UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA, MANAGUA

UNAN - MANAGUA

FACULTAD REGIONAL MULTIDISCIPLINARIA DE CHONTALES

FAREM-CHONTALES

PROGRAMA DE DOCTORADO EN MATEMÁTICA APLICADA

Tesis presentada para optar al título de Doctor en Matemática Aplicada

ALGORITMOS HEURÍSTICOS DE COINCIDENCIA PARA LA ESTIMACIÓN DE MOVIMIENTOS EN COMPRESION DE IMÁGENES

Autor: M.Sc.: Fernando José Hernández Gómez

Director de Tesis: M.Sc.-Ph.D.: Antonio Parajón Guevara

Consejera de Estudios: Ph.D.: Irene Loiseau

Diciembre, 2017

Índice general

ANTONIO PARAJÓN GUEVARA, profesor titular del Departamento de Matemática de la Facultad de Educación e Idiomas de la Universidad Nacional Autónoma de Nicaragua, UNAN-Managua.

CERTIFICA que la presente memoria de investigación:

ALGORITMOS HEURÍSTICOS DE COINCIDENCIA PARA LA ESTIMACIÓN DE MOVIMIENTOS EN COMPRENSION DE IMÁGENES

Ha sido realizada bajo su dirección en el *PROGRAMA DE DOCTORADO EN MATEMÁTICA APLICADA* por el Máster Fernando José Hernández Gómez, y constituye su tesis para optar al grado de Doctor en Matemática Aplicada.

Y para que así conste, en cumplimiento con la normativa vigente de posgrado, autoriza su presentación ante la Facultad Regional Multidisciplinaria De Chontales (FAREM-Chontales) para que pueda ser tramitada su lectura y Defensa pública.

Managua, Nicaragua, 23 de Noviembre de 2017.

El DIRECTOR DE LA TESIS

Antonio Parajón Guevara, MS.c - Ph.D

DEDICATORIA

A mi esposa T.H.C,

a mis hijas: E.M.H^2 *- A.M.H*^{2}

y mi familia H.G.

AGRADECIMIENTOS

En primer lugar, le doy gracias a nuestro ser supremo, por estar conmigo en cada escalón que doy, por fortalecer mi corazón, darme la sabiduría, salud, paciencia e iluminar mi mente y sobre todo por haber puesto en mi camino a aquellas personas que han sido mi soporte y compañía durante todo el periodo de estudio.

A mis padres, Felipe Hernández y Virginia Gómez, por su apoyo incondicional en cada una de las etapas de mi vida. A mis hermanas y hermano, por estar siempre conmigo.

A mi tutor de tesis: Ph.D Antononio Parajón Guevara por su apoyo incondicional, consejos y paciencia brindada en mi formación académica. Además, por brindarme tiempo, conocimientos al aportar ideas para la revisión y/o elaboración del informe final.

A mi Asesora de Estudios: A la Dra. Irene Loiseau, quien abrió por primera vez las puertas en el campo de la investigación científica en el Departamento de Computación de la Universidad de Buenos Aires y que me ofreció un lugar en el **Grupo de Investigación Operativa, Optimización Combinatoria y Grafos**, además ha confiado en mis capacidades y cualidades, al llevarme con su equipo de trabajo a Argentina; también me ha ayudado en mi trayectoria profesional en cada uno de los momentos que le he consultado, al brindarme por primera vez temas de investigación.

Ella ha estado siempre a mi lado, sin importar la distancia, apoyándome en cada instante de mi formación. Sin ella no habría podido alcanzar este primer logro. Gracias por el apoyo, la ayuda incondicional, por su amistad y por darme la oportunidad de viajar y conocer investigadores reconocido en la Argentina y el mundo.

A cada unos de mis compañeros y amigos, por el apoyo brindado durante el tiempo que estuve en Buenos Aires, Argentina.

A mis amigos, compañeros doctorandos y comité de tesis, por aportar ideas para las mejoras de este trabajo.

A todos mis profesores de la carrera de Pregrado Matemática de la UNAN-Managua, posgrado: doctorado en Ciencias de la Computación de la Universidad de Buenos Aires, y Matemática Aplicada de la UNAN-Chontales, por los conocimientos que me compartieron a lo largo de estos años.

Agradezco plenamente a todas y cada una de las personas que han vivido las mismas experiencias en el transcurso de la realización de la tesis doctoral, con nuestros altos, bajos, y que no necesito nombrarlos. Desde lo más profundo de mi corazón les agradezco el haberme brindado toda la ayuda, colaboración, ánimo, paciencia, tolerancia, sobre todo cariño y amistad.

TRAYECTORIA ACADÉMICA DEL DOCENTE INVESTIGADOR

Fernando J. Hernández G., obtuvo su licenciatura en Ciencias de la Educación y Humanidades con mención en Matemática por la Universidad Nacional Autónoma de Nicaragua (UNAN-Managua, 2006). Es Egresado de la carrera de Doctorado en Ciencias de la Computación en la Facultad de Ciencias Exactas y Naturales Universidad de Buenos Aires, en el marco del trabajo del grupo de ***Optimización Combinatoria*** bajo la dirección de Dra. Irene Loiseau, PICT 2006-01600 (UBA,2011), en la línea de investigación ***Problemas de Ruteo de Vehículo Generalizados***, que son problemas $NP-$Hard en Optimización combinatoria. Es Máster en Matemática Aplicada en la Facultad Regional Multidisciplinaria de Chontales, de la Universidad Nacional Autónoma de Nicaragua (FAREM-Chontales, 2017). He trabajado como profesor horario en la UNAN-UPOLI-Managua desempeñandome como docente en las áreas de Matemática, Informática, Estadística, Matemáticas Financieras, Método Cuantitativos, Investigación de Operaciones. También, he trabajado como profesor de matemática de Escuela Secundaria, docente de Matemática y su didácticas en la Escuela Normal Alesio Blandón Juárez. Realiza estudio de doctorado en Matemática Aplicada coordinado por la Universidad Central **Marta Abreu** de las Villas, de la República de Cuba. Desarrolló Tesis Doctoral, en Investigación de Operaciones, bajo la tutoría del Dr. Antonio Parajón Guevara y la asesoría de la Dra. Irene Loiseau. Línea de investigación sobre el problema de Estimación de Movimiento en Compresión de Imágenes, en los que se diseñaron e implementaron algoritmos heurísticos y metaheurísticos eficientes para resolver el (EM y BM).

ABREVIATURAS

2-D: *(en inglés)* Bidimensional.

3-D: *(en inglés)* Tridimensional.

4SS: *(en inglés)* Four Step Search Algorithm.

ABC: *(en inglés)* Artificial Bee Colony Algorithm.

ARPS: *(en inglés)* Adaptive Rood Pattern Search.

BM: *(en inglés)* Block Matching Algorithm.

BMP: *(en inglés)* Windows bitmap.

CCF: *(en inglés)* Cross Correlation Function.

CCN: *(en inglés)* Normalized Cross Correlation.

CD: *(en inglés)* Czekonowsky Distance.

CD-ROM: *(en inglés)* Compact Disc Read-Only Memory.

DCT: *(en inglés)* Discrete Cosine Transform.

DFT: *(en inglés)* Discrete Fourier Transform.

DS: *(en inglés)* Diamond Search.

DVD: *(en inglés)* Digital Versatile Disc.

FS: *(en inglés)* Full Search Algorithm.

FTSS: *(en inglés)* First Technology Safety Systems.

GPU: *(en inglés)* Graphics Processor Unit.

H.261/H.263/H.264: *(en inglés)* Advanced Video Coding.

HDL: *(en inglés)* Hardware Description Language.

HDVT: *(en inglés)* High Definition Television.

IO: Investigación de Operaciones.

IDCT: *(en inglés)* Inverse Discrete Cosine Transform.

IQM: *(en inglés)* Image Quality Metrics.

ISO: *(en inglés)* International Organization for Standardization.

ITU: *(en inglés)* International Telecommunication Union.

JPEG: *(en inglés)* Joint Photographic Experts Group.

JVT: *(en inglés)* Joint Video Team.

LDSP: *(en inglés)* Large Diamond Search Pattern.

MAD: *(en inglés)* Mean Absolute Deviation.

MAE: *(en inglés)* Mean Absolute Error.

MM: *(en inglés)* The Minkowski Metric.

MPEG: *(en inglés)* Moving Picture Experts Group.

MSE: *(en inglés)* Mean Squared Error.

NASA: *(en inglés)* National Aeronautics and Space Administration.

NP: *(en inglés)* nondeterministic polynomial time.

NP-Hard: *(en inglés)* Non Deterministic Polynomial Time hardness.

PCX: *(en inglés)* Picture Exchange.

PSNR: *(en inglés)* Peak Signal to Noise Ratio.

RGB: *(en inglés)* Red, Green, Blue.

RLE: *(en inglés)* Run Length Encoding.

SAD: *(en inglés)* Sum of Absolute Differences.

SDSP: *(en inglés)* Small Diamond Search Pattern.

SN: *(en inglés)* Structural Content.

TIFF: *(en inglés)* Tagged Image File Format.

TSS: *(en inglés)* Three Step Search.

USB: *(en inglés)* Tridimensional.

VHDL: *(en inglés)* es acrónimo proveniente de la combinación de: VHSIC y HDL.

VHSIC: *(en inglés)* Very High Speed Integrated Circuit.

WAN: *(en inglés)* Wide Area Network.

WHT: *(en inglés)* Walsh−Hadamard Transform.

RESUMEN

La teoría de NP-Completitud plantea que los algoritmos exactos y eficientes son poco probable que exista para la clase de problemas $NP-$difíciles. Una forma de hacer frente a la dureza NP es relajar el requisito de optimalidad y en su lugar buscar soluciones que estén probablemente *cerca del óptimo*. Esta es la idea principal detrás de los algoritmos de aproximación, que tienen por nombre heurísticos o metaheurísticos.

El problema de estimación de movimiento es un proceso con alto grado de complejidad computacional, requiere suficiente espacio de memoria y tiempo de ejecución. Representa $\frac{2}{3}$ del costo de codificación de secuencia de imágenes estática, dinámicas y de vídeo. La principal tarea consiste en minimizar la *tasa de distorsión* y mejorar la calidad visual. Esto hace que las investigaciones en el campo de la codificación, compresión de imágenes y vídeo se centre en buscar algoritmos eficientes para llevar a cabo estimación de movimiento en un tiempo razonable.

Si se analiza una lista de imágenes de n elementos, hay 2^n soluciones factibles. Entonces, una búsqueda exhaustiva es demasiado lenta, incluso para pequeños valores del espacio de soluciones. Por lo tanto, desde un punto de vista práctico, es crucial contar con algoritmos heurísticos eficientes y rápidos que eviten la búsqueda exhaustiva.

En esta investigación diseñamos e implementamos algoritmos heurísticos, basados en el dominio de frecuencias, que se aplican sobre los coeficientes de la transformada discreta del coseno, wavelets. También, proponemos los algoritmos de dominio temporal como los de coincidencia por bloque, que centran su búsqueda en la máxima coincidencia de la imagen actual con la de referencia. Los algoritmos utilizados durante la implementación de este trabajo de investigación fueron escritos con el lenguaje de programación matemático MATLAB. Además, revisamos los conceptos básicos del procesamiento de imágenes, vídeo, algoritmos de compresión y estimación de movimiento frecuentemente utilizados.

La evaluación de los algoritmos se realizó con un conjunto de imágenes proporcionadas por un sistema de adquisición previo. Mostramos la mejora de la calidad visual, la cantidad de información comprimida o reconstruida y el comportamientos de los métodos en la búsqueda de similitudes entre píxeles o imágenes.

Por último, contribuimos a la difusión de nuevas investigación científica que conllevan a la ampliación y mejora del estudio, la generación de nuevos conocimientos, ya es un área joven dentro de la disciplina de la Educación de Nicaragua.

Palabras Claves: Estimación de movimiento, Heurísticos, Transformada discreta del coseno y Wavelets, Algoritmos de búsqueda.

ABSTRACT

Combinatorial optimization problems appear very often in everyday life. In a combinatorial optimization problem we want to minimize or maximize a function defined on a discrete set. Exact methods for combinatorial optimization problems provide always the optimal solution but in some cases there are not able to do so in an efficient way. That is, their execution times grow exponentially with the size of the problem. There are lots of combinatorial optimization problems that belongs to the NP-hard class for which no polynomial algorithm are known. In these cases the goal is to design heuristics that provide good, approximate, solutions in an affordable computational time.

Motion estimation is very hard from the computational complexity point of view.It requires a big amount of memory and running time. It represents $\frac{2}{3}$ of the cost of encoding static or dynamic image sequences or video. The task is then, to minimize the *distortion rate* and to improve the visual quality. So, the focus of the research within the field of encoding is to look for efficient algorithms for carrying out the motion estimation.

In this work we propose heuristic algorithms, based on the frequency domain, that are applied on the coefficients of the discrete cosine transform or the wavelets transform. We also propose temporal domain algorithms as those of block matching which intend to maximize the coincidence of the present image with the reference one.

We implemented the algorithms with the software MATLAB. We also review basic concepts of image and video processing and of the compression and motion estimation algorithms most frequently used.

If the list of images to be analyzed has n elements there are 2^n feasible solutions. So, an exhaustive search is too slow and then, from the practical point of view, exhaustive search has to be avoided.

The evaluation of the algorithms was done with images provided by an acquisition system. We showed the improvement of the visual quality, the amount of information compressed or reconstructed and the behavior of the methods when looking for similarities between pixels or images.

We would like to contribute to the diffusion of new research in Nicaragua. This area is pretty new in the country.

Keywords: Motion estimation, Heuristics, Discrete cosine transformation, Wavelets transformation, Local search algorithms.

Índice de figuras

Capítulo 1

INTRODUCCIÓN

El origen de la Investigación de Operaciones se remonta hace muchas décadas, con los primeros intentos en utilizar el enfoque científico en la administración de una empresa. Sin embargo, las primeras tareas fueron aplicadas a los servicios militares prestados a principios de la Segunda Guerra Mundial, ya que se tenía que asignar los escasos recursos existentes a las diferentes operaciones militares en la forma más efectiva. El buen uso de los recursos en la parte administrativa permitió el triunfo del combate aéreo inglés en la isla de Campaña en el Pacífico, de la batalla del Atlántico Norte y de muchos otros conflictos bélicos.

Los científicos e investigadores de esa época se encontraban estimulados por el evidente éxito de la investigación de operaciones en la milicia, permitiendo a las grandes industrias interesarse en este campo de investigación. Se comenzó a explorar y expandir las aplicaciones en la industria, los negocios y el gobierno. Dentro de los principales factores para el desarrollo del campo de investigación de operaciones se menciona los siguientes:

- El avance que se había realizado, para el perfeccionamiento de las técnicas disponibles en esta área. Siendo uno de los aportes esencial y sustancial, la creación del método simplex para resolver problemas de programación lineal de gran complejidad y envergadura, desarrollada en 1947 por el científico George Dantzig. Además, se puede señalar el auge y el desarrollo en gran medida de las herramientas, métodos, técnicas, y características de la investigación de operaciones, tales como: programación lineal, programación dinámica, líneas de espera y teoría de inventarios desarrolladas en su totalidad en la década de 1950.

- Debido al crecimiento teórico, se puede observar de forma inmediata el segundo factor que dio gran apertura al renacimiento y florecimiento del campo de la investigación de operaciones, siendo la aparición y aplicación de las computadoras, con el fin de resolver problemas de gran com-

plejidad, de forma eficiente e inmediato. Por tanto, problemas que al ser humano le lleva mucho tiempo de cálculos o en algunos casos imposibles de resolver, el ordenador con su capacidad para realizar cálculos aritméticos, lo venía a resolver de forma inmediata y precisa.

A continuación se menciona un recorrido de las grandes mentes que dieron aportes significativos al área de Investigación de Operaciones. En 1951, se introdujo por completo en el Reino de Gran Bretaña y Estados Unidos. En 1759 los economistas franceses François Quesnay comienza a emplear modelos primitivos de programación matemática, y en 1874 Léon Walras utiliza herramientas análogas.

Los modelos lineales de la Investigación de Operaciones tienen como fundador a Wilhelm Jordan, conocido por la creación de su algoritmo de Eliminación de Gauss-Jordan, aplicado para resolver el problema de mínimos cuadrados, técnica algebraica que apareció en la obra Handbuch der Vermessungskunde en 1873. También, se puede señalar a Hermann Minkowsky en 1896 y a Karl Farkas en 1903, que aplicaron técnicas similares y realizaron significativos aportes a la ciencia.

Los modelos dinámicos probabilísticos dieron inicios con los aportes del matemático ruso Andréi Andréyevich Márkov a fines del siglo pasado. El desarrollo de los modelos de inventarios, de tiempos y movimientos, se da en los años veinte de este siglo, mientras que los modelos de línea de espera a principios del siglo XX con los trabajos realizados por Agner Krarup Erlang y en la actualidad existe un lenguaje de programación en su honor, que lleva nombre Erlang.

Los problemas de asignación aparecen en las investigaciones de los matemáticos Johann Konig - Egervary, proponiendo el método húngaro, y su principal aporte se centra en los algoritmos de optimización el cual resuelve problemas de asignación en orden de complejidad computacional teórica $O\left(n^3\right)$. El ruso Leonid Vitálievich Kantoróvich en 1939, se centra en el estudio de los problemas de distribución.

El matemático y físico John Von Neumann en 1937 realizó grandes aportes a la ciencia y en especial al área de las ciencias de la computación, con la presentación del modelo de la máquina de Von Neumann o arquitectura Princeton, que es una arquitectura de computadoras basada en la descrita en 1945, cuya estructura es la que se utiliza en los procesadores modernos. También, da aportes a la teoría de preferencias o teoría de juegos en unión con Morgenstern.

Hay que señalar y dar el crédito que los modelos matemáticos utilizados en el campo de la Investigación de Operaciones empleados por los pioneros o precursores, se centraron en las teorías de los trabajos de Cálculo Diferencial e Integral de las grandes mentes de matemáticos como: Isaac Newton, Joseph-Louis Lagrange, Pierre-Simon Laplace , Henri Léon Lebesgue, Gottfried Wilhelm Von Leibniz, Georg Friedrich Bernhard Riemann, Thomas Joannes Stieltjes, entre otros científicos e investigadores célebres de la historia de la Matemática. Así como, en la teoría de Probabilidad y la Estadística de:

Daniel Bernoulli, Siméon Denis Poisson, Johann Carl Friedrich Gauss, Thomas Bayes, entre otros.

En 1947 George Dantzig, realiza un aporte valioso al área de la investigación de operaciones e inventa el método Simplex, dando primicia a la línea de investigación de Programación Lineal. Que con el auge de los ordenadores digitales se empezó a extender. Más tarde aparece el matemático aplicado Richard Ernest Bellman cuyos trabajos versan sobre la metodología Programación Dinámica. Además, da inicio la Programación No Lineal con los investigadores Harold William Kuhn y Albert William Tucker, el matemático Ralph Edward Gomory realiza principales aportes acerca la Programación Entera.

Dentro de los principales precursores o pioneros de la Investigación de Operaciones podemos mencionar Charles Churchman, Russell Ackoff y Arnoff Leonard, ya que fueron los primeros en realizar publicaciones para definir el campo. También, podemos señalar dentro de la líneas la programación lineal Gaspard Monge en 1776, y Leonid Vitálievich Kantoróvich como uno de los fundadores.

Las definiciones de Investigación de Operaciones varían desde técnicas matemáticas específicas utilizadas hasta el método científico en sí. La Mayoría de las definiciones incluyen tres aspectos básicos al enfoque de la $I.O.$, para la toma de decisiones administrativas; las que se menciona a continuación:

- Una visión sistemática del problema a ser tratado y resuelto.
- Una concordancia en cuanto al uso de método científico en la solución de los problemas.
- La utilización de técnicas específicas de estadística, probabilidad y modelos matemáticos, y algoritmos para facilitar a quien toma las decisiones en la solución del problema.

En el campo de la Optimización encontramos Delbert Ray Fulkerson con el desarrollo de nuevas teorías matemáticas y algoritmos de optimización, así como sus aplicaciones prácticas a problemas de planificación.

La Optimización Combinatoria es una rama de la teoría de Optimización en matemáticas aplicadas y en Ciencias de la Computación, que se encuentra estrechamente asociada a la Investigación de Operaciones o Programación Matemática, teoría de algoritmos y teoría de la complejidad computacional. Los algoritmos desarrollados resuelven instancias de problemas que suelen ser difíciles o en algunos casos imposibles de resolver; en general se explora un espacio de soluciones, relativamente grande, para ellas. Estos algoritmos lo logran reduciendo el tamaño del espacio, explorando y explotando al máximo las propiedades de forma eficiente y rápida. Con el desarrollo del estudio de la teoría de la complejidad computacional, es posible entender la importancia de los mismos.

En las ciencias matemáticas, optimización o programación matemática lo define como el proceso de selección del mejor elemento, con respecto a algún criterio, de un conjunto de elementos disponibles.

Los casos más sencillos de un problema de optimización consisten en maximizar o minimizar una función objetivo, tales como: maximizar los ingresos, los bienes, las utilidades o minimizar los costos de mano de obra de una empresa, la minimización de emisiones lumínicas, minimización de residuos, entre otros. La generalización de la teoría de la optimización y técnicas para otras formulaciones comprende una gran área de las matemáticas aplicadas. De forma general, la optimización incluye el descubrimiento de los "mejores valores" de alguna función objetivo dado un dominio definido, incluyendo una variedad de funciones objetivos y diferentes tipos de dominios.

Por ejemplo: Dada una función $f : A \longrightarrow R$.
Buscar: un elemento x_0 en A tal que $f(x_0) \leq f(x)$ para todo x en A "minimización" o tal que $f(x_0) \geq f(x)$ para todo x en A "maximización". A esta formulación se denomina problema de optimización o un problema de programación matemática.

La mayoría de problemas tratado desde las perspectivas teóricas y del mundo real se modelan bajo este enfoque general. Los problemas enunciados y formulados utilizan este tipo de modelos, enfoques y técnicas en los diferentes campos tales como: la física, algoritmia y visión por computadora, etc. refiriéndose a la minimización de la energía, como el valor de la función f que representa toda la energía del sistema de referencia que está siendo modelado.

Cabe señalar, que el término programación no se refiere a la programación de computadoras que se utiliza en las áreas de ciencias de la computación e informática o áreas afines, sino que, el término viene del uso de programa por el ejército de Estados Unidos al referirse a la propuesta de entrenamiento y planificación logística, el cual fue el problema estudio en aquella época por el científico George Dantzig. A continuación, se mencionará una buena lista, de aquellos investigadores que dieron sus valiosos aportes al avance del campo de la optimización matemática, dentro de ellos tenemos:

Albert Tucker	Joseph Louis Lagranger	Leonid Khachiyan
Arkadi Nemirovski	Naum Z. Shor	Lev Pontryagin
Bernard Koopman	Narendra Karmarkar	László Lovász
Boris Polyak	Richard Bellman	Tyrrell Rockafellar
Harold W. Kuhn	Ronald A. Howard	Yurii Nesterov
James Renegar	Michael J. Todd	William Karush

También, podemos mencionar una inmensa lista de las líneas de investigación en las cuales realizaron sus avances científicos y que en la actualidad siguen siendo objetos de estudio:

Programación convexa	Programación lineal
Programación entera	Programación semidefinida
Programación geométrica	Programación cónica
Programación de cono de segundo orden	Programación cuadrática
Programación fraccionaria	Programación no lineal
Programación robusta	Programación disyuntiva
Programación dinámica	Optimización combinatoria
Optimización de la WAN	Optimización estocástica
Optimización dimensional-infinita	Optimización discreta
Optimización Multiobjetivo	Optimización de topología de multifase
Optimización de consultas	Heurísticas y Metaheurísticas, etc.

Dentro del campo de la Investigación de Operaciones o programación matemática ha surgido una serie de procedimientos heurísticos y metaheurísticos para dar solución a estos problemas, los cuales son flexibles al involucrar características específicas y permiten la interacción con el decisor para buscar aproximaciones a la solución ideal, sobre la base del desarrollo tecnológico en el área de la computación, pues generalmente los procedimientos heurísticos son iterativos y requieren de una gran cantidad de cálculos.

El término de "Heurística" es un concepto que se originó en la Grecia clásica, de la palabra griega *heuriskein* que significa encontrar o descubrir; según la historia se deriva de *eureka*, famosa exclamación atribuida a Arquímedes.

De esta manera, se define heurístico como un procedimiento en que la solución no se determina en forma directa, sino mediante ensayos, pruebas y reensayos; obteniéndose un alto grado de confianza en las soluciones de calidad a un costo computacional razonable, aunque no se garantice la solución óptima o su factibilidad, en algunos casos, no se llega a establecer lo cerca que se encuentra de la solución óptima.

Siempre en la búsqueda de mejores soluciones sobre el espacio de soluciones y de buena calidad, la investigación en este campo en los últimos años, ha centrado su atención en el diseño de técnicas de propósito general para orientar la construcción de soluciones a partir de las diferentes heurísticas. Estas técnicas se denomina metaheurística y son estrategias para diseñar y/o mejorar los procedimientos heurísticos orientados a obtener un alto rendimiento en las soluciones de problemas que son difíciles de resolver.

En los últimos años, ha aparecido una serie de métodos bajo el nombre de metaheurísticas con el propósito de obtener mejores resultados que los alcanzados por los métodos heurísticos tradicionales. Estos

surgen como un intento de producir una sinergia, es decir, que se pueden alcanzar mejores resultados haciendo uso de una combinación de métodos, proporcionando un marco general para crear o desarrollar nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos, entre otros.

El sufijo *meta* significa *más allá*, las metaheurísticas son estrategias para diseñar o mejorar los procedimientos heurísticos con miras a obtener un alto rendimiento. El término metaheurística fue introducido en [21] por (Glover, 2006), y a partir de entonces han aparecido muchas propuestas para el diseño de mejores procedimientos de solución a problemas combinatorios.

A continuación, se muestra algunos métodos, técnicas o algoritmos de búsqueda aproximados, heurísticos y metaheurísticos más utilizados para resolver problemas clásicos de orden de complejidad computacional teórica $NP - duro$, tales como:

- Algoritmos de búsqueda de soluciones óptimas,
- Algoritmos voraces,
- Algoritmos evolutivos,
- Algoritmos multiarranque,
- Algoritmos de enjambre,
- Algoritmos genéticos,
- Algoritmos meméticos,
- Algoritmos de aceptación por umbrales,
- Algoritmos de ascensión de colinas,
- Algoritmos de ascensión de colinas con reinicialización aleatoria,
- Backtracking,
- Branch & Bound,
- Búsqueda exhaustiva,
- Búsqueda primero el mejor,
- Búsqueda local,
- Búsqueda local iterada,

- Búsqueda en entornos variados,
- Búsqueda tabú,
- Búsqueda dispersa,
- Búsqueda basada en trayectorias,
- Procedimientos de búsqueda voraces aleatorizados y adaptativos,
- Optimización basada en colonias de hormigas y de abejas,
- Optimización aleatoria,
- Recocido simulado,
- Re-encadenamiento de trayectorias, entre otros.

Los algoritmos genéticos básicos fueron propuestos en [26] por (Holland, 1975), y se encuentran en los textos: [23] descritos por (Goldberg, 1989), en [18] desarrollados por (Davis, 1991), en [60] de (Michalewicz, 1992), en [45] definidos (Reeves, 1993 - 1996), y en [35] implementados por (Lee, 2008).

Mediante el estudio de la teoría Complejidad Computacional es posible comprender la importancia de estos tipos de problemas que en su mayoría son difíciles de resolver o imposible; pertenecen a la clase de problemas de optimización combinatoria $NP - Hard$, conforme crece el tamaño de los casos. Además, cabe señalar, que no existe una metodología única que los resuelva de forma eficientemente y exacta, o bien, si existe el tiempo de cómputo es muy elevado en casos a gran escala o en el caso que el espacio de búsqueda sea demasiado grande. Dichas instancias a menudo tienen ramificaciones teóricas - prácticas muy importantes, apareciendo en un gran número de aplicaciones en problemas que se presentan en la industria, logística, ciencias, ingenierías y en la administración, es decir en la vida real como en casos de estudios académicos, que se analiza desde el punto de vista teórico.

