	30,23	36,60
	MSE	18,59	201,37	61,68	14,27
	SSIM	0,9992	0,9997	0,9993	0,9990
0,5	**PSNR(dB)**	34,08	24,47	27,57	34,41
	MSE	25,44	201,37	113,62	23,55
	SSIM	0,9991	0,9997	0,9994	0,9988
0,6	**PSNR(dB)**	33,55	24,09	26.97	32,58
	MSE	28,87	258,28	130,49	35,85
	SSIM	0,9988	0,9996	0,9993	0,9987
0,65	**PSNR(dB)**	32,85	No se logra esta capacidad	26,75	31,71
	MSE	33,77		137,68	43,83
	SSIM	0,9989		0,9992	0,9987
0,7	**PSNR(dB)**	32,43	No se logra esta capacidad	No se logra esta capacidad	31,29
	MSE	37,11			48,27
	SSIM	0,9988			0,9987

Tabla 4.5 - Tian sobre Quads- Capacidad (bpp) vs. PSNR (dB) -Cantidad de Bloques = 4

Carga Útil (bpp)	Calidad	Imagen			
		lena (512x512)	**baboon (512x512)**	**barbara (512x512)**	**boats (576x720)**
0,1	**PSNR(dB)**	44,97	29,13	35,56	41,04
	MSE	2,07	78,60	18,06	5,11
	SSIM	0,9997	0,9996	0,9994	0,9989
0,2	**PSNR(dB)**	40,62	26,40	34,78	38,54
	MSE	5,603	149,27	21,66	9,09
	SSIM	0,9996	0,9996	0,9994	0,9991
0,3	**PSNR(dB)**	37,62	25,58	31,48	36,07
	MSE	11,24	179,94	26,29	16,09
	SSIM	0,9994	0,9996	0,9994	0,9991
0,5	**PSNR(dB)**	33,88	24,67	27,94	32,99
	MSE	26,60	224,77	104,32	32,63
	SSIM	0,9992	0,9997	0,9994	0,9988
0,6	**PSNR(dB)**	32,61	24,09	27,03	31,89
	MSE	35,62	254,59	128,72	42,05
	SSIM	0,9992	0,9997	0,9993	0,9987

Tabla 4.6 - Tian sobre Quads- Capacidad (bpp) vs. PSNR (dB)- Cantidad de Bloques = 64

lena	Capacidad (bpp)						
	0,1	0,2	0,3	0,4	0,5	0,6	0,7
PSNR (dB)	45,29	40,70	37,68	35,46	33,93	32,81	32,43
MSE	1,92	5,55	11,08	18,71	26,34	34,05	37,11

Tabla 4.7 - Tian sobre Quads- Capacidad (bpp) vs. PSNR (dB)- Imagen ‘lena’ - Bloques =16

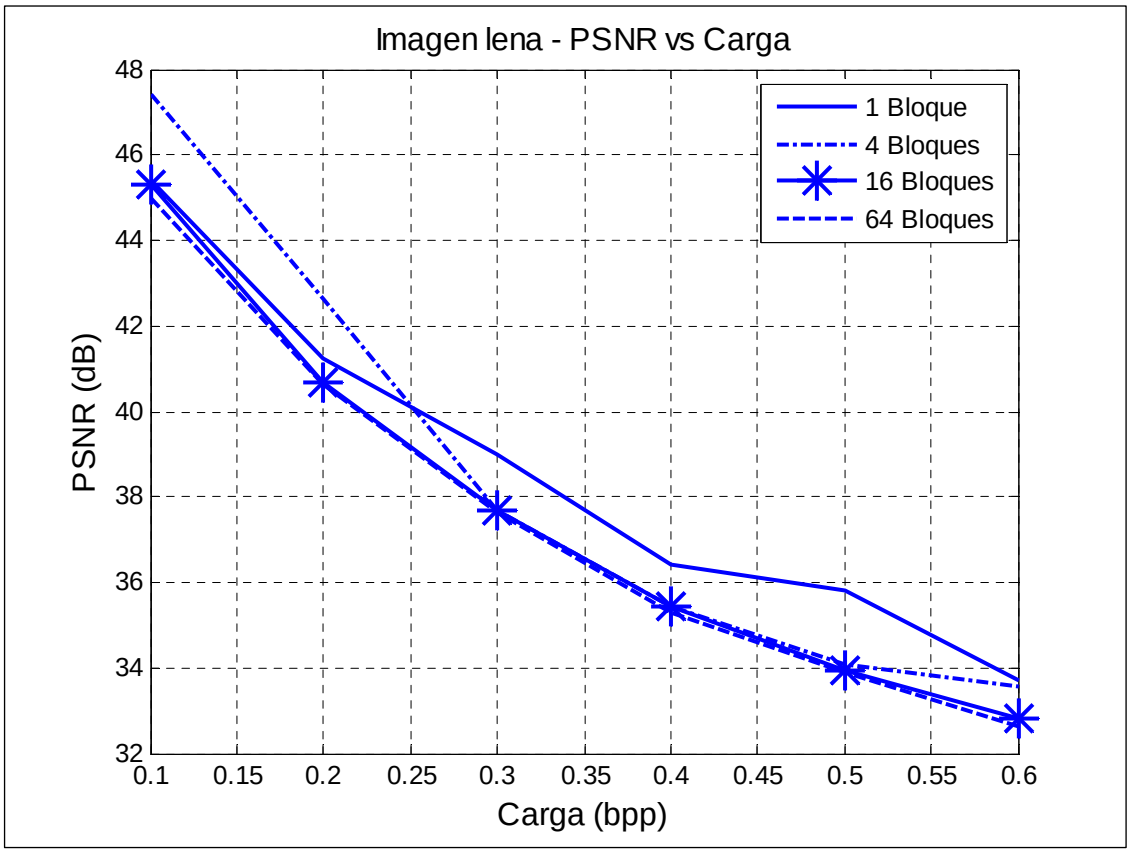

Figura 4.10 – Carga (bpp) vs. PSNR (dB) - 'lena'

Carga Útil (bpp)	**Imágenes Médicas PSNR (dB)**					
	cr-tórax 440x440x16	**mr-columna 512x512x16**	**mr-rodilla 256x256x16**	**mr-abdo 256x256x16**	**cr-riñón 512x512x8**	**cr-intestino 512x512x8**
0,1	88,48	75,78	86,81	70,10	45,62	44,61
0,2	86,04	75,25	82,94	68.09	44,24	42,31
0,3	84,74	75,16	80,67	66,75	43,72	39,,93
0,35	84,21	74,42	79,81	66,39	43,29	39,23
0,4	83,67	72,80	79,11	---	---	---
0,45	83,23	72,53	78,58	---	---	---
0,5	82,83	---	78,88	---	---	---
0,6	81,99	---	77,18	---	---	---
0,7	81,39	---	76,77	---	---	---

Tabla 4.8 - Tian sobre Quads con GA- Capacidad (bpp) vs. PSNR (dB)- Imágenes Médicas

En cuanto a las imágenes médicas, se consideraron en particular estudios de resonancia magnética y radiografías computadas. Por limitaciones computacionales, una vez lograda la capacidad requerida se abortó el procesamiento por tiempo.

En la tabla 4.8 se pueden apreciar regularidades en el comportamiento, por ejemplo la mayor capacidad en las imágenes que tienen áreas uniformes.

4.2 Algoritmo de Xuan con GA

En este algoritmo que se trató en la Sección 2.2 puede ser necesario comprimir el histograma para obtener reversibilidad, lo que produce una degradación extra en la imagen. Esto se conoce a posteriori de la aplicación de la compresión de los coeficientes, el embebido de la marca y la transformación inversa que produce la imagen marcada. Una vez verificado el desbordamiento, se comprime el histograma de la imagen original y se itera el procedimiento hasta que se logra reversibilidad. Es un proceso largo y costoso computacionalmente, además de aumentar la degradación. Se puede tomar una decisión previa inspeccionando el histograma.

En la experimentación se tomó un bloque por subbanda de detalle (no se marca la subbanda de aproximación), aunque también es posible dividir las subbandas en bloques. Se asigna un umbral diferente a cada bloque.

Con GA se determinaron los umbrales en forma automática. En el caso de la imagen ‘lena.pgm’, y en otras, no hace falta comprimir el histograma para lograr reversibilidad y la imagen marcada es de muy buena calidad. En algunas imágenes, como ‘baboon’ o ‘barbara’ para cargas de 0,1 bpp ya es necesario comprimir el histograma para lograr reversibilidad. Lo mismo sucede con ciertas imágenes médicas. En general, todas son de muy buena calidad hasta 0,1 bpp, pero luego es necesario comprimir el histograma. Para la inserción de un resumen como MD5, el método es preferible al anterior en cuanto a la calidad. Una automatización del método requiere considerar la compresión del histograma. De la inspección de valores obtenidos luego del embebido surge la posibilidad de comprimir el histograma de los coeficientes de la subbandas de

detalle en lugar de modificar el histograma de la imagen original, ya que a menudo son muy pocos los que producen desbordamiento.

En la Figura 4.11 se muestra una comparación de la ejecución de los algoritmos de Tian con GA y Xuan con GA en la imagen 'lena'. En la tabla 4.9 se presentan los valores obtenidos de PSNR y MSE para 'lena' y 'baboon' y en la tabla 4.10 para imágenes médicas.

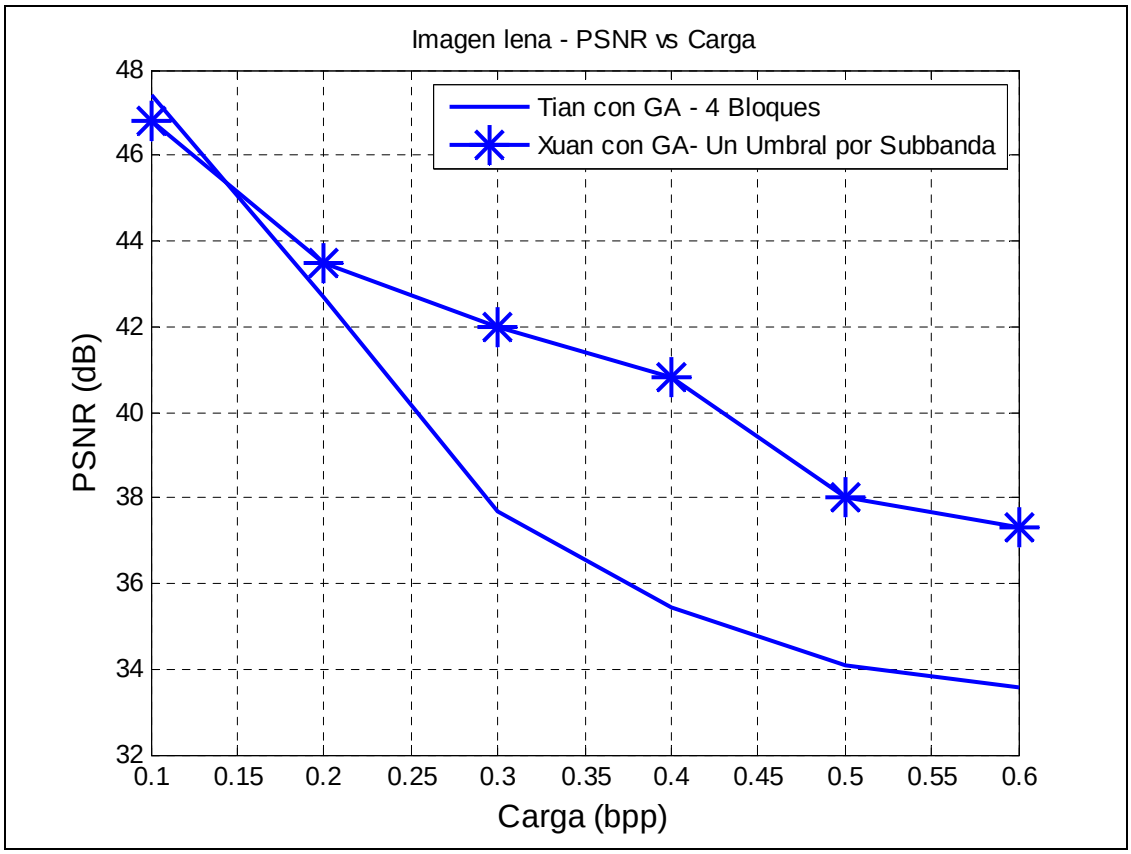

Figura 4.11- Comparación Tian y Xuan con GA – Capacidad (bpp) vs. PSNR (dB) - 'lena'

Carga útil (bpp)	**Imagen**			
	lena 512x512x8		**baboon 512x512x8**	
	PSNR(dB)	**MSE**	**PSNR(dB)**	**MSE**
0,04	50,48	0,58	31,71	43,82
0,1	46,80	1,54	Desbordamiento- Se debe comprimir el histograma	
0,2	43,61	2,82		
0,3	42,06	4,04		
0,4	40,81	5,28		
0,5	38,04	10,20		
0,6	37,35	11,94		

Tabla 4.9 -Xuan con GA- Capacidad (bpp) vs. PSNR (dB)- Un Umbral por Subbanda de Detalle

Carga Útil (bits)	cr-tórax 440x440x16		mr-rodilla 256x256x16		mr-columna 512x512x16	
	Capacidad (bpp)	PSNR(dB)	Capacidad (bpp)	PSNR(dB)	Capacidad (bpp)	PSNR(dB)
1024	0,005	83,71	0,015	91,73	0,004	121
2048	0,010	82,77	0,03	90,68	0,008	118
4096	0,015	83,30	0,06	87,07	Debe comprimirse el histograma	

Tabla 4.10 – Xuan con GA –Carga (bits) vs. PSNR (dB)- Imágenes Médicas

El histograma de la MR de rodilla, está centrado por lo que tiene gran capacidad de embebido sin necesidad de compresión del mismo. No ocurre lo mismo con la MR de columna. De este modo se complica automatizar el marcado con solo aplicar GA para determinar el mejor umbral. En algunas imágenes es posible, en otras no. En el caso de la CR del riñón cuyas intensidades pertenecen al intervalo [0 245] basta con comprimir 20 intensidades a la izquierda [0 20], para poder marcar. Se obtiene un PSNR de alrededor de 30 dB para cargas de 2048 bits.

