INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

COMPRESIÓN DE IMÁGENES MEDIANTE MEMORIAS ASOCIATIVAS

T E S I S

QUE PARA OBTENER EL GRADO DE

DOCTOR EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA

MC. ENRIQUE GUZMÁN RAMÍREZ

DIRECTOR: DR. OLEKSIY POGREBNYAK

MÉXICO, D.F. JUNIO, 2008

.

Agradecimientos

Al concluir una etapa más de mi vida profesional, quiero agradecer al **Instituto Politécnico Nacional**, especialmente al **Centro de Investigación en Computación**, por la formación académica que me ha brindado.

Quiero expresar mi más sincero agradecimiento a mi director de tesis, el **Dr. Oleksiy Pogrebnyak** por su generosidad al brindarme la oportunidad de recurrir a su capacidad y experiencia científica en un marco de confianza, afecto y amistad, fundamentales para la concreción de este trabajo. Al **Dr. Cornelio Yánez Márquez** por sus valiosas sugerencias, acertados aportes, apoyo incondicional y valiosa amistad ofrecidos durante el desarrollo de este trabajo. Al **Dr. Edgardo Manuel Felipe Riverón**, **Dr. Sergio Suárez Guerra**, **Dr. Volodymyr Ponomaryov**, **Dr. Ricardo Barrón Fernández** y al **Dr. Luis Pastor Sánchez Fernández** por el tiempo invertido en la revisión, corrección y atinados comentarios de este escrito.

Dedicatoria

Este trabajo está dedicado a quienes sin escatimar esfuerzo alguno, sacrificaron gran parte de su vida para formarme y educarme. A quienes la ilusión de su vida ha sido convertirme en persona de provecho. A quienes nunca podré pagar por todo lo que me entregaron ni aún con las riquezas más grandes del mundo. A **Rebeca** y **Rafael**, mis Padres.

Especialmente, quiero dedicar este trabajo a quien ha sabido apoyarme en todo momento. A quien me ha colmado de comprensión, aliento y estímulo, mismos que posibilitaron la conquista de esta meta. Como un testimonio de gratitud ilimitada y por que este logro también es de ella, a **Carmen**, mi esposa, porque desde que unimos nuestras vidas su presencia ha sido y será la inspiración más grande que impulsa cada etapa de mi vida.

A quienes, por la elaboración de este trabajo, he dejado de dedicarles tiempo y han sabido asimilarlo. A quienes son el motivo que me impulsa a seguir superándome. Por que sin ellos nada tendría sentido, a **Daniel Enrique** y **Luís Eduardo**, mis hijos.

A quienes siempre me han colmado de un cariño sincero que sólo una hermana es capaz de dar. A **Pilar**, **Ivonne** y **Diana**.

Resumen

En este trabajo de tesis se propone un nuevo modelo de transformada con base en las Memorias Asociativas Morfológicas (MAM, *Morphological Associative Memories*), llamaremos a este modelo *Transformada Morfológica* (**TM**) y será aplicada en la etapa de transformación de un compresor de imágenes remplazando a métodos tradicionales como la Transformada Discreta del Coseno o la Transformada Discreta *Wavelet*. El algoritmo de la **TM** hace uso de las MAM heteroasociativas; el proceso de aprendizaje se desarrolla entre vectores de entrada predefinidos, llamados *matriz de transformación* y sub-bloques de la imagen, los cuales representan a los vectores de salida. La matriz de transformación es definida en ambos procesos, compresión y descompresión, de esta forma se puede realizar el proceso inverso de la **TM**. Al aplicar el algoritmo de la **TM**, dependiendo de la matriz de transformación usada, la imagen toma una representación morfológica y las etapas que siguen a la transformación en un compresor de imágenes pueden tomar ventaja de esta nueva representación para desarrollar la compresión de los datos. Además, se propone un esquema de compresión de imágenes en cuyo bloque de transformación se hace uso de la **TM**. Los resultados experimentales mostraron que en comparación con los métodos tradicionales, la principal ventaja que una **TM** ofrece es la velocidad de procesamiento y bajos requerimientos de memoria, demostrando además una alta competitividad en los parámetros de compresión y relación señal a ruido.

Índice

Índice de figuras

Índice de tablas

Lista de acrónimos

BAM	Memoria asociativa bidireccional (*Bidirectional Associative Memories*)
BEP	Algoritmo de propagación inversa de error (*Backward Error Propagation*)
bpp	Bits por píxel
CCITT	Comité Consultivo Internacional Telegráfico y Telefónico (*Consultative Committee for International Telegraphy and Telephony*)
CODEC	Codificador-Decodificador
EBCOT	Codificación de bloques con truncamiento óptimo (*Embedded Block Coding with Optimal Truncation*)
ECP	Error cuadrático ponderado
EZW	Codificación progresiva mediante *zerotree* de coeficientes *wavelet* (*Embedded Zerotree Wavelet*)
DCT	Transformada discreta del coseno (*Discrete Cosine Transform*)
DC	Coeficiente de frecuencia cero o coeficiente continuos de una DCT (*Direct Current*)
DFT	Transformada discreta de Fourier (*Discrete Fourier Transform*)
DM	Distorsión máxima
DWT	Transformada discreta *wavelet* (*Discrete Wavelet Transform*)
FFT	Transformada rápida de Fourier (*Fast Fourier Transform*)

FNN	Red neuronal con propagación hacia delante (*Feedforward Neural Network*)
FWT	Transformada rápida *wavelet* (*Fast Wavelet Transform*)
GHA	Algoritmo de aprendizaje Hebbiano generalizado (*Generalized Hebbian Learning Algorithm*)
HVS	Sistema visual humano (*Human Visual System*)
IDCT	DCT inversa (*Inverse Discrete Cosine Transform*)
ISO	Organización Internacional para la Estandarización (*International Organization for Standardization*)
ITU	Unión Internacional de Telecomunicaciones (*International Telecommunication Union*)
JPEG	Grupo Conjunto de Expertos Fotográfico (*Joint Photographic Experts Group*)
LBG	Algoritmo Linde-Buzo-Gray
MAE	Error absoluto medio (*Mean Absolute Error*)
MAM	Memorias asociativas morfológicas (*Morphological Associative Memories*)
MD	Pendiente máxima (*Maximum Descent*)
MEZW	EZW modificado (*Modified* EZW)
MMAA	Memorias morfológicas autoasociativas
MMH	Memorias morfológicas heteroasociativas
MNN	Redes neuronales morfológicas (*Morphological Neural Networks*)
MPEG	Grupo de Expertos de Imágenes en Movimiento (*Motion Pictures Experts Group*)
MSE	Error cuadrático medio (*Mean Square Error*)
mt	Matriz de transformación
NNIC	Programa para la compresión de imágenes con base en redes neuronales (*Neural Network Image Compressor*)
OE	Operaciones elementales
PNN	Algoritmo del vecino más cercanos por parejas (*Pairwise Nearest Neighbor*)
PPM	Predicción por coincidencia parcial (*Prediction by Partial Matching*)
PSNR	Relación señal a ruido (*Peak Signal to Noise Ratio*)
RLE	Codificación por longitud de series o codificación de la carrera (*Run Length Encoding*).
RNA	Redes Neuronales Artificiales
SA	Algoritmo del recocido simulado (*Simulated Annealing*)
sb	Sub-bloque de imagen
SOM	Mapas Auto-Organizados (*Self-Organizing Map*)

SPIHT	División del conjunto en árboles jerárquicos (*Set Partitioning in Hierarchical Trees*)
SQ	Cuantificación escalar (*Scalar Quantization*)
TM	Transformada Morfológica
TMI	Transformada Morfológica Inversa
VQ	Cuantificación vectorial (*Vector Quantization*)
vi	Vector de una imagen
vt	Vector de transformación

Resumen

La compresión de imágenes es una rama de las ciencias fundamental en áreas como la multimedia, las comunicaciones móviles, el procesamiento de video, etc. Un sistema de compresión de imágenes busca codificar los datos de una imagen en una forma compacta, reduciendo el número de bits usados en su representación y minimizando la distorsión causada por el proceso de compresión. Un compresor de imágenes está formado por tres bloques principales: un *transformador de código*, un *cuantificador* y un *codificador*. En el diseño de estos bloques se deben buscar las siguientes características esenciales: elevar la relación de compresión, minimizar la distorsión que genera un proceso de compresión sobre la imagen recuperada y una alta velocidad de procesamiento con la menor cantidad posible de recursos requeridos por el sistema durante su operación.

En este trabajo de tesis se propone un nuevo modelo de transformada con base en las Memorias Asociativas Morfológicas (MAM, *Morphological Associative Memories*), llamaremos a este modelo *Transformada Morfológica* (**TM**) y será aplicada en la etapa de transformación de un compresor de imágenes remplazando a métodos tradicionales como la Transformada Discreta del Coseno o la Transformada Discreta *Wavelet*. El algoritmo de la **TM** hace uso de las MAM heteroasociativas; el proceso de aprendizaje se desarrolla entre vectores de entrada predefinidos, llamados *matriz de transformación* y sub-bloques de la imagen, los cuales representan a los vectores de salida. La matriz de transformación es definida en ambos procesos, compresión y descompresión, de esta forma se puede realizar el proceso inverso de la **TM**. Al aplicar el algoritmo de la **TM**, dependiendo de la matriz de transformación usada, la imagen toma una representación morfológica y las etapas que siguen a la transformación en un compresor de imágenes pueden tomar ventaja de esta nueva representación para desarrollar la compresión de los datos.

Finalmente, se propone un esquema de compresión de imágenes en cuyo bloque de transformación se hace uso de la **TM**. Los resultados experimentales mostraron que en comparación con los métodos tradicionales, la principal ventaja que una **TM** ofrece es la velocidad de procesamiento y bajos requerimientos de memoria, demostrando además una alta competitividad en los parámetros de compresión y relación señal a ruido.

Abstract

Image compression is one of the fundamental scientific branches in areas such as multimedia, mobile communications, video processing, etc. A system for image compression looking for data codifications of an image in a compact form reducing the number of bits used for its representations and minimizing the distortions cased by the compression process. An image compressor is formed by three principal blocks: a code transformer, quantifier and codifier. The design of these blocks should search for the following essential characteristics: increase the compression ratio, minimize the distortion generated by compression process on the restored image and a high processing speed with the possible minimal quantity of the resources required by system during its operations.

In this thesis a new model of transform based on the Morphologic Associative Memories (MAM) called Morphologic Transform (**TM**) is proposed. **TM** is applied at the transformation stage of an image compressor replacing the traditional methods such as Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT). The **TM** algorithm uses heteroassociative MAM. The learning process is developed among a predefined input vectors called transformation matrix and image sub-blocks that represent the output vectors. The transformation matrix is defined in processes, compression and decompression. This way an inverse **TM** process can be realized. Applying **TM** algorithm, the image take a morphologic representation depending on the used transformation matrix. Next compression stages after the transformations can use the advantages of this new representation to develop data compression.

Finally, an image compression scheme with **TM** at the transformation stage is proposed. The experimental results show that in comparison to the traditional methods the principal advantage of **TM** is high processing speed and low memory requirements; moreover, a high competitiveness of the compression rate and signal to noise ratio was obtained.

Capítulo 1

Introducción

1.1 Contexto

El propósito de un sistema de compresión de imágenes es codificar los datos en una forma compacta reduciendo el número de bits usados en su representación y minimizando la distorsión causada por la compresión. Un compresor de imagen está formado por tres bloques principales: un *transformador de código*, un *cuantificador* y un *codificador* [4]; cada uno de estos bloques tiene por función reducir o eliminar las redundancias que pueden aparecer en una imagen.

El creciente uso y demanda que las tecnologías de multimedia y comunicaciones móviles han experimentado en los últimos años, y seguirá en aumento en las décadas venideras, han originado que las técnicas de compresión de imágenes y video jueguen un papel primordial dentro de una nueva área en el mundo de las telecomunicaciones llamada Comunicaciones Multimedia Móviles (*Mobile Multimedia Comunications*).

En el año 2005 los servicios de multimedia móvil tuvieron aproximadamente 32 millones de usuarios tan sólo en el oeste de Europa, lo cual representa un mercado de 24 billones de euros y 3800 millones de Mbytes de tráfico por año. Para el año 2010 se prevé que el crecimiento de usuarios de multimedia móvil será de 1.7 billones [4], [90].

En este trabajo de investigación se estudian las técnicas más importantes en el área de compresión de imágenes con la finalidad de proponer y desarrollar nuevos algoritmos que proporcionen mejores resultados en la relación de compresión, la relación señal a ruido y la complejidad computacional con respecto a los algoritmos actuales.

A mediados de la década de los 80's, el CCITT (*Consultative Committee for International Telegraphy and Telephony*) y el ISO (*International Organization for Standardization*) empezaron a trabajar en conjunto en el desarrollo de un estándar internacional para la compresión de imágenes, pero no fue sino hasta el año de 1992 que se logra el primer estándar internacional para este propósito bajo el acrónimo JPEG (*Joint Photographic Experts Group*) [92]. El nombre formal del estándar JPEG es ISO/IEC IS 10918-1 | ITU-T Recommendation T.81, tal como fue publicado por ISO y CCITT, ahora llamado ITU-T (*International Telecommunication Union*) [37].

El estándar JPEG describe una familia de técnicas de compresión de imágenes fijas de tonalidad continua en escala de grises o color (24 bits). Sin embargo, numerosas aplicaciones han usado la técnica también para compresión de video, porque proporciona descompresión de imagen de calidad bastante alta a una razón de compresión muy buena, y requiere menos poder de cálculo que la compresión MPEG (*Motion Pictures Experts Group*).

Debido a la cantidad de datos involucrada y la redundancia psicovisual en las imágenes, JPEG emplea un esquema de compresión con pérdidas basado en la codificación por transformación. El estándar resultante tiene tantas alternativas como sean necesarias para servir a una amplia variedad de propósitos. El estándar JPEG define tres sistemas diferentes de codificación: 1. Un sistema de codificación básico, con pérdidas, que se basa en la Transformada Discreta del Coseno (DCT, *Discrete Cosine Transform*) y es apropiado para la mayoría de las aplicaciones de compresión. 2. Un sistema de codificación extendida, para aplicaciones de mayor compresión, mayor precisión, o de reconstrucción progresiva. 3. Un sistema de codificación independiente sin pérdidas, para la compresión reversible [92].

En el año de 1997 surge el interés de desarrollar nuevas contribuciones al estándar JPEG con la finalidad de mejorarlo; este nuevo estándar fue nombrado JPEG2000 [39] y es referenciado como ISO/IEC JTC 1/SC 29/ WG 1 N1646R [38].

El estándar JPEG2000 se basa en un esquema de codificación por transformación, pero a diferencia del JPEG está fundamentalmente basado en la Transformada Discreta *Wavelet* (DWT, *Discrete Wavelet Transform*) y en el algoritmo *Embedded Block Coding with Optimal Truncation* (EBCOT) [88]. La DWT es aplicada a una imagen, entonces los coeficientes transformados son cuantificados y codificados para formar una secuencia de información comprimida [17].

Actualmente siguen surgiendo nuevas variantes en los esquemas de compresión, muestra de ello es el trabajo presentado por el investigador C. M. Sousa *et al.* [85], en el cual proponen un sistema de compresión de imágenes basado en el concepto *"Efficient Coding"*, concepto derivado de modelos neuronales para el procesamiento de la información. Este sistema esta formado por dos fases: aprendizaje y proyección. La fase de aprendizaje hace uso de un análisis independiente de componentes (ICA, *Independent Component Analisis*) para formar las funciones base a partir de una clase de datos específica. La fase de proyección consiste en encontrar la representación de la imagen para un subespacio (formado por las funciones base) que reduzca la dependencia estadística entre los coeficientes de proyección utilizados en la reconstrucción de la imagen.

En [62] A. Ouani *et al.* proponen un nuevo algoritmo para la compresión de imágenes denominado *Modified* EZW (*Embedded Zerotree Wavelet*); como su nombre lo indica se trata de una modificación al algoritmo de Shapiro, el EZW [81]. Una de las modificaciones hechas al algoritmo de Shapiro es el incremento de cuatro a seis símbolos en la etapa de prueba de significancía, los dos símbolos nuevos son usados cuando se determina que todos los descendientes de un coeficiente son insignificantes. Otra importante diferencia entre el EZW y el MEZW es la optimización de la codificación mediante el agrupamiento de elementos; debido a que existen seis símbolos, éstos son representados usando tres bits; el algoritmo MEZW lleva a cabo una agrupación de símbolos con la finalidad de poder representarlos con nueve bits, lo que se hace antes del proceso de codificación aritmética generando una optimización en este proceso. El algoritmo MEZW produce resultados que son significativamente mejores que los parámetros de relación a ruido y relación de compresión obtenidos por el algoritmo de Shapiro, sin afectar significativamente el tiempo de procesamiento.

S. Minasyan *et al.* en [56] proponen un método de representación unificada de un amplio número de algoritmos rápidos usados por las transformadas discretas ortogonales, obteniendo una clase de transformada la cual puede definir diferentes familias de transformadas dentro de ella. Este enfoque permite describir muchos algoritmos rápidos de transformadas en una forma unificada. En particular, las familias de transformadas Haar, Hadamard y Slant son definidas basadas en este enfoque. Los autores también proponen dos esquemas de compresión de imágenes utilizando este método de transformación; los experimentos realizados con estos esquemas muestran desempeños moderadamente mejores con imágenes típicas y significativos con imágenes médicas de rayos X.

1.2 Problema a resolver

Con el creciente desarrollo de aplicaciones en Internet y de las tecnologías de multimedia, la cantidad de información que maneja una computadora ha crecido en forma exponencial con respecto a décadas anteriores. Esta información requiere gran cantidad de espacio para su almacenamiento y un amplio ancho de banda para su transmisión, en las cuales las nuevas tecnologías están limitadas técnica y económicamente. Posibles soluciones a este problema se encuentran en la compresión de la información, buscando que los requerimientos en su procesamiento, almacenamiento y transmisión sean reducidos.

Cuando la información es representada como imágenes, la importancia de la compresión es acentuada por la enorme cantidad de información existente en ellas: una imagen en escala de grises de 512×512 píxeles, cada uno de ellos representado por 8 bits, contiene 256 Kbytes de datos; si se trata de una imagen de color la información se triplica; si lo que se desea procesar es una secuencia de video de 25 cuadros por segundo, se necesitan aproximadamente 19 Mbytes de memoria para almacenar un segundo de dicha secuencia.

Debido a esta cantidad de información, en los sistemas de comunicación donde se requiere transmitir imágenes y video, el procesamiento y codificación de los mismos son una parte esencial, además de que estas operaciones son las que más recursos del sistema consumen. Si bien los sistemas computacionales actuales poseen gran capacidad de procesamiento, es necesaria la optimización de los algoritmos encargados del tratamiento de imágenes y video para obtener un mejor desempeño por parte de estos.

Debido a esto, las técnicas de compresión de imágenes son de importancia fundamental para reducir la cantidad de información necesaria en su representación sin que el sistema de visión humano sea capaz de detectar la pérdida de calidad de la misma.

En el diseño de un sistema de compresión de imágenes se deben considerar tres parámetros esenciales: la relación de compresión obtenida, la distorsión que generó el proceso de compresión sobre la imagen recuperada y la velocidad de procesamiento y recursos requeridos por el sistema.

Algunas memorias asociativas han demostrado ser una excelente herramienta en el reconocimiento y recuperación de patrones, sin importar que éstos contengan ruido aditivo, sustractivo o mixto; características que hacen atractivas a este tipo de memorias son la robustez al ruido, la capacidad de aprendizaje, la rapidez en el proceso de aprendizaje y la eficiencia y velocidad en el proceso de recuperación de patrones.

Tomando en cuenta estas consideraciones, la propuesta de usar memorias asociativas en la compresión de imágenes fundamenta sus bases en aprovechar las características mencionadas y aplicarlas en la obtención de algoritmos de compresión con una alta velocidad de procesamiento y alta inmunidad a la distorsión propia de un proceso de compresión.

1.3 Objetivo

El objetivo central de esta investigación consiste en la obtención de un sistema de compresión de imágenes mediante el desarrollo de un nuevo modelo de transformada con base en las Memorias Asociativas Morfológicas, denominada *Transformada Morfológica.*

1.4 Contribuciones

La aportación principal de este trabajo de tesis es la *Transformada Morfológica*, un nuevo modelo de transformada que basa su funcionamiento en las Memorias Asociativas Morfológicas y que surge como una alternativa a métodos tradicionales usados en la etapa de transformación de un sistema de compresión de imágenes.

Una contribución adicional es un esquema de compresión de imágenes desarrollado con base en la *Transformada Morfológica*, sustituyendo a métodos tradicionales como la DCT y la DWT. La principal ventaja que la *Transformada Morfológica* ofrece con respecto a los métodos tradicionales son: la velocidad de procesamiento y bajos requerimientos de memoria, demostrando además una alta competitividad en los parámetros de compresión y relación señal a ruido.

1.5 Organización del documento

Este primer capítulo incluye: el contexto, la definición del problema a resolver, el objetivo y las contribuciones de este trabajo de tesis. El resto de este trabajo de tesis está organizado de la siguiente forma:

El Capítulo 2 incluye los conceptos básicos relacionados con las memorias asociativas y se describen brevemente los modelos propuestos más importantes relacionados con esta rama de la ciencia. Además, incluye los fundamentos teóricos de la compresión de imágenes, así como un compendio de algunos trabajos que pueden tener estrecha relación con el propuesto en esta investigación.

Las Memorias Asociativas Morfológicas son base esencial del modelo propuesto en este trabajo de tesis; por tal motivo el Capítulo 3 versa sobre los fundamentos teóricos y matemáticos de este tipo de memorias asociativas.

El Capítulo 4 describe un nuevo modelo de transformada con base en las Memorias Asociativas Morfológicas, denominado *Transformada Morfológica*, el cual es aplicado en la etapa de transformación de un compresor de imágenes remplazando a métodos tradicionales como la DCT o la DWT. Contiene los algoritmos de la *Transformada Morfológica* y de la *Transformada Morfológica Inversa*, así como el fundamento matemático que sustenta su diseño. Finalmente, incluye un análisis de la complejidad del modelo propuesto.

En el Capítulo 5 se presentan la implementación de un esquema de compresión de imágenes con base en la *Transformada Morfológica* y los resultados experimentales obtenidos con este esquema. En primer término se estudia el desempeño de la *Transformada Morfológica* cuando se aplica a la imagen transformada un proceso de cuantificación vectorial con libros de códigos de diferentes tamaños; posteriormente se analiza el desempeño de la *Transformada Morfológica* cuando opera en forma conjunta con diversos algoritmos de codificación; además, se compara su desempeño con el de los métodos tradicionales de transformación, DCT y DWT; finalmente, se presenta un análisis comparativo del número y tipo de operaciones y de la cantidad de memoria que utilizan la *Transformada Morfológica* y los métodos tradicionales durante un proceso de transformación de una imagen.

El Capítulo 6 presenta las conclusiones obtenidas de este trabajo de tesis y las perspectivas o trabajos futuros que se plantean para la continuación de esta investigación.

Dos apéndices complementan la estructura de esta tesis.

El apéndice A contiene la simbología empleada en la tesis.

El propósito del apéndice B es describir cada uno de los bloques que integran un sistema de compresión de imágenes, centrándose especialmente en las principales técnicas empleadas en cada una de estas etapas.

Finalmente, se presentan las referencias bibliográficas.

Capítulo 1

Introducción

1.1 Contexto

El propósito de un sistema de compresión de imágenes es codificar los datos en una forma compacta reduciendo el número de bits usados en su representación y minimizando la distorsión causada por la compresión. Un compresor de imagen está formado por tres bloques principales: un *transformador de código*, un *cuantificador* y un *codificador* [4]; cada uno de estos bloques tiene por función reducir o eliminar las redundancias que pueden aparecer en una imagen.

El creciente uso y demanda que las tecnologías de multimedia y comunicaciones móviles han experimentado en los últimos años, y seguirá en aumento en las décadas venideras, han originado que las técnicas de compresión de imágenes y video jueguen un papel primordial dentro de una nueva área en el mundo de las telecomunicaciones llamada Comunicaciones Multimedia Móviles (*Mobile Multimedia Comunications*).

En el año 2005 los servicios de multimedia móvil tuvieron aproximadamente 32 millones de usuarios tan sólo en el oeste de Europa, lo cual representa un mercado de 24 billones de euros y 3800 millones de Mbytes de tráfico por año. Para el año 2010 se prevé que el crecimiento de usuarios de multimedia móvil será de 1.7 billones [4], [90].

En este trabajo de investigación se estudian las técnicas más importantes en el área de compresión de imágenes con la finalidad de proponer y desarrollar nuevos algoritmos que proporcionen mejores resultados en la relación de compresión, la relación señal a ruido y la complejidad computacional con respecto a los algoritmos actuales.

A mediados de la década de los 80's, el CCITT (*Consultative Committee for International Telegraphy and Telephony*) y el ISO (*International Organization for Standardization*) empezaron a trabajar en conjunto en el desarrollo de un estándar internacional para la compresión de imágenes, pero no fue sino hasta el año de 1992 que se logra el primer estándar internacional para este propósito bajo el acrónimo JPEG (*Joint Photographic Experts Group*) [92]. El nombre formal del estándar JPEG es ISO/IEC IS 10918-1 | ITU-T Recommendation T.81, tal como fue publicado por ISO y CCITT, ahora llamado ITU-T (*International Telecommunication Union*) [37].

El estándar JPEG describe una familia de técnicas de compresión de imágenes fijas de tonalidad continua en escala de grises o color (24 bits). Sin embargo, numerosas aplicaciones han usado la técnica también para compresión de video, porque proporciona descompresión de imagen de calidad bastante alta a una razón de compresión muy buena, y requiere menos poder de cálculo que la compresión MPEG (*Motion Pictures Experts Group*).

Debido a la cantidad de datos involucrada y la redundancia psicovisual en las imágenes, JPEG emplea un esquema de compresión con pérdidas basado en la codificación por transformación. El estándar resultante tiene tantas alternativas como sean necesarias para servir a una amplia variedad de propósitos. El estándar JPEG define tres sistemas diferentes de codificación: 1. Un sistema de codificación básico, con pérdidas, que se basa en la Transformada Discreta del Coseno (DCT, *Discrete Cosine Transform*) y es apropiado para la mayoría de las aplicaciones de compresión. 2. Un sistema de codificación extendida, para aplicaciones de mayor compresión, mayor precisión, o de reconstrucción progresiva. 3. Un sistema de codificación independiente sin pérdidas, para la compresión reversible [92].

En el año de 1997 surge el interés de desarrollar nuevas contribuciones al estándar JPEG con la finalidad de mejorarlo; este nuevo estándar fue nombrado JPEG2000 [39] y es referenciado como ISO/IEC JTC 1/SC 29/ WG 1 N1646R [38].

El estándar JPEG2000 se basa en un esquema de codificación por transformación, pero a diferencia del JPEG está fundamentalmente basado en la Transformada Discreta *Wavelet* (DWT, *Discrete Wavelet Transform*) y en el algoritmo *Embedded Block Coding with Optimal Truncation* (EBCOT) [88]. La DWT es aplicada a una imagen, entonces los coeficientes transformados son cuantificados y codificados para formar una secuencia de información comprimida [17].

Actualmente siguen surgiendo nuevas variantes en los esquemas de compresión, muestra de ello es el trabajo presentado por el investigador C. M. Sousa *et al.* [85], en el cual proponen un sistema de compresión de imágenes basado en el concepto *"Efficient Coding"*, concepto derivado de modelos neuronales para el procesamiento de la información. Este sistema esta formado por dos fases: aprendizaje y proyección. La fase de aprendizaje hace uso de un análisis independiente de componentes (ICA, *Independent Component Analisis*) para formar las funciones base a partir de una clase de datos específica. La fase de proyección consiste en encontrar la representación de la imagen para un subespacio (formado por las funciones base) que reduzca la dependencia estadística entre los coeficientes de proyección utilizados en la reconstrucción de la imagen.

En [62] A. Ouani *et al.* proponen un nuevo algoritmo para la compresión de imágenes denominado *Modified* EZW (*Embedded Zerotree Wavelet*); como su nombre lo indica se trata de una modificación al algoritmo de Shapiro, el EZW [81]. Una de las modificaciones hechas al algoritmo de Shapiro es el incremento de cuatro a seis símbolos en la etapa de prueba de significancía, los dos símbolos nuevos son usados cuando se determina que todos los descendientes de un coeficiente son insignificantes. Otra importante diferencia entre el EZW y el MEZW es la optimización de la codificación mediante el agrupamiento de elementos; debido a que existen seis símbolos, éstos son representados usando tres bits; el algoritmo MEZW lleva a cabo una agrupación de símbolos con la finalidad de poder representarlos con nueve bits, lo que se hace antes del proceso de codificación aritmética generando una optimización en este proceso. El algoritmo MEZW produce resultados que son significativamente mejores que los parámetros de relación a ruido y relación de compresión obtenidos por el algoritmo de Shapiro, sin afectar significativamente el tiempo de procesamiento.

S. Minasyan *et al.* en [56] proponen un método de representación unificada de un amplio número de algoritmos rápidos usados por las transformadas discretas ortogonales, obteniendo una clase de transformada la cual puede definir diferentes familias de transformadas dentro de ella. Este enfoque permite describir muchos algoritmos rápidos de transformadas en una forma unificada. En particular, las familias de transformadas Haar, Hadamard y Slant son definidas basadas en este enfoque. Los autores también proponen dos esquemas de compresión de imágenes utilizando este método de transformación; los experimentos realizados con estos esquemas muestran desempeños moderadamente mejores con imágenes típicas y significativos con imágenes médicas de rayos X.

1.2 Problema a resolver

Con el creciente desarrollo de aplicaciones en Internet y de las tecnologías de multimedia, la cantidad de información que maneja una computadora ha crecido en forma exponencial con respecto a décadas anteriores. Esta información requiere gran cantidad de espacio para su almacenamiento y un amplio ancho de banda para su transmisión, en las cuales las nuevas tecnologías están limitadas técnica y económicamente. Posibles soluciones a este problema se encuentran en la compresión de la información, buscando que los requerimientos en su procesamiento, almacenamiento y transmisión sean reducidos.

Cuando la información es representada como imágenes, la importancia de la compresión es acentuada por la enorme cantidad de información existente en ellas: una imagen en escala de grises de 512×512 píxeles, cada uno de ellos representado por 8 bits, contiene 256 Kbytes de datos; si se trata de una imagen de color la información se triplica; si lo que se desea procesar es una secuencia de video de 25 cuadros por segundo, se necesitan aproximadamente 19 Mbytes de memoria para almacenar un segundo de dicha secuencia.

Debido a esta cantidad de información, en los sistemas de comunicación donde se requiere transmitir imágenes y video, el procesamiento y codificación de los mismos son una parte esencial, además de que estas operaciones son las que más recursos del sistema consumen. Si bien los sistemas computacionales actuales poseen gran capacidad de procesamiento, es necesaria la optimización de los algoritmos encargados del tratamiento de imágenes y video para obtener un mejor desempeño por parte de estos.

Debido a esto, las técnicas de compresión de imágenes son de importancia fundamental para reducir la cantidad de información necesaria en su representación sin que el sistema de visión humano sea capaz de detectar la pérdida de calidad de la misma.

En el diseño de un sistema de compresión de imágenes se deben considerar tres parámetros esenciales: la relación de compresión obtenida, la distorsión que generó el proceso de compresión sobre la imagen recuperada y la velocidad de procesamiento y recursos requeridos por el sistema.

Algunas memorias asociativas han demostrado ser una excelente herramienta en el reconocimiento y recuperación de patrones, sin importar que éstos contengan ruido aditivo, sustractivo o mixto; características que hacen atractivas a este tipo de memorias son la robustez al ruido, la capacidad de aprendizaje, la rapidez en el proceso de aprendizaje y la eficiencia y velocidad en el proceso de recuperación de patrones.

Tomando en cuenta estas consideraciones, la propuesta de usar memorias asociativas en la compresión de imágenes fundamenta sus bases en aprovechar las características mencionadas y aplicarlas en la obtención de algoritmos de compresión con una alta velocidad de procesamiento y alta inmunidad a la distorsión propia de un proceso de compresión.

1.3 Objetivo

El objetivo central de esta investigación consiste en la obtención de un sistema de compresión de imágenes mediante el desarrollo de un nuevo modelo de transformada con base en las Memorias Asociativas Morfológicas, denominada *Transformada Morfológica.*

1.4 Contribuciones

La aportación principal de este trabajo de tesis es la *Transformada Morfológica*, un nuevo modelo de transformada que basa su funcionamiento en las Memorias Asociativas Morfológicas y que surge como una alternativa a métodos tradicionales usados en la etapa de transformación de un sistema de compresión de imágenes.

Una contribución adicional es un esquema de compresión de imágenes desarrollado con base en la *Transformada Morfológica*, sustituyendo a métodos tradicionales como la DCT y la DWT. La principal ventaja que la *Transformada Morfológica* ofrece con respecto a los métodos tradicionales son: la velocidad de procesamiento y bajos requerimientos de memoria, demostrando además una alta competitividad en los parámetros de compresión y relación señal a ruido.

1.5 Organización del documento

Este primer capítulo incluye: el contexto, la definición del problema a resolver, el objetivo y las contribuciones de este trabajo de tesis. El resto de este trabajo de tesis está organizado de la siguiente forma:

El Capítulo 2 incluye los conceptos básicos relacionados con las memorias asociativas y se describen brevemente los modelos propuestos más importantes relacionados con esta rama de la ciencia. Además, incluye los fundamentos teóricos de la compresión de imágenes, así como un compendio de algunos trabajos que pueden tener estrecha relación con el propuesto en esta investigación.

Las Memorias Asociativas Morfológicas son base esencial del modelo propuesto en este trabajo de tesis; por tal motivo el Capítulo 3 versa sobre los fundamentos teóricos y matemáticos de este tipo de memorias asociativas.

El Capítulo 4 describe un nuevo modelo de transformada con base en las Memorias Asociativas Morfológicas, denominado *Transformada Morfológica*, el cual es aplicado en la etapa de transformación de un compresor de imágenes remplazando a métodos tradicionales como la DCT o la DWT. Contiene los algoritmos de la *Transformada Morfológica* y de la *Transformada Morfológica Inversa*, así como el fundamento matemático que sustenta su diseño. Finalmente, incluye un análisis de la complejidad del modelo propuesto.

En el Capítulo 5 se presentan la implementación de un esquema de compresión de imágenes con base en la *Transformada Morfológica* y los resultados experimentales obtenidos con este esquema. En primer término se estudia el desempeño de la *Transformada Morfológica* cuando se aplica a la imagen transformada un proceso de cuantificación vectorial con libros de códigos de diferentes tamaños; posteriormente se analiza el desempeño de la *Transformada Morfológica* cuando opera en forma conjunta con diversos algoritmos de codificación; además, se compara su desempeño con el de los métodos tradicionales de transformación, DCT y DWT; finalmente, se presenta un análisis comparativo del número y tipo de operaciones y de la cantidad de memoria que utilizan la *Transformada Morfológica* y los métodos tradicionales durante un proceso de transformación de una imagen.

El Capítulo 6 presenta las conclusiones obtenidas de este trabajo de tesis y las perspectivas o trabajos futuros que se plantean para la continuación de esta investigación.

Dos apéndices complementan la estructura de esta tesis.

El apéndice A contiene la simbología empleada en la tesis.

El propósito del apéndice B es describir cada uno de los bloques que integran un sistema de compresión de imágenes, centrándose especialmente en las principales técnicas empleadas en cada una de estas etapas.

Finalmente, se presentan las referencias bibliográficas.

Capítulo 2

Antecedentes

Este capítulo incluye tres temas vinculados al objetivo de este trabajo de tesis; en los dos primeros temas se presentan los conceptos básicos relacionados con las memorias asociativas y se describen brevemente los modelos propuestos más importantes relacionados con esta rama de la ciencia. El tercer tema, contiene los fundamentos teóricos de la compresión de imágenes, así como un compendio de algunos trabajos que tienen estrecha relación con el propuesto en esta investigación.

2.1 Memorias asociativas, conceptos generales

La capacidad asociativa de la memoria humana es considerada como una propiedad esencial en el funcionamiento del cerebro humano. El científico Donald O. Hebb en [33] hizo conjeturas acerca de la memoria biológica y su capacidad asociativa, sosteniendo que la memoria biológica debía residir en las conexiones sinápticas y que los procesos de aprendizaje y de memorización involucraban cambios en los valores de dichas conexiones. Así, estas sinapsis que unen a neuronas activadas simultáneamente, se hacen más fuertes y favorecen las rutas en el cerebro que son excitadas por experiencias específicas, lo que implica cambios locales progresivos en estas conexiones, en vez de cambios globales, consiguiendo una considerable economía en tiempo y espacio y por tanto mejorando el rendimiento. Es decir, según este modelo, la información disponible para cada neurona es local (disponible físicamente por la sinapsis) reduciendo así el conjunto de conocimientos a tratar en cada momento.

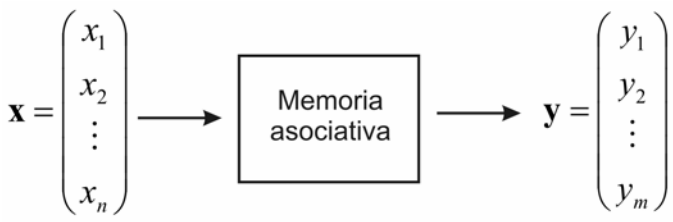


Figura 2.1. Esquema de una memoria asociativa.

Las características generales de una memoria asociativas son [49]:

- Mantiene una relación de dato con dato.
- Cada elemento en la estructura guarda información local y global.
- La información similar almacena y recupera información similar.
- Sólo se requiere una pequeña parte para recuperar la información completa.
- Permite inferir de lo desconocido lo conocido.
- El almacenamiento y la recuperación son consecuencias de la misma estructura.
- Actúa como base de datos estableciendo asociaciones de dos o más elementos.
- El tiempo de búsqueda es independiente del número de elementos almacenados.
- Los elementos aprendidos comparten la misma zona de memoria.

Una memoria asociativa es un elemento formado mediante la asociación de elementos (patrones de entrada y salida) y cuyo propósito fundamental es recuperar patrones de salida completos a partir de patrones de entrada que pueden estar alterados con ruido sustractivo, aditivo o mixto.

El esquema de una memoria asociativa genérica **Mg** se observa en la figura 2.1; se trata de un sistema de entrada-salida, donde los patrones de entrada y salida están representados por los vectores columna **x** y **y** respectivamente, donde *n* y *m* son números enteros positivos y representan las dimensiones de los patrones de entrada y salida respectivamente.

Además, sean $\{(\mathbf{x}^1,\mathbf{y}^1),(\mathbf{x}^2,\mathbf{y}^2),...,(\mathbf{x}^k,\mathbf{y}^k)\}$ *k* pares de asociaciones de vectores de entrada y vectores de salida; a esta combinación se le denomina *conjunto fundamental de asociaciones* [101]. El conjunto fundamental de asociaciones queda representado de la siguiente manera:

$$\{(\mathbf{x}^\mu,\mathbf{y}^\mu)\mid \mu=1,2,...,k\} \qquad \textbf{(2.1)}$$

A partir de este momento, la ecuación 2.1 representará al conjunto fundamental de asociaciones utilizado en las explicaciones siguientes, a menos que se aclare lo contrario. La memoria asociativa **Mg** se representa mediante una matriz y es generada a partir del conjunto fundamental de asociaciones.

Considerando como está constituido el conjunto fundamental de asociaciones, las memorias asociativas se clasifican en dos grupos:

- *Memorias Autoasociativas*. Una memoria asociativa es de carácter autoasociativo si se cumple que $\mathbf{x}^\mu=\mathbf{y}^\mu \forall \mu\in\{1,2,...,k\}$ [106].
- *Memorias Heteroasociativas*. Una memoria asociativa es de carácter heteroasociativo si se cumple que $\mathbf{x}^\mu\neq\mathbf{y}^\mu \exists \mu\in\{1,2,...,k\}$ [107].

Considerando estas definiciones, se puede establecer que una memoria autoasociativa puede ser considerada como un caso particular de una memoria heteroasociativa.

Generalizando, una memoria asociativa cumple con su función en dos procesos básicos: *proceso de aprendizaje* o *entrenamiento* y *proceso de recuperación* o *reconocimiento*.

El proceso de aprendizaje, utilizado para crear la memoria asociativa, consiste en codificar la relación existente entre la información de cada uno de los elementos del conjunto fundamental de asociaciones, $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$, y almacenarla en la memoria asociativa. A la solución de un problema de aprendizaje con base en reglas previamente establecidas se le conoce como *algoritmo de aprendizaje*. Existen dos tipos de entrenamiento:

- Entrenamiento supervisado. En este tipo de entrenamiento la memoria recibe ambos vectores, de entrada y de salida, y realiza una asociación entre éstos.
- Entrenamiento no supervisado. En este tipo de entrenamiento únicamente los vectores de entrada son canalizados a la memoria en una secuencia definida; entonces por cada vector la memoria genera un identificador y realiza una asociación entre ellos.

El proceso de recuperación o reconocimiento, operación de la memoria asociativa, permite la recuperación de un patrón de salida, **y**, mediante su vector de entrada asociado, **x**, y la memoria creada en el proceso de aprendizaje, es decir, cuando un patrón de entrada es presentado a la memoria asociativa, el vector de salida asociado es generado. En este caso se dice que la recuperación para ese patrón fue perfecta. A la solución de un problema de reconocimiento con base en reglas previamente establecidas se le conoce como *algoritmo de reconocimiento*.

2.2 Evolución de las memorias asociativas

El desarrollo de las memorias asociativas se ha efectuado en forma paralela a las redes neuronales, desde que McCulloch y Pitts crearon el primer modelo de una neurona artificial [55], hasta los modelos de redes neuronales basados en conceptos modernos como la morfología matemática [73]. A continuación se hace mención de los modelos de memorias asociativas más importantes.

2.2.1 Lernmatrix

En el año 1961 el científico Karl Steinbuch desarrolló un método para codificar información en arreglos cuadriculados conocidos como *crossbar*. Este modelo fue llamado Lernmatrix [102], [78]. La Lernmatrix es una memoria heteroasociativa que con una selección adecuada de los patrones de salida puede funcionar como un clasificador de patrones binarios.

Es un sistema que al operar acepta como entrada un patrón binario $\mathbf{x}^{\mu} \in A^n, A = \{0,1\}$ y produce como salida la clase $\mathbf{y}^{\mu} \in A^k$ que le corresponde. Para representar la clase $t \in \{1,2,...k\}$ se asignan a los componentes del vector de salida los siguientes valores: $y_t^{\mu} = 1$, y $y_j^{\mu} = 0$ para $j = 1,2,...,t-1,t+1,...,k$.