Es difícil realizar un análisis simple de un algoritmo que determine la cantidad exacta de tiempo que éste requiere para ser ejecutado, porque depende en gran parte de la eficiencia en la implementación del algoritmo y de los procesos de ejecución de la computadora. Además, conocer el tiempo exacto que tardará el cómputo o cálculo en generar los resultados, es una tarea difícil de encontrar, la principal tarea consiste en calcular la cantidad de operaciones que se realiza de acuerdo con los datos de entrada del problema a tratar, esto se conoce como "cálculo de la función temporal" y es utilizado desde el punto de vista teórico dentro de la teoría de complejidad computacional en [16] por (Cook,1971).

Una vez que se cuenta con un algoritmo que funciona de manera idónea, la siguiente tarea es definir los criterios que permitan medir su rendimiento o comportamiento, es decir, se ha de considerar el uso

eficiente de los recursos y la simplicidad del algoritmo. Un algoritmo sencillo no le quita el crédito o desmerita la calidad, ya que su simplicidad facilita su mantenimiento, su verificación y su eficiencia.

Al hablar del uso eficiente de los recursos, éste puede medirse en función de dos indicadores: *el espacio*, cantidad de memoria que utiliza, y el *tiempo* que tarda en ejecutarse. Si para resolver un problema P, un algoritmo A, necesita de poca memoria de ejecución en un equipo con un pequeño número de instrucciones comparado con el resto de los algoritmos conocidos que resuelven el P, entonces se puede afirmar que A es más eficiente que los restantes, cuando se resuelve P.

Las principales líneas de estudio en el campo de la Visión Artificial son: generar descripciones inteligentes y útiles de: escenas, films, secuencias visuales sobre imágenes y vídeos, haciendo énfasis en características esenciales, objetos relevantes que aparecen en ellas. Algunos de los problemas que se estudia en está área son:

- La detección de movimiento, cuyo objetivo principal consiste en detectar la estimación movimiento en la escenas de una película, o pequeños films, compresión de imágenes digitales.
- Detección y localización de objetos en movimiento. Es uno de los problemas más complicados para la reconstrucción de escenas de películas, incluyendo también la detección y predicción de futuras trayectorias.
- Obtención de propiedades $3-D$ de objetos a partir del movimiento. Es uno de los problemas clásicos de Visión Artificial.
- Un cuarto grupo de estudio que está en relación con el primero: Estimación de movimiento en problemas de compresión de imágenes y vídeo.
- Detección, reconocimiento, identificación, seguimiento de objetos, calibración de cámaras y visión estéreo, etc.

1.1. MOTIVACIÓN

Con el auge de la era multimedia y la difusión del Internet, almacenamiento de vídeo en CD-ROM / DVD / USB / Discos Duros y las Video llamadas en líneas en computadoras: de escritorio, personales, tablets, teléfonos, estéreos, etc. ha ido ganando mucha divulgación. Los estándares de codificación de vídeo del Grupo de Expertos de Imágenes ISO (MPEG) se refieren al almacenamiento de imágenes y vídeo comprimido en medios físicos como CD-ROM / DVD / USB, disco duro; donde La Unión Internacional de Telecomunicaciones se ocupa de las comunicaciones punto-a-punto o multipunto en

tiempo real. Comunicaciones a través de una red y tiene la ventaja de tener mayor ancho de banda para la transmisión de datos, para las vídeos llamadas o conferencias.

Toda la idea detrás de la compresión de imágenes, secuencias o vídeo, consiste en la codificación de la Estimación de Movimiento con el frame [1] actual con respecto al frame previo. Se crea entonces una imagen compensada por movimiento para el frame actual que está construida de bloques de la imagen desde la frame anterior. Se envían los vectores de movimiento para los bloques utilizados para la estimación de movimiento, así como la diferencia de la imagen compensada con el frame actual; también se codifica y envía $JPEG$. La imagen codificada que se envía, se decodifica en el codificador y se utiliza como marco de referencia para los frame siguientes. El decodificador invierte la proceso y crea un marco completo.

Cabe señalar que el primer marco siempre es enviado completo, y también lo son algunos otros marcos que pueden ocurrir en algún intervalo regular. Los estándares no especifican esto y podría cambiar con cada imagen, en escala gris o color, o vídeo que se envía basado en la dinámica del vídeo.

En general, para poder emplear los métodos, técnicas, modelos matemáticos y algoritmos de estimación de movimiento entre pares de imágenes, se establecen los siguientes supuestos sobre la secuencias de las imágenes adquiridas:

- Color/brillo del píxel constante, en el caso de imágenes a color.
- Los píxeles en una misma superficie se mueven de forma similar, coherentemente.
- Movimientos “suaves” y continuos;

La creciente complejidad de los sistemas de Visión Artificial hace que la posibilidad de probar fácilmente distintos algoritmos y métodos alternativos para llevarlos a cabo, suponga un ahorro considerable a la hora de diseñar e implementar dichos sistemas de visión artificial. Algunas de sus aplicaciones principales destacan:

- Detección de movimiento aplicado a seguridad,
- Robótica y navegación Robótica,
- Reconocimiento óptico de caracteres,
- Construcción de modelos $3 - D$,
- Aplicaciones médicas, biología, geología y meteorología,

[1] “es cada una de las imágenes instantáneas en las que se divide una película de cine que dan sensación de movimiento al ser proyectadas secuencialmente” (Konigsberg, 2004, p. 235).

- Seguridad automovilística,
- Captura de movimiento,
- Vigilancia en los parqueos, zonas residenciales, etc.
- Biométrica, control de calidad, etc.

El análisis de movimientos en la restauración y compresión de imágenes, de escenas de películas, secuencias de imágenes y vídeos es uno de los problemas principales en lo que respecta a la extracción de información, características y pistas relevantes para el análisis posterior de su contenido. La detección y segmentación de objetos se ven ampliamente beneficiadas si se cuenta con *algoritmos eficientes para estimación de movimientos*. Es por tal razón, que se tiene por objetivo obtener una buena correspondencia de puntos entre imágenes, secuencias de imágenes e imágenes consecutivas de un vídeo y así lograr la discriminación de los objetos según su movimiento.

1.2. ESTADO DEL ARTE

Los métodos más aplicados para estimación de movimientos son los de análisis de comparación por bloques bajo el criterio de máxima similitud. En la búsqueda de los algoritmos de estimación de movimientos para vídeo se ha de seleccionar aquel con mejor desempeño, en la que se tendrá en cuenta los métodos de comparación tanto en la velocidad, y su costo de complejidad computacional, así como en su precisión.

La operación más costosa desde el punto de vista computacional y con pocos recursos en todo el proceso de compresión es la Recuperación de imágenes digitales y la Estimación de Movimiento de imágenes y vídeos. Por lo tanto, este campo ha visto la mayor actividad e interés de investigación en las últimas dos décadas; en la que se implementa y evalúa los algoritmos de coincidencia de bloques fundamentales desde mediados de la década de 1980, los recientes algoritmos de comparación rápida de bloques del año 2002, hasta la actualidad.

Los métodos de estimación de movimiento más básicos se basan, en la diferencia entre imágenes consecutivas; la mayor parte de la totalidad de los trabajos sobre estimación de movimiento se centra sobre la hipótesis de restricción del flujo óptico u optical flow propuesto en [39] por (Lucas, 1981) y en [25] se postula que la intensidad de un punto al que seguimos a lo largo de su movimiento permanece constante, desarrollado e implementado por (Herbst, 2013).

El algoritmo genético planteado en [33] por (Kung, 2014) examina el error cuadrático medio más próximo al algoritmo, con el fin de proponer una nueva técnica que combina el algoritmo de búsqueda

basado en hexágono, que es más rápido que en la búsqueda de diamantes, y el algoritmo genético. En este se realizan experimentos para demostrar la velocidad de codificación y la precisión del método de patrón de búsqueda basado en hexágono y el método propuesto.

En [17] se realiza un estudio de la estimación de movimiento de coincidencia de bloques, el que desempeña un papel muy importante en la codificación de vídeo propuesto por (Cuevas, 2013). En este se plantea que, la estimación de movimiento se puede abordar como un problema de optimización, donde el objetivo es encontrar el mejor bloque de coincidencia dentro de un espacio de búsqueda. Además, se propone un nuevo algoritmo basado en la optimización de la Colonia de Abejas Artificiales (ABC) para reducir el número de ubicaciones de búsqueda en el proceso BM.

En este algoritmo, el cálculo del espacio de búsqueda se reduce drásticamente mediante la consideración de una estrategia de cálculo de aptitud que indica cuándo es factible calcular o sólo estimar nuevas ubicaciones de búsqueda. Dado que el algoritmo propuesto no considera ningún patrón de búsqueda fijo o cualquier otra suposición de movimiento como la mayoría de otros enfoques BM, existe una alta probabilidad de encontrar el mínimo real (Vector de movimiento preciso). Las simulaciones conducidas demuestran que el método propuesto alcanza el mejor equilibrio sobre otros algoritmos rápidos del BM, en términos de la exactitud de la estimación y del costo computacional.

En [59] se evaluá e implementa cuatro algoritmos de coincidencia de bloques utilizando la estimación de movimiento aplicados por (Yaakob, 2013). Además, se evalúan los efectos del tamaño del bloque empleado para encontrar el mejor algoritmo entre ellos y el más óptimo. Los resultados obtenidos sugieren que entre todos los algoritmos evaluados, $ARPS$ tiene el mejor $PSNR$ basado en el tiempo de cómputo.

En [54] se plantea que los algoritmos de compresión de vídeo tienen un gran número de aplicaciones que van desde la videoconferencia hasta el vídeo a pedido o televisión a la carta y los teléfonos de vídeo presentado por (Verma, 2012). Además, se reduce el requisito de alto ancho de banda en gran medida. La adaptación de bloques es un método ampliamente utilizado para la visión estéreo, el seguimiento visual y la compresión de vídeo. La estimación de movimiento eficiente es un problema importante porque determina la eficiencia de compresión y la complejidad de un codificador de vídeo.

En este se presenta una arquitectura paralela para un algoritmo de búsqueda rápida en tres pasos para la estimación de movimiento, que implica un número reducido de puntos de verificación comparados con el algoritmo estándar de búsqueda en tres pasos. La estimación de movimiento de concordancia de bloques es la parte más importante del sistema de codificación de vídeo. Se ha demostrado que la degradación del rendimiento al aplicar el algoritmo mejorado a varias imágenes estándar es insignificante en comparación con el algoritmo estándar. La arquitectura propuesta utiliza solamente tres elementos de

procesamiento acompañados con el uso de la disposición inteligente de datos y la configuración de la memoria. Debido a su bajo requerimiento de área y a los bajos requerimientos de ancho de banda de memoria, la arquitectura propuesta proporciona una solución eficiente para estimaciones de movimiento en tiempo real como requerido por aplicaciones de vídeo de varias velocidades de datos, desde vídeo de baja velocidad de bits a sistemas $HDTV$ y $FTSS$ es mucho más robusto, produce menor error de compensación de movimiento y tiene una complejidad computacional muy compatible.

En [19] se ofrece una clasificación de los métodos propuestos por (Dufaux y Moscheni, 2010) y los dividen en cuatros categorías: Las técnicas de gradientes, recursividad, comparación de bloques y las de dominio de frecuencias.

El seguimiento preciso del movimiento para un gran conjunto de puntos es una tarea que ha sido abordada en muchas ocasiones en [11] por (Brox, $2004 - 2014$) y que aún es objeto de estudio para la comunidad científica.

En [38] centran su interés en el estudio del algoritmo de búsqueda en tres etapas para la estimación de movimiento en coincidencia bloques, debido a su simplicidad, reducción computacional significativa y buen rendimiento, se ha utilizado ampliamente en aplicaciones de vídeo en tiempo real, analizados, desarrollados e implementados por (Liou, 1997). Además, se propone un nuevo algoritmo de búsqueda para una mayor reducción en el orden de la complejidad computacional para la estimación de movimiento y demuestra que el algoritmo propuesto es simple, eficiente y que requiere alrededor de la mitad del cálculo para TSS manteniendo la misma regularidad y buen desempeño.

Como se ha mencionado los algoritmos de estimación de movimientos utilizados en gran medida a lo largo de la historia, dentro de los cuales podemos detallar algunos de ellos: Algoritmo de búsqueda exhaustiva, Algoritmo de búsqueda en tres pasos, Algoritmo de búsqueda en cuatros pasos, Algoritmo de búsqueda en diamante, Algoritmo de búsqueda logarítmica, Algoritmo de búsqueda en cruz, Algoritmos de búsqueda de los vecinos más próximos, entre otros con sus variantes.

La operación más costosa desde el punto de vista computacional y con pocos recursos en todo el proceso de compresión de imágenes o vídeo consiste en la recuperación y en la *Estimación de Movimiento*. Por lo tanto, este campo ha visto la mayor actividad e interés de investigación científica en las últimas décadas. Es por tal razón, que nos motiva a investigar, implementar y evaluar los algoritmos de coincidencia de bloques fundamentales desde mediados de la década de 1980, los recientes algoritmos de comparación rápida de bloques del año 2002, hasta la actualidad basados en la selección y transformación de los coeficientes a través de la DCT, DFT y los algoritmos de máximas coincidencias.

1.3. PLANTEAMIENTO DEL PROBLEMA

Una de las partes fundamentales del trabajo se centra en el campo de las Ciencias de la Matemática, Computación e Informática, ya que, ha sido tradicionalmente el diseño e implementación de algoritmos cada vez más eficiente para la solución de problemas, tanto de optimización como de búsqueda. La investigación de algoritmos exactos, heurísticos y metaheurísticos para resolver problemas de optimización combinatoria, puesto que nos enfrentamos a nuevos problemas de aplicación en las diferentes áreas del conocimiento, al mismo tiempo que disponemos de nuevos recursos tecnológicos y computacionales, tales como nuevos tipos de ordenadores con un poder de cálculo mucho mayor, redes y entornos virtuales, como el Internet, y clúster computadores con mejores procesadores.

Los problemas de optimización combinatoria como los problemas de búsqueda abundan en la vida cotidiana. Desde el punto de vista teórico, un problema de este tipo equivale a encontrar una solución con un costo mínimo o máximo entre un gran número de soluciones factibles. Se dice que un algoritmo para un problema de optimización dado es exacto si siempre encuentra la solución óptima y se dice que es eficiente si se ejecuta en tiempo polinómico según su tamaño entrada.

La principal ventaja de la utilización de algoritmos exactos es que garantizan encontrar el óptimo global de cualquier problema, pero tienen el inconveniente que en problemas reales de complejidad computacional NP o $NP - Hard$ su tiempo de ejecución crece de forma exponencial conforme crece el número instancia o espacio de soluciones analizados. Es por tal razón que se analizará los heurísticos que suelen ser bastante rápidos, pero la calidad de las soluciones encontradas suele ser bastante mala. Las metaheurísticas utilizadas ofrecen un equilibrio entre ambos extremos; son métodos genéricos que ofrecen una buena solución y en muchos casos se obtiene óptimo global en un tiempo de cómputo razonable.

El desarrollo de Algoritmos de Estimación de Movimiento en Comprensión de Imágenes es un área de la visión artificial que ha tenido avances significativos en las últimas décadas. Dentro de sus aplicaciones pueden mencionarse: detección de movimiento aplicado a seguridad (reconocimiento de huellas dactilares, rostros, reconocimientos de iris y retina, vigilancia, etc.) de un banco o en las terminales autobuses, o de los aeropuertos, entre otros; el seguimiento visual, detección, reconocimiento, localización de objetos en movimiento, identificación de una serie de características locales(objetos, bloques, segmentos, etc.) en imágenes, incluyendo detección y predicción de trayectorias; obtención de propiedades en $2 - D$; la estimación de movimiento en problemas importante porque se determina la eficiencia de compresión de imágenes digitales, vídeo, control de calidad y la complejidad de un codificador de vídeo.

Dado que el procesamiento de imágenes, de escenas de películas, secuencias de imágenes o vídeos

requiere más recursos computacionales que en el análisis de imágenes estáticas, ya que muestran una cantidad de información para ser procesada de forma rápida y eficiente. Estos hechos implican, generalmente, que los métodos de estimación de movimiento de alta calidad sean bastante lentos y requieran de la adquisición de costosos equipos o clústers de computadoras para acelerarlos. Una solución a estos problemas es la implementación de los algoritmos en placas gráficas utilizando técnicas de procesamiento paralelo en la GPU. Esto permite realizar el procesamiento en tiempo real, con la desventaja adicional de que puede realizarse en computadoras personales a altos costo.

Los procesos visuales en el procesamiento de imágenes y vídeos son caracterizados utilizando modelos, técnicas heurísticas o metaheurística y estadísticos, algoritmos que describen el comportamiento de cada píxel a lo largo del tiempo. La teoría de NP-Completitud sugiere que los algoritmos exactos y eficientes son poco probable que exista para la clase de problemas NP$-$difíciles. Una forma de hacer frente a la dureza NP es relajar el requisito de optimalidad y en su lugar buscar soluciones que estén probablemente cerca del óptimo. Esta finalidad principal de desarrollar los algoritmos de aproximación, que tienen por nombre heurísticos o metaheurísticos.

Con base a esto, estamos interesados en diseñar, implementar y proponer nuevas metodologías numéricas, modelos particulares, métodos específicos para resolverlo, algoritmos computacionales para su solución eficiente y aproximada mediante heurísticos y metaheurísticas, para la más costosa de las tareas como: ***estimación de movimiento*** que corresponde a los algoritmos de búsqueda sobre el macrobloque de referencia, centradas en el análisis de los algoritmos de dominio de frecuencia y de dominio temporal.

El procedimiento consiste en encontrar un conjunto de esquemas permitidos, aquel que tenga un costo asociado y al reconstruir la imagen completa sea buena o de calidad visual óptima. Cabe señalar que los errores que se presentan con frecuencia al reconstruir las imágenes son: ***el efecto bloque y el efecto sombras***, debido a la cantidad de información analizada. La naturaleza de estos subproblemas hace que rara vez pueden ser tratados teóricamente debido a la gran cantidad de imágenes que se procesan y de las características estudiadas. Por lo tanto, nos vemos motivados a eludir el cálculo directo y considerar la utilización de ordenadores y algoritmos específicos que nos auxilien en nuestra tarea.

Por ejemplo, si tomamos como muestra una imagen de 512×16 píxeles, y con un valor de $p = 16$, el número total de operaciones por imagen es de $82{,}85 \times 10^8$. Con lo que para una secuencia con $30\ fps$, el número se incrementa a $98{,}55 \times 10^9$operaciones, lo que resulta una cifra astronómica. Por tanto, si tomamos una imagen tamaño más grande, o una conjunto de imagen esto tiende hacer impracticable.

Siendo ésta, una de las dificultades imprevistas, la cual radica en el tiempo que demora el ordenador en reconstruir la solución, puesto que muchos de estos problemas de optimización combinatoria pertenecen a la clase $NP-hard$ tratados en [9] por (Garey y Johnson, 1979). Es decir, el tiempo de cómputo

y de construcción que se requiere se incrementa conforme crece el tamaño de la imagen, el número de imágenes analizadas o de las características en estudio.

Los estudios de hibridación y paralelismo algorítmica pueden realizarse en múltiples entornos y familias. Así, tomaremos como hipótesis de trabajo el campo de investigación de algoritmos, algoritmos heurísticos, metaheurísticas en general. Estos representan una plantilla genérica de optimización que da lugar a multitud de distintas técnicas concretas que hoy en día están proporcionando grandes éxitos en optimización combinatoria, ingeniería, economía, telecomunicaciones, bioinformática, visión artificial, inteligencia artificial, biología molecular y otras aplicaciones muy variadas propuesta en [8] por (Blum y Roli, 2003).

Después de evaluar varias áreas de estudio, con la ayuda de mi director de tesis y mi consejera de estudio se eligió el tema: ***Algoritmos Heurísticos de Coincidencia Para la Estimación de Movimiento en Compresión de Imágenes***. Los algoritmos a tratar son los siguientes: Los algoritmos de dominio de frecuencia que se aplican sobre los coeficientes de las transformadas:**DFT, DCT, Wavelets**) y los algoritmos de dominio temporal (**block matching o de coincidencias**) tales como: el algoritmo de búsqueda exhaustiva, algoritmo de búsqueda en tres pasos, algoritmo de búsqueda en cuatros pasos, algoritmo de búsqueda en diamante, algoritmo de mejora del heurístico de búsqueda en tres pasos simple y eficiente, heurístico mejorado en la búsqueda de patrón en tres pasos, algoritmo de búsqueda logarítmica, algoritmo de búsqueda en cruz y algoritmos de búsqueda de los vecinos más próximos, entre otros.

La importancia de elegir el tema se debe a que es original y de alcance a nuestra realidad, pues aborda problemas que aportan resultados prácticos para la solución de los mismos. De igual manera, el enfoque del tema corresponde a la disciplina de los estudios de doctorado en las universidades a nivel nacional e internacional, refleja interés social y tecnológico, puesto que gira alrededor de una de las problemáticas del mundo moderno, que es la optimización de los recursos de compresión, transmisión y almacenamiento de información visual.

La finalidad de esta tesis se centra en el diseño e implementación de algoritmos de coincidencia basados en la Estimación de Movimiento, que requieren de las mejores implementaciones de algoritmos de dominio frecuencias basados en las transformadas de los coeficientes a través de DFT, DCT y $Wavelets$, así como los algoritmos de coincidencia que pertenecen al dominio temporal, que ha sido escasamente abordada en Nicaragua desde el punto de vista teórico y computacional.

Al implementar e intentar resolver alguno de estos problemas de estimación de movimientos dentro de la compresión imágenes y vídeo, al igual que otros casos de problemas de optimización combinatoria que pertenecen a la clase $NP - hard$, hay que proponer un modelo particular y desarrollar métodos específicos para resolverlo, (aunque algunas ideas usadas al resolver uno de ellos puedan servir para

otros, por ejemplo en el caso de desarrollar un algoritmo de "búsqueda de tres pasos" servirá de base para desarrollar la búsqueda en cuatro paso, y poder generalizar, si se implementa o desarrolla algún tipo de TSS para una variante puede servir de ayudad para las propuestas de otra aunque no haya sido de mucho éxito, o las ideas de un heurístico exitoso en uno de ellos pueden ayudar elaborar algoritmos que provean buenas soluciones para otra, es decir reducir el error en la recuperación, transmisión de las imágenes y vídeos, así como suavizar la estimación o predicción de movimientos en la búsqueda de los objetos en estudio.

1.4. JUSTIFICACIÓN DEL PROBLEMA

El potencial de la investigación radica en intentar resolver este problema que es de ***carácter teórico - aplicado***, ya que al diseñar, implementar y evaluar los algoritmos de coincidencia para la estimación de movimientos con el fin obtener una mejora mínima en la solución, dar otras alternativas; en las que se puede llegar a obtener buena ***eficiencia algorítmica***, siendo de **relevancia social** ya que estas mejoras generan desde el punto económico grandes ganancias a las industrias cinematográficas, sistema de seguridad informáticas o instituciones afines.

Otra de las razones que nos motiva el estudio, es su ***utilidad metodológica***, puesto que puede servir de bases para posteriores investigaciones en Nicaragua, ya que aún no se han realizado, es un tema poco investigado y de interés para la comunidad científica, servirá de soporte para ciertos cursos que se podrían implementarse en las diferentes universidades del país, por lo que es un campo nuevo o nulo dentro del ámbito de la Educación en Nicaragua, las distintas universidades de nuestro país no ofertan ni un curso o conferencias relacionada con el tema en estudio, en que la nueva generación haga uso y que sea de utilidad para la sociedad.

También, la importancia e interés del estudio radica en que se dejará un soporte del trabajo realizado tanto de la parte teórica como de la parte experimental, así como sus implementaciones en el lenguaje de programación $MATLAB$ que le pueda ser de utilidad o de apoyo a los diferentes estudiantes de las carreras de Ciencias de la Computación, Sistema, Informática, Matemática, y otras carreras afines, que se encuentren pensando en trabajar en esta área y que sea asequible para ellos y al público en general.

También, tiene *Implicaciones prácticas*, debido a que sus resultados permite resolver un conjunto de problemas del área de investigación de operaciones, visión artificial, diversas área del conocimiento y de la vida cotidiana. Por ejemplo, los resultados obtenidos se aplican a problemas de la sociedad (como la mejora de una fotografía antigua, o reconstrucción algunas que se encuentre en mal estado, es decir agregar ciertos coeficientes para mejorar su calidad visual, en el caso de salas de cine proyectar una

imagen o película con mayor amplitud nos generan ciertos efectos de mala calidad, o en caso de una vídeo llamada, etc.). Además, es aplicable en el análisis de imágenes médicas, ya que, puede convertirse en una herramienta de apoyo para el diagnóstico en las que necesitamos mayor visión y precisión. De la misma manera, nos puede servir de soporte en los diferentes campos en los que se necesita resguardar evidencias tales como: la genética, la botánica, la zoología, la patología, la medicina forense, el diagnóstico por imágenes, así como instituciones de la salud, vigilancia y el ejército. También, es aplicable desde el punto de vista de la mercadotecnia, la publicidad, los cuales desean vender y ofertar sus productos a los clientes de una forma visual y eficaz, etc.

De igual manera, tiene una **una viabilidad o factibilidad de ejecución**, siendo una de las razones principales que motivan el estudio, puesto que el único equipamiento necesario para llevar a cabo la investigación es una computadora personal con buen poder de cálculo, acceso a internet, y bibliotecas, bibliotecas virtuales facilitadas por docentes investigadores de otras universidades nacionales e internacionales, para la revisión de la literatura y la redacción del mismo, y de esta manera contribuir en desarrollo de investigaciones científicas de las diferentes universidades del país, siendo muy escasas.

1.5. ÁREA DE CONOCIMIENTO

De acuerdo con el reglamento del Sistema de Estudios de Posgrado y Educación Continua de La Universidad Nacional Autónoma de Nicaragua, $UNAN - MANAGUA$ (2011), el área de conocimiento en el que versa el presente trabajo es el **Ciencias Aplicadas**, centrada en la **línea de Investigación Modelación y Programación Matemática**, puesto que, obedece a la naturaleza del estudio teórico de las propiedades de los modelos, métodos, técnicas matemáticas, algoritmización y herramientas generales de Optimización Combinatoria.

1.6. TEMA Y OBJETIVOS DE INVESTIGACIÓN

1.6.1. Tema de Investigación

ALGORITMOS HEURÍSTICOS DE COINCIDENCIA PARA LA ESTIMACIÓN DE MOVIMIENTO EN COMPRESIÓN DE IMÁGENES.

1.6.2. Objetivos

1.6.2.1. Objetivo General

Diseñar e Implementar Algoritmos Heurísticos de Coincidencia para la Estimación de Movimiento en compresión de imágenes.

1.6.2.2. Objetivos Específicos

1. Revisar el estado del arte de los diferentes Algoritmos de Coincidencia para la Estimación de Movimiento en compresión de imágenes.

2. Analizar la teoría matemática necesaria para el diseño e implementación de los Algoritmos de Coincidencia para la Estimación de Movimiento en compresión de imágenes.

3. Diseñar Algoritmos basados en el dominio de frecuencia (DCT, Wavelets) y Heurísticos (FS, TTS, 4SS, DS, MHTSS, MHCS) de Coincidencia para la Estimación de Movimiento en compresión de imágenes.

4. Implementar los Algoritmos del dominio de frecuencia (DCT, Wavelets) y Heurísticos (FS, TTS, 4SS, DS, MHTSS, MHCS) de Coincidencia para la Estimación de Movimiento en compresión de imágenes, escritos en el lenguaje de Programación Matemático $MATLAB$.