4.3 Conclusiones de la Experimentación

El principal logro alcanzado es la automatización de la elección del umbral, incluso en aquellos casos en que la semilla inicial no permite alcanzar la capacidad pedida, el sistema se acomoda rápidamente y arroja buenos resultados. Si bien hay diferencias según el tipo de imagen, incluso dentro las imágenes no médicas aquellas como ‘baboon’ permiten marcado, solo que en este caso la capacidad efectiva es menor y la imagen obtenida tiene bajo PSNR, pero su SSIM es superior a otras, y visualmente no se advierte la marca, debido a la textura de la imagen.

Respecto a la división en bloques para poder detectar la zona de la adulteración, en caso de producirse esta, algunas imágenes la toleran mejor que otras (‘lena’, mejor que ‘boats’), pero en general todas mantienen bajo esta división un nivel de calidad aceptable.

Para mejorar los valores obtenidos se debería contar con equipos de computación de alta perfomance y aplicar programación en paralelo.

El uso de la transformada wavelet cdf(2,2) y compansión arroja imágenes de muy buena calidad, solo que hay que automatizar también la compresión del histograma, lo que aumenta la complejidad computacional. Una variante sería no comprimir el histograma de la imagen original, sino corregir aquellos coeficientes de las subbandas de detalle que producen el desbordamiento, almacenando esta información para lograr reversibilidad.

CAPÍTULO 5

CONCLUSIONES Y FUTUROS TRABAJOS

Conclusiones

Los algoritmos reversibles se han desarrollado especialmente en la última década, ya que los robustos no satisfacían las necesidades de las aplicaciones de seguridad, militares o médicas. Centrado el interés en algoritmos reversibles para ser aplicados a imágenes médicas, se procedió a la comparación de los métodos reversibles más relevantes en la literatura, eligiendo el de Tian por su buena capacidad, ya que no tiene tanta importancia la imperceptibilidad puesto que se va a recuperar la imagen original. Sin embargo se busca una imagen marcada de calidad aceptable para usuarios comunes aunque los profesionales, usuarios finales, estén autorizados a recuperar el archivo original. Otros algoritmos reversibles, como el de Ni, tienen muy buena calidad, pero su capacidad es baja, varía mucho con la imagen anfitriona, no es predecible, por lo que no son de utilidad para marcar imágenes médicas.

La contribución del trabajo es la optimización del algoritmo reversible para imágenes propuesto por Tian que aplicado a cuaternas posee una capacidad teórica máxima de embebido (0,75 bpp), la que se puede aumentar todavía más si se hace más de un recorrido de la imagen.

La optimización producida se logró a través del empleo de algoritmos genéticos. Se aplicó compansión en la transformada directa de Haar, lo que significa que para conseguir una determinada capacidad se varía un umbral, por encima del cual se comprimen los coeficientes a embeber, aumentando la capacidad del método en desmedro de la calidad.

La originalidad de la presentación consiste en conseguir que automáticamente, mediante la aplicación de GA, se calcule este umbral y se obtenga la mejor calidad de la imagen marcada, indicada por el MSE mínimo o bien por el máximo PSNR o SSIM, para una cierta capacidad. De este modo no es necesaria la intervención del usuario o actuar por "prueba y error" así como tampoco se deben emplear métodos de fuerza bruta explorando todo el espacio de soluciones para obtener la mejor. Es posible que para la población inicial no se pueda alcanzar la capacidad deseada, sin embargo el algoritmo converge hasta cumplir el requisito, dentro de los límites teóricos y de lo que permita la imagen en particular.

La convergencia obtenida es automática y rápida, ventaja aprovechable en todo tipo de imágenes, encontrando siempre la mejor solución cualitativa (menor MSE, o mayor PSNR o SSIM) para la carga deseada, dentro de los límites teóricos impuestos por el método.

En el algoritmo presentado se ha alcanzado como extremo inferior una capacidad de 0,35 bpp para todo tipo de imágenes, esto significa un mínimo de carga de 91.750 bits para imágenes de 512x512 píxeles, lo que es suficiente para asegurar identidad, autenticar y agregar datos sobre el origen del archivo. Por otra parte, esta capacidad se puede aumentar, a costa de una degradación de la calidad, mediante un segundo recorrido de la imagen (variando el píxel pivote) o bien tomando 2 bits para marcar, factible en imágenes médicas que tienen 16 bpp.

Se aplicó el mismo método para el embebido en los coeficientes de la transformada wavelet cdf(2,2). Se concluyó que el algoritmo que utiliza la transformada entera de Haar presenta ventajas con respecto al que utiliza la transformada wavelet cdf(2,2), ya que no requiere preprocesamiento. En este último caso, en cambio, puede ser necesaria la compresión del histograma para lograr reversibilidad. Basta que un solo píxel se trunque para impedir la recuperación de la imagen original. El algoritmo de Tian previene el desbordamiento previamente a la incrustación de la marca, mediante un sencillo cálculo aritmético. Por el contrario, cuando se aplica cdf(2,2), luego de realizado el marcado se sabe si hay o no truncamiento de los valores de intensidad con pérdida de información y si lo hubo se debe comprimir el

histograma de la imagen original, y repetir el marcado tantas veces como sea necesario hasta evitar el desbordamiento, considerando que dicha compresión debe ser la mínima posible puesto que produce una degradación extra en la imagen. Otra posibilidad es predecir la compresión que debe hacerse en el histograma, pero de todas maneras el método precisa mayor carga extra para lograr reversibilidad y tiene más complejidad computacional. No obstante, se debe reconocer que el algoritmo que utiliza la transformada wavelet cdf(2,2), en aquellos casos en que no es necesaria la compresión del histograma, produce una imagen de muy buena calidad.

Otra contribución realizada consiste en la experimentación de la división de la imagen en bloques utilizando el algoritmo de Tian. Esta operación si bien trae aparejada una sobrecarga puesto que debe embeberse el umbral de cada bloque, en algunas imágenes mejora la perfomance aprovechando la correlación entre vecinos. Esta división por bloques adiciona robustez, ya que se puede embeber y recuperar la misma información (por ejemplo un logo o un resumen hash) en varios bloques. También se puede determinar en qué bloque se produjo una alteración, si la hubo, así como evitar el embebido en ciertos bloques (los de la ROI), o bien extraer un resumen de la ROI e insertarlo en los bloques de la RONI.

El embebido puede estar acompañado de métodos criptográficos clásicos, y es deseable que así lo sea para introducir un segundo nivel de seguridad. Por ejemplo, puede embeberse la marca según una clave y también encriptar los metadatos previamente a su incrustación. Para determinar que se mantuvo la integridad de los datos, parte de la marca debe estar compuesta por un resumen MD5 o SHA. La realización de este embebido por bloques, permite determinar qué bloque fue modificado, si así hubiera ocurrido.

Futuros Trabajos

Sería interesante estudiar si es posible determinar un umbral típico para un tipo dado de imagen médica, algo que no es posible para las comunes. Eso acotaría la búsqueda del umbral en imágenes médicas del mismo tipo.

Por otra parte, interesa analizar qué sucede con la calidad ante ataques no intencionales tales como la compresión, aunque ya dejaría de cumplirse el objetivo principal: que los médicos accedan a la imagen original.

Es de interés seguir estudiando la aplicación de algoritmos genéticos en el marcado de imágenes con otros objetivos tales como decidir dónde conviene embeber, aunque en este caso no se aproveche toda la capacidad.

El método empleado se puede extender a videos (por ejemplo los producidos por cámaras usadas con fines de seguridad) y otros objetos digitales como textos y bases de datos.

Actualmente, las plataformas de cloud computing requieren especiales defensas contra ataques. Esto es particularmente exigido por las plataformas de e-health. Están surgiendo nuevos paradigmas de watermarking aplicables a las mismas, donde se marca también el software. A estos escenarios se puede extender el estudio realizado.

Se espera también abordar el problema inverso, dado un objeto digital detectar si está marcado. Estos temas son motivo de investigación en el mundo actual, no solo para proteger derechos de autor e integridad de datos, sino como medio de defensa de los pueblos. Su tratamiento forma parte de lo que actualmente se conoce como Cyberseguridad.

Bibliografía

[1] I. Cox, M. Miller and J. Bloom. *"Digital Watermarking and Fundamentals"*. Ed. Morgan Kaufmann, Series in Multimedia Information and Systems, 2002.

[2] I. Cox and M. Miller. *"The first 50 years of electronic watermarking*". EURASIP Journal on Applied Signal Processing, pp. 126-132, 2002.

[3] I. Cox, J. Kilian, T. Leighton and T. Shamoon. *"Secure Spread Spectrum watermarking for multimedia"*. NEC Research Institute- Technical Report, 1995.

[4] M. Barni, F. Bartolini, V. Cappellini and A. Piva. *"A DCT-domain System for Robust Image Watermarking"*. Journal on Signal Processing, Vol. 66 , Issue 3, pp. 357-372, Ed. Elsevier, 1998.

[5] D. Kundur and D.Hatzinakos. *"Digital watermarking for tamper proofing and authentication"*. Proceedings of the IEEE Special Issue on Identification and Protection of Multimedia Information, Vol. 87, pp. 1167-1180, 1999

[6] J. Fridrich. *"Aplications of data hiding in digital images"*. IEEE Proceedings of the 5th Symposium on Signal Processing and its Applications, ISSPA'99, Vol. 1, 1999.

[7] M. Lewandovski, R. Meana, M. Morrison and S. Katkouri. *"A novel method for watermarking sequential circuits"*. IEEE International Symposium on Hardware Oriented Security, pp. 21-24, 2012.

[8] R. Sion, M. Atallah and S. Prabhakar. *"Rights protection for relational data"*. IEEE Transactions on Knowledge and Data Engineering, Vol. 16, Issue 12, pp. 1509-1525, 2004.

[9] K. Jawad and A. Khan. *"Genetic Algorithm and Difference Expansion based Reversible Watermarking for Relational Databases"*. The Journal of Systems and Software, Ed. Elsevier, pp. 2742-2753, 2013.

[10] F. Petitcolas, R. Anderson and M. Kuhn. *"Information hiding- A survey"*. Proceedings IEEE Special Issue on Protection of Multimedia Content, pp. 1062-1078, 1999.

[11] M. Barni. *"What is the future of Watermarking"*. IEEE Signal Processing Magazine, Vol 20, Issue 6, pp. 53-59, 2003.

[12] *http://www.digitalwatermarkingalliance.org*

[13] *http:/ www.infoleg.gov.ar/infolegInternet/anexos/160000-164999/160432/norma.htm*

[14] *http://www.nema.org*

[15] G. Coatrieux, H. Maitre, B. Sankur, Y. Rolland and R. Collorec. *"Relevance of Watermarking in Medical Imaging"*. Proceedings of the IEEE EMBS Conference on Information Technology Applications in Biomedicine, Arlington, USA, pp. 250-255, 2000.

[16] U. Rajendra Acharya, R. Acharya, P. Subanna Bhat and U. Niranjan. *"Compact Storage of medical images with patient information"*. IEEE Transactions on Information Technology in Biomedicine, Vol. 5, Issue 4, pp. 320-323, 2001.

[17] M. Planitz and A. Maeder. *"Medical image watermarking: a study on image degradation"*. Workshop on Digital Image Computing, WDIC, pp. 3-8, 2005.

[18] J. Dowling, B. Planitz, A. Maeder, J. Du, B. Pham *et al*. *"Visual Quality Assessment of Watermarked Medical Images*". Proceedings Medical Imaging 2007: Image Perception, Observer Performance, and Technology Assessment: SPIE, Vol. 6515, Progress in Biomedical Optics and Imaging, San Diego, CA, USA, 2007.

[19] A. Wakatani. *"Digital Watermarking for ROI Medical Images by Using Compressed Signature Image"*. Proceedings of the 35th Hawai International Conference on System Sciences, 2002.

[20] J. Zain , L. Baldwin and M. Clarke. *"Reversible Watermarking for Authentication of Dicom Images"*. Engineering in Medicine and Biology Society, 2004. IEMBS'04. 26th Annual International Conference of the IEEE, Vol. 2, pp. 3237-3240, 2004.

[21] A. De Rosa, F. Bartolini, V. Cappellini and A. Piva. *"Watermarking Systems For Hidden Annotation In Medical Images"*. Proceedings of 4th COST 276 Workshop on Information and Knowledge Management for Integrated Media Communications, Francia, 2003.

[22] N. Memon, S. Gilani and A. Ali. "*Watermarking of Chest CT Scan Medical Images for Content Authentication"*. ICICT 09, International Conference on Information and Computation Technologies, pp. 175-180, 2009.

[23] C. Woo, J. Du and B. Pham. "*Multiple Watermark Method For Privacy Control and Tamper Detection in Medical Images"*. Workshop on Digital Image Computing, WDIC 2005, Australia, 2005.