Al iniciar la fase de aprendizaje, cada uno de los componentes de la matriz **M** tienen un valor inicial igual a cero y se actualiza de acuerdo con la regla $m_{ij} + \Delta m_{ij}$, definiendo a Δm_{ij} como:

$$\Delta m_{ij} = \begin{cases} +\varepsilon \text{ si } x_i^{\mu} = 1 = y_j^{\mu} \\ -\varepsilon \text{ si } x_i^{\mu} = 0 \text{ y } y_j^{\mu} = 1 \\ 0 \text{ en otro caso} \end{cases} \quad \textbf{(2.2)}$$

donde ε es una constante positiva escogida previamente.

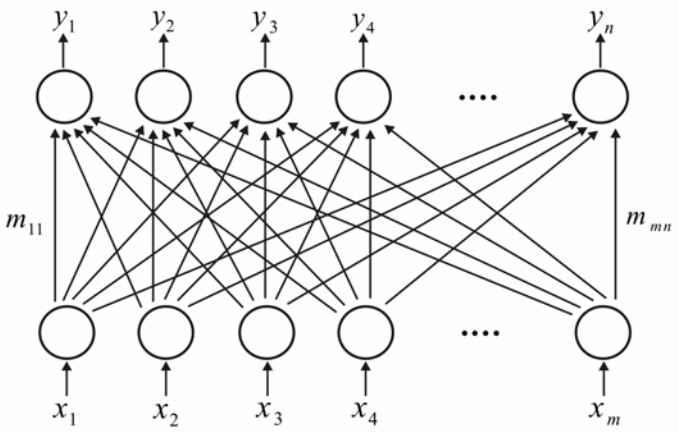


Figura 2.2. Modelo del Asociador lineal.

La memoria **M** resultante del proceso de aprendizaje tiene la siguiente estructura:

$$\begin{array}{c|cccccc} & x_1^\mu & x_2^\mu & \cdots & x_j^\mu & \cdots & x_n^\mu \\ \hline y_1^\mu & m_{11} & m_{12} & \cdots & m_{1j} & \cdots & m_{1n} \\ y_2^\mu & m_{21} & m_{22} & \cdots & m_{2j} & \cdots & m_{2n} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ y_i^\mu & m_{i1} & m_{i2} & \cdots & m_{ij} & \cdots & m_{in} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ y_k^\mu & m_{k1} & m_{k2} & \cdots & m_{kj} & \cdots & m_{kn} \end{array} \qquad \textbf{(2.3)}$$

La fase de recuperación consiste en encontrar la clase a la que pertenece un vector de entrada $\mathbf{x}^\omega$ dado, esto es, obtener las coordenadas del vector $\mathbf{y}^\omega$ que le corresponde a $\mathbf{x}^\omega$ sin equívocos, la *i-ésima* coordenada de $\mathbf{y}^\omega$ se obtiene de la siguiente forma:

$$y_i^\omega = \begin{cases} 1 \text{ si } \sum_{j=1}^{n} m_{ij}.x_j^\omega = \bigvee_{h=1}^{k} \left[\sum_{j=1}^{n} m_{hj}.x_j^\omega \right] \\ 0 \text{ en otro caso} \end{cases} \qquad \textbf{(2.4)}$$

donde **V** es el operador *máximo*.

2.2.2 Asociador lineal

El Asociador lineal (*Linear Associator*) es un modelo precursor de las memorias asociativas producto de 2 trabajos publicados en el año 1972, *Models of Interative Memory* de James A. Anderson [9], y *Correlation Matrix Memories* de Teuvo Kohonen [43].

El Asociador lineal es uno de modelos de memoria asociativa más simples. La figura 2.2 muestra su arquitectura básica.

La fase de aprendizaje del Asociador lineal consiste de dos etapas:

1. Para el conjunto fundamental de asociaciones $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) | \mu = 1,2,...,k\}$, se encuentra la matriz $\mathbf{y}^\mu \cdot (\mathbf{x}^\mu)^t$ de dimensiones $m \times n$; $\mathbf{x}^\mu \in A^n$ y $\mathbf{y}^\mu \in A^m$, $A = \{0,1\}$

$$\mathbf{y}^{\mu}\cdot\left(\mathbf{x}^{\mu}\right)^{t}=\begin{pmatrix} y_1^{\mu}x_1^{\mu} & y_1^{\mu}x_2^{\mu} & \cdots & y_1^{\mu}x_j^{\mu} & \cdots & y_1^{\mu}x_n^{\mu} \\ y_2^{\mu}x_1^{\mu} & y_2^{\mu}x_2^{\mu} & \cdots & y_2^{\mu}x_j^{\mu} & \cdots & y_2^{\mu}x_n^{\mu} \\ \vdots & \vdots & & \vdots & & \vdots \\ y_i^{\mu}x_1^{\mu} & y_i^{\mu}x_2^{\mu} & \cdots & y_i^{\mu}x_j^{\mu} & \cdots & y_i^{\mu}x_n^{\mu} \\ \vdots & \vdots & & \vdots & & \vdots \\ y_m^{\mu}x_1^{\mu} & y_m^{\mu}x_2^{\mu} & \cdots & y_m^{\mu}x_j^{\mu} & \cdots & y_m^{\mu}x_n^{\mu} \end{pmatrix} \tag{2.5}$$

2. La memoria es obtenida sumando las matrices resultantes

$$\mathbf{M}=\sum_{\mu=1}^{k}\mathbf{y}^{\mu}\cdot\left(\mathbf{x}^{\mu}\right)^{t}=\left[m_{ij}\right]_{m\times n} \tag{2.6}$$

$$m_{ij}=\sum_{\mu=1}^{k}y_i^{\mu}x_j^{\mu}$$

La fase de recuperación consiste en presentar a la memoria un patrón de entrada $\mathbf{x}^{\omega}$, donde $\omega\in\{1,2,\ldots k\}$ y realizar la operación

$$\mathbf{M}\cdot\mathbf{x}^{\omega}=\left[\sum_{\mu=1}^{k}\mathbf{y}^{\mu}\cdot\left(\mathbf{x}^{\mu}\right)^{t}\right]\cdot\mathbf{x}^{\omega}=\mathbf{y}^{\omega} \tag{2.7}$$

Para que la expresión anterior dé como resultado el patrón y^{ω}, es preciso que los vectores de entrada sean ortonormales; en caso contrario, debido al ruido producido por la interacción de los patrones de entrada, la recuperación no es perfecta, excepto si el número de patrones almacenado es pequeño comparado con la dimensión de los vectores de entrada. Lo expuesto anteriormente es un extracto de [103], donde se hace una explicación amplia y clara del Asociador lineal.

2.2.3 Red de auto-organización de elementos de umbral, Amari

Shun-ichi Amari publicó en 1972 un trabajo teórico titulado *Learning Patterns and Pattern Sequences by Self – Organizing Nets of Threshold Elements* [5], en el cual estableció un importante antecedente para la creación de lo que se convertiría en el modelo de la memoria asociativa.

Parte del trabajo de Amari pretende establecer nuevas teorías sobre el funcionamiento de las memorias asociativas, proponiendo novedosos métodos y enfoques teóricos desarrollados en su artículo, los que son resumidos en [100] y son los siguientes:

1. Una corta descripción de las ya conocidas (en 1972) redes de elementos de umbral, las transiciones de estados y estados de equilibrio;
2. Definiciones de los números de estabilidad de la transición de estados y los números de estabilidad de los estados de equilibrio;
3. Condiciones de convergencia para los números de estabilidad de los estados de equilibrio;
4. Los fundamentos de la auto-organización de las redes de elementos de umbral;
5. Las redes auto-organizativas de elementos de umbral como memorias asociativas.

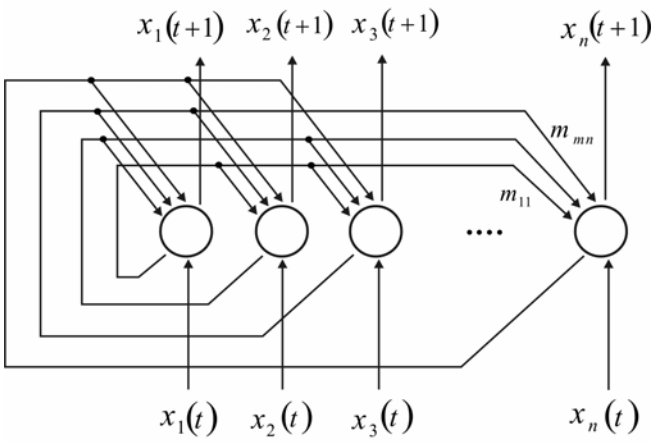


Figura 2.3. Modelo Hopfield.

2.2.4 Memoria Hopfield

Antes del trabajo del científico John Hopfield la investigación sobre memorias asociativas y redes neuronales tenía más de una década de no producir avances importantes. En el año de 1982 Hopfield describe la memoria asociativa Hopfield en [34]; este punto en la historia puede ser considerado como el nacimiento de la era moderna de las memorias asociativas, ya que sirvió para recuperar el área de investigación de las memorias asociativas y las redes neuronales.

Este modelo consiste en una red monocapa con *n* nodos, figura 2.3, cuyos valores de salida son binarios: 0/1 ó -1/+1. Existen conexiones laterales (cada nodo se encuentra conectado a todos los demás) pero no autorrecurrentes (no consigo mismo). Los pesos asociados a las conexiones entre pares de nodos son simétricos $m_{ij} = m_{ji}$.

La memoria de Hopfield es de carácter autoasociativo; por consiguiente el conjunto fundamental para la memoria Hopfield está definido como $\{(\mathbf{x}^\mu, \mathbf{x}^\mu) \mid \mu = 1,2,...,k\}$, donde $x^\mu \in A^n$ y $A = \{1,-1\}$.

En la fase de aprendizaje, cada componente m_{ij} de la memoria **M** se construye en 3 etapas como sigue:

1. Para cada par del conjunto fundamental, se calcula la matriz $\mathbf{x}^\mu \cdot (\mathbf{x}^\mu)^t$ de dimensiones $n \times n$, la cual tiene 1's en su diagonal principal.

$$\mathbf{x}^\mu \cdot (\mathbf{x}^\mu)^t = \begin{pmatrix} x_1^\mu x_1^\mu & x_1^\mu x_2^\mu & \cdots & x_1^\mu x_j^\mu & \cdots & x_1^\mu x_n^\mu \\ x_2^\mu x_1^\mu & x_2^\mu x_2^\mu & \cdots & x_2^\mu x_j^\mu & \cdots & x_2^\mu x_n^\mu \\ \vdots & \vdots & & \vdots & & \vdots \\ x_i^\mu x_1^\mu & x_i^\mu x_2^\mu & \cdots & x_i^\mu x_j^\mu & \cdots & x_i^\mu x_n^\mu \\ \vdots & \vdots & & \vdots & & \vdots \\ x_n^\mu x_1^\mu & x_n^\mu x_2^\mu & \cdots & x_n^\mu x_j^\mu & \cdots & x_n^\mu x_n^\mu \end{pmatrix} \quad \textbf{(2.8)}$$

2. A cada una de las matrices se le resta la matriz identidad **I** de dimensiones $n \times n$, con el fin de lograr 0's en la diagonal principal.

3. Se suman las matrices $\mathbf{x}^{\mu} \cdot \left(\mathbf{x}^{\mu}\right)^{t} - \mathbf{I}$ para finalmente obtener la matriz **M**:

$$\mathbf{M} = \sum_{\mu=1}^{k} \left[\mathbf{x}^{\mu} \cdot \left(\mathbf{x}^{\mu}\right)^{t} - \mathbf{I}\right] = \left[m_{ij}\right]_{n \times n} \qquad \textbf{(2.9)}$$

$$m_{ij} \begin{cases} \sum_{\mu=1}^{k} x_i^{\mu} x_j^{\mu} & \text{si} \quad x \neq j \\ 0 & \text{si} \quad x = j \end{cases}$$

Durante el proceso de aprendizaje, la memoria Hopfield cambia su estado con el tiempo; cada neurona x_i ajusta su valor en función de la comparación resultante entre la cantidad $\sum_{j=1}^{n} m_{ij} x_j$ y un umbral cuyo valor normalmente es cero. Al representar el estado de la memoria Hopfield en el tiempo t por $x_i(t+1)$, $x_i(t)$ representara el valor de la neurona x_i en el tiempo t y $x_i(t+1)$ el valor de x_i en el tiempo siguiente $t+1$.

Dado un vector columna de entrada $\tilde{\mathbf{x}}$, la fase de recuperación consta de 3 pasos:

1. Para $t = 0$, se hace $\mathbf{x}(t) = \tilde{\mathbf{x}}$, es decir, $x_i(0) = \tilde{x}_i, \forall i \in \{1,2,3,...,n\}$;
2. $\forall i \in \{1,2,3,...,n\}$ se calcula $x_i(t+1)$ de acuerdo con la condición siguiente:

$$x_i(t+1) = \begin{cases} 1 \text{ si } \sum_{j=1}^{n} m_{ij} x_j(t) > 0 \\ x_i(t) \text{ si } \sum_{j=1}^{n} m_{ij} x_j(t) = 0 \\ -1 \text{ si } \sum_{j=1}^{n} m_{ij} x_j(t) < 0 \end{cases} \qquad \textbf{(2.10)}$$

3. Si $x(t+1) = x(t)$ el proceso termina y el vector recuperado es $\mathbf{x}(0) = \tilde{\mathbf{x}}$. En cualquier otro caso los pasos 2 y 3 se iteran tantas veces como sea necesario hasta llegar al punto límite localmente establecido en el tiempo $t = \tau$, para el cual $x_i(\tau+1) = x_i(\tau) \forall i \in \{1,2,3,...,n\}$; entonces el proceso termina y el patrón recuperado es $\mathbf{x}(\tau)$.

2.2.5 Memoria Asociativa Bidireccional

En 1988 el científico Barl Kosko define la Memoria Asociativa Bidireccional (BAM, *Bidirectional Associative Memories*) en [48]. Esta memoria asociativa es una extensión del modelo Hopfield mediante la incorporación de una capa adicional para realizar autoasociaciones recurrentes así como heteroasociaciones en las memorias almacenadas.

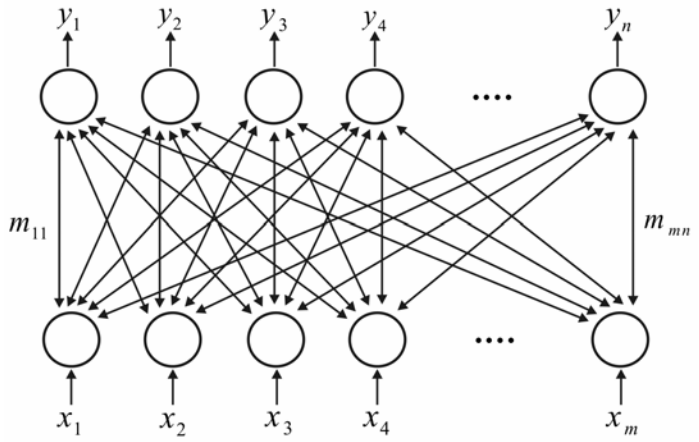


Figura 2.4. Modelo de la Memoria Asociativa Bidireccional**.**

La estructura del modelo BAM es similar a la del Asociador lineal, pero las conexiones en la BAM son bidireccionales, esto es, $m_{ij} = m_{ji}$, para $i = 1,2,...,m$ y $j = 1,2,...,n$. Una característica importante de este modelo es que ambos capas puede fungir como entradas o como salida, en dependencia de la dirección de propagación. La figura 2.4 ilustra la arquitectura de una BAM.

En la fase de aprendizaje del modelo BAM se generan dos matrices **M** y su transpuesta **W**, sin la restricción de que la diagonal principal tenga 0's. Para el cálculo de la memoria asociativa, como en los modelos del Asociador lineal y Hopfield, primero se calcula la matriz $\mathbf{y}^{\mu} \cdot \left(\mathbf{x}^{\mu}\right)^t$ de dimensiones $m \times n$, para cada par del conjunto fundamental de asociaciones, dado por $\left\{\left(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}\right) | \, \mathbf{x}^{\mu} \in A^n, \mathbf{y}^{\mu} \in A^m, \mu = 1,2,...,k\right\}$, donde $A = \{-1,1\}$, y la memoria **M** es obtenida aplicando la ecuación 2.6.

La transpuesta de la memoria **M** representa a la memoria **W** de dimensiones $n \times m$

$$\begin{aligned} \mathbf{W} &= \left(\mathbf{M}\right)^t \\ &= \left[\sum_{\mu=1}^{k} \mathbf{x}^{\mu} \cdot \left(\mathbf{y}^{\mu}\right)^t\right] = \left[w_{ij}\right]_{n \times m} \end{aligned} \qquad \textbf{(2.11)}$$

En una BAM, el proceso de recuperación implica el reflejar la información distribuida entre las dos capas hasta que la red llegue a ser estable. Al introducir un patrón de entrada, la memoria propaga este vector hacia la otra capa permitiendo que sus unidades actualicen sus valores; el patrón producido es entonces devuelto a la primera capa permitiendo que sus unidades calculen sus nuevos valores. El nuevo patrón producido por la primera capa es nuevamente propagado a la otra capa. Este proceso es repetido hasta que los estados de las unidades de ambos lados no presenten cambios significativos.

En una BAM, al propagar un vector de entrada desde la capa ***X*** hacia la capa ***Y***, la capa ***Y*** calculará los valores de entrada a los nodos usando:

$$in\_y_j = \sum_{i=1}^{n} x_i m_{ij} \qquad \textbf{(2.12)}$$

y determina sus valores de salida usando:

$$y_i(t+1) = \begin{cases} 1 \text{ si } in\_y_j > 0 \\ y_i(t) \text{ si } in\_y_j = 0 \\ -1 \text{ si } in\_y_j < 0 \end{cases} \quad \textbf{(2.13)}$$

para $i = 1,2,...,m$.

El patrón producido en la capa ***Y*** es entonces propagado de regreso a la capa ***X***, permitiendo que los valores de entrada de los nodos se actualicen mediante

$$in\_x_j = \sum_{i=1}^{m} y_i w_{ij} \quad \textbf{(2.14)}$$

y determina sus valores de salida usando:

$$x_i(t+1) = \begin{cases} 1 \text{ si } in\_x_j > 0 \\ x_i(t) \text{ si } in\_x_j = 0 \\ -1 \text{ si } in\_x_j < 0 \end{cases} \quad \textbf{(2.15)}$$

para $i = 1,2,...,n$.

Existe una función de energía que caracteriza el estado de la memoria. La función de energía de una BAM está dada por:

$$E = -\sum_{i=1}^{m}\sum_{j=1}^{n} x_i m_{ij} y_j \quad \textbf{(2.16)}$$

Kosko demostró que la energía de una BAM decrece o se mantiene después de que cada nodo se actualiza; de esta forma la memoria eventualmente convergerá a un valor mínimo que corresponde al patrón de salida asociado al patrón de entrada inicial. La energía de una BAM es limitada por:

$$E = -\sum_{i=1}^{m}\sum_{j=1}^{n} \left| m_{ij} \right| \quad \textbf{(2.17)}$$

Una BAM tiene una capacidad de memoria seriamente baja. Para que una BAM tenga una recuperación perfecta, Kosko determinó que la capacidad de almacenamiento sea menor que $\min(m,n)$. Un estudio más reciente hecho por Tanaka *et al.* reveló que para una BAM, que tiene *n* unidades en cada una de sus capas, su capacidad de almacenamiento es de 0.1998*n* [87].

2.2.6 Memorias Asociativas Morfológicas

El algoritmo propuesto en este trabajo de tesis hace uso de las Memorias Asociativas Morfológicas; por tal motivo se ha decidido dedicar el apartado 3.1 para hacer una descripción más detallada de este modelo de memoria asociativa.

2.2.7 Memorias Asociativas $\alpha\beta$

En el año 2002, un nuevo modelo de memoria asociativa denominadas memorias asociativas $\alpha\beta$ fue desarrollado por C. Yáñez en [99]. Una memoria asociativa $\alpha\beta$ puede ser de carácter heteroasociativo o de carácter autoasociativo.

Estas memorias reciben ese particular nombre debido a que sus operaciones se basan en dos operadores nuevos: α (*alfa*) y β (*beta*), definidos por:

$$\begin{aligned} \alpha &= A \times A \to B \\ \beta &= B \times A \to A \end{aligned} \tag{2.18}$$

donde los conjuntos *A* y *B* son definidos como: $A = \{0,1\}$ y $B = \{0,1,2\}$. Las operaciones binarias realizadas por estos operadores son definidos por:

$$\begin{aligned} \alpha(x,y) &= x - y + 1 \\ \beta(x,y) &= \begin{cases} 1 \text{ si } y + x > 1 \\ 0 \text{ en otro caso} \end{cases} \end{aligned} \tag{2.19}$$

Las operaciones binarias α y β exhiben ciertas propiedades en las que se ven involucrados los operadores máximo $\vee$ y mínimo $\wedge$. Esto genera un sistema algebraico $(A, B, \alpha, \beta, \wedge, \vee)$ en el que están involucradas las memorias asociativas $\alpha\beta$.

Las memorias asociativas $\alpha\beta$ hacen uso de cuatro operaciones matriciales. Sean $P = [p_{ij}]_{m\times h}$ y $Q = [q_{ij}]_{h\times n}$ dos matrices, entonces:

$$\begin{aligned} &\text{Operación } \alpha\max: P \cup_\alpha Q = [f_{ij}^\alpha]_{m\times n}, \text{ donde } f_{ij}^\alpha = \bigvee_{k=i}^{h} \alpha(p_{ik}, q_{kj}). \\ &\text{Operación } \alpha\min: P \cap_\alpha Q = [g_{ij}^\alpha]_{m\times n}, \text{ donde } g_{ij}^\alpha = \bigwedge_{k=i}^{h} \alpha(p_{ik}, q_{kj}). \\ &\text{Operación } \beta\max: P \cup_\beta Q = [f_{ij}^\beta]_{m\times n}, \text{ donde } f_{ij}^\beta = \bigvee_{k=i}^{h} \beta(p_{ik}, q_{kj}). \\ &\text{Operación } \beta\min: P \cap_\beta Q = [g_{ij}^\beta]_{m\times n}, \text{ donde } g_{ij}^\beta = \bigwedge_{k=i}^{h} \beta(p_{ik}, q_{kj}). \end{aligned} \tag{2.20}$$

Las operaciones $\cup_\alpha$ y $\cap_\alpha$ entre vectores están definidas por:

$$\mathbf{y} \cup_\alpha \mathbf{x}^t = \mathbf{y} \boxtimes \mathbf{x}^t = \mathbf{y} \cap_\alpha \mathbf{x}^t \tag{2.21}$$

donde el símbolo $\boxtimes$ representa ambas operaciones $\cup_\alpha$ y $\cap_\alpha$; $\mathbf{y} = [y_i]_m$ y $\mathbf{x} = [x_j]_n$ son vectores de dimensiones *m* y *n* respectivamente.

El resultado de esta operación es una matriz de dimensión $m \times n$:

$$\mathbf{y} \boxtimes \mathbf{x}^t = \begin{pmatrix} \alpha(y_1, x_1) & \alpha(y_1, x_2) & \cdots & \alpha(y_1, x_n) \\ \alpha(y_2, x_1) & \alpha(y_2, x_2) & \cdots & \alpha(y_2, x_n) \\ \vdots & \vdots & & \vdots \\ \alpha(y_m, x_1) & \alpha(y_m, x_2) & \cdots & \alpha(y_m, x_n) \end{pmatrix} \tag{2.22}$$

donde la *ij*-ésima componente de la matriz está dada por:

$$\left[\mathbf{y} \boxtimes \mathbf{x}^t\right]_{ij} = \alpha(y_i, x_j) \tag{2.23}$$

Extendiendo para cada elemento del conjunto fundamental de asociaciones:

$$\left[\mathbf{y}^\mu \boxtimes \left(\mathbf{x}^\mu\right)^t\right]_{ij} = \alpha\left(y_i^\mu, x_j^\mu\right) \tag{2.24}$$

Las memorias asociativas $\alpha\beta$ pueden ser heteroasociativas y autoasociativas; a su vez cada una de ellas puede ser de tipo *max* (**V**) o *min* (**Λ**).

La fase de aprendizaje para una memoria heteroasociativa $\alpha\beta$ *max* está definida por los siguientes pasos:

1. Para cada elemento del conjunto fundamental de asociaciones se construye la matriz $\left[\mathbf{y}^\mu \boxtimes \left(\mathbf{x}^\mu\right)^t\right]_{m \times n}$
2. El operador binario *máximo* $\bigvee$ es aplicado a las matrices obtenidas del paso anterior:

$$\mathbf{V} = \bigvee_{\mu=1}^{k} \left[\mathbf{y}^\mu \boxtimes \left(\mathbf{x}^\mu\right)^t\right] = \left[v_{ij}\right]_{m \times n} \tag{2.25}$$

$$v_{ij} = \bigvee_{\mu=1}^{k} \alpha\left(y_i^\mu, x_j^\mu\right)$$

La fase de recuperación se resume en los siguientes pasos:

1. Al presentar un vector $\mathbf{x}^\omega$, $\omega = 1,2,...,k$, a la memoria, se realiza la operación $\mathbf{V} \mathbin{\text{⋒}}_\beta \mathbf{x}^\omega$. El resultado de esta operación es un vector columna de dimensión *m*, cuya *i*-ésima componente es:

$$\left(\mathbf{V} \mathbin{\text{⋒}}_\beta \mathbf{x}^\omega\right)_i = \bigwedge_{j=1}^{n} \beta\left(v_{ij}, x_j^\omega\right) \tag{2.26}$$

$$\left(\mathbf{V} \mathbin{\text{⋒}}_\beta \mathbf{x}^\omega\right)_i = \bigwedge_{j=1}^{n} \beta\left\{\left[\bigvee_{\mu=1}^{k} \alpha\left(y_i^\mu, x_j^\mu\right), x_j^\omega\right]\right\} \tag{2.27}$$

Para el caso de una memoria heteroasociativa $\alpha\beta$ *min* se usa la propiedad de dualidad con respecto a las memorias heteroasociativas $\alpha\beta$ *max*. Por tanto, para los procesos de aprendizaje y recuperación de una memoria heteroasociativa $\alpha\beta$ *min*, se realizan las siguientes modificaciones a las expresiones 2.25, 2.26 y 2.27:

- Se sustituye el operador $\vee$ por el operador $\wedge$.
- Se sustituye el operador $\wedge$ por el operador $\vee$.
- Se usa el operador $\cup_\beta$ en lugar del operador $\cap_\beta$.

Las memorias heteroasociativa $\alpha\beta$ *min* son robustas al ruido sustractivo, mientras las memorias heteroasociativas $\alpha\beta$ *max* lo son al ruido aditivo.

Para una memoria autoasociativa $\alpha\beta$ el conjunto fundamental de asociaciones queda definido como $\{(\mathbf{x}^\mu, \mathbf{x}^\mu) \mid \mu = 1,2,...,k\}$; debido a que se puede considerar a una memoria autoasociativa $\alpha\beta$ como un caso particular de las heteroasociativas, entonces todas las consideraciones de estas últimas puede aplicarse a las primeras con sólo sustituir al vector $\mathbf{y}$ por el vector $\mathbf{x}$; la memoria resultante, para ambos tipos $\mathbf{V}$ y $\mathbf{\Lambda}$, será de dimensión $n \times n$, $\mathbf{V} = [v_{ij}]_{n\times n}$ y $\mathbf{\Lambda} = [\lambda_{ij}]_{n\times n}$.

Las memorias asociativas $\alpha\beta$ ofrecen la ventaja, con respecto a otros modelos de memorias asociativas como los morfológicos, de tener una menor densidad aritmética.

Las memorias asociativas $\alpha\beta$ tienen 2 limitaciones importantes: únicamente pueden operar con vectores binarios y son robustas sólo a ruido sustractivo o aditivo, no al mezclado. El grupo de trabajo del Dr. Yáñez ha desarrollado nuevos modelos basados en memorias asociativas $\alpha\beta$ que han subsanado dichas carencias.

En [26] se describe un modelo de memoria asociativa $\alpha\beta$ que es capaz de aceptar valores reales en los patrones; este modelo hace uso de un código derivado del Johnson-Möbius para codificar los valores de los patrones; además, se presenta una aplicación de este modelo de memoria asociativa $\alpha\beta$ en el área del tratamiento industrial del color [108].

En [1] se presenta un modelo de memorias asociativas bidireccionales $\alpha\beta$; este modelo hace uso del código binario *one-hot* y de un nuevo código propuesto, el código binario *zero-hot*.

2.2.8 Memorias Asociativas tipo Mediana

Las memorias asociativas tipo mediana fueron propuestas por J. H. Sossa, R. Barrón y R. A. Vázquez en el año 2003 [83], [84]. Este modelo de memorias presenta un gran desempeño en procesos de recuperación de patrones con valores reales en sus componentes y que se ven alterados por ruido mezclado.

Las memorias asociativas tipo mediana pueden ser de carácter heteroasociativo o autoasociativo. Este modelo de memoria hace uso de los operadores A y B, definidos de la siguiente forma:

$$\begin{aligned} \mathrm{A}(x, y) &= x - y \\ \mathrm{B}(x, y) &= x + y \end{aligned} \qquad \textbf{(2.28)}$$

Las memorias asociativas tienen dos operaciones básicas, medA $(\Diamond_{\mathrm{A}})$ y medB $(\Diamond_{\mathrm{A}})$.

Sean $P=\left[p_{ij}\right]_{m\times h}$ y $Q=\left[q_{ij}\right]_{h\times n}$ dos matrices, entonces:

$$\begin{aligned}&\text{Operación medA: } P\lozenge_{\mathbf{A}}Q=\left[f_{ij}^{\mathbf{A}}\right]_{m\times n},\ \text{donde } f_{ij}^{\mathrm{A}}=\bigotimes_{k=i}^{h}\mathrm{A}\left(p_{ik},q_{kj}\right).\\&\text{Operación medB: } P\lozenge_{\mathbf{B}}Q=\left[f_{ij}^{\mathbf{B}}\right]_{m\times n},\ \text{donde } f_{ij}^{\mathrm{B}}=\bigotimes_{k=i}^{h}\mathrm{B}\left(p_{ik},q_{kj}\right).\end{aligned}\qquad \textbf{(2.29)}$$

Donde el operador $\otimes$ representa a los operadores max $(\vee)$, min $(\wedge)$ o mediana $(\lozenge)$ para los casos en que se desea compensar la influencia en los patrones de ruido aditivo, sustractivo o mixto respectivamente.

Además se definen las operaciones donde intervienen vectores; si $\mathbf{y}=\left[y_i\right]_m$ y $\mathbf{x}=\left[x_j\right]_n$ son vectores de dimensiones *m* y *n* respectivamente, entonces $\mathbf{y}\lozenge_{\mathrm{A}}\mathbf{x}$ es una matriz de dimensión $m\times n$:

$$\mathbf{y}\lozenge_{\mathrm{A}}\mathbf{x}^{t}=\begin{pmatrix}\mathrm{A}\left(y_1,x_1\right) & \mathrm{A}\left(y_1,x_2\right) & \cdots & \mathrm{A}\left(y_1,x_n\right)\\ \mathrm{A}\left(y_2,x_1\right) & \mathrm{A}\left(y_2,x_2\right) & \cdots & \mathrm{A}\left(y_2,x_n\right)\\ \vdots & \vdots & & \vdots\\ \mathrm{A}\left(y_m,x_1\right) & \mathrm{A}\left(y_m,x_2\right) & \cdots & \mathrm{A}\left(y_m,x_n\right)\end{pmatrix}\qquad \textbf{(2.30)}$$

La operación $P\lozenge_{\mathrm{B}}\mathbf{x}$, donde $P=\left[p_{ij}\right]_{m\times h}$ es una matriz de dimensión $m\times n$ y $\mathbf{x}=\left[x_j\right]_n$ es un vector, es definida por:

$$P\lozenge_{\mathrm{B}}\mathbf{x}=\underset{j=1}{\overset{n}{\mathrm{med}}}\mathrm{B}\left(p_{ij},x_j\right)\qquad \textbf{(2.31)}$$

La fase de entrenamiento de una memoria asociativa tipo mediana se define en dos pasos:

1. Para cada elemento del conjunto fundamental de asociaciones se construye la matriz $\left[\mathbf{y}^{\mu}\lozenge_{\mathbf{A}}\left(\mathbf{x}^{\mu}\right)^{t}\right]_{m\times n}$.
2. El operador mediana es aplicado a las matrices obtenidas del paso anterior, lo cual genera la memoria **M**:

$$\begin{aligned}\mathbf{M}&=\underset{\mu=1}{\overset{k}{\mathrm{med}}}\left[\mathbf{y}^{\mu}\lozenge_{\mathrm{A}}\left(\mathbf{x}^{\mu}\right)^{t}\right]=\left[m_{ij}\right]_{m\times n}\\ m_{ij}&=\underset{\mu=1}{\overset{k}{\mathrm{med}}}\mathrm{A}\left(y_i^{\mu},x_j^{\mu}\right)\end{aligned}\qquad \textbf{(2.32)}$$

La fase de recuperación se resume en los siguientes pasos:

1. Al presentar, a la memoria, un vector $\mathbf{x}^{\omega}$, $\omega=1,2,...,k$, se realiza la operación: $\mathbf{M}\lozenge_{\mathrm{B}}\mathbf{x}^{\omega}$. El resultado de esta operación es un vector columna de dimensión *m*, cuya *i*-ésima componente es:

$$\left(\mathbf{M}\lozenge_{\mathrm{B}}\mathbf{x}^{\omega}\right)_i=\underset{j=1}{\overset{n}{\mathrm{med}}}\mathrm{B}\left(m_{ij},x_j^{\omega}\right)\qquad \textbf{(2.33)}$$

2.3 Fundamentos de la compresión de imágenes

Con la finalidad de tener un marco de referencia y lograr un mejor entendimiento sobre los algoritmos propuestos en este trabajo, esta sección incluye los fundamentos teóricos relacionados con la compresión de imágenes.

La importancia de la compresión de una imagen es acentuada por la enorme cantidad de información existente en ella: una imagen en escala de grises de 512×512 píxeles, cada uno de ellos representado por 8 bits, contiene 256 Kbytes de datos; si se trata de una imagen de color la información se triplica; si lo que se desea procesar es una secuencia de video de 25 cuadros por segundo, se necesitan aproximadamente 19 Mbytes de memoria para almacenar un segundo de dicha secuencia. Entonces, la necesidad de comprimir la información es obvia.

2.3.1 Teoría de la compresión de datos

En el año 1948 Claude E. Shannon publico el artículo "*A Mathematical Theory of Communication*" [80], donde formuló la teoría de la compresión de datos. Shannon definió la información contenida en un evento *I(E)*, medida en bits, en términos de la probabilidad de dicho evento *p(E)*. En general podemos decir que la cantidad de información contenida en un evento *E* que posee una probabilidad de aparición *p(E)* es:

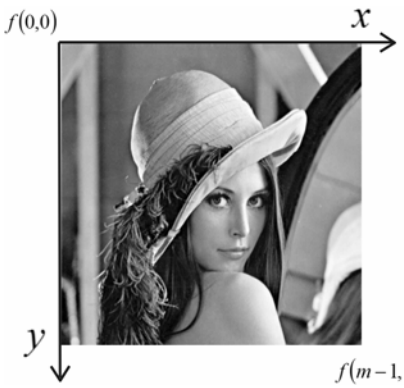
$$I(E) = \log_2\left(\frac{1}{p(E)}\right) \tag{2.34}$$

Siendo la base del logaritmo la que determina las unidades de información.

Para un mensaje o secuencia de eventos, la probabilidad de un mensaje en particular es el producto de las probabilidades de los eventos que forman la secuencia y la información contenida en el mensaje es la suma de la información de cada evento. Shannon estableció que existe un límite fundamental en la compresión de datos sin pérdida de información denominado *entropía* y denotado por *H*. El valor de *H* depende de la naturaleza estadística de la fuente de información. La entropía indica el límite teórico para la compresión de datos. También es una medida de la información contenida en un mensaje. Debido a que la probabilidad de cada miembro del alfabeto determina su frecuencia, entonces el promedio de información por símbolo es la suma de la información de cada miembro del alfabeto multiplicada por su probabilidad. Si se tiene un alfabeto de *n* símbolos, s_1 a s_n, y la probabilidad del miembro s_i es $p(s_i)$, entonces la entropía $H(S)$ puede ser calculada por:

$$\begin{aligned} H(S) &= p(s_1)I(s_1) + p(s_2)I(s_2) + \ldots + p(s_n)I(s_n) \\ H(S) &= \sum_{i=1}^{n} p(s_i)I(s_i) \\ H(S) &= \sum_{i=1}^{n} p(s_i)\log_2\left(\frac{1}{p(s_i)}\right) \end{aligned} \tag{2.35}$$

donde $H(S)$ es el número promedio de bits requeridos para codificar un símbolo, conocida como distribución de probabilidad de los símbolos.



$$f(x,y) \approx \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,n-1) \\ f(1,0) & f(1,1) & \cdots & f(1,n-1) \\ \vdots & \vdots & & \vdots \\ f(m-1,0) & f(m-1,1) & \cdots & f(m-1,n-1) \end{bmatrix}$$

Figura 2.5. Representación discreta de una imagen.

2.3.2 Representación discreta de una imagen

Una imagen digital es la discretización tanto en coordenadas como en tonos de gris de una imagen analógica. El proceso de la digitalización de las coordenadas espaciales (x, y) en una imagen digital se le denomina muestreo de la imagen y la digitalización de la amplitud es la cuantificación del nivel de gris. Entonces, una imagen puede ser representada por una función $f(x, y)$, donde x y y representan las coordenadas espaciales y el valor de f es proporcional a la intensidad (o nivel de gris) de la imagen en ese punto [66].

Se puede establecer que una imagen analógica puede ser representada en forma aproximada por una serie de muestras igualmente espaciadas, obteniendo una matriz de $m \times n$ elementos (figura 2.5). En la figura se muestra cómo los píxeles de la imagen son dispuestos en la matriz $f(x, y)$. Las filas se enumeran de arriba hacia abajo y las columnas de izquierda a derecha. Los índices de los renglones y columnas empiezan en cero y tienen un valor máximo de $m-1$ y $n-1$ respectivamente.

2.3.3 Tipos de imágenes

Considerando las técnicas existentes para la compresión de imágenes, existen los siguientes tipos de imágenes [13]:

- Imágenes binarias.
- Imágenes en tonos de gris.
- Imágenes de color.
- Cuadros de video.

Una ilustración de esta clasificación es mostrada en la figura 2.6. Las imágenes en tonos de gris, de color y los cuadros de video están estrechamente relacionadas, esto es, los métodos diseñados para comprimir imágenes en tonos de gris pueden ser usados en imágenes de color y a cuadros de video; sin embargo, estos métodos usualmente no son aplicados a imágenes binarias; por tanto, se considera a estas imágenes una clase distinta desde el punto de vista de las técnicas utilizadas en su compresión.

La figura 2.6 muestra también la información tipo texto. Existen dos diferencias fundamentales entre la información de una imagen y la secuencia de caracteres contenida en un texto, la primera de ellas es que una imagen es de 2 dimensiones, la segunda, frecuentemente llamada correlación, establece la estrecha relación existente entre muestras contiguas en una imagen; esta última propiedad es de gran importancia cuando se aplica un algoritmo de compresión a una imagen.

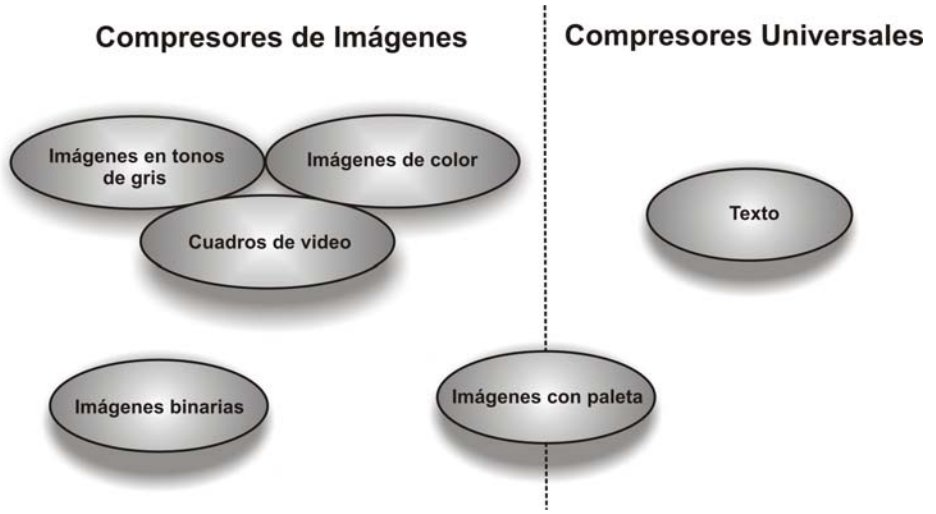


Figura 2.6. Clasificación de las imágenes.

2.3.4 Redundancias en las imágenes

El término compresión de datos se refiere al proceso de reducción del volumen de datos necesarios para representar una determinada cantidad de información. Se pueden utilizar distintas cantidades de datos para describir la misma cantidad de información; por lo tanto, existen datos que proporcionan información sin relevancia. A esta definición se le conoce como redundancia de los datos; este concepto es un punto clave en la compresión de datos digitales. En la compresión de imágenes, se pueden identificar y aprovechar tres tipos básicos de redundancias [50]:

- Redundancia de codificación.
- Redundancia entre píxeles.
- Redundancia psicovisual.

La compresión de datos se consigue cuando una o varias de estas redundancias se reducen o se eliminan.

2.3.4.1 Redundancia de codificación

Si los niveles de gris de una imagen están codificados de forma que se emplean más símbolos que los estrictamente necesarios para representar cada uno de ellos, entonces se dice que la imagen resultante tiene redundancia de código. En general, la redundancia de código aparece cuando los códigos asignados a un conjunto de niveles de gris no han sido seleccionados de modo que se obtenga el mayor rendimiento posible de las probabilidades de estos niveles.

Los compresores basados en la eliminación de la redundancia de codificación aprovechan el hecho de que la frecuencia de uso de los símbolos del alfabeto, utilizado para representar la información, no sigue una distribución uniforme.

2.3.4.2 Redundancia entre píxeles

La redundancia entre píxeles aparece debido a la correlación existente entre píxeles cercanos. Estas correlaciones resultan de las relaciones estructurales o geométricas entre los objetos presentes en la imagen.