5. Comparar los resultados obtenidos mediante la experimentación computacional de los Algoritmos de Coincidencia para la Estimación de Movimiento en compresión de imágenes.

1.7. Organización de la Tesis

La presente investigación está estructurada en cinco capítulos, el Capítulo **1** corresponde a la introducción en las que se aborda la idea general de las Investigación de Operaciones, su principales autores y sus aportes, sus aplicaciones a la I. O, al campo de Optimización, Optimización Combinatoria, sus grandes aportes a las ciencias aplicadas, investigación de la matemática aplicada, visión artificial,teoría de la complejidad computacional y la metodología científica aplicada en el estudio.

Además, se explica el **problema estimación de movimiento de compresión de imágenes**, así como los subproblemas que forman parte de nuestro principal interés. Se cita las investigaciones sobresalientes haciendo énfasis en sus aportes, aplicabilidad, sencillez, originalidad y su garantía de desempeño.

Finalmente, se desarrolla aspectos esenciales tales como: la motivación, el estado del arte, planteamiento del problema, la justificación de la investigación, el área y línea de investigación, el tema, objetivos de investigación, y la organización que describe el cuerpo de la tesis doctoral.

El Capítulo **2** se expone los conceptos esenciales del procesamiento de imágenes y vídeo. Primero se realizará una breve reseña histórica para comprender la génesis del procesamiento de imágenes y vídeo, se desarrolla los conceptos básicos tener en cuenta a lo largo esta investigación como: compresión de imágenes, las medidas de comprensión, los métodos de determinación de flujo ópticos, los aspectos generales de la transformada de Fourier, los principales aspectos de la transformada discreta del Coseno unidimensional y bidimensional, finalizando con la comprensión de vídeo digital.

En el Capítulo **3** se desarrolla los diferentes métodos y algoritmos de Estimación de Movimiento basados en coincidencia por bloques. Además, se plantean los principios fundamentales para su implementación. Se inicia con una revisión de los modelos de la estimación de movimiento en **2-D**. Se diseñan los algoritmos de dominio temporal: El algortimo búsqueda exhaustiva, algoritmo de búsqueda en tres pasos, algoritmo de búsqueda en cuatros pasos, algoritmo de búsqueda en diamante, algoritmo de búsqueda logarítmica, algoritmo de búsqueda en cruz, algoritmos de búsqueda de los vecinos más cercano, mejora del heurístico de búsqueda en tres pasos simple y eficiente, heurístico mejorado en la búsqueda de patrón en tres pasos.

En el Capítulo **4** se muestra la modalidad de investigación, la herramienta computacional empleada y el cronograma de investigación. Además, se profundiza en la metodología desarrollada en la implementación de los diferentes algoritmos, se comparan los resultados de simulación consigo mismos y con alguna literatura existente.

En el Capítulo **5** se plantea la implementación y los resultados computacionales, de los tres conjuntos de prueba. Se inicia mostrando la secuencia de imágenes de pruebas; se implementa los algoritmos basados en el dominio de frecuencia, y planteando cada uno de los resultados obtenidos al variar los diferentes parámetros(tamaño del macrobloques, pasos, etc.), haciendo énfasis en los aspectos fundamentales tales como: el potencial de compresión de la transformada discreta del coseno, la distribución de la información y los diferentes efectos encontrados, la calidad visual, los casos bien y mal condicionado, la compresión alcanzada y transmisión progresiva de la DCT.

También, se implementa y se desarrolla los diferentes aspectos obtenidos al variar los diferentes parámetros del algoritmo y procedimientos centrado en dominio frecuencial de los coeficientes de la transformada Wavelets.

De la misma manera, se implementa y se analiza los resultados obtenidos al variar sus parámetros y comparar los resultados de simulación consigo mismos de los diferentes algoritmos de coincidencia y procedimientos centrado en dominio temporal (FS, TTS, 4SS, DS, MHTSS, MHCS).

En el penúltimo apartado se presenta la **bibliografía**, utlizadas.

En el Capítulo **7** se presentan como anexos las funciones, procedimientos, en la que se muestra los módulos esenciales de algoritmos codificados en el lenguaje de programación matemático MATLAB.

Capítulo 2

PROCESAMIENTO DE IMAGÉNES Y VÍDEO

En este capítulo se expone las ideas relevantes del procesamiento de imágenes y vídeo. Se inicia con una breve reseña histórica para comprender la génesis del procesamiento de imágenes, se mencionan algunos autores principales que brindaron aportes esenciales al desarrollo, se abordaron los conceptos básicos que se tendrán en cuenta a lo largo esta investigación tales como: la compresión y la transmisión de imágenes digitales, los tipos de redundancia existentes, las diferentes técnicas de compresión, la codificación de transformación por bloques, los procesos de transformación y cuantización de las imágenes, la representación matemática y sus propiedades, las diferentes medidas o métricas existentes para el proceso de compresión de imágenes digitales y vídeos. Se describen los métodos de determinación de flujo óptico, se hace mención de la transforma de Walsh-Hadamard, y los aspectos generales de la transformada de Fourier y principales aspectos de la transformada discreta del Coseno, la Wavelets, así como los detalles al implementar este tipo de métodos.

2.1. Reseña Histórica del Procesamiento de Imágenes

La compresión de imágenes surge debido al almacenamiento gigantesco y la transmisión de imágenes(en escala gris o color), y sobre todo en imágenes animadas. Desde hace algunas décadas, la demanda y la calidad eras mínimas, debido a la escasez de la tecnología y diferentes equipos electrónicos, es una de las razones para la búsqueda de nuevas técnicas, métodos, modelos matemáticos para aprovechar el limitado recurso de los canales de transmisión y de los medios masivos de almacenamiento.

Una imagen tiene una gran cantidad de información y detalles, que una simple foto tomada por una cámara fotográfica, por un teléfono, cámara de un ordenador o un satélite; no sólo proporciona una

vista de la superficie, sino más bien, si es analizada con detenimiento, puede proporcionar información sobre el objeto o características relevantes que deseamos estudiar. De la misma forma, que un film o una imagen animada, no es otra que una cadena o secuencia de imágenes, que nos proporciona una información muy superior, cuyo análisis es mucho más complejo que una imagen estática.

El área o campo de visión artificial (máquinas autónomas) que se encarga del análisis e interpretación de imágenes, ha realizado varios intentos a través de la historia, por desarrollar dispositivos electrónicos e integrados de alta velocidad y procesadores de instrucciones avanzadas; ha abierto las puertas a la interpretación de imágenes por máquinas autónomas, pero queda mucho camino que recorrer, puesto que la sociedad nos exige mayores demandas de los mismos.

La mayor parte del trabajo en el campo de la investigación **visión artificial** se refiere a la **creación de nuevos algoritmos y su implementación** a través de procesadores sofisticados. Sin embargo, las altas velocidades involucradas y el volumen de datos en estudio obligan al uso de novedosas arquitecturas de procesamiento en paralelo, a un costo mucho mayor.

Con los avances en los dispositivos electrónicos programables e integrados en tarjetas madre, que años anteriores eran imposibles de conseguir debido al avance de la tecnología, hoy en día es una realidad a implementar sistemas en un circuito integrado “system on a chip”, de esta manera mejorar el rendimiento y velocidad de los mismos. Uno de los grandes problemas, se presenta en la creación de modelos, métodos o técnicas para diseñar, transferir el diseño al circuito integrado y realizar las pruebas al diseño. Por lo cual, los fabricantes han venido proporcionando una serie de herramientas y adoptando el lenguaje de descripción de hardware VHDL para su desarrollo y simulación, son tratados a profundidad en [51] por (Shahdad, 1985) y [50] por (Scarpino, 1997).

En el campo de la Visión Artificial, la estimación de movimientos de objetos en una secuencia de imágenes y vídeo es un área joven, en la que se desarrolla muchos trabajos que tienen objetivos y métodos distintos. Uno de esos métodos, es a través de la obtención del “flujo óptico denso” el cual busca la estimación del movimiento de los píxeles que componen las imágenes y que tienen diferentes aplicaciones, en [28] trata caracterizar el movimiento de un cuerpo de forma automática a partir del análisis de un vídeo digital propuesto por (Hueso, 2013) y en [40] se determina el flujo óptico de una secuencia de imágenes a una velocidad superior o igual a tiempo real. Los primeros trabajos se originaron en el año 1920, y su finalidad principal era la mejora de imágenes transmitidas para su uso en periódicos. Se ha recorrido mucho desde entonces, pero sigue siendo una de las aplicaciones con mayor campo de acción, desde la reconstrucción de imágenes y el coloreo de películas antiguas al desarrollo de nuevas técnicas de fotografía.

En el año 1884 el alemán Paul Nipkow, patenta un sistema básico de barrido mecánico de imágenes,

en la que divide la imagen en líneas continuas, siendo la base para la transmisión de imágenes analógicas de televisión; a diferencia de G. Carey, que las dividía en píxeles, utilizada para el procesamiento digital de imágenes.

En el año 1900, en París se realiza el primer Congreso Internacional de Electricidad, donde por primera vez se menciona el término "televisión" por Constantin Perski para definir la transmisión de imágenes animadas mediante un sistema de comunicación. En 1970, W. Boyle y G. Amelio de los laboratorios Bell inventaron el dispositivo de carga acoplada: Charge Coupled Device, utilizado como memoria semiconductora, y posteriormente (Radio Corporation of America, 1973) se utilizó como sensor de vídeo por excelencia. En la actualidad, la mayor parte de las cámaras de vídeo profesional y casero están basadas en dispositivos.

2.1.1. La transmisión de imágenes

Los primeros indicios realizados con éxito de codificar y enviar imágenes fijas a distancia lo realizó el físico italiano Abbe Caselli en 1862, el cual transmitió la primera imagen por medio de ondas electromagnéticas, en 1884 el ingeniero alemán Paul Nipknow patenta su disco de exploración lumínica, en 1897 el físico Karl Ferdinand Braun construye el primer tubo catódico (el cual se utiliza para, las cámaras y receptores de televisión modernos), en 1907 el científico alemán Arthur Korn (logra el envío de imágenes de gran complejidad desde Berlin hacía Paris), en 1923 el escocés John Logie Baird desarrolla y perfecciona el disco de Nipknow. Para transmitir una secuencia de imágenes, se requiere una serie de señales para enviar la información y el sincronismo de los sistemas involucrados.

Años después, Farnsworth tras varios ensayos y errores, el 7 de septiembre de 1927, lográ con éxito realizar la transmisión de una señal, que consistía en una simple línea recta en movimiento, un año más tarde, en 1928, había desarrollado el sistema lo suficiente como para hacer una manifestación pública. Se ha de tener en cuenta:

- La señal de información de nivel de brillo (luminancia) y color (crominancia).
- La señal de sincronización horizontal, que indica el comienzo de barrido de una nueva línea.
- La señal de sincronización vertical que indica el comienzo de barrido de un nuevo cuadro.

En la señal analógica de vídeo de estas tres señales se combinan en una sola, dando como resultado un "vídeo compuesto". El origen de los sistemas digitales hace necesario un método de transmisión de acuerdo a la tecnología actual, hay que pensar en placas **integradas o algoritmos más eficientes**.

Las imágenes a color en los sistemas de vídeo se han codificado tradicionalmente, bajo ciertos parámetros de luminancia y crominancia, de manera que los primeros sistemas de televisión a color fueran compatibles con los sistemas de televisión en blanco y negro. La luminancia se refiere a la parte de la imagen asociada con la luminosidad, y la crominancia se refiere a la asociada con el color. La señal de crominancia posee dos componentes, una lleva información de la diferencia entre el color rojo y la luminancia, y la otra lleva información de la diferencia entre el color azul y la luminancia. Con estas señales es posible construir las imágenes a color.

Los métodos y técnicas empleadas en la transmisión analógica y digital se basan, en dos de las tres señales mencionadas la cuales son: las señales de luminancia y crominancia que se utilizan por razones históricas de compatibilidad y por su simplicidad en sus aplicaciones. No obstante, en la actualidad existen aplicaciones basadas en las señales RGB, principalmente en el campo de la informática, ciencias de la computación y áreas afines.

2.2. Compresión de Imágenes

Un sistema de compresión de Imágenes consiste en dos bloques estructurales distintos: un codificador y un decodificador. La imagen original entra en el codificador, el cual crea una serie de símbolos a partir de los datos de entrada. Después de la transmisión a través del canal, la representación codificada entra en el decodificador, donde se genera una reconstrucción de la imagen codificada. Si el sistema no está libre de error, se tendrá un cierto nivel de distorsión en la imagen decodificada. Tanto el codificador y el decodificador están formados por bloques independientes.

La compresión de una imagen se define como el proceso de reducir la información redundante e irrelevante con la menor pérdida posible, para permitir su almacenamiento o transmisión de forma eficiente. Según (González y Wood, 1996) en [24] lo define como el proceso de reducir el volumen de la información a representar. En general, las técnicas de compresión de imágenes se pueden agrupar en dos grandes clases: Sin pérdida de información y con pérdida de información. La aplicación se hará de acuerdo con el tipo de imagen tratada, en las imágenes médicas no se permite la pérdida de información en el proceso de compresión, en cambio, en imágenes en vídeo conferencias es posible permitir cierto grado de error, aunque manteniendo la calidad visual de la imagen, con la finalidad de optimizar la compresión de imágenes.

2.2.1. Representación de Imágenes Digilates

Una imagen está definida como una función bidimensional de intensidad de luz $f(x, y)$, donde x e y representa las coordenadas espaciales y el valor de f en un punto cualquiera (x, y) es proporcional al brillo o nivel de gris de la imagen en ese punto.

Una imagen digital es una imagen $f(x, y)$ que se ha discretizado, tanto en las coordenadas espaciales como en el brillo, y puede considerarse como una matriz cuyos índices de fila y de columna identifican un punto de la imagen y el valor del correspondiente al elemento de la matriz indica el nivel del píxel en ese punto. Es decir, que el valor correspondiente a una fila y de la columna define una pequeña área de la imagen denominado píxel (por las siglas *Picture Element*). Una imagen es una fotografía de una escena en un instante de tiempo, en cambio un vídeo representa una escena sobre un periodo de tiempo.

Una imagen de color es una matriz de tamaño $N \times M \times 3$, formada por píxeles de color, siendo una terna de valores que corresponden a las componentes rojo, verde y azul en una localización espacial específica como se muestra en la siguiente figura:

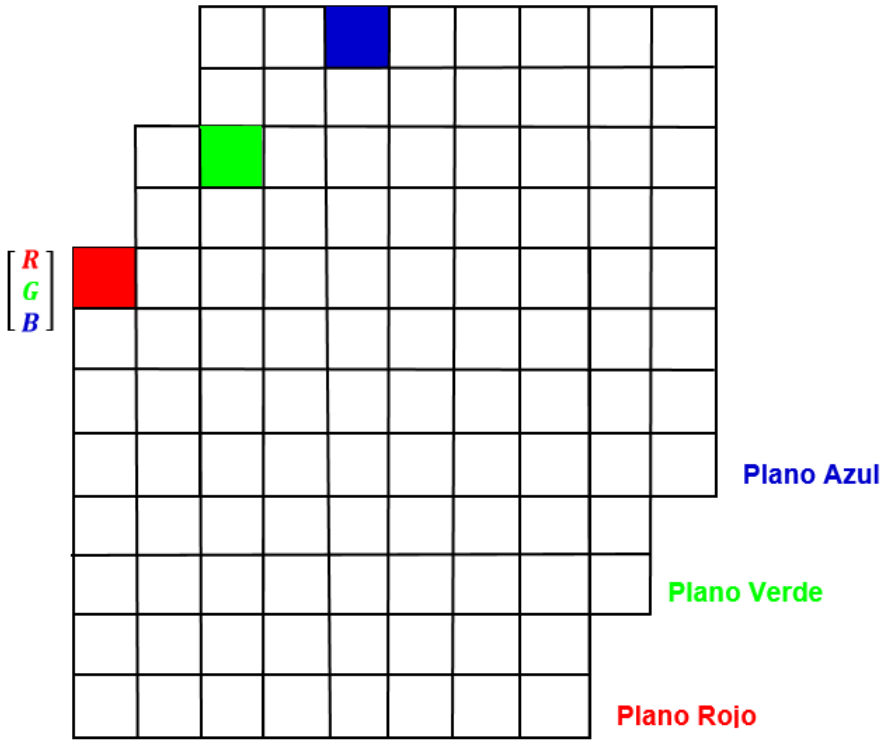


Figura 2.1: Componentes de una imagen a color RGB

2.2.1.1. Representación de Imágenes Digilates

El espacio de colores de una imagen en las escalas RGB (Red, Green, Blue) que es utilizado en cada sistema computacional, tanto en televisión o cualquier equipo digital y vídeo es tratado por (Asmare, 2009) a profundidad en [2] quien introduce un nuevo método para mejorar las imágenes a color, basados en un modelo de descomposición empírico, además proponen un Algoritmo Genético que permite que el método presentado establezca automáticamente los parámetros requeridos. La siguiente figura nos muestra un cubo formado tradicionalmente por el espacio de coordenadas $-$ RGB. Puesto que, cada componente tiene la misma importancia para obtener la representación de color final en la imagen, los sistemas RGB usualmente representan a cada componente con la misma precisión. Por ejemplo, si cada componente utiliza 8 bits, en total se tienen $3 \times 8 = 24$ bits para representar un píxel.

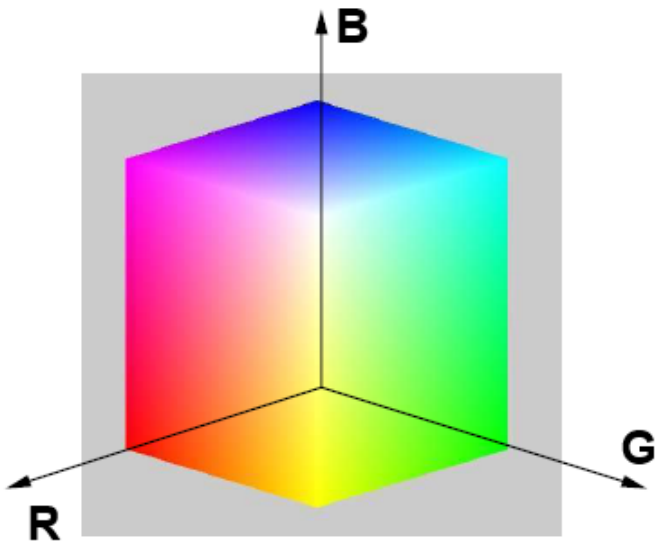


Figura 2.2: Espacio de color RGB tridimensional

En la siguiente figura nos muestra una imagen propia en las que se toman cada uno de sus canales R, G, y B:

Imagen RGB sin descomponer

R

G

B

Cuando una imagen se carga sobre la pantalla de un dispositivo electrónico, los elementos son mapeados a un único píxel de la pantalla. La combinación de los valores en las componentes R (Rojo), G (Verde), B (Azul) dan como resultado al color final de píxel en la imagen.

Representar una imagen de $n \times n$ píxeles mediante una matriz requiere:

- $O(n^2)$ bits si la imagen es binaria
- $O(n^2 \log_2 (L))$ bits si la imagen es en niveles de grises, con L niveles.
- $O(3n^2 \log_2 (L))$ bits si la imagen es a color (RGB) con niveles.

2.2.1.2. Propiedades

La finalidad principal de la compresión de imagen es, generar la mejor calidad de la imagen con una tasa de compresión dada en $bits$. No obstante, se pueden señalar otras propiedades importantes obtenidas de los esquemas de compresión:

La Escalabilidad: Consiste en la reducción de calidad lograda por la manipulación de los datos del archivo (sin descomprimir y comprimir nuevamente). Siendo de utilidad para previsualizar las imágenes, mientras se descargan en algún navegador web o para proporcionar accesos de calidad variable. Existen varios tipos, los cuales se menciona algunos de ellos a continuación:

- **La calidad progresiva**: Consiste en que los tramas de bits refinan sucesivamente la imagen reconstruida.
- **La resolución progresiva**: acá, primero se codifica una imagen a baja tasas de resolución; luego codifica la diferencia a resoluciones mayores.
- **La componente progresiva**: en este tipo de escalabilidad primero codifica el gris y luego el color.

Codificación por regiones de interés: consiste en que ciertas partes de la imagen se codifican con una calidad mayor que otras. Esto se puede combinar con la escalabilidad, es decir, codificar primero los puntos o regiones de interés, y por último las demás que tienen poca relevancia.

Recursos de procesamiento: consiste en que los algoritmos de compresión requieren diferentes cantidades de recursos para codificar y descodificar, la idea detrás de toda esta información es agilizar el proceso. La calidad de un método de compresión se mide a menudo por la Relación señal a ruido de pico $PSNR$, que mide la cantidad de ruido introducida en la compresión con pérdida de la imagen. Sin embargo, la opinión subjetiva de quien observa la imagen se toma en cuenta en este proceso, siendo también una de las medidas más importante.

2.2.2. Tipos de Redundacia en Compresión de Imágenes

En [48] se realiza un tratamiento de la compresión de imágenes digitales, descritos en la bibliografía de (Ruíz y González, 2000) y en [24] plantea tres tipos diferentes de redundancia empleados por (González y Woods, 1996), que son abordados de diferentes formas, según el autor.

La comprensión de datos se define como: el proceso de reducir la cantidad de datos necesarios para representar eficazmente una información, es decir, la eliminación de datos redundantes.

2.2.2.1. Redundancia de datos

Sean n_P y n_Q el número de de bits necesario para almacenar dos representaciones, P y Q, distintas de la misma imagen. La redundancia relativa de P (respecto de Q) es:

$$R = 1 - \frac{1}{C}$$

donde $C = \frac{n_P}{n_Q}$ es el radio de compresión.

Para una imagen existen tres formas de reducir la redundancia de los datos, las cuales se detallan a continuación:

2.2.2.2. Redundancia en el Código

La redundancia en el código de una imagen o un sistema de símbolos representa el cuerpo de la información mediante un conjunto de símbolos. A veces, la longitud de las palabras usadas en el código es mayor de lo necesario y frecuentemente, se utiliza el **Código de Huffman**. El algoritmo de codificación Huffman, utilizada en el estándar JPEG, diseñado por (David Huffman, 1951), trata de acercarse al límite impuesto por la entropía del sistema, y una revisión a profundidad se halla en [29]. Esta técnica de compresión utiliza cálculos estadísticos para lograr eliminar este tipo de redundancia y reducir la ocupación original de los datos.

2.2.2.3. Redundancia entre píxeles

Debido a que las imágenes presentan correlación espacial entre un píxel y sus vecinos. Estas se deben a la presencia de estructuras semejantes o similares en las imágenes, ya que no son completamente aleatorias. Esta es la razón de utilizar el valor de un píxel para predecir el de sus vecinos.

La compresión sin pérdidas: consiste en aquellas técnicas o métodos en las que la imagen original puede recuperarse de su versión codificada sin ningún tipo de alteración. Se trata de compresiones reversibles y en consecuencia no incluyen una fase de cuantificación, pues este proceso es intrínsecamente con pérdidas. Sin embargo, una codificación directa de las imágenes con algún tipo de codificador de entropía no supone una tasa de compresión relevante. Por ello es necesario un proceso previo de correlación.

Dentro de las técnicas de codificación sin pérdida en imágenes en escala de grises, se tienen:

- La técnica de compresión Lempel $-$ Ziv, implementa algoritmos basados en sustituciones para lograr la eliminación de esta redundancia. La estrategia es efectiva debido a la frecuencia de símbolos, repetición de caracteres y patrones de alto uso.
- La técnica de compresión RLE, empleada en los formatos de imagen (BMP, PCX, TIFF), se centra en la repetición de elementos consecutivos. La idea de utilizarse en imágenes es la siguiente: una imagen en tonos de gris contiene una serie de números que se repiten, en particular tripletas de píxeles (en sus componentes RGB), los cuales son el mismo color. Si tenemos regiones en

donde se repiten estos valores R, G y B, bien podríamos pensar en sustituirlos por el byte que hemos leído y un contador que nos indique cuantas veces se repite el mismo.

- Planos de bits, es un método de descomposición alternativo (que reduce el efecto de pequeñas variaciones de niveles de gris) consiste en representar primero la imagen mediante un código de Gray de L bits. En general, todos los archivos de imágenes pertenecen a una de las dos categorías básicas: mapa de bits (bitmap) o vectores. Los mapas de bits se usan con frecuencia en fotografías o imágenes de fotos reales, sin embargo los vectores son utilizados en composición y diseño gráfico.

Codificación sin pérdida en imágenes binarias se da mediante la representación por bloques, árbol cuaternario, por borde de la imagen y conjuntos derivados.

2.2.2.4. Redundacia Psico−Visual

La codificación con pérdida se centra en la idea de comprometer la precisión de la imagen descomprimida con el fin de lograr una mayor compresión, y se hace necesario indicadores que nos permitan medir el error que se comete después de comprimir y descomprimir con respecto a la imagen original. En este tipo de compresión, la imagen decodificada es distinta a la original, pero las pérdidas sufridas no son detectables por el ojo humano; tomándose en consideración las características de la visión humana en el momento de decidir que alteraciones puede sufrir la imagen durante el proceso de compresión.

A continuación, se muestra una imagen digitalizada sin pérdida visual (imagen de la izquierda) y con pérdida (imagen de derecha) utilizando 8 niveles de cuantización.

Figura 2.3: Imagen médica cuatizada

Dentro de la codificación con pérdida en imágenes en escala de grises, se tiene las siguientes:

- La Transformada de Fourier: consiste en hacer corresponder la imagen con un conjunto de coeficientes de la transformada, a los que se le aplica un proceso de cuantificación, eligiendo un número significativo de los coeficientes, siendo los valores más pequeños poco significativos, los cuales se pueden eliminar mediante el proceso de cuantización, produciéndose la pérdida de información y generándose una distorsión mínima en las imágenes en estudio. A través, de este proceso se obtiene un número reducido de datos de la imagen, a los cuales usualmente se les aplica una técnica de codificación sin pérdida para mejorar los resultados sin perder de vista la calidad de la imagen.

- La Transformada WHT, conocida como transformación de Walsh-Hadamard es un ejemplo de una clase generalizada de transformadas de Fourier, es una de las más conocidas de las transformaciones ortogonales no-sinusoidales. Ésta ha ganado aceptación en varias aplicaciones de procesamiento de señales digitales, puesto que se puede calcular esencialmente utilizando sólo adiciones y sustracciones. Siendo su implementación por hardware más simples.

  La NASA empleó Transformada WHT como base principal para comprimir fotografías de las sondas interplanetarias durante los años sesenta y principios de los setenta. Puesto que, esta transformada es computacionalmente la más simple de implementar, que la transformada de Fourier, ya que no requiere operaciones de multiplicación o división (todos los factores son más o menos uno). Las operaciones de multiplicar y dividir fueron extremadamente complejas de implementar en el tiempo de los pequeños ordenadores utilizados a bordo de las naves espaciales, por lo que evitarlos fue beneficioso tanto en términos de tiempo de cálculo y el consumo de energía.

- La transformada discreta del coseno, se desarrollará en las secciones posteriores de una forma exhaustiva.

2.2.2.5. Codificación por transformación en bloques

Las técnicas de compresión con pérdida que estudiaremos son las técnicas de codificación por transformación en bloques. Es decir, la imagen original se divide en bloques de tamaño $n \times n$ y en cada uno de ellos se realiza una codificación por transformación. En esta transformación, se utiliza una transformada lineal, reversible, para hacer corresponder la imagen con un conjunto de coeficientes de la transformada de Fourier y del Coseno, a los que se les aplica un proceso de cuantificación, tomando un número significativo de los coeficientes con valores pequeños que son poco significativos, los cuales se pueden eliminar mediante el proceso cuantización, produciéndose la pérdida de información, aunque no suponga una distorsión apreciable de la imagen. De esta forma se obtiene un número reducido de datos de la imagen, a los cuales usualmente se les aplica una técnica de codificación sin pérdida para mejorar los

resultados.