[24] A. Giakoumaki, S. Pavlopoulos and D. Koutsouris. "*Multiple image watermarking applied to health information management"*. Information Technology in Biomedicine, IEEE Transactions, Vol. 4, Issue 4, pp. 722-732, 2006

[25] N. Memon, S. Gilani and S. Qayoom. "*Multiple Watermarking of Medical Images for Content Authentication and Recovery"*. IEEE, 13th Conference Multitopic, INMIC 2009, pp. 1-6, 2009.

[26] C. Gutiérrez, G. Kakani, R. Verma and T. Wang. *"Digital Watermaking of Medical Imagen for Mobile Devices"*. IEEE International Conference on Sensor Networks, Ubiquitous and Trustworthy Computing, pp. 421-425, 2010.

[27] J. Zain and M. Clarke. "*Security in Telemedicine: Issues in Watermarking Medical Images"*. SETIT 2005, 3th Conference on Sciences of Electronic, Technologies of Information and Telecommunications, Túnez, 2005.

[28] A. Kannammal, K. Pavithra and S. SubhaRani. "*Double Watermarking of Dicom Medical Images using Wavelet Decomposition Technique"*. European Journal of Scientific Research, Vol. 70, N°1, pp. 46-55, 2012.

[29] O. Rubio, A. Alesanco and J. García. "*Seamless Integration of Watermarks in Dicom Images*". Computing in Cardiology Conference (CinC), IEEE Publication Conferences, pp. 25-28, 2013.

[30] S. Hisham, S. Liew and J. Zain. "*A Quick Glance at Digital Watermarking in Medical Images"*. Biomedical Engineering Research, Vol. 2, Issue 2, pp. 79-87, Jun 2013.

[31] Z. Wang, A. Bovik, H. Sheikh and E. Simoncelli. *"Image Quality Assurement: from Error Visibility to Structural Similarity"*. IEEE Transactions on Image Processing, Vol. 13, N°4, pp. 600-612, 2004.

[32] F. Hartung, J. Su and B. Girod. "*Spread Spectrum Watermarking: Malicious Attacks and Counterattacks"*. Proceedings of SPIE Security and Watermarking on Multimedia Contents, Vol. 3657, pp. 147-158, 1999.

[33] D. Kundur and D. Hatzinakos. "*Attack Characterization for Effective Watermarking"*. Proceedings of the IEEE International Conference on Image Processing, ICIP 99, Vol. 2, pp. 240-244, 1999.

[34] J. Tian. "*Reversible Data Embedding Using a Difference Expansion"*. IEEE Transactions on Circuits and Systems for Video, Vol. 13, N°8, pp. 890-896, 2003.

[35] Z. Ni, Q. Shi, N. Ansari and W. Su. *"Reversible Data Hiding"*. IEEE Transactions on Circuits and Systems for Video Technology, Vol. 16, N°3, pp. 354-361, 2006.

[36] G. Xuan, J. Zhu, J. Chen, Y. Q. Shi, Z. Ni and W. Su. *"Distortionless data hiding based on integer wavelet transform"*. Electronics Letters, pp. 1646-1648, 2002.

[37] R. Caldelli, F. Filippini and R. Becarelli. *"Reversible Watermarking Techniques: An Overview and Classification"*. Eurasip Journal on Information Security, Vol. 2010, Art. ID 134546, 19 páginas, 2010.

[38] A. M. Alattar. *"Reversible Watermark Using Difference Expansion of Quads"*. Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, Vol. 3, pp. 377-380, 2004.

[39] S. Weng, Y. Zhao, J. Pan and R. Ni. *"A novel reversible watermarking based on an integer transform"*. International Conference on Image Processing ICIP, Vol. 3, pp. 241-243, 2007.

[40] B. Sklar. *"Digital Communications: Fundamentals and Applications"*. Ed. Prentice Hall, pp. 84-85, 2001.

[41] B. Yang, M. Schmucker, W. Funk, C. Busch and S. Sun. *"Integer DCT-based Reversible Watermarking for Images Using Companding Technique"*.Proceedings of SPIE, Vol 5306, pp. 5306-5041, 2004.

[42] G. Xuan, C. Yang, Y. Zhen, Y. Shi and Z. Ni. *"Reversible Data Hiding using Integer Wavelet Transform and Companding Technique"*. Lecture Notes in Computer Science 3304, pp. 115-125, Ed. Springer Verlag, Cox *et al*. Ed., 2005.

[43] G. Xuan, Y. Q. Shi, C. Yang, Y. Zheng, D. Zou and P. Chai. *"Lossless Data Hiding Using Integer Wavelet Transform and Threshold Embedding Technique"*. Multimedia and Expo. ICME 2005. IEEE International Conference, pp. 1520-1523, 2005.

[44] L. Vargas and E. Vera. "A*n Implementation of Reversible Watermarking for Still Images"*. IEEE Latin- America Transaction Vol. 11 Issue: 1 Feb 2013 ISSN: 1548-0992. pp. 54-59. Available: http:// ewh.ieee.org/ reg/9/ etrans/.

[45] M. Arsalan, S. Malik and A. Khan. *"Intelligent Reversible Watermarking in Integer Wavelet Domain for Medical Images"*. The Journal of Systems and Software, Ed. Elsevier, Vol. 85, Issue 4, pp. 883-894, 2012.

[46] K. Sayood. *"Introduction to Data Compression"*. Ed. Morgan Kauffman Series in Multimedia Information and Systems, 3th Ed, pp. 81-115, 183-190, 2006.

[47] K. Soman, K. Ramachandran and N. Resmi. *"Insight into Wavelets, from Theory to Practice"*. Ed. Prentice Hall, 3th Ed., 2010.

[48] J. Holland. *"Outline for a logical theory of adaptive systems"*. JACM (Journal of ACM), Vol. 9 N°. 3, pp. 279-314, 1962.

[49] J. Holland. *"Genetic Algorithms"*. Scientific American, pp. 67-72, 1992.

[50] R. L. Haupt and S. E. Haupt. *"Practical Genetic Algorithms"*. Ed. Wiley-Interscience, 2nd E. Edition, 2004.

[51] M. Affenzeller, S. Winkler, S. Wagner and A. Beham. *"Genetic Algorithms and Genetic Programming. Modern Concepts and Practical Applications"*. CRC Press, Taylor and Francis Group, LLC, 2009.

[52] T. Blickle and L. Thiele. *"A Comparison of Selection Schemes used in Genetic Algorithms*". ·Ed. TIK Report, 2th Edition, 1995.

[53] F. Y. Shih and Y. T. Wu. *"Robust Watermarking and Compression for Medical Images Based on Genetic Algorithm"*. Journal of Information Sciences, pp. 200-216, 2005.

[54] H. Huang, J. Pan, Y. Huang, F. Wang and K. Huang. *"Progressive Watermarking Techniques using Genetic Algorithms"*. Circuits, Systems and Signal Processing, Vol. 26, Issue 5, pp. 671-687. Ed. Springer, 2007.

[55] B. Sikander, M. Ishtiqa, M. Arfan, M. Tariq and A. Mirza. *"Adaptive Digital Watermarking of Images Using Genetic Algorithm"*. 2010 ICISA, International Conference on Information Science and Applications, pp. 1-8, 2010.

[56] S. Goyal and R. Gupta. *"Optimization of Fidelity with Adaptive Genetic Watermarking Algorithm using Tournament Selection"*. International Journal of Advanced Science and Technologie, Vol. 30, pp. 55-65,2011.

[57] J. Usman. A. Khan, A. Ali and T. Choi. *"Reversible Watermarking Based on Intelligent Coefficient Selection and Integer Wavelet Transform"*. IJICIC (International Journal of Innovative Computing Information and Control), pp. 1909-1914, 2010.

[58] K. Jawad and A. Khan. *"Genetic Algorithm and Difference Expansion Based Reversible Watermarking for Relational Databases"*. The Journal of Systems and Software, Ed. Elsevier, pp. 2742-2753, 2013.

[59] *http://www.mathworks.com/matlabcentral/fileexchange/authors/26208*

Listado de Programas

Tian con Quads y GA

Programa Principal

```
%% Diseño e implementación de marcas de agua
%  Laura Vargas
% Embebido con GA aplicado a Selección Umbral en Tian con Quads
clear all;
close all;
clc;
global imgoriginal
global TamF
global TamC
global TotalBloques
global M
global N
global K
global L
global LargoWaterPD
global WFinal
global V0
global V1
global V2
global V3
global DIREC
global Tm
global TM
global BPP
global LMax
global OrdenF
global OrdenC

BPP=8;%bits por pixel
%BPP=16
%Se puede tomar de la información del archivo
%info=iminfo(imagen); o info=dicominfo(imagen)
%BPP=info.depth;

LMax=(2^BPP)-1;
OrdenF=2; %Cantidad de bloques según fila
OrdenC=2; %Cantidad de bloques según columna

%%Lectura de imagen de entrada
% la matriz de la imagen leida se guarda en una variable "imgoriginal"
imagen = input('Por favor, ingrese el nombre de la imagen a marcar con la extension (ej. nombre.ext): \n','s');
imgoriginal = imread(imagen); %para imágenes comunes
%imgoriginal=dicomread(imagen); %Si es imagen dicom
%save imgoriginal se puede grabar para pruebas

if(length(size(imgoriginal)) == 3)    %funcion para saber si la imagen esta en colores
   imgoriginal = rgb2gray(imgoriginal);         %convierte en escala de grises
```

```
end

%Cuando la matriz tiene alguna dimensión impar es necesario sumar 1 en la dimensión
%en M y/o N segun corresponda
[M,N]=size(imgoriginal);
if  rem(M,2)~= 0
   imgoriginal(M+1,:)=imgoriginal(M,:);
end
if  rem(N,2)~= 0
   imgoriginal(:,N+1) = imgoriginal(:,N);
end

[M,N]=size(imgoriginal);%Obtengo los nuevos valores si fueron cambiados
%M filas, N columnas
K=M/2;
L=N/2;
%Se debe validar que M y N sean múltiplos

TamC=N/OrdenC;
TamF=M/OrdenF;%El tamaño en bits de los bloques será TamF*TamC

TotalBloques=M*N/TamF/TamC;%Total Bloques

%%Generación Marca
LargoWaterPD=round(0.1*M*N); %Se va variando para probar distintas cargas
WFinal=randi([0 1],1,LargoWaterPD);%Se toma una única marca generada aleatoriamente
save WFinal
%%Si deseamos usar una función resumen para asegurar integridad, hacer lo siguiente
%WatermarkPD=CalcMD5(imgoriginal); %Función que calcula resumen
%WPD=uint8(WatermarkPD);%Se debe cambiar por 16 si corresponde en lo que sigue
%WP=dec2bin(WPD,8); %cada 8 binario
%LargoWaterPD=length(WatermarkPD)*8;

%WPN=reshape(WP,1,LargoWaterPD);
%for i=1:LargoWaterPD
%   Wfinal(i)=str2num(WPN(i));
%end
%save WPD
%save WP

%% Determinación de la transformada
DIREC=[2 2];%según matriz dos por dos . La va a tomar como vector
%Calculo promedio del quad y las diferencias
%Funciones generales
avFunc=inline('floor( ( x(1)+x(2)+x(3)+x(4) )/4)');
dif1=inline(' x(1)-x(3)');%Ojo numera de arriba hacia abajo de izquierda a derecha
dif2=inline('x(1)-x(2)');
dif3=inline('x(1)-x(4)');

%Aplicación a la imagen- Obtención de la transformada
V0=blkproc(double(imgoriginal),DIREC,avFunc);
V1=blkproc(double(imgoriginal),DIREC,dif1);
V2=blkproc(double(imgoriginal),DIREC,dif2);
V3=blkproc(double(imgoriginal),DIREC,dif3);
Concatenada=[V0 V1; V2 V3];
```

```
%% Determinación del Umbral Inicial
%Tomo las medianas de los bloques
%Se puede hacer con los valores medios. Función mean2(Matriz)
%ParaUmbral=[V1 V2 V3];Son las matrices que serán comprimidas
p=1;
for i=1:TamF:M
  for j=1:TamC:N
     S(p)=round(median(median(abs(Concatenada(i:i+TamF-1,j:j+TamC-1)))));
     p=p+1;
  end
end

%Genero por comodidad matriz umbral
p=1;
for i=1:M
    for j=1:N
      if mod(j,TamC)==1 && j ~= 1
         p=p+1;%cambio al pasar de columna
      end
      MUmbral(i,j)=S(p);
    end    %Fin Columna
    %Paso de fila
     if mod(i,TamF)==0 && i~= 1
        p=p+1;
     else
        p=p-N/TamC+1;
     end
end

%% Restricciones
%los valores bajos no requieren compresión porque no hay
%chances de under u overflow
%para los valores bajos f(x)=x
%para los coeficientes mayores al umbral se hará compresión lineal
%Cada una de las diferencias se clasifica en 2 subconjuntos
%según sea mayor o menor que el umbral
%Conjunto menor que umbral
%Conjunto mayor o igual que umbral
% Por facilidad tomamos matriz umbral
% y matriz comprimida

%También calculamos el mínimo y el máximo de Umbrales

p=1;
for i=1:TamF:M
  for j=1:TamC:N
     Tm(p)=min(min(abs(Concatenada(i:i+TamF-1,j:j+TamC-1))));
     TM(p)=max(max(abs(Concatenada(i:i+TamF-1,j:j+TamC-1))));
     p=p+1;
  end
end


%% Parámetros Algoritmo Genético
```

```
% ****************************************************
% Configuración Algoritmo Genético
% ****************************************************
options = gaoptimset;

% Población Genética
options = gaoptimset(options,'PopulationSize', 30);
options = gaoptimset(options, 'PopulationType', 'doubleVector');%
options=gaoptimset(options,'InitialPopulation',S);
%En la mayoría de los casos conviene que la semilla sea la mediana de los bloques
%Abajo se indican otras posibilidades
%options = gaoptimset(options, 'InitialPopulation',TM-1);
%options = gaoptimset(options, 'InitialPopulation',round((Tm+TM)/2));
%options= gaoptimset(options, 'InitialPopulation',Tm+1);
options = gaoptimset(options,'PopInitRange',[Tm+1;TM-1]);
options = gaoptimset(options, 'CreationFcn', @gacreationuniform);

% Criterios de parada
options = gaoptimset(options, 'Generations',50);
%options = gaoptimset(options, 'FitnessLimit', 1.0001);
%options = gaoptimset(options,'TolFun', 3);
options = gaoptimset(options,'StallTimeLimit',3000);
options = gaoptimset(options,'StallGenLimit', 40);

% ****************************************************
%% Operadores Genéticos
% ****************************************************
% Elitismo
options = gaoptimset(options, 'EliteCount', 2);
% Operador de selección
options = gaoptimset(options, 'SelectionFcn', @selectionroulette);
%options= gaoptimset(options, 'SelectionFcn', @selectiontournament);
% Algoritmo de cruce
options = gaoptimset(options, 'CrossoverFcn', @crossoverscattered);
options = gaoptimset(options, 'CrossoverFraction', 0.8);
% Algoritmo de mutación
% Determinamos que no sea posible que arroja valores de umbral fuera de los límites
options = gaoptimset(options, 'MutationFcn', @mutationadaptfeasible);
%options= gaoptimset(options, 'MutationFcn', @mutationuniform) ;
%Algoritmo de selección
%rank
options = gaoptimset(options, 'FitnessScalingFcn', @fitscalingrank);

%% Llamado a función que ejecuta el cálculo del fitness (MSE)
%% GA y Embebido Final
[x, fval, reason, output, population, scores]= ga(@gaembebido1,TotalBloques,options);
fprintf('El mejor valor encontrado fue : %g\n', fval);
fprintf('La cantidad de generaciones para hallarlo fue : %d\n', output.generations);
fprintf('La cantidad de evaluaciones realizadas fue : %d\n', output.funccount);

%%Embebido Final
%Lo separamos para grabar la imagen obtenida
disp('Embebido Final: ');
[fitnessf] = gaembebido1f(x);
```