Puesto que es posible predecir razonablemente el valor de un determinado píxel a partir del valor de sus vecinos, la información que aporta individualmente un píxel es relativamente pequeña; por tanto, la mayor parte de la contribución visual de un único píxel a una imagen es redundante debido a que puede ser inferida usando los valores de sus vecinos. En relación con estas dependencias entre píxeles se han generado una serie de nombres como redundancia espacial, redundancia geométrica y redundancia interna.

Con el fin de reducir las redundancias entre píxeles de una imagen, la distribución bidimensional de píxeles normalmente empleada para la percepción e interpretación humana debe ser transformada a un formato que elimine la correlación y resulte más simple y eficaz su compresión.

Los algoritmos que eliminan la redundancia espacial tienen en cuenta que normalmente existe algún tipo de correlación entre uno o varios de los píxeles ya aparecidos y él o los que van a aparecer.

2.3.4.3 Redundancia psicovisual

El ojo humano no responde con la misma sensibilidad a toda la información visual. Cierta información tiene menor importancia relativa que otra en el proceso visual normal. Se dice que esta información es psicovisualmente redundante y se puede eliminar sin que se altere significativamente la calidad de la percepción de la imagen.

En general, un observador intenta fijarse en características tales como las fronteras, el color, las aristas o las texturas de las regiones y las combina mentalmente para realizar asociaciones con otros grupos de patrones conocidos. El cerebro correlaciona esta información con el conocimiento anterior y así realiza la interpretación de la imagen

Al contrario de la redundancia de codificación y la redundancia entre píxeles, la redundancia psicovisual está asociada a la información visual real o cuantificable. Su eliminación es únicamente posible porque la propia información no es esencial para el procesamiento visual normal. Como la eliminación de los datos psicovisualmente redundantes se traduce en una pérdida de información cuantitativa, a menudo se denomina cuantificación. Cuantificación significa que a un amplio rango de valores de entrada le corresponden un número limitado de valores de salida. Puesto que es una operación irreversible, ya que se pierde información visual, la cuantificación conduce a una compresión con pérdida de datos.

2.3.5 Compresión sin pérdida vs. compresión con pérdida

Una clasificación de algoritmos de compresión considera si estos generan o no pérdida en la información de la imagen durante el proceso compresión-descompresión al cual es sometida.

Un algoritmo de compresión es considerado sin pérdida (proceso reversible) si la imagen descomprimida es idéntica a la original, es decir, la información de la imagen se preserva. Un algoritmo de compresión es considerado con pérdidas (proceso irreversible) si la imagen reconstruida es sólo una aproximación de la imagen original, es decir, la información de la imagen se degrada. La propiedad de preservar la información es deseable en los métodos de compresión, pero muchas aplicaciones se pueden permitir excluirla.

Los métodos de compresión sin pérdida de información se caracterizan porque la tasa de compresión que proporcionan está limitada por la entropía de la imagen original. Entre estas técnicas destacan las que emplean métodos estadísticos basados en la teoría de Shannon.

Los métodos de compresión con pérdida de información logran alcanzar tasas de compresión más elevadas a costa de sufrir una pérdida de información de la imagen original.

2.3.6 Criterios de desempeño en compresión de imágenes

Los dos criterios principales que miden el desempeño de un algoritmo de compresión de imágenes son: la *eficiencia en la compresión* y la *distorsión* en la imagen causada por el algoritmo [2].

Algunos otros criterios que permiten de alguna manera medir el rendimiento de un algoritmo de compresión de imágenes son: la velocidad del proceso de compresión-descompresión de la imagen, la robustez contra los errores en la transmisión, la complejidad del algoritmo y los requerimientos de memoria del mismo.

2.3.6.1 Eficiencia en la compresión

La forma más simple de medir la eficiencia en la compresión es determinar el número promedio de bits almacenados por píxel de la imagen, *bit rate*

$$bit\ rate = \frac{\textbf{tamaño del archivo comprimido}}{\textbf{píxeles en la imagen}} = \frac{C}{N} \quad \textbf{(bits por píxel)} \tag{2.36}$$

donde C es el número de bits en el archivo comprimido y $N = m \cdot n$ es el número de píxeles en la imagen original. Si el *bit rate* tiene un valor muy bajo, la *relación de compresión* puede ser una medida más práctica

$$relación\ de\ compresión = \frac{\textbf{tamaño del archivo original}}{\textbf{tamaño del archivo comprimido}} = \frac{N \cdot k}{C} \tag{2.37}$$

donde k es el número de bits por píxel en la imagen original.

2.3.6.2 Distorsión

La medida de la distorsión puede ser dividida en dos categorías: subjetiva y objetiva, [2], [19], [65]. Una medida de la distorsión es subjetiva si la calidad es evaluada por humanos. Sin embargo, el uso de analistas humanos es poco práctico y por lo tanto raramente utilizado.

En mediciones objetivas la distorsión es calculada por una función predefinida como la diferencia entre la imagen original y la imagen reconstruida. Todos los cambios se consideran como distorsión, no importa si un observador humano los distingue o no. La distorsión cuantitativa de la imagen reconstruida se mide comúnmente mediante el error absoluto medio (MAE, *Mean Absolute Error*), el error cuadrático medio (MSE, *Mean Square Error*) y la relación señal a ruido (PSNR, *Peak Signal to Noise Ratio*) [86], [2].

El MAE se define como:

$$\textbf{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|\tilde{x}_i - x_i\right| \tag{2.38}$$

El MSE se define como:

$$\textbf{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(\tilde{x}_i - x_i\right)^2 \tag{2.39}$$

La PSNR se define como:

$$\mathbf{PSNR} = 10\log_{10}\left(\frac{\left(2^n - 1\right)^2}{\mathbf{MSE}}\right) \tag{2.40}$$

donde N es el número de píxeles en la imagen, x los elementos originales de la imagen, $\tilde{x}$ los elementos de la imagen recuperada y n es el número de bits por píxel.

Las medidas objetivas no siempre coinciden con las subjetivas, por ejemplo, el ojo humano no distingue pequeños cambios de intensidad entre píxeles, pero es sensible a los cambios en el contraste. Otra deficiencia de las medidas objetivas de distorsión es que únicamente miden diferencias locales entre píxeles, y no consideran detalles o efectos visuales globales como la distrosión por bloques o efecto pixelado [71] (*blockiness*, distorsión que aparece cuando se usan algoritmos basados en bloques; como la correlación espacial entre bloques adyacentes no es considerada en el procesamiento, esto hace visibles los límites de los bloques cuando se reconstruye la imagen. También se caracteriza por diferente brillo de los bloques, en otras palabras, el nivel promedio entre bloques vecinos es diferente, lo cual se refleja más claramente en píxeles que están cerca de los márgenes de los bloques), la distorsión por el efecto del desenfoque [71] (*blurring*, distorsión que se presenta como una imagen fuera de foco; esto ocurre cuando cada píxel en la imagen reconstruida está compuesto por una mezcla de píxeles circundantes de la imagen original), o la distorsión de escalera [42] (*jaggedness of the edges* o *edge jittering*, discontinuidad de los bordes, esta distorsión aparece como pasos visibles de escalera donde debe haber líneas rectas o curvas lisas).

2.3.7 Compresor de imágenes

El propósito de un sistema de compresión es codificar los datos de una imagen en una forma compacta reduciendo el número de bits usados en su representación y minimizando la distorsión causada por la compresión. Un compresor de imagen está formado por tres bloques principales: un *transformador de código*, un *cuantificador* y un *codificador* [4], figura 2.7. Cada uno de estos elementos es diseñado para reducir las redundancias de una imagen. El apéndice B describe cada uno de los bloques que integran un sistema de compresión de imágenes, prestando atención especial en las principales técnicas empleadas en cada una de estas etapas.

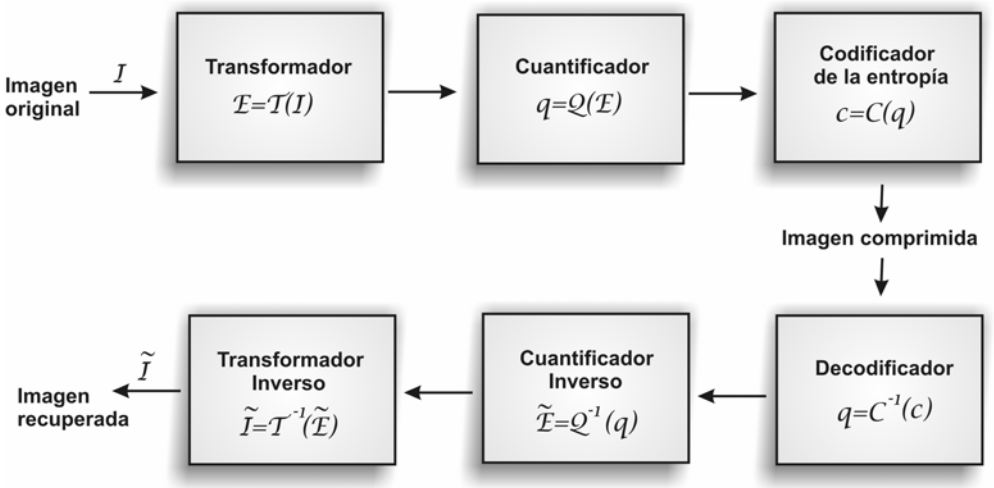


Figura 2.7. Elementos de un compresor de imágenes.

2.4 Estado del arte

Una nueva tecnología para compresión de imágenes con base en Redes Neuronales Artificiales (RNA) ha surgido como una alternativa a los métodos tradicionales. Utilizando este novedoso enfoque, nuevos esquemas de compresión de imágenes han sido creados o algoritmos existentes han sido modificados. Como se ha mencionado, en este trabajo de tesis se propone aplicar las MAM en la compresión de imágenes; una MAM puede ser considerada una subclase de las Redes Neuronales Morfológicas (MNN *Morphological Neural Networks*) [75], [73], [74]. Por tal motivo y para tener un panorama de la influencia de las RNA y las memorias asociativas en el área de la compresión de imágenes, en este apartado se hará mención de algunos trabajos que hacen uso de estos enfoques.

Los Mapas Auto-Organizados (SOM, *Self-Organizing Map*), creados por Teuvo Kohonen a inicios de la década de los 80 [44], [45] y considerados una RNA de aprendizaje competitivo, han sido utilizados con mucho éxito en la creación de nuevos esquemas para la cuantificación vectorial (VQ) [46]; la VQ es parte primordial de un compresor de imágenes.

Uno de los primeros trabajos donde los SOM fueron empleados para la compresión de imágenes fue presentado por A. Bogdan y H. E. Meadows [12]. Su algoritmo hace uso de los SOM y de la codificación mediante fractales para encontrar rasgos similares en representaciones de diferente resolución de una imagen; estos rasgos son caracterizados como patrones. Continuando con el proceso, los patrones son mapeados dentro de un arreglo de dos dimensiones de neuronas con la finalidad de formar un libro de códigos (*codebook*). La propiedad de organización u ordenamiento de los SOM permite encontrar la mejor correlación entre un patrón de entrada y un vector de reconstrucción que forma parte del *codebook*. Una característica importante de este algoritmo es la reducción de carga computacional cuando encuentra y remueve redundancias entre las diferentes representaciones de la imagen original.

C. Amerijckx *et al.* propusieron en [7] un esquema de compresión con pérdida para imágenes digitales fijas usando el algoritmo de la red neuronal de Kohonen. Ellos aplicaron los SOM en las etapas de cuantificación y codificación de un compresor de imágenes. En la etapa de cuantificación, el algoritmo de los SOM crea una nueva correspondencia entre un espacio de entrada de estímulos (formado por bloques de coeficientes DCT de una imagen) y un espacio de salida constituido por los elementos del *codebook* (vectores de reconstrucción, *codewords*) obtenidos mediante la distancia euclidiana y representados por neuronas. Después de este proceso de entrenamiento de la red, esta nueva correspondencia es empleada en el proceso de cuantificación y su proceso inverso. En la etapa de codificación, con la finalidad de alcanzar una mejor relación de compresión, un codificador por entropía diferencial hace uso de la topología de los SOM resultante del proceso de aprendizaje y de la hipótesis de que los bloques consecutivos en una imagen son a menudo similares. Comparado con el estándar JPEG, los resultados de este trabajo muestran que la calidad de las imágenes comprimidas con el algoritmo propuesto es mejor para relaciones de compresión mayores a 25. En [6], C. Amerijckx *et al.* propusieron un esquema de compresión sin pérdida de imágenes usando los SOM y los mismos principios.

M. Mokhtari y A. Boukelif [57] presentaron un nuevo algoritmo con base en la red neuronal de Kohonen, el cual acelera le proceso de compresión de imágenes basado en fractales. La red de Kohonen es usada en un algoritmo adaptativo que busca la mejor aproximación de un bloque de entrada con respecto a un conjunto definido de bloques mediante los parámetros de brillo y contraste de los coeficientes transformados. Cuando la diferencia entre bloques excede un límite predefinido, el bloque de entrada es dividido en cuatro subbloques. Este proceso de división se sigue repitiendo hasta que la diferencia entre bloques es inferior al límite, o el tamaño mínimo de bloque es alcanzado.

La desventaja principal de utilizar el algoritmo SOM es que se requiere un tiempo de entrenamiento largo debido a que la red debe comenzar con pesos iniciales arbitrarios. En [63], S. Panchanathan *et al.* usaron el algoritmo de propagación de error inversa o hacia atrás (BEP, *Backward Error Propagation*), creado por P. J. Werbos en 1974 [93], con la finalidad de obtener rápidamente los pesos iniciales, los cuales son utilizados para aumentar la velocidad del proceso de entrenamiento del algoritmo SOM. Este nuevo enfoque propuesto, BEP-SOM, combina las ventajas de ambas técnicas permitiendo obtener un buen desempeño en la codificación y un tiempo corto de entrenamiento.

Otro tipo de RNA que ha sido usada ampliamente en compresión de imágenes es la red neuronal con propagación hacia delante (FNN, *Feedforward Neural Network*). La FNN es clasificada en la categoria de RNA de transferencia de señal y se trata de la primera y posiblemente más simple RNA ideada. En esta red, la información fluye en una sola dirección, de los nodos de entrada a los nodos de salida pasando a través de los nodos ocultos.

En [79], R. Setiono y G. Lu aplicaron el algoritmo de una FNN a la compresión de imágenes. La construcción de la red neuronal empieza con una simple topología de red, la cual contiene únicamente una unidad en la capa oculta. Un conjunto de pesos óptimo para esta red es calculado aplicando una variante del método quasi-Newton [8], [22], también conocido como método métrico variable. Si este conjunto de pesos no da una red con la precisión deseada, entonces una unidad es añadida a la capa oculta y la red es nuevamente entrenada. El proceso es repetido hasta que la red genere la precisión deseada. Como resultado de este proceso, el algoritmo necesita de largos tiempos de entrenamiento, pero cada vez que una unidad es añadida a la red, la relación señal a ruido entre la imagen original y la imagen comprimida es incrementada.

Qiang Ji presentó un esquema para compresión de imágenes con base en una FNN lineal auto-organizada en [67]. El primer paso en el proceso de compresión es dividir la imagen en bloques de tamaño $m \times m$, cada uno de los cuales es representado como un vector de dimensión m^2; entonces, una red neuronal con entrada de dimensión m^2 y salida de dimensión m extrae los componentes principales de la matriz de auto-correlación de cada bloque de la imagen usando el algoritmo de aprendizaje Hebbiano generalizado (GHA, *Generalized Hebbian Learning Algorithm*). El proceso de entrenamiento de la red con base en el GHA genera una matriz de pesos de dimensión $m \times m^2$; los renglones de esta matriz son los vectores principales (*eigenvectors*) de las matrices de auto-correlación de los bloques de la imagen. La proyección de cada bloque de la imagen dentro de la matriz de vectores principales genera m coeficientes por cada bloque; la compresión de la imagen es completada mediante la cuantificación y codificación de los coeficientes de cada bloque.

En [76] S. B. Roy, K. Kayal, y J. Sil desarrollaron una técnica para compresión de imágenes mediante la conservación de bordes usando una capa oculta de una FNN. Las neuronas de la red son determinadas en forma adaptativa con base en la imagen a ser comprimida. En la primera fase y con la finalidad de reducir el tamaño de los datos a comprimir, se aplican los siguientes procesos a la imagen: detección de bordes, umbralizado y aclaración. La función principal de la segunda fase es determinar en forma adaptativa la estructura de la red usando el método de entrenamiento de propagación inversa; haciendo uso de esta red la imagen será codificada. Además, este método propone la inicialización de los pesos entre la capa de entrada y la capa oculta, transformando las coordenadas de los píxeles del bloque a su representación equivalente de una dimención; este proceso de inicialización exhibe un menor tiempo de convergencia del algoritmo de entrenamiento en comparación con un proceso que inicializa los pesos en forma aleatoria.

Los siguientes dos ejemplos muestran una relación directa entre las RNA y los métodos tradicionales de transformación: DCT y DWT. En el primero, Ng K. S. y Cheng L. M. propusieron la implementación de la DCT utilizando una RNA [60]. La estructura de la RNA es dividida en cuatro sub-redes tipo BEP: la primera tiene una estructura de $64\times16\times63$ y es utilizada para calcular la DCT; la segunda tiene una estructura de $64\times32\times4$ utilizada para clasificar la energía de los coeficientes; la tercera tiene una estructura $63\times16\times64$ y es utilizada para calcular la DCT inversa; la última es utilizada para ajustar el coeficiente de frecuencia cero o coeficiente continuo (DC, *Direct Current*) y tiene una estructura $64\times2\times1$. Cada sub-red es entrenada y probada en forma independiente, excepto la encargada de ajustar los coeficientes DC.

En el segundo ejemplo, Christopher J.C. Burges *et al.* utilizan un predictor no lineal, implementado con una RNA, para predecir los coeficientes *wavelets* en un sistema de compresión de imágenes [14]. El proceso consiste en reducir la varianza de los coeficientes residuales predecidos mediante el criterio objetivo MSE; entonces, el predictor no lineal puede ser utilizado para reducir la compresión de las cadenas de datos formadas. En la implementación del predictor mediante la RNA, los autores utilizan una red de 2 capas, donde la unidad de salida es una sigmoide que puede tomar valores comprendidos dentro del intervalo $[0,1]$. La red es entrenada utilizando cada sub-banda de cada nivel *wavelet*. Finalmente, las salidas son reescaladas al intervalo $[-1,1]$.

En [58], Nait H. y Salam F. M. describieron un práctico y efectivo sistema de compresión de imágenes con base en una red neuronal multicapa. El sistema sugerido está formado por dos redes neuronales multicapa; el proceso de compresión de la imagen se lleva a cabo en dos etapas: en la primera, una red neuronal comprime la imagen; en la segunda etapa, la otra red neuronal comprime la diferencia entre la imagen reconstruida y la imagen original.

Recientemente, Pavel Danchenko *et al.* desarrollaron un programa para compresión de imágenes de color y escala de gris usando RNA [20]. Este programa fue nombrado “Red Neuronal para la Compresión de Imágenes” (NNIC, *Neural Network Image Compressor*); NNIC hace uso en su etapa de transformación de la DCT, sobre la base de los coeficientes generados por la DCT es posible aplicar dos métodos de compresión, el primero de ellos hace uso de una arquitectura tipo perceptron multicapa; el segundo método está basado en una red neuronal de Kohonen.

Capítulo 3

Marco teórico

Este trabajo de tesis propone un modelo de compresor de imágenes utilizando un nuevo algoritmo en la etapa de transformación, la cual ha sido denominada *Transformada Morfológica*. Las Memorias Asociativas Morfológicas (MAM, *Morphological Associative Memories*) son la base esencial de esta nueva transformada, por lo que este capítulo versa sobre los fundamentos teóricos y matemáticos de las MAM.

3.1 Memorias Asociativas Morfológicas

En el año 1989, G. X. Ritter y J. Wilson introducen el concepto de Redes Neuronales Morfológicas [75]. Ellos propusieron calcular el efecto total que tiene de un valor de entrada sobre una neurona con la ayuda de las operaciones morfológicas de dilatación y de erosión. En 1996, G. X. Ritter y P. Sussner proponen las MAM con base en las Redes Neuronales Morfológicas [73]. Dos años después, G. X. Ritter, P. Sussner y J. L. Díaz de León desarrollaron en forma exhaustiva el concepto de las MAM [74].

Las MAM basan su operación en las operaciones morfológicas de dilatación y erosión, es decir, hacen uso de máximos o mínimos de sumas. Esta característica las distingue de las memorias Hopfield, las cuales utilizan sumas de productos.

Las MAM han demostrado ser una excelente herramienta en el reconocimiento y recuperación de patrones, sin importar que éstos contengan ruido aditivo, sustractivo o mixto [15], [106], [107].

Características que hacen atractivas a las MAM son la robustez al ruido, la capacidad de aprendizaje, la rapidez en el proceso de aprendizaje y la eficiencia y velocidad en el proceso de la recuperación de patrones.

Las operaciones necesarias para el proceso de aprendizaje y recuperación de una MAM hacen uso de los operadores *máximo* $\vee$ y *mínimo* $\wedge$ [107], [106], y son las siguientes:

Sean $D = [d_i]_m$ un vector columna de dimensión *m* y $F = [f_j]_n$ un vector fila de dimensión *n*, el *producto máximo* $D \nabla F$ da como resultado una matriz $C = [c_{ij}]_{m \times n}$, donde $c_{ij} = (d_i + f_j)$.

Generalizando para un conjunto fundamental de asociaciones tenemos:

$$c_{ij} = \bigvee_{l=1}^{k} (d_{il} + f_{lj}) \qquad \textbf{(3.1)}$$

De igual manera, el *producto mínimo* $D \Delta F$ da como resultado la matriz $C = [c_{ij}]_{m \times n}$; para un conjunto fundamental de asociaciones tenemos que:

$$c_{ij} = \bigwedge_{l=1}^{k} (d_{il} + f_{lj}) \qquad \textbf{(3.2)}$$

Por otra parte, sea $D = [d_{ij}]_{m \times n}$ una matriz y $F = [f_j]_n$ un vector columna; el cálculo del *producto máximo* $D \nabla F$ da como resultado un vector columna $C = [c_i]_m$ de dimensión *m*:

$$c_i = \bigvee_{j=1}^{n} (d_{ij} + f_j) \qquad \textbf{(3.3)}$$

Para el *producto mínimo* C = $D \Delta F$:

$$c_i = \bigwedge_{j=1}^{n} (d_{ij} + f_j) \qquad \textbf{(3.4)}$$

Estas operaciones se aplican a las MAM en sus dos modos de operación, heteroasociativo y autoasociativo.

3.1.1 Memorias Morfológicas Heteroasociativas

Una MAM es de carácter heteroasociativo si $\exists \mu \in \{1,2,...,k\}$ para el que se cumple que $\mathbf{x}^\mu \neq \mathbf{y}^\mu$. Existen dos tipos de Memorias Morfológicas Heteroasociativas (MMH): *max*, simbolizadas con **M,** y *min*, simbolizadas con **W**.

3.1.1.1 Memorias Morfológicas Heteroasociativas *max*

Las MMH *max* (**M**) son aquellas que utilizan el producto mínimo y el operador máximo en su fase de aprendizaje y el producto mínimo en su fase de recuperación.

Fase de aprendizaje:

1. Se calculan las matrices $\mathbf{y}^\mu \Delta (-\mathbf{x}^\mu)^t$ para cada uno de los *k* elementos del conjunto fundamental de asociaciones $(\mathbf{x}^\mu, \mathbf{y}^\mu)$.

$$\mathbf{y}^{\mu}\Delta\left(-\mathbf{x}^{\mu}\right)^{t} = \begin{pmatrix} \left[y_1^{\mu}+\left(-x_1^{\mu}\right)\right] & \cdots & \left[y_1^{\mu}+\left(-x_j^{\mu}\right)\right] & \cdots & \left[y_1^{\mu}+\left(-x_n^{\mu}\right)\right] \\ \left[y_2^{\mu}+\left(-x_1^{\mu}\right)\right] & \cdots & \left[y_2^{\mu}+\left(-x_j^{\mu}\right)\right] & \cdots & \left[y_2^{\mu}+\left(-x_n^{\mu}\right)\right] \\ \vdots & & \vdots & & \vdots \\ \left[y_i^{\mu}+\left(-x_1^{\mu}\right)\right] & \cdots & \left[y_i^{\mu}+\left(-x_j^{\mu}\right)\right] & \cdots & \left[y_i^{\mu}+\left(-x_n^{\mu}\right)\right] \\ \vdots & & \vdots & & \vdots \\ \left[y_m^{\mu}+\left(-x_1^{\mu}\right)\right] & \cdots & \left[y_m^{\mu}+\left(-x_j^{\mu}\right)\right] & \cdots & \left[y_m^{\mu}+\left(-x_n^{\mu}\right)\right] \end{pmatrix} \quad \textbf{(3.5)}$$

2. Se obtiene la memoria **M** aplicando el operador máximo $\bigvee$ a las matrices resultantes del paso 1. **M** queda expresada como:

$$\mathbf{M} = \bigvee_{\mu=1}^{k}\left[\mathbf{y}^{\mu}\Delta\left(-\mathbf{x}^{\mu}\right)^{t}\right] = \left[m_{ij}\right]_{m\times n} \quad \textbf{(3.6)}$$
$$m_{ij} = \bigvee_{\mu=1}^{k}\left(y_i^{\mu} - x_j^{\mu}\right)$$

Fase de recuperación:

1. Se calcula el producto mínimo $\mathbf{M}\Delta\mathbf{x}^{\omega}$, donde $\omega \in \{1,2,...,k\}$, obteniendo un vector columna $\mathbf{y} = \left[y_i\right]_m$:

$$\mathbf{y} = \mathbf{M}\Delta\mathbf{x}^{\omega} \quad \textbf{(3.7)}$$
$$y_i = \bigwedge_{j=1}^{n}\left(m_{ij} + x_j^{\omega}\right)$$

Para que una MMH *max* asegure una respuesta perfecta debe cumplir con el teorema 2 y el corolario 2.1 de [74]. Además, el teorema 3 de la misma referencia indica la cantidad de ruido que es permisible en los patrones de entrada para obtener una respuesta perfecta:

Teorema 2 [74]**:** $\mathbf{M}\Delta\mathbf{x}^{\omega} = \mathbf{y}^{\omega}$ para todo $\omega = 1,...,k$ si y sólo si para cada índice fila $i = 1,...,m$ existen índices columna $j_i^{\omega} \in \{1,...,n\}$ tales que $m_{ij_i^{\omega}} = y_i^{\omega} - x_{j_i^{\omega}}^{\omega}$ para todo $\omega = 1,...,k$.

Corolario 2.1 [74]**:** $\mathbf{M}\Delta\mathbf{x}^{\omega} = \mathbf{y}^{\omega}$ para todo $\omega = 1,...,k$ si y sólo si para cada índice fila $i = 1,...,m$ y cada $\gamma \in \{1,...,k\}$ existen índices de columna $j_i^{\gamma} \in \{1,...,n\}$ tales que

$$x_{j_i^{\gamma}}^{\gamma} = \bigwedge_{\varepsilon=1}^{k}\left(x_{j_i^{\gamma}}^{\varepsilon} - y_i^{\varepsilon}\right) + y_i^{\gamma} \quad \textbf{(3.8)}$$

Teorema 3 [74]**:** Para $\gamma = 1,...,k$, sea $\tilde{\mathbf{x}}^{\gamma}$ una versión distorsionada del patrón $\mathbf{x}^{\gamma}$. Entonces $\mathbf{M}\Delta\tilde{\mathbf{x}}^{\gamma} = \mathbf{y}^{\gamma}$ si y sólo si se cumple que:

$$\tilde{x}_j^{\gamma} \geq x_j^{\gamma} \wedge \bigvee_{i=1}^{m}\left(\bigwedge_{\varepsilon\neq\gamma}\left[y_i^{\gamma} - y_i^{\varepsilon} + x_i^{\varepsilon}\right]\right) \forall j = 1,...,n \quad \textbf{(3.9)}$$

y para cada índice renglón $i \in \{1,...,m\}$ existe un índice columna $j_i \in \{1,...,n\}$ tal que:

$$\tilde{x}_{ji}^{\gamma} = x_{ji}^{\gamma} \wedge \left(\bigwedge_{\varepsilon \neq \gamma} \left[y_i^{\gamma} - y_i^{\varepsilon} + x_{ji}^{\varepsilon} \right] \right) \tag{3.10}$$

3.1.1.2 Memorias Morfológicas Heteroasociativas *min*

Las MMH *min* (**W**) son aquéllas que utilizan el producto máximo y el operador mínimo en su fase de aprendizaje y el producto máximo en su fase de recuperación.

Fase de aprendizaje:

1. Se calculan las matrices $\mathbf{y}^{\mu} \nabla \left(-\mathbf{x}^{\mu} \right)^t$ para cada uno de los *k* elementos del conjunto fundamentas de asociaciones $\left(\mathbf{x}^{\mu}, \mathbf{y}^{\mu} \right)$.

$$\mathbf{y}^{\mu} \nabla \left(-\mathbf{x}^{\mu} \right)^t = \begin{pmatrix} \left[y_1^{\mu} + \left(-x_1^{\mu} \right) \right] & \cdots & \left[y_1^{\mu} + \left(-x_j^{\mu} \right) \right] & \cdots & \left[y_1^{\mu} + \left(-x_n^{\mu} \right) \right] \\ \left[y_2^{\mu} + \left(-x_1^{\mu} \right) \right] & \cdots & \left[y_2^{\mu} + \left(-x_j^{\mu} \right) \right] & \cdots & \left[y_2^{\mu} + \left(-x_n^{\mu} \right) \right] \\ \vdots & & \vdots & & \vdots \\ \left[y_i^{\mu} + \left(-x_1^{\mu} \right) \right] & \cdots & \left[y_i^{\mu} + \left(-x_j^{\mu} \right) \right] & \cdots & \left[y_i^{\mu} + \left(-x_n^{\mu} \right) \right] \\ \vdots & & \vdots & & \vdots \\ \left[y_m^{\mu} + \left(-x_1^{\mu} \right) \right] & \cdots & \left[y_m^{\mu} + \left(-x_j^{\mu} \right) \right] & \cdots & \left[y_m^{\mu} + \left(-x_n^{\mu} \right) \right] \end{pmatrix} \tag{3.11}$$

2. Se obtiene la memoria **W** aplicando el operador mínimo $\wedge$ a las matrices resultantes del paso 1. **W** queda expresada como:

$$\mathbf{W} = \bigwedge_{\mu=1}^{k} \left[\mathbf{y}^{\mu} \nabla \left(-\mathbf{x}^{\mu} \right)^t \right] = \left[w_{ij} \right]_{m \times n} \tag{3.12}$$

$$w_{ij} = \bigwedge_{\mu=1}^{k} \left(y_i^{\mu} - x_j^{\mu} \right)$$

Fase de recuperación:

1. Se calcula el producto máximo $\mathbf{W} \nabla \mathbf{x}^{\omega}$, donde $\omega \in \{1,2,...,k\}$, donde $\omega \in \{1,2,...,k\}$, obteniendo un vector columna $\mathbf{y} = \left[y_i \right]_m$:

$$\mathbf{y} = \mathbf{W} \nabla \mathbf{x}^{\omega} \tag{3.13}$$

$$y_i = \bigvee_{j=1}^{n} \left(w_{ij} + x_j^{\omega} \right)$$

El teorema 2 y el corolario 2.1 de [74] rigen las condiciones que una MMH *min* debe cumplir para obtener una respuesta perfecta.

Teorema 2 [74]**:** $\mathbf{W} \nabla \mathbf{x}^{\omega} = \mathbf{y}^{\omega}$ para todo $\omega = 1,...,k$ si y sólo si para cada índice fila $i = 1,...,m$ existen índices columna $j_i^{\omega} \in \{1,...,n\}$ tales que $w_{ij_i^{\omega}} = y_i^{\omega} - x_{j_i^{\omega}}^{\omega}$ para todo $\omega = 1,...,k$.

Corolario 2.1 [74]**:** $\mathbf{W}\nabla\mathbf{x}^{\omega}=\mathbf{y}^{\omega}$ para todo $\omega=1,...,k$ si y sólo si para cada índice fila $i=1,...,m$ y cada $\gamma\in\{1,...,k\}$ existen índices de columna $j_i^{\gamma}\in\{1,...,n\}$ tales que

$$x_{j_i^{\gamma}}^{\gamma}=\bigvee_{\varepsilon=1}^{k}\left(x_{j_i^{\gamma}}^{\varepsilon}-y_i^{\varepsilon}\right)+y_i^{\gamma} \tag{3.14}$$

Por otra parte, el teorema 3 de [74] indica la cantidad de ruido que es permisible en los patrones de entrada para obtener una respuesta perfecta.

Teorema 3 [74]**:** Para $\gamma=1,...,k$, sea $\tilde{\mathbf{x}}^{\gamma}$ una versión distorsionada del patrón $\mathbf{x}^{\gamma}$. Entonces $\mathbf{W}\nabla\tilde{\mathbf{x}}^{\gamma}=\mathbf{y}^{\gamma}$ si y sólo si se cumple que:

$$\tilde{x}_j^{\gamma}\geq x_j^{\gamma}\vee\bigwedge_{i=1}^{m}\left(\bigvee_{\varepsilon\neq\gamma}\left[y_i^{\gamma}-y_i^{\varepsilon}+x_i^{\varepsilon}\right]\right)\forall j=1,...,n \tag{3.15}$$

y para cada índice renglón $i\in\{1,...,m\}$ existe un índice columna $j_i\in\{1,...,n\}$ tal que:

$$\tilde{x}_{ji}^{\gamma}=x_{ji}^{\gamma}\vee\left(\bigvee_{\varepsilon\neq\gamma}\left[y_i^{\gamma}-y_i^{\varepsilon}+x_{ji}^{\varepsilon}\right]\right) \tag{3.16}$$

3.1.2 Memorias Morfológicas Autoasociativas

Una MAM es de carácter Autoasociativo si se cumple que $\mathbf{x}^{\mu}=\mathbf{y}^{\mu}\forall\mu\in\{1,2,...,k\}$. Existen dos tipos de Memorias Morfológicas Autoasociativas (MMAA): *max,* simbolizadas con **M,** y *min*, simbolizadas con **W**.

Para una MMAA el conjunto fundamental de asociaciones queda definido como: $(\mathbf{x}^{\mu},\mathbf{x}^{\mu})$.

3.1.2.1 Memorias Morfológicas Autoasociativas *max*

Las MMAA *max* (**M**) son aquéllas que utilizan el producto mínimo y el operador máximo en su fase de aprendizaje y el producto mínimo en su fase de recuperación.

Fase de aprendizaje:

1. Se calculan las matrices $\mathbf{x}^{\mu}\Delta\left(-\mathbf{x}^{\mu}\right)^{t}$ para cada uno de los *k* elementos del conjunto fundamental de asociaciones $(\mathbf{x}^{\mu},\mathbf{x}^{\mu})$.

$$\mathbf{x}^{\mu}\Delta\left(-\mathbf{x}^{\mu}\right)^{t}=\begin{pmatrix}\left[x_1^{\mu}+\left(-x_1^{\mu}\right)\right] & \cdots & \left[x_1^{\mu}+\left(-x_j^{\mu}\right)\right] & \cdots & \left[x_1^{\mu}+\left(-x_n^{\mu}\right)\right]\\ \left[x_2^{\mu}+\left(-x_1^{\mu}\right)\right] & \cdots & \left[x_2^{\mu}+\left(-x_j^{\mu}\right)\right] & \cdots & \left[x_2^{\mu}+\left(-x_n^{\mu}\right)\right]\\ \vdots & & \vdots & & \vdots\\ \left[x_i^{\mu}+\left(-x_1^{\mu}\right)\right] & \cdots & \left[x_i^{\mu}+\left(-x_j^{\mu}\right)\right] & \cdots & \left[x_i^{\mu}+\left(-x_n^{\mu}\right)\right]\\ \vdots & & \vdots & & \vdots\\ \left[x_n^{\mu}+\left(-x_1^{\mu}\right)\right] & \cdots & \left[x_n^{\mu}+\left(-x_j^{\mu}\right)\right] & \cdots & \left[x_n^{\mu}+\left(-x_n^{\mu}\right)\right]\end{pmatrix} \tag{3.17}$$

2. Se obtiene la memoria **M** aplicando el operador máximo $\vee$ a las matrices resultantes del paso 1. **M** queda expresada como:

$$\mathbf{M} = \bigvee_{\mu=1}^{k}\left[\mathbf{x}^{\mu}\Delta\left(-\mathbf{x}^{\mu}\right)^{t}\right] = \left[m_{ij}\right]_{n\times n} \qquad \textbf{(3.18)}$$
$$m_{ij} = \bigvee_{\mu=1}^{k}\left(x_i^{\mu} - x_j^{\mu}\right)$$

Fase de recuperación:

1. Se calcula el producto mínimo $\mathbf{M}\Delta\mathbf{x}^{\omega}$, donde $\omega \in \{1,2,...,k\}$, obteniendo un vector columna $\mathbf{x} = \left[x_i\right]_n$:

$$\mathbf{x} = \mathbf{M}\Delta\mathbf{x}^{\omega} \qquad \textbf{(3.19)}$$
$$x_i = \bigwedge_{j=1}^{n}\left(m_{ij} + x_j^{\omega}\right)$$

El teorema 1 y el corolario 1.1 en [104] rigen las condiciones que una MMAA *max* debe cumplir para efectuar los procesos de aprendizaje del conjunto fundamental de asociaciones y recuperar los patrones de entrada cuando son presentados sin ruido.

Teorema 1 [104]**:** **M** es perfecta si y sólo si para cada $\mu = 1,2,...,k$, cada renglón de la matriz $\left[\mathbf{x}^{\mu}\Delta\left(-\mathbf{x}^{\mu}\right)^{t}\right] - \mathbf{M}$ contiene una entrada cero.

Corolario 1.1 [104]**:** $\mathbf{M}\Delta\mathbf{x}^{\omega} = \mathbf{x}^{\omega}$ para todo $\omega = 1,...,k$ si y sólo si para cada índice fila $i = 1,...,n$ y cada $\gamma \in \{1,...,k\}$ existen índices de columna $j_i^{\gamma} \in \{1,...,n\}$ tales que

$$x_{j_i^{\gamma}}^{\gamma} = \bigwedge_{\varepsilon=1}^{k}\left(x_{j_i^{\gamma}}^{\varepsilon} - x_i^{\varepsilon}\right) + x_i^{\gamma} \qquad \textbf{(3.20)}$$

El teorema 2 en [104] muestra que una MMAA *max* tiene capacidad ilimitada y una recuperación perfecta de los patrones de entrada sin ruido.

Teorema 2 [104]**:** Si **M** es una MMAA *max*, entonces recupera de manera perfecta el conjunto fundamental de asociaciones; es decir, se cumple que $\mathbf{M}\Delta\mathbf{x}^{\omega} = \mathbf{x}^{\omega}$ pata todo $\omega = 1,...,k$.

En el caso que los patrones estén corrompidos por ruido una MMAA *max* no pierde la propiedad de respuesta perfecta, para cumplir esta propiedad el ruido permisible debe cumplir con el teorema 3 [104].

Teorema 3 [104]**:** Para $\gamma = 1,...,k$ sea $\tilde{\mathbf{x}}^{\gamma}$ una versión distorsionada del patrón $\mathbf{x}^{\gamma}$. Entonces $\mathbf{M}\Delta\tilde{\mathbf{x}}^{\gamma} = \mathbf{x}^{\gamma}$ si y sólo si se cumple que:

$$\tilde{x}_j^{\gamma} \geq x_j^{\gamma} \wedge \bigvee_{i=1}^{n}\left(\bigwedge_{\varepsilon\neq\gamma}\left[x_i^{\gamma} - x_i^{\varepsilon} + x_i^{\varepsilon}\right]\right) = x_j^{\gamma} \wedge \bigvee_{i=1}^{n}\left(\bigwedge_{\varepsilon\neq\gamma} x_i^{\gamma}\right) \forall j = 1,...,n \qquad \textbf{(3.21)}$$

y para cada índice renglón $i \in \{1,...,n\}$ existe un índice columna $j_i \in \{1,...,n\}$ tal que:

$$\tilde{x}_{ji}^{\gamma} = x_{ji}^{\gamma} \wedge \left(\bigwedge_{\varepsilon\neq\gamma}\left[x_i^{\gamma} - x_i^{\varepsilon} + x_{ji}^{\varepsilon}\right]\right) \qquad \textbf{(3.22)}$$

3.1.2.2 Memorias Morfológicas Autoasociativas *min*

Las MMAA *min* (**W**) son aquellas que utilizan el producto máximo y el operador mínimo en su fase de aprendizaje y el producto máximo en su fase de recuperación.

Fase de aprendizaje:

1. Se calculan las matrices $\mathbf{x}^{\mu} \nabla \left(-\mathbf{x}^{\mu}\right)^{t}$ para cada uno de los *k* elementos del conjunto fundamental de asociaciones $\left(\mathbf{x}^{\mu}, \mathbf{x}^{\mu}\right)$.

$$\mathbf{x}^{\mu} \nabla \left(-\mathbf{x}^{\mu}\right)^{t} = \begin{pmatrix} \left[x_1^{\mu} + \left(-x_1^{\mu}\right)\right] & \cdots & \left[x_1^{\mu} + \left(-x_j^{\mu}\right)\right] & \cdots & \left[x_1^{\mu} + \left(-x_n^{\mu}\right)\right] \\ \left[x_2^{\mu} + \left(-x_1^{\mu}\right)\right] & \cdots & \left[x_2^{\mu} + \left(-x_j^{\mu}\right)\right] & \cdots & \left[x_2^{\mu} + \left(-x_n^{\mu}\right)\right] \\ \vdots & & \vdots & & \vdots \\ \left[x_i^{\mu} + \left(-x_1^{\mu}\right)\right] & \cdots & \left[x_i^{\mu} + \left(-x_j^{\mu}\right)\right] & \cdots & \left[x_i^{\mu} + \left(-x_n^{\mu}\right)\right] \\ \vdots & & \vdots & & \vdots \\ \left[x_n^{\mu} + \left(-x_1^{\mu}\right)\right] & \cdots & \left[x_n^{\mu} + \left(-x_j^{\mu}\right)\right] & \cdots & \left[x_n^{\mu} + \left(-x_n^{\mu}\right)\right] \end{pmatrix} \quad \textbf{(3.23)}$$

2. Se obtiene la memoria **W** aplicando el operador mínimo $\bigwedge$ a las matrices resultantes del paso 1. **W** queda expresada como:

$$\mathbf{W} = \bigwedge_{\mu=1}^{k} \left[\mathbf{x}^{\mu} \nabla \left(-\mathbf{x}^{\mu}\right)^{t}\right] = \left[w_{ij}\right]_{n \times n} \quad \textbf{(3.24)}$$

$$w_{ij} = \bigwedge_{\mu=1}^{k} \left(x_i^{\mu} - x_j^{\mu}\right)$$

Fase de recuperación:

1. Se calcula el producto máximo $\mathbf{W} \nabla \mathbf{x}^{\omega}$, donde $\omega \in \{1, 2, ..., k\}$, obteniendo un vector columna $\mathbf{x} = \left[x_i\right]_n$:

$$\mathbf{x} = \mathbf{W} \nabla \mathbf{x}^{\omega} \quad \textbf{(3.25)}$$

$$x_i = \bigvee_{j=1}^{n} \left(w_{ij} + x_j^{\omega}\right)$$

El teorema 1 y el corolario 1.1 en [105] rigen las condiciones que una MMAA *min* debe cumplir para efectuar los procesos de aprendizaje del conjunto fundamental de asociaciones y recuperar los patrones de entrada cuando son presentados sin ruido.