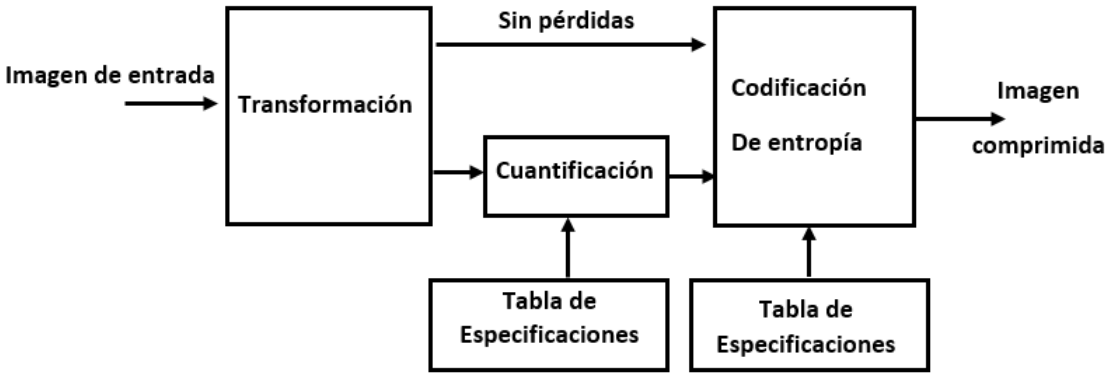


Figura 2.4: Esquema de codificación sin pérdida

A continuación, se muestra el esquema de codificación híbrido utilizado frecuentemente en la literatura existente:

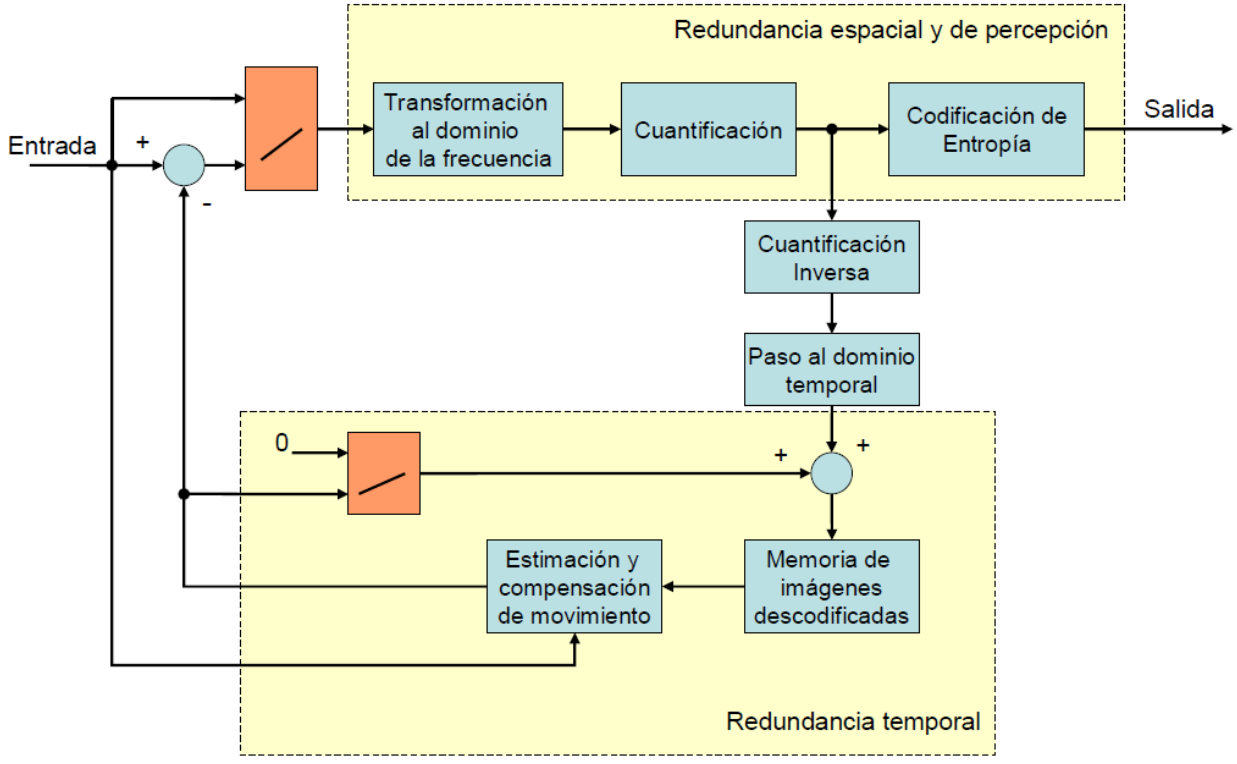


Figura 2.5: Esquema de codificación híbrido

La mayoría de las imágenes naturales, se puede trabajar con un número significativo de coeficientes, en el dominio transformado, ya que, tienen pequeñas magnitudes, y se pueden cuantificar de forma precisa con pocos coeficientes sin que ello suponga una distorsión apreciable en la imagen original. A continuación, se muestra la imagen original, la imagen actual y dos imágenes reconstruidas con y sin

estimación de movimientos:

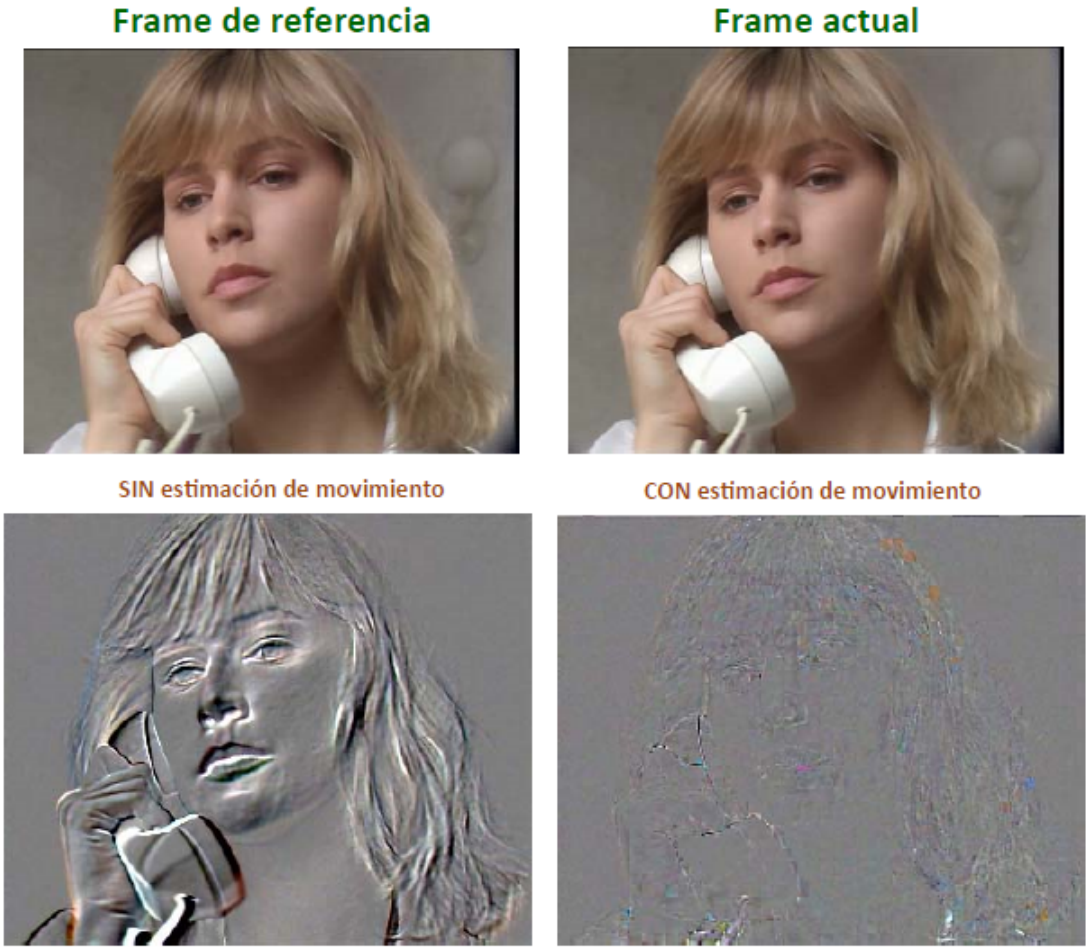


Figura 2.6: Imagen reconstruidas con y sin estimación de Movimientos

Las técnicas de compresión de imágenes o señales digitales, constan de tres etapas las que se definen a continuación:

- Transformación de la imagen: consiste en una de-correlación de la señal, es decir, una reducción de su rango dinámico que elimine información redundante. Los coeficientes transformados deben ser estadísticamente independientes y la energía de la imagen transformada debe compactarse en un número mínimo de ellos.

- Cuantificación de la imagen: Consiste en reducir la precisión de los coeficientes transformados, sin sobrepasar los límites de calidad señalados. Esta etapa implica siempre algún tipo de pérdida, con lo que las técnicas sin pérdidas carecerán de ella.

- Codificación basada en entropía: Consiste en aumentar la compresión, sin aumentar las pérdidas de los coeficientes transformados de forma más compacta.

2.2.3. Medidas de Compresión de Imágenes

Los diferentes modelos de cámara producen imágenes de diferente calidad. Puede haber diferencias en la luminosidad de la imagen, la nitidez o en la calidad del color. Estas diferencias que se observan al comparar fotos de varias cámaras hacen que se proponga un conjunto de métricas de calidad IQM como características que nos ayudan a diferenciar la fuente de las imágenes. Existen diferentes categorías para las $IQM's$:

- Medidas basadas en las diferencia de los píxeles.
- Medidas basadas en la correlación.
- Medidas basadas en la distancia espectral.

Para emplear este conjunto de métricas es necesario contar con la imagen original, y una imagen filtrada para poder realizar los diferentes cálculos necesarios. Además, para el análisis de la compresión y medición del desempeño de los algoritmos se utilizan las funciones de costos o métricas, estas medidas se calculan expresando una relación entre las longitudes de las secuencias-código, entre la imágenes originales y la obtenida después del proceso de compresión y predicción, dentro de literatura utilizada para la compresión con frecuencia se aborda en [48] por $(Ruiz, 2000)$, en [43] por $(Pourreza, 2000)$ y también es tratada en [30] por $(Jagalingam, 2000)$.

A continuación, se muestra las métricas para la estimación de movimientos basados en los algoritmos de **block-matching**, en la que se determina el vector de movimiento (mv_x, mv_y) óptimo a partir de una *función de costos*. Esta función nos indica el grado de coincidencia entre la señal o frame original I_C y la señal reconstruida o el frame residual I_R, los parámetros m, n representan el largo y ancho de la imagen. Las diferencias entre las funciones de costos varían en términos de la complejidad y eficiencia de las métricas a utilizar, entre ellas están:

2.2.3.1. Distancia de Czekonowsky (CD)

La distancia Czekonowsky es una métrica que se utiliza para comparar vectores con componentes no negativas, en imágenes a color y sus ecuación se muestra a continuación:

$$CD\left(d_x, d_y\right) = \frac{1}{M \cdot N} \sum_{i=1}^{N} \sum_{j=1}^{M} \left(1 - \frac{2 \sum_{k=1}^{3} \text{mín}\left(I_C\left(i,j\right), I_R\left(i + d_x,\ \ j + d_y\right)\right)}{\sum_{k=1}^{3} \left(I_C\left(i,j\right) + I_R\left(i + d_x,\ \ j + d_y\right)\right)} \right)$$

2.2.3.2. La Métrica de Minkowsky (MM)

Las métricas de $Minkowski$ para $\alpha = 1$ y $\alpha = 2$ se basan en la siguiente fórmula:

$$MM\left(d_x, d_y\right) = \frac{1}{K}\sum_{k=1}^{N}\left\{\frac{1}{N^2}\sum_{i,j=1}^{M}\left[I_C\left(i,j\right) - I_R\left(i+d_x,\ \ j+d_y\right)\right]^{\alpha}\right\}^{\frac{1}{\alpha}}$$

que determina la norma L_α de disimilitud entre un par de imágenes. La variable K se refiere a cada uno de los canales de la imagen N^2 define el número total de píxeles. Cabe señalar que si $\alpha = 1$ se obtiene la medida del *error medio absoluto* (MAE) y $\alpha = 2$ coincide *con el error cuadrático medio* (MSE), que se analizará más adelante. En ambos casos, para valores altos del MAE o MSE se corresponden con imágenes de calidad no deseadas ya que su calidad de recuperación es muy baja.

2.2.3.3. Función de Correlación Cruzada (CCF)

En el procesado digital de señales se necesita cuantificar el grado de interdependencia entre dos procesos o la similitud entre dos señales, imágenes, secuencias de imágenes o vídeo, tal medida es arrojada por la función de correlación cruzada, que se muestra a continuación:

$$\begin{aligned} CCF\left(d_x, d_y\right) &= \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} I_C\left(i,j\right) I_R\left(i+d_x,\ \ j+d_y\right)}{\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{M} I_C\left(i,j\right)}\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{M} I_R^2\left(i+d_x,\ \ j+d_y\right)}} \\ \left(mv_x, mv_y\right) &= \text{máx}_{(d_x,d_y)}\left\{CCF\left(d_x, d_y\right)\right\} \end{aligned}$$

Debido a que el orden de complejidad computacional es elevado, se hace impracticable en las aplicaciones relacionadas con la codificación de vídeo en tiempo real, pero es una buena medida desde el punto teórico matemático y computacional.

2.2.3.4. Correlación Cruzada Normalizada(CCN)

La correlación cruzada normalizada nos permite cuantificar la similitud o coincidencias entre imágenes, las imágenes son similares si su valor tiende a ser la unidad, de tal manera que a mayores diferencias respecto del valor uno indica menor calidad entre la imagen original y la reconstruida. Esta medida está definida como:

$$
\begin{aligned}
CCN\left(d_x, d_y\right) &= \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}\left(I_C\left(i,j\right)\times I_R\left(i+d_x,\ \ j+d_y\right)\right)}{\sum_{i=1}^{N}\sum_{j=1}^{M}\left(I_C\left(i,\ \ j\right)\right)^2} \\
\left(mv_x, mv_y\right) &= \text{máx}_{(d_x,d_y)}\left\{CCN\left(d_x, d_y\right)\right\}
\end{aligned}
$$

2.2.3.5. Contenido Estructural (SN)

La medida de contenido estructural está basada en el grado de correlación y mide la similitud entre las imágenes. Las imágenes son similares si el valor obtenido por SC tiende a ser 1, mientras que el valor mayor se distancie de la unidad, la imagen reconstruida será de menor calidad, tanto en exceso como por defecto. Esta métrica está definida por la ecuación:

$$
\begin{aligned}
SC\left(d_x, d_y\right) &= \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}\left(I_R\left(i+d_x,\ \ j+d_y\right)\right)^2}{\sum_{i=1}^{N}\sum_{j=1}^{M}\left(I_C\left(i,\ \ j\right)\right)^2} \\
\left(mv_x, mv_y\right) &= \text{máx}_{(d_x,d_y)}\left\{SC\left(d_x, d_y\right)\right\}
\end{aligned}
$$

El parámetro pico de la razón pico señal a ruido $(PSNR)$ que es una medida relativa de la calidad visual de una imagen. Se basa en MSE el cual mide el error cuadrático medio.

2.2.3.6. Error Cuadrático Medio (MSE)

El error cuadrático medio es una de las métricas que se utiliza con frecuencia en los distintos campos, y en particular en nuestro campo de estudio, a pesar de tener una alta complejidad computacional, pero se usa por estar muy próximo a las características de la percepción y visión del ojo humano.

$$
\begin{aligned}
MSE\left(d_x, d_y\right) &= \frac{1}{M\cdot N}\sum_{i=1}^{N}\sum_{j=1}^{M}\left[I_C\left(i,j\right) - I_R\left(i+d_x,\ \ j+d_y\right)\right]^2 \\
\left(mv_x, mv_y\right) &= \text{mín}_{(d_x,d_y)}\left\{MSE\left(d_x, d_y\right)\right\}
\end{aligned}
$$

2.2.3.7. Relación Señal a Ruido de Pico $(PSNR)$

Es una medida cuya unidad son los decibelios, que es inversamente proporcional al MSE. Cuya ecuación se muestra a continuación

$$PSNR = 10 \cdot \log_{10}[\frac{(2^N - 1)}{\sqrt{MSE}}]$$

donde, n es el número de bits de la imagen. Un valor bajo del MSE, significa menos error en la señal o frame residual con respecto al frame de referencia; traduciéndose en un valor grande de $PSNR$ en decibeles. Es decir, un valor grande de $PSNR$, es bueno puesto que significa que la razón entre el frame de referencia y el residual es grande.

2.2.3.8. Error Medio Absoluto (MAE)

El error medio Absoluto o sus siglas en inglés (MAD), es muy popular por su sencilla implementación que las otras métricas.

$$\begin{aligned} MAE\left(d_x, d_y\right) &= \frac{1}{M \cdot N} \sum_{i=1}^{N} \sum_{j=1}^{M} \left|I_C\left(i, j\right) - I_R\left(i + d_x,\ \ j + d_y\right)\right| \\ \left(mv_x, mv_y\right) &= \text{mín}_{(d_x, d_y)} \left\{MAE\left(d_x, d_y\right)\right\} \end{aligned}$$

2.2.3.9. Suma de diferencias absolutas (SAD)

La Suma de diferencias absolutas es la implementación práctica del MAE, y consiste en eliminar la división MN; es decir, esta medida calcula la diferencia entre los valores de los píxeles de la imagen original o de referencia y la imagen reconstruida. Mientras más se aproxime a cero, las imágenes comparadas tienden a ser iguales y por tanto si se obtienen pequeños valores en este nivel, no dará una

mayor calidad de la imagen. Esta medida se define según la siguiente ecuación:

$$\begin{aligned} SAD\left(d_x, d_y\right) &= \sum_{i=1}^{N}\sum_{j=1}^{M}\left|I_C\left(i,j\right) - I_R\left(i+d_x,\ \ j+d_y\right)\right| \\ \left(mv_x, mv_y\right) &= \text{mín}_{(d_x,d_y)}\left\{SAD\left(d_x, d_y\right)\right\} \end{aligned}$$

2.2.3.10. Diferencia Promedio (AD)

La diferencia promedio es más simple que la medida (SAD) y relaciona el promedio de la diferencia entre la imagen de referencia y la modificada, está definida por la ecuación:

$$\begin{aligned} AD\left(d_x, d_y\right) &= \frac{1}{M \cdot N}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(I_C\left(i,j\right) - I_R\left(i+d_x,\ \ j+d_y\right)\right) \\ \left(mv_x, mv_y\right) &= \text{mín}_{(d_x,d_y)}\left\{AD\left(d_x, d_y\right)\right\} \end{aligned}$$

Los niveles negativos de AD, nos indican que los valores de los píxeles de la imagen reconstruida o fusionada son mayores en promedio que los de la imagen original y viceversa. Al obtener un valor cero para dicha medida, nos indica similitud entre la imagen reconstruida y la original. A mayores desviaciones de cero, menor calidad de la imagen obtenida.

2.2.3.11. Clasificación por diferencia de píxeles (PDC)

La clasificación por diferencia de píxeles es sencilla de implementar, pero existe la dificultad para establecer el valor adecuado del Theshold(límite), la cual se muestra a continuación:

$$\begin{aligned} PDC\left(d_x, d_y\right) &= \sum_{i=1}^{N}\sum_{j=1}^{M} T_{i,j}\left(d_x, d_y\right) \\ T_{i,j}\left(d_x, d_y\right) &= \begin{cases} 1 & si\ \ \left|I_C\left(i,j\right) - I_R\left(i+d_x,\ \ j+d_y\right)\right| < Theshold \\ 0 & en\ otros\ casos \end{cases} \\ \left(mv_x, mv_y\right) &= \text{mín}_{(d_x,d_y)}\left\{SAD\left(d_x, d_y\right)\right\} \end{aligned}$$

2.2.3.12. Entropía

La entropía es una medida estadística utilizada para medir el contenido de la información de una imagen o de una serie de imágenes. Un alto valor en la entropía nos indica que es alto el contenido de la información en una imagen. Se define de la siguiente forma:

$$H = -\sum_{k=1}^{M} p_k \log_{10}(p_k)$$

donde, M es el número de posibles niveles de gris de la imagen y p_k es la probabilidad de ocurrencia de una determinada escala de gris. A diferencia de las métricas expuestas anteriormente, la entropía no necesita una imagen de referencia.

2.3. Métodos para la determinación de flujo óptico

El método de flujo óptico, es definido como el campo de movimiento del patrón de brillo de la escena proyectada, causado por el movimiento relativo entre un observador y la escena. Este concepto fue introducido por primera vez en por el psicólogo (Von Helmholtz, 1925) y es tratado a profundidad en [55], dentro los campos vectoriales bidimensional podemos encontrar una revisión más compacta de estos métodos en [6] propuestos por (Barron I, 1994).

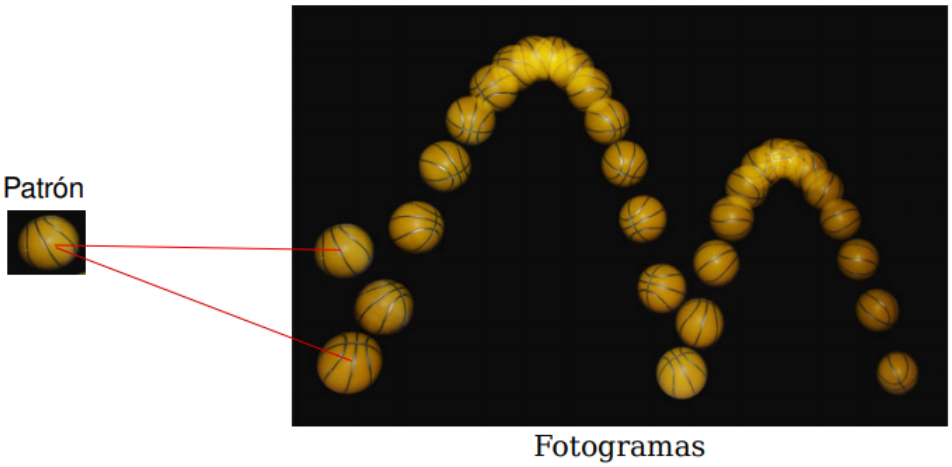
2.3.1. Estimación del flujo óptico

El enfoque para la determinación del flujo óptico consiste en la búsqueda del reconocimiento de ciertos patrones de cada uno fotogramas[1]. Si el patrón a identificar es $h(i, j)$ y la imagen $I(i, j)$, se calcula el máximo de la correlación entre ambas imágenes

[1]Un fotograma es cada una de las imágenes impresas químicamente en la tira de celuloide del cinematógrafo o bien en la película fotográfica (Konigsberg, 2004, p. 235). Cuando una secuencia de fotogramas es visualizada de acuerdo a una determinada frecuencia de imágenes por segundo se logra generar la sensación de movimiento en el espectador

$$corr\,(i,j) = \sum_{l}\sum_{m} I\,(l,m)\,h\,(i+l, j+m)$$

En la siguiente figura, nos muestra la correlación de un patrón de un fotograma



El cálculo de correlación de las dos imágenes es costoso y se puede agilizar haciendo uso de la transformada rápida de Fourier, del Coseno y sus propiedades. Los algoritmos centrados en la restricción del flujo óptico se pueden distinguir entre algoritmos dispersos y densos.

Los algoritmos dispersos determinan el flujo óptico de una serie de puntos de interés de la imagen en estudio. Uno de los algoritmos más utilizados para el cálculo del flujo disperso se encuentra en [39] cuyo algoritmo fue propuesto por (Lucas y Kanade, 1981) que se basa sólo en información local del píxel, derivada de una pequeña ventana del píxel de interés y en [4] por (Baker, 2004) en las que se proponen algunas de sus variantes. En el caso del tratamiento del flujo óptico denso lo encontramos en [27] desarrollada por (Horm y Schunck, 2004), la que se centran en la información global de la imagen.

2.3.2. Método de Segmentación

Un problema importante en la segmentación por movimiento ha sido la identificación de discontinuidades en el movimiento. Este problema ha sido abordado desde diferentes enfoques, como el mostrado por Wang y Adelson en [56] es uno de los primeros trabajos propuestos por (Wang, 1994), en la que se realiza un enfoque iterativo de crear y poner segmentos para obtener mejores resultados. También, ha habido propuestas aplicadas por (Ayer, 1995) con el fin de maximizar el enfoque anterior y que es tra-

tado con mayor detenimiento, profundidad en [3]. Sin embargo, la segmentación suele ser impredecible con estos métodos como los trabajados realizados por (Zitnick, 2005) y que son abordado a profundidad en [61].

2.4. Transformada Discreta del Coseno Directa

La aplicación de las transformadas al procesamiento digital de imágenes ha crecido considerablemente en las últimas décadas. La transformada discreta del coseno (DCT: Discrete Cosine Transform), es la más empleada en las aplicaciones de compresión de imágenes y vídeo.

Se han desarrollado algoritmos rápidos o eficiente de la transformada discreta del coseno, propuestos por (Chen and Smith, 1997) en [14] en la que proporciona un factor de mejora en su complejidad computacional, cuando se compara con los algoritmos tradicionales de Transformada Discreta de Coseno y la Transformada Rápida de Fourier. También, (Lee, 1984) introduce en [34] un nuevo algoritmo para la transformada de coseno discreta, en la que se reduce el número de multiplicaciones aproximadamente a la mitad de las requeridas por los algoritmos eficientes existentes.

Las propiedades únicas que posee la DCT permiten obtener altos índices de compresión a muy bajo costo, puesto que tienen una buena propiedad de compactación de energía. La decorrelación de los coeficientes es una de las tareas fundamentales para compresión, ya que, el posterior tratamiento de cada coeficiente se puede realizar de forma independiente, sin pérdida de eficiencia al cuantificar los coeficientes que se eligen de forma visual.

Siendo la transformada más utilizada, en los estándares de compresión de imágenes o tratamiento digital modernos, debido al hecho de que, para los datos de una imagen convencional, que tiene una alta correlación entre sus elementos y minimizan algunos problemas al aplicar otras transformadas, cuando se aplica a una serie de datos. Como los desarrollados por $MPEG$ o JVT, como en estándares de codificación de imágenes extendidos como $JPEG$.

La transformada discreta de coseno directa (DCT) es el proceso de descomponer un conjunto de

muestras en un conjunto ponderado de funciones base cosenoidales y al método de reconstruir el conjunto de muestras a partir del conjunto ponderado de funciones bases cosenoidales se llama Transformada Discreta de Coseno Inversa $(IDCT)$.

La DCT provee coeficientes de frecuencias correspondientes a una señal discreta variable en el tiempo (luminancia y crominancia). El concepto se basa sintéticamente en tomar cada píxel de un bloque de tamaño $n \times n$ pixel. Ese píxel es una "Muestra"(*sample*) de una señal variable en el tiempo, proporcional a la luminancia y de otra señal variable en el tiempo, proporcional a la crominancia.