Función Embebido

```
function [fitness]=gaembebido1(T)
%T es el umbral que se elegirá según MSE (fitness) y capacidad
% Función para Tian con Cuaternas
global imgoriginal
global TamF
global TamC
global TotalBloques
global M
global N
global K
global L
global LargoWaterPD
global WFinal
global V0
global V1
global V2
global V3
global DIREC
global Tm
global TM
global BPP
global LMax
global OrdenF

T=round(abs(T));

for i=1:length(T)
   if (Tm(i)>T(i) || TM(i)<T(i))
      fitness=10000;
      return
   end
end

%Se utiliza matriz umbral por facilidad

MUmbral=zeros(M,N);
Error=zeros(M,N);

p=1;
for i=1:M
    for j=1:N
      if mod(j,TamC)==1 && j ~= 1
          p=p+1;%cambio al pasar de columna
      end
      MUmbral(i,j)=T(p);
    end    %Fin Columna
    %Paso de fila
     if mod(i,TamF)==0 && i~= 1
        p=p+1;
     else
        p=p-N/TamC+1;%N o M
     end
end
```

```
%%Compresión
%Cada una de las diferencias V1, V2 y V3 se clasifica en 2 subconjuntos
%según sea mayor o menor que el umbral
%Conjunto menor que umbral: VxQ1
%Conjunto mayor o igual que umbral: VxQ2
%Luego se comprime VXQ2
V1Q1=abs(V1)<MUmbral(1:M/2, N/2+1:N);
V1Q1=V1Q1.*V1;
V1Q2=abs(V1)>=MUmbral(1:M/2, N/2+1:N);
V1Q2=V1Q2.*sign(V1).*(floor((abs(V1)-MUmbral(1:M/2, N/2+1:N))/2)+MUmbral(1:M/2,
N/2+1:N));
V1Q=V1Q1+V1Q2;

V2Q1=abs(V2)<MUmbral(M/2+1:M, 1:N/2);
V2Q1=V2Q1.*V2;
V2Q2=abs(V2)>=MUmbral(M/2+1:M, 1:N/2);
V2Q2=V2Q2.* sign(V2).*(floor((abs(V2)-MUmbral(M/2+1:M,
1:N/2))/2)+MUmbral(M/2+1:M, 1:N/2));
V2Q=V2Q1+V2Q2;

V3Q1=abs(V3)<MUmbral(M/2+1:M, N/2+1:N);
V3Q1=V3Q1.*V3;
V3Q2=abs(V3)>=MUmbral(M/2+1:M, N/2+1:N);
V3Q2=V3Q2.* sign(V3).*(floor((abs(V3)-MUmbral(M/2+1:M,
N/2+1:N))/2)+MUmbral(M/2+1:M, N/2+1:N));
V3Q=V3Q1+V3Q2;
%Fin de la compresión verificada

%Se calculan errores para los valores mayores o iguales al umbral
%Estos se deben almacenar para lograr reversibilidad
%Primer paso: calcular la expansión de V1Q, V2Q y V3Q
%Expansion- Principio
V1E1=abs(V1Q)<MUmbral(1:M/2, N/2+1:N);
V1E1=V1E1.*V1Q;
V1E2=abs(V1Q)>=MUmbral(1:M/2, N/2+1:N);
V1E2=V1E2.*sign(V1Q).*(2*abs(V1Q)-MUmbral(1:M/2, N/2+1:N));
V1E=V1E1+V1E2;

V2E1=abs(V2Q)<MUmbral(M/2+1:M, 1:N/2);
V2E1=V2E1.*V2Q;
V2E2=abs(V2Q)>=MUmbral(M/2+1:M, 1:N/2);
V2E2=V2E2.*sign(V2Q).*(2*abs(V2Q)-MUmbral(M/2+1:M, 1:N/2));
V2E=V2E1+V2E2;

V3E1=abs(V3Q)<MUmbral(M/2+1:M, N/2+1:N);
V3E1=V3E1.*V3Q;
V3E2=abs(V3Q)>=MUmbral(M/2+1:M, N/2+1:N);
V3E2=V3E2.*sign(V3Q).*(2*abs(V3Q)-MUmbral(M/2+1:M, N/2+1:N));
V3E=V3E1+V3E2;
%Fin de la expansión-

%Sigue cálculo de error de expansión en C1 mayor o igual a T
Error1=abs(V1)-abs(V1E);
Error2=abs(V2)-abs(V2E);
```

```
Error3=abs(V3)-abs(V3E);

RErrort=reshape([Error1 Error2 Error3],K*L*3,1);
%Coloca un vector a continuación del otro por columna
cuentaRErrort=zeros(2,1);
%Todos los calculos anteriores se hicieron en forma global
%sin importar si se usará o no el valor

%%Proceso de clasificación
%Se clasifica en conjuntos C1 (expansibles), C2 (cambiables) y C3(inútiles)
%Aplico DE según Tian en C1 (grupo de los expansibles)
%Primer paso: cálculo de los Vtilde y de las diferencias
%Para determinar si pertenece al grupo C1
V1tilde0=2*V1Q+0;
V1tilde1=2*V1Q+1;
V2tilde0=2*V2Q+0;
V2tilde1=2*V2Q+1;
V3tilde0=2*V3Q+0;
V3tilde1=2*V3Q+1;

U0tilde0=V0+ceil((V1tilde0+V2tilde0+V3tilde0)/4);
U0tilde1=V0+ceil((V1tilde1+V2tilde1+V3tilde1)/4);

DIFV1tilde0=(U0tilde0-V1tilde0);% Es el nuevo U1 para b=0
DIFV1tilde1=(U0tilde0-V1tilde1);% Es el nuevo U1 para b=1
DIFV2tilde0=(U0tilde0-V2tilde0);% Es el nuevo U2 para b=0
DIFV2tilde1=(U0tilde0-V2tilde1);
DIFV3tilde0=(U0tilde0-V3tilde0);
DIFV3tilde1=(U0tilde0-V3tilde1);

DIFV1tilde01=(U0tilde1-V1tilde0);%
DIFV1tilde11=(U0tilde1-V1tilde1);%
DIFV2tilde01=(U0tilde1-V2tilde0);%
DIFV2tilde11=(U0tilde1-V2tilde1);
DIFV3tilde01=(U0tilde1-V3tilde0);
DIFV3tilde11=(U0tilde1-V3tilde1);

%%Calcula si VQ pertenece a C2 (cambiable):
V1tilde20=2*floor(V1Q/2)+0;
V1tilde21=2*floor(V1Q/2)+1;
V2tilde20=2*floor(V2Q/2)+0;
V2tilde21=2*floor(V2Q/2)+1;
V3tilde20=2*floor(V3Q/2)+0;
V3tilde21=2*floor(V3Q/2)+1;

U0tilde20=V0+ceil((V1tilde20+V2tilde20+V3tilde20)/4);
U0tilde21=V0+ceil((V1tilde21+V2tilde21+V3tilde21)/4);

DIFV1tilde20=(U0tilde20-V1tilde20);%Es el nuevo U1 para b=0 esto para C2
DIFV1tilde21=(U0tilde20-V1tilde21);%Es el nuevo U1 para b=1 esto para C2
DIFV2tilde20=(U0tilde20-V2tilde20);
DIFV2tilde21=(U0tilde20-V2tilde21);
DIFV3tilde20=(U0tilde20-V3tilde20);
DIFV3tilde21=(U0tilde20-V3tilde21);
```

```
DIFV1tilde201=(U0tilde21-V1tilde20);
DIFV1tilde211=(U0tilde21-V1tilde21);
DIFV2tilde201=(U0tilde21-V2tilde20);
DIFV2tilde211=(U0tilde21-V2tilde21);
DIFV3tilde201=(U0tilde21-V3tilde20);
DIFV3tilde211=(U0tilde21-V3tilde21);

% Se optó por hacer el cálculo en bloques
%Determinación del Mapa de Ubicación
IndiceC2=0;
IndiceC3=0;
C2i=[];
C2j=[];
C3i=[];
C3j=[];
C2V1bitperdido=[];
C2V2bitperdido=[];
C2V3bitperdido=[];
LMap=zeros(K,L);
%Guarda unos si el quad pertenece a C1

%Contadores totales de cada subconjunto
CuentaClaseC1mayorigualT=0;
CuentaClaseC1menorT=0;
CuentaClaseC2=0;
%CuentaResto=0; No hace falta contar aquellos que no hace falta marcar

for i=1:K
  for j=1:L
    if (U0tilde0(i,j)>=0 && U0tilde0(i,j)<=LMax && U0tilde1(i,j)>=0 &&
U0tilde1(i,j)<=LMax ...
        && DIFV1tilde0(i,j)>=0 && DIFV1tilde0(i,j)<=LMax ...
        &&DIFV1tilde1(i,j)>=0  && DIFV1tilde1(i,j)<=LMax ...
        &&DIFV2tilde0(i,j)>=0 &&DIFV2tilde0(i,j)<=LMax && ...
        DIFV2tilde1(i,j)>=0 && DIFV2tilde1(i,j)<=LMax && ...
        DIFV3tilde0(i,j)>=0 && DIFV3tilde0(i,j)<=LMax ...
        && DIFV3tilde1(i,j)>=0 && DIFV3tilde1(i,j)<=LMax ...
        && DIFV1tilde01(i,j)>=0 && DIFV1tilde01(i,j)<=LMax ...
        &&DIFV1tilde11(i,j)>=0 && DIFV1tilde11(i,j)<=LMax ...
        &&DIFV2tilde01(i,j)>=0 &&DIFV2tilde01(i,j)<=LMax ...
        && DIFV2tilde11(i,j)>=0 && DIFV2tilde11(i,j)<=LMax ...
        && DIFV3tilde01(i,j)>=0 && DIFV3tilde01(i,j)<=LMax ...
        && DIFV3tilde11(i,j)>=0 && DIFV3tilde11(i,j)<=LMax)

        if ( abs(V1(i,j))>=MUmbral(i,j) || abs(V2(i,j))>=MUmbral(i,j) ||
abs(V3(i,j))>=MUmbral(i,j) )
         CuentaClaseC1mayorigualT=CuentaClaseC1mayorigualT+1;
       else
         CuentaClaseC1menorT=CuentaClaseC1menorT+1;
    end
       LMap(i,j)=1;
    else
       if (U0tilde20(i,j)>=0 && U0tilde20(i,j)<=LMax && U0tilde21(i,j)>=0 &&
U0tilde21(i,j)<=LMax ...
        && DIFV1tilde20(i,j)>=0 && DIFV1tilde20(i,j)<=LMax ...
```

```
        &&DIFV1tilde21(i,j)>=0 && DIFV1tilde21(i,j)<=LMax ...
        &&DIFV2tilde20(i,j)>=0 &&DIFV2tilde20(i,j)<=LMax && ...
        DIFV2tilde21(i,j)>=0 && DIFV2tilde21(i,j)<=LMax ...
        && DIFV3tilde20(i,j)>=0 && DIFV3tilde20(i,j)<=LMax ...
        && DIFV3tilde21(i,j)>=0 && DIFV3tilde21(i,j)<=LMax ...
        && DIFV1tilde201(i,j)>=0 && DIFV1tilde201(i,j)<=LMax ...
        &&DIFV1tilde211(i,j)>=0  && DIFV1tilde211(i,j)<=LMax ...
         &&DIFV2tilde201(i,j)>=0 &&DIFV2tilde201(i,j)<=LMax && ...
        DIFV2tilde211(i,j)>=0 && DIFV2tilde211(i,j)<=LMax ...