Teorema 1 [105]**: W** es perfecta si y sólo si para cada $\mu = 1, 2, ..., k$, cada renglón de la matriz $\mathbf{W} - \left[\mathbf{x}^{\mu} \nabla \left(-\mathbf{x}^{\mu}\right)^{t}\right]$ contiene una entrada cero.

Corolario 1.1 [105]**:** $\mathbf{W} \nabla \mathbf{x}^{\omega} = \mathbf{x}^{\omega}$ para todo $\omega = 1, ..., k$ si y sólo si para cada índice fila $i = 1, ..., n$ y cada $\gamma \in \{1, ..., k\}$ existen índices de columna $j_i^{\gamma} \in \{1, ..., n\}$ tales que

$$x_{j_i^{\gamma}}^{\gamma} = \bigvee_{\varepsilon=1}^{k} \left(x_{j_i^{\gamma}}^{\varepsilon} - x_i^{\varepsilon}\right) + x_i^{\gamma} \quad \textbf{(3.26)}$$

El teorema 2 en [105] muestra que una MMAA *min* tiene capacidad ilimitada y una recuperación perfecta de los patrones de entrada sin ruido.

Teorema 2 [105]**:** Si **W** es una MMAA *min*, entonces recupera de manera perfecta el conjunto fundamental de asociaciones; es decir, se cumple que $\mathbf{W} \nabla \mathbf{x}^{\omega} = \mathbf{x}^{\omega}$ pata todo $\omega = 1,...,k$.

En el caso que los patrones estén corrompidos por ruido, una MMAA *min* no pierde la propiedad de respuesta perfecta; para cumplir esta propiedad el ruido permisible debe cumplir con el teorema 3 [105].

Teorema 3 [105]**:** Para $\gamma = 1,...,k$ sea $\tilde{\mathbf{x}}^{\gamma}$ una versión distorsionada del patrón $\mathbf{x}^{\gamma}$. Entonces $\mathbf{W} \nabla \tilde{\mathbf{x}}^{\gamma} = \mathbf{x}^{\gamma}$ si y sólo si se cumple que:

$$\tilde{x}_j^{\gamma} \geq x_j^{\gamma} \vee \bigwedge_{i=1}^{n} \left(\bigvee_{\varepsilon \neq \gamma} \left[x_i^{\gamma} - x_i^{\varepsilon} + x_i^{\varepsilon} \right] \right) = x_j^{\gamma} \vee \bigwedge_{i=1}^{n} \left(\bigvee_{\varepsilon \neq \gamma} x_i^{\gamma} \right) \forall j = 1,...,n \qquad \textbf{(3.27)}$$

y para cada índice renglón $i \in \{1,...,n\}$ existe un índice columna $j_i \in \{1,...,n\}$ tal que:

$$\tilde{x}_{ji}^{\gamma} = x_{ji}^{\gamma} \vee \left(\bigvee_{\varepsilon \neq \gamma} \left[x_i^{\gamma} - x_i^{\varepsilon} + x_{ji}^{\varepsilon} \right] \right) \qquad \textbf{(3.28)}$$

Capítulo 4

Transformada Morfológica

En este capítulo se describe un nuevo modelo de transformada con base en las MAM, al que llamaremos *Transformada Morfológica* (**TM**) y será aplicada en la etapa de transformación de un compresor de imágenes remplazando a métodos tradicionales como la DCT o la DWT. El algoritmo de la **TM** hace uso de las MAM heteroasociativas; el proceso de aprendizaje se desarrolla entre vectores de entrada predefinidos, llamados *matriz de transformación*, y sub-bloques de la imagen, los cuales representan a los vectores de salida. La matriz de transformación es definida en ambos procesos, compresión y descompresión; de esta forma se puede realizar el proceso inverso de la **TM**. Al aplicar el algoritmo de la **TM**, dependiendo de la matriz de transformación usada, la imagen toma una representación morfológica y las etapas que siguen a la transformación en un compresor de imágenes pueden tomar ventaja de esta nueva representación para desarrollar la compresión de los datos. En comparación con los métodos tradicionales, las principales ventajas que una **TM** ofrece es la velocidad de procesamiento y bajos requerimientos de memoria, demostrando además una alta competitividad en los parámetros de compresión y relación señal a ruido.

4.1 Introducción

La etapa de transformación de los datos en un sistema de codificación de imágenes tiene por finalidad facilitar la compresión de la información en las etapas posteriores. Aunque existe una gran variedad de transformadas que pueden ser usadas en la compresión de imágenes, sólo dos de ellas han alcanzado y mantenido un amplio uso en esta área: la DCT y la DWT.

La **TM** es propuesta como una alternativa a las transformadas tradicionales. El algoritmo de este modelo hace uso de las MAM para generar una representación morfológica de la imagen procesada. Una MAM basa su operación en las operaciones morfológicas de dilatación y erosión, es decir, hacen uso de máximos y mínimos de sumas [74]. Esta característica hace de las MAM un modelo con una alta velocidad de procesamiento, misma propiedad que le hereda a la **TM**.

Las siguientes características hacen de la **TM** atractiva para ser usada en la etapa de transformación de un sistema de compresión de imágenes:

- La representación morfológica de la imagen generada por la **TM** puede facilitar la compresión de la información en las etapas siguientes.
- La **TM** es reversible.
- La **TM** tiene bajos requerimientos de memoria, usa una limitada precisión aritmética y se implementa con pocas operaciones aritméticas básicas.

Por otra parte, como se ha mencionado anteriormente, las MAM han demostrado ser una excelente herramienta en la recuperación de patrones alterados por ruido sustractivo o aditivo [15], [106], [107]; esta característica le permite al proceso inverso de la **TM** suprimir gran cantidad de ruido generado por las etapas que le anteceden.

Considerando cómo está constituido el conjunto fundamental de asociaciones, una MAM puede ser de carácter autoasociativo o heteroasociativo, sección 2.1. Una MAM autoasociativa requiere que el vector de salida sea igual al vector de entrada; este hecho descarta su uso en el algoritmo de la **TM**, ya que la imagen a comprimir tendría que estar disponible en el proceso de descompresión para llevar a cabo el proceso inverso de la **TM**.

Una MAM heteroasociativa, por el contrario, asocia vectores de entrada con vectores de salida diferentes en contenido y en dimensión; considerando esta propiedad, una MAM de este tipo puede ser utilizada en el algoritmo de la **TM**, donde la imagen será seccionada para formar los vectores de salida y los vectores de entrada serán predefinidos como una matriz de transformación, la que estará disponible en ambos procesos, compresión y descompresión, permitiendo implementar la transformación morfológica inversa (**TMI**).

La MAM heteroasociativa utilizada en la **TM** puede ser de tipo *min* o *max*, confiriéndole a la **TM** inmunidad a ruido sustractivo o aditivo respectivamente.

4.2 Conceptos previos

El algoritmo de la **TM** propuesto es aplicado sobre sub-bloques individuales de la imagen y se trata de una transformación sin pérdida de información. Durante este proceso se considera que una imagen puede ser representada por la matriz $\mathbf{A} = \left[a_{ij} \right]_{m \times n}$, donde m y n representan el alto y el ancho de la imagen respectivamente y a representa el valor del ij-ésimo píxel; $a \in \left\{ 0,1,2,...,2^L - 1 \right\}$, donde L es el número de bits necesarios para representar el valor de un píxel.

Al ser **TM** una trasformada que se aplica a bloques de la imagen, es necesario definir los términos *sub-bloque de imagen* y *vector de una imagen*.

Definición 4.1. *Sub-bloque de imagen* (**sb**). Sea $\mathbf{A}=\left[a_{ij}\right]$ una matriz de orden $m\times n$ que representa a una imagen y sea $\mathbf{sb}=\left[sb_{ij}\right]$ una matriz de orden $d\times d$. Se define a la matriz **sb** como un sub-bloque de imagen representada por **A**, si la matriz **sb** es un subconjunto de la matriz **A** y cumple que:

$$sb_{ij}=a_{\delta_i\tau_j} \tag{4.1}$$

donde $i,j=1,2,3,...,d$, $\delta=1,2,3,...,m$, $\tau=1,2,3,...,n$ y $a_{\delta_i\tau_j}$ representa el valor del píxel determinado por la coordenadas $(\delta+i,\tau+j)$, donde (δ,τ) determina el inicio del sub-bloque de imagen y $(\delta+d,\tau+d)$ el final del mismo.

Definición 4.2. *Vector de una imagen* (**vi**). Sea $\mathbf{sb}=\left[sb_{ij}\right]$ un sub-bloque de imagen y sea $\mathbf{vi}=\left[vi_i\right]$ un vector de dimensión *d*. Se define como vector de una imagen a la μ-ésima fila de la matriz **sb** tal que:

$$vi_i=sb_{\mu i} \tag{4.2}$$

donde $i=1,2,3,...,d$.

De cada sub-bloque de la imagen se obtienen μ vectores de una imagen:

$$\mathbf{vi}^{\mu}=[vi_i^{\mu}]=[sb_{\mu i}] \tag{4.3}$$

donde $\mu=1,2,3,...,d$.

La **TM** hace uso de una matriz de transformación, la cual está formada por vectores denominados vectores de transformación. A continuación se definen estos dos términos.

Definición 4.3. *Vector de transformación* (**vt**). Sea $\mathbf{vt}=\left[vt_i\right]$ un vector fila de dimensión *d*. Se define como vector de transformación al vector de entrada **vt** que interviene en los procesos de aprendizaje y recuperación de una MAM y cuya generación está gobernada por el teorema 2 y el corolario 2.1 de [74].

Definición 4.4. *Matriz de transformación* (**mt**). Sea $\mathbf{vt}=\left[vt_i\right]_d$ un vector de transformación. Se define como matriz de transformación $\mathbf{mt}=\left[mt_{ij}\right]_{d\times d}$, al conjunto formado por *d* vectores de transformación $\{\mathbf{vt}^1, \mathbf{vt}^2,\ldots, \mathbf{vt}^d\}$, donde la *i*-ésima fila de la matriz **mt** es representada por el vector $\mathbf{vt}^i$. Entonces el *ij*-ésimo componente de **mt** es definido por:

$$mt_{ij}=vt_j^i \mid i,j=1,2,...,d \tag{4.4}$$

Cuando se hace uso de una **TM** en un proceso de transformación de una imagen, la selección adecuada de una matriz **mt** repercute directamente en los parámetros de relación de compresión y de relación señal a ruido obtenidos al final del proceso de compresión. La **mt** debe ser conocida por los procesos de compresión y descompresión. Gran variedad de valores pueden satisfacer el teorema que rige la generación de la **mt**; una opción es usar el siguiente criterio:

$$vt_j^i=\begin{cases}=0 & j\neq i\\ >e & j=i\end{cases} \tag{4.5}$$

donde *e* es el valor máximo que puede tomar un elemento de la imagen **A**.

Siguiendo este criterio, la matriz de transformación obtenida es una matriz diagonal; considerando $e = 256$ la matriz de transformación **mt** queda constituida de la siguiente manera:

$$\mathbf{mt} = \begin{pmatrix} e & 0 & \cdots & 0 \\ 0 & e & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & e \end{pmatrix} \begin{matrix} \rightarrow & \mathbf{vt}^1 \\ \rightarrow & \mathbf{vt}^2 \\ \vdots & \vdots \\ \rightarrow & \mathbf{vt}^d \end{matrix} \tag{4.6}$$

donde cada fila de la matriz **mt** es un vector de transformación **vt**.

Considerando a los vectores de transformación **vt** como los vectores de entrada y a los vectores de imagen **vi** como los vectores de salida, se puede formar el siguiente conjunto fundamental de asociaciones:

$$\left\{ \left(\mathbf{vt}^{\mu}, \mathbf{vi}^{\mu} \right) \mid \mu = 1,2,...,d \right\}$$

Al aplicar el proceso de aprendizaje (formación de la memoria) de una MMH sobre este conjunto fundamental, la estructura de los *d* vectores de entrada (**vt**) permite la codificación individual de cada uno de los vectores de salida (**vi**) dentro de la MMH. Es claro que en la MMH generada existirán tantas columnas como elementos contenga un vector **vt**; mientras que la dimensión de un vector **vi** determinara el número de filas de la misma memoria. Cada uno de los vectores **vi** quedará codificado en una columna de la MMH.

Al aplicar el producto máximo entre cada elemento del conjunto fundamental de asociaciones, la posición que guarda *e* dentro de los vectores **vt** indicará en cual columna quedara codificado cada vector **vi**. Esto es posible debido a que la operación $vi_i - vt_j$ generará valores negativos en la *j*-ésima columna si $vt_j = e$ y valores positivos en las columnas restantes; cuando se aplica el operador mínimo sobre las matrices obtenidas con la finalidad de generar la MMH, los elementos afectados por *e* son los que formarán a la memoria.

Esta forma en que los vectores **vi** quedan codificados en la MMH garantiza su recuperación perfecta al aplicar la fase de recuperación de una MMH sobre cada uno de los vectores de entrada **vt**.

4.3 Transformada Morfológica utilizando MAM

Podemos ahora definir a la **TM** y su proceso inverso la **TMI**.

Definición 4.5. *Transformada Morfológica* (**TM**). Sea $\mathbf{mt} = \left[mt_{ij} \right]_{d \times d}$ una matriz de transformación, sea $\mathbf{A} = \left[a_{ij} \right]_{m \times n}$ una matriz que representa a una imagen, donde *m* y *n* significan el alto y el ancho de la imagen respectivamente y a_{ij} define el valor del *ij*-ésimo píxel, y sea $\mathbf{sb} = \left[sb_{ij} \right]_{d \times d}$ un sub-bloque de imagen de **A**, obteniendo como resultado un conjunto de sub-bloques de imagen $\mathbf{sb}^{\omega} \mid \omega = 1,2,...,N$, donde $N = \left(m/d \right) \cdot \left(n/d \right)$. Se define como transformada morfológica al conjunto de memorias asociativas resultantes de la transformación aplicada a bloques individuales de una imagen haciendo uso de las MMH, considerando para ello a los vectores de una imagen $\mathbf{vi}^{\mu} = \left[vi_i \right]_d \mid \mu = 1,2,...,d$, los cuales componen al sub-bloque de

imagen, como vectores de salida y a los vectores de transformación $\mathbf{vt}^{\mu} = [vt_i]_d \mid \mu = 1,2,...,d$, que conforman a la matriz de transformación, como los vectores de entrada.

Este proceso genera N MMH las cuales estructuradas en forma matricial representan a la transformada morfológica de la imagen

$$\mathbf{TM}\{\mathbf{A} = O\{\mathbf{sb}^{ij}\}, \mathbf{mt}\} \rightarrow O\{\mathrm{MMH}^{ij}\}$$
$$O\{\mathrm{MMH}^{ij}\} = \begin{pmatrix} \mathrm{MMH}^{11} & \mathrm{MMH}^{12} & \cdots & \mathrm{MMH}^{1\eta} \\ \mathrm{MMH}^{21} & \mathrm{MMH}^{22} & \cdots & \mathrm{MMH}^{2\eta} \\ \vdots & \vdots & & \vdots \\ \mathrm{MMH}^{\lambda 1} & \mathrm{MMH}^{\lambda 2} & \cdots & \mathrm{MMH}^{\lambda\eta} \end{pmatrix} \quad \textbf{(4.7)}$$

donde $\omega = 1,2,...,N$, $i = 1,2,...,\lambda$, $j = 1,2,...,\eta$, $\lambda = m/d$ y $\eta = n/d$; además se define el operador $O\{\cdot\}$ para representar la organización que las MMH deben guardar para constituir a la **TM**; de esta forma MMH^{ij} representa a la memoria generada de aplicar el proceso de aprendizaje de una MMH entre el *ij*-ésimo bloque de imagen y la matriz de transformación.

Anteriormente se discutió el por qué las MAM heteroasociativas pueden ser utilizadas para implementar la **TM**; debido al uso de este tipo de memorias en el algoritmo de la transformada propuesta se definen dos tipos de **TM**: transformada morfológica *min* ($\mathbf{TM}_{\min}$) y transformada morfológica *max* ($\mathbf{TM}_{\max}$). Cuando se usa una MMH *min* para transformar un sub-bloque de imagen de dimensión $d \times d$, la **TM** es definida por:

$$\mathbf{TM}_{\min}\{\mathbf{A}, \mathbf{mt}\} \rightarrow O\{\mathrm{MMH}^{xy}_{\min} \mid x = 1,2,...,\lambda, y = 1,2,...,\eta\}$$
$$\mathrm{MMH}^{xy}_{\min} = \bigwedge_{\mu=1}^{d} [\mathbf{vi}^{\omega_\mu} \nabla (-\mathbf{vt}^{\mu})^t] = [w_{ij}]^{xy}_{d\times d} \mid \omega = 1,2,..., N \quad \textbf{(4.8)}$$
$$[w_{ij}]^{xy}_{d\times d} = \bigwedge_{\mu=1}^{d} \left(vi_i^{\omega_\mu} - vt_j^{\mu}\right) \mid i, j = 1,2,...,d$$

donde ω define a cuál de los N sub-bloques de imagen **sb** pertenecen los vectores de imagen **vi**, es decir, $\mathbf{vi}^{\omega_\mu}$ es la μ-ésima fila del sub-bloque de imagen ω.

Cuando se usa una MMH *max* para transformar un sub-bloque de imagen de dimensión $d \times d$, la **TM** es definida por:

$$\mathbf{TM}_{\max}\{\mathbf{A}, \mathbf{mt}\} \rightarrow O\{\mathrm{MMH}^{xy}_{\max} \mid x = 1,2,...,\lambda, y = 1,2,...,\eta\}$$
$$\mathrm{MMH}^{xy}_{\max} = \bigvee_{\mu=1}^{d} [\mathbf{vi}^{\omega_\mu} \Delta (-\mathbf{vt}^{\mu})^t] = [m_{ij}]^{xy}_{d\times d} \mid \omega = 1,2,..., N \quad \textbf{(4.9)}$$
$$[m_{ij}]^{xy}_{d\times d} = \bigvee_{\mu=1}^{d} \left(vi_i^{\omega_\mu} - vt_j^{\mu}\right) \mid i, j = 1,2,...,d$$

Donde ω define a cuál de los N sub-bloques de imagen **sb** pertenecen los vectores de imagen **vi**.

Como resultado de aplicar una **TM** sobre una imagen, N MMH de dimensión $d \times d$ tipo **W** o **M** son generadas; estas memorias en conjunto forman la imagen transformada. La transformada permite aplicarle un proceso de cuantificación escalar. Recordemos que la **TM** es robusta a ruido sustractivo o aditivo según la MMH que se use en su implementación, minimizando los efectos de este proceso y al mismo tiempo concentrando los valores de la imagen en un menor

número de valores. Esta propiedad es la que hace posible tomar ventaja de esta nueva representación de la imagen en las siguientes etapas para conseguir la compresión de la información.

Definición 4.6. *Transformada Morfológica Inversa* (**TMI**). Sea $\mathbf{mt} = \left[mt_{ij} \right]_{d \times d}$ una matriz de transformación, sea **TM** la transformada morfológica de una imagen, la **TM** está formada por N MMH distribuidas matricialmente: $\mathbf{TM}_{\psi} \{\mathbf{A}, \mathbf{mt}\} \rightarrow O\{\text{MMH}_{\psi}^{xy} \mid x = 1,2,..., \lambda, y = 1,2,..., \eta\}$, donde $\lambda = m/d$, $\eta = n/d$, $N = \lambda \cdot \eta$ y ψ indica si las MMH son del tipo *max* o *min*. Se define como transformada morfológica inversa al proceso que aplica la fase de recuperación de una MMH entre los vectores de transformación $\mathbf{vt}^{\mu} = \left[vt_i \right]_d \mid \mu = 1,2,..., d$, los cuales conforman a la matriz de transformación y que son considerados como los vectores de entrada, y cada una de las memorias MMH que forman a la **TM**.

Como resultado del proceso de **TMI** N sub-bloques de imagen **sb** son generados, los cuales en conjunto representan a la imagen original transformada por la **TM**.

$$\mathbf{TMI}\{\text{MMH}_{\psi}^{xy}, \mathbf{mt}\} \rightarrow O\{\mathbf{sb}^{ij}\}$$

$$\mathbf{TMI} = O\{\mathbf{sb}^{ij}\} = \begin{pmatrix} \mathbf{sb}^{11} & \mathbf{sb}^{12} & \cdots & \mathbf{sb}^{1\eta} \\ \mathbf{sb}^{21} & \mathbf{sb}^{22} & \cdots & \mathbf{sb}^{2\eta} \\ \vdots & \vdots & & \vdots \\ \mathbf{sb}^{\lambda 1} & \mathbf{sb}^{\lambda 2} & \cdots & \mathbf{sb}^{\lambda\eta} \end{pmatrix} \tag{4.10}$$

donde $i = 1,2,..., \lambda$, $j = 1,2,..., \eta$, $\lambda = m/d$ y $\eta = n/d$. Se hace uso del operador $O\{\}$ debido a que las matrices **sb** guardan la misma posición, dentro de la **TMI**, que la MAM utilizada en su recuperación guarda dentro de la **TM**.

Una transformación morfológica inversa (**TMI**) es posible debido a que:

- La imagen transformada es un conjunto de MMH.
- La matriz de transformación está disponible para el proceso de descompresión.

Para un proceso **TMI** pueden ser definidos 2 casos:

Caso 1. Cuando la **TM** no ha sido alterada por ruido. En este caso se habla de un proceso reversible, es decir, la imagen original es recuperada. Sin embargo, la relación de compresión alcanzada por el proceso completo de compresión no resulta significativa.

Al existir dos tipos de transformadas, la $\mathbf{TM}_{\min}$ y la $\mathbf{TM}_{\max}$, entonces es necesario definir una **TMI** para cada una de ellas.

Se denominará como $\mathbf{TMI}_{\min}$ cuando sea necesario recuperar una imagen transformada por una $\mathbf{TM}_{\min}$, quedando definida por:

$$\begin{aligned} \mathbf{TMI}_{\min} = \mathbf{TMI}\{\text{MMH}_{\min}^{xy}, \mathbf{mt}\} &\rightarrow O\{\ \mathbf{sb}_{\min}^{xy} \mid x = 1,2,..., \lambda,\ y = 1,2,..., \eta\} \\ \mathbf{sb}_{\min}^{xy} &= \mathbf{vi}^{(xy)_{\mu}} \mid \mu = 1,2,..., d \\ \mathbf{vi}^{(xy)_{\mu}} &= \text{MMH}_{\min}^{xy} \nabla \mathbf{vt}^{\mu} = \left[vi_i^{(xy)_{\mu}} \right]_d \\ vi_i^{(xy)_{\mu}} &= \bigvee_{j=1}^{d} \left(w_{ij}^{xy} + vt_j^{\mu} \right) \end{aligned} \tag{4.11}$$

donde *xy* define a cuál de los *N* sub-bloques de imagen **sb** pertenecen los vectores de imagen **vi**, es decir, $\mathbf{vi}^{(xy)_\mu}$ representa la μ -ésima fila del sub-bloque de imagen denotado por *xy*.

Se denominará como $\mathbf{TMI}_{\max}$ cuando sea necesario recuperar una imagen transformada por una $\mathbf{TM}_{\max}$, quedando definida por:

$$\begin{aligned} \mathbf{TMI}_{\max} = \mathbf{TMI}\{\mathrm{MMH}^{xy}_{\max}, \mathbf{mt}\} \rightarrow O\{\ \mathbf{sb}^{xy}_{\max} \mid x = 1,2,...,\lambda\ ,\ y = 1,2,...,\eta\} \\ \mathbf{sb}^{xy}_{\max} = \mathbf{vi}^{(xy)_\mu} \mid \mu = 1,2,...,d \\ \mathbf{vi}^{(xy)_\mu} = \mathrm{MMH}^{xy}_{\max} \Delta\, \mathbf{vt}^{\mu} = \left[vi_i^{(xy)_\mu} \right]_d \\ vi_i^{(xy)_\mu} = \bigwedge_{j=1}^{d} \left(m_{ij}^{xy} + vt_j^{\mu} \right) \end{aligned} \tag{4.12}$$

donde *xy* define a cuál de los *N* sub-bloques de imagen **sb** pertenecen los vectores de imagen **vi**.

Caso 2. Cuando la **TM** ha sido alterada por ruido mediante un proceso de cuantificación. En este caso se habla de un proceso irreversible, es decir, la imagen recuperada es una versión alterada de la imagen original. Sin embargo, la relación de compresión alcanzada por el proceso completo de compresión resulta significativa.

En un sistema de compresión de imágenes, la etapa que sigue al transformador es la cuantificación, se trata de un proceso irreversible que modifica la información resultante de la **TM**. La **TM** está formada por un conjunto de MMH y la teoría expuesta sobre las MAM en [74] y [72] sólo considera el reconocimiento de patrones en el caso que el ruido esté añadido a los patrones de entrada. Si el proceso de cuantificación altera la información de las MMH que forman a la imagen trasformada por la **TM**, entonces, ¿cómo afecta esta modificación a las memorias asociativas en su proceso de recuperación de los patrones de salida originales (bloques de la imagen original)?

Para responder a esta pregunta se ha formulado un nuevo teorema referente a las MAM.

Teorema 4.1. Si $\hat{\mathbf{W}}$ denota la versión distorsionada de una memoria asociativa **W**

$$\begin{aligned} \hat{\mathbf{W}} = \mathbf{W} \pm r \\ \hat{w}_{ij} = w_{ij} \pm r \end{aligned} \tag{4.13}$$

donde *r* representa el ruido inducido a la memoria por la etapa de cuantificación.

Entonces, el proceso de recuperación queda definido por:

$$\hat{\mathbf{W}} \nabla \mathbf{x}^{\gamma} = \hat{\mathbf{y}}^{\gamma} = y_i^{\gamma} \pm r \tag{4.14}$$

Similarmente, si $\hat{\mathbf{M}}$ denota la versión distorsionada de una memoria asociativa **M**

$$\begin{aligned} \hat{\mathbf{M}} = \mathbf{M} \pm r \\ \hat{m}_{ij} = m_{ij} \pm r \end{aligned} \tag{4.15}$$

donde *r* representa el ruido inducido a la memoria por la etapa de cuantificación.

Entonces, el proceso de recuperación queda definido por:

$$\hat{\mathbf{M}} \Delta \mathbf{x}^{\gamma} = \hat{\mathbf{y}}^{\gamma} = y_i^{\gamma} \pm r \tag{4.16}$$

Demostración. Partiendo del teorema 2 de [74], el cual enuncia que $\mathbf{y}^{\gamma} = \mathbf{W} \nabla \mathbf{x}^{\gamma}$, se sustituye a **W** por su versión distorsionada:

$$\mathbf{y}^{\gamma} = \hat{\mathbf{W}} \nabla \mathbf{x}^{\gamma}$$
$$y_i^{\gamma} = \bigvee_{j=1}^{n} \left(\hat{w}_{ij} + x_j^{\gamma} \right) \tag{4.17}$$

donde, $\gamma = 1,2,...,k$, $i = 1,2,...,m$ y $j = 1,2,...,n$. Esta expresión resulta en:

$$y_i^{\gamma} \geq \hat{w}_{ij} + x_j^{\gamma} \tag{4.18}$$

Del corolario 2.1 de [74] se desprende que:

$$x_j^{\gamma} = \bigvee_{\varepsilon=1}^{k} \left(x_j^{\varepsilon} - y_i^{\varepsilon} \right) + y_i^{\gamma} \tag{4.19}$$

donde $\varepsilon \in \{1,2,...,k\}$. Sustituyendo la expresión 4.19 en la expresión 4.18:

$$y_i^{\gamma} = \hat{w}_{ij} + \left(\bigvee_{\varepsilon=1}^{k} \left(x_j^{\varepsilon} - y_i^{\varepsilon} \right) + y_i^{\gamma} \right) \tag{4.20}$$

Haciendo uso de la propiedad de conjugada aditiva, donde $\bigvee_{\varepsilon=1}^{k} \left(x_j^{\varepsilon} - y_i^{\varepsilon} \right) = - \bigwedge_{\varepsilon=1}^{k} \left(y_j^{\varepsilon} - x_i^{\varepsilon} \right)$, entonces la nueva representación es:

$$y_i^{\gamma} = \hat{w}_{ij} + y_i^{\gamma} - \bigwedge_{\varepsilon=1}^{k} \left(y_i^{\varepsilon} - x_j^{\varepsilon} \right) \tag{4.21}$$

donde $w_{ij} = \bigwedge_{\varepsilon=1}^{k} \left(y_i^{\varepsilon} - x_j^{\varepsilon} \right)$, entonces:

$$y_i^{\gamma} = \hat{w}_{ij} + y_i^{\gamma} - w_{ij} \tag{4.22}$$

Finalmente sustituyendo la expresión 4.13 en la expresión 4.22 y simplificando:

$$y_i^{\gamma} = w_{ij} \pm r + y_i^{\gamma} - w_{ij} \tag{4.23}$$
$$y_i^{\gamma} = y_i^{\gamma} \pm r$$

Para el caso de una MMH **M** se sigue la misma secuencia usada en la demostración del teorema que emplea una MMH **W**.

$$
\begin{aligned}
\mathbf{y}^{\gamma} &= \hat{\mathbf{M}} \Delta \mathbf{x}^{\gamma} \\
y_i^{\gamma} &= \bigwedge_{j=1}^{n} \left(\hat{m}_{ij} + x_j^{\gamma} \right) \\
&\leq \hat{m}_{ij} + x_j^{\gamma} \\
&= \hat{m}_{ij} + \left(\bigwedge_{\varepsilon=1}^{k} \left(x_j^{\varepsilon} - y_i^{\varepsilon} \right) + y_i^{\gamma} \right) \\
&= \hat{m}_{ij} + y_i^{\gamma} - \bigvee_{\varepsilon=1}^{k} \left(y_i^{\varepsilon} - x_j^{\varepsilon} \right) \\
&= \hat{m}_{ij} + y_i^{\gamma} - m_{ij} \\
&= m_{ij} \pm r + y_i^{\gamma} - m_{ij} \\
&= y_i^{\gamma} \pm r
\end{aligned}
\tag{4.24}
$$

El **Teorema 4.1** muestra que el ruido *r* asociado a la memoria asociativa afecta directamente al patrón de salida recuperado y a la propiedad de la **TM** para obtener una recuperación perfecta de la imagen. La naturaleza de *r* depende directamente del tipo de cuantificación utilizada.

Las expresiones 4.11 y 4.12, las cuales definen a la $\mathbf{TMI}_{\min}$ y la $\mathbf{TMI}_{\max}$ para el caso que la **TM** no ha sido alterada por ruido, son re-escritas con base en el **Teorema 4.1** para definir el caso donde la **TM** ha sido alterada por ruido por el proceso de cuantificación

$$
\begin{aligned}
\mathbf{TMI}_{\min} = \mathbf{TMI}\{\mathrm{MMH}_{\min}^{xy}, \mathbf{mt}\} \rightarrow O\{ \mathbf{sb}_{\min}^{xy} \mid x = 1,2,...,\lambda,\ y = 1,2,...,\eta \} \\
\mathbf{sb}_{\min}^{xy} = \mathbf{vi}^{(xy)_\mu} \mid \mu = 1,2,...,d \\
\mathbf{vi}^{(xy)_\mu} = \mathrm{MMH}_{\min}^{xy} \nabla \mathbf{vt}^{\mu} = \left[vi_i^{(xy)_\mu} \right]_d \\
vi_i^{(xy)_\mu} = \bigvee_{j=1}^{d} \left(w_{ij}^{xy} + vt_j^{\mu} \pm r \right)
\end{aligned}
\tag{4.25}
$$

$$
\begin{aligned}
\mathbf{TMI}_{\max} = \mathbf{TMI}\{\mathrm{MMH}_{\max}^{xy}, \mathbf{mt}\} \rightarrow O\{ \mathbf{sb}_{\max}^{xy} \mid x = 1,2,...,\lambda,\ y = 1,2,...,\eta \} \\
\mathbf{sb}_{\max}^{xy} = \mathbf{vi}^{(xy)_\mu} \mid \mu = 1,2,...,d \\
\mathbf{vi}^{(xy)_\mu} = \mathrm{MMH}_{\max}^{xy} \Delta \mathbf{vt}^{\mu} = \left[vi_i^{(xy)_\mu} \right]_d \\
vi_i^{(xy)_\mu} = \bigwedge_{j=1}^{d} \left(m_{ij}^{xy} + vt_j^{\mu} \pm r \right)
\end{aligned}
\tag{4.26}
$$

donde *xy* define a cuál de los *N* sub-bloques de imagen **sb** pertenecen los vectores de imagen **vi**, es decir, $\mathbf{vi}^{(xy)_\mu}$ representa la μ-ésima fila del sub-bloque de imagen denotado por *xy*.

4.4 Algoritmo de la Transformada Morfológica

En esta sección se describen con detalle los pasos que forman parte de una **TM** aplicada a una imagen y de su proceso inverso, la **TMI**. También se detalla la implementación de dichos algoritmos utilizando un lenguaje de alto nivel y un análisis de la complejidad de los mismos.

4.4.1 Transformada Morfológica

El proceso de transformación morfológica de una imagen puede ser resumido en los siguientes pasos:

1. Al tratarse de una transformada aplicada a bloques individuales de la imagen, ésta es seccionada en bloques de dimensión $d \times d$. Considerando a una imagen como una matriz $\mathbf{A} = [a_{ij}]$ de orden $m \times n$, donde m y n representan el alto y el ancho de la imagen respectivamente y a representa el valor del ij-ésimo píxel; $a \in \{0,1,2,...,2^L - 1\}$, donde L es el número de bits necesarios para representar el valor de un píxel. La imagen es dividida en $N = (m/d) \cdot (n/d)$ sub-bloques de imagen $\mathbf{sb}^{\omega} | \omega = 1,2,..., N$.
2. Se inicializa la **TM**.
3. Se define la matriz de transformación $\mathbf{mt} = [mt_{ij}]_{d \times d}$ a utilizar en el proceso de **TM**.
4. Se aplica la **TM** sobre cada sub-bloque de imagen, haciendo uso de la matriz de transformación definida en el paso anterior.

   $\mathrm{MMH}_{\min}^{xy} = \bigwedge_{\mu=1}^{d} [\mathbf{vi}^{\omega_\mu} \nabla (\text{-}\mathbf{vt}^{\mu})^t] = [w_{ij}]_{d \times d}^{xy} | \omega = 1,2,..., N$; para una $\mathbf{TM}_{\min}$.

   $\mathrm{MMH}_{\max}^{xy} = \bigvee_{\mu=1}^{d} [\mathbf{vi}^{\omega_\mu} \Delta (\text{-}\mathbf{vt}^{\mu})^t] = [m_{ij}]_{d \times d}^{xy} | \omega = 1,2,..., N$; para una $\mathbf{TM}_{\max}$.

   donde, el vector de imagen $\mathbf{vi}^{\omega_\mu} \in \mathbf{sb}^{\omega}$ y el vector de transformación $\mathbf{vt}^{\mu} \in \mathbf{mt}$; $\mu = 1,2,..., d$.
5. Como resultado se obtiene N MMH de dimensión $d \times d$, las cuales en conjunto forman a la imagen transformada mediante una **TM**.

   $\mathbf{TM}_{\min} \{\mathbf{A}, \mathbf{mt}\} \rightarrow O\{\mathrm{MMH}_{\min}^{xy}\}$.

   $\mathbf{TM}_{\max} \{\mathbf{A}, \mathbf{mt}\} \rightarrow O\{\mathrm{MMH}_{\max}^{xy}\}$.

   donde $x = 1,2,..., \lambda$, $y = 1,2,..., \eta$; $\lambda = m/d$ y $\eta = n/d$.

La figura 4.1 muestra un esquema general de la **TM**.

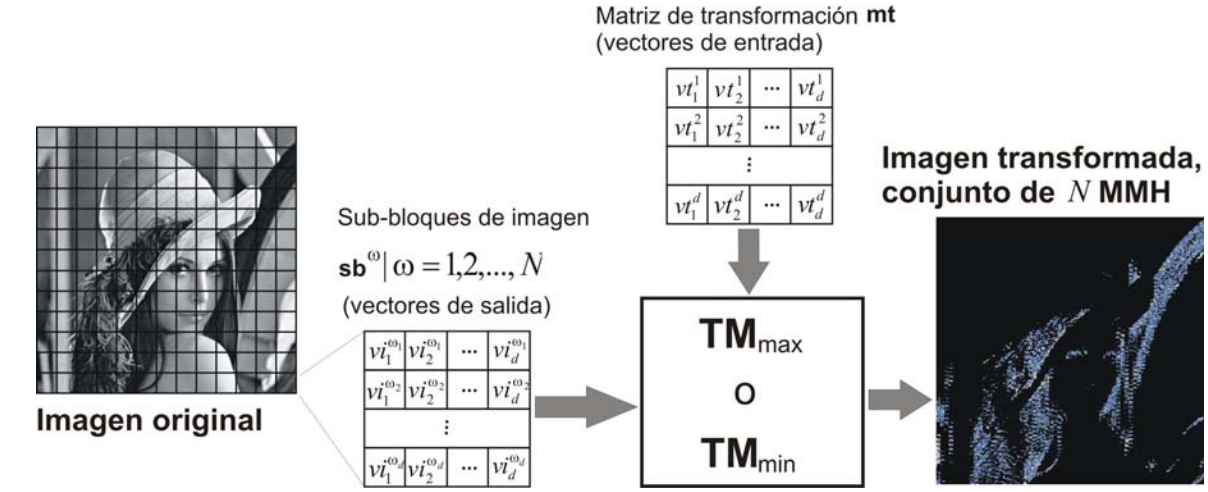


Figura 4.1. Esquema de la **TM**.

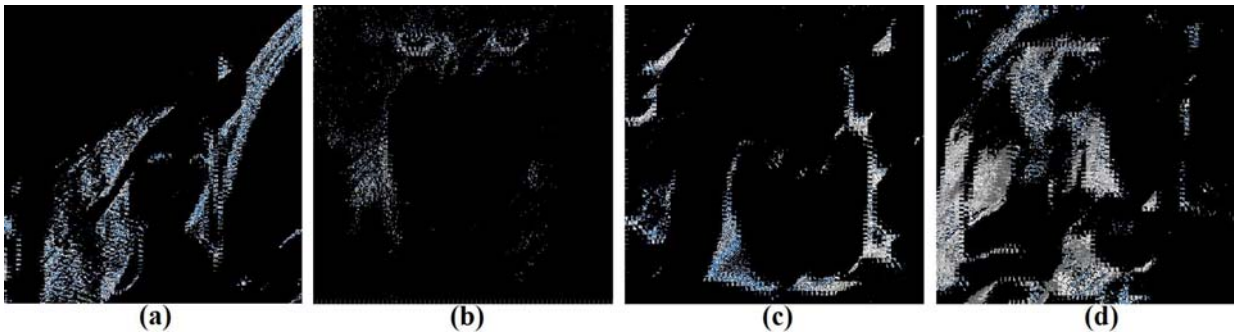


Figura 4.2. Resultados de aplicar la **TM** sobre las imágenes: (a) Lena, (b) Mandril (*Baboon*), (c) Pimientos (*Peppers*), (d) Hombre (*Man*).

La figura 4.2 muestra los resultados de aplicar la **TM** sobre imágenes en escala de grises de 512×512 píxeles: 8 bits/píxel. La tabla 4.1 muestra el seudo-código del algoritmo de una **TM** y las tablas 4.2 y 4.3 muestran el seudo-código de los algoritmos para el cálculo de una MMH *min* y MMH *max* sobre un bloque de imagen de dimensión $d \times d$ respectivamente.

Tabla 4.1. Seudo-código del algoritmo de una **TM**.

```
01| variables  // Definición de variables y constantes globales
02|   alto_i,ancho_i: entero
03|   A[alto_i][ ancho_i], TM[alto_i][ ancho_i]: entero
04|   m[d][d], w[d][d], sb[d][d], vi[dxd]: entero
05| constantes
06|   vt[dxd]: entero
07|   d=8: entero
08| programa_principal()
09| variables
10|    λ,η,l,y,x: entero
11| inicio
12|    // Barrido de la imagen en bloques de  d × d
13|    para (λ=0 mientras_que λ<alto_i [operaciones λ=λ+d]) hacer
14|      para (η =0 mientras_que η< ancho_i [operaciones η=η+d]) hacer
15|          // Selección del bloque de imagen a procesar e inicialización de la MMH
16|          l=0;
17|          para (y=0 mientras_que y<d [operaciones y=y+1) hacer
18|              para (x=0 mientras_que x<d [operaciones x=x+1]) hacer
19|                vi[l]=A[η +x][ λ +y]; // Selección del bloque de imagen a procesar
20|                l=l+1;
21|                w[h][k]=2^L;      // m[h][k]=-2^L para una TM_max.. Inicialización de la MMH
22|              fin_para
23|          fin_para
24|          subrutina P_min()   // P_max() para una TM_max
25|          // Almacena la MMH obtenida, formando a la TM
26|          para (y=0 mientras_que y<d [operaciones y=y+1) hacer
27|              para (x=0 mientras_que x<d [operaciones x=x+1]) hacer
28|                 TM[η +x][ λ +y]=w[x][y];  // m[x][y] para una TM_max
29|              fin_para
30|          fin_para
31|       fin_para
32|    fin_para
33| fin_programa_principal
```

Tabla 4.2. Seudo-código del algoritmo del cálculo de una MMH *min*.

```
01| subrutina P_min()
02| variables
03|   y,x,l,aux: entero
04| inicio
05|   para (l=0 mientras_que l<(d)(d) [operaciones l=l+d]) hacer
06|     para (y=0 mientras_que y<d [operaciones y=y+1]) hacer
07|       para (x =0 mientras_que x<d [operaciones x=x+1]) hacer
08|         aux=vi[l+y]-vt[l+x]:
09|         si (aux<w[x][y]) entonces
10|           w[x][y]=aux;
11|         fin_si
12|       fin_para
13|     fin_para
14|   fin_para
15| fin_subrutina
```

Tabla 4.3. Seudo-código del algoritmo del cálculo de una MMH *max*.

```
01| subrutina P_max()
02| variables
03|   y,x,l,aux: entero
04| inicio
05|   para (l=0 mientras_que l<(d)(d) [operaciones l=l+d]) hacer
06|     para (y=0 mientras_que y<d [operaciones y=y+1]) hacer
07|       para (x =0 mientras_que x<d [operaciones x=x+1]) hacer
08|         aux=vi[l+y]-vt[l+x];
09|         si (aux>m[x][y]) entonces
10|           m[x][y]=aux;
11|         fin_si
12|       fin_para
13|     fin_para
14|   fin_para
15| fin_subrutina
```

4.4.2 Transformada Morfológica Inversa

El proceso inverso de una transformación morfológica de una imagen puede ser resumido en los siguientes pasos:

1. La **TM** está formada por N MMH organizadas en forma matricial, $\mathbf{TM}_{\psi}\{\mathbf{A},\mathbf{mt}\} \rightarrow O\{\mathrm{MMH}_{\psi}^{xy} \mid x = 1,2,...,\lambda, y = 1,2,...,\eta\}$, donde $\lambda = m/d$, $\eta = n/d$, $N = \lambda \cdot \eta$ y ψ indica si las MMH son del tipo *max* o *min*; considerando esto, el primer paso es aislar a cada una de estas memorias que forman a la **TM**.
2. Se hace uso de la matriz de transformación $\mathbf{mt} = [mt_{ij}]_{d\times d}$ definida en el proceso de **TM**.