2.4.1. Tranformada Discreta del Coseno Unidimensional

La expresión de la transformada directa de la DCT unidimensional viene dada por:

$$c(u) = \alpha(u) \sum_{x=1}^{N-1} f(x) \cos\left[\frac{(2x+1)\,u\pi}{2N}\right], \qquad u = 0, 1, \ldots, N-1 \qquad (1)$$

y la de la transformada inversa unidimensional por:

$$f(x) = \sum_{u=0}^{N-1} \alpha(u)\, c(u) \cos\left[\frac{(2x+1)\,u\pi}{2N}\right], \qquad x = 0, 1, \ldots, N-1 \qquad (2)$$

donde,

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}}, & u = 0 \\ \sqrt{\frac{2}{N}}, & u \neq 0 \end{cases} \qquad (3)$$

El conjunto de funciones base $\{b_u\}$ con $u = 0, 1, \ldots, N-1$, que forma el núcleo de transformación de la DCT viene dado por:

$$b_u(x) = \alpha(u) \cos\left[\frac{(2x+1)\,u\pi}{2N}\right], \qquad x = 0, 1, \ldots, N-1 \qquad (4)$$

Como se puede observar, las funciones base son ortogonales, por lo que ninguna de ellas puede ser representada como una combinación de las otras funciones base, es decir, son independientes. Esto provoca que el conjunto de funciones base (vectores) que forman el núcleo de transformación de la DCT inversa sea el mismo que el de la DCT directa.

Estas son las expresiones matriciales de las DCT directa e inversa unidimensionales:

$$C = BF \text{ y } F = B^T C$$

donde, F es un vector de tamaño $N \times 1$ que representa un vector de píxeles de una imagen, C es un vector de tamaño $N \times 1$ que contiene los coeficientes transformados, y B es una matriz de tamaño $N \times N$ que representa el núcleo de transformación de la DCT y sus filas corresponden con los vectores base de la DCT unidimensional $\{B_u\}$, donde

$$B = \begin{bmatrix} B_0^T \\ B_1^T \\ \vdots \\ B_{N-1}^T \end{bmatrix}$$

2.4.2. Transformada del Coseno Bidimencional

La DCT directa bidimensional es una extensión del caso unidimensional, y viene dada por:

$$C(u,v) = \alpha(u)\,\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)u\pi}{2N}\right], \quad u,v = 0,1,\ldots,N-1$$

Ya que hemos realizado la DCT, ahora para la decodificación utilizamos la transformada inversa (IDCT) que se utiliza para la obtención de un píxel aplicando la antitransformada DCT a un bloque de coeficientes. Del mismo modo, la DCT inversa bidimensional viene dada por:

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\,\alpha(v)\,C(u,v) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)u\pi}{2N}\right], \quad x,y = 0,1,\ldots,N-1$$

El núcleo de transformación de la DCT cumple las propiedades de separabilidad, simetría y ortogonalidad, por lo que la DCT bidimensional, directa o inversa, puede calcularse como dos DCT unidimensionales del siguiente modo

$$C = BFB^T \text{ y } F = B^TCB$$

donde, F es una matriz de tamaño $N \times N$ que representa un bloque de píxeles de una imagen en escala de grises, C es una matriz de tamaño$N \times N$ que contiene los coeficientes del bloque transformado, y B es una matriz de tamaño $N \times N$ que se corresponde con la matriz de transformación de la DCT unidimensional.

Como ya se ha comentado, las transformadas bidimensionales se pueden interpretar como la descomposición de una imagen en imágenes base en el caso de la transformada directa o como la composición

de imágenes base en el de la inversa. Las imágenes base de la DCT se pueden generar fácilmente a partir de los vectores base de la DCT unidimensional, mediante la multiplicación de dichos vectores con sus traspuestos:

$$B_{(u,v)} = B_u B_v^T \begin{bmatrix} b_{(u,0)} \\ b_{(u,1)} \\ \vdots \\ b_{(u,n-1)} \end{bmatrix} \begin{bmatrix} b_{(v,0)} & b_{(v,1)} & \cdots & b_{(v,N-1)} \end{bmatrix}$$

2.4.2.1. Espectro de la DCT

Aplicar la DCT a un bloque es equivalente a representarlo con el conjunto de imágenes base del núcleo de transformación de la DCT, lo que es equivalente a aislar las distintas frecuencias contenidas en el bloque, indicando cada coeficiente transformado el peso de dicha frecuencia en la imagen. Las bajas frecuencias contienen la información importante (psicovisual), mientras que las altas frecuencias corresponden con los detalles de la imagen por lo que son menos importantes. Gracias a esta distinción en las frecuencias, se utiliza distintos escalones de cuantificación para cuantificar los coeficientes transformados, de forma que el escalón es más pequeño para las bajas frecuencias y va aumentando a medida que crece la frecuencia según lo planteado en [49] por (*Salomon*, 2007).

En general, si los datos de entrada están altamente correlacionados, entonces la mayoría de los coeficientes transformados resultantes, después de la aplicación de la DCT son cero o cercanos a cero, y sólo unos pocos son valores altos (normalmente los primeros), es decir que el proceso de cuantificación la mayoría de los coeficientes de la DCT serán cero.

La imagen a la izquierda presenta las imágenes originales ((a) la imagen médica y (d) la imagen fractal), en la parte central nos muestra el resultado de la transformación de Fourier (correspondiente a la imagen (b) y (e)) y más a la derecha se muestra la trasformada del coseno (perteneciente a las imágenes (c) y (f)). Se puede observar que los puntos de la transformada del coseno están más cargados en la esquina superior izquierda, y podemos decir que los coeficientes de las transformadas representan a

estas imágenes de manera más eficientes (en el caso de la transformada del coseno solo determina la parte real correspondiente al primer cuadrante, en cambio la transformada de Fourier necesita la información de todos los cuadrantes por tratarse de datos muy complejos y la información esta distribuida en toda la imagen), esto se muestra en las siguientes figuras:

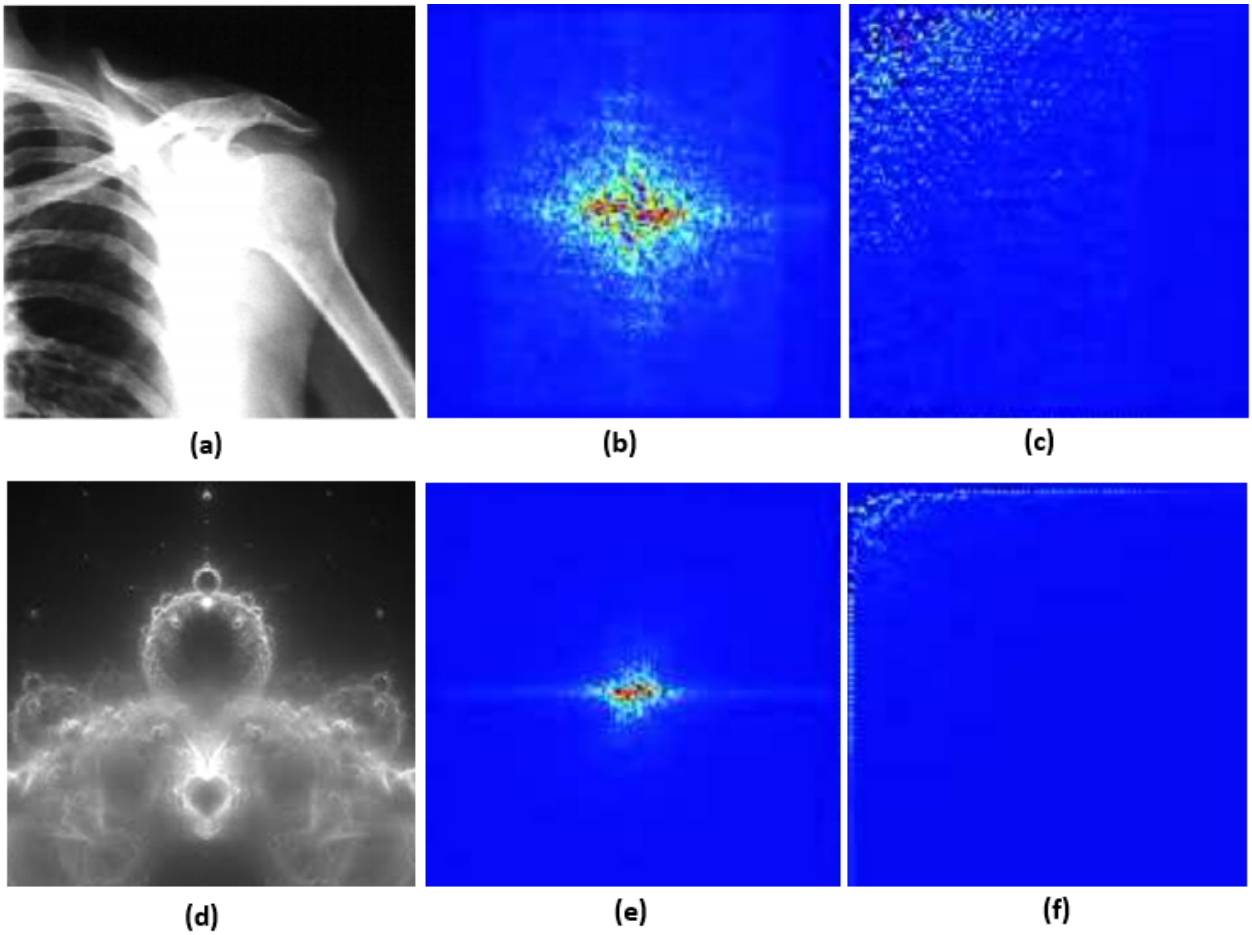


Figura 2.7: Espectro de la DCT y TDF

2.4.2.2. Selección Del Tamaño De Bloque

Con la transformada discreta lo que se busca es, a partir de unos píxeles altamente correlacionados, obtener unos coeficientes con la menor correlación posible, de tal forma que se produzca una alta compactación de la energía en unos cuantos coeficientes y se pueda conseguir una mayor compresión en los procesos posteriores de cuantificación y codificación entrópica. En cambio, si la correlación entre los píxeles es débil, la compactación de la energía que la transformada discreta provoca, disminuirá consi-

derablemente, por lo que aumentará la energía eliminada en el proceso de cuantificación y con ello la medida del error MSE.

Como ya se ha visto con anterioridad, la correlación entre píxeles cercanos es alta, pero dicha correlación disminuye según aumenta la distancia entre los píxeles. Por tal motivo, la imagen se subdivide en bloques, es decir cuadrados de tamaño $N \times N$ píxeles, y es a los bloques a los que se les aplica la transformada.

En [49] plantea que la mayoría de las imágenes están correlacionadas de forma significativa entre sus píxeles y que sólo se necesita entre 20 píxeles adyacentes según lo propuesto por (**Salomon, 2007**). Aunque, este número depende en gran medida de la cantidad de detalle de la imagen analizada, de hecho, no se puede garantizar que para tamaños con $N > 16$ el rendimiento no se vea reducido, según lo demostrado (**Wintz, 1972**) y tratado a profundidad en [58] . El tamaño adoptado en el estándar de codificación de imágenes $JPEG$, así como en los estándares de codificación de vídeo $H.261$, $H.263$, $MPEG-1$ y $MPEG-2$ es $N = 8$.

2.4.2.3. Interpretación de la DCT

La transformada del coseno se puede interpretar como la proyección de la imagen sobre cada una de las imágenes base. Los coeficientes representan la presencia de dichas componentes o imágenes bases. Las funciones base, el valor de cada uno de los coeficientes $C[u, v]$ se determina como un producto escalar de dos vectores: mediante la suma de los productos entre elementos homólogos de la función base, y de la matriz a transformar, afectado por los factores $\alpha(u)$ y $\beta(v)$. Para una imagen de 8×8 las funciones base correspondientes a cada una de las frecuencias de la DCT bidimensional se muestra a continuación:

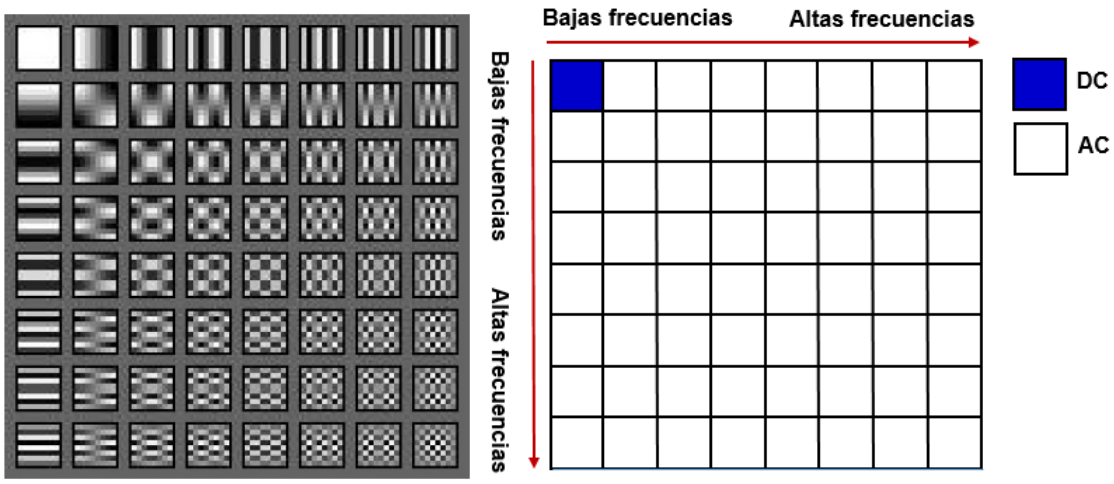


Figura 2.8: Frecuencias de la DCT

Al analizar las figuras anteriores, se tiene que cada coeficiente de la transformada representa la cantidad de información de cada uno de los bloques de tamaño 8×8 del conjunto inicial, las frecuencias verticales y horizontales denotadas por los valores de u y v, éstas comprenden desde la frecuencia cero correspondiente a los coeficientes DC localizado en la esquina superior de cada bloque, hasta las máximas frecuencias espacial horizontal posible, representada en la parte superior derecha, o la máxima frecuencia espacial vertical posible representada en la parte inferior izquierda, pasando por todas las posibles combinaciones de frecuencias espaciales horizontales y verticales correspondientes a los coeficientes AC.

La lectura de los coeficientes de la DCT se realiza en orden zig$-$zag, iniciando con el coeficiente correspondiente DC, y finalizando en su lado opuesto. Permitiendo procesar las bajas frecuencias espaciales y tomar la decisión si se procesan o se eliminan los coeficientes relacionados a las altas frecuencias. Esto se observa con mayor detalle en la siguiente figura:

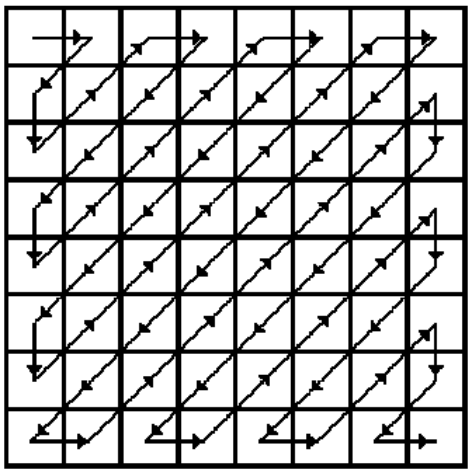
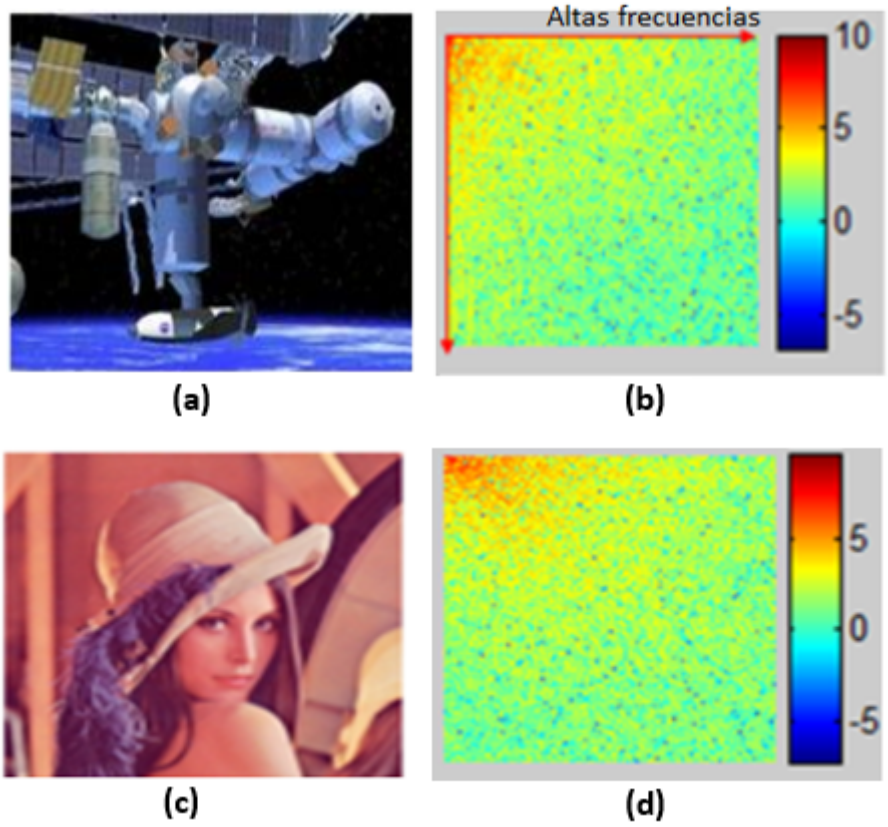


Figura 2.9: Lectura de los coeficientes

La transformada discreta del coseno DCT concentra la mayor parte de la información en los primeros coeficientes como se ha mostrado en las figuras anteriores, así como en las siguientes figuras (Las imágenes (a) Satélite y (c) Lena) que nos muestra estos detalles en (b) y (d) y la mayor concentración de la información en la esquina superior izquierda, esto se muestra a continuación:



2.4.2.4. Algoritmo básico de la DCT

A continuación, se muestra el algoritmo básico de la transformada discreta del coseno y que será implementado en el software matlab en los siguientes capítulos:

Algoritmo 1 Transformada Discreta del Coseno

$N = 32;$
$C = zeros(N);$
$C = zeros\,(N)\,;$
for $u = 1 : N$
    for $v = 1 : N$
        if $(u == 1)\ \&\ (v == 1)$
            for $x = 1 : N$
                for $y = 1 : N$

$$C\,(v,u) = C\,(v,u) + \frac{1}{N} A\,(y,x) \cos\left(\left(2\,(x-1)+1\right)\pi\frac{u-1}{2N}\right)\cos\left(\left(2\,(y-1)+1\right)\pi\frac{v-1}{2N}\right)$$

                end;
            end;
        else
            for $x = 1 : N$
                for $y = 1 : N$

$$C\,(v,u) = C\,(v,u) + \frac{2}{N} A\,(y,x) \cos\left(\left(2\,(x-1)+1\right)\pi\frac{u-1}{2N}\right)\cos\left(\left(2\,(y-1)+1\right)\pi\frac{v-1}{2N}\right)$$

                end;
            end;
        end;
    end;
end;

2.5. Compresión de vídeo digital

Los estándares modernos sobre el proceso de compresión de vídeo digital están constituidos por diferentes etapas, las que se encargan de reducir la redundancia existentes entre una imagen o una secuencia de imágenes. Los tres tipos de algoritmos de compresión de vídeo digital, que se pueden enunciar de acuerdo a la reducción de redundancia, son los siguientes:

- **Los algoritmos espaciales:** estos aprovechan el hecho de que el ojo humano no es capaz de distinguir pequeñas diferencias de color, así como las pequeñas diferencias en brillo. Es importante señalar que la redundancia a explorar y explotar se da únicamente sobre el espacio o tratamiento dentro de una imagen.

- **Los algoritmos de dominio temporal:** estos se benefician de la la redundancia de la información existente entre imágenes consecutivas y se emplean en el tratamiento de imágenes en movimiento, es decir, dentro de los estándares de vídeo. Este tipo de compresión contiene implícitamente la compresión espacial, ya que se necesita comparar dentro de una sola imagen fija antes de comparar con imágenes consecutivas. La importancia del uso de esta técnica, radica en que para una secuencia en movimiento muchos de los píxeles de una imagen a otra no se modifican en su totalidad, codificándose únicamente aquellos píxeles que hayan sufridos ciertas alteraciones.

- **Los algortimos espectrales:** se aprovechan de la correlación encontradas entre diferentes aspectos de color o bandas de frecuencias.

De forma general, existen dos grandes grupos de técnicas de compresión de vídeo digital: sin pérdidas y con pérdidas, que han sido estudiadas en secciones anteriores y que son válidas para la compresión de vídeos. La codificación sin pérdidas, la descompresión tiene como objetivo recuperar exactamente la información original de la imagen de tal forma que el porcentaje de compresión sea buena, mientras que la compresión con pérdidas el proceso de compresión permite una pérdida de información que no se recupera en la descompresión, como afirma (Bovik, 2010) en su libro [10] de tratamiento de imágenes y vídeo.

Capítulo 3

ALGORITMOS DE ESTIMACIÓN DE MOVIMIENTOS

La idea central de la estimación de movimiento es la obtención de la información de movimiento percibida a través de la información de la imagen actual y una o varias imágenes de referencia, parte esencial de los codificadores de vídeo modernos de la familia de $MPEGs$ (MPEG-1, MPEG-2, MPEG-3), el último H.264 (= AVC, MPEG-4), la mayoría se basa en la estimación por bloques, por su simplicidad y eficiencia algorítmica. En cambio, la compensación de movimiento es el proceso de generación de la imagen predicción a partir de una o varias imágenes de referencia y la información de movimiento obtenida en la estimación.

La **estimación de movimientos** es la parte principal del proceso de reconstrucción de imágenes digitales y codificación de vídeo. La **implementación de algoritmos eficientes** nos permite **reducir** la **redundancia temporal** en imágenes, secuencias de imágenes o vídeo, representa el $\frac{2}{3}$ del **costo de complejidad computacional** del proceso de reconstrucción y codificación de vídeo. Sin embargo, el problema tiende hacerse más complejo a medida que aumentan el número de áreas o regiones que se analizan y/o se mueven, aumenta el tamaño de las imágenes. Por otra lado, la codificación de segmentos de bordes, aumenta la complejidad computacional y el tamaño del código a implementar. Además, se

aplica sobre diferentes soportes de modelo de movimiento, siendo utilizados: imagen completa, píxel a píxel, por bloques o por regiones de interés.

La estimación de movimiento realizada en una imagen de forma completa **píxel a píxel:** se representa por un conjunto de parámetros (desplazamiento, velocidad y aceleración), siendo computacionalmente muy costoso de implementarlo de forma exacta.

Estimación de movimiento de forma recursiva píxel a píxel: consiste en que cada bloque se establece una estimación inicial de su desplazamiento, esta predicción se va corrigiendo iterativamente en base al error de predicción medido, siendo también muy costoso computacionalmente.

Estimación de movimiento por bloques la imagen se divide en bloques rectangulares disjuntos entre sí, de tamaño fijo. Se considera que sólo se producen traslaciones, siendo el modelo por excelencia utilizado con frecuencia en el proceso de compresión, principalmente por su simplicidad y sus buenos resultados de implementación. Uno de los **objetivos consiste en la búsqueda de la máxima coincidencia del movimiento real de los objetos, pero la tarea más fuerte es la reducción del error de predicción y la calidad visual de las imágenes**. Sin embargo, este presenta diferentes problemas ante movimientos más generales como una sencilla transformación afín que incluya escalado y/o rotación. Los modelos de movimiento se pueden agrupar en dos grandes clases: los modelos paramétricos y no paramétricos.

Los modelos de estimación paramétricos pueden llevar a cabo estimaciones en perspectiva o modelos afines, lográndose modelar movimientos más complejos que la traslación, mayor complejidad y la necesidad de una adecuada cuantificación de los parámetros de los modelos afines; mientras que en los modelos no paramétricos se pueden encontrar métodos de estimación de flujo óptico muy utilizados en segmentación, métodos por bloques, recursividad por bloques y bayesianos. Bajo este contexto únicamente se menciona los últimos, haciendo énfasis en los métodos por bloques debido a su predominancia dentro de los estándares de compresión.

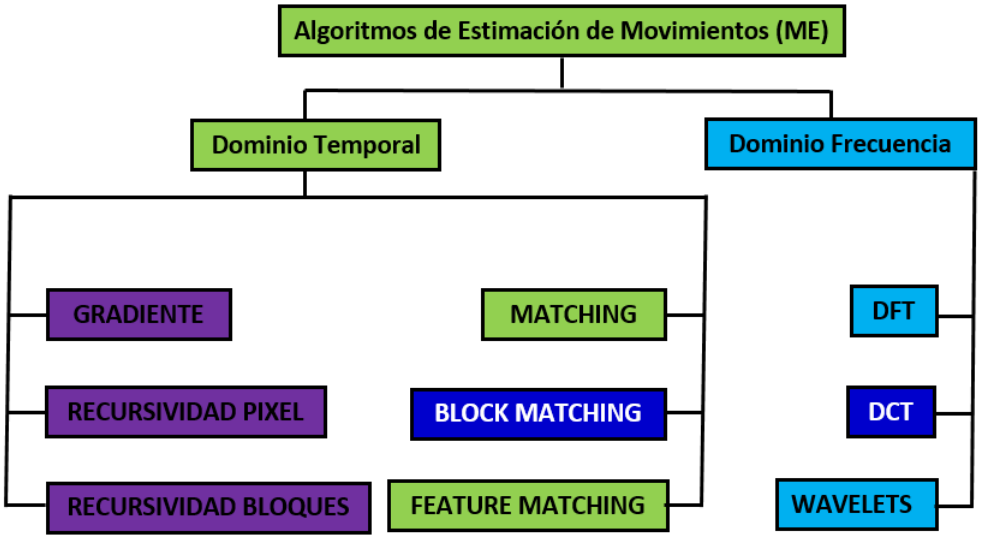
3.1. Métodos de Coincidencia por Bloques

Los métodos de coincidencia por bloques, cuyo término se deriva de inglés "Block matching" comprende toda una categoría de algoritmos que se comportan de forma similar dividiendo la imagen en regiones pequeñas denominados bloques, con frecuencia en cuadrados, rombos, entre otros. Estos intentan dividir las imágenes previa y actual en bloques y luego calcular el movimiento, o la estimación de estos bloques. Además, son la base para muchos algoritmos de coincidencia para la estimación de movimiento en compresión de imágenes, así como en los algoritmos de flujo óptico de visión por computadora.

En los trabajos [52] propuesto por $(Stiller, 1999)$ y [44] por $(Redert, 1999)$ se analizan los métodos que utilizan la imagen en proceso y los que no, en las que se trabaja la información para el cálculo de los vectores. Dentro de los métodos que no utilizan la imagen en proceso se encuentran las técnicas recursivas, centradas directamente en la ecuación de flujo óptico y que tienen la ventaja que no necesitan transmitir ninguna información de movimiento; sin embargo el costo que deben pagar es una predicción con un mayor error que el que se obtiene con otras técnicas. Por otro lado, las técnicas que emplean para realizar la estimación de la información sobre el propio cuadro en proceso, necesitan transmitir información de movimiento al decodificador, pero obtienen una mejor predicción, por lo que el análisis se ajustará a ellas.

A lo largo del estudio se revisaron y analizaron los algoritmos de coincidencia por bloques, la estimación de movimiento de búsqueda completa tratado en [13] por $Ratnam$, el algoritmo de búsqueda en cruz en [20] por $(Ghanbari, 1990)$, el algoritmo de búsqueda de tres pasos en [37] por (Li, Zeng y Liou, 1994), el algoritmo de búsqueda de cuatro pasos en [42] por $(PoL.yMaW., 1996)$, el algoritmo de Búsqueda en Diamante en [15] por $(Cheung, 2002)$, y la búsqueda de Patrón Adaptativo Rood en [41] por (Nie, 2002).

El siguiente esquema reúne los diferentes algoritmos de estimación de movimientos más utilizados e implementados a lo largo de los trabajos analizados:



Los algoritmos de dominio de frecuencia se aplican sobre los coeficientes de las transformadas **DFT, DCT o Wavelets**, las que se implementaran en los siguientes capítulos.