        && DIFV3tilde201(i,j)>=0 && DIFV3tilde201(i,j)<=LMax ...
        && DIFV3tilde211(i,j)>=0 && DIFV3tilde211(i,j)<=LMax)
          IndiceC2=IndiceC2+1;
           C2i(IndiceC2)=i;
           C2j(IndiceC2)=j;
      CuentaClaseC2=CuentaClaseC2+1;
           C2V1bitperdido(IndiceC2)=bitget(abs(V1Q(i,j)),1);
           C2V2bitperdido(IndiceC2)=bitget(abs(V2Q(i,j)),1);
           C2V3bitperdido(IndiceC2)=bitget(abs(V3Q(i,j)),1);
           else
           IndiceC3=IndiceC3+1;
            C3i(IndiceC3)=i;
            C3j(IndiceC3)=j;
%       CuentaResto=CuentaResto+1;
        end
      end
  end
end
Capacidad=3*(CuentaClaseC2+CuentaClaseC1menorT+CuentaClaseC1mayorigualT);

%Codificación aritmética de RErrort
cuentaRErrort=zeros(2,1);
RErrort=RErrort+1; %Debo aumentarle uno para que sea 1 o 2
for i=1:(3*L*K)
   cuentaRErrort(RErrort(i))=cuentaRErrort(RErrort(i))+1;
end

flag_bitserror=0;% la uso para validar que ninguna frecuencia sea 0
if cuentaRErrort(1)==0 || cuentaRErrort(2)==0
   codigoRErrort=0;
   largocodigoRErrort=1;
else
codigoRErrort=arithenco(RErrort,cuentaRErrort);
largocodigoRErrort=length(codigoRErrort);
end

%strRErrort=sprintf('1- Longitud Errores Guardados = %d. Se deben a la Compresión-
Expansión. Codificados aritméticamente su longitud es= %d',3*L*K, largocodigoRErrort);
%disp(strRErrort);

%Codificación aritmética del Mapa que ubica C1
Lvmap=reshape(LMap,K*L,1);%Longitud mapa vectorizado
secuenciaLvmap=Lvmap+1;%toma valor 1 o 2
cuentaLvmap=zeros(2,1);
 for i=1:L*K
   cuentaLvmap(secuenciaLvmap(i))=cuentaLvmap(secuenciaLvmap(i))+1;
```

```
end

flag_mapa=0;%Valido que ninguna frecuencia en el mapa dé 0
if cuentaLvmap(1)==0 || cuentaLvmap(2)==0
   flag_mapa=1;
   largocodigoLvmap=1;
   codigoLvmap=0;
else
 codigoLvmap=arithenco(secuenciaLvmap,cuentaLvmap);
 largocodigoLvmap=length(codigoLvmap);
end
%strLvmap=sprintf('2- Longitud mapa original = %d Longitud mapa comprimido = %d',L*K,
largocodigoLvmap);
%disp(strLvmap);

%En la extracción se decodificará de la siguiente manera
%decodigo=arithdeco(codigo,cuentaLvmap,L*K);
%decodigof=decodigo-1;
%arithenco en matlab no trabaja con ceros pide enteros finitos positivos
%Por eso sumé 1

%Formateo de los bits perdidos en un vector
C2bitperdidos=reshape([C2V1bitperdido C2V2bitperdido C2V3bitperdido],IndiceC2*3,1);

%Se debe embeber:
%1- El mapa
%2- Los LSBs de v1, v2 y v3 que están en C2
%3- El bitstream R formado por los errores de compansión
%4- La marca (por ejemplo el md5)
%Para la carga total debo considerar los bits extra en la cabecera
%donde se leerán las longitudes de los valores embebidos

%TotalPayload=largocodigoLvmap+largocodigoRErrort+largoWaterPD+largofijocabecera;

%Convierto a binario números decimales que indican:
%las longitudes de los códigos obtenidos
%y la frecuencia necesaria para decodificar
%con esto convierto los datos necesarios en la cabecera
%en una cadena de bits
binsecuencia=dec2bin(largocodigoLvmap,24);
binsecuencia=strcat(binsecuencia,dec2bin(cuentaLvmap(1,1),24));
binsecuencia=strcat(binsecuencia,dec2bin(cuentaLvmap(2,1),24));
binsecuencia=strcat(binsecuencia,dec2bin(length(C2bitperdidos),24));
binsecuencia=strcat(binsecuencia,dec2bin(largocodigoRErrort,24));
binsecuencia=strcat(binsecuencia,dec2bin(cuentaRErrort(1,1),24));
binsecuencia=strcat(binsecuencia,dec2bin(cuentaRErrort(2,1),24));
binsecuencia=strcat(binsecuencia,dec2bin(LargoWaterPD,24));

binsecuencia=strcat(binsecuencia,dec2bin(OrdenF,8));
binsecuencia=strcat(binsecuencia,dec2bin(TotalBloques,8));
%Guardo los valores del Umbral
p=1;
for p=1:TotalBloques
  binsecuencia=strcat(binsecuencia,dec2bin(T(p),10));
end
```

```
largocabecerafija=length(binsecuencia);

BH=zeros(1,largocabecerafija);
for i=1:largocabecerafija
   BH(i)=str2num(binsecuencia(i));
end
B=zeros(1,largocabecerafija+length(codigoLvmap)+length(C2bitperdidos)+length(codigoRErro
rt)+length(WFinal));
B=[BH codigoLvmap' C2bitperdidos' codigoRErrort' WFinal];
flag_bits=0;
largototalmarca=length(B);

%save largototalmarca para pruebas
%disp('Necesito insertar');
%disp(largototalmarca);
%disp('La capacidad es');
%disp(Capacidad);
%Valido que se verifique la capacidad con una bandera
if(Capacidad<largototalmarca)
   flag_bits=1;
end

%Incorporo la cabecera y la marca
C2ind=1;
%Cambio de variable para controlar
V1Qf=V1Q;
V2Qf=V2Q;
V3Qf=V3Q;

%Marcado
i=1;j=1;
kk=1;
while kk<=largototalmarca
   if (LMap(i,j)==1)
       V1Qf(i,j)=2*V1Qf(i,j)+B(kk);%Caso expansión
       kk=kk+1;
       if((kk)>largototalmarca)
          break;
       end
       V2Qf(i,j)=2*V2Qf(i,j)+B(kk);%Caso expansión
       kk=kk+1;
       if((kk)>largototalmarca)
          break
       end
       V3Qf(i,j)=2*V3Qf(i,j)+B(kk);      %Caso expansión
       kk=kk+1;
 end
 %disp(C2i);
 %disp(C2j);
   if LMap(i,j)==0

           C2ind=C2ind+1;
           V1Qf(i,j)=2*floor(V1Qf(i,j)/2)+B(kk);
           kk=kk+1;
```

```
            if(kk>largototalmarca)
              break;
            end
            V2Qf(i,j)=2*floor(V2Qf(i,j)/2)+B(kk);
            kk=kk+1;
            if(kk>largototalmarca)
               break;
            end
            V3Qf(i,j)=2*floor(V3Qf(i,j)/2)+B(kk);
            kk=kk+1;
            if(kk>largototalmarca)
               break;
            end
 end
j=j+1;
 if(j==L+1)
         j=1;
         i=i+1;
         if(i>K)
            break;
         end
end
end

% Funcion inversa
promfun=inline('x(1) + ceil( ( x(2)+x(3)+x(4) )/4) ');
dif1f=inline('x(1)+ceil((x(2)+x(3)+x(4))/4)-x(2)');
dif2f=inline('x(1)+ceil((x(2)+x(3)+x(4))/4)-x(3)');
dif3f=inline('x(1)+ceil((x(2)+x(3)+x(4))/4)-x(4)');
%Trabajo con la traspuesta porque el reshape trabaja por columna
largo=K*L;
%V0ff=V0';
%V1Qff=V1Qf';
%V2Qff=V2Qf';
%V3Qff=V3Qf';
l=0;
for i=1:K
   for j=1:L
      l=l+1;
      V0t(l)=V0(i,j);
      V1t(l)=V1Qf(i,j);
      V2t(l)=V2Qf(i,j);
      V3t(l)=V3Qf(i,j);
   end
end
%Armo matriz total
x=1;
Fila=1;
for z=1:largo
   P(Fila,[x:x+1])=[V0t(z) V1t(z)];
   P(Fila+1,[x:x+1])=[V2t(z) V3t(z)];
   x=x+2;
   if (x>2*L)
      x=1;
      Fila=Fila+2;
```

```
   end
end
U0final=blkproc(double(P),DIREC,promfun);
U1final=blkproc(double(P),DIREC,dif1f);
U2final=blkproc(double(P),DIREC,dif2f);
U3final=blkproc(double(P),DIREC,dif3f);
x=1;
Fila=1;
for z=1:largo
   IM(Fila,[x:x+1])=[U0final(z) U1final(z)];
   IM(Fila+1,[x:x+1])=[U2final(z) U3final(z)];
   x=x+2;
   if (x>2*K)
      x=1;
      Fila=Fila+2;
   end
end
IM=IM';
IMF=uint8(IM);
%O IMF=uint16(IM);

[PSNR_OUT,mse_OUT] = measerr(imgoriginal,IMF, BPP );
if (flag_bits==1)
   fitness=10000;
else
   fitness=mse_OUT;
end

str = sprintf('PSNR = %f,  mse=%f fitness=%f',PSNR_OUT,mse_OUT,fitness);
disp(str);
%Diferencia con función embebido1f- Se graba la imagen
%indicessim=ssim(imgoriginal,IMF,BPP);
%strssim=sprintf('SSIM=%f', indicessim);
%disp(strssim);
%imwrite(IMF,'Marcada.pgm');
%dicomwrite(IMF,'Marcada.dcm');
%disp('------------------------------------------------------------------');
```