3. Se aplica el proceso de la **TMI** entre cada una de las MMH que forman a la **TM** de la imagen y la matriz de transformación, obteniendo d vectores de imagen $\mathbf{vi}^{\mu} = \left[vi_i \right]_d \mid \mu = 1,2,...,d$

$$vi_i^{(xy)_\mu} = \bigvee_{j=1}^{d} \left(w_{ij}^{xy} + vt_j^{\mu} \right); \text{ para una } \mathbf{TM}_{\min}.$$

$$vi_i^{(xy)_\mu} = \bigwedge_{j=1}^{d} \left(m_{ij}^{xy} + vt_j^{\mu} \right); \text{ para una } \mathbf{TM}_{\max}.$$

4. Al agrupar estos vectores de imagen se forman los sub-bloques de imagen $\mathbf{sb} = \left[sb_{ij} \right]_{d \times d}$

$$\mathbf{sb}^{xy} = \mathbf{vi}^{(xy)_\mu} \mid \mu = 1,2,...,d \text{ ; para una } \mathbf{TM}_{\min}.$$

$$\mathbf{sb}^{xy} = \mathbf{vi}^{(xy)_\mu} \mid \mu = 1,2,...,d \text{ ; para una } \mathbf{TM}_{\max}.$$

donde xy define cuál de los N sub-bloques de imagen se ha recuperado.

5. Finalmente la agrupación en forma matricial de los sub-bloques de imagen constituyen la imagen recuperada

$$\textbf{Imagen recuperada} = O\left\{ \mathbf{sb}^{xy} \mid x = 1,2,...,\lambda,\ y = 1,2,...,\eta \right\}.$$

El esquema de una **TMI** es mostrado en la figura 4.3.

La tabla 4.4 muestra el seudo-código del algoritmo de una **TMI** y las tablas 4.5 y 4.6 muestran el seudo-código de los algoritmos para la recuperación de los sub-bloques de imagen para una $\mathbf{TM}_{\min}$ y una $\mathbf{TM}_{\max}$ respectivamente.

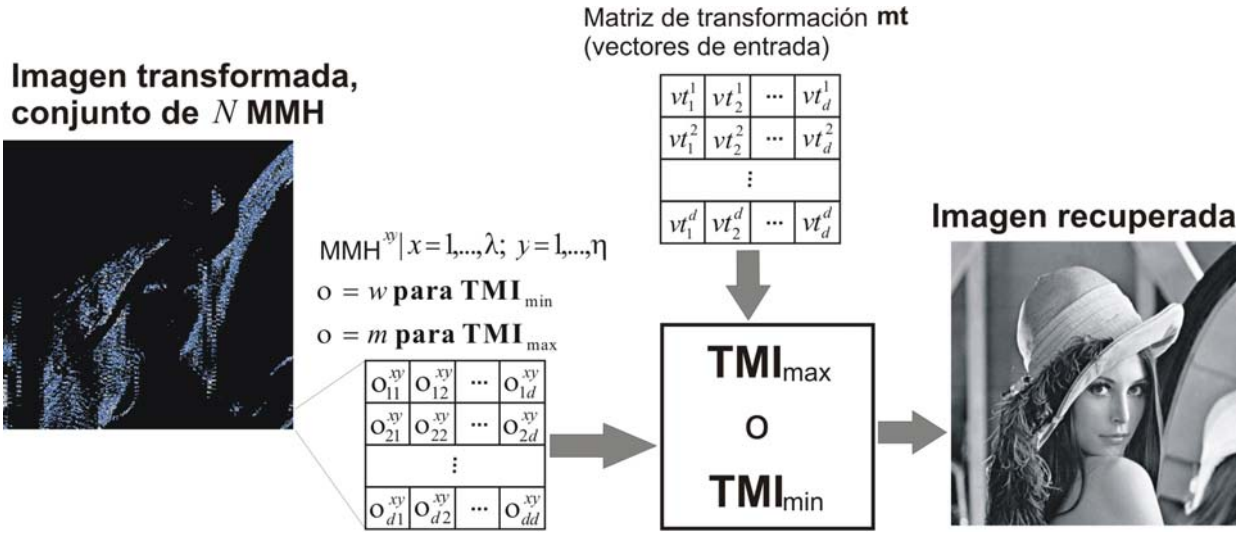


Figura 4.3. Esquema de la **TMI**.

Tabla 4.4. Seudo-código del algoritmo de una **TMI**.

```
01| programa_principal()
02| variables
03|    λ,η,k,h,l,y,x: entero
04| inicio
05|    // Barre la TM en bloques de  d × d
06|    para (λ=0 mientras_que λ<alto_i [operaciones λ=λ+d]) hacer
07|      para (η =0 mientras_que η< ancho_i [operaciones η=η+d]) hacer
08|         // Separa cada una de la MMH e inicializa el sub-bloque de imagen
09|         para (y=0 mientras_que y<d [operaciones y=y+1) hacer
10|        para (x=0 mientras_que x<d [operaciones x=x+1]) hacer
11|               w[x][y]=TM[η +x][ λ +y];   // m[x][y] para una TM_max. Extrae una MMH
12|               sb[x][y]= 2^L;     // =-2^L para una TM_max. Inicialización del
13|            fin_para                                  // sub-bloque de imagen
14|         fin_para
15|         subrutina rec_min()   //o rec_max()
16|         // Almacena los bloques de imagen obtenidos
17|         para (y=0 mientras_que y<d [operaciones y=y+1) hacer
18|        para (x=0 mientras_que x<d [operaciones x=x+1]) hacer
19|               A[η +x][ λ +y]=sb[x][y];
20|            fin_para
21|         fin_para
22|       fin_para
23|    fin_para
24| fin_programa_principal
```

Tabla 4.5. Seudo-código del algoritmo para la recuperación de un sub-bloque de imagen transformado con una $\mathbf{TM}_{min}$.

```
01| subrutina rec_min()
02| variables
03|    y,x,l,s,aux: entero
04| inicio
05|    s=0;
06|    para (l=0 mientras_que y<(d)(d) [operaciones l=l+d]) hacer
07|       para (y=0 mientras_que y<d [operaciones y=y+1]) hacer
08|          para (x =0 mientras_que x<d [operaciones x=x+1]) hacer
09|             aux=w[x][y]+vt[l+x];
10|             si (aux>sb[y][s]) entonces
11|                sb[y][s]=aux;
12|             fin_si
13|          fin_para
14|       fin_para
15|       s=s+1;
16|    fin_para
17| fin_subrutina
```

Tabla 4.6. Seudo-código del algoritmo para la recuperación de un sub-bloque de imagen transformado con una $\mathbf{TM}_{max}$.

```
01| subrutina rec_max()
02| variables
03|  y,x,l,s,aux: entero
04| inicio
05|    s=0;
06|    para (l=0 mientras_que y<(d)(d) [operaciones l=l+d]) hacer
07|       para (y=0 mientras_que y<d [operaciones y=y+1]) hacer
08|          para (x =0 mientras_que x<d [operaciones x=x+1]) hacer
09|             aux=m[x][y]+vt[l+x];
10|             si (aux<sb[y][s]) entonces
11|                sb[y][s]=aux;
12|             fin_si
13|          fin_para
14|       fin_para
15|       s=s+1;
16|    fin_para
17| fin_subrutina
```

La **TM** es una transformada basada en bloques; es decir, el algoritmo de la **TM** divide la imagen en bloques no traslapados y los procesa en forma separada. La elección del tamaño del bloque tiene una relación directa con el proceso de cuantificación ya que para obtener un valor alto en el parámetro de relación de compresión, los coeficientes de la transformada necesitan ser cuantificados; este proceso provocará pérdidas de información y por lo tanto un error en la reconstrucción de la imagen. Este error es más visible en los límites de los bloques, causando una discontinuidad en la imagen, esta discontinuidad es conocida como distrosión por bloques (*blockiness*) y es debida a que la correlación espacial entre bloques adyacentes no es considerada en el proceso de transformación de la imagen.

Por lo tanto, la elección del tamaño del bloque en el proceso de la **TM** resulta de un compromiso entre la eficiencia en la compresión y la pérdida de información provocada por la distorsión por bloques; esto es, la elección de tamaños de bloques grandes provoca una buena eficiencia en la compresión, pero el efecto de distorción por bloques es, desde un punto de vista subjetivo, muy notorio; por otra parte, al elegir tamaños de bloques pequeños se obtendrá una baja eficiencia en la compresión, pero el efecto de distorción por bloques será muy dificil de apreciar.

Con base en los diversos experimentos realizados en este trabajo y en la información consultada referente a otras transformadas basadas en bloques, los tamaños de bloques que mejores resultados ofrecen son: 4×4, 8×8 y 16×16.

4.4.3 Complejidad del algoritmo

Se define como *algoritmo* al procedimiento o método de cálculo, con reglas bien definidas, que conduce a la resolución de un problema específico.

Una vez que se dispone de un algoritmo que funciona correctamente, es necesario definir criterios para medir su rendimiento o comportamiento. Estos criterios determinan la eficiencia de un algoritmo, la cual permite evaluar de alguna forma el coste que consume un algoritmo en encontrar la solución al problema planteado; además, ofrece la posibilidad de comparar distintos algoritmos que resuelven el mismo problema.

El uso eficiente de los recursos por parte del algoritmo, suele medirse en función de dos parámetros: el *espacio*, es decir, la cantidad de memoria que utiliza, y el *tiempo*, lo que tarda en ejecutarse. Ambos parámetros representan los costes que supone encontrar la solución al problema planteado mediante un algoritmo y dependen directamente de la complejidad intrínseca de éste.

La *complejidad de un algoritmo* se define como la medida de los recursos que requiere su ejecución en función del tamaño de los datos de entrada.

Se entiende por *tamaño de los datos de entrada* al número de componentes sobre los que se va a ejecutar el algoritmo. Por ejemplo, la dimensión del vector a ordenar o el tamaño de las matrices a multiplicar.

Cuando se desea medir la cantidad de memoria requerida por el algoritmo para su implementación se habla de *complejidad en espacio*; si lo que se desea medir es el tiempo que consume el algoritmo durante su ejecución se habla de *complejidad en tiempo*.

La complejidad en espacio tiene por función determinar la cantidad de memoria del sistema que utiliza el algoritmo en la solución del problema planteado, dependiendo exclusivamente del tipo de datos utilizados por el algoritmo.

Con respecto a la complejidad en tiempo, hay dos casos posibles de estudio:

- Tiempo de ejecución. Consiste en medir el tiempo de ejecución en función del número de operaciones que el algoritmo realiza.
- Cota de complejidad. Permite determinar el *orden de crecimiento* del algoritmo.

La unidad de tiempo utilizada en estas medidas de eficiencia no puede ser expresada en segundos u otra unidad de tiempo concreta; por tanto, $T(n)$ denota el tiempo de ejecución de un algoritmo para una entrada de datos de tamaño *n*.

En este momento es necesario definir el *Principio de Invariancia*. Dado un algoritmo y dos implementaciones suyas I_1 e I_2, que tardan $T_1(n)$ y $T_2(n)$ tiempos respectivamente, el principio de invarianza afirma que existe una constante real $c > 0$ y un número natural n_0 tales que para todo $n \geq n_0$ se verifica que $T_1(n) \leq cT_2(n)$. Es decir, el tiempo de dos implementaciones distintas de un algoritmo dado no va a diferir más que en una constante multiplicativa. Con esto se puede definir que un algoritmo tarda un tiempo del orden de $T(n)$ si existe una constante real $c > 0$ y una implementación *I* del algoritmo que tarda menos que $cT(n)$, para todo *n* tamaño de entrada.

Retomando los posibles casos de estudio de una complejidad en tiempo de un algoritmo, con respecto al tiempo de ejecución, suelen estudiarse tres casos para un mismo algoritmo: *caso peor*, *caso medio* y *caso mejor*.

El caso mejor corresponde a la secuencia de sentencias del algoritmo que realiza menos operaciones. El caso peor corresponde a la secuencia de sentencias del algoritmo que realiza más operaciones. Respecto al caso medio, corresponde a la secuencia de sentencias del algoritmo que realiza un número de instrucciones igual a la esperanza matemática de la variable aleatoria definida por todas las posibles secuencias de sentencias del algoritmo para un tamaño de entrada dado, con las probabilidades de que éstas ocurran para esa entrada.

La medición del tiempo de ejecución se hace en función del número de *operaciones elementales* (OE) que realiza el algoritmo.

Las OE son aquellas que una computadora realiza en un tiempo acotado por una constante. Se consideran como OE las operaciones aritméticas básicas, asignación a variables, llamadas a funciones, regreso de ellas, comparaciones lógicas y el acceso a estructuras indexadas básicas, como son los vectores y matrices. Cada una de ellas se contabiliza como 1 OE.

Una vez determinado el tiempo de ejecución de un algoritmo es posible clasificarlo con la finalidad de tener un parámetro comparativo. Para ello se hace uso de clases de equivalencia correspondientes a funciones que crecen de la misma forma. Existen varias cotas que permiten caracterizar la complejidad de un algoritmo; una de las más usadas es la función O (Omicrón), empleada como cota superior. Dada una función f, al conjunto de funciones g que a lo sumo crecen tan deprisa como f se le denomina cota superior de f y se representa por $\mathrm{O}(f)$. Conociendo la cota superior de un algoritmo se puede asegurar que, en ningún caso, el tiempo empleado por éste será de un orden superior al de la cota. La función O es definida a continuación.

Cota superior, notación O. Sea $f: \mathbb{N}\to[0,\infty)$, se define el conjunto de funciones de orden O de f como [18]:

$$\mathrm{O}(f)=\{g:\mathbb{N}\to[0,\infty)\mid \exists c\in\mathbb{R},\ c>0, \exists n_0\in\mathbb{N},\ g(n)\le cf(n)\forall n\ge n_0\} \qquad \textbf{(4.27)}$$

Se dice entonces, que una función $t: \mathbb{N}\to[0,\infty)$ es de orden O de f si $t\in\mathrm{O}(f)$.

4.4.3.1 Complejidad en tiempo del algoritmo de la TM

Para medir la complejidad en tiempo del algoritmo de la **TM** se obtendrá en primer término el tiempo de ejecución, en función del número de OE que realiza, del elemento más representativo del algoritmo, la subrutina `P_max()` o `P_min()` encargada de calcular la transformada morfológica de un sub-bloque de imagen; considerando el seudo-código de la tabla 4.2:

- En la línea 5 se realizan 3 OE (una asignación, una comparación y una suma) en cada iteración del bucle más otras tres cuando se efectúa la salida del mismo.
- Lo mismo ocurre en la línea 6, realiza 3 OE en cada iteración y 3 más al salir del bucle.
- Lo mismo ocurre en la línea 7, realiza 3 OE en cada iteración y 3 más al salir del bucle.
- En la línea 8 se realizan 6 OE: 2 sumas, 2 accesos a vector, 1 resta y 1 asignación.
- En la línea 9 se realiza una condición, con un total de 2 OE: un acceso a una matriz y una comparación.
- La línea 10 sólo se ejecuta si se cumple la condición de la línea 9; si es así, se realizan 2 OE: un acceso a una matriz y una asignación.

Tomando en cuenta estas consideraciones:

En el *caso mejor* para el algoritmo la condición de la línea 9 será siempre falsa y la línea 10 nunca será ejecutada. Así, el bucle más interno realizará *d* iteraciones, cada una de ellas con 8 OE (correspondientes a las líneas 8 y 9) más 3 OE implícitas a él. Por tanto, este bucle realiza un total de

$$\left(\sum_{x=0}^{d}(8+3)\right)+3=11\left(\sum_{x=0}^{d}1\right)+3=11(d)+3$$

OE, añadiendo el 3 por la condición de salida del bucle.

El siguiente bucle repetirá esas $11(d)+3$ OE en cada iteración:

$$\left(\sum_{y=0}^{d}(11d+3)+3\right)+3=\left(\sum_{y=0}^{d}11d+6\right)+3=d(11d+6)+3=11d^2+6d+3$$

El último bucle repetirá esas $11d^2+6d+3$ OE en cada iteración; es claro que este bucle se repetirá *k* veces, donde k representa el número de elementos que forman parte del conjunto fundamental de asociaciones; entonces, el número total de OE que realiza el algoritmo es:

$$T(n)=\left(\sum_{l=0}^{k}\left(11d^2+6d+3\right)+3\right)+3=k\left(11d^2+6d+6\right)+3$$

En el *caso peor*, la condición de la línea 9 será siempre verdadera; por tanto, la línea 10 será ejecutada en todas las iteraciones, entonces el bucle más interno realizará el siguiente número de OE:

$$\left(\sum_{x=0}^{d}(10+3)\right)+3=13\left(\sum_{x=0}^{d}1\right)+3=13d+3$$

Los siguientes bucles realizarán el mismo número de iteraciones que el caso anterior; por tanto el número de OE para este caso es:

$$T(n)=k\left(13d^2+6d+6\right)+3$$

Para el *caso medio*, la condición de la línea 9 será verdadera con probabilidad ½. Así, la línea 10 será ejecutada la mitad de las iteraciones del bucle más interno; por tanto el número de OE que realiza es:

$$\left(\sum_{x=0}^{d}\left(8+\frac{2}{2}+3\right)\right)+3=12\left(\sum_{x=0}^{d}1\right)+3=12d+3$$

Los siguientes bucles realizarán el mismo número de iteraciones que los casos anteriores, por tanto el número de OE para este caso es:

$$T(n)=k\left(12d^2+6d+6\right)+3$$

Una vez expresado el algoritmo en función de las OE que realiza, ahora es necesario determinar cuál es su orden de crecimiento. Para este propósito se hace uso de la cota superior O.

Considerando el caso peor y fijando $k=8$, el número de OE que realiza el algoritmo es:

$$f(n)=8\left(13n^2+6n+6\right)+3 \qquad \textbf{(4.28)}$$
$$f(n)=104n^2+48n+51$$

Para determinar el orden de crecimiento de la función $f(n)$ se hace uso de las reglas definidas en [32]:

1. Si $f(n)$ es un polinomio de grado *x*, entonces $f(n)$ es $\mathrm{O}(n^x)$, es decir,
   i. Prescindir de los términos de orden menor.
   ii. Prescindir de los factores constantes.

2. Usar la clase más pequeña posible de funciones, es decir, "$2n$ es $\mathrm{O}(n)$" en vez de "$2n$ es $\mathrm{O}(n^2)$".
3. Usar la expresión más simple de la clase, es decir, "$3n+5$ es $\mathrm{O}(n)$" en vez de "$3n+5$ es $\mathrm{O}(3n)$".

Con base en estas reglas se concluye que el orden de crecimiento del algoritmo propuesto es $\mathrm{O}(n^2)$.

Una forma de demostrar esta afirmación es considerar la definición de la notación O; considerando la expresión 4.27, para el caso $\mathrm{O}(n)$ se tiene:

$$104n^2 + 48n + 51 \leq cn \tag{4.29}$$

Si $n_0 = 1$, entonces:

$$104 + 48 + 51 \leq c$$
$$c = 200$$

La expresión 4.27 debe cumplirse para cualquier $n > 1$; se evalúa para $n = 2$:

$$104n^2 + 48n + 51 \leq 200n$$
$$563 \leq 400$$

La expresión no se cumple, entonces la función $f(n)$ no es $\mathrm{O}(n)$.

Para el caso $\mathrm{O}(n \log_2 n)$ se tiene:

$$104n^2 + 48n + 51 \leq cn \log_2 n \tag{4.30}$$

Si $n_0 = 2$, entonces:

$$416 + 96 + 51 \leq 2c$$
$$c = 281.5$$

La expresión 4.27 debe cumplirse para cualquier $n > 2$; se evalúa para $n = 4$:

$$104n^2 + 48n + 51 \leq 281.5n \log_2 n$$
$$2248 \leq 1907$$

La expresión no se cumple, entonces la función $f(n)$ no es $\mathrm{O}(n \log_2 n)$.

Para el caso $\mathrm{O}(n^2)$ se tiene:

$$104n^2 + 48n + 51 \leq cn^2 \tag{4.31}$$

Si $n_0 = 1$, entonces:

$$104 + 48 + 51 \leq c$$
$$c = 200$$

La expresión 4.27 debe cumplirse para cualquier $n > 1$; se evalúa para $n = 2$:

$$104n^2 + 48n + 51 \leq 200n^2$$
$$563 \leq 800$$

La expresión se cumple, entonces la función $f(n)$ es $O(n^2)$.

4.4.3.2 Complejidad en espacio del algoritmo de la TM

La complejidad en espacio del algoritmo de la **TM** es determinado por la cantidad de memoria requerida durante su ejecución. Para obtener este parámetro consideraremos los seudo-códigos de las tablas 4.2 y 4.3:

El proceso de transformación de un sub-bloque de imagen de dimensión $d \times d$ requiere de 2 vectores $\mathbf{vt}[d \times d]$ y $\mathbf{vi}[d \times d]$, y de una matriz $\mathbf{w}[d][d]$; el número de localidades de memoria requeridas para este proceso será:

$$\text{loc\_P1} = \text{loc\_}\mathbf{vt} + \text{loc\_}\mathbf{vi} + \text{loc\_}\mathbf{w}$$

La imagen transformada necesita para su almacenamiento una matriz $\text{TM}[alto\_i][ancho\_i]$, donde $alto\_i$ y $ancho\_i$ representan el alto y el ancho de la imagen original respectivamente; el número de localidades de memoria requeridas para este proceso será:

$$\text{loc\_P2} = \text{loc\_TM}$$

Entonces el número total de localidades de memoria requeridas por el algoritmo de la **TM** será la suma de las localidades requeridas por los procesos que lo forman:

$$\text{loc\_P1} + \text{loc\_P2}$$
$$\text{loc\_}\mathbf{vt} + \text{loc\_}\mathbf{vi} + \text{loc\_}\mathbf{w} + \text{loc\_TM}$$
$$(d)(d) + (d)(d) + (d)(d) + (ancho\_i)(alto\_i)$$
$$3d^2 + (ancho\_i)(alto\_i)$$

Es claro que el algoritmo de la **TM** hace uso únicamente de operaciones de suma, resta y comparación, por lo que el resultado nunca será un número de punto flotante. Además el algoritmo de la **TM** es implementado para utilizarse en la compresión de imágenes en tonos de gris, el valor máximo que puede tomar un píxel es 255; por tanto, el resultado de la transformación de un sub-bloque de imagen puede generar valores negativos, requiriendo para su representación variables de más de 8 bits. En la mayoría de los copiladores existe la posibilidad de declarar variables tipo *short*; las variables declaradas como tipo *short* denotan enteros con signo de 16 bits.

Tomando en cuenta las consideraciones anteriores, el número total de bytes requeridos por el algoritmo de la **TM** es:

$$2\left(3d^2 + (ancho\_i)(alto\_i)\right) \qquad \textbf{(4.32)}$$

Considerando esta expresión, se concluye que el número de bytes totales requeridos por el algoritmo de la **TM** depende directamente de la dimensión de la imagen y del tamaño del sub-bloque de imagen elegido para la transformación de la imagen.

Capítulo 5

Resultados experimentales

Este capítulo presenta los resultados experimentales obtenidos cuando se usa la **TM** en un sistema de compresión de imágenes. En el primer experimento se estudia el desempeño de la **TM** cuando se aplica a la imagen transformada un proceso de cuantificación vectorial con libros de códigos de diferentes tamaños; posteriormente se analiza el desempeño de la **TM** cuando opera en forma conjunta con diversos algoritmos de codificación; finalmente, se compara su desempeño con el de los métodos tradicionales de transformación, DCT y DWT.

Para la realización de estos experimentos se eligió usar un conjunto de 5 imágenes: Lena, Pimientos (*Peppers*), Elaine, Barco (*Boat*) y Colina de oro (*Goldhill*) (tamaño: 512×512 píxeles, escala de grises 8 bits/píxel); las imágenes son mostradas en la figura 5.1. La figura 5.2 muestra el esquema del compresor de imágenes utilizado en los experimentos, siendo elegido un sistema de compresión con pérdidas.

Figura 5.1. Conjunto de imágenes de prueba: (a) Lena, (b) Pimientos, (c) Elaine, (d) Barco, (e) Colina de oro.

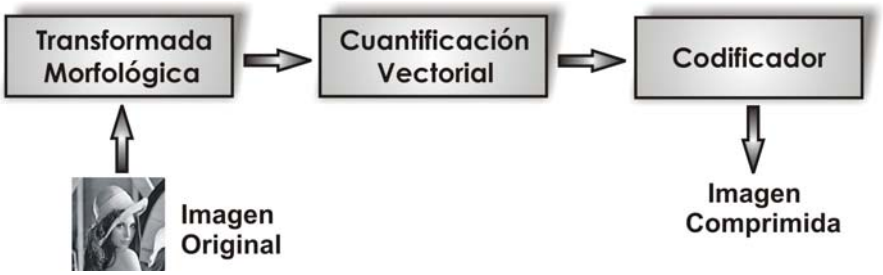


Figura 5.2. Esquema del sistema de compresión con pérdidas utilizado en los experimentos.

Para medir el desempeño en los experimentos realizados, se hace uso del popular criterio objetivo denominado relación señal a ruido (PSNR), definido en el apartado 2.3.6.2.

$$PSNR = 10\log_{10}\left(\frac{\left(2^n - 1\right)^2}{\frac{1}{M}\sum_{i=1}^{M}\left(p_i - \tilde{p}_i\right)^2}\right) \qquad \textbf{(5.1)}$$

donde n es el número de bits utilizados en la representación de un píxel, M es el número de píxeles que forman la imagen, p_i es el i-ésimo píxel en la imagen original y $\tilde{p}_i$ es el i-ésimo píxel en la imagen reconstruida.

5.1 Experimento 1. Desempeño de la Transformada Morfológica cuando se usa una cuantificación vectorial

El primer experimento consistió únicamente de las dos primeras etapas del sistema mostrado en la figura 5.2, la **TM** y la cuantificación vectorial. Al ser la etapa de cuantificación la causante de pérdida de información en la imagen, este primer experimento tiene por finalidad analizar cómo el proceso de **TMI** reduce la cantidad de datos degradados por el cuantificador. Para este proceso se ha hecho uso de una cuantificación vectorial LBG multi-etapa; este método emplea el algoritmo LBG [52]. La tabla 5.1 muestra la PSNR obtenida cuando se aplica una cuantificación vectorial con diversos tamaños de libros de códigos sobre los resultados de aplicar la **TM** en las imágenes de prueba.

Tabla 5.1. Desempeño de la **TM** sobre las imágenes de prueba cuando se usa cuantificación vectorial con varios tamaños en el libro de códigos.

Imagen	Desempeño de la **TM** (PSNR)			
	Cuantificación vectorial LBG multi-etapa			
	64 vectores de reconstrucción	128 vectores de reconstrucción	256 vectores de reconstrucción	512 vectores de reconstrucción
Lena	27.12	28.20	29.09	29.99
Elaine	28.87	29.67	30.32	30.72
Pimientos	26.20	27.13	27.64	28.15
Colina de oro	26.19	26.92	27.54	28.00
Barco	24.91	25.79	26.33	26.84

Con la finalidad de poder aplicar una medida subjetiva, la figura 5.3 muestra las imágenes resultantes de aplicar el proceso mencionado sobre las imágenes Lena, Pimientos y Barco y la figura 5.4 muestra el error entre estas imágenes y las originales.

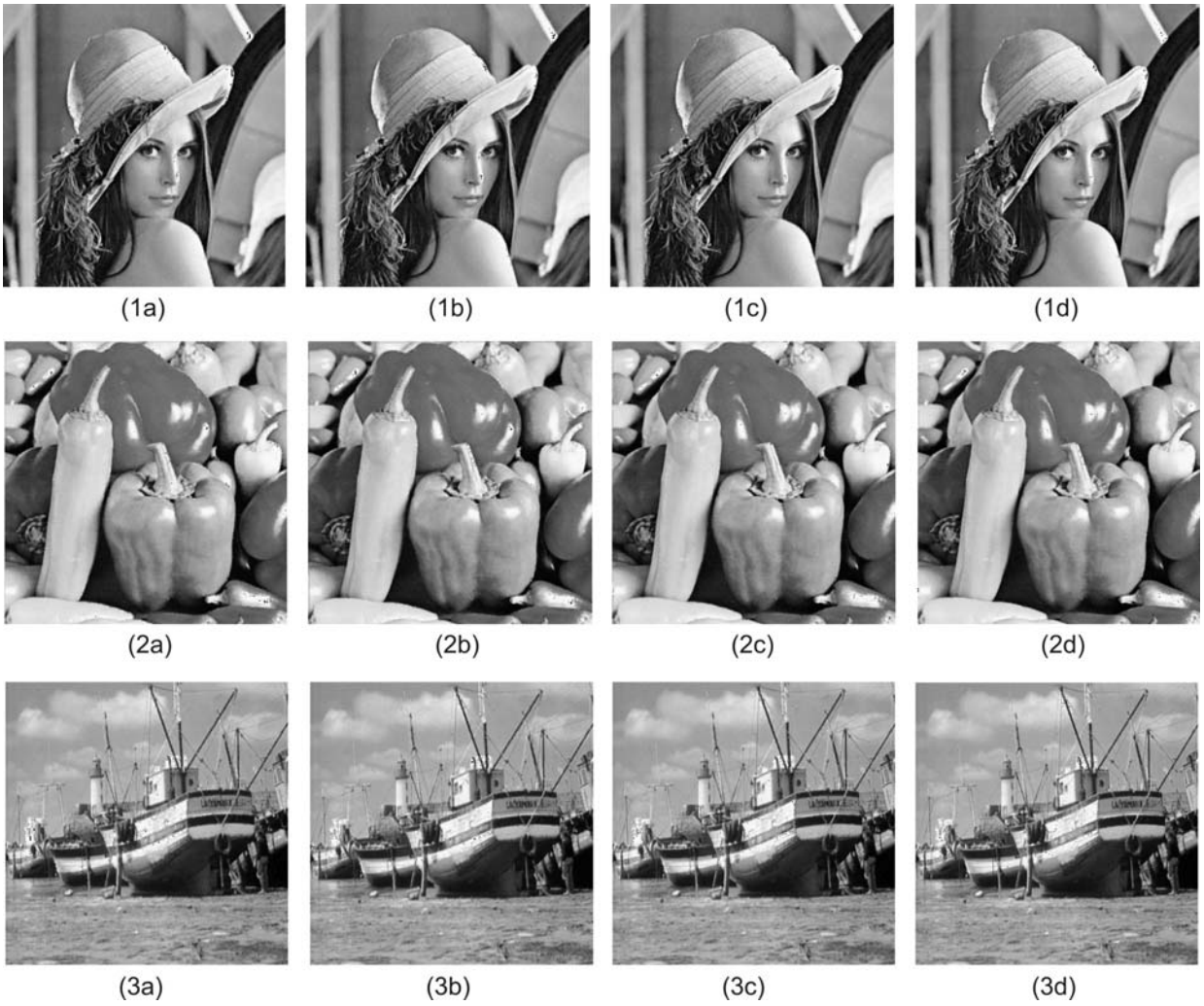


Figura 5.3. Resultados de aplicar la **TM** y VQ sobre las imágenes de prueba; (a) 64 vectores de reconstrucción, (b) 128 vectores de reconstrucción, (c) 256 vectores de reconstrucción, (d) 512 vectores de reconstrucción.

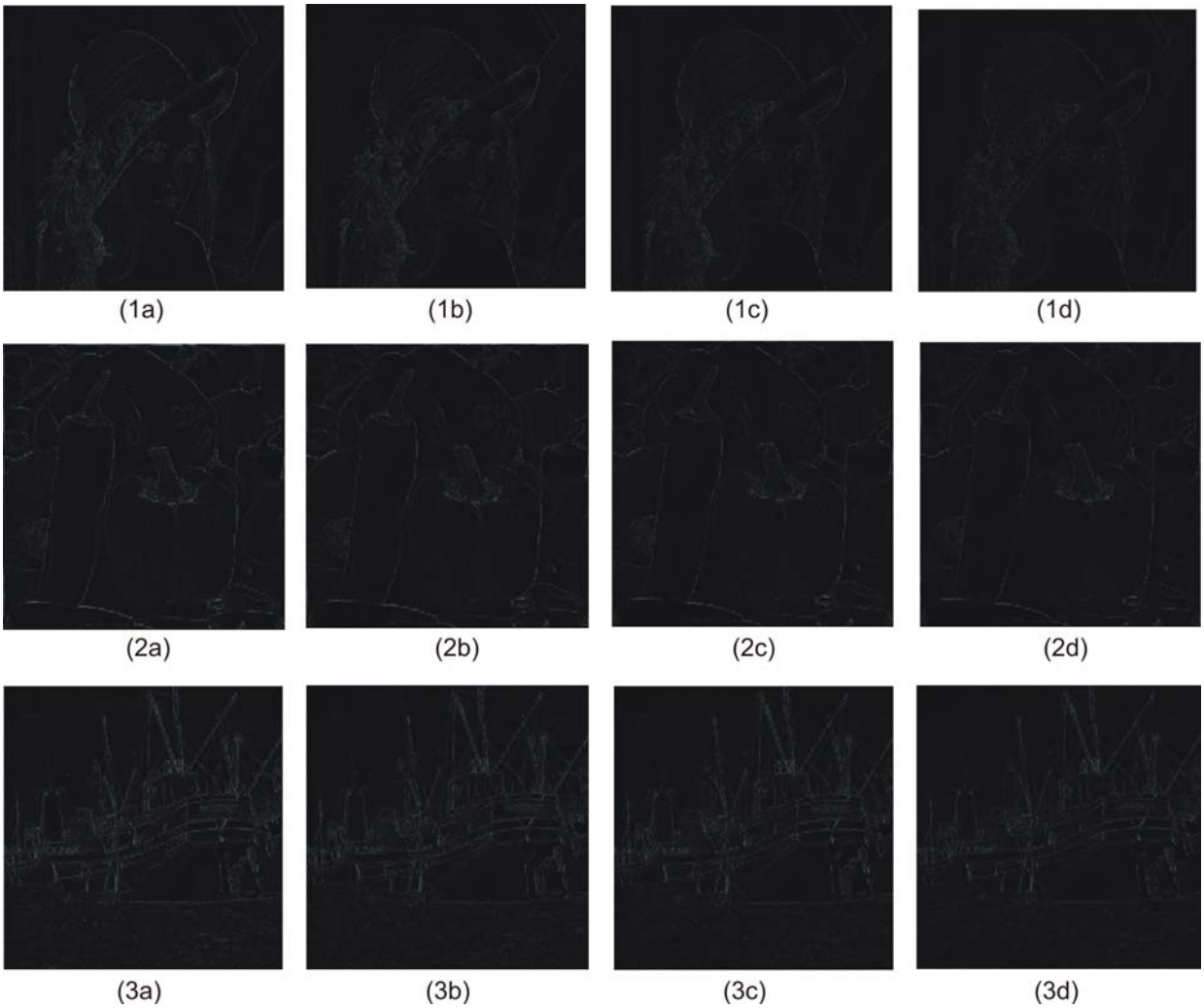


Figura 5.4. Imágenes del error entre las imágenes originales y las imágenes reconstruidas; (a) 64 vectores de reconstrucción, (b) 128 vectores de reconstrucción, (c) 256 vectores de reconstrucción, (d) 512 vectores de reconstrucción.

Un proceso de VQ es el causante de la pérdida de información dentro de un sistema de compresión de imágenes; como se mencionó, este experimento tiene por finalidad analizar el desempeño de un sistema formado por la **TM** y una VQ-LBG. Una forma de medir el desempeño es aplicar el criterio objetivo PSNR; los resultados de este experimento, concentrados en la tabla 5.1, muestran que el sistema formado por la **TM** y una VQ-LBG tiene un desempeño moderado cuando se utilizan vectores de reconstrucción de tamaños pequeños (64, 128). Cuando el tamaño en los vectores de reconstrucción aumenta (512) el desempeño del sistema es relativamente bueno, tomando en cuenta que valores por arriba de 30 dB en el parámetro de PSNR es considerado aceptable en la mayor parte de la bibliografía especializada en este tema [40], [61].

Otra forma de medir la calidad de la imagen procesada por el esquema propuesto (**MT** y VQ-LBG) es utilizar un criterio subjetivo, criterio individual que depende exclusivamente del punto de vista de una persona experta que en ese momento está analizando el desempeño del sistema. Para que el lector pueda sacar sus propias conclusiones, se han añadidos las figuras 5.3 y 5.4, las cuales contienen los resultados de aplicar la **TM** y VQ-LBG sobre las imágenes de prueba (utilizando diversos tamaños de vectores de reconstrucción) y las respectivas imágenes de error. Aspectos a considerar en dicha evaluación son incluidos en [28], [53].

5.2 Experimento 2. Implementación de un compresor de imágenes usando la Transformada Morfológica

El segundo experimento tiene por finalidad analizar la repercusión que tiene la representación de la imagen, generada por la **TM**, en el proceso de compresión de la imagen. Para este propósito fue necesario implementar un *CODEC* (Codificador-Decodificador) de imágenes como el mostrado en la figura 5.2; en la etapa de codificación fueron aplicados diversos métodos estándares de codificación; en tales métodos se incluyen algoritmos basados en técnicas estadísticas, como el aritmético, Huffman, range, Burrows-Wheller y PPM, y algoritmos basados en técnicas de diccionario, como LZ77 y LZP.

Para analizar el desempeño del compresor de imágenes constituido por la **TM**, una cuantificación vectorial LBG y un codificador, la tabla 5.2 contiene los resultados obtenidos en el segundo experimento y en la figura 5.5 se muestran las gráficas de PSNR versus *bit rate* (bpp) y de PSNR versus relación de compresión obtenidas de aplicar el compresor sobre las imágenes de prueba. En estos experimentos el tamaño del libro de códigos del cuantificador vectorial se varió con la finalidad de observar cómo repercute en el parámetro de la relación de compresión.

En el primer experimento se analizó el parámetro de PSNR, en este segundo experimento podemos observar la relación obtenida, con el uso de la **TM** en el sistema de compresión, entre la PSNR y la relación de compresión. De estos resultados se puede concluir que la técnica de codificación que ofrece un mejor desempeño, en la relación de compresión con la menor distorsión, al aplicarla sobre los resultados generados por la **TM** es el algoritmo estadístico PPM; este algoritmo es un método estadístico adaptativo de compresión de datos que basa su operación en la predicción por coincidencia parcial, es decir, PPM predice el valor de un elemento con base en secuencias previas de elementos.

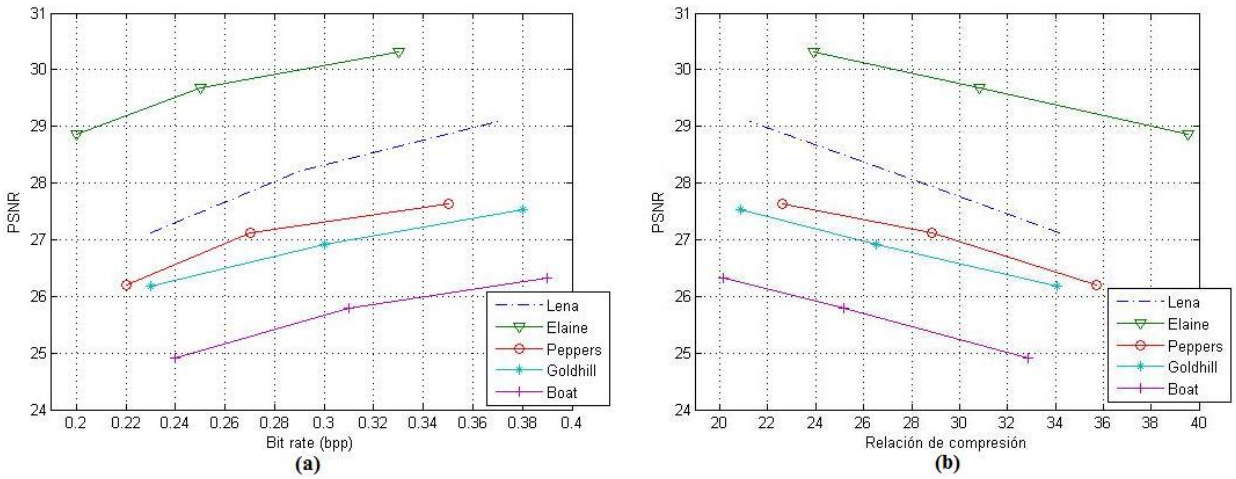


Figura 5.5. Gráficas del desempeño del compresor formado por **TM**, VQ-LBG y PPM; (a) PSNR vs. *Bit rate*, (b) PSNR vs. relación de compresión.

Tabla 5.2. Comparación del desempeño de varias técnicas de codificación de entropía aplicadas sobre imágenes transformadas con la **TM**.