Los algoritmos de dominio temporal se centran en la búsqueda de la máxima coincidencia del área de búsqueda, dentro de éstos forman parte los **algoritmos de coincidencia** que son objetos principales de nuestros estudio y los algoritmos basados en métodos de los gradientes. Estos centran su búsqueda en las coincidencias de la información (datos, características, objetos,etc.) del bloque actual con la del frame, bloque o marco, de referencia.

La idea más simple consiste en dividir una imagen en pequeños bloques rectangulares o cuadrangulares, de tamaño fijo, bajo el supuesto de que cada uno realiza una traslación independiente. Si estos bloques son suficientemente pequeños, se pueden aproximar movimientos como la rotación de objetos grandes, o el zoom de la imagen, a partir de traslaciones de estas áreas o regiones, este método evita la codificación relativa a la segmentación y sólo es necesario codificar el vector de desplazamiento de cada bloque. La hipótesis sobre la que se sustentan estos métodos es que un bloque de tamaño $L \times L$ en la imagen $k-1$ centrado en el punto (x_0, y_0) coincide con un bloque del mismo tamaño en la imagen k.

Esto expresado matemáticamente quedaría

$$f(x, y, k) = f(x + d_1, y + d_2, k - 1) \ \ \forall (x, y) \in B$$

Dicho de otro modo, cada uno de los bloques que componen cada una de las imágenes o frames de la secuencia de imágenes que posee un homólogo, decir, un bloque del mismo tamaño que representa la misma porción de la escena en otra imagen o frame consecutivo de la secuencia de imágenes.

Para determinar cuál es el bloque del frame objetivo de mejor coincidencia, con respecto del bloque del frame actual, es necesario emplear una función de costo. Para este tipo de problema, se consideran las funciones de costos más utilizadas tales como:

$$MAD(i, j) = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |C_{ij} - R_{ij}|$$

$$MSE(i, j) = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (C(i, j) - R(i, j))^2,$$

donde,

- (i, j) está definido dentro del área de búsqueda.
- $(N \times M)$ determina las dimensiones del macrobloque.
- C_{ij} y R_{ij} definen los pixels del macrobloque actual y referencia respectivamente.

Las coordenadas (i, j) que menor MAD exhiban determinarán el vector de movimiento del macrobloque actual.

En la estimación de movimientos forward, es decir hacia adelante, se definen los macrobloques en la imagen referencia, se determinan los vectores movimientos a partir de la imagen referencia. Para cada uno de los macrobloques de la imagen de referencia se busca la imagen actual, que se codificará, y a partir de la nueva posición, se extrae los nuevos vectores de movimientos. Algunos macrobloques o píxeles de la imagen referencia no aparecen en la original, puesto que se identifica una zona de la imagen a codificar como la más similar a más de uno de los macrobloques de la imagen de referencia.

En la siguiente figura se muestra la estimación de movimiento por bloques tipo forward:

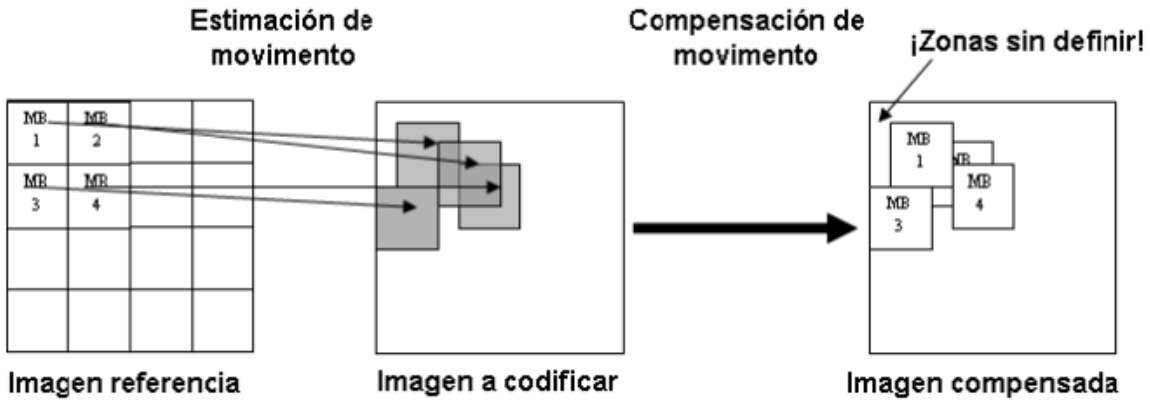


Figura 3.1: Estimación de movimientos forward

La estimación de movimientos backward, hacia atrás, define los macrobloques en la imagen a codificar y se busca dónde se encontraban en la imagen de referencia, se determinan los vectores de movimientos a partir de la imagen actual. Para llenar cada macrobloque de la imagen compensada se selecciona el área donde se encontraba en la imagen de referencia y copia todos los píxeles. Lo único que se hace es llenar los macrobloques según los vectores de movimientos que han de encontrar, por lo que no quedará píxel sin definir. En la siguiente figura se muestra la estimación de movimiento por bloques tipo backward:

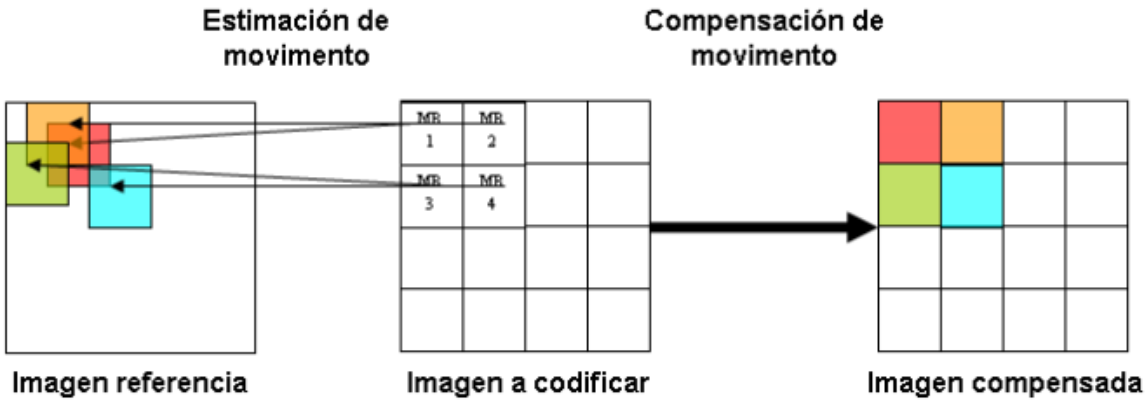


Figura 3.2: Estimación de movimientos backward

Una vez introducidas las técnicas de estimación de movimiento por bloques, pasaremos a presentar cada uno de los métodos, procedimientos y algoritmos, algunas de las variantes más conocidas dentro

del conjunto de algoritmos a implementar en los siguientes apartados.

3.2. Algoritmos Heurísticos de coincidencia utilizados para la Estimación de Movimiento

Los algoritmos diseñados **block matching** (FS, TTS, 4SS, DS, MHTSS, MHCS), centran su búsqueda en la máxima coincidencia entre todos o algunos de píxeles en el bloque actual con el bloque de referencia sobre el área de búsqueda, basados en alguna métrica o función de costos que han sido abordada a profundidad en el capítulo anterior.

La mayor parte del estudio se centra tanto en las técnicas de estimación y compensación de movimientos como en el análisis de la segmentación de las imágenes para lograr un mayor nivel de estimación y compensación, para esto se estudia la segmentación por bloques fijos, por bloques variables y la segmentación basada en regiones tomando los objetos dentro de la imagen.

3.2.1. Algoritmo de Búsqueda Exhaustiva (FS)

Los algoritmos utilizado van desde los analizados de forma exacta o en la búsqueda en todo el espacio de soluciones, ya que que tienen la ventaja de garantizar siempre el óptimo global, pero con el inconveniente que en problemas reales su tiempo de ejecución crece exponencialmente a medida que aumenta el número de imágenes o frame, secuencias imágenes y vídeos analizados. Es la razón principal por la que se analizará los heurísticos que son bastante rápidos, pero la calidad de las soluciones encontradas suele ser bastante pésimas. Por otro lado, se desarrollará las metaheurísticas ya que nos ofrecen un equilibrio entre ambos extremos; estos son métodos genéricos que ofrecen una buena solución y en muchos casos se llega obtener el óptimo global en un tiempo razonable de cómputo.

Los algoritmos de búsqueda exhaustiva (FS) exploran todas las posibles combinaciones de los elementos. Estos tipos algoritmos con frecuencia raramente exploran todas las posibles combinaciones de

los elementos sobre el espacio de búsqueda, sino que expanden las combinaciones más prometedoras, de acuerdo a un cierto criterio relacionado con la función objetivo que se pretende optimizar.

Este algoritmo calcula la función de costo para todas las posibles localizaciones dentro de la ventana de búsqueda. Este algoritmo encontrará el homólogo que proporciona la mejor $PSNR$ de entre todos los posibles bloques de la imagen objetivo dentro de la ventana de búsqueda fijada. El único inconveniente es que es muy costoso desde el punto de vista computacional. El problema anterior es una de las razones que nos motiva para definir e implementar nuevos algoritmos de búsqueda que logren el mismo valor para $PSNR$ y la calidad visual que este método, pero realizando el menor número de cálculos posibles.

El algoritmo de búsqueda completa diseñado es el siguiente, basados en la máximas coincidencia para la estimación de movimientos, el cual se muestra a continuación:

Algoritmo 2 Búsqueda Exhaustiva

1: $MinMAD = MAXVALUE;;$
2: $MV = (0, 0);$
3: **for** $m = -p$ **to** $+p$ **do**
4: **for** $m = -p$ **to** $+p$ **do**
5: $MAD\,(m, n) = 0;$
6: **for** $i = 0$ **to** $N - 1$ **do**
7: **for** $j = 0$ **to** $N - 1$ **do**
8: $MAD\,(m, n) = MAD\,(m, n) + |cur\,(i, j) - ref\,(i + m, j + n)|$
9: **end for**;
10: **end for**;
11: **if** $MAD\,(m, n) < MinMAD$ **then**
12: $MinMAD = MAD\,(m, n)$
13: $MV = (m, n)$
14: **end if**;
15: **end for**;
16: **end for**;

Algunas de las características esenciales que se tendrá en cuenta en la implementación de la búsqueda exhaustiva o completa son:

- Examina todos los puntos del área búsqueda $(+/-p)$.
- Costo computacional alto.
- El algoritmo calcula la función de costo para cada posible posición dentro de la ventana de búsqueda.

En general, para una imagen de tamaño $N \times N$ píxeles, y un rango de búsqueda de $\pm p$ píxeles, tanto en el sentido horizontal como vertical y con un paso de un píxel, el número total de bloques candidatos para cada bloque en la imagen previa es $(2R+1)^2$. El número de operaciones necesarias para calcular el MAD para cada candidato es N^2, es decir, cada bloque necesitará $(2R+1)^2 N^2$ operaciones.

Por ejemplo, para una imagen de tamaño $M \times M$ píxeles, se tienen $\left(\frac{M}{N}\right)^2$ bloques (asumiendo que N es múltiplo de M). Con estas premisas, para una imagen determinada, se deben realizar $M^2(2R+1)^2$ operaciones. Es muy importante resaltar que el número de operaciones totales no depende del tamaño de bloque sino del valor de N.

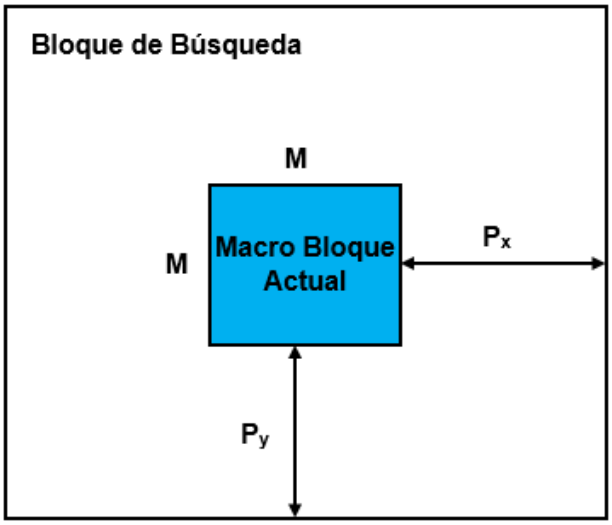
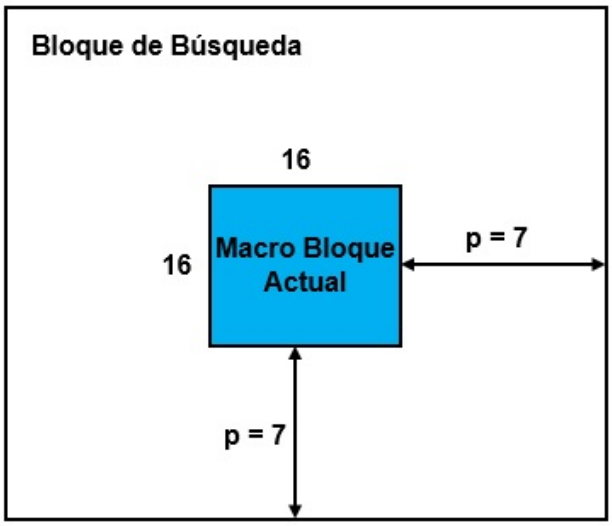


Figura 3.3: Sistema de búsqueda de tamaño $M \times N$

El procedimiento que se sigue para el caso particular de este tipo de métodos consiste en dividir el frame actual de la secuencia de imágenes en una matriz de macro bloques y luego buscar este mismo bloque en alguno de los frames de la secuencia de imágenes consecutivos. En el caso particular, los tamaños de bloque que suelen considerarse son de tamaño 16×16 píxels. La zona o bloque de búsqueda queda determinada por un parámetro p denominado parámetro de búsqueda. El significado de este parámetro queda ilustrado en la siguiente figura.



En la figura anterior muestra una zona de búsqueda dentro del frame objetivo determinada por un bloque de tamaño 16×16 y un parámetro de búsqueda $p = 7$. Para cada bloque se tratará de localizar otro bloque en una imagen de referencia de tal forma que minimice la función de costos.

3.2.1.1. Complejidad Computacional

En el caso particular, para una imagen de 512×16 píxeles, y con un valor de $p = 16$, el número total de operaciones por imagen es de $82{,}85 \times 10^8$. Con lo que para una secuencia con 30 fps, el número se incrementa a $98{,}55 \times 10^9$operaciones, lo que resulta una cifra astronómica. Por otro lado, si la fuente de información es de tamaño $N = 100$, y supongamos que una computadora realiza 10^{10}, es decir 10 giga de operaciones por segundo, encontrar una solución nos llevaría aproximadamente 4×10^{12} años para completarse sobre el espacio de búsqueda, mucho más tiempo que la edad actual del universo.

De esta manera, se observa cómo el método exhaustivo requiere un costo computacional muy alto, que puede impedir que sea utilizado en aplicaciones en tiempo real. Es por tal razón, que se diseñara y analizará los heurísticos que suelen ser bastante rápidos (eficientes) y de la calidad de las soluciones encontradas suele ser bastante buenas.

Un ***algoritmo eficiente*** es un algoritmo de complejidad polinomial. Un problema está bien resuelto si se conocen algoritmos eficientes para resolverlo. Un ***problema es intratable*** si no puede ser resuelto por algún algoritmo eficiente. Un problema puede ser intratable por distintos motivos: El problema re-

quiere una repuesta de longitud exponencial, es indecidible o es decidible pero no se conocen algoritmos polinomial que lo resuelvan.

La teoría de la complejidad computacional plantea que un problema de decisión se puede clasificar en: la clae L, NL, P, NP, NP-Completos y NP-difícil, Co-NP . Sin embargo, para esta distinción es necesario considerar el modelo teórico de las máquinas de Turing (MT):

- ***MT Determinista***: en la que para cada par (estado, símbolo), existe como máximo una transición a otro estado.
- ***MT No Determinista***: en la que existe al menos un par (estado, símbolo), con más de una transición a estados diferentes).
- ***Clase P(Polynomial-time)***: Contiene aquellos problemas de decisión que una MT determinista puede resolver en tiempo polinómico, estos tipos de problemas son tratables, es decir en la práctica se pueden resolverse en un tiempo razonable.
- ***Clase NP (Non-Determinitic Polynomial-time)***: Contiene aquellos problemas de decisión que una MT no determinista puede resolver en tiempo polinómico.
- ***Clase NP-Completo***: Un problemas de decisión es NP-Completo si y sólo si es NP y todos los demás problemas de NP se pueden se pueden reducir a él en tiempo polinómico. El teorema de Cook (1971), demuestra que el problema SAT es NP-completo.
- Un problema de decisión π es NP-difícil si todo otro problema de NP se puede transformar polinomialmente a π.
- Un problema de decisión pertenece a la clase Co-NP si dada una instancia de **NO** y evidencia de la misma, puede ser verificada en tiempo polinomial.

  El problema complemento de un problema de decisión π, π^c, es el problema de decisión que responde al complemento de la decisión de π.

En la siguiente figura se resume lo anterior

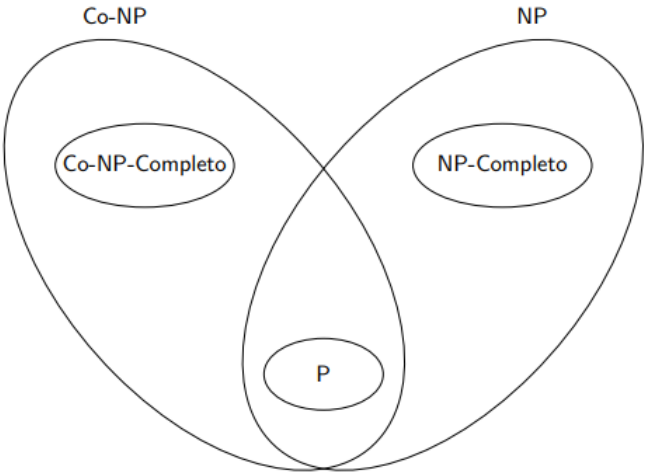


Figura 3.4: Complejidad teórica computacional

de lo anterior surgen las siguientes interrogantes:

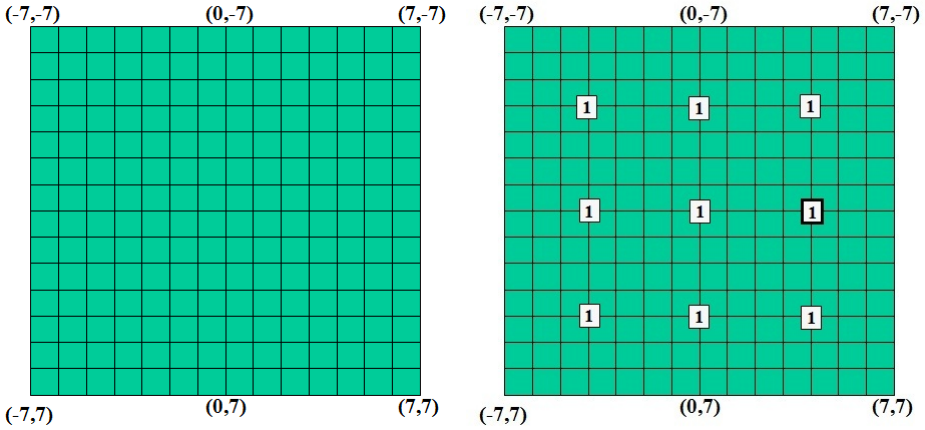
$$P = NP, Co - NP = NP, P = Co - NP \cap NP,$$

que en la actualidad sigue siendo objeto de estudio para la comunidad científica.

3.2.2. Búsqueda en tres pasos (TTS)

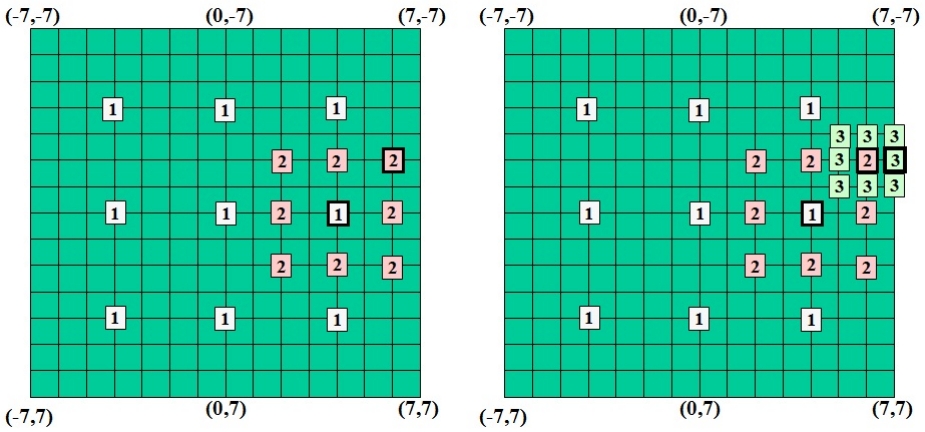
El algoritmo de búsqueda en tres pasos (TTS) es uno de los primeros algoritmos implementado para la estimación rápida del movimiento, en la que el costo computacional es reducido considerablemente respecto a las búsquedas exhaustivas, es simple de implementar y con alto rendimiento en el costo computacional con respecto a los demás métodos de estimación.

El algoritmo parte con la definición de un paso de búsqueda S y de un punto central. Luego, se compara el bloque a estimar de la imagen previa con los bloques que forman un cuadrado alrededor del mismo punto en la imagen actual. De esta manera se tiene el punto central y ocho puntos alrededor a una distancia entre sí de un paso de búsqueda. El macrobloque inicial de búsqueda en tres pasos se muestra en imagen a la izquierda y el algoritmo con un paso 1 se muestra a la derecha, como se muestra en la siguiente figura:



Luego, el siguiente procedimiento comienza con un paso de tamaño $S = 4$, con un parámetro de búsqueda de 7. Se evalúa la función de costo o métrica utilizada en las 8 localizaciones situadas en torno al píxel central del bloque. Además, del propio píxel central. De las 8 localizaciones mencionadas, 4 se encontrarán a una distancia raíz de dos veces el tamaño del paso S y 4 a distancia S.

Una vez que se ha evaluado la función de costo en estos 9 puntos, se comprueba cuál de estos da un valor mínimo, se toma este nuevo píxel como central, se reduce el tamaño del paso S a la mitad y se repite el procedimiento. Este proceso continuará hasta que el valor de S sea igual a 1 , que nuestro caso, será la siguiente iteración (de ahí el nombre del algoritmo). En la siguiente figura nos muestra el paso 2 de la imagen, cuya imagen es el de la izquierda y con un paso de 3 a la derecha, con un vector de movimiento encontrado en $(7, -3)$.



Finalmente, se muestra el diseño del heurísticos de búsqueda por bloques en tres pasos, que más adelante se implementará.

Algoritmo 3 Búsquedad en tres pasos

1: **Se busca en la posición** $(0, 0)$**.**
2: **Se define el tamaño del paso** $S = 2^n$**.**
3: **Seleccionamos 8 posiciones** $\pm S$ **píxeles próximos al orige** $(0, 0)$**.**
4: **De las nuevas posiciones se elige aquella con menor valor del** MAD**.**
5: **Se toma** $S = \frac{S}{2}$**, el nuevo origen de búsqueda y el punto obtenido en** 4**.**
6: **Repetir los pasos** $3 - 5$ **hasta que** $S = 1$**.**

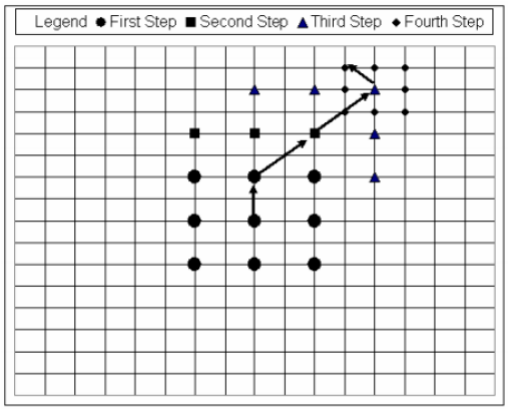
Es interesante notar que en la figura anterior se determina la función de costo para 25 macro bloques, mientras que, si usásemos el método de búsqueda exhaustiva, se hubiese calculado para 255 bloques. Por tanto, se ha reducido el costo computacional en un factor 9 veces con respecto búsqueda exhaustiva con un valor de $p = 7$, el algoritmo ES posee un costo computacional de 225 bloques, mientras que con TSS se comparan únicamente 25 bloques. La desventaja de aplicar la búsqueda exhaustiva es que existe el riesgo de que la función de costo, métrica o error no sea unimodal, es decir, no se encuentre el mínimo o máximo global. En este caso la función de calidad $PSNR$ y la imagen obtenida se verá degradada (pérdida en la calidad visual de la imagen al ser recuperada).

3.2.3. Algoritmo de Búsqueda en cuatro pasos (4SS)

El diseño del algoritmo $4SS$ es similar al algoritmo SES, el $4SS$ aplica una búsqueda parcial y tiene la posibilidad de saltarse pasos intermedios en el caso de que se cumpla alguna condición. Además, fija un patrón paso de tamaño $S = 2$ en el primer paso, independientemente del valor asignado a p.

Se realiza las diferentes comparaciones con el punto central y los 8 puntos alrededor del centro, en las direcciones cardinales, teniendo una ventana de búsqueda de 5×5. En el caso que se determine el error mínimo con respecto al píxel central, el algoritmo salta al cuarto paso de búsqueda, que consiste en una búsqueda con paso de $S = 1$ en torno al punto central. Con una ventana de 3×3. El punto con menor peso es el de mejor coincidencia para el macrobloque y el vector de dirección al que apunta a esa dirección.

La siguiente figura muestra este hecho:



Es importante notar que este algoritmo evalúa la función de costo 17 veces mejor y 27 veces peor de los casos que la búsqueda completa.

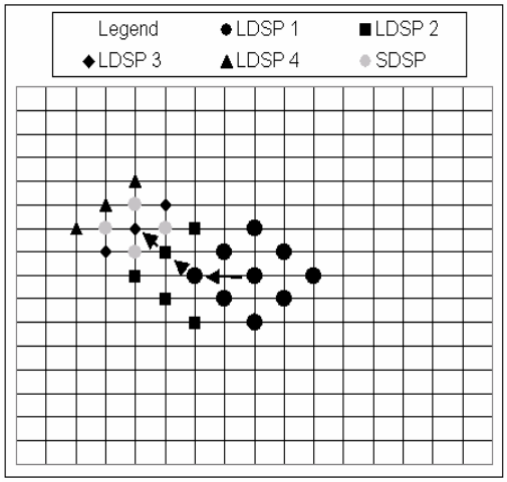
3.2.4. Búsqueda en diamante (DS)

El diseño del algoritmo heurístico de búsqueda en diamante (DS), es similar al algoritmo en la búsqueda con cuatro pasos, sólo que en lugar de aplicar un patrón de búsqueda en forma de cuadrado usa un patrón en forma de rombo. Además, no hay límite en el número de pasos a utilizar. El algoritmo usa dos patrones de búsqueda, uno grande: conocido como LDSP y uno pequeño, que lo emplea en el último paso de la búsqueda conocido como SDSP.

El criterio de parada se da cuando llega al último paso, siendo necesario para ello que en un paso intermedio se obtenga un valor mínimo para la función de costo del punto central.

Este método no tiene número de pasos limitados, los resultados ofrecidos pueden llegar a ser muy precisos, alcanzando un valor del $PSNR$ muy próxima al obtenido mediante el método búsqueda exhaustivo, mientras que el costo computacional es significativamente menor.

La siguiente figura nos detalla esta idea:



3.2.5. Búsqueda logarítmica

El algoritmo búsqueda logarímica DMA, Direction of Minimun Distortion, utiliza una función de costo relacionada con el cálculo del MSE/SAD, la búsqueda se inicia alrededor de cinco posiciones, la central, una a cada lado de ella en dirección vertical y otras dos en dirección horizontal y con un paso de tamaño igual a $\frac{S}{2}$ o a la potencia de 2 más próxima a $\frac{S}{2}$.