Extracción

```
%%gaextraccion1
%Recuperación de la imagen original
%La imagen fue marcada con Tian para Quads

I=imread('lena.pgm'); %Para comparar
%La que correspondiera

BPP=8;%Se puede obtener de la imagen
LMax=2^BPP-1;
IW=imread('Marcada.pgm');
[M N]=size(IW);
%Se deben verificar dimensiones
K=M/2;
L=N/2;
```

```
largo=K*L;

DIREC=[2 2];%según matriz dos por dos . La va a tomar como vector
%Calculo del promedio del quad y de las diferencias
%Funciones generales

avFunc=inline('floor( ( x(1)+x(2)+x(3)+x(4) )/4)');
dif1=inline(' x(1)-x(3)');%Ojo numera de arriba hacia abajo de izquierda a derecha
dif2=inline('x(1)-x(2)');
dif3=inline('x(1)-x(4)');

%Aplicación a la imagen
V0rec=blkproc(double(IW),DIREC,avFunc);
V1rec=blkproc(double(IW),DIREC,dif1);
V2rec=blkproc(double(IW),DIREC,dif2);
V3rec=blkproc(double(IW),DIREC,dif3);

%Clasifico en C3 y no C3.
%Ahora son todos cambiables o no cambiables

%Extracción: las vtilde se obtienen por la ecuación 11
%y deben satisfacer la ecuación 9
V1tilde0=(2*floor(V1rec/2)+0);
V1tilde1=(2*floor(V1rec/2)+1);
V2tilde0=(2*floor(V2rec/2)+0);
V2tilde1=(2*floor(V2rec/2)+1);
V3tilde0=(2*floor(V3rec/2)+0);
V3tilde1=(2*floor(V3rec/2)+1);

U0tilde0=V0rec+ceil((V1tilde0+V2tilde0+V3tilde0)/4);
U0tilde1=V0rec+ceil((V1tilde1+V2tilde1+V3tilde1)/4);

DIFV1tilde0=(U0tilde0-V1tilde0);
DIFV1tilde1=(U0tilde0-V1tilde1);
DIFV2tilde0=(U0tilde0-V2tilde0);
DIFV2tilde1=(U0tilde0-V2tilde1);
DIFV3tilde0=(U0tilde0-V3tilde0);
DIFV3tilde1=(U0tilde0-V3tilde1);
%Preferí el cálculo en bloques

DIFV1tilde01=(U0tilde1-V1tilde0);
DIFV1tilde11=(U0tilde1-V1tilde1);
DIFV2tilde01=(U0tilde1-V2tilde0);
DIFV2tilde11=(U0tilde1-V2tilde1);
DIFV3tilde01=(U0tilde1-V3tilde0);
DIFV3tilde11=(U0tilde1-V3tilde1);

%Clasificamos todo
%Extraemos la marca

ClaseNoEsC3=zeros(K,L);
kke=1;
for i=1:K
   for j=1:L
```

```
    if (U0tilde0(i,j)>=0 && U0tilde0(i,j)<=LMax && U0tilde1(i,j)>0 && U0tilde1(i,j)<=LMax
...
        && DIFV1tilde0(i,j)>=0 && DIFV1tilde0(i,j)<=LMax ...
        &&DIFV1tilde1(i,j)>=0  && DIFV1tilde1(i,j)<=LMax ...
        &&DIFV2tilde0(i,j)>=0 &&DIFV2tilde0(i,j)<=LMax && ...
        DIFV2tilde1(i,j)>=0 && DIFV2tilde1(i,j)<=LMax && ...
        DIFV3tilde0(i,j)>=0 && DIFV3tilde0(i,j)<=LMax ...
        && DIFV3tilde1(i,j)>=0 && DIFV3tilde1(i,j)<=LMax ...
        && DIFV1tilde01(i,j)>=0 && DIFV1tilde01(i,j)<=LMax ...
        &&DIFV1tilde11(i,j)>=0  && DIFV1tilde11(i,j)<=LMax ...
        &&DIFV2tilde01(i,j)>=0 &&DIFV2tilde01(i,j)<=LMax && ...
        DIFV2tilde11(i,j)>=0 && DIFV2tilde11(i,j)<=LMax ...
        && DIFV3tilde01(i,j)>=0 && DIFV3tilde01(i,j)<=LMax ...
        && DIFV3tilde11(i,j)>=0 && DIFV3tilde11(i,j)<=LMax)

         ClaseNoEsC3(i,j)=1; %Será C1 o C2
           %Recupero la marca
           marca(kke)=bitget(abs(V1rec(i,j)),1);
           marca(kke+1)=bitget(abs(V2rec(i,j)),1);
           marca(kke+2)=bitget(abs(V3rec(i,j)),1);
           kke=kke+3;
           %Pongo todos los últimos bits
           %Es más de lo necesario
           %Después me fijo el límite para la marca
    % else
       % ClaseNoEsC3(i,j)=0; %No hace falta porque se inicializó
  end
 end
end

%Parte de los cálculos corresponden a pruebas

marca161=marca(1:24);
largocodigoLvmape=binarioadecimal(marca161);
%Recupero los primeros 16 bits
%que me dan la longitud del mapa comprimido
marca162=marca(25:48);
cuentaLvmape(1,1)=binarioadecimal(marca162);
%Los segundos 16 bits me dan la frecuencia de los datos para el mapa
marca163=marca(49:72);
cuentaLvmape(2,1)=binarioadecimal(marca163);
%Los terceros 16 bits me dan la otra frecuencia necesaria para el mapa

marca164=marca(65:96);
largocodigobitsperdidose=binarioadecimal(marca164);
%Los cuartos 16 bits me dan la longitud comprimida de los bits perdidos
%marca165=marca(97:120);
%cuentabitsperdidose(1,1)=binarioadecimal(marca165);
%Los quintos 16 bits me dan la frecuencia necesaria en los bits perdidos
%marca166=marca(121:144);
%cuentabitsperdidose(2,1)=binarioadecimal(marca166);
%Los sextos 16 bits me dan la segunda frecuencia necesaria
%No se codifica
marca167=marca(97:120);
largocodigoRErrorte=binarioadecimal(marca167);
```

```
%largo de la cadena de errores producidos por la expansión, una vez
%comprimidos
marca168=marca(121:144);
cuentaRErrorte(1,1)=binarioadecimal(marca168);
%Frecuencia necesaria para la recuperación
marca169=marca(145:168);
cuentaRErrorte(2,1)=binarioadecimal(marca169);
%Frecuencia necesaria para la recuperación

marca1610=marca(169:192);
LargoWaterPDe=binarioadecimal(marca1610);
%Finalmente obtengo la longitud de la marca de agua


OrdenFR=binarioadecimal(marca(193:200));
TotalBloquesR=binarioadecimal(marca(201:208));
TamFR=M/OrdenFR;
OrdenCR=TotalBloquesR/OrdenFR;
TamCR=N/OrdenCR;

j=208;
TR=zeros(1,TotalBloquesR);
for p=1:TotalBloquesR
TR(p)=binarioadecimal(marca(j+1:j+10));
j=j+10;
end

p=1;
for i=1:M
    for j=1:N
      if mod(j,TamCR)==1 && j ~= 1
         p=p+1;%cambio al pasar de columna
      end
      MUmbralR(i,j)=TR(p);
    end    %Fin Columna
    %Paso de fila
     if mod(i,TamFR)==0 && i~= 1
        p=p+1;
     else
        p=p-N/TamCR+1;
     end
end

Lvmape=zeros(1,largocodigoLvmape);
for i=1:largocodigoLvmape
   Lvmape(i)=marca(i+208+TotalBloquesR*10);
end
if largocodigoLvmape>1
Lvmape1=arithdeco(Lvmape',cuentaLvmape,L*K);
Lvmape=Lvmape1-1;
LMape=reshape(Lvmape,K,L);
else if cuentaLvmape(1)==0
    LMape=ones(K,L);
   else
    LMape=zeros(K,L);
```

```
   end
end

codigobitsperdidose=zeros(largocodigobitsperdidose,1);
for i=1:largocodigobitsperdidose
   codigobitsperdidose(i)=marca(i+208+TotalBloquesR*10+largocodigoLvmape);
end
lbitsperdidose=largocodigobitsperdidose;
%lbitsperdidose=sum(cuentabitsperdidose);
%if largocodigobitsperdidose>1
%bitsperdidose=arithdeco(codigobitsperdidose,cuentabitsperdidose,sum(cuentabitsperdidose));
%bitsperdidose=bitsperdidose-1;
%else
 %   if cuentabitsperdidose(1)==0
 %   bitsperdidose=zeros(lbitsperdidose);
  %  else
  %      bitsperdidose=ones(lbitsperdidose);
  %  end
%end
%Calculamos los bits perdidos de C2 en caso de codificar aritméticamente

RErrorte=zeros(largocodigoRErrorte,1);
for i=1:largocodigoRErrorte

codigoRErrorte(i)=marca(i+208+TotalBloquesR*10+largocodigoLvmape+largocodigobitsperdi
dose);
end
%Verificado el código
if largocodigoRErrorte>1
   RErrorte=arithdeco(codigoRErrorte',cuentaRErrorte,L*K*3);
   RErrortef=RErrorte-1;
else
   if cuentaRErrorte(1)==0
      RErrortef=zeros(1,L*K*3);
   else
      RTErrortef=ones(1,L*K*3);
   end
end

%Calculamos los valores de error producidos por la compresión
%asociados a C1, siempre que superemos el umbral

WaterPDe=zeros(LargoWaterPDe,1);
for i=1:LargoWaterPDe

WaterPDe(i)=marca(i+208+TotalBloquesR*10+largocodigoLvmape+largocodigobitsperdidose
+largocodigoRErrorte);
end
%Recupero la transpuesta de Wfinal

largototalmarcae=208+TotalBloquesR*10+largocodigoLvmape+largocodigobitsperdidose+larg
ocodigoRErrorte+LargoWaterPDe;
%Debe coincidir con el largo original
%Ahora se deben modificar los valores para volver al original
```

```
%Recuperación matriz original
V0n=V0rec;
V1n=V1rec;
V2n=V2rec;
V3n=V3rec;
kkf=1;
z=1;%Contará los bits de C2 perdidos
pos=1;%Busca la posición
for i=1:K
  for j=1:L
     if (ClaseNoEsC3(i,j)==1 && kkf<=largototalmarcae)
       if (LMape(i,j)==1) %pertenece a C1
          V1n(i,j)=floor(V1n(i,j)/2);
          kkf=kkf+1;
          if(kkf>largototalmarcae)
             break;
          end
          V2n(i,j)=floor(V2n(i,j)/2);
          kkf=kkf+1;
          if(kkf>largototalmarcae)
             break;
          end
          V3n(i,j)=floor(V3n(i,j)/2);
          kkf=kkf+1;
          %Falta considerar lo que perdió C2
        else %No pertenece a C1 pero no es C3: o sea es C2
         if (z<=floor(lbitsperdidose/3))
          V1n(i,j)=2*floor(V1n(i,j)/2)+bitsperdidose(z);
          kkf=kkf+1;
          if(kkf>largototalmarcae)
             break;
          end
          V2n(i,j)=2*floor(V2n(i,j)/2)+bitsperdidose(z+floor(lbitsperdidose/3));
          kkf=kkf+1;
          if(kkf>largototalmarcae)
             break;
          end
          V3n(i,j)=2*floor(V3n(i,j)/2)+bitsperdidose(z+2*floor(lbitsperdidose/3));
          kkf=kkf+1;
          z=z+1;
         end
       end
     end
  end
end

%Es la hora de expandir Todo porque ...
%Todo se comprimió
MUmbralR1=MUmbralR(1:M/2, N/2+1:N);
MUmbralR2=MUmbralR(M/2+1:M,1:N);
MUmbralR3=MUmbralR(M/2+1:M,N/2+1:N);

for j=1:L
   for i=1:K
      if (abs(V1n(i,j))>=MUmbralR1(i,j))
```

```
          V1n(i,j)=sign(V1n(i,j))*(2*abs(V1n(i,j))-MUmbralR1(i,j)+RErrortef((j-1)*L+i));
      end
      if (abs(V2n(i,j))>=MUmbralR2(i,j) )
         V2n(i,j)=sign(V2n(i,j))*(2*abs(V2n(i,j))-MUmbralR2(i,j)+RErrortef((j-1)*L+i+L*K));
      end
      if (abs(V3n(i,j))>=MUmbralR3(i,j)  )
         V3n(i,j)=sign(V3n(i,j))*(2*abs(V3n(i,j))-MUmbralR3(i,j)+RErrortef((j-
1)*L+i+2*L*K));
      end
      %Lo que no es mayor que MUmbralR(i,j) queda igual
   end
end

V0nf=reshape(V0n',largo,1);
V1nf=reshape(V1n',largo,1);
V2nf=reshape(V2n',largo,1);
V3nf=reshape(V3n',largo,1);
%Armo matriz total

x=1;
Fila=1;
for i=1:largo
   Pfn(Fila,[x:x+1])=[V0nf(i) V1nf(i)];
   Pfn(Fila+1,[x:x+1])=[V2nf(i) V3nf(i)];
   x=x+2;
   if (x>2*L)
      x=1;
      Fila=Fila+2;
   end
end


promfun=inline('x(1) + ceil( ( x(2)+x(3)+x(4) )/4) ');
dif1f=inline('x(1)+ceil((x(2)+x(3)+x(4))/4)-x(2)');
dif2f=inline('x(1)+ceil((x(2)+x(3)+x(4))/4)-x(3)');
dif3f=inline('x(1)+ceil((x(2)+x(3)+x(4))/4)-x(4)');

U0finaln=blkproc(double(Pfn),DIREC,promfun);
U1finaln=blkproc(double(Pfn),DIREC,dif1f);
U2finaln=blkproc(double(Pfn),DIREC,dif2f);
U3finaln=blkproc(double(Pfn),DIREC,dif3f);

x=1;
Fila=1;
for i=1:largo
   IFn(Fila,[x:x+1])=[U0finaln(i) U1finaln(i)];
   IFn(Fila+1,[x:x+1])=[U2finaln(i) U3finaln(i)];
   x=x+2;
   if (x>2*L)
      x=1;
      Fila=Fila+2;
   end
end

IRFn=uint8(IFn');
```

```
imshow(IRFn); % Se recupera imagen original
disp('');

disp('Marcado con Marcado Corregido(IW,IRFn)');
[PsnrF,MSEF]=measerr(IW,IRFn) %El marcado con el corregido

disp('Original con Marcado Corregido(I,IRFn)');
[PsnrF2, measerr2,Z2]=measerr(I,IRFn) %El original con el marcado ya corregido
```