Imagen	Técnica de codificación		Cuantificación vectorial LBG multi-etapa No. de vectores de reconstrucción		
			64	128	256
Lena	PPM	Relación de Compresión	34.17	27.11	21.26
		Bit rate	0.23 bpp	0.29 bpp	0.37 bpp
	Burrown W.	Relación de Compresión	33.11	26.33	20.56
		Bit rate	0.24 bpp	0.30 bpp	0.38 bpp
	LZP	Relación de Compresión	30.98	25.56	20.35
		Bit rate	0.25 bpp	0.31	0.39 bpp
	LZ77	Relación de Compresión	30.23	24.21	19.21
		Bit rate	0.26 bpp	0.33 bpp	0.41 bpp
	Range	Relación de Compresión	23.86	20.24	17.27
		Bit rate	0.33 bpp	0.39 bpp	0.46 bpp
Barco	PPM	Relación de Compresión	32.85	25.17	20.12
		Bit rate	0.24 bpp	0.31 bpp	0.39 bpp
	Burrown W.	Relación de Compresión	32.40	25.05	19.88
		Bit rate	0.24 bpp	0.32 bpp	0.40 bpp
	LZP	Relación de Compresión	30.56	23.95	19.32
		Bit rate	0.26 bpp	0.33 bpp	0.41 bpp
	LZ77	Relación de Compresión	30.06	23.81	19.46
		Bit rate	0.26 bpp	0.33 bpp	0.41 bpp
	Range	Relación de Compresión	27.15	22.19	18.72
		Bit rate	0.29 bpp	0.36 bpp	0.42 bpp
Elaine	PPM	Relación de Compresión	39.55	30.83	23.90
		Bit rate	0.20 bpp	0.25 bpp	0.33 bpp
	Burrown W.	Relación de Compresión	36.70	28.84	22.74
		Bit rate	0.21 bpp	0.27 bpp	0.35 bpp
	LZP	Relación de Compresión	34.90	28.04	22.32
		Bit rate	0.22 bpp	0.28 bpp	0.35 bpp
	LZ77	Relación de Compresión	33.00	26.49	21.04
		Bit rate	0.24 bpp	0.30 bpp	0.38 bpp
	Range	Relación de Compresión	27.65	23.16	19.45
		Bit rate	0.28 bpp	0.34 bpp	0.41 bpp
Colina de oro	PPM	Relación de Compresión	34.05	26.54	20.86
		Bit rate	0.23 bpp	0.30 bpp	0.38 bpp
	Burrown W.	Relación de Compresión	33.22	26.19	20.62
		Bit rate	0.24 bpp	0.30 bpp	0.38 bpp
	LZP	Relación de Compresión	31.31	25.08	20.02
		Bit rate	0.25 bpp	0.31 bpp	0.39 bpp
	LZ77	Relación de Compresión	30.87	24.88	20.12
		Bit rate	0.25 bpp	0.32 bpp	0.39 bpp
	Range	Relación de Compresión	26.42	22.48	19.11
		Bit rate	0.30 bpp	0.35 bpp	0.41 bpp
Pimientos	PPM	Relación de Compresión	35.72	28.83	22.62
		Bit rate	0.22 bpp	0.27 bpp	0.35 bpp
	Burrown W.	Relación de Compresión	34.33	27.76	21.79
		Bit rate	0.23 bpp	0.28 bpp	0.36 bpp
	LZP	Relación de Compresión	32.12	26.63	21.39
		Bit rate	0.24 bpp	0.30 bpp	0.37 bpp
	LZ77	Relación de Compresión	31.19	25.70	20.35
		Bit rate	0.25 bpp	0.31 bpp	0.39 bpp
	Range	Relación de Compresión	25.20	21.78	18.53
		Bit rate	0.31 bpp	0.36 bpp	0.43 bpp

La etapa de transformación de un compresor de imágenes no produce reducción de información por si sola, su principal propósito es facilitar la compresión de la información en las etapas siguientes. Por tal motivo y para tener un punto de comparación de cómo la representación de la imagen, generada por la **TM**, tiene repercusión en el proceso de compresión de la imagen, se ha omitido la etapa de transformación, la **TM**, del esquema de compresión utilizado en este experimento. Los resultados de esta parte del experimento son mostrados en la tabla 5.3; estos resultados muestran que el uso de la **TM** mejora considerablemente la relación de compresión y en algunos casos mejora la relación señal a ruido.

Tabla 5.3. Resultados obtenidos en los parámetros de relación compresión y PSNR cuando se omite la **TM** del esquema de compresión.

Imagen	Técnica de codificación		Cuantificación vectorial LBG multi-etapa No. de vectores de reconstrucción		
			64 (PSNR 26.66)	128 (PSNR 27.28)	256 (PSNR 27.57)
Lena	PPM	Relación de Compresión	14.23	11.50	9.43
		Bit rate	0.56 bpp	0.69 bpp	0.84 bpp
	Burrows W.	Relación de Compresión	13.79	11.27	8.94
		Bit rate	0.58 bpp	0.71 bpp	0.89 bpp
	Range	Relación de Compresión	11.98	10.08	8.61
		Bit rate	0.66 bpp	0.79 bpp	0.92 bpp
			Cuantificación vectorial LBG multi-etapa No. de vectores de reconstrucción		
			64 (PSNR 28.95)	128 (PSNR 29.61)	256 (PSNR 30.10)
Colina de oro	PPM	Relación de Compresión	14.23	11.40	9.35
		Bit rate	0.56 bpp	0.70 bpp	0.85 bpp
	Burrows W.	Relación de Compresión	13.76	11.29	8.96
		Bit rate	0.58 bpp	0.70 bpp	0.89 bpp
	Range	Relación de Compresión	12.34	10.46	8.90
		Bit rate	0.64 bpp	0.76 bpp	0.89 bpp
			Cuantificación vectorial LBG multi-etapa No. de vectores de reconstrucción		
			64 (PSNR 25.81)	128 (PSNR 27.64)	256 (PSNR 27.75)
Pimientos	PPM	Relación de Compresión	15.06	12.12	9.82
		Bit rate	0.53 bpp	0.65 bpp	0.81 bpp
	Burrows W.	Relación de Compresión	14.52	11.83	9.62
		Bit rate	0.55 bpp	0.67 bpp	0.83 bpp
	Range	Relación de Compresión	12.43	10.56	8.97
		Bit rate	0.64 bpp	0.75 bpp	0.89 bpp

5.3 Experimento 3. Comparación del desempeño de la Transformada Morfológica y de los métodos tradicionales de transformación

El tercer experimento tiene por finalidad comparar el desempeño del compresor implementado en el experimento 2, constituido por la **TM**, una cuantificación vectorial LBG y un codificador PPM, con métodos de compresión estándares; en el experimento se incluyeron los siguientes métodos de compresión estándares: JPEG [96], [97], DCT *Based Embedded* [96], EZW [81], [97], SPIHT [77], [97], EBCOT [88].

La tabla 5.4 presenta los resultados comparativos entre el sistema propuesto y los métodos de compresión estándares mencionados, aplicados a la imagen Lena de 512×512 píxeles, en escala de grises: 8 bits/píxel.

Tabla 5.4. Resultados de aplicar el CODEC formado por la **TM**, VQ LBG y el codificador PPM, y métodos de compresión estándares sobre las imágenes Lena y Barbara (512×512 píxeles, en escala de grises: 8 bits/píxel).

Método de compresión	*Bit rate*	Imagen	
		Lena (PSNR)	Barbara (PSNR)
Bseline JPEG [96], [97]	0.25	31.6	25.2
	0.50	34.9	28.3
DCT-Based Embedded [96]	0.25	32.25	26.83
	0.50	36.0	30.82
EZW [81], [97]	0.25	33.17	26.77
	0.50	36.28	30.53
SPIHT [77], [97]	0.25	34.11	27.58
	0.50	37.21	31.39
EBCOT [88]	0.25	34.40	29.03
	0.50	37.49	33.06
Transformada morfológica (**TM**)	0.23	27.12	23.12
	0.52	31.14	25.93

Para establecer una comparación del desempeño entre el compresor propuesto, el cual incluye a la **TM**, y algunos métodos de compresión estándares, la interpretación de los resultados de la tabla 5.4 se puede dar desde dos puntos de vista: un criterio objetivo (medida cuantitativa) y un criterio subjetivo (medida cualitativa)

Para el criterio objetivo, el parámetro a considerar es la PSNR de la imagen reconstruida con los diferentes métodos utilizados en este experimento; la PSNR permite tener una medida relativa de la calidad de la imagen reconstruida. Si consideramos el valor de PSNR obtenido por el mejor método, EBCOT [88] con 37,49 dB, y el obtenido por el método que usa la **TM**, 31.14 dB, ambos para un *bit rate* de 0.50, podemos concluir que el método propuesto es superado por el algoritmo EBCOT; sin embargo, es necesario mencionar que el valor en PSNR obtenido por nuestra propuesta es considerado como de una calidad aceptable por la bibliografía especializada en este tema [40], [61].

Con respecto al criterio subjetivo, se trata de procedimientos que involucran al sistema visual humano (HVS, *human visual system*) como una medida integral de calidad de la imagen. El lector puede consultar las figuras 5.3 y 5.4 y sacar sus propias conclusiones. Los trabajos [28], [53] incluyen aspectos a considerar en dicha evaluación.

Como una conclusión derivada de estos tres experimentos y con base en los resultados generados por los mismos, podemos mencionar que el sistema propuesto, constituido por la **TM**, una cuantificación vectorial LBG y el codificador PPM, demostró una alta competitividad en las características de compresión y PSNR comparado con métodos tradicionales basados en la DCT (JPEG) o en la Transformada *Wavelet* (EZW, SPIHT y EBCOT).

5.4 Análisis comparativo entre la Transformada Morfológica y los métodos tradicionales de transformación: velocidad de procesamiento y recursos requeridos

Finalmente se hará un análisis del número y tipo de operaciones y del tipo de datos que utilizan la **TM** y los métodos tradicionales durante un proceso de transformación de una imagen, con la finalidad de establecer una comparación con respecto a sus velocidades de procesamiento y de la cantidad de memoria que requieren durante su operación. Cabe mencionar que este análisis es independiente de la implementación de los algoritmos de estas transformadas con un lenguaje especializado.

Primero se analiza el número y tipo de operaciones. Para todos lo casos, *op* representa el número y tipo de operaciones a realizar y *alto_i* y *ancho_i* son el alto y el ancho de la imagen respectivamente.

Se analiza primero una implementación eficiente de la DCT, la cual fue propuesta por Y. Arai en [11]; el número de operaciones que utiliza este algoritmo en la transformación de una imagen está en función de la dimensión de la imagen y del tamaño del bloque elegido.

$$\frac{alto\_i}{d}\left(\frac{ancho\_i}{d}\left(d(op)+d(op)\right)\right)$$

para una dimensión de bloque $d \times d$, y donde *op* = 29 sumas y 5 multiplicaciones. Reduciendo queda,

$$\frac{2(ancho\_i \times alto\_i)}{d}(op) \qquad \textbf{(5.2)}$$

Para el análisis del número y tipo de operaciones que son necesarias para transformar una imagen mediante la DWT, se considerará uno de los filtros *wavelet* más simples, el filtro Haar. Este filtro hace uso de operaciones de suma y multiplicación. Definido el filtro *wavelet* que se utilizará, el número de operaciones queda determinado por la dimensión de la imagen y el número de escalas a obtener.

$$operacionesE1 + operacionesE2 + ... + operacionesEn$$

Para una DWT de 3 escalas:

$$\frac{alto\_i}{2}\left(\frac{ancho\_i}{2}(op)\right)+\frac{alto\_i}{4}\left(\frac{ancho\_i}{4}(op)\right)+\frac{alto\_i}{8}\left(\frac{ancho\_i}{8}(op)\right)$$

donde *op* = 12 sumas y 8 multiplicaciones. Si $\dim = ancho\_i \times alto\_i$

$$\frac{\dim}{4}(op)+\frac{\dim}{16}(op)+\frac{\dim}{64}(op)$$

Generalizando para *n* escalas:

$$(op)\sum_{u=1}^{n}\frac{\dim}{2^{u+u}} \qquad \textbf{(5.3)}$$

Para una **TM**, el número de operaciones necesarias para transformar una imagen está en función de la dimensión de la imagen y la dimensión del sub-bloque de imagen seleccionado,

$$\frac{alto\_i}{d}\left(\frac{ancho\_i}{d}(d(d(d(op))))\right)$$

donde *d* es la dimensión de los vectores de imagen y *op* = 1 suma y 1 comparación; reduciendo

$$(ancho\_i \times alto\_i)(d)(op) \qquad \textbf{(5.4)}$$

Ahora se analizarán los requerimientos de memoria que cada transformada demanda durante su operación. Siguiendo la secuencia del análisis anterior, el primer estudio le corresponde a la DCT. Para procesar bloques de imagen de dimensión $d \times d$, esta transformada requiere durante su operación una matriz $DCT[alto\_i][ancho\_i]$, una matriz $a[d][d]$, 2 vectores $b[d]$, $c[d]$ y un vector $e[d/2]$; tomando en cuanta esta información, las localidades totales requeridas por la DCT, para una dimensión de bloque $d \times d$, son:

$$loc\_a + loc\_b + loc\_c + loc\_e + loc\_DCT$$
$$(d)(d) + d + d + d/2 + (alto\_i)(ancho\_i)$$
$$d^2 + \frac{5d}{2} + (alto\_i)(ancho\_i)$$

Las operaciones de una DCT son de punto flotante; entonces los elementos anteriores son definidos como flotantes; una variable de este tipo requiere 4 bytes; entonces, el número total de bytes requeridos por la DCT son:

$$4\left(d^2 + \frac{5d}{2} + (alto\_i)(ancho\_i)\right) \qquad \textbf{(5.5)}$$

Por otra parte, una DWT que emplea filtros *wavelet* tipo Haar requiere de dos matrices $a[alto\_i][ancho\_i]$ y $DWT[alto\_i][ancho\_i]$ durante su operación; una de éstas contiene la imagen transformada; entonces, las localidades totales requeridas por la DWT son:

$$loc\_a + loc\_DWT$$
$$(alto\_i)(ancho\_i) + (alto\_i)(ancho\_i)$$
$$2(alto\_i)(ancho\_i)$$

Al igual que una DCT, en una DWT las operaciones son de punto flotante; por tanto las matrices *a* y *DWT* son definidas como flotantes; una variable de este tipo requiere 4 bytes; entonces, el número total de bytes requeridos por la DWT es:

$$8(alto\_i)(ancho\_i) \qquad \textbf{(5.6)}$$

El análisis de la complejidad de espacio para un **TM** fue hecho en la sección 4.4.3.2 obteniendo por resultado:

$$2(3d^2 + (ancho\_i)(alto\_i))$$

La tabla 5.5 incluye la memoria requerida y el número y tipo de operaciones que son necesarias para calcular la transformación de una imagen de 512×512 píxeles, en escala de grises: 8 bits/píxel; para una DCT y **TM** se eligió un tamaño de bloques de 8×8 y para una DWT se elige calcular 3 niveles.

Tabla 5.5. Operaciones y memoria requerida por las **TM**, DCT y DWT para transformar una imagen de 512×512 píxeles, en escala de grises: 8 bits/píxel.

Transformada	Tipo de datos de entrada	Memoria requerida (bytes)	Tipo de datos de salida	Número y tipo de operaciones
DCT bloques 8x8	Enteros	1,048,912	Flotantes	1,900,544 sumas, 327,680 multiplicaciones
DWT Filtros Haar, 3 niv.	Enteros	2,097,152	Flotantes	1,032,192 sumas, 688,128 multiplicaciones
TM bloques 8x8	Enteros	524,672	Enteros	2,097,152 sumas, 2,097,152 comparaciones

Para poder obtener conclusiones de los resultados de este análisis es necesario tomar en cuenta los siguientes aspectos:

- En todos los casos, por lo menos un operador de las multiplicaciones está expresado como un número de punto flotante.
- Como se mencionó anteriormente, un filtro *wavelet* tipo Haar es uno de los más simples. Normalmente esquemas de compresión de imágenes estándares hacen uso de filtros *wavelet* de mayor complejidad; un ejemplo significativo es el JPEG 2000; el banco de filtros de este estándar está formado por 9 filtros FIR pasa-bajos y 7 filtros FIR pasa-altos [2]; los filtros *wavelet* usados por este estándar son de tipo Daubechies [21].
- Las operaciones utilizadas por una **TM** son más simples que las requeridas por la DCT y DWT.
- Todos los operadores que intervienen en las operaciones hechas para el cálculo de la **TM** son de tipo entero.
- El espacio de memoria requerido durante el cálculo de una **TM** es menor al necesitado por una DCT o una DWT.

Con base en estas consideraciones, los resultados obtenidos en este análisis y los obtenidos en los experimentos anteriores, podemos obtener la siguiente conclusión general: la **TM** resulta ser un proceso de transformación más eficiente en los parámetros de velocidad de procesamiento y cantidad de memoria requerida durante su funcionamiento, con respecto a los métodos de transformación tradicionales, la DCT y la DWT, mientras que un sistema de compresión basado en la **TM** demostró una alta competitividad en las características de compresión y PSNR comparado con métodos tradicionales basados en la DCT o la DWT.

Capítulo 6

Conclusiones y trabajo futuro

Este capítulo contiene las conclusiones a las que se ha llegado después de analizar el modelo propuesto en este trabajo de tesis, tras comparar los resultados obtenidos al aplicarlo en un sistema real con técnicas tradicionales. Finalmente, se proponen las perspectivas o trabajos futuros que plantea la continuación de esta investigación.

6.1 Conclusiones

1. Un estudio profundo del estado del arte de las diversas técnicas utilizadas en las etapas que forman un compresor de imágenes, permitió conocer las características que debe reunir un algoritmo para poder ser considerado competitivo en esta área de las ciencias.
2. La *Transformada Morfológica*, producto original generado en este trabajo de tesis, es propuesta como una alternativa a las transformadas tradicionales; el algoritmo de este modelo hace uso de las Memorias Asociativas Morfológicas permitiendo generar una representación morfológica de la imagen procesada. Una Memoria Asociativa Morfológica tiene una baja complejidad computacional al basar su funcionamiento en las operaciones morfológicas de dilatación y erosión, es decir, hacen uso de máximos y mínimos de sumas. Esta característica permite que una MAM tenga una alta velocidad de procesamiento, misma propiedad que le hereda a la *Transformada Morfológica*.

3. Se definieron dos tipos de *Transformada Morfológica*: *max* y *min*. Al tener como base a las Memorias Heteroasociativas Morfológicas del mismo nombre, éstas heredan la robustez al ruido aditivo o sustractivo respectivamente; esta propiedad permite a la *Transformada Morfológica* minimizar los efectos de distorsión que un proceso de cuantificación genera sobre la imagen recuperada.
4. La formulación de un nuevo teorema referente a las Memorias Asociativas Morfológicas permitió determinar la distorsión que un proceso de cuantificación introduce en una imagen procesada por la *Transformada Morfológica*. Este teorema muestra cómo se verá afectado el proceso de recuperación de patrones cuando existe ruido en la memoria asociativa y no los propios patrones.
5. La *Transformada Morfológica* es un proceso que se aplica a bloques de la imagen; el desarrollo de un algoritmo permitió aplicar esta transformada a una imagen completa.
6. El estudio de la complejidad del algoritmo de la *Transformada Morfológica* que se aplica a una imagen completa, permitió concluir que se trata de un algoritmo de complejidad $O(n^2)$; además, se determinó que el número de bytes totales requeridos por el algoritmo depende directamente de la dimensión de la imagen y del tamaño del sub-bloque de imagen elegido para la transformación de la imagen.
7. El uso de la *Transformada Morfológica* en la etapa de transformación de un compresor de imágenes demostró una alta competitividad en su rendimiento comparado con métodos tradicionales basados en la DCT (JPEG) o en la Transformada *Wavelet* (EZW, SPIHT y EBCOT). La principal ventaja que la *Transformada Morfológica* ofrece, con respecto a métodos tradicionales de transformación, es la velocidad de procesamiento y bajos requerimientos de memoria durante su funcionamiento, demostrando además una alta competitividad en los parámetros de compresión y relación señal a ruido.
8. La técnica de codificación que ofrece una mejor relación de compresión, al aplicarla sobre los resultados generados por la *Transformada Morfológica*, es el algoritmo estadístico PPM. Este algoritmo es un método estadístico adaptativo de compresión de datos que basa su operación en la predicción por coincidencia parcial, es decir, PPM predice el valor de un elemento con base en secuencias previas de elementos.

6.2 Trabajo futuro

- Una mejor respuesta en la relación señal a ruido dentro de un codificador basado en memorias asociativas puede ser alcanzada utilizando memorias asociativas bidireccionales, debido a que el proceso de cuantificación introduce ruido mixto en la imagen transformada.
- Implementar la *Transformada Morfológica* mediante otro tipo de memorias asociativas.
- Con este trabajo de investigación queda abierta una línea de investigación de gran envergadura: el estudio de matrices de transformación óptimas para alcanzar mejores resultados en los parámetros de la relación de compresión y de la relación señal a ruido.
- La cuantificación vectorial, al basar su funcionamiento en un algoritmo de búsqueda de igualación, lo hace un proceso lento y de una complejidad computacional considerable. Se propone la implementación de una cuantificación vectorial basada en memorias asociativas con la intención de volverlo un proceso más eficiente y de una baja complejidad computacional.

6.3 Artículos publicados en revistas

- JCR (*Journal Citation Reports*):
  - Guzmán E., Pogrebnyak O., Yáñez C. and Sánchez L.P. "Morphological Transform for Image Compression", EURASIP *Journal on Advances in Signal Processing*, Hindawi. ISSN: 1687-6172, 2008.
- ISI (*Institute for Scientific Information*):
  - Guzmán E., Pogrebnyak O. and Yáñez C. "Image Compression Algorithm Based on Morphological Associative Memories", *Lecture Notes in Computer Science*, LNCS 4225, Springer-Verlag Berlin Heidelberg, pp. 519-528. ISSN: 0302-9743, 2006.
- Internacionales con arbitraje:
  - Guzmán E., Pogrebnyak O. and Yáñez C. "Design of an Evolutionary Codebook Based on Morphological Associative Memories", *Lecture Notes in Artificial Intelligence*. LNAI 4827, Springer-Verlag Berlin Heidelberg, pp. 601-611. ISSN: 0302-9743, 2007.
  - Guzmán E., Pogrebnyak O. and Yáñez C. "A Fast Search Algorithm for Vector Quantization based on Associative Memories", Accepted in: 13th Iberoamerican Congress on Pattern Recognition. *Lecture Notes in Computer Science*. LNCS, Springer-Verlag Berlin Heidelberg, 2008.
- Congresos Nacionales:
  - Guzmán E., Pogrebnyak O. y Yáñez C. "Compresión de Imágenes utilizando Memorias Asociativas Morfológicas". En: *Proc. de la Decimosexta Reunión de Otoño de Comunicaciones, Computación, Electrónica y Exposición Industrial* "La Robótica en la Era Digital", ROC&C'2005, IEEE Sección México, Acapulco, Guerrero, México, 2005.

Referencias

[1] Acevedo M. E. “Memorias asociativas bidireccionales Alfa-Beta”. Tesis doctoral. Centro de Investigación en Computación, IPN. 2006.

[2] Acharya T. and Tsai P-S. *JPEG 2000 Standard for Image Compression. Concepts, Algorithms and VLSI Architectures*. John Wiley & Sons. USA. ISBN 0-471-48422-9. 2005.

[3] Ahmed N., Natarajan T. and Rao K. R. “Discrete Cosine Transform”, *IEEE Transaction on Computers*, Vol. 23, No 1, pp. 90-93. 1974.

[4] Al-Mualla M., Nishan C. and Bull D. *Video Coding for Mobile Communications*. Ed. Academic Press. USA. ISBN: 0 12 053079 1. 2002.

[5] Amari S. “Learning Patterns and Pattern Sequences by Self-Organizing Nets of Threshold Elements”. *IEEE Transaction on Computers*, Vol. C-21, 11, pp. 1197-1206. 1972.

[6] Amerijckx C., Legat J.-D. and Verleysen M. “Image Compression using Self-Organizing Maps”. *Taylor & Francis Ltd., Systems Analysis Modelling Simulation*, Vol. 43, No. 11, pp. 1529-1543. 2003.

[7] Amerijckx C., Verleysen M., Thissen P. and Legat J.-D. “Image Compression by Self-Organized Kohonen Map”. *IEEE Transactions on Neural Networks*, Vol. 9, No 3, pp. 503-507. 1998.

[8] Anderson H. L. and Davidon W. C. “Machine analysis of pion scattering by the maximum likelihood method”. *Il Nuovo Cimento*, Serie X, Vol. 5, pp. 1238-1255. 1957.

[9] Anderson J. A. “Models of interactive memory”. In: *Proceedings of the fifth Asilomar Conference on Circuits and Systems*. S. Parker (Ed.). Los Angeles, California: Western Periodicals. 1972.

[10] Antonini M., Barlaud M., Mathieu P. and Daubechies I. “Image Coding Using Wavelet Transform”. *IEEE Transactions on Image Processing*, Vol. 1, No. 2, pp. 205-220. 1992.

[11] Arai Y., Agui T. and Nakajima M. “A Fast DCT-SQ Scheme for Image”. *Transactions of the IEICE*, Vol. E 71, No 11, pp. 1095-1097. 1988.

[12] Bogdan A. and Meadows H. E. “Kohonen neural network for image coding based on iteration transformation theory”. *SPIE Proceedings on Neural and Stochastic Methods in Image and Signal Processing*, Vol. 1766, pp. 425-436. 1992.

[13] Bovik A. *Handbook of Image and Video Processing*. Academic Press. Canada. ISBN 0-12- 119790-5. 2000.

[14] Burges C. J., Malvar H. S. and Simard P. Y. *Improving Wavelet Image Compression with Neural Networks*. Microsoft Corporation, Microsoft Research, Technical Report MSR-TR-2001-47, 18 p. 2001.

[15] Castellanos C., Díaz De León J. L. y Sánchez A. “Análisis Experimental con las Memorias Asociativas Morfológicas”. En: *XXI Congreso Internacional de Ingeniería Electrónica, Electro 99*, Instituto Tecnológico de Chihuahua, Chih., México, pp. 11-16. 1999.

[16] Chen W. H., Smith C. H. and Fralick S.C. “A Fast Computational Algorithm for the Discrete Cosine Transform”. *IEEE Transactions on* Communications, Vol. COM-25, pp. 1004-1009. 1977.

[17] Chistopoulos C.A., Hebrahimi T. and Skodras A. N. “JPEG2000: The New Still Picture Compression Standard”. In: *Proceedings of the 2000 ACM Workshop on Multimedia*. 2000.

[18] Cormen T. H., Leiserson C. E. and Rivest R. L. *Introductions to Algorithms*. McGraw-Hill. USA. ISBN 0-07-013143-0. 1995.

[19] Cosman P., Gray R. and Olshen R. *Handbook of Medical Imaging Processing and Analisis*. Academic Press. USA. ISBN 0-12-077790-8. 2000.

[20] Danchenko P., Lifshits F., Orion I., Koren S., Solomon A. D. and Mark S. “NNIC—neural network image compressor for satellite positioning system”. *Elsevier Ltd, International Academy of Astronautics*, Vol. 60, Issues 8-9, pp. 622-630. 2007.

[21] Daubechies I. “The Wavelet Transform, Time-Frequency Localization and Signal Analysis”. *IEEE Transaction on Information Theory*, Vol. 36, No. 5, pp. 961-1005. 1990.

[22] Davidon W.C. *Variable metric methods for minimization*. Research and Development Report ANL-5990 (Rev.), Argonne National Laboratory, Argonne, Illinois, 1959.

[23] DeVore R. A., Jawerth B. and Lucier B. J. “Image Compression Through Wavelet Transform Coding”. *IEEE Transactions on Information Theory*, Vol 38, No. 2, pp 719-746. 1992.

[24] Equitz W. H. “A New Vector Quantization Clustering Algorithm”. *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 37, No. 10, pp. 1568-1575. 1989.

[25] Flanagan J. K., Morrell D. R., Frost R. L., Read C. J. and Nelson B.E. “Vector Quantization Codebook Generation Using Simulated Annealing”. In: *IEEE Proc ICASPP`89*, pp. 1759-1762. 1989.

[26] Flores R. “Memorias asociativas Alfa-Beta basadas en el código Johnson-Möbius modificado”. Tesis de maestría. Centro de Investigación en Computación, IPN. 2006.

[27] Ghanbari M. *Standard Codecs: Image Compression to Advanced Video Coding*. The Institution of Electrical Engineers, United Kingdom. ISBN 0-85296-710-1. 2003.

[28] Giaime Ginesu F. M. and Giusto D. “A Multi-factors Approach for Image Quality Assessment based on a Human Visual System Model”. *Elsevier Signal Processing: Image Communications*, No. 21, pp. 316-333, 2006.

[29] Gonzalez R. C. and Woods R. E. *Digital Image Processing*. Prentice Hall, 2nd edition, USA. ISBN-13: 978-0131687288. 2002.

[30] Gray R. M. “Vector Quantization”. *IEEE ASSP Magazine*, 1984, Vol. 1, pp. 4-29. 1984.

[31] Gray R. M. and Neuhoff D. L. “Quantization”. *IEEE Transactions on Information Theory*, Vol. 44, No. 6, pp. 2325-2383. 1998.

[32] Guerequeta R. y Vallecillo A. *Técnicas de Diseño de Algoritmos*. Servicio de Publicaciones de la Universidad de Málaga. España. ISBN 84-7496-666-3. 1998.

[33] Hebb D. O. *The Organization of Behavior: A Neuropsychological Theory*. John Wiley and Sons, New York. ISBN-13: 978-0805843002. 1949.

[34] Hopfield J. J. “Neural Networks and Physical Systems with Emergent Collective Computational Abilities”. *Proceedings of the National Academy of Sciences of the USA*, Vol. 79, pp. 2554-2558. 1982.

[35] Huang C. M. and Harris R. W. “A Comparison of Several Vector Quantization Codebook Generations Approaches”. *IEEE Transactions on Image Processing*, Vol. 2, No. 1, pp. 108-112. 1993.

[36] Huffman D. A. “A Method for the Construction of Minimum-Redundancy Codes”. *Proceedings of the Institute of Radio Engineers*, 40(9), pp. 1098-1101. 1952.

[37] ISO/IEC IS 10918-1 | CCITT T.81: *Digital Compression and Coding of Continuous-Tone Still Image*. ISO/IEC. 1992.

[38] ISO/IEC JTC1/SC29/WG1 N1646R: *JPEG2000 Final Committee Draft v 1.0*. ISO/IEC. 2000.

[39] ISO/IEC JTC1/SC29/WG1 N505: *Call for Contributions for JPEG2000 (JTC 1.29.14, 15444): Image Coding System*. ISO/IEC. 1997.

[40] ISO/IEC JTC1/SC29/WG1 N943: *JPEG2000 Requirement and Profiles*. ISO/IEC. Copenhagen. 1999.

[41] Jain A. K. “A sinusoidal family of unitary transforms”. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 1, No. 4, pp. 356-365. 1979.

[42] Khambete M. and Joshi M. “Blur and Ringing Artifact Measurement in Image Compression using Wavelet Transform”. *Proceedings of World Academy of Science, Engineering and Technology*, Vol. 20, pp. 183-186. 2007.

[43] Kohonen T. “Correlation Matrix Memories”. *IEEE Transaction on Computers*, Vol. C-21, pp. 353-359. 1972.

[44] Kohonen T. “Automatic formation of topological maps of patterns in a self-organizing system”. In: *Proceedings of 2SCIA, Scand. Conference on Image Analysis*, Oja, E. and Simula, O., editors, Helsinki, Finland, pp. 214-220. 1980.

[45] Kohonen T. “Self-organizing formation of topologically correct feature maps”. *Biological Cybernetics*, 43(1), pp. 59-69. 1982.

[46] Kohonen T. *Self-Organizing Maps*. Ed. Springer-Verlag Berlin Heidelberg. Third edition. Germany. ISBN 3-540-67921-9. 2001.

[47] Kok C. W. “Fast Algorithm for Computing Discrete Cosine Transform”. *IEEE Transactions on Signal Processing*, Vol. 45, No. 3, pp 757-760. 1997.

[48] Kosko B. “Bidirectional Associative Memories”. *IEEE Transaction on Systems, Man and Cybernetics*, Vol. 18, No. 1, pp 49-60. 1988.

[49] Kuri A. y Galaviz J. *Algoritmos Genéticos*. Instituto Politécnico Nacional - Centro de Investigación en Computación (CIC), Universidad Autónoma de México, Fondo de Cultura Económica, D.R. ©, México. ISBN: 968-16-6383-7. 2002.

[50] Lakac R. and Plataniotis K. N. *Color Image Processing Methods and Applications*. CRC Press, Taylor & Francis. USA. ISBN 978-0-8493-9774-5. 2007.

[51] Lewis A. S. and Knowles G. “Image Compression Using the 2-D Wavelet Transform”. *IEEE Transactions on Image Processing*, Vol. 1, No. 2, pp. 244-250. 1992.

[52] Linde Y., Buzo A. and Gray R. “An Algorithm for Vector Quantizer Design”. *IEEE Transactions on Communications*, Vol 28, No. 1, pp. 84-95. 1980.

[53] Maeder M. “The image Importance Approach to Human Vision based Image Quality Characterzation”. *Elsevier Pattern Recognition Letters*, No. 26, pp. 347-354. 2005.

[54] Mallat S. “A Theory for Multiresolution Signal Decomposition: The Wavelet Representation”. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 7, pp. 674-693. 1989.

[55] McCulloch, W. S. and Pitts, W. H. “A logical calculus of the ideas immanent in nervous activity”. *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133. 1943.

[56] Minasyan S., Astola J. and Guevorkian D. “On a Class of Parametric Transforms and Its Application to Image Compression”. *EURASIP Journal on Advances in Signal Processing*, Hindawi Publishing Corporation, Vol. 2007, Article ID 58416, 14 pages. 2007.

[57] Mokhtari M. and Boukelif A. “Optimization of Fractal Image Compression based on Kohonen Neural Networks”. In: *The second IEEE-EURASIP International Symposium on Control, Communications, and Signal Processing*. Marrakech, Morocco, 2006.

[58] Nait H. and Salam F. M. “Neural networks-based image compression system”. In: *Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems*, Vol. 2, pp. 846-849. 2000.

[59] Nelson M., and Gailly J-L. *The Data Compression Book*. Ed. M&T Books. Second edition. USA. ISBN 1558514341. 1996.

[60] Ng K. S. and Cheng L. M. "Artificial Neural Network for Discrete Cosine Transform and Image Compression". In: *Proceedings of the 4th International Conference on Document Analysis and Recognition, IEEE Computer Society*, Vol. 2, pp. 675-678. 1997.

[61] Odegard J., Burrus C. "Smooth Biorthogonal Wavelet for Applications in Image Compression. In: *Proceedings IEEE Digital Signal Processing Workshop*. 1996.

[62] Ouafi A., Taleb A., Baarir Z. and Zitouni A. "A Modified Embedded Zerotree Wavelet (MEZW) Algorithm for Image Compression". *Journal of Mathematical Imaging and Vision*, Vol. 30, No. 3, pp 298-307. 2008.

[63] Panchanathan S., Yeap Tet H. and Pilache B. "Neural Network for Image Compression" *Proceeding SPIE Applications of Artificial Neural Networks*, Vol. 1709, pp. 376-385. 1992.

[64] Pan W. "A Fast 2-D DCT algorithm Via Distributed Arithmetic Optimization". *IEEE Proceeding Image Processing*, Vol. 3, pp 114-117. 2000.

[65] Pinson M. and Wolf S. "A new standardized method for objectively measuring video quality". *IEEE Transactions on Broadcasting*, Vol. 50 No. 3 pp 312–322. 2004.

[66] Pratt W. K. *Digital Image Processing*. John Wiley and Son, Inc, Third Edition, USA, ISBN 0-471-37407-5. 2001.

[67] Qiang Ji. "Image compression using a self-organized neural network". *SPIE Proceeding on Applications of Artificial Neural Networks in Image Processing*, Vol. 3030, p. 56-59. 1997.

[68] Ramírez J., García A., Frenadse P. G., Parrilla L. and Lloris A. "A New Architecture to Compute the Discrete Cosine Transform Using the Quadratic Residue Number System". In: *ISCAS 2000, IEEE International Symposium on Circuits and Systems*, 2000.

[69] Rao K. R. and Yip P. C. *The Transform and Data Compression Handbook*. The Electrical Engineering and Signal Processing Series. ISBN 0-8493-3692-9. 2001.

[70] Richardson I. E. G. *Video Codec Design, Developing Image and Video Compression Systems*. John Wiley & Sons. Great Britain. ISBN 0-471-48553-5. 2003.

[71] Richardson I. E. G. *H.264 and MPEG-4 Video Compression*. John Wiley & Sons. Great Britain. ISBN 0-470-84837-5. 2003.

[72] Ritter G. X., Díaz de León J. L. and Sussner P. "Morphological Bidirectional Associative Memories". *Neural Networks*, Vol. 12, No. 6, pp. 851-867. 1999.

[73] Ritter G. X. and Sussner P. "An Introductions to Morphological Neural Networks". In: *Proceedings of the 13th International Conference on Pattern Recognition*, Vol. IV, Track D, pp. 709-717. 1996.

[74] Ritter G. X., Sussner P. and Díaz de León J. L. "Morphological Associative Memories". *IEEE Transactions on Neural Networks*, Vol. 9, No. 2, pp. 281-293. 1998.

[75] Ritter G. X. and Wilson J. N. "Image Algebra and its Relationship to Neural Networks". *SPIE Conf. Proc. Aerospace Pattern Recognition*, Vol. 1098, Orlando, pp. 90-101. 1989.

[76] Roy S. B., Kayal K. and Sil J. "Edge Preserving Image Compression Technique using Adaptive Feed Forward Neural Network". In: *EuroIMSA, European Internet and Multimedia Systems and Applications*, Grindelwald, Switzerland, 2005.

[77] Said A. and Pearlman W. A. "A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees". *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 6, No. 3, pp. 243-250. 1996.

[78] Sánchez F. A., Díaz de León J. L. and Yánez C. "Lernmatrix de Steinbuch: Avances Teóricos". *Computación y Sistemas*, CIC-IPN, Vol. 7, No. 3, pp. 175-189. 2004.

[79] Setioro R. and Lu G. "Image compression using a feedforward neural network". In: *IEEE International Conference on Neural Networks, IEEE World Congress on Computational Intelligence*, Vol. 7, pp. 4761-4765. 1994.

[80] Shannon C. E. "A Mathematical Theory of Communication". *Bell System Technical Journal*, Vol. 27, pp. 379-423 and 623-656. 1948.

[81] Shapiro J. M. "Embedded Image Coding Using Zerotrees of Wavelet Coefficients". *IEEE Transactions on Signal Processing*, Vol. 41, No. 12, pp. 3445-3462. 1993.

[82] Sherlock B. G. and Monro D. M. "Algorithm 749: Fast Discrete Cosine Transform". *ACM Transactions on Mathematical Software*, Vol. 21, No. 4, pp. 372-378. 1995.

[83] Sossa H. and Barrón R. "New associative model for pattern recall in the presence of mixed noise". In: *Proceedings of the fifth IASTED International Conference on Signal and Image Processing*, Acta press 399, pp. 485-490. 2003.

[84] Sossa H., Barrón R. and Vázquez R. A. "New associative memories for recall real-value patterns". In: *Ninth Iberoamerican Congress on Pattern Recognitions, CIARP* 2004, LNCS 3287, pp. 195-202. 2004.

[85] Sousa C. M., Cavalcante A. B., Guilhon D. and Kardec A. "Image Compression by Redundancy Reduction", ICA 2007, Springer-Verlag Berlin Heidelberg, LNCS 4666, pp. 422–429. 2007.

[86] Symes P. *Video Compression Demystified*. Ed. Mc Graw Hill. USA. ISBN 0-07-136324-6. 2001.

[87] Tanaka T., Kakiya S. and Kabashima Y. "Capacity Analysis of Bidirectional Associative Memory". In: *Proc. Seventh Int. Conf. Neural Information Processing*, Taejon, Korea, Vol. 2, pp. 779-784. 2000.

[88] Taubman D. "High Performance Scalable Image Compression with EBCOT". *IEEE Transactions on Image Processing*, Vol. 9, No. 7, pp. 1158-1170. 2000.

[89] Tseng B. D. and Miller W. C. "On Computing the Discrete Cosine Transform". *IEEE Transactions on Computer*, C-27, 10, pp. 966-968. 1978.

[90] UMTS Forum. *The Future Mobil Market. Global trends and development with a focus on Western Europe*. United Kingdom: UTMS Forum. Report No. 8. 1999.

[91] Vaisey J. and Gersho A. "Simulated Annealing and Codebook Design". In: *IEEE Proc. ICASSP`88*, pp. 1176-1179. 1988.

[92] Wallace G. K. "The JPEG still picture compression standard". *Proceedings of Communications of the ACM*, Vol. 34, No. 4, pp. 30-44. 1991.

[93] Werbos P. *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*, John. Wiley and Sons, Inc., New York, ISBN 0-471-59897-6. 1994.

[94] Winograd S. "On Computing the Discrete Fourier Transform". *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 73, No. 4, pp. 1005-1006. 1976.

[95] Witten I. H., Neal R. M. and Cleary J. G.. "Arithmetic Coding for Data Compression". *Communications of ACM*, Vol. 30, No. 6, pp. 520-540. 1987.

[96] Xiong Z., Guleryuz O. G. and Orchard M. T. "A DCT-Based Embedded Image Coder". *IEEE Signal Processing Letters*, Vol. 3, No. 11, pp. 289-290. 1996.

[97] Xiong Z., Ramchandran K., Orchard M. T. and Zhang Y. "A Comparative Study of DCT- and Wavelet-Based Image Coding". *IEEE Transactions on Circuits and Systems for Video Technology*, Vol 9, No 5, pp. 692-695. 1999.

[98] Yair E., Zeger K. and Gersho A. "Competitive Learning and Soft Competition for Vector Quantizer Design". *IEEE Transactions on Signal Processing*, Vol. 40, No. 2, pp. 294-309. 1992.

[99] Yáñez C. "Memorias Asociativas basadas en Relaciones de Orden y Operadores Binarios". Tesis doctoral. Centro de Investigación en Computación, IPN. 2002.

[100] Yáñez C. y Díaz de León J. L. *Algunas Aportaciones de S. Amari en Memorias Asociativas*. México: Centro de Investigación en Computación, IPN. IT 51, Serie Verde, ISBN 970-18-6691-6. 2001.

[101] Yáñez, C. y Díaz de León J. L. *Introducción a las Memorias Asociativas*. Centro de Investigación en Computación, IPN. México. ISBN 970-36-0116-2. 2003.

[102] Yáñez C. y Díaz de León J. L. *Lernmatrix de Steinbuch*. México: Centro de Investigación en Computación, IPN. IT 48, Serie Verde, ISBN 970-18-6688-6. 2001.