En la segunda iteración se determinan tres posiciones más con el mismo paso seleccionado y tomando como posición central la del mínimo resultado de la búsqueda anterior. Este proceso se repite hasta que los mínimos de dos etapas coinciden. En ese momento se divide el tamaño del paso por 2 y se repite el proceso hasta que el paso sea igual a 1. En este momento se busca en los ocho puntos que rodean al central. El mínimo de esta última búsqueda se toma como posición óptima. A continuación, se muestra su algoritmo:

Algoritmo 4 Búsqueda Logarítmica

1: Se realiza una búsqueda en la posición $(0, 0)$ y se define un paso de tamaño $S = N$.
2: Se selecciona 4 posiciones a una distancia S píxeles con respecto al y los ejes x e y.
3: Se determina la posición con el menor error MAE/SAD y se fija como el nuevo origen de la búsqueda.
4: Se calcula si esta posición es la central con respecto a las 5 seleccionadas $S = \frac{S}{2}$.
5: **Si** $(S == 1)$ ir al paso 6, **sino** ir al paso 2.
6: Se selecciona el origen actual, y las 8 posiciones de alrededor según el paso seleccionado, se determina la que de menor valor para el MAE/SAD;

3.2.6. Búsqueda en cruz (CS)

El algoritmo Búsqueda en cruz (CS) se muestra a continuación:

Algoritmo 5 Búsqueda en Cruz

1: Se establece el origen en la posición $(0, 0)$ con un paso de tamaño $S = 2^{N-1}$.
2: Se selecciona 4 posiciones a $\pm S$ píxeles del origen formando una cruz (X) y considerando, también el propio origen.
3: Calcula la posición que ofrece el menor valor para SAE, fijándola como el nuevo origen de la búsqueda.
4: Se calcula si esta posición es la central con respecto a las 5 seleccionadas $S = \frac{S}{2}$.
5: **Si** $((S > 1))$, **entonces** $S = \frac{S}{2}$ y se dirige al punto 2.
6: **Si** la mejor posición está en el punto superior izquierda o inferior derecha de la X, se evalúa 4 puntos más en forma de X a una distancia de ± 1 píxel.
7: **Sino** hacer realizar nuevamente el mismo proceso, pero con los 4 puntos de distribución en "+".

3.2.7. Vecinos más próximos (Nearest Neighbours Search)

El algoritmo Vecinos más próximos (Nearest Neighbours Search), a seguir se muestra a continuación:

Algoritmo 6 Vecino más Próximos

1: Se calcula el SAD del $(0, 0)$.
2: Se establece el origen de búsqueda a la posición del vector supuesto (predicted vector).
3: Seleccionamos 4 posiciones alrededor del origen en forma de "+".
4: Se calcula si esta posición es la central con respecto a las 5 seleccionadas $S = \frac{S}{2}$.
5: **Si** el origen de búsqueda (o la posición 0,0 en la primera iteración) ofrece el menor SAD entonces "fin de búsqueda".
6: **Sino** establece el nuevo origen de búsqueda en la posición que menor valor para SAD ha ofrecido.

3.2.8. Mejora del heurístico de búsqueda en tres pasos simple y eficiente

El diseño del algoritmo heurístico MHTSS mejorado emplea un patrón de búsqueda uniformemente y no tiene en cuenta movimientos pequeños. En la primera iteración, se examina 16 puntos, con el fin de conseguir el punto con menor error, donde los 16 puntos están se encuentra a una distancia alrededor del

punto central, 8 de ellos alrededor y separados una distancia descrita y con un paso de búsqueda $S = 4$ y los puntos restantes están ubicados alrededor del punto inicial con $S = 1$. Luego, se consigue el menor peso entre los 16 puntos comparados. Si el punto con menor costo se halla en los puntos exteriores sobre el área de búsqueda, el algoritmo es similar a la solución que da TSS. Si el punto con menor costo está en los puntos centrales, se efectúa una comparación entre los 3 o 5 puntos alrededor del punto con mínimo costo entre aquellos que no han sido comparados. El heurístico compara en el mejor de los casos 17 posiciones, mientras que examina 33 puntos en el peor de los casos.

3.2.9. Mejora del heurístico en el patrón de búsqueda en cruz

El diseño del heurístico $MHACS$ mejorado parte del hecho, de que el movimiento global de un frame con respecto los demás son similares, es decir, que si los macro bloques que hay alrededor del macro bloque actual se han desplazado en una dirección en especifico, entonces existe una probabilidad muy alta de que el macro bloque actual también se haya movido en esa dirección.

El heurístico utiliza el vector de movimiento del macro bloque ubicado a la izquierda del actual para predecir el vector de movimiento. Una vez cotejado el vector de movimiento del macro bloque de la izquierda, se evalúa la función de costo empleando un patrón en cruz cercanos al punto central. Los 4 puntos que no forman parte centro se determinan a una distancia de este usando un paso $S = Max(|x|, |y|)$, siendo x e y las coordenadas del vector de movimiento del macro bloque situado a la izquierda y más próximo a el. Sobre el patrón que ha definido se evalúa la función de costo para cada uno de los puntos y se elige aquel punto que proporcione un menor valor. Posteriormente, se aplica iterativamente el patrón $SDSP$ empleado el algoritmo DS y este termina de ejecutarse hasta que la función de costo dé valor mínimo para un píxel central de la muestra o patrón prefijado.

Capítulo 4

METODOLOGÍA APLICADA

Para alcanzar los propósitos de la tesis se uso instancias o bases de datos de imágenes disponible en internet y que son de interés para la comunidad científica para la comprensión de secuencias de imágenes y tratamiento de vídeos en general, tratando de encontrar algunas soluciones exactas, y/o aproximadas utilizando heurísticos y sus mejoras para las soluciones presentadas por otros investigadores, para obtener primero la madurez necesaria del problema y de los subproblemas y elaborar, diseñar e implementar los algoritmos.

Se procedió a diseñar los diferentes algoritmos, los de dominio de frecuencia que se basan sobre los coeficientes de las transformadas: **la transformada Discreta del Coseno y la Wavelets**, se eligieron estos ya que nos sirven de base y son los que mejor se adaptan al tratamiento de los **algoritmos de dominio temporal (búsqueda de la máxima coincidencias)** y que dan respuesta al problema en estudios, diseñándose también los siguientes heurísticos: el algoritmo de búsqueda exhaustiva, algoritmo de búsqueda en tres pasos, algoritmo de búsqueda en cuatros pasos, algoritmo de búsqueda en diamante, algoritmo y los heurísticos mejorados: búsqueda en tres pasos simple y eficiente, algoritmo de búsqueda de patrones en tres pasos.

El diseño metodológico de este trabajo se aprecia en la siguiente figura:

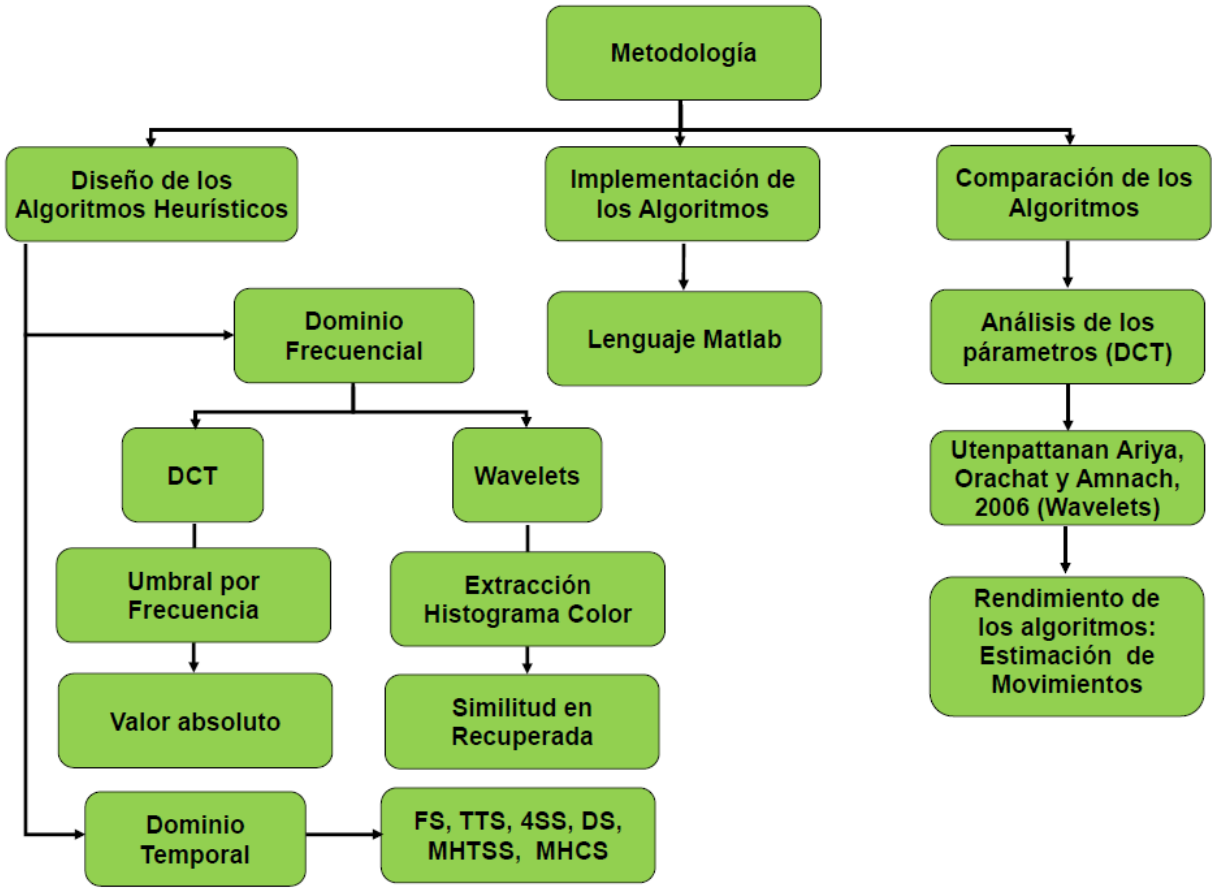


Figura 4.1: Esquema de la Metodología utilizada

Luego, se implementó cada uno de estos algoritmos en el lenguaje de programación $MATLAB$. Por último, comparó los resultados con los diferentes algoritmos diseñados e implementados, se procedió a comparar los resultados con otros algoritmos propuesto por la comunidad científica que son más robusto y eficiente, cuya complejidad computacional son buenas.

Los métodos empleados en la investigación son: **Los métodos empíricos generales y puro**, puesto que la idea es partir de algoritmos de búsqueda completa, heurísticos ya conocidos por la sociedad científica, aplicarlos a distintas instancias del problema de compresión de imágenes estáticas, dinámicas y la estimación de movimientos. Además, se varia los parámetros de los modelos en las técnica de los heurísticos, realizándose híbridos de estos.

También, se utilizó **el método de análisis síntesis** puesto que particionó un problema complejo en

otros más pequeños, después combinar los resultado de algoritmos obtenidos con el fin de equiparar o mejorar los resultados obtenidos, además abstracción y concreción puesto que hay que demostrar y comprobar cada uno de los resultados algorítmicos sean válidos. Debe de señalarse y dejar bien claro que en está investigación el más utilizado es el **método algorítmico**, puesto que se ha de implementar algoritmos heurísticos de búsqueda en el lenguaje de programación de alto nivel y trabajar con diferentes experimentos de laboratorio computacional para aprovechar las ventajas de los mismos.

4.1. Modalidad de la Investigación

Debido a la forma de la manipulación el objeto de análisis de esta investigación, se clasifica como **una investigación aplicada exploratoria**. Puesto que, se justifica por la búsqueda e implementación de técnica de compresión de imágenes, aplicación de heurísticas en la búsqueda de la información y de los algoritmos de coincidencia de bloque para la estimación de movimientos, que ya se ha mencionado y con una finalidad hacia un problema práctico y relativamente soluble en el espacio de soluciones. Por tal razón, dejamos a un lado, la creación de teoría de nuevos algoritmos y tratar obtener las soluciones optimas, es decir la solución exacta del problema en estudio. Sin embargo, el hecho de construir un algoritmo sencillo o bien una implementación de las misma nos garantiza un valiosos aportes al conocimiento teórico - aplicado, y sobre todo en Nicaragua que es un área inexplorada y con mucho camino que recorrer.

4.2. Herramienta computacional utilizada

Para las diferentes ejecuciones de las técnicas y algoritmos implementados se necesitó del uso adecuado de la herramienta $MATLAB$ versión $R2015b$ y su biblioteca de visión artificial para la implementación de los algoritmos elegidos para el procesamiento de imágenes o secuencias, la estimación de movimientos aplicando las transformada discreta del coseno y wavelets, y los heurísticos.

4.3. Cronograma de la Investigación

El cronograma seguido a lo largo de la investigación se aprecia a continuación:

Fecha	Actividad a realizar	Responsables	Observación
Enero a Diciembre, 2016	Desarrollo de tesis doctoral y diseño de Algoritmos Heurísticos de Coincidencia Para la Estimación de Movimiento en Compresión de Imágenes	Fernando José Hernández Gómez y Dr. Antonio Parajón.	Tutorías del Dr. Antonio Parajón.
Enero, 2017	Elaboración del protocolo de tesis doctoral	Fernando José Hernández Gómez y Dr. Antonio Parajón.	Tutorías del Dr. Antonio Parajón. Taller de tesis #1
Marzo, 2017	Presentación de protocolo de tesis doctoral	Fernando José Hernández Gómez	Taller de tesis #2
Abril, 2017	Entrega del Protocolo de tesis doctoral	Fernando José Hernández Gómez	Tutorías del Dr. Antonio Parajón. Declaración de intenciones
Mayo, 2017	Defensa del Protocolo de tesis doctoral	Fernando José Hernández Gómez	
Noviembre, 2017	Entrega de tesis doctoral	Fernando José Hernández Gómez	Incluye sugerencias al protocolo
Diciembre, 2017	Defensa pública de tesis doctoral escrita.	Fernando José Hernández Gómez	

Figura 4.2: Cronograma de Investigación

Capítulo 5

IMPLEMENTACIÓN Y RESULTADOS COMPUTACIONALES

Para la implementación de los algoritmos se tenían dos alternativas, la primera es a partir de la teoría, bases matemática y programarlos en su totalidad; la segunda opción es utilizar alguna de las librerías o funciones específica del software Matlab, ya que el desarrollo matemático como lo pudimos notar a lo largo de los capítulos anteriores es muy extenso y la codificación en programas de software nos restaría mucho tiempo, necesario para poder alcanzar los objetivos de la tesis.

Las funciones a utilizadas se implementaron y se codificaron en el software $Matlab$ que tendrá como entrada una variable de tipo matriz que representa a una imagen en niveles de gris (blanco y negro) o a color con sus respectivos niveles. En el caso que se utilice una imagen a color y se necesite en sus niveles de gris, se utilizará la función de $Matlab$ $rgb2gray()$. A esta matriz de entrada, que al ser en niveles de gris será bidimensional, se le calculará la $DCT-2D$ mediante la función $dct2()$, y finalmente se representarán gráficamente sus coeficientes mediante la función $mesh()$ para medr el desempeño cuando no se logra apreciar la calidad visual. Para realizar la selección de coeficientes de la DCT por frecuencia o con base a su valor absoluto, se implementan las funciones dct_ufrec_modif y $dct\_valor\_absoluto\_modif$ que utiliza los percentiles (garantizados por la función prctile) en vez

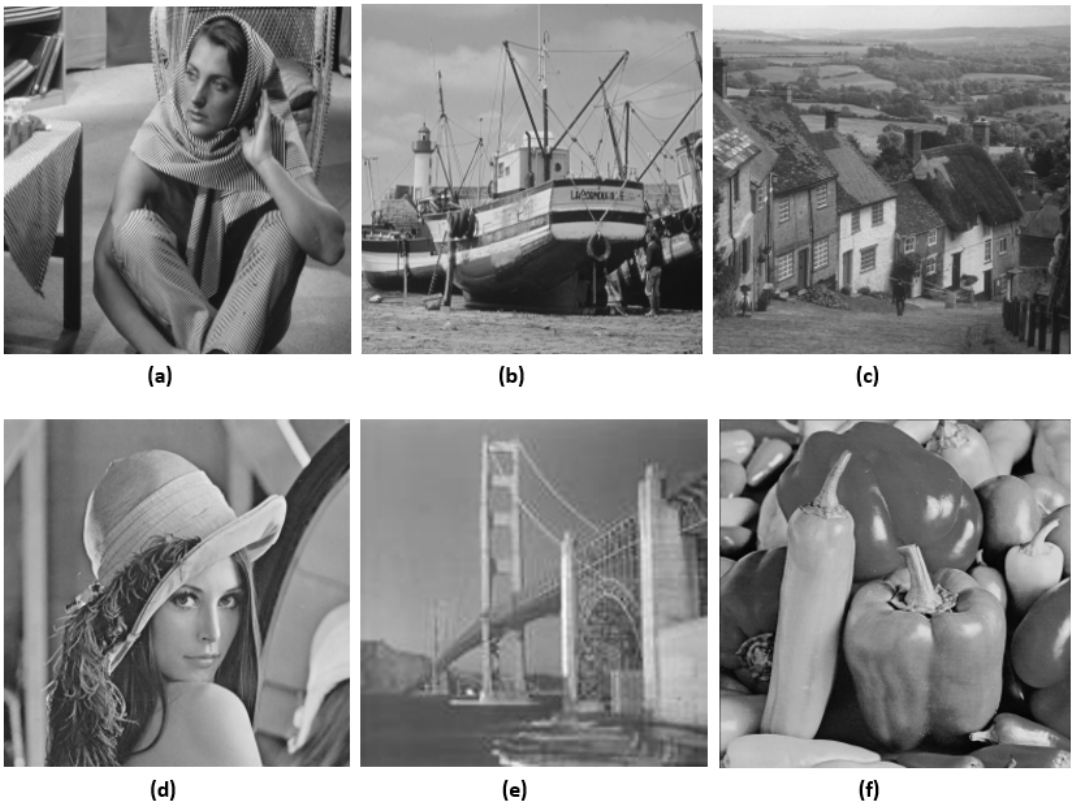
de dividir el el rango de los coeficientes de la DCT en intervalos iguales, puesto que estos no siempre nos garantizan que estemos seleccionando realmente un buen porcentaje de coeficientes de la DCT que deseamos.

Los experimentos realizados para el análisis del desempeño de los algoritmos descritos en los capítulos 2, sección 2.4 consistieron en codificar un conjunto de secuencias de imágenes de prueba, con ciertas características particulares. Para realizar las mediciones de compresión y de calidad de recuperación, la búsqueda del número de puntos y error en la estimación de movimientos aplicando los algoritmos de dominio de frecuencias, como la transformada discreta del coseno, $Wavelet$ y los algoritmos de dominio temporal tales como, los algoritmos de estimación de movimientos basados en la coincidencia máxima por bloque y recursividad. Las imágenes se transfirieron en tiempo real y se almacenaron en la computadora para su posterior análisis. En este capítulo, se describe las condiciones de los experimentos, se presentan y comentan los resultados obtenidas tras la ejecución de los diferentes algoritmos desarrollados e implementados.

5.1. Secuencia de prueba

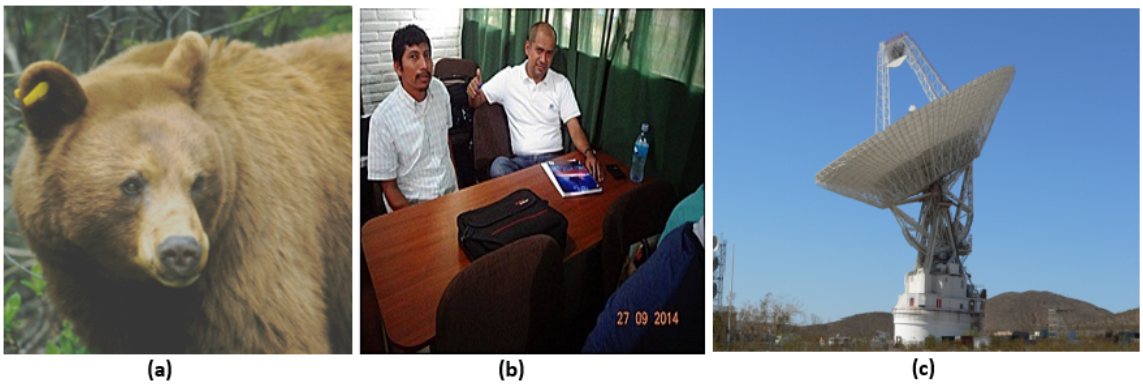
En esta sección se hará mención y se describe las características principales de las secuencias de imágenes utilizadas en la parte experimental. Esta se divide en dos categorías, según las características, cantidad de información y movimientos que presentan la imágenes analizadas. El primer grupo, lo forma las imágenes en escala gris *amber, boat, barbara, goldhill, lena, mandril, peppers, puente y soldados franceses* con un bajo contenido de información y por su alto grado de calidad visual, el segundo grupo los conforman *Satelite, efectoImagen* y tres imágenes propias *tomate, PhD, FerArg* con un alto grado de contenido de información, degradación y de movimientos de las mismas. Las características de las secuencias forman la base para realizar las debidas interpretaciones de los resultados, en conjunto de las características específicas de los métodos tales como, DCT, $Wavelets$ y de estimación de movimientos. Las secuencias originales y los resultados se anexan en un disco compacto a este documento. En las

siguientes figuras, se pueden observar algunas de las secuencia de imágenes utilizadas en escala gris:



a) Bárbara, b) boat, c) goldhill, d) Lena, e) puente y g) peppers.

Así como, la secuencia de algunas imágenes a color utilizadas, tal como se muestra a continuación:



Bear, b) FerPhd, c) satelite.

5.2. Potencial de compresión de la transformada discreta del coseno DCT

La idea central de este apartado es comprobar en la práctica algunas de las propiedades de la Transformada Discreta del Coseno (DCT) y realizar un análisis de los pasos fundamentales del compresor $JPEG$ y $MPEG$, que hacen posible conseguir la compresión subjetiva sin pérdidas en secuencias de imágenes estáticas, secuencias de imágenes y vídeo.

Se analizará la importancia de seleccionar los diferentes coeficientes de la DCT, la diferencia de seleccionar los coeficientes de la transformada en base a su frecuencia y en base a su valor absoluto, así como el tamaño de bloque a elegir; con la finalidad de eliminar el ruido subyacente, obteniéndose de esta manera una **compresión intracuadro**.

Se hará uso del dominio transformado en vez del dominio espacio-temporal, puesto que multiplica los factores de compresión, afectando únicamente la forma ligera a la calidad de la imagen obtenida. La transformación utilizada en el tratamiento de imágenes estáticas es la transformada discreta del coseno (DCT); no por ser la única, ni la mejor, sino por el simple hecho de que los métodos $MPEG - JPEG$ de compresión, están basados e implementados con esta transformada y que han sido aceptados ampliamente por todo el campo científico.

Los algoritmos de compresión son bastante complejos de desarrollar. No obstante, en la práctica se han elegido, de todos los aspectos involucrados en la compresión, aquéllos que son más representativos y cuya dificultad no es excesiva, de forma que el desarrollo de ésta no lleve más tiempo del disponible para su implementación y ejecución del mismo. Se analizará con detalles los siguientes aspectos:

- La distribución de la información, en el dominio de la DCT, de una imagen. Se comprobará que la mayor parte de la información reside en los coeficientes de baja frecuencia.
- Iniciando en el terreno de la compresión, veremos que la eliminación de gran parte de los coeficientes de la DCT en la selección del tamaño bloque 8×8 no degrada en exceso la calidad de las

imágenes.

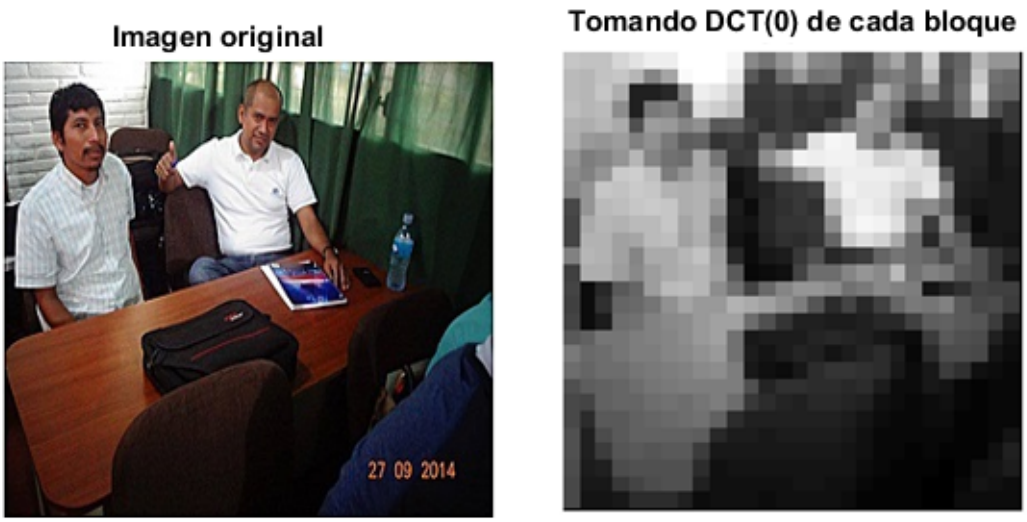
- También, se observará que la cuantificación del coeficiente DC sí puede afectar drásticamente calidad de la imagen a analizar, mediante denominado efectos **bloque** y **sombra**.

- Por último, se utilizará una matriz de cuantificación real, cuya matriz de cuantificación es la empleada por defecto por el método $JPEG$ y y $MPEG$ para cuantificar los coeficientes de la DCT y se analizaran los resultados.