Xuan con GA

Principal

```
%% Diseño e implementacion de marcas de agua en imagenes medicas
%%Laura Vargas
%Xuan con GA
clear all;
close all;
clc;

global imgoriginal
global TamF
global TamC
global cant
global Concatenada
global m
global n
global d
global f
global WP
global LargoWaterPD
global LargoScan1
global LargoScan2
global ScanHist1
global ScanHist2
global minimo
global maximo

OrdenF=1;
OrdenC=1;

%% PASO Nº1: leer imagen de entrada
% la matriz de la imagen leida se guarda en una variable "imgoriginal"
imagen = input('Por favor, ingrese el nombre de la imagen a marcar con la extension (ej.
nombre.ext): \n','s');
imgoriginal = imread(imagen);
save imgoriginal %graba para probar

if(length(size(imgoriginal)) == 3)                    %funcion equivalente para saber si la imagen
esta en colores
   imgoriginal = rgb2gray(imgoriginal);               %si entra la convierte en escala de grises
end
%Cuando la matriz tiene alguna dimensión impar es necesario sumar 1 en la dimension
%en m y/o n segun corresponda
[m,n]=size(imgoriginal);
if  rem(m,2)~= 0
   imgoriginal(m+1,:)=imgoriginal(m,:);
end
if  rem(n,2)~= 0
   imgoriginal(:,n+1) = imgoriginal(:,n);
end

[m,n]=size(imgoriginal);%Obtengo los nuevos valores si fueron cambiados
LargoWaterPD=round(0.1*m*n);
```

```
WP=randi([0 1],1,LargoWaterPD);%Se toma una única marca
%save WP

d=m/2;%seguramente el resultado será entero
f=n/2;
%m filas, n columnas

TamF=d/OrdenF;
TamC=f/OrdenC;

TotalBloques=OrdenF*OrdenC;%Total Bloques en cada subbanda
cant=TotalBloques*4;

%Maximo y minimo del histograma original
min_hist = min(min(imgoriginal));
max_hist = max(max(imgoriginal));
str1 = sprintf('Valor mínimo y máximo de intensidad de la imagen: %i, %i\n\n',min_hist,
max_hist);
disp(str1);%Informativo
%imhist(imgoriginal);title('Histograma Original');
opcion_comp = input('Ingrese el número de la opción deseada.\n 1- Comprimir sólo en extremo
izquierdo; \n 2- Comprimir sólo en extremo derecho; \n 3- Comprimir en ambos extremos; \n 4-
No comprimir \n');
%En el caso que no se ingrese un numero valido. Se vuelve a pedir hasta que
%sea una opcion disponible.
while(opcion_comp > 4)
   disp('Ingrese una opción válida.');
   opcion_comp = input('Ingrese el número de la opción deseada.\n 1- Comprimir sólo en
extremo izquierdo; \n 2- Comprimir sólo en extremo derecho; \n 3- Comprimir en ambos
extremos; \n 4- No comprimir \n');
end
if(opcion_comp == 1)
    val_compizq = input('Ingrese la cantidad de valores a comprimir en el extremo izquierdo: ');
    val_compder = 0;
elseif(opcion_comp == 2)
   val_compder = input('Ingrese la cantidad de valores a comprimir en el extremo derecho: ');
   val_compizq = 0;
elseif(opcion_comp == 3)
   val_compizq = input('Ingrese la cantidad de valores a comprimir en el extremo izquierdo: ');
   val_compder = input('Ingrese la cantidad de valores a comprimir en el extremo derecho: ');
elseif(opcion_comp == 4)
   val_compizq = 0;
   val_compder = 0;
end
minimo=val_compizq;
maximo=val_compder;
%CGD=minimo+maximo;
%figure, imshow(imgoriginal),title('Imagen Original');

%% Compresión del histograma según pedido
imgmodificada = imgoriginal; %se copia para solo modificar los valores que hagan falta
                    %el resto queda igual a la original

% Se crean 2 arrays para la secuencia de escaneo
ScanHist1(1)=1;
```

```
ScanHist2(1)=1;

k=1; %variables que sirven para incrementar el indice de los scanhist
l=1;

Horiginal= imhist(imgoriginal);%Histograma original

for i=1:m % para cada fila
   for j=1:n %para cada columna
      if(opcion_comp == 1 || opcion_comp == 3)
         if(imgoriginal(i,j) <= min_hist+val_compizq-1) %0
            imgmodificada(i,j) = imgoriginal(i,j) + val_compizq;
            ScanHist1(k)=0;
            k=k+1;
         elseif(min_hist+val_compizq-1 < imgoriginal(i,j) && imgoriginal(i,j) <
min_hist+val_compizq*2)
            ScanHist1(k)=1;
            k=k+1;
         end
      end

      if(opcion_comp == 2 || opcion_comp == 3)
         if(imgoriginal(i,j) >= max_hist-val_compder+1) %255
            imgmodificada(i,j) = imgoriginal(i,j) - val_compder;
            ScanHist2(l)=0;
            l=l+1;
         elseif(max_hist-val_compder*2 < imgoriginal(i,j) && imgoriginal(i,j) < max_hist-
val_compder+1)
            ScanHist2(l)=1;
            l=l+1;
         end
      end
   end
end

Hmodificado = imhist(imgmodificada);%Histograma modificado
%imhist(imgmodificada),title('Histograma Modificado');
LargoScan1=length(ScanHist1);
LargoScan2=length(ScanHist2);

[PSNR_OUTh,Zh] = measerr(imgmodificada,imgoriginal);
str = sprintf('PSNR original vs imagen con hist. modificado = %f',PSNR_OUTh);
disp(str); %Informa la diferencia entre original e imagen con histograma modificado
%save ScanHist1;
%save ScanHist2;
%figure, imshow(imgoriginal),title('Imagen Original');

%%Transformada wavelet cdf2.2 de la nueva imagen modificada
LS=liftwave('cdf2.2','int2int');%int2intproduce vector de enteros
[CA,CH,CV,CD] = lwt2(double(imgmodificada),LS);
Concatenada =[CA CH; CV CD];
%ParaUmbral=[CH CV CD];
%Tomo la concatenada por facilidad
%Luego no se usará CA
S=zeros(1,cant);
```

```
p=1;
p=1;
for i=1:TamF:m
  for j=1:TamC:n
     S(p)=round(median(median(abs(Concatenada(i:i+TamF-1,j:j+TamC-1)))));
     p=p+1;
  end
end

p=1;
for i=1:m
    for j=1:n
      if mod(j,TamC)==1 && j ~= 1
         p=p+1;%cambio al pasar de columna
      end
      MUmbral(i,j)=S(p);
    end    %Fin Columna
    %Paso de fila
     if mod(i,TamF)==0 && i~= 1
        p=p+1;
     else
        p=p-n/TamC+1;
     end
end

%%UMBRAL y COMPRESIÓN
%Por facilidad tomamos la matriz total
p=1;
for i=1:TamF:m
  for j=1:TamC:n
     Tm(p)=min(min(abs(Concatenada(i:i+TamF-1,j:j+TamC-1))));
     TM(p)=max(max(abs(Concatenada(i:i+TamF-1,j:j+TamC-1))));
     p=p+1;
  end
end

% ******************************************************
%% Configuración Algoritmo Genético
% ******************************************************
options = gaoptimset;

% Población Genética
options = gaoptimset(options,'PopulationSize', 30);
options = gaoptimset(options, 'PopulationType', 'doubleVector');%
options=gaoptimset(options,'InitialPopulation',S);
%En la mayoría de los casos conviene que la semilla sea la mediana de los bloques
%Abajo se indican otras posibilidades
%options = gaoptimset(options, 'InitialPopulation',TM-1);
%options = gaoptimset(options, 'InitialPopulation',round((Tm+TM)/2));
%options= gaoptimset(options, 'InitialPopulation',Tm+1);
options = gaoptimset(options,'PopInitRange',[Tm+1;TM-1]);
options = gaoptimset(options, 'CreationFcn', @gacreationuniform);

% Criterios de parada
options = gaoptimset(options, 'Generations',50);
```

```
%options = gaoptimset(options, 'FitnessLimit', 1.0001);
%options = gaoptimset(options,'TolFun', 3);
options = gaoptimset(options,'StallTimeLimit',3000);
options = gaoptimset(options,'StallGenLimit', 40);

% ****************************************************
%% Operadores Genéticos
% ****************************************************
% Elitismo
options = gaoptimset(options, 'EliteCount', 2);
% Operador de selección
options = gaoptimset(options, 'SelectionFcn', @selectionroulette);
%options= gaoptimset(options, 'SelectionFcn', @selectiontournament);
% Algoritmo de cruce
options = gaoptimset(options, 'CrossoverFcn', @crossoverscattered);
options = gaoptimset(options, 'CrossoverFraction', 0.8);
% Algoritmo de mutación
% Determinamos que no sea posible que arroja valores de umbral fuera de los límites
options = gaoptimset(options, 'MutationFcn', @mutationadaptfeasible);
%options=gaoptimset(options ; 'MutationFcn', @mutationuniform) ;
%Algoritmo de seleccion
%rank
options = gaoptimset(options, 'FitnessScalingFcn', @fitscalingrank);


%% Llamado a función que ejecuta el cálculo del fitness (MSE)
% GA y Embebido Final
[x, fval, reason, output, population, scores]= ga(@gaembebido2,cant,options);
fprintf('El mejor valor hallado fue : %g\n', fval);
fprintf('La cantidad de generaciones para hallarlo fue : %d\n', output.generations);
fprintf('La cantidad de evaluaciones fue : %d\n', output.funccount);

%Embebido Final
disp('Embebido Final: ');
[fitnessf] = gaembebido2(x);
```

Función Embebido

```
%% Diseño e implementacion de marcas de agua en imagenes medicas
%Función embebido según Xuan con GA
%Laura Vargas
function [fitness] = gaembebido2(S)
global imgoriginal
global TamC
global TamF
global cant
global Concatenada
global m
global n
global d
global f
global WP
global LargoWaterPD
global LargoScan1
global LargoScan2
```

```
global ScanHist1
global ScanHist2
global minimo
global maximo

S2=round(abs(S));

%% UMBRAL y COMPRESIÓN
flag_bits=0;
flag_rest=0;
MUmbral=zeros(m,n);
Error=zeros(m,n);
p=1;
for i=1:m
    for j=1:n
      if mod(j,TamC)==1 && j ~= 1
          p=p+1;%cambio al pasar de columna
      end
      MUmbral(i,j)=S2(p);
    end    %Fin Columna
    %Paso de fila
     if mod(i,TamF)==0 && i~= 1
        p=p+1;
     else
        p=p-n/TamC+1;
     end
end
Mcomprimida=Concatenada;
Mexpandida=Concatenada;

for i=1:m
   for j=1:n
  if (i>d ||j>f) %solamente en las subbandas de detalle
          if( abs(Concatenada(i,j)) > MUmbral(i,j))
            Mcomprimida(i,j)=sign(Concatenada(i,j))* ( floor((abs(Concatenada(i,j))-
MUmbral(i,j))/2) + MUmbral(i,j));
            Mexpandida(i,j)=sign(Mcomprimida(i,j)) *(2*abs(Mcomprimida(i,j))-MUmbral(i,j));
            Error(i,j)=  abs(Concatenada(i, j))- abs(Mexpandida(i,j));
            %Ya calculo el error que habrá
            %Debe ser usado en la restauración
            %Solo para eso
          end
      end
   end
end
%save Error
%%Embebido de la marca de agua
% Ademas de incrustar la marca, se agrega la informacion adicional de
% cabecera
%CAe = Error(1:m/2 , 1:n/2);
CHe = Error(1:m/2 , n/2+1:n);
CVe = Error(m/2+1:m , 1:n/2);
CDe = Error(m/2+1:m , n/2+1:n);
%ERROR
%Codificación aritmetica del error
```

```
Capacidad=3*d*f;
MatrizError=[CHe CVe CDe];
save MatrizError;
VError = reshape(MatrizError,1,Capacidad);
%Coloca un vector a continuación del otro por columna
%si no va la %transpuesta
%Se guarda la frecuencia de los 3 valores posibles: -1 0 y 1
%se toman dos : 1 y 0- Luego se trabajarás con los signos
FrecError = zeros(1,2);
VErrorF = VError+1; %Se debe aumentar 1 para tener valores entre 1 y 2
%Esto se hace antes de codificar
for i=1:(3*d*f)
    FrecError(VErrorF(i)) = FrecError(VErrorF(i))+1;
end
%disp('F1');
%disp(FrecError(1));
%disp('F2');
%disp(FrecError(2));
%save FrecError;
if (FrecError(1)==0 || FrecError(2)==0)
   VErrorCod=0;
     else
    %Se realiza la codificación aritmética
   VErrorCod = arithenco(VErrorF, FrecError);
   %longitud de los errores codificados
end
BLongErrorCod = length(VErrorCod);
%strRError = sprintf('Longitud Errores Guardados = %d debidos a la Expansión\nCodificados
aritméticamente su longitud es= %d.',3*d*f, BLongErrorCod);
%disp(strRError);
 %Datos a guardar
 %concatenación de los datos
%save BLongErrorCod
%save VErrorCod %Es el resultado de la codificación o 0
%Se debe guardar la propia longitud de la cabecera
%Se debe almacenar la longitud del error codificado aritméticamente
%Se deben guardar las frecuencias de los errores para decodificar
binsecuencia=dec2bin(BLongErrorCod,24);
binsecuencia=strcat(binsecuencia,dec2bin(FrecError(1,1),24));
binsecuencia=strcat(binsecuencia,dec2bin(FrecError(1,2),24));
binsecuencia=strcat(binsecuencia,dec2bin(TamF,10));
binsecuencia=strcat(binsecuencia,dec2bin(TamC,10));
binsecuencia=strcat(binsecuencia,dec2bin(cant,8));
for p=1:cant
   binsecuencia=strcat(binsecuencia,dec2bin(S2(p),16));
end
save S2
%Se deben guardar las modificaciones al histograma
binsecuencia=strcat(binsecuencia,dec2bin(LargoScan1,16));%
binsecuencia=strcat(binsecuencia,dec2bin(LargoScan2,16));%
binsecuencia=strcat(binsecuencia,dec2bin(minimo,16));%
binsecuencia=strcat(binsecuencia,dec2bin(maximo,16));%
binsecuencia=strcat(binsecuencia,dec2bin(LargoWaterPD,24));%
LargoCabecera=length(binsecuencia)+24;%voy a incluir largo total de la marca
LargoMarcaTotal=LargoCabecera+BLongErrorCod+LargoWaterPD+LargoScan1+LargoScan2;
```

```
binsecuencia=strcat(binsecuencia,dec2bin(LargoMarcaTotal,24));
%BLongErrorCod+LargoWat2erPD+LargoScan1+LargoScan2 son los bits de info a
%embeber
if (Capacidad <= LargoMarcaTotal)
          flag_bits=1;
         % disp('Los bits a embeber deben ser menos que los bits disponibles');
end

BH=zeros(1,LargoCabecera);
for i=1:LargoCabecera
   BH(i)=str2double(binsecuencia(i));%Es el header
end
%Bfinal es lo que se debe embeber
BFinal=zeros(1,LargoMarcaTotal);
BFinal=[BH VErrorCod WP ScanHist1 ScanHist2];
%save BFinal
%save LargoMarcaTotal
%%Marcado
Mmarcada=Mcomprimida;
p=1;
for i=1:m % para cada fila
   for j=1:n %para cada columna
      if(i > m/2 || j > n/2)
        if(p<=LargoMarcaTotal)
          Mmarcada(i,j) =2 * (Mcomprimida(i,j)) + BFinal(p);
          p=p+1;
        end
       end
      end
end
%Divide la matriz_marcada para recuperar los cuadrantes
CAF = Mmarcada(1:m/2 , 1:n/2);
CHF = Mmarcada(1:m/2 , n/2+1:n);
CVF = Mmarcada(m/2+1:m , 1:n/2);
CDF = Mmarcada(m/2+1:m , n/2+1:n);
%save Mmarcada
%% Transformada inversa, para obtener imagen con marca
LS = liftwave('cdf2.2','Int2Int');
MmarcadaE = ilwt2(CAF,CHF,CVF,CDF,LS);
%figure, imshow(ImMarcada),title('Marcada');
imwrite(uint8(MmarcadaE),'Imagen_Marcada.pgm');
%save MmarcadaE
MRecup=imread('Imagen_Marcada.pgm');
[CAp,CHp,CVp,CDp] = lwt2(double(MRecup),LS);
Concatenadap=[CAp CHp;CVp CDp];
%save Concatenadap
tprueba=isequal(Concatenadap,Mmarcada);
if(tprueba==0)
   flag_rest=1;
end
%Los valores típicos que adopta este parámetro están entre 30 y 50 dB,
[PSNR_OUT,mse_OUT] = measerr(imgoriginal,uint8(MmarcadaE));%o uint16
str = sprintf('Entre la imagen original y la marcada, PSNR = %f,
mse=%f',PSNR_OUT,mse_OUT);
disp(str);
```

```
if(flag_bits == 1 )
   fitness = 10000;
   disp('Capacidad insuficiente');
elseif flag_rest==1
   fitness=10000;
   disp('No se recupera');
   else
   fitness = mse_OUT;
end
disp('fitness');disp(fitness);
end
```