[103] Yáñez C. y Díaz de León J. L. *Linear Associator de Anderson-Kohonen*. México: Centro de Investigación en Computación, IPN. IT 50, Serie Verde, ISBN 970-18-6690-8. 2001.

[104] Yáñez C. y Díaz de León J. L. *Memorias Autoasociativas Morfológicas max: Condiciones Suficientes para Convergencia, Aprendizaje y Recuperación de Patrones*. México: Centro de Investigación en Computación, IPN. IT 175, Serie Azul, ISBN 970-36-0034-4. 2003.

[105] Yáñez C. y Díaz de León J. L. *Memorias Autoasociativas Morfológicas min: Condiciones Suficientes para Convergencia, Aprendizaje y Recuperación de Patrones*. México: Centro de Investigación en Computación, IPN. IT 177, Serie Azul, ISBN 970-36-0036-0. 2003.

[106] Yáñez C. y Díaz de León J. L. *Memorias Morfológicas Autoasociativas*. México: Centro de Investigación en Computación, IPN. IT 58, Serie Verde, ISBN 970-18-6698-3. 2001.

[107] Yáñez C. y Díaz de León J. L. *Memorias Morfológicas Heteroasociativas*. México: Centro de Investigación en Computación, IPN. IT 57, Serie Verde, ISBN 970-18-6697-5, 2001.

[108] Yáñez C., Felipe E. M., I. López y Flores R. “A Novel Approach to Automatic Color Matching”. Springer-Verlag Berlin Heidelberg, LNCS 4225, pp. 529–538. 2006.

[109] Yu S. and Swartzlander E. E. “DCT Implementation with Distributed Arithmetic”. *IEEE Transactions on Computers*, Vol. 50, No. 9, pp 985-991. 2001.

[110] Zhu C., Li L., He Z. and Wang J. “A New Competitive Learning Algorithm for Vector Quantization”. In: *IEEE Proc. ICASSP`94*, pp 557-560. 1994.

Apéndice A

Simbología

$\mathbf{x}$	patrón de entrada, vector columna
$\mathbf{y}$	patrón de salida, vector columna
x_i^{φ}	*i*-ésimo elemento del patrón de entrada $\mathbf{x}^{\varphi}$
n	dimensión de los patrones de entrada, ancho de una imagen
m	dimensión de los patrones de salida, alto de una imagen
$\left(\mathbf{x}^{\mu}\right)^t$	transpuesto del vector $\mathbf{x}^{\mu}$
$\mathbf{I}$	Matriz identidad
$\tilde{\mathbf{x}}$	versión alterada del vector $\mathbf{x}$
$(\mathbf{x},\mathbf{y})$	asociación entre un patrón de entrada y un patrón de salida
$\left\{\left(\mathbf{x}^{\mu},\mathbf{y}^{\mu}\right) \mid \mu = 1,2,...,k\right\}$	conjunto fundamental de asociaciones
$\left(\mathbf{x}^{\mu},\mathbf{y}^{\mu}\right)$	μ-ésimo elemento del conjunto fundamental de asociaciones
A	conjunto al que pertenecen $\mathbf{x}$ y $\mathbf{y}$
$\mathbf{M}$	memoria asociativa, memoria asociativa morfológica *max*
$\hat{\mathbf{M}}$	versión alterada de la memoria asociativa morfológica *max*
m_{ij}	*ij*-ésimo elemento de la matriz $\mathbf{M}$
$\mathbf{W}$	memoria asociativa morfológica *min*
$\hat{\mathbf{W}}$	versión alterada de la memoria asociativa morfológica *min*

w_{ij}	*ij*-ésimo elemento de la matriz **W**
Δm_{ij}	incremento en m_{ij}
$\bigvee$	operador máximo
$\bigwedge$	operador mínimo
∇	producto máximo
Δ	producto mínimo
α y β	operadores en que se basan las memorias Alfa-Beta
$\alpha(x, y)$	operación binaria α con argumentos *x* y *y*
$\beta(x, y)$	operación binaria β con argumentos *x* y *y*
V	memoria asociativa Alfa-Beta tipo *max*
Λ	memoria asociativa Alfa-Beta tipo *min*
$\Cup_{\alpha}$	operador α *max*
$\Cap_{\alpha}$	operador α *min*
$\Cup_{\beta}$	operador β *max*
$\Cap_{\beta}$	operador β *min*
$\boxtimes$	símbolo que representa ambas operaciones: $\Cup_{\alpha}$ y $\Cap_{\alpha}$
A y B	operadores en que se basan las memorias asociativas tipo Mediana
medA $(\lozenge_{A})$	operación mediana A
medB $(\lozenge_{A})$	operación mediana B
$\otimes$	símbolo que puede representar a los operadores $\bigvee$, $\bigwedge$ o $\lozenge$
sb	sub-bloque de imagen
sb_{ij}	*ij*-ésimo elemento de un sub-bloque de imagen
vi	vector de imagen
vi_{i}	*i*-ésimo elemento de un vector de imagen
mt	matriz de transformación
mt_{ij}	*ij*-ésimo elemento de una matriz de transformación
vt	vector de transformación
vt_{i}	*i*-ésimo elemento de un vector de transformación
TM	transformada morfológica
$\mathbf{TM}_{\min}$	transformada morfológica *min*
$\mathbf{TM}_{\max}$	transformada morfológica *max*
TMI	transformada morfológica inversa
$\mathbf{TMI}_{\min}$	transformada morfológica inversa *min*
$\mathbf{TMI}_{\max}$	transformada morfológica inversa *max*
$\mathrm{MMH}^{xy}_{\min}$	*xy*-ésima MMH que forma a una $\mathbf{TM}_{\min}$
$\mathrm{MMH}^{xy}_{\max}$	*xy*-ésima MMH que forma a una $\mathbf{TM}_{\max}$
$O\{\cdot\}$	operador usado para representar la organización que las MMH deben guardar para constituir a la **TM**

$\mathbf{vi}^{\omega_\mu}$	μ-ésima fila del sub-bloque de imagen ω
$T(n)$	tiempo de ejecución de un algoritmo
O	función Omicrón

Apéndice B

Elementos de un sistema de compresión de imágenes

El propósito de este apéndice es describir cada uno de los bloques que integran un sistema de compresión de imágenes, centrándose especialmente en las principales técnicas empleadas en cada una de estas etapas.

B.1 Transformador de imagen

La esencia de una transformada aplicada a una imagen es ofrecer una representación que muestre características que en el dominio original son muy difíciles o sencillamente imposibles de detectar.

El objetivo principal de un transformador de imagen es reducir la dependencia entre los píxeles de una imagen, por lo que este bloque está diseñado para minimizar la redundancia espacial. Un transformador no ofrece compresión de información, su finalidad es facilitar la compresión de la información en los bloques posteriores. La elección de la transformada a emplear depende de varios criterios:

- Los datos en el dominio de la transformada deben de estar separados en componentes con una mínima interdependencia.
- la mayor parte de la energía en los datos transformados debe estar concentrada en un mínimo de valores.

- La transformada debe ser reversible.
- La transformada debe ser computacionalmente tratable (bajos requerimientos de memoria, almacenable usando una precisión aritmética limitada, bajo número de operaciones aritméticas).

Aunque existe una gran variedad de transformadas que pueden ser empleadas en la etapa de transformación de un compresor de imágenes, solo dos de ellas han logrado gran popularidad para este propósito.

Una de ellas es un método de transformación matemática conocido como Transformada Discreta del Coseno (DCT, *Discrete Cosine Transform*). La DCT es miembro de la familia de las transformadas sinusoidales desarrolladas por Jain [41]. Se trata de un mapeo uno a uno, por lo que este proceso es reversible, es decir, no comprime la imagen y no tiene pérdida de información. La DCT es una transformación que se aplica a bloques de la imagen, frecuentemente bloques de 8×8 ó 16×16 [69].

La segunda transformada más frecuentemente utilizada en la compresión de imágenes es una transformada que puede ser aplicada a una imagen completa: la Transformada Discreta *Wavelet* (DWT, *Discrete Wavelet Transform*). Aquí la imagen es procesada en una serie de etapas para producir la descomposición *Wavelet* de la imagen. Esta transformada tiene un alto requerimiento de memoria debido a que trabaja con una imagen completa en lugar de procesarla por bloques.

B.1.1 Transformada Discreta del Coseno

La DCT fue originalmente propuesta por N. Ahmed, T. Natarajan y K. R. Rao en el año 1974 [3]. Esta transformada es ampliamente utilizada en el procesamiento de señales e imágenes, particularmente en el bloque de transformación de sistemas de compresión/descompresión de datos; se ha convertido en una de las transformadas más populares en la compresión de imágenes debido a su efectiva transformación de imágenes a una forma que es fácil de comprimir y a su eficiente implementación en software o hardware.

La DCT tiene dos propiedades muy útiles en la compresión de imágenes: la concentración de la energía de la imagen en un pequeño número de coeficientes (compactación de la energía) y la minimización de la interdependencia de los coeficientes (decorrelación) [70].

La DCT de dos dimensiones (DCT 2-D) y su inversa son usadas como elementos de procesamientos básicos en estándares internacionales de compresión de imágenes y video.

El algoritmo que aplica la DCT en la etapa de transformación de un compresor de imágenes divide la imagen en bloques, la DCT 2-D de un bloque de una imagen, $f_{x,y} \mid x, y = 0,1,..., M-1$, de $M \times M$ muestras es definida por la siguiente relación matemática:

$$F_{i,j} = \frac{4C(i)C(j)}{M^2}\sum_{x=0}^{M-1}\sum_{y=0}^{M-1} f_{x,y}\cos\left(\frac{\pi(2x+1)i}{2M}\right)\cos\left(\frac{\pi(2y+1)j}{2M}\right) \qquad \textbf{(B.1)}$$
$$i, j = 0,1,..., M-1$$

y la DCT inversa (IDCT) 2-D

$$f_{x,y} = \sum_{i=0}^{M-1}\sum_{j=0}^{M-1} C(i)C(j)F_{i,j}\cos\left(\frac{\pi(2x+1)i}{2M}\right)\cos\left(\frac{\pi(2y+1)j}{2M}\right) \qquad \textbf{(B.2)}$$
$$x, y = 0,1,..., M-1$$

donde M es un entero de potencia de 2 y $F_{i,j}$ son los coeficientes resultantes de la DCT 2-D.

$C(p),\ p = i, j,$ son constantes definidas por:

$$C(p) = \begin{cases} 1/\sqrt{2}, & p = 0 \\ 1, & p \neq 0 \end{cases} \qquad \textbf{(B.3)}$$

La IDCT hace uso de un bloque de coeficientes DCT, $F_{i,j}$, para reconstruir un bloque de la imagen, $f_{x,y}$. Cabe mencionar que algunos estándares internacionales de compresión de imágenes y video aplican sobre las ecuaciones B.1 y B.2 un factor de escalamiento de 4.

La DCT es una transformada separable, propiedad que permite el cálculo de una DCT 2-D aplicando la transformada DCT de 1 dimensión primero por filas y después por columnas, figura B.1.

La DCT de 1 dimensión para una secuencia de datos, $f_x \mid x = 0,1,..., M-1$, está definida como:

$$F_i = \frac{2C(i)}{M} \sum_{x=0}^{M-1} f_x \cos\left(\frac{\pi(2x+1)i}{2M}\right) \qquad \textbf{(B.4)}$$
$$i = 0,1,..., M-1$$

donde F_i son los coeficientes de la DCT y $C(i)$ es especificada por la ecuación B.3.

Por tanto, la DCT 2-D puede ser representada como:

$$F_{i,j} = \frac{2C(j)}{M} \sum_{y=0}^{M-1} F_i \cos\left(\frac{\pi(2y+1)j}{2M}\right) \qquad \textbf{(B.5)}$$
$$F_{i,j} = \textbf{DCT 1-D}_{\textbf{dirección y}}\left(\textbf{DCT 1-D}_{\textbf{dirección x}}\right)$$

donde $F_{i,j}$ son los coeficientes de la DCT 2-D, $C(j)$ es especificada por la ecuación B.3 y la variable x de la ecuación B.4 ha sido remplazada por y para indicar una DCT sobre columnas.

De la misma forma, la IDCT 2-D puede ser calculada aplicando la IDCT de 1 dimensión primero sobre filas y luego sobre columnas. La ecuación B.6 define a la IDCT de 1 dimensión.

$$f_x = \sum_{i=0}^{M-1} C(i) F_i \cos\left(\frac{\pi(2x+1)i}{2M}\right) \qquad \textbf{(B.6)}$$
$$x = 0,1,..., M-1$$

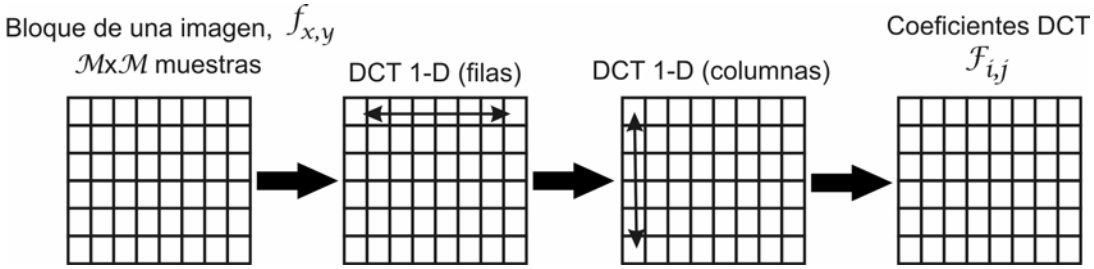


Figura B.1. Esquema de la DCT 2-D usando DCT de 1 dimensión.

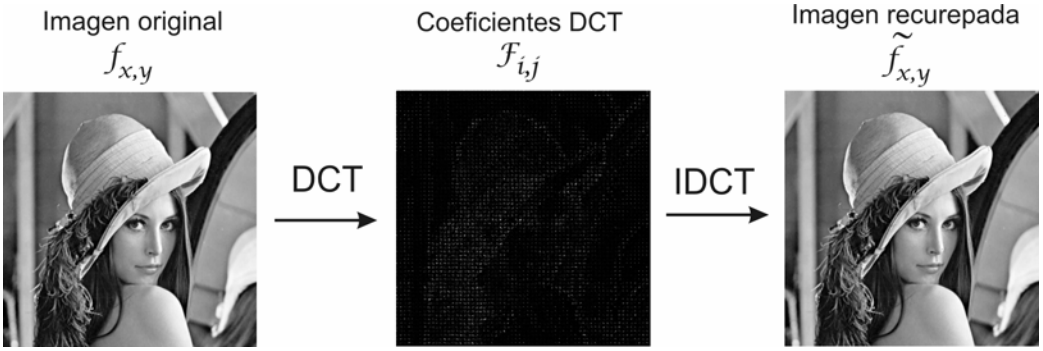


Figura B.2. Proceso de transformación sobre la imagen Lena usando la DCT 2-D.

Como consecuencia del amplio uso que ha tenido la DCT en aplicaciones como el procesamiento de señales, han surgido diversas versiones intentando implementarla con la mayor eficiencia posible. Uno de los primeros algoritmos que describe la implementación rápida de la DCT fue propuesto por Chen en [16]; este algoritmo aprovecha la simetría de la función coseno para reducir el número de operaciones necesarias en el cálculo de esta transformada.

Una variante más expedita es descrita por Arai en [11]. Se trata de un esquema rápido de la DCT aplicado a imágenes, que hace uso únicamente de la parte real de la Transformada Discreta de Fourier (DFT) como lo describe Tseng en [89]; el cálculo de los coeficientes lo hace utilizando el algoritmo de la Transformada Rápida de Fourier (FFT) descrito en [94] por Winograd.

Aunque los métodos mencionados son los más populares, siguen surgiendo nuevas formas de implementar una DCT rápida y eficaz; ejemplos de estas implementaciones son [82], [47], [64] [68] y [109]. Por último, cabe destacar que la DCT forma parte del estándar JPEG [92], [37]. La figura B.2 muestra el resultado de aplicar la DCT 2-D sobre la imagen Lena.

B.1.2 Transformada Discreta Wavelet

En el año 1989 S. Mallat propone la teoría de análisis de señales con base en una descomposición multiresolución de señales usando *wavelets* en el espacio tiempo-escala y propone el algoritmo de la pirámide [54]. El desarrollo histórico y los fundamentos matemáticos de la transformada *wavelet* son descritos por I. Daubechies en [21]. Las aplicaciones de la transformada *wavelet* en el área de compresión de imágenes empiezan en el año 1992 cuando Ronald A. DeVore *et al.* desarrollaron una teoría matemática que permite emplear esta transformada en dicha área [23]. La DWT es un método alternativo a la DCT en la etapa de transformación en un sistema para la compresión de imágenes. Se trata de una transformada que se aplica a toda la imagen, a diferencia de la DCT que se aplica a bloques de la imagen. La DWT es una descomposición de una señal en frecuencias. El uso de esta transformada en el tratamiento de imágenes es adecuado debido a varios factores:

- Al tener las *wavelet* localización espacial y de frecuencia, hace viable el tratamiento de cambios bruscos en una imagen.
- Al ser una transformada que se aplica a toda la imagen, no presenta la distorsión que se produce al dividir a una imagen en bloques para su procesamiento, como lo hace la DCT.
- Se consigue una alta compactación de energía de la imagen.

En el proceso de análisis de las *wavelets*, las señales son representadas utilizando un grupo de funciones básicas producidas por el desplazamiento y escalamiento de una función madre o función principal. Esto es debido a que una señal puede ser representada mediante una secuencia de coeficientes usando series de expansión *wavelets*. La representación de una señal $f(x)$ mediante una serie de expansión *wavelet* usando la *wavelet* $\psi(x)$ y la función de escalamiento $\varphi(x)$, es definida por [29]:

$$f(x) = \sum_k c_{j_0}(k)\varphi_{j_0,k}(x) + \sum_{j=j_0}^{\infty}\sum_k d_j(k)\psi_{j,k}(x) \qquad \textbf{(B.7)}$$

donde j_0 es una escala arbitraria de inicio, $c_{j_0}(k)$ son los coeficientes de escalamiento o aproximación y $d_j(k)$ son referenciados como coeficientes detalle o *wavelets*. Estas definiciones parten de los siguientes hechos: primero, la sumatoria de la izquierda en la ecuación B.7 usa funciones de escalamiento para proveer una aproximación de $f(x)$ en escala j_0; segundo, para la escala $j \geq j_0$ en la segunda sumatoria una función de resolución fina, una suma de *wavelets* es añadida a la aproximación para incrementar el detalle de $f(x)$. Los coeficientes de expansión son calculados mediante:

$$c_{j_0}(k) = \langle f(x), \varphi_{j_0,k}(x) \rangle = \int f(x)\varphi_{j_0,k}(x)dx \qquad \textbf{(B.8)}$$

$$d_j(k) = \langle f(x), \psi_{j,k}(x) \rangle = \int f(x)\psi_{j,k}(x)dx \qquad \textbf{(B.9)}$$

Cuando se representa una señal mediante series de expansión *wavelets* como muestras de una función continua $f(x)$, los coeficientes resultantes son nombrados DWT de $f(x)$, entonces la series de expansión definida en las ecuaciones B.7, B.8 y B.9 llega a definir el proceso de análisis de la DWT de 1 dimensión [29]:

$$W_\varphi(j_0,k) = \frac{1}{\sqrt{M}}\sum_x f(x)\varphi_{j_0,k}(x) \qquad \textbf{(B.10)}$$

$$W_\psi(j,k) = \frac{1}{\sqrt{M}}\sum_x f(x)\psi_{j,k}(x) \qquad \textbf{(B.11)}$$

para $j \geq j_0$; el proceso de síntesis de la DWT de 1 dimensión es definido por:

$$f(x) = \frac{1}{\sqrt{M}}\sum_k W_\varphi(j_0,k)\varphi_{j_0,k}(x) + \frac{1}{\sqrt{M}}\sum_{j=j_0}^{\infty}\sum_k W_\psi(j,k)\psi_{j,k}(x) \qquad \textbf{(B.12)}$$

Donde $f(x)$, $\varphi_{j_0,k}(x)$ y $\psi_{j,k}(x)$ son funciones de la variable discreta x, M es el número de muestras de $f(x)$, entonces $x = 0,1,...,M-1$, $j_0 = 0$, $j = 0,1,...,J-1$, $J = \log_2 M$ y $k = 0,1,...,2^j - 1$. La transformada es compuesta de M coeficientes, la escala mínima es 0 y la escala máxima $J-1$. $W_\varphi(j_0,k)$ y $W_\psi(j,k)$ son los coeficientes de aproximación y de detalle, respectivamente.

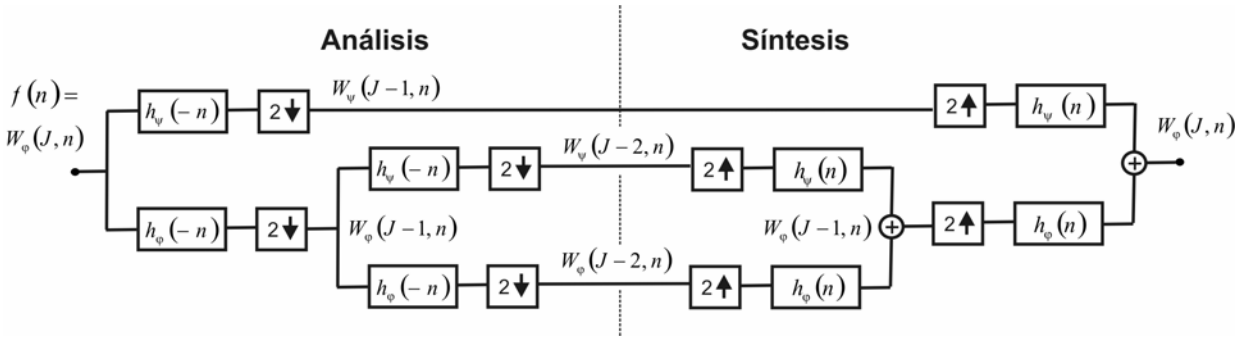


Figura B.3. Banco de análisis y síntesis de una FWT de dos escalas.

Una versión computacionalmente eficiente de la DWT es la Transformada Rápida *Wavelet* (FWT, *Fast Wavelet Transform*) también conocida como algoritmo *herringbone* de Mallat [54]. La FWT es definida por las siguientes expresiones:

$$W_\psi(j,k)=\sum_m h_\psi(m-2k)W_\varphi(j+1,m) \qquad \textbf{(B.13)}$$

$$W_\varphi(j,k)=\sum_m h_\varphi(m-2k)W_\varphi(j+1,m) \qquad \textbf{(B.14)}$$

h_φ es referenciado como un vector de escalamiento, $h_\varphi(n)$ son los coeficientes de la función de escalamiento; h_ψ es referenciado como un vector de detalle, $h_\psi(n)$ son los coeficientes de la función de detalle.

Las ecuaciones B.13 y B.14 muestran una estrecha relación existente entre coeficientes de escalas adyacentes; los coeficientes de aproximación y detalle de la DWT de la escala j están expresados en función de los coeficientes de aproximación de la DWT de la escala $j+1$.

Los coeficientes de la DWT de la escala j también pueden ser calculados mediante la convolución de los coeficientes de aproximación de la escala $j+1$ con los vectores de escalamiento y detalle invertidos en tiempo, $h_\varphi(-n)$ y $h_\psi(-n)$, y el sub-muestreo por 2 de los resultados:

$$W_\psi(j,k)=h_\psi(-n)*W_\varphi(j+1,n)\big|_{n=2k,\ k\geq 0} \qquad \textbf{(B.15)}$$

$$W_\varphi(j,k)=h_\varphi(-n)*W_\varphi(j+1,n)\big|_{n=2k,\ k\geq 0} \qquad \textbf{(B.16)}$$

La convolución es evaluada en instantes $n=2k,\ k\geq 0$. La figura B.3 muestra el diagrama de bloques de los procesos de análisis y síntesis de la FWT. En el proceso de análisis la figura muestra un banco de filtros de dos escalas; se asume que la escala más alta, $W_\varphi(J,n)$, son las muestras de función $f(x)$. El primer banco de filtros divide la función original en componentes de aproximación $W_\varphi(J-1,n)$, a través de un filtro pasa-bajos, y en componentes detalle $W_\psi(J-1,n)$, haciendo uso de un filtro pasa altos; el proceso se repite con el segundo banco aplicándolo a $W_\varphi(J-1,n)$ para obtener $W_\varphi(J-2,n)$ y $W_\psi(J-2,n)$.

La FWT inversa (FWT^{-1}) es definida por:

$$W_\varphi(J+1,k)=h_\varphi(k)*W_\varphi^{\mathbf{up}}(j,k)+h_\psi(k)*W_\psi^{\mathbf{up}}(j,k)\big|_{k\geq 0} \qquad \textbf{(B.17)}$$

Debido a que una imagen es una señal de dos dimensiones, $f(x,y)$, para el tratamiento de éstas se hace uso de la DWT 2-D, la cual requiere de una función de escalamiento bidimensional $\varphi(x,y)$ y de tres *wavelets* bidimensionales o partes detalle direccionalmente sensibles $\psi^H(x,y)$, $\psi^V(x,y)$ y $\psi^D(x,y)$. Las *wavelet* miden las variaciones de intensidad o nivel de gris en una imagen, ψ^H mide las variaciones a lo largo de las columnas (bordes horizontales), ψ^V a lo largo de los renglones (bordes verticales) y ψ^D variaciones diagonales. Las funciones básicas de traslación y escalamiento son:

$$\varphi_{j,m,n}(x,y) = 2^{j/2}\varphi\left(2^j x - m, 2^j y - n\right) \tag{B.18}$$

$$\varphi^i_{j,m,n}(x,y) = 2^{j/2}\psi^i\left(2^j x - m, 2^j y - n\right) \qquad i = \{H,V,D\} \tag{B.19}$$

La DWT de una función $f(x,y)$ de tamaño $M \times N$ es definida por

$$W_\varphi(j_0,m,n) = \frac{1}{\sqrt{MN}}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\varphi_{j_0,m,n}(x,y) \tag{B.20}$$

$$W^i_\psi(j,m,n) = \frac{1}{\sqrt{MN}}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\varphi^i_{j,m,n}(x,y) \qquad i = \{H,V,D\} \tag{B.21}$$

Los coeficientes $W_\varphi(j_0,m,n)$ definen la aproximación de $f(x,y)$ en la escala j_0, los coeficientes $W^i_\psi(j,m,n)$ los detalles horizontales, verticales y diagonales para escalas $j \geq j_0$; normalmente $j_0 = 0$, $N = M = 2^J$, $j = 0,1,...,J-1$, $J = \log_2 M$ y $m,n = 0,1,...,2^j - 1$. La DWT 2-D inversa es:

$$\begin{aligned} f(x,y) &= \frac{1}{\sqrt{MN}}\sum_m\sum_n W_\varphi(j_0,m,n)\varphi_{j_0,m,n}(x,y) \\ &+ \frac{1}{\sqrt{MN}}\sum_{i=H,V,D}\sum_{j=j_0}^{\infty}\sum_m\sum_n W^i_\psi(j,m,n)\psi^i_{j,m,n}(x,y) \end{aligned} \tag{B.22}$$

La figura B.4 muestra el diagrama de bloques de los procesos de análisis y síntesis de la DWT 2-D; ésta puede ser implementada tomando ventaja de la propiedad de separabilidad; simplemente se aplica la FWT 1-D sobre las filas de $f(x,y)$ seguida por una FWT 1-D sobre las columnas, ambas seguidas de una reducción en la resolución por un factor de 2.

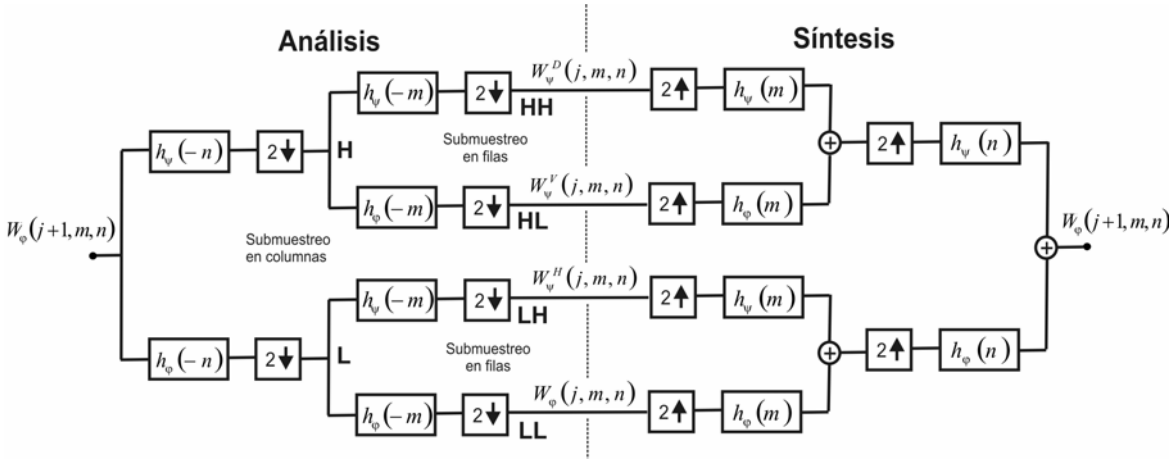


Figura B.4. Banco de análisis y síntesis de la DWT 2-D usando dos FWT 1-D.

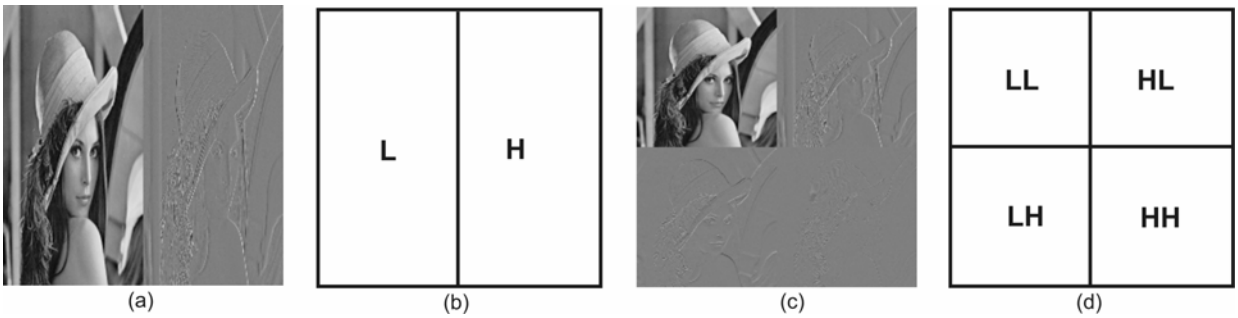


Figura B.5. DWT sobre la imagen Lena: (a) DWT aplicada sobre las filas, (b) Bloques de coeficientes de frecuencias bajas (**L**) y altas (**H**), (c) DWT aplicada sobre filas y columnas, (d) Bloques de las diferentes frecuencias.

En la etapa de análisis mostrada en la figura B.4, el resultado correspondiente al primer banco de filtros y un sub-muestreo sobre las columnas por un factor de 2 es interpretado como una imagen compuesta por una versión extendida y escalada de la original y de los detalles (figura B.5(a)); en la figura B.5(b) se representa esta interpretación como los bloques de los coeficientes de baja y alta frecuencia, denotados por **L** y **H** respectivamente.

A estos resultados se les aplica un segundo banco de filtros y un sub-muestreo sobre las filas por un factor de 2. El resultado es mostrado en la figura B.5(c); éste es descompuesto en cuatro cuadrantes, figura B.5(d):

- **LL**. Formado por los coeficientes obtenidos de aplicar 2 filtros pasa-bajos, en un primer plano sobre las columnas y en un segundo sobre las filas. Este sub-bloque es un suavizado de baja resolución de la imagen original.
- **HL**/**LH**. Estos bloques son filtrados a lo largo de columnas y filas con un filtro pasa-bajos y un filtro pasa-altos en una forma alternada. El bloque **LH** contiene los bordes verticales. En contraste, el boque **HL** muestra los bordes horizontales.
- **HH**. Este bloque es obtenido tras aplicar 2 filtros pasa-altos y puede ser interpretado como el área donde se encuentran los bordes en dirección diagonal de la imagen original.

De los primeros trabajos que hacen referencia al uso de la DWT en compresión de imágenes, destacan dos presentados en el año 1992. En el primero de ellos M. Antonini *et al.* propusieron un esquema para la compresión de imágenes usando la DWT [10]; este esquema obtiene un conjunto de subclases de imágenes biortogonales, donde la imagen original es descompuesta en diferentes escalas usando un algoritmo de arquitectura piramidal. Esta descomposición provee sub-imágenes correspondientes a diferentes niveles de resolución y orientación. Los coeficientes *wavelet* son cuantificados vectorialmente utilizando un libro de códigos multiresolución.

En el segundo trabajo, A. S. Lewis y G. Knowles proponen un esquema donde la imagen es comprimida a través de su descomposición en coeficientes espaciales y espectrales con base en la transformada *wavelet* 2-D [51]. Los coeficientes *wavelet* son cuantificados con un modelo basado en el HVS. Para la codificación, la estructura de descomposición es representada mediante un árbol. Partiendo del hecho que cada coeficiente en las bandas pasa- altos de los coeficientes *wavelet* tiene cuatro coeficientes relacionados con él; cada coeficiente del árbol es

codificado si su valor absoluto es mayor a un límite fijo; este proceso se aplica también a los cuatro coeficientes que se le relacionan.

Trabajos recientes y de gran relevancia que versan sobre la aplicación de la DWT en la compresión de imágenes, los cuales toman ventaja de la estrecha relación existente entre los coeficientes de las diferentes escalas, son:

Un método de compresión progresiva, denominado EZW (*Embedded Zerotree Wavelet Algorithm*), fue propuesto por J. M. Shapiro en 1993 [81]. Se trata de un codificador especialmente diseñado para trabajar con la transformada *wavelet* y fue originalmente diseñado para operar con imágenes. Shapiro destaca cuatro conceptos básicos en los cuales se basa el algoritmo EZW: 1) Una DWT o descomposición jerárquica de subbandas, que provee una representación compacta multiresolución de la imagen, 2) "*Zerotree*", predicción de la ausencia de información significativa a través de escalas mediante la exploración de auto-similitudes inherentes en la imagen, 3) Aproximaciones sucesivas que proveen una representación compacta de los coeficientes significativos y 4) Codificación aritmética adaptiva.

A. Said y W. A. Pearlman en [77] proponen una nueva y mejor implementación del EZW, el algoritmo SPIHT (*Set Partitioning in Hierarchical Trees*), basado en la utilización de conjuntos de datos organizados en árboles jerárquicos, es decir, el SPIHT tiene en cuenta la jerarquía de la descendencia del coeficiente que codifica. El tipo de codificación que realiza se basa en la clasificación por orden de bits significativos, resultando ser un método efectivo y económico en el uso de recursos. Al igual que el EZW, el SPIHT transforma a la imagen mediante la DWT y organiza los coeficientes *wavelet* resultantes en árboles de orientación espacial.

Un nuevo algoritmo para compresión de imágenes conocido como EBCOT (*Embedded Block Coding with Optimized Truncation*) fue propuesto por D. Taubman en el año 2000 [88]. En este algoritmo, cada sub-banda es dividida en pequeños bloques de coeficientes *wavelets*, llamados "bloques de código"; entonces genera cadenas de bits en forma separada para cada uno de ellos, que pueden ser truncadas independientemente a diferentes longitudes. Este algoritmo es combinado con una codificación por plano de bit (*bit plane coding*) y un codificador aritmético adaptativo. Cada "bloque de código" es nuevamente dividido en sub-bloques de 16×16, entonces, por cada plano de bit (*bit plane*) el mapa significativo de sub-bloques es, primero codificado mediante una codificación por cuadrante (*quadtree coding*) para posteriormente aplicarles cuatro operaciones de codificación: 1) codificación cero (*zero coding*), aplicada en función de los valores significativos, $\text{valor} \neq 0$, de los ocho vecinos del elemento codificado, 2) codificación de signo (*sign coding*); en este paso el signo de la muestra es codificado, 3) refinamiento de magnitud, que se aplica en función de los vecinos horizontales y verticales y si es o no el primer plano de bit donde se aplica esta codificación, y 4) una codificación por longitud de series o codificación de la carrera (RLE, *run-length encoding*). Finalmente EBCOT organiza la cadena final en capas mediante truncamiento.

El estándar JPEG2000 está fundamentalmente basado en la DWT y el algoritmo EBCOT [38].

B.2 Cuantificador

La matriz de salida del transformador no reduce el espacio de almacenaje que la matriz de píxeles original emplea. La acción utilizada para reducir el número de bits requeridos para almacenar la matriz resultante del bloque anterior es nombrada cuantificación. La cuantificación es simplemente el proceso efectuado para reducir el número de bits necesarios

para almacenar los coeficientes transformados mediante la reducción de la precisión de estos valores [59]. El cuantificador está enfocado a disminuir la redundancia psicovisual.

La etapa de cuantificación es indispensable para obtener una alta relación de compresión; sin embargo, este proceso genera pérdidas de información y además es un proceso irreversible. Debido a esto, el diseño de un proceso de cuantificación tiene importantes repercusiones en la compresión y en la calidad de la imagen reconstruida.

El ejemplo más antiguo de cuantificación es el redondeo [31]. Cualquier número real x puede ser redondeado al entero más cercano.

Si $q(x)$ representa el resultado de una cuantificación, con un error en la cuantificación expresado por $e = q(x) \pm x$, entonces el número cuantificado es $x = q(x) \pm e$.

Expresado en una forma general, un cuantificador está conformado por un conjunto de intervalos $\mathbf{S} = \{S_i;\ i \in I\}$, donde I es una colección de números enteros consecutivos; además, se define un conjunto de valores de reproducción o niveles $C = \{y_i;\ i \in I\}$, de tal forma que el cuantificador q queda definido por $q(x) = y_i$ para $x \in S_i$, el cual puede ser expresado como:

$$q(x) = \sum_i y_i 1_{s_i}(x) \qquad \textbf{(B.23)}$$

donde la función $1_{s_i}(x)$ es 1 si $x \in S_i$ y 0 en cualquier otro caso.

En un compresor de imágenes, la cuantificación es el proceso mediante el cual un conjunto de valores continuos en una imagen es aproximado a un conjunto de valores finito. La cuantificación se puede realizar sobre coeficientes individuales, en cuyo caso el proceso de cuantificar es conocido como cuantificación escalar (SQ, *Scalar Quantization*). La cuantificación también puede ser realizada sobre un grupo de coeficientes, conocida como cuantificación vectorial (VQ, *Vector Quantization*).

En términos generales, un cuantificador q es una función que va de $\mathbf{R}^N$ a un conjunto de códigos $C = \{y_i;\ i \in I\} \subset \mathbf{R}^N$, donde I es un conjunto finito

$$\begin{aligned} q &: \mathbf{R}^N \to C \\ x &\to q(x) = y_i \end{aligned} \qquad \textbf{(B.24)}$$

La cuantificación puede ser descompuesta en dos funciones $q = \alpha$ o β. El compresor define la función $\alpha : \mathbf{R}^N \to I$ dividiendo $\mathbf{R}^N$ en un conjunto de celdas $S_i = \{x \in \mathbf{R}^N \mid \alpha(x) = i\},\ i \in I$. El descompresor define $\beta : I \to \mathbf{R}^N$ especificado por C. Si $N = 1$ se trata de un SQ y si $N > 1$ se trata de un VQ.

Se pueden clasificar los cuantificadores en diferentes categorías según distintos criterios:

- Según la distancia existente entre dos niveles de cuantificación:
  i. Cuantificadores uniformes: la distancia entre los niveles de cuantificación siempre es la misma.
  ii. Cuantificadores no uniformes: en caso de tener una buena descripción estadística de la señal de entrada, se puede diseñar un cuantificador óptimo que sitúe los

niveles de cuantificación de tal forma que se minimice el error de cuantificación, con distancias variables entre los niveles.

- Según el tratamiento de la señal de entrada
  i. Cuantificación Vectorial o de Bloques: toma la señal de entrada en bloques de una dimensión dada y los cuantifica; tiene la ventaja de aprovechar las relaciones que pueda haber entre las distintas componentes de cada bloque.
  ii. Cuantificación Escalar (Vectorial de Dimensión 1): toma cada valor de la señal de entrada de forma independiente.
- Según si son predictivos o no.
  i. Predictivos: aprovecha las relaciones existentes entre muestras consecutivas de la señal para la cuantificación, cuantificando no el valor de la señal, sino su diferencia con la señal anterior.
  ii. No Predictivos: no aprovechan las relaciones mencionadas en el caso de los predictivos.

B.2.1 Cuantificación escalar

La cuantificación escalar puede ser descrita como una función que lleva cada elemento de un subconjunto de la recta real a un elemento de un conjunto de *K* números $\{y_1,...,y_K\}$, figura B.6, acorde a la siguiente relación

$$q(x) = y_i \text{ si } d_{i-1} < x \le d_i, \quad i = 1,...,K. \tag{B.25}$$

donde $d_0 < y_1 < d_1 < y_2 < \cdots < y_i < d_K$. Los números $d_0,...,d_K$ son llamados *niveles de decisión*, y los números $y_1,...,y_K$ son llamados *niveles de reconstrucción* o *niveles de representación*. Cada partición $(d_0,d_1],(d_1,d_2],...,(d_{K-1},d_K)$ es llamada *región de decisión* o *región de cuantificación*.

B.2.1.1 Cuantificación escalar uniforme

La cuantificación escalar uniforme es la forma más simple de cuantificar, pero también la menos efectiva. Se caracteriza porque todas las regiones de cuantificación son del mismo tamaño, figura B.7(a). El proceso de cuantificación escalar uniforme es muy semejante a la forma en que un convertidor analógico digital lleva acabo su tarea: concentrando un conjunto de valores en uno solo.

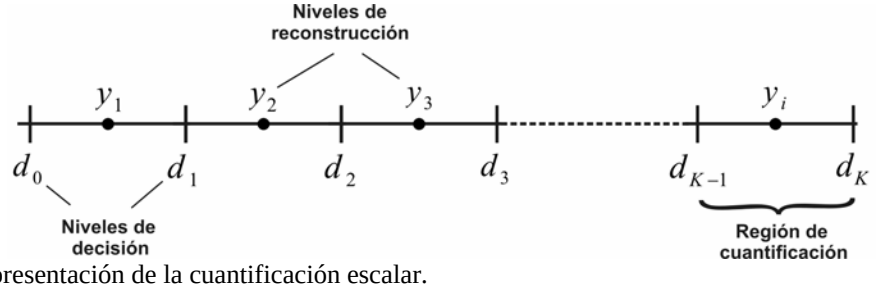


Figura B.6. Representación de la cuantificación escalar.

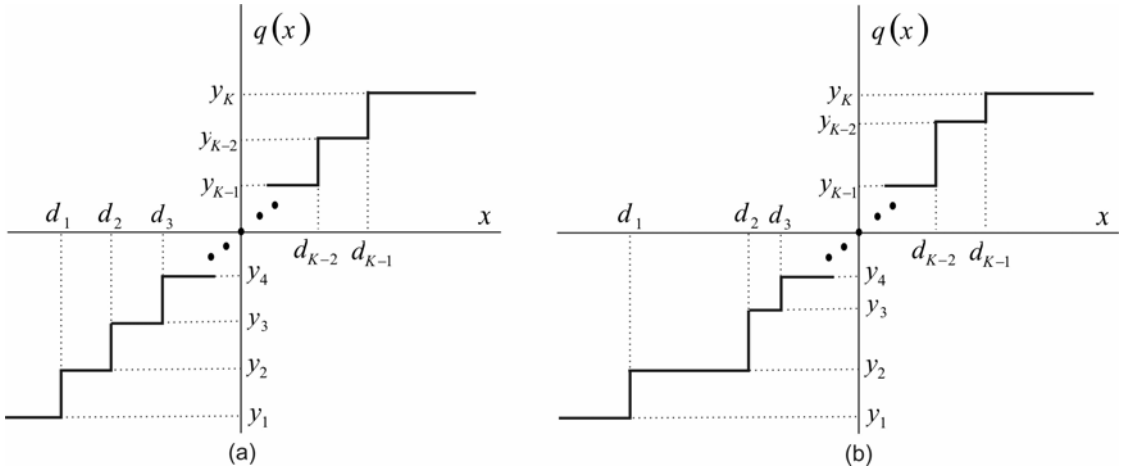


Figura B.7. (a) Cuantificación escalar uniforme. (b) Cuantificación escalar no uniforme.

Una forma práctica de cuantificación escalar uniforme es la de dividir todos los símbolos de la imagen entre un número *Q*, el cual representa el factor de cuantificación, para después aplicarle una función de redondeo al entero próximo

$$q(x_{uv}) = round\left(\frac{x_{uv}}{Q}\right) \qquad \textbf{(B.26)}$$

donde los índices *u* y *v* definen las dimensiones de la imagen. Aplicado el proceso inverso del método utilizado para cuantificar, la información recuperada por este proceso ya no representa fielmente a la original, existe pérdida de información; estas pérdidas dependerán del factor de cuantificación utilizado.

$$\tilde{x}_{uv} = q(x_{uv}) \times Q \qquad \textbf{(B.27)}$$

B.2.1.2 Cuantificación escalar no uniforme

Una forma alternativa de cuantificación escalar es la cuantificación escalar no uniforme. Se caracteriza porque las regiones de cuantificación son de diferente tamaño, figura B.7(b). Este tipo de cuantificación considera la distribución de los coeficientes resultantes de la transformación de la imagen. Partiendo del hecho, en el caso de una transformada DCT, que el coeficiente de la esquina superior izquierda del bloque representa el promedio de la intensidad del bloque (y la frecuencia más baja), que moviéndose hacia la derecha los coeficientes representan frecuencias altas horizontales, que moviéndose hacia abajo representan frecuencias altas verticales y que con un movimiento diagonal los coeficientes representan la combinación de frecuencias altas verticales y horizontales, entonces es recomendable implementar una cuantificación que considere estas características. Además, con la finalidad de obtener mejores resultados en la etapa de codificación, estándares como el JPEG explotan las características del HVS, tales como:

- Si existen coeficientes resultantes de la etapa de transformación con valores cercanos a cero, estos valores pueden igualarse a cero; el HVS es completamente insensible a estos errores.
- El HVS es sensible a pequeños errores en los coeficientes DC y en los coeficientes de baja frecuencia, pero es mucho menos sensible a errores de amplitud en coeficientes de alta frecuencia.

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

(a)

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

(b)

Figura B.8. Matrices para cuantificación escalar no uniforme de un compresor de imágenes de color basado en la DCT; (a) matriz para los coeficientes de la componente de luminancia, (b) matriz para los coeficientes de las componentes de crominancia.

Algunos estándares definen matrices de cuantificación considerando la estructura de los coeficientes de la DCT y el comportamiento del HVS [86]. La figura B.8 muestra un ejemplo de matrices usadas por un compresor de imágenes de color basado en la DCT.

Las matrices de la figura B.8 representan intervalos de cuantificación. Debido a esto, el proceso consiste en dividir cada coeficiente resultante de la etapa de transformación, entre el correspondiente valor de la matriz de cuantificación. Para facilitar el redondeo y la reconstrucción, la magnitud de cada coeficiente es incrementado con la mitad del elemento de la matriz correspondiente antes de la división; el resultado es entonces redondeado al entero más cercano; este proceso modifica la ecuación B.26, obteniendo la expresión:

$$q(x_{uv}) = round\left(\frac{x_{uv} + \frac{Q_{uv}}{2}}{Q_{uv}}\right) \qquad \textbf{(B.28)}$$

En este caso, el proceso de cuantificación inversa es definido por:

$$\tilde{x}_{uv} = q(x_{uv}) \times Q_{uv} \qquad \textbf{(B.29)}$$

En el caso particular de los coeficientes obtenidos de una transformada *wavelet*, se pueden tener cuantificadores independientes para cada escala resultante, tomando en cuenta la información que cada escala representa, figura B.9.

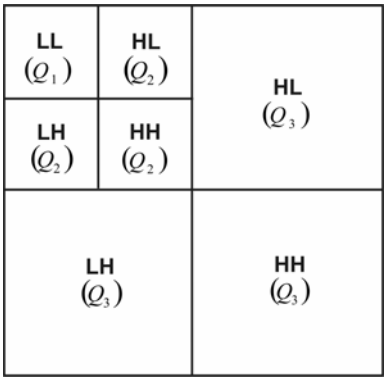


Figura B.9. Cuantificación escalar no uniforme para coeficientes *wavelet.*

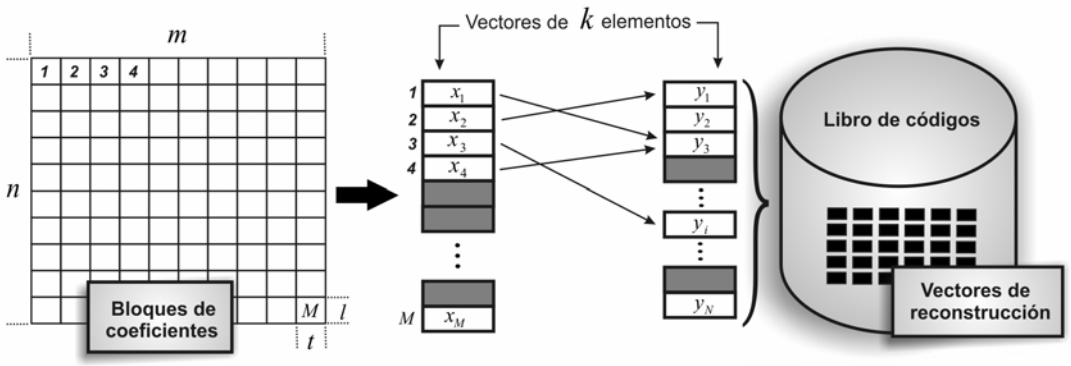


Figura B.10. Estructura básica de una cuantificación vectorial.

B.2.2 Cuantificación vectorial

La SQ no puede tomar ventaja de la correlación existente entre píxeles cercanos en una imagen, ya que actúa en forma individual sobre una variable de una dimensión como puede ser el valor o intensidad de un píxel. En contraste, una VQ opera sobre un vector multidimencional representado por más de una coordenada.

Una VQ actúa sobre vectores, un vector al estar formado por un conjunto de elementos individuales (conjunto ordenado de escalares), entonces puede ser usado para describir casi cualquier tipo de patrón, como podría ser un segmento de una señal de voz o de una imagen, simplemente formando un vector de muestras de la forma de onda o de la imagen.

La figura B.10 muestra un esquema básico de una VQ. Los coeficientes resultantes de la transformación de una imagen de $m \times n$ píxeles, pueden ser divididos en M bloques de $t \times l$, los que pueden ser representados como *vectores de entrada* de k elementos, $X = \{x_i : i = 1, 2, ..., M\}$, $k = t \times l$; VQ hace uso de un conjunto de N *vectores de reconstrucción* de k elementos, $C = \{y_i : i = 1, 2, ..., N\}$ [30]. En la VQ se mapean los vectores de entrada de un espacio vectorial $\mathbf{R}^k$ dentro de un conjunto finito de vectores de reconstrucción, $q : \mathbf{R}^N \to C$, $X \in \mathbf{R}^N$. A cada vector de reconstrucción y_i también se le conoce como *vector codificado* o *codeword* y al conjunto de todos los vectores codificados como *libro de códigos* (*codebook*). Asociado a cada uno de los vectores codificados del libro de códigos se tiene una celda denominada región de "*Voronoi*", V_i; la i-ésima región está definida por:

$$V_i = \{x \in \mathbf{R}^k : q(x) = y_i\}$$
$$V_i = \{x \in \mathbf{R}^k : \|x - y_i\| \le \|x - y_j\|, \ \forall \ j \ne i\} \qquad \textbf{(B.30)}$$

El conjunto de regiones Voronoi forman una partición de $\mathbf{R}^k$, tal que:

$$\bigcup_{i=1}^{N} V_i = \mathbf{R}^k$$
$$V_i \bigcap_{i=1}^{N} V_j = \phi, \ \forall \ j \ne i \qquad \textbf{(B.31)}$$

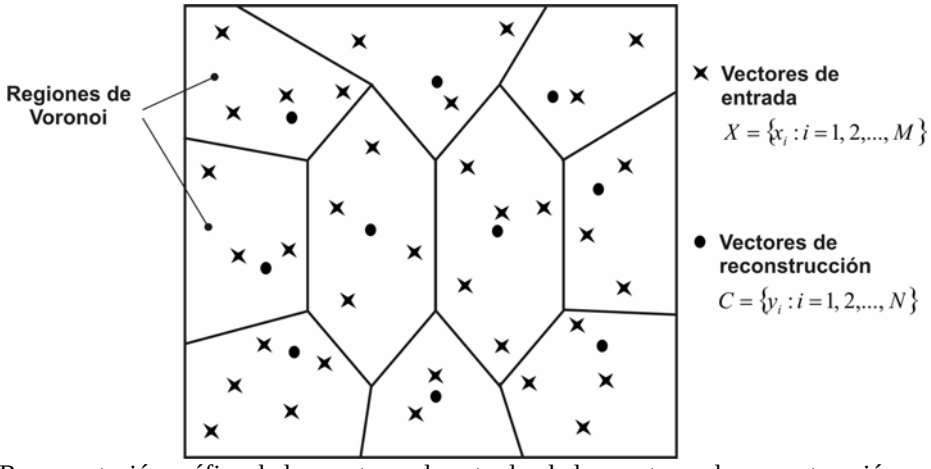


Figura B.11. Representación gráfica de los vectores de entrada, de los vectores de reconstrucción y de la región de Voronoi.

La figura B.11 muestra cómo un segmento de $\mathbf{R}^k$ es dividido en N regiones Voronoi. Estas regiones son separadas por líneas imaginarias y cada una de ellas contiene sólo un vector de reconstrucción. Cada vector de entrada que cae dentro de la región V_i es substituido por el índice, i, del vector de reconstrucción y_i con base en un criterio de similitud entre ambos vectores. Uno de los criterios más populares para realizar la asignación es la distancia euclidiana entre el vector de entrada y el vector de reconstrucción. La distancia euclidiana está definida por:

$$d(x, y_i) = \sqrt{\sum_{j=1}^{k} (x_j - y_{ij})^2} \qquad \textbf{(B.32)}$$

donde x_j es el j-ésimo componente del vector de entrada, y y_{ij} es el j-ésimo componente del vector codificado y_i.

En términos generales, una VQ consta de dos fases, la primera de éstas se lleva a cabo en el cuantificador, que toma un vector de entrada y entrega como salida el índice del vector de reconstrucción que ofrece la distorsión más baja, la cual es determinada mediante la evaluación de la distancia euclidiana entre el vector de entrada y cada vector de reconstrucción perteneciente al libro de códigos. La segunda fase se realiza cuando el cuantificador inverso recibe el índice, remplazándolo por el vector de reconstrucción asociado. La figura B.12 muestra un diagrama de bloques de estas operaciones.

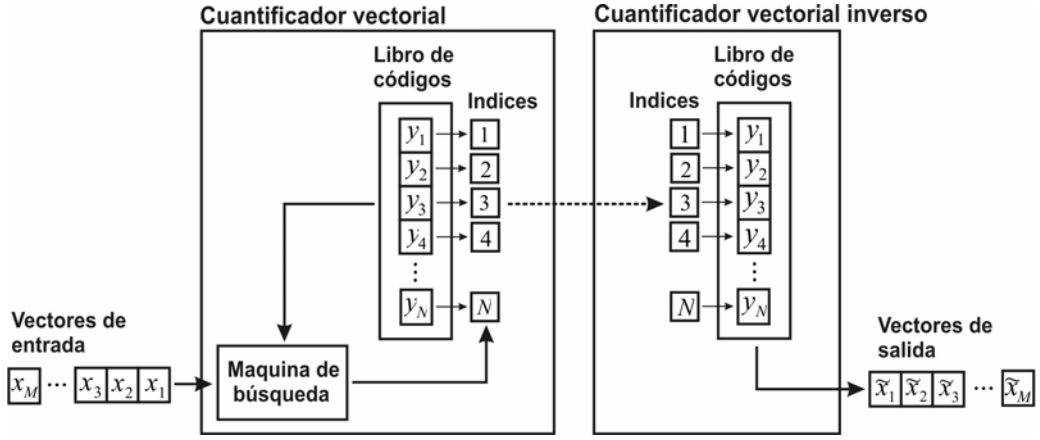


Figura B.12. Diagrama de bloques de un proceso de cuantificación vectorial.

La cuantificación vectorial es una técnica que puede producir resultados muy próximos a los límites teóricos; su principal desventaja es que dos de sus principales procesos, la generación del libro de códigos y el proceso de búsqueda, son altamente complejos. En la generación del libro de códigos se utilizan técnicas de alta demanda computacional y el proceso de búsqueda del vector de reconstrucción idóneo para un vector de entrada es muy lento.

El diseño de un libro de códigos que implemente la mejor representación de un vector de entrada es un proceso muy complejo. Esto significa que se requiere una exhaustiva búsqueda para encontrar el mejor libro de códigos, y la búsqueda se incrementa exponencialmente cuando en número de vectores codificados aumenta.

El esquema más simple para encontrar un libro de códigos es el LBG por Linde-Buzo-Gray, los autores de esta idea [52], [35]. Este esquema, aplicado a la compresión de imágenes, sigue el siguiente algoritmo:

1. Se determina el tamaño del libro de códigos o número de vectores de reconstrucción, N.
2. Se realiza una selección aleatoria de N bloques del conjunto de entrenamiento (vectores de entrada), que conformarán el libro de códigos inicial. En este punto se pueden emplear una o más imágenes diferentes.
3. Se toma cada vector de entrada y se encuentra la distancia euclidiana entre éste y cada vector de reconstrucción. El vector de entrada pasa a formar parte del grupo de vectores que tiene la mínima distancia con respecto a un vector de reconstrucción común.
4. Se recalcula el libro de códigos, mediante la obtención de un promedio de cada grupo con la suma de los componentes de cada grupo y divididos entre el número de vectores en el grupo:

$$y_i = \frac{1}{m}\sum_{j=1}^{m} x_{ij} \qquad \textbf{(B.33)}$$

   donde *i* es el componente de cada vector y *m* es el número de vectores en el grupo.
5. Se repiten los pasos 3 y 4 hasta que no existan cambios en los vectores de reconstrucción o los cambios sean muy pequeños.

Este algoritmo de generación del libro de códigos es de los más populares principalmente por su simplicidad, pero es un proceso en extremo lento debido a que por cada iteración, un vector de entrada es comparado con cada vector de reconstrucción del libro de códigos. Existen una gran cantidad de métodos que se aplican al diseño del libro de códigos, ejemplos de éstos son: *Pairwise Nearest Neighbor* (PNN) [24], *Simulated Annealing* (SA) [25], [91], *Maximum Descent* (MD), y *Competitive Learning* [98], [110].

Con respecto al proceso de búsqueda, el método más simple, conocido como “búsqueda total”, es también el más lento. Si existieran M vectores de entrada y N vectores de reconstrucción, y cada vector es de *k* dimensiones, entonces el número de multiplicaciones que haría el proceso de búsqueda sería igual a kMN, el número de sumas y restas sería $MN((k-1)+k) = MN(2k-1)$ y el número de comparaciones sería $MN(k-1)$. Esto hace de este método un proceso con un alto coste computacional.

Una medida de la distorsión producida por un cuantificador es una asignación de coste no negativa $D(x, y)$ asociada a la cuantificación de un vector $x \in \mathbf{R}^k$ sobre un vector de reconstrucción $y = q(x) \in C$. Normalmente, la cuantificación será efectiva si la distorsión es baja; esta medida da una idea de la diferencia entre el vector de entrada original y el vector de reconstrucción que se le asocia.

Se han utilizado numerosas medidas para el cálculo de la distorsión, cada una de ellas adecuada para un problema concreto, siendo conveniente cambiar de heurística según se trate con diferentes casos de entradas. Se muestran a continuación algunas de interés:

- Error cuadrático medio, ecuación 2.39.
- Error cuadrático ponderado:

$$\mathbf{ECP} = \sum_{i=1}^{k} \omega_j \left(x_j^i - y_j^i\right)^2 \qquad \textbf{(B.34)}$$

- Distorsión máxima:

$$\mathbf{DM} = \max_j \left|x_j^i - y_j^i\right| \qquad \textbf{(B.35)}$$

donde, x_j^i es el *j*-ésimo elemento del vector original *i*, y_j^i es el *j*-ésimo elemento del vector reconstruido *i*, $\omega_j \geq 0$ es el peso del elemento *j*-ésimo, $j = 0,..., k-1$.

Otra definición importante es la de resolución de un cuantificador. La resolución de un VQ es:

$$r = \frac{\log_2 N}{k} \qquad \textbf{(B.36)}$$

La resolución mide el número de bits por componente de vector empleado para representar el vector de entrada y da una idea de la precisión alcanzable con un VQ si el alfabeto está bien diseñado. La resolución de un SQ es con $k = 1$

$$r = \log_2 N \qquad \textbf{(B.37)}$$

B.3 Codificación

El codificador asigna un código a cada símbolo que entrega el cuantificador. El codificador debe ser diseñado para reducir la redundancia en la codificación. La codificación puede considerarse como una función que asocia un conjunto de símbolos, *X*, a un *alfabeto de códigos*, *C*, con el propósito de reducir el número de bits utilizados en su representación; matemáticamente se puede expresar como $fc : X \rightarrow C$. Este proceso mantiene una correspondencia uno a uno entre los símbolos fuente y los símbolos del alfabeto de códigos, garantizando con esto la reversibilidad del proceso.

Una imagen transformada y cuantificada típicamente contiene pocos coeficientes diferentes de cero y una gran cantidad de coeficientes con valor cero. Una imagen con esta estructura puede ser eficientemente comprimida de acuerdo al siguiente proceso:

- Reordenamiento de los coeficientes cuantificados.
- Codificación por longitud de series o codificación de la carrera (RLE, *run-length encoding*).
- Codificación por entropía.

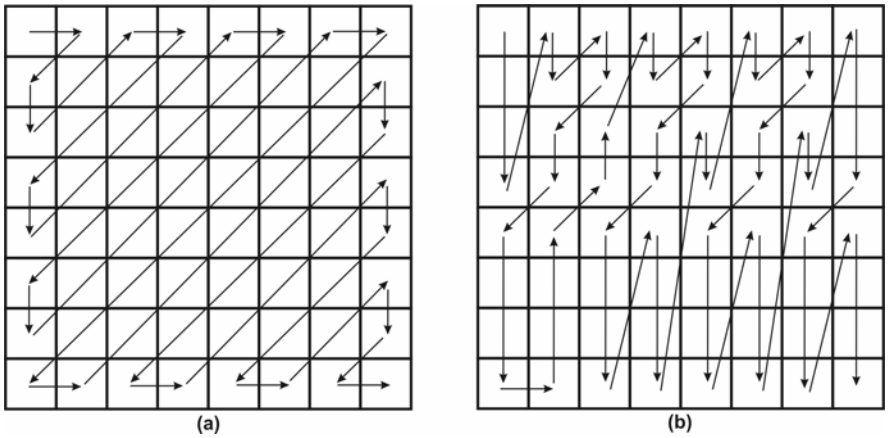


Figura B.13. Técnicas de exploración: (a) exploración en zig-zag. (b) exploración en zig-zag modificada.

B.3.1 Reordenamiento de los coeficientes cuantificados

En el proceso de transformación de una imagen la distribución de los coeficientes guarda una estructura fija. En el caso de la DCT, el coeficiente de la esquina superior izquierda representa el promedio de la intensidad del bloque; a partir de este punto, desplazándose hacia la derecha, los coeficientes van representando frecuencias horizontales más altas y con un desplazamiento hacia abajo los coeficientes van representando frecuencias verticales más altas. Cuando se usa una DWT la relación existe entre coeficientes de diferentes niveles, ya que en una transformación *wavelet* cada coeficiente tiene relación con los coeficientes de niveles anteriores. Cuando se usa la transformada DCT, después de aplicarles el proceso de cuantificación, los coeficientes diferentes de cero en un bloque de la imagen generalmente quedan agrupados en la esquina superior izquierda del bloque, es decir, la DCT concentra la energía en los coeficientes en esta esquina.

Considerando esta propiedad de la DCT, es recomendable reordenar la distribución de los coeficientes mediante una técnica denominada exploración. Esta técnica permite adecuar los coeficientes en orden descendente según su probabilidad permitiendo una codificación más eficiente. La figura B.13 muestra 2 esquemas empleados en la reorganización de los coeficientes resultantes de una DCT. Estos esquemas redistribuyen los coeficientes considerando la información que cada uno representa.

Una exploración en forma diagonal a 45° es denominada exploración en zig-zag, la cual representa la mejor secuencia para emplear en un sistema no-entrelazado, figura B.13(a). Con el propósito de conseguir más detalle acerca de una imagen, en la reordenación para una fuente entrelazada la exploración se extiende dos veces más por encima del área vertical, es decir, las frecuencias verticales aparecen dos veces más que las frecuencias horizontales; considerando esta característica se determina que la exploración ideal para una imagen entrelazada será sobre una diagonal de 67.5°; la figura B.13(b) muestra que esta forma de exploración entrega primero las frecuencias espaciales verticales y luego las frecuencias espaciales horizontales.

Cuando el transformador emplea una DWT, las técnicas de reordenamiento de coeficientes emplean algoritmos que explotan la correlación entre coeficientes de diferentes niveles; esto se debe a que en una transformación *wavelet* cada coeficiente tiene relación con coeficientes de niveles anteriores, figura B.14. Algoritmos que aprovechan esta propiedad de la DWT para obtener mejores resultados en la compresión de imágenes son el EZW, SPIHT y EBCOT, discutidos en la sección B.1.2.

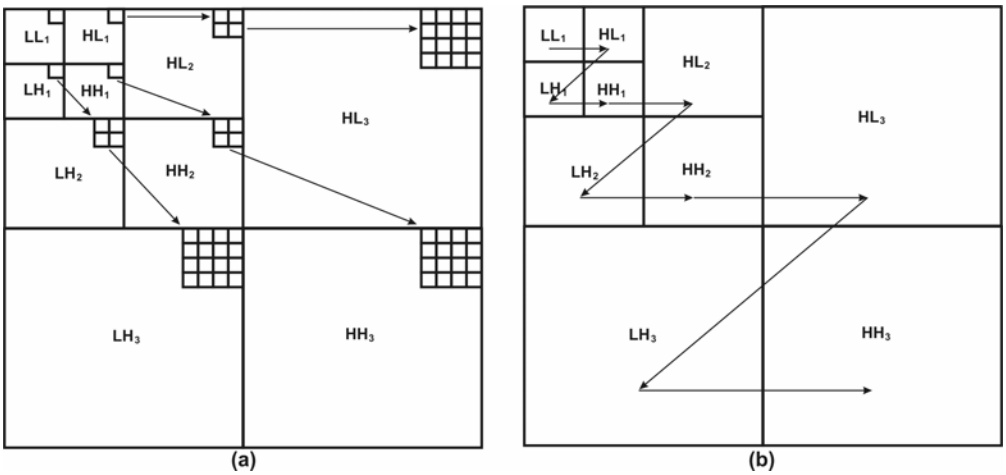


Figura B.14. (a) Dependencia entre coeficientes de diferente nivel. (b) Orden de exploración.

B.3.2 Codificación por longitud de series

El método RLE es el tipo de codificación más simple. El principio de funcionamiento de este método es la búsqueda de repeticiones consecutivas de un símbolo para posteriormente substituirlo por un símbolo especial, el símbolo a codificar y el número de veces que se repite [27]. La figura B.15 muestra cómo una cadena de símbolos es codificada usando el método RLE. El símbolo especial, en este caso denotado por *, tiene por función indicar que un símbolo se repite un determinado número de veces, por ejemplo la secuencia *B8 indica que el símbolo B se repite 8 veces consecutivas.

Resulta claro determinar que este método de codificación demuestra gran eficiencia cuando hay un alto número de repeticiones consecutivas de un determinado símbolo. RLE es el algoritmo utilizado en los formatos gráficos BMP y PCX, aunque cada uno usa un método distinto de implementación.

La codificación RLE realiza una compresión de datos sin pérdidas y es muy utilizado en imágenes de 8 bits indexadas (en un principio fue utilizado para imágenes en blanco y negro). No funciona tan bien en imágenes donde varía constantemente el color de los píxeles como en las fotografías, aunque JPEG lo utiliza de forma efectiva en los coeficientes que quedan después de transformar y cuantificar bloques de imágenes.

B.3.3 Codificación por entropía

Anteriormente se ha definido una cantidad llamada entropía, que indica el límite teórico para la compresión de datos, así como mide la cantidad de información media de una fuente. El valor de la entropía está íntimamente ligado con la eficiencia que pueden alcanzar los algoritmos de codificación, entendiendo por eficiencia representaciones que requieran menor cantidad de información. Los codificadores por entropía son aquellos que consideran las características estadísticas de la fuente de información a codificar, es decir, toman en cuenta la probabilidad de aparición de los símbolos al momento de asignarles los códigos del alfabeto.

Figura B.15. Ejemplo de una codificación RLE.

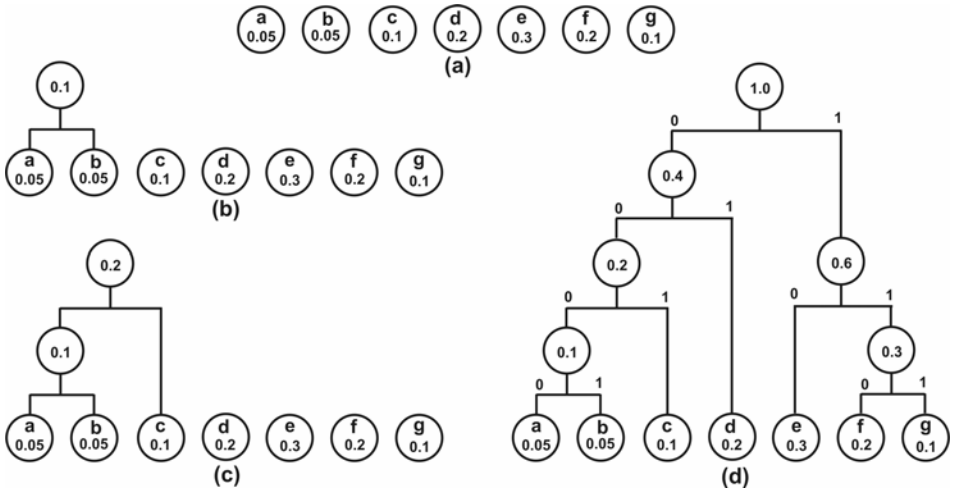


Figura B.16. Construcción de un árbol Huffman. (a) hojas o nodos; (b) y (c) combinación de nodos; (d) árbol de Huffman completo.

La codificación por entropía es un mapeo de un alfabeto fuente de dimensión *m*, representado por $X = \{x_i \mid 0 \le i < m\}$, a un alfabeto de códigos, $\{c_i = C(x_i) \mid 0 \le i < m\}$, con el propósito de minimizar el tamaño del alfabeto fuente. Si la probabilidad del símbolo x_i en el alfabeto fuente es $p(x_i)$, la longitud esperada (en bits) de un mensaje compuesto de *n* símbolos es:

$$R = n\sum_i p(x_i)l(C(x_i)) \qquad \textbf{(B.38)}$$

Donde $l(c_i)$ es la longitud (en bits) del código c_i. Para que el proceso sea reversible es necesario que exista un código por cada símbolo fuente, es decir que se trate de un mapeo uno a uno. Shannon en sus trabajos acerca de la teoría de la información estableció que la entropía de una fuente es el límite inferior del número medio de bits que son necesarios para codificar las salidas de dicha fuente [80]. Existen diversos algoritmos que se aproximan a este límite, entre los que destacan la codificación Huffman y la codificación aritmética; estos algoritmos también son considerados codificadores de longitud variable en virtud de que asignan códigos más pequeños a los símbolos fuente más probables.

B.3.3.1 Codificación Huffman

En el año 1952, David A. Huffman propone un método de codificación de información basado en parámetros estadísticos denominado código de Huffman [36]. Un código de Huffman es un código de longitud variable, es decir, la longitud de cada código asociado a un símbolo depende de la frecuencia de aparición de este último; cuanto más frecuente sea un símbolo, su código asociado será más corto. Además, un código de Huffman es un código libre de prefijos, es decir, ningún código forma la primera parte de otro código; esto permite que los mensajes codificados no sean ambiguos.

El algoritmo Huffman crea un *árbol de códigos*, llamado *árbol Huffman*, con base en la distribución de la probabilidad. Al inicio del algoritmo, por cada símbolo se crea una *hoja* o *nodo* el cual está integrado por el símbolo y su probabilidad, figura B.16(a). Los dos nodos con la probabilidad más baja son combinados para crear un nuevo nodo figura B.16(b); en esta nueva estructura a los nodos originales se les denomina *nodos hijos* y al nodo creado *nodo padre*, la probabilidad del nodo padre es igual a la suma de las probabilidades de los nodos hijos.

Este tipo de unión es repetida escogiendo otra vez a los dos nodos con las probabilidades menores, los nodos padres son también considerados en este proceso, por ejemplo, en la figura B.16(c) se forma un nuevo nodo padre mediante la unión del nodo *c* y del nodo padre formado por los nodos *a* y *b*. El proceso continua hasta que no queden nodos sin pertenecer a un nodo padre y finalmente formar el árbol Huffman completo, figura B.16(d). Los brazos de cada nodo padre son etiquetados con valor 0 y 1, los códigos de los símbolos son obtenidos recorriendo el árbol Huffman desde el nodo padre superior hasta el nodo que representa al símbolo que se desea codificar; los códigos son la concatenación de las etiquetas que contienen los brazos recorridos en el trayecto mencionado; estos códigos son mostrados en la tabla B.1.

Tabla B.1. Símbolos, probabilidad y correspondientes códigos Huffman.

Símbolo	Probabilidad	Código
a	0.05	0000
b	0.05	0001
c	0.1	001
d	0.2	01
e	0.3	10
f	0.2	110
g	0.1	111

En el caso particular de una imagen, la codificación Huffman convierte los valores de brillo de los píxeles en códigos de longitud variable con base en su frecuencia de ocurrencia en la imagen. De esta manera, a los valores de brillo que ocurren más frecuentemente se les asignan los códigos más cortos y a los valores de brillo que ocurren con menos frecuencia se les asignan los códigos más largos. El resultado es que la imagen comprimida requerirá de menos bits para describir la imagen original. El esquema de compresión Huffman empieza evaluando el histograma de brillo de una imagen. El histograma proporciona la frecuencia de ocurrencia para cada valor de brillo de la imagen. Ordenando los valores de brillo por sus frecuencias de ocurrencia, se obtiene una lista donde el primer valor es aquel que se encuentra más a menudo en la imagen, y el último valor es el que se encuentra con menos frecuencia en la imagen. Con esta lista, el codificador Huffman asigna nuevos códigos a cada valor de brillo. Los códigos asignados son de longitudes variables; los códigos más cortos son asignados a los primeros (más frecuentes) valores de la lista y, eventualmente, los códigos más largos se asignan a los últimos (menos frecuentes) valores de la lista. Finalmente, la imagen comprimida es creada simplemente sustituyendo los nuevos códigos de valores de brillo de longitud variable por los códigos de valores de brillo originales de 1 byte. Resulta obvio que la lista de códigos Huffman debe formar parte de la operación de decodificación.

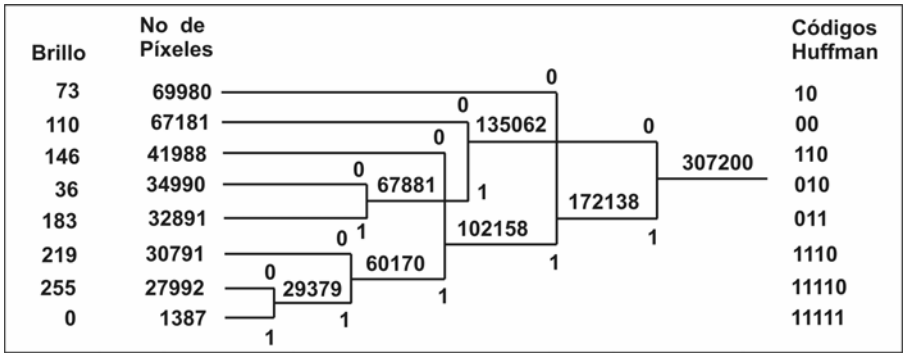


Figura B.17. Ejemplo de la codificación Huffman.

Los códigos Huffman asignados forman el árbol Huffman haciendo combinaciones con los valores de brillo basado en la suma de las frecuencias de ocurrencia. El árbol asegura que los códigos más largos se asignen a los brillos menos frecuentes y los códigos más cortos se asignen a los brillos más frecuentes. Usando el brillo clasificado en orden de sus frecuencias de ocurrencia, los dos del final de la lista (menos frecuentes) se combinan y se etiquetan como 0 y 1. Los brillos combinados son representados por la suma de las frecuencias de ocurrencia. Entonces, se determinan y se combinan las próximas dos frecuencias de ocurrencia más bajas. De nuevo, el siguiente par se etiqueta con 0 y 1, y es representado por la suma de las frecuencias de ocurrencia. Esto continúa hasta que todos los valores de los píxeles se han combinado. El resultado es un árbol que, cuando se sigue del final hasta el principio, indica el nuevo código Huffman binario para cada brillo en la imagen, figura B.17.

B.3.3.2 Codificación aritmética

La codificación aritmética sortea algunas de las limitaciones básicas de la codificación Huffman. Por ejemplo, considere un símbolo sumamente frecuente de probabilidad 0.95, la palabra más corta que el código Huffman puede asignar tiene largo 1, mientras que de acuerdo con la teoría sólo se necesitarán $\log_2(1/0.95)$ bits para codificarla, pero $\log_2(1/0.95) < 0.075$. La codificación aritmética fusiona secuencias enteras de símbolos y los codifica como un único símbolo para superar esta dificultad; de esta manera alcanza resultados casi óptimos para ésta y cualquiera otra distribución de probabilidad.

En la codificación aritmética, una secuencia de símbolos es codificada como un número real en un intervalo de cero a uno. Con la codificación aritmética típicamente se obtiene mejor relación de compresión que con una codificación Huffman, debido a que una codificación aritmética produce un único símbolo como resultado del proceso de codificación, mientras que una codificación Huffman produce un código por cada símbolo [95].

En la codificación aritmética no existe una correspondencia biunívoca entre los símbolos fuente y las palabras código. En cambio, se asigna una sola palabra código aritmética a una secuencia completa de símbolos fuente. La propia palabra código define un intervalo de números reales entre 0 y 1. Conforme aumenta el número de símbolos del mensaje, el intervalo utilizado para representarlo se va haciendo menor y se va incrementando el número de unidades de información necesarias para representar dicho intervalo. Cada símbolo del mensaje reduce el tamaño del intervalo según su probabilidad de aparición. Puesto que esta técnica no requiere, como sucedía con la técnica de Huffman, que cada símbolo de la fuente se traduzca en un número entero de símbolos del código (esto es, que los símbolos se codifiquen uno a uno), se alcanza (sólo en teoría) el límite establecido por el teorema de codificación sin ruido.

La codificación aritmética es una técnica de codificación estadística sin pérdida. Existen pocas desventajas o limitantes en la codificación aritmética: una de ellas es que el código completo debe ser recibido para comenzar el proceso de decodificación; además, si un bit del código es erróneo, el mensaje entero también lo será. Otra restricción es que existe un límite en la precisión del número que representará al código, lo que limita el número de símbolos que pueden ser codificados dentro de un código.

El algoritmo básico de una codificación aritmética es el siguiente:

1. Se define un intervalo $[L, H)$ y se inicializa a $[0,1)$.

2. Por cada símbolo de entrada, se llevan a cabo los siguientes pasos:
   i. Se subdivide el intervalo $[L, H)$ en sub-intervalos, uno por cada símbolo. El tamaño del sub-intervalo de un símbolo es proporcional a la probabilidad estimada del mismo.
   ii. Se selecciona el sub-intervalo correspondiente al símbolo entrante al codificador y se hace un nuevo intervalo.
   iii. Se transmiten suficientes bits para distinguir el intervalo final de otros intervalos finales posibles.

Un codificador aritmético debe trabajar en conjunto con un modelo que determine la probabilidad de cada elemento en una secuencia de símbolos; de esta forma, el código final será una secuencia de decisiones y en cada punto de decisión se estima la probabilidad de cada elemento. El codificador entonces usa el conjunto de probabilidades para codificar la decisión actual.

La longitud del archivo codificado depende del modelo para estimar la probabilidad y cómo es usado. La mayoría de los sistemas de compresión con codificación aritmética, estiman la probabilidad *p* de un símbolo mediante:

$$p = \frac{\text{número de repeticiones del símbolo}}{\text{número total de símbolos en la secuencia}} \tag{B.39}$$

Este modelo puede ser codificado en $-\log_2 p$ bits usando codificación aritmética. La forma más comúnmente usada para estimar la probabilidad es la adaptativa (estimar la probabilidad de cada símbolo basada en todos los símbolos que la preceden).

En el paso dos es necesario calcular el sub-intervalo correspondiente al símbolo actual a_i. Para lograr esto es necesario obtener las probabilidades acumuladas:

$$P_C = \sum_{k=1}^{i-1} p_k$$
$$P_N = \sum_{k=1}^{i} p_k \tag{B.40}$$

Entonces el nuevo sub-intervalo será:

$$[L + P_C(H - L), L + P_N(H - L)) \tag{B.41}$$

Sea *M* una secuencia de símbolos cuyo alfabeto es $A = a, b, c, d$. Las probabilidades de los símbolos en un mensaje son:

$$P_M(a) = 0.5,\ P_M(b) = 0.25,\ P_M(c) = 0.125,\ P_M(d) = 0.125.$$

La figura B.18 y tabla B.2 muestran la inicialización del intervalo $[L, H) = [0,1)$ en función de las probabilidades de los símbolos del alfabeto *A*.

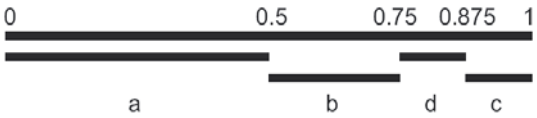


Figura B.18. División del intervalo $[0,1)$.

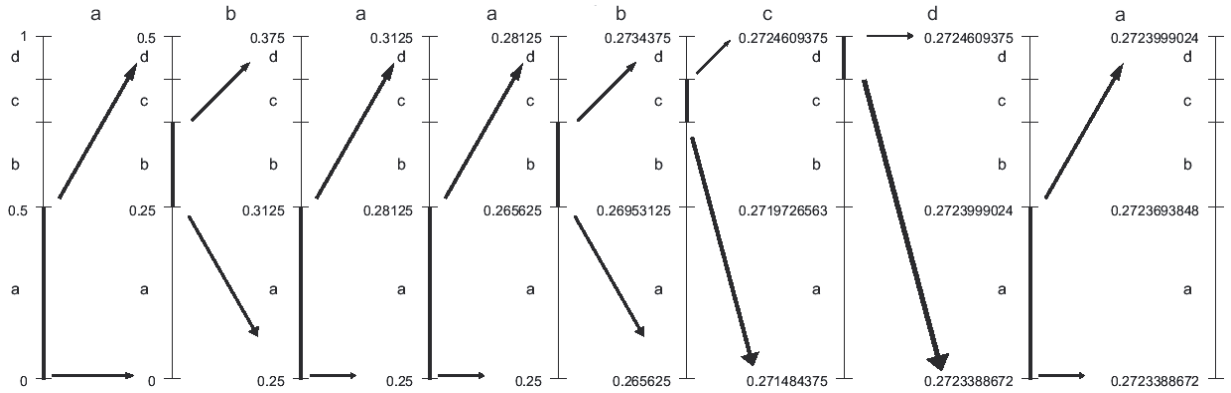


Figura B.19. Flujo del algoritmo de codificación aritmética para la secuencia $M = abaabcda$.

Tabla B.2. Inicialización del intervalo $[0,1)$.

Símbolo fuente	Probabilidad	Sub-intervalo inicial
a	0.5	$[0,0.5)$
b	0.25	$[0.5,0.75)$
c	0.125	$[0.75,0.875)$
d	0.125	$[0.875,1)$

La figura B.19 muestra el flujo que sigue el algoritmo de codificación aritmética para la secuencia de símbolos $M = abaabcda$, el cual hace uso de las ecuaciones B.40 y B.41 para determinar los sub-intervalos intermedios y el intervalo final. El intervalo final resultante es $[0.2723388672,0.2723999024)$; se puede utilizar cualquier número que esté dentro del sub-intervalo, como por ejemplo el 0.272399902, para representar el mensaje.

En el mensaje codificado aritméticamente de la figura B.19, se utilizan nueve dígitos decimales para representar el mensaje de ocho símbolos. Esto se traduce en 10/8, ó 1.11 dígitos decimales por símbolo fuente, lo que se aproxima bastante a la entropía de la fuente, que resulta ser de 0.978 dígitos decimales por símbolo. Conforme aumenta la longitud de la secuencia a codificar, el código aritmético resultante se aproxima al límite establecido por el teorema de la codificación sin ruido. En la práctica, existen dos factores que hacen que el rendimiento de la codificación se aleje de este límite:

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