Extracción

```
%% Diseño e implementacion de marcas de agua en imagenes medicas
%Recuperación imagen original Xuan con GA
%Laura Vargas

%% Lectura imagen marcada
% la matriz de la imagen leida se guarda en una variable "imrecuperada"
Irecuperada= imread('Imagen_Marcada.pgm');              %abre imagen
[m n]=size(Irecuperada);
d=m/2;f=n/2;
if(length(size(Irecuperada)) == 3)                  %funcion equivalente para saber si la imagen
esta en colores
   Irecuperada = rgb2gray(Irecuperada);             %si entra la convierte en escala de grises
end
%% Tranformada
LSR = liftwave('cdf2.2','Int2Int');
[CAR,CHR,CVR,CDR] = lwt2(double(Irecuperada),LSR);
ConcatenadaR=[CAR CHR;CVR CDR];

%% Extraccion de la marca
LSBS=zeros(m,n);
for i=1:m % para cada fila
   for j=1:n %para cada columna
      if (i > d || j > f)
          LSBS(i,j)=mod(abs(ConcatenadaR(i,j)),2);
      end
   end
end

%LSBS Se separan en bloques
LSBS3 = LSBS(m/2+1:m , 1:n/2);
LSBS4 = LSBS(m/2+1:m , n/2+1:n);
LSBS34 = [LSBS3 LSBS4];

LSBS2_R = reshape(LSBS(1:m/2 , n/2+1:n)',1,m/2*n/2); %2do cuadrante del LSBS
LSBS34_R = reshape(LSBS34',1,m/2*n);

BFR = [LSBS2_R LSBS34_R]; %incluye ceros del final

BLongErrorCodR=binarioadecimal(BFR(1:24));
F1=binarioadecimal(BFR(25:48));
%F2=3*d*f-F1;
```

```
%disp(F1);disp(F2);
F2=binarioadecimal(BFR(49:72));
TamFR=binarioadecimal(BFR(73:82));
TamCR=binarioadecimal(BFR(84:92));
CantR=binarioadecimal(BFR(93:100));%Cantidad de bloques
j=100;
for p=1:CantR
TR(p)=binarioadecimal(BFR(j+1:j+16));
j=j+16;
end

%Para matrices cuadradas
MUmbralR=zeros(m,n);
Error=zeros(m,n);

p=1;
for i=1:m
    for j=1:n
      if mod(j,TamCR)==1 && j ~= 1
         p=p+1;%cambio al pasar de columna
      end
      MUmbralR(i,j)=TR(p);
    end    %Fin Columna
    %Paso de fila
     if mod(i,TamFR)==0 && i~= 1
        p=p+1;
     else
        p=p-n/TamCR+1;1;
     end
end


LargoScan1R=binarioadecimal(BFR(100+16*CantR+1:100+16*CantR+16));
LargoScan2R=binarioadecimal(BFR(100+16*CantR+16+1:100+16*CantR+16+16));
minimoR=binarioadecimal(BFR(100+16*CantR+16+16+1:100+16*CantR+16+16+16));
maximoR=binarioadecimal(BFR(100+16*CantR+48+1:100+16*CantR+48+16));
BWPR=BFR(100+16*CantR+64+1:100+16*CantR+64+24);
LargoWaterPDR=binarioadecimal(BWPR);
%Se recupera el largo de la Marca

BMTR=BFR(100+16*CantR+88+1:100+16*CantR+88+24);
LargoMarcaTotalR=binarioadecimal(BMTR);
%Recupera largo de la Marca Total

BFinalR=BFR(1:LargoMarcaTotalR);
%Recuperación de la matriz de Errores
%Decodificaciòn aritmética
VErrorCodR=BFR(100+16*CantR+112+1:100+16*CantR+112+BLongErrorCodR);
%Vector que coincide con VErrorCod

FrecuenciaR=[F1 F2];
if F1==0
MatrizErrorR=ones(d,f*3);
elseif F2==0;
   MatrizErrorR=zeros(d,f*3);
```

```
else
VErrorFR = arithdeco(VErrorCodR,FrecuenciaR,d*f*3);
%Ya fue decodificada y recuperó el tamaño original
VErrorFFR = VErrorFR-1;
MatrizErrorR=reshape(VErrorFFR',d,f*3);%La total recuperada por columna
end
%Esta matriz es igual a la original
MatrizErrorRF1=zeros(d,f);%Queda con ceros porque no se tocó
MatrizErrorRF2=MatrizErrorR(:,1:f);
MatrizErrorRF3 =MatrizErrorR(:,f+1:2*f);
MatrizErrorRF4=MatrizErrorR(:,(2*f)+1:3*f);

MatrizErrorRF=[MatrizErrorRF1 MatrizErrorRF2;
   MatrizErrorRF3 MatrizErrorRF4];
%Debe compararse con A= [CAe CHe;CVe CDe]- Se hizo
%Finaliza la recuperación del Error

WPR=BFR(100+16*CantR+112+BLongErrorCodR+1:100+16*CantR+112+BLongErrorCodR
+LargoWaterPDR);
%Me da la marca

Scanhist1R=BFR(100+16*CantR+112+BLongErrorCodR+LargoWaterPDR+1:100+16*CantR
+112+BLongErrorCodR+LargoWaterPDR+LargoScan1R);
%Recuperé el vector para el histograma izquierdo
Scanhist2R=BFR(100+16*CantR+112+BLongErrorCodR+LargoWaterPDR+LargoScan1R+1:1
00+16*CantR+112+BLongErrorCodR+LargoWaterPDR+LargoScan1R+LargoScan2R);
%Recuperé el vector para el histograma derecho

% Recuperamos la marca ppiamente dicha
%Expansión y corrección del error
RecR=ConcatenadaR;
MExpandidaR=ConcatenadaR;
pr=1;
for i=1:m
   for j=1:n
      if (i > m/2 || j > n/2)
       if (pr<=LargoMarcaTotalR)
          RecR(i,j)=(ConcatenadaR(i,j)-LSBS(i,j))/2;
          MExpandidaR(i,j)=RecR(i,j);
          pr=pr+1;
       end
       if (abs(MExpandidaR(i,j))>=MUmbralR(i,j))
          MExpandidaR(i,j)=sign(MExpandidaR(i,j))* (2*abs(MExpandidaR(i,j))-
MUmbralR(i,j));
       end
      end
   end
end
CARC = RecR(1:m/2 , 1:n/2);
CHRC = RecR(1:m/2 , n/2+1:n);
CVRC = RecR(m/2+1:m , 1:n/2);
CDRC = RecR(m/2+1:m , n/2+1:n);

MexpandidaRF=MExpandidaR;
MExpandidaRF=sign(MExpandidaR).*(abs(MExpandidaR)+MatrizErrorRF);
```

```
CARF = MExpandidaRF(1:m/2 , 1:n/2);
CHRF = MExpandidaRF(1:m/2 , n/2+1:n);
CVRF = MExpandidaRF(m/2+1:m , 1:n/2);
CDRF = MExpandidaRF(m/2+1:m , n/2+1:n);

%% PASO Nº
% DESCRIPTIVE TEXT
IRecuperada = ilwt2(CARF,CHRF,CVRF,CDRF,LSR);
IRecuperadaF=uint8(IRecuperada);
%figure,imshow(IRecuperadaF),title('Final con histograma cambiado');

%% PASO Nº
%Recupera histograma
s=1;
p=1;
%
minhistIR=min(min(IRecuperadaF));
maxhistIR=max(max(IRecuperadaF));
Ifinal=IRecuperadaF;
 for i=1:m % para cada fila
    for j=1:n %para cada columna
        if(minhistIR<= IRecuperadaF(i,j) && IRecuperadaF(i,j) <= minhistIR+minimoR-1) %1
            if(s<=LargoScan1R)
               if(Scanhist1R(s)== 0)
                  Ifinal(i,j) = IRecuperadaF(i,j) - minimoR;
               % else
                %    Ifinal(i,j) = IRecuperadaF(i,j);
                end
            end
            s=s+1;
         end

        if(maxhistIR>=IRecuperadaF(i,j) && IRecuperadaF(i,j) >= maxhistIR-maximoR+1) %1
            if(p<=LargoScan2R)
               if(Scanhist2R(p)== 0)
                  Ifinal(i,j) = IRecuperadaF(i,j) + maximoR;
               %else
                %   Ifinal(i,j) = IRecuperadaF(i,j);
               end
            end
            p=p+1;
         end
      end
 end
%
% hist_rec = imhist(Ifinal);
load imgoriginal
[PSNR_OUTFF,MSEF] = measerr(imgoriginal,Ifinal);
str = sprintf('ImagenOriginal versus Recuperada PSNR = %f MSE=%f',PSNR_OUTFF,MSEF);
disp(str);
```

Listado de Publicaciones, Presentaciones a Congresos y Jornadas

- Vargas. *"Watermarking based on difference Expansion and genetic Algorithm"*. Accepted for Oral Presentation and Publication in Early Bird Round of "Second International Conference on Advances in Computing, Control and Networking - ACCN 2015", Bangkok, Thailand, 2015.
- Vargas, Vera. *"Marcado de agua reversible en imágenes fijas optimizado mediante algoritmos genéticos"*. Asociación Argentina de Matemática Aplicada. *Aceptado para publicación*, 2014.
- Vargas, Vera. *"Un algoritmo reversible para imágenes fijas basado en Run-Length"*, ISSN 2314-3282. Asociación Argentina de Matemática Aplicada, Vol. 4, pp. 573-578, 2013. Disponible en http://asamaci.org.ar/wp-content/uploads/2013/06/MACI-Vol42013-web.pdf.
- Vargas, Vera. *"An implementation of reversible watermarking for still images"* ISSN 1548-0992 - IEEE LATIN AMERICA TRANSACTIONS, Vol.11, Tomo 1, pp.54-59, 2013. Disponible en http://www.ewh.ieee.org/reg/9/etrans/ieee/issues/vol11/vol11issue1Feb.2013/11TLA1_09Vargas.pdf.

- **Disertante y coautora** (Vargas, Vera), *"Marcado de agua reversible en imágenes fijas optimizado mediante el uso de algoritmos genéticos"*, MACI 2015, Tandil, 2015.
- **Disertante** y **coautora** (Vargas, Vera), "*Un algoritmo reversible para imágenes fijas basado en Run-Length***",** MACI 2013, CABA, 2013.
- **Disertante** y **coautora** (Vargas, Vera), *"An implementation of reversible watermarking for still images"***,** Argencon 2012, Córdoba.
- **Expositora y coautora** (Vargas, Vera)**,** *"Seguridad en Imágenes: marcas múltiples*", Congreso Mundial de Ingeniería, CABA, octubre 2010.
- **Expoxitora y coautora** (Vargas, Vera), "Seguridad en Imágenes", CNIT 2009, Córdoba, 2009.
- **Expositora y coautora** (Vargas, Vera)**,** "*Una implementación de marcas de agua múltiples utilizando enmascaramiento visual"*, Congreso Internacional de Matemática Aplicada, INMAT 2008, agosto 2008 en la Facultad Ingeniería, UBA.

- **Conferencista** (Vargas), "*Seguridad en Transmisión de Imágenes*", en las Jornadas de Informática de UNDEC, Chilecito, 28 de septiembre de 2011.

- **Comunicación** (Vargas), “Seguridad en Transmisión de Imágenes”, Encuentro Regional de Procesamiento de Imágenes, Facultad de Matemática, Astronomía y Física, UNC, CIEM (Centro de Investigación y Estudios de Matemática de Córdoba), 22 de septiembre 2011.