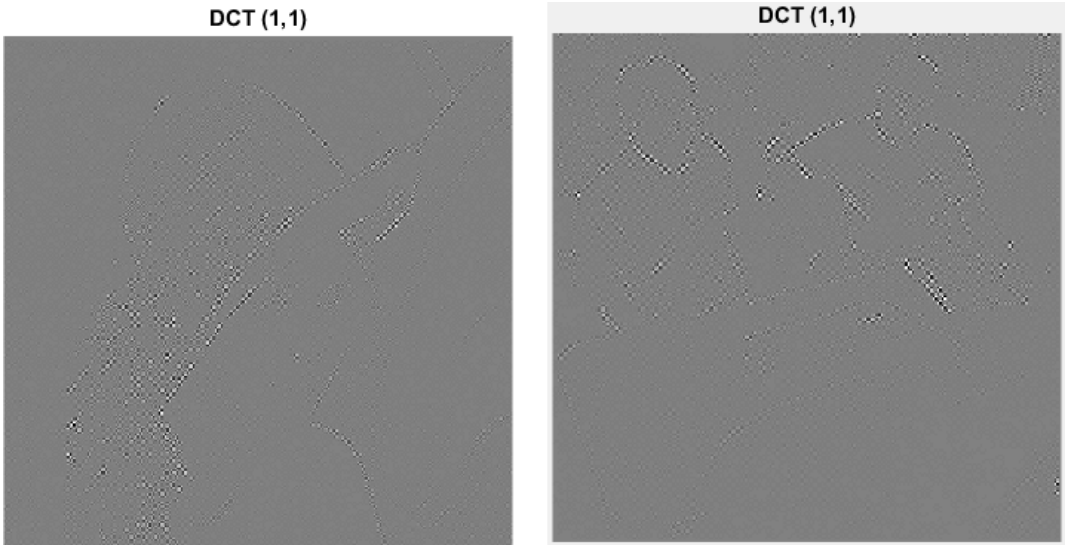
5.2.1. Distribución de la información en la DCT y Artifacts

Al aplicar esta transformada una imagen estática, la mayor parte de la información se concentra en los coeficientes que representan a las frecuencias más bajas, las que corresponden a la esquina superior izquierda de cada bloque o macrobloque analizado. A continuación, se presenta este hecho con algunas imágenes reconstruida tomando únicamente el coeficiente $D(0,0)$ para imagen Lena :

También, se muestra el mismo experimento con la siguiente imagen, en el que se puede observar los mismos resultados



En la siguiente figura nos muestra la reconstrucción de las imágenes tomando únicamente el coeficiente $D(1,1)$ de cada uno de los bloques:



Nuestros experimentos para este análisis y los siguientes consistieron en:

- Elegir una imagen completa,
- aplicar la transformada discreta del coseno DCT sobre toda la imagen,
- seleccionar el mismo porcentaje de coeficientes de la DTC en base *a su frecuencia y en base a su valor absoluto*, y con el mismo tamaño de bloque.
- reconstruir la imagen, y comparar la calidad de la imagen actual con la de referencia, permitiéndonos comparar para la misma tasa de compresión qué método proporciona mejor calidad visual

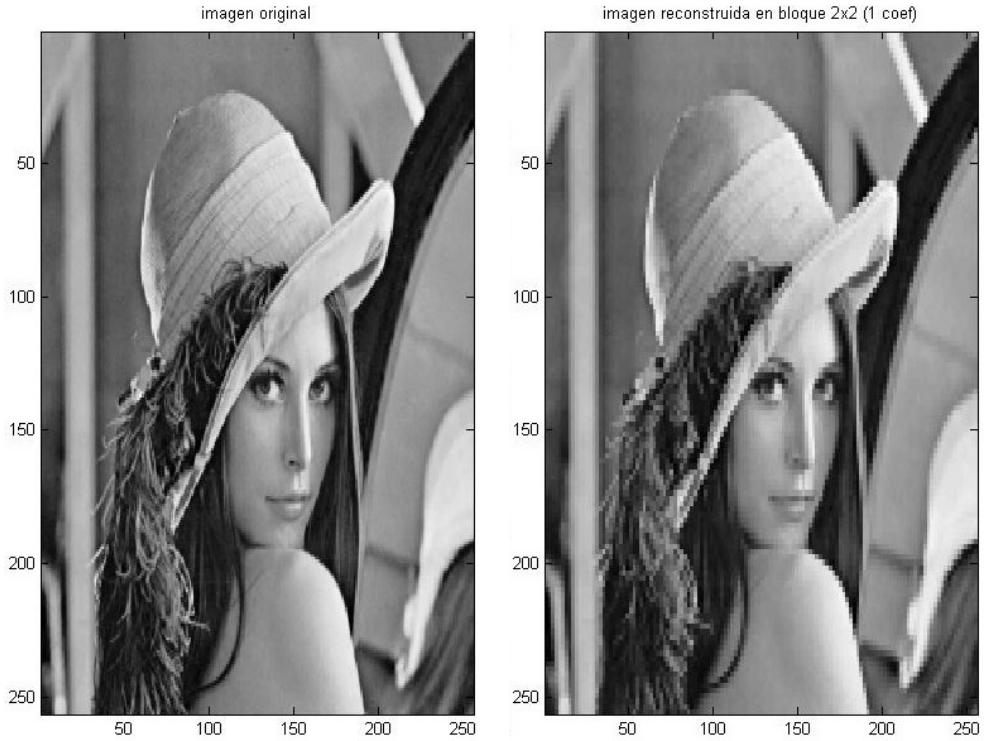
empleando las medidas de comprensión, tales como la entropía H y $PSNR$.

Aunque las dos imágenes reconstruidas presentan un halo blanco alrededor de los bordes de la chaqueta, este es más notorio en la imagen reconstruida eligiendo los coeficientes por frecuencia.

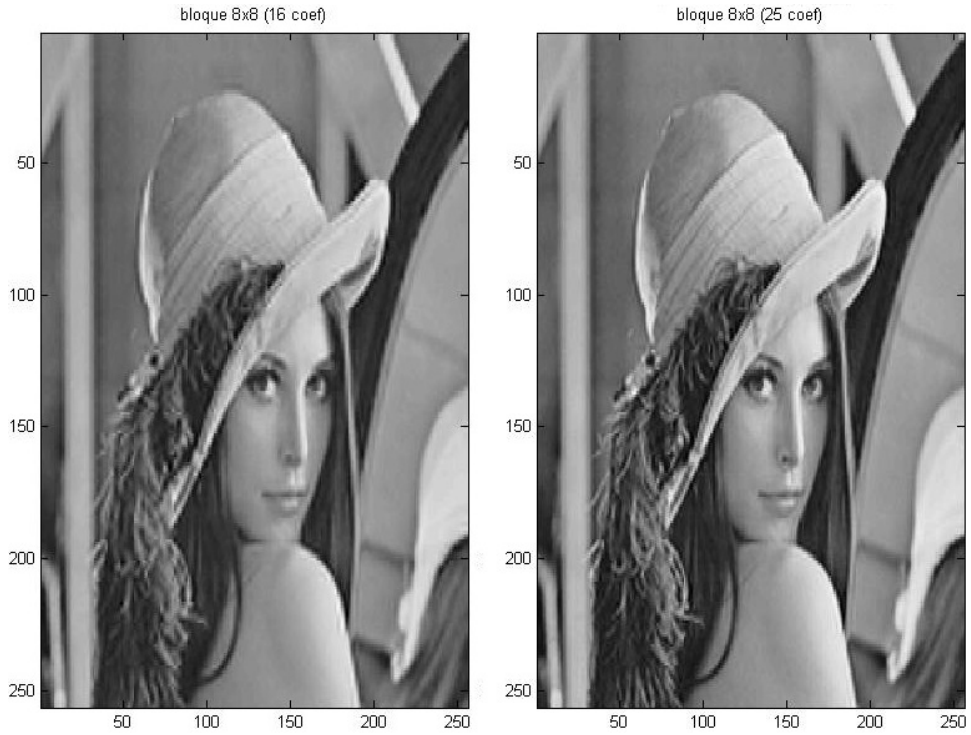
Podemos observar en general, que al aplicar este proceso a las diferentes imágenes de prueba el artifacts o efecto sombra aparece de manera mas notoria en las imágenes reconstruidas eligiendo los coeficientes por frecuencia. También, mostramos este hecho en la siguiente figura en donde la reconstrucción es muy buena en ambos casos y muy similar, pero se sigue detectando el halo (efecto sombra) al eligir los coeficientes por frecuencia:

Además, notamos el efecto bloque o cuadro una diferencia bien pronunciada que se origina al dividir la imagen en bloques de diferentes tamaños, al aplicar la la transforma a cada bloque, eligiendo el mismo porcentaje de coeficientes de la DCT en ambos casos en reconstrucción de la imagen. Por otro lado, detectamos que tanto al dividir la imagen en bloques de tamaño 2×2 como al dividirla en bloques de 8×8, al aplicar la DCT a cada bloque y reconstruir la imagen, la misma aparece pixelada, sobre toda la imagen si los bloques considerados son de 2×2, independientemente de que si aplicamos un cuantizador

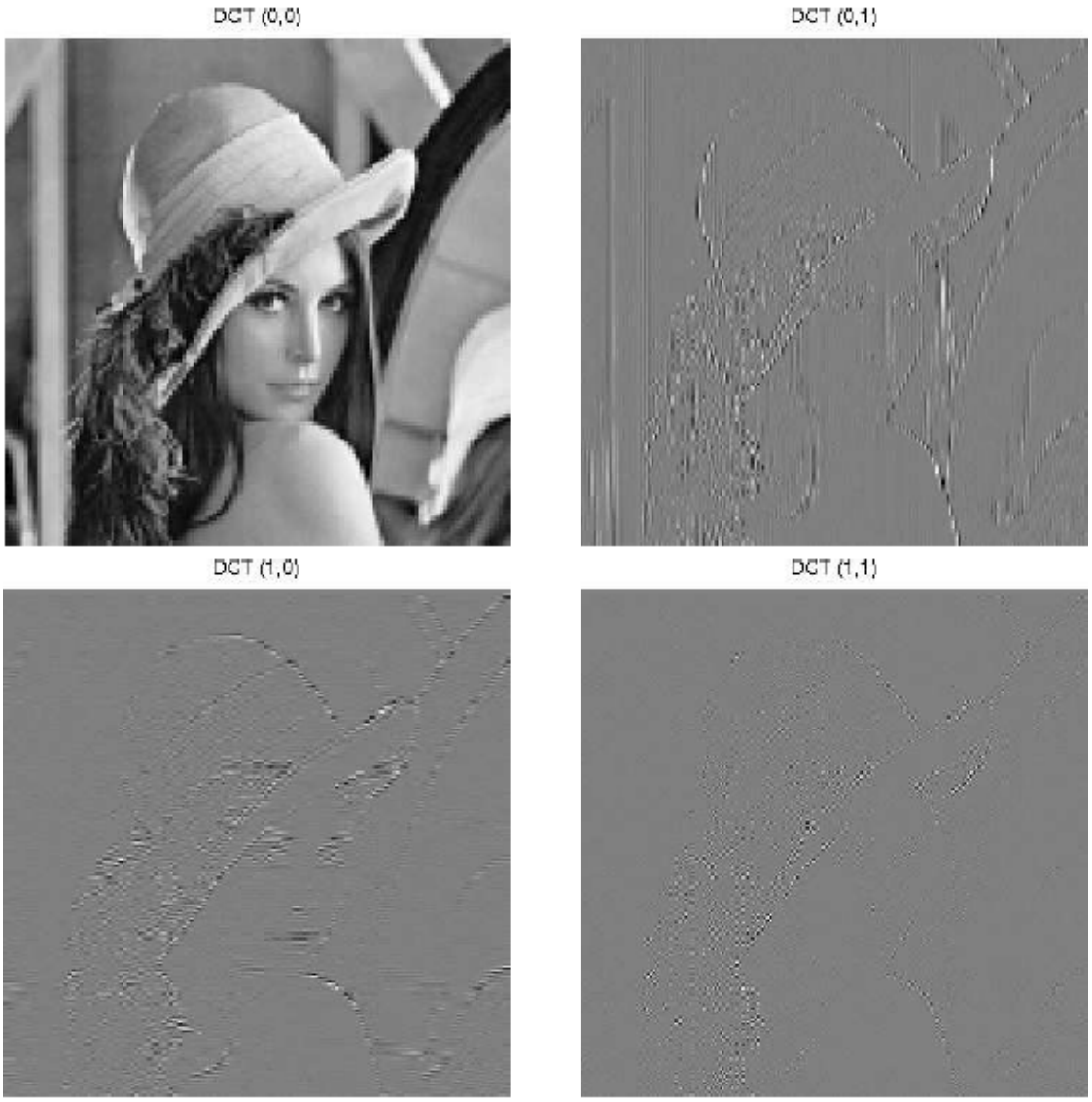
uniforme a toda la imagen, si seleccionamos coeficientes en base a su frecuencia o en base a su valor absoluto. Por ejemplo, presentamos la imagen de Lena reconstruida tomando únicamente $DCT(0,0)$ de cada bloque de tamaño 2×2 y se considera únicamente el primer coeficiente al que corresponde $DCT(0,0)$ de cada bloque, vemos pronunciado el efecto bloque si realizamos un zoom sobre un área en específico, como se muestra a continuación.



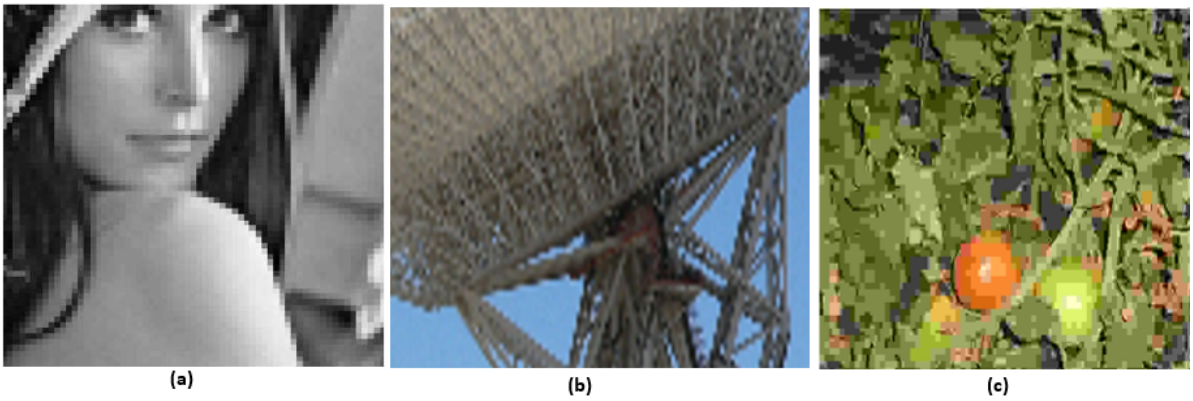
Cabe señalar que al tomar el $16\,\%$ y $25\,\%$ de los coeficientes de la DCT se sigue apreciando este mismo artifact o efecto al reconstruir la imagen, pero este hecho sólo se logra apreciar si observamos fijamente la imagen. Presentamos la imagen de Lena reconstruida tomando de cada bloque de 8×8 los coeficientes $DCT(0,0)$ y $DCT(0,1)$. Como se muestra a continuación:



Creemos oportuno aclarar aquí, que no detectamos este artifacts cuando tomamos el 75 % de los coeficientes de la DCT. Además, que en todos nuestros experimentos para considerar el 25 % de los coeficientes trabajamos con D(0,0) de cada bloque, para considerar el 50 % de los coeficientes consideramos D(0,0), y D(1,0) o D(0,1) obteniendo resultados similares y para trabajar con el 75 % de los datos D(0,0), D(1,0) y D(0,1) de cada bloque debido a que D(0,0) es el que más aporta para la reconstrucción seguido de D(1,0) y D(0,1) que aportan de manera similar a la reconstrucción y por último D(1,1) que es el coeficiente que menos aporta a la reconstrucción. Por ejemplo, estos fueron los resultados que obtuvimos al reconstruir la imagen de Lena tomando únicamente un coeficiente de cada bloque al reconstruir la imagen:



Si en vez de tomar bloques de tamaño 2×2 seleccionamos bloques de 8×8 la imagen no se pixela tanto, y no se observa a simple vista por el ojo humano. Por ejemplo, presentamos la imagen de Lena, satélite y tomate reconstruida tomando el 25 % de los coeficientes de la DCT, lo que equivale a tomar únicamente coeficiente D(0,0) de cada bloque de de tamaño 2×2 y tomar $DCT(i,j)\ \forall\ 1 \leq i,j \leq 3$ en cada bloque de 8×8. Ampliamos un determinada área de las imágenes para que se aprecie mejor la diferencia de pixelado de las imágenes en estudio, lo cual se muestra a continuación:



Lena, b) satelite, c) tomate.

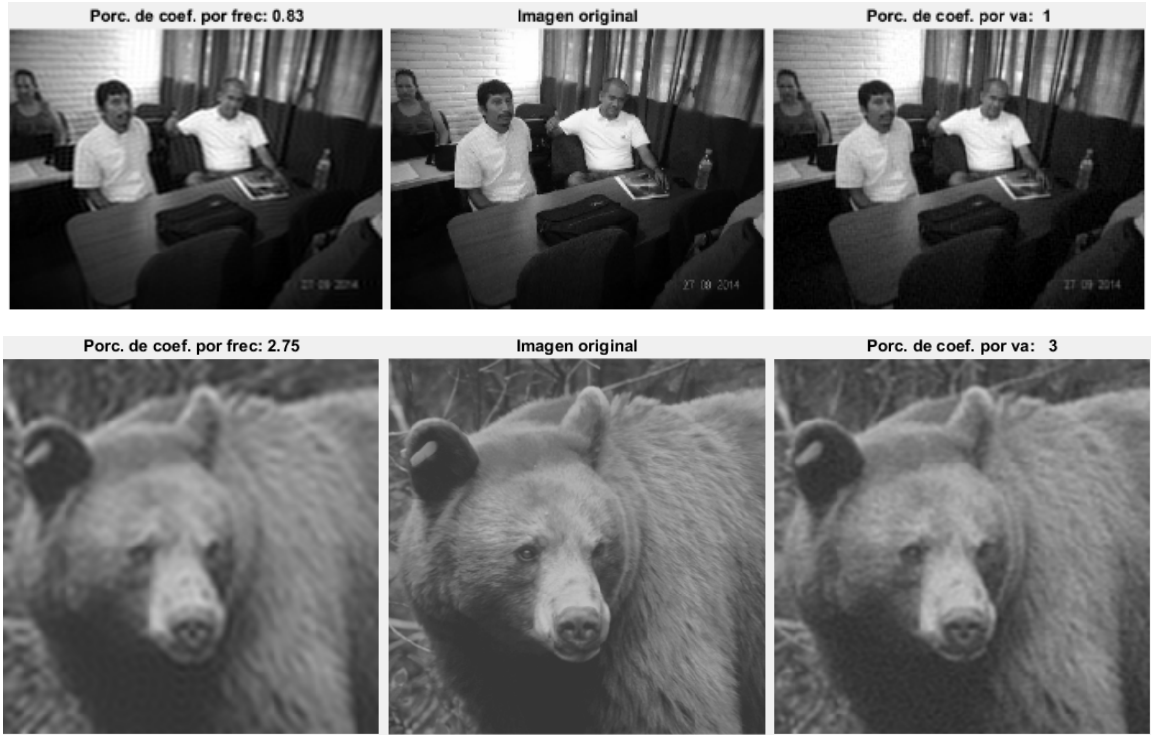
5.2.2. Calidad visual, casos bien y mal condicionados

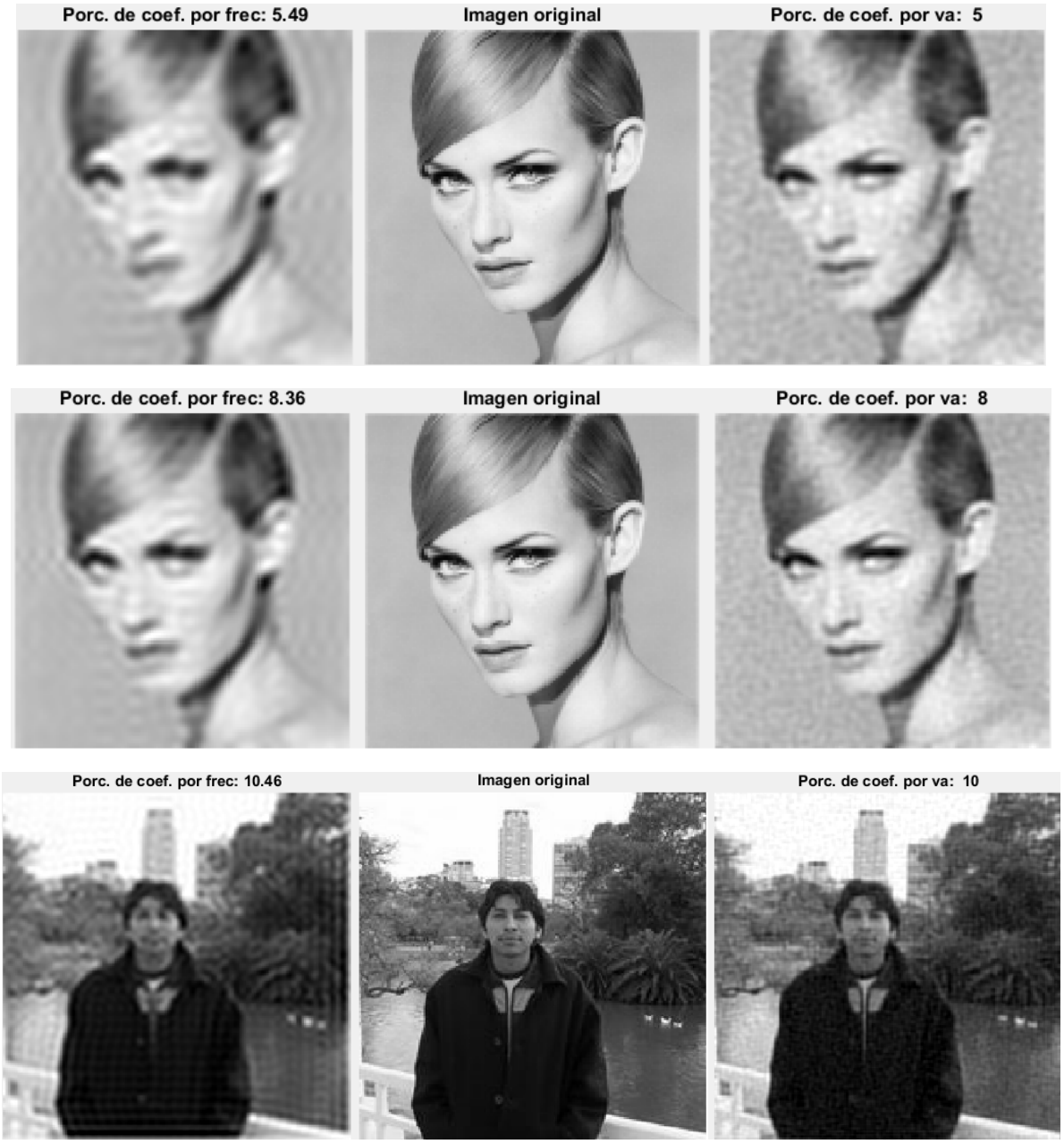
En los diferentes experimentos realizados y analizados obtuvimos que, de acuerdo a nuestra óptica, la calidad visual de la imagen reconstruida seleccionando los coeficientes de la transformada discreta del coseno DCT con base a su valor absoluto es al menos igual que la calidad visual de la reconstruida seleccionando los coeficientes de la DCT en base a su frecuencia.

Antes de realizar las pruebas conjeturamos que **esta diferencia sería significativa en imágenes que presenten altas frecuencias, puesto que en la selección de los coeficientes por frecuencia éstas se descartarían y ocasionarían que la imagen reconstruida pierda esos detalles**. Después de las diferentes corridas o ejecución de los algoritmos obtuvimos que para imágenes que presentan zonas de alta frecuencia esas zonas son las últimas en quedar bien reconstruidas utilizando la selección de coeficientes por frecuencia.

Por ejemplo, en la imagen $amber.png$ tomando aproximadamente 22 porciento de los coeficientes de la DCT obtenemos que ambas reconstrucciones son muy parecidas salvo porque en la que se eligen por frecuencia la zona de los labios, los ojos y el contorno del cabello presenta altas frecuencias, no es bien reconstruidas. Este hecho se puede observar a continuación:

Además, obtuvimos grandes diferencias donde no habíamos esperado hallar, puesto que eran imágenes que no presentan altas frecuencias. Esta diferencia es muy significativa sobre todo cuando el porcentaje de coeficientes tomados para la reconstrucción es bajo, es decir se toma por coeficientes de la transformada discreta del coseno. A continuación, presentamos un experimento que consistió en seleccionar diferentes imágenes(FerPhd con aproximadamente 1 % , bear aproximadamente 3 %, Amber aproximadamente 5 % y 8 % y FerBA con aproximadamente 10 % de los coeficientes seleccionados), en la mayoría de los casos obtuvimos la misma conclusión con respecto esta diferencias tomando un porcentaje bajo de coeficientes:

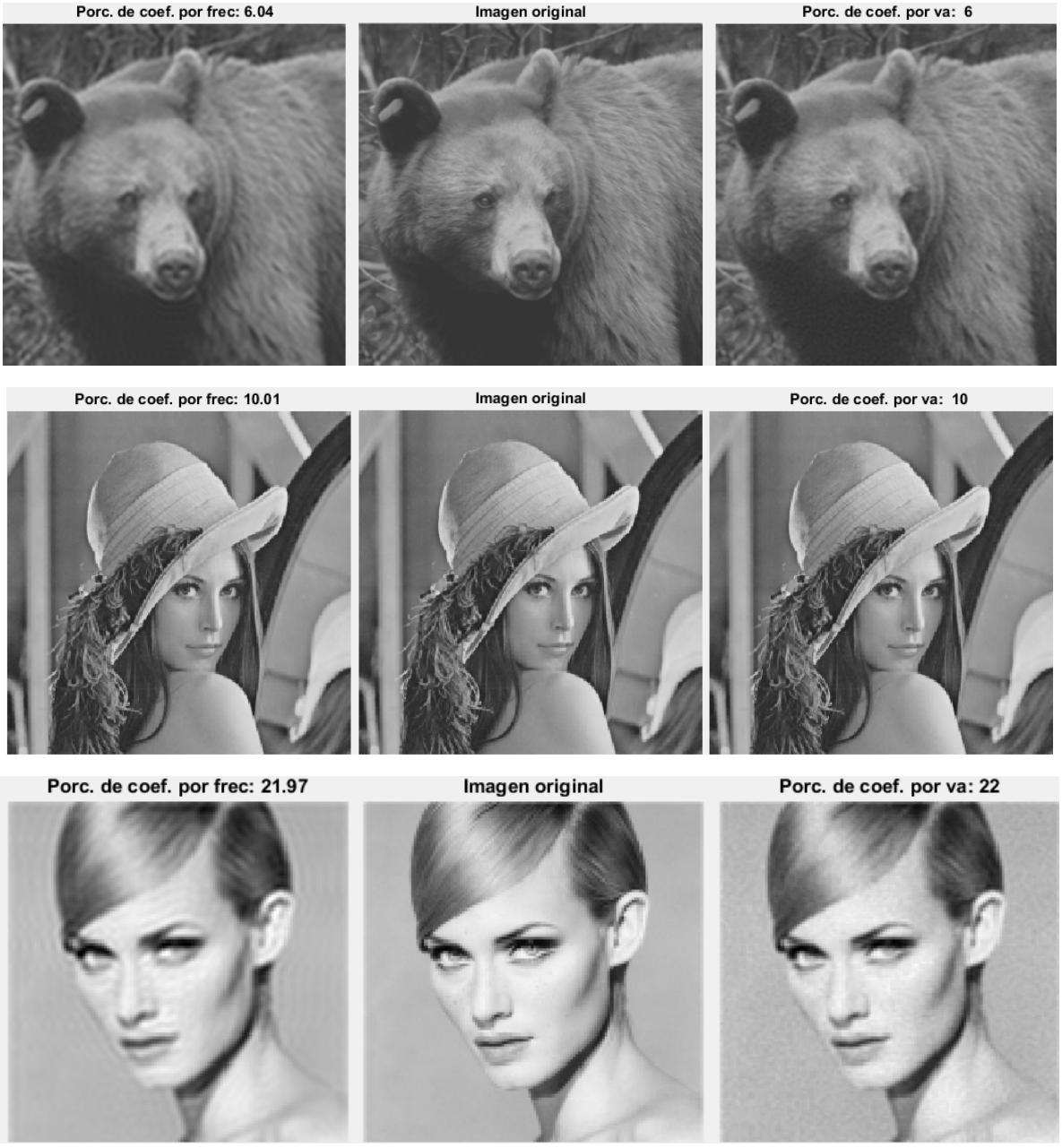




Este hecho, se debe a que la función implementada por valor absoluto elige los coeficientes de la DCT más representativos desde un principio, mientras que la función implementada por frecuencia toma los coeficientes de más baja frecuencia que no siempre son los más significativos, incluso cuando las frecuencias de la imagen no son altas.

Siendo una de las ventajas o superioridad, no sólo la observamos cuando se toma un porcentaje bajo de coeficientes sino también cuando buscamos que la reconstrucción sea muy buena, es decir, que visualmente no podamos distinguir entre la imagen original y la reconstruida. En todo los casos analizados obtuvimos que no podíamos notar diferencias entre la imagen original y la reconstruida por

frecuencia. Esta conclusión se puede notar en las imágenes (bear con aproximadamente 6 % , Lena aproximadamente 10 %, y Amber aproximadamente 22 % de los coeficientes seleccionados), como se muestra a continuación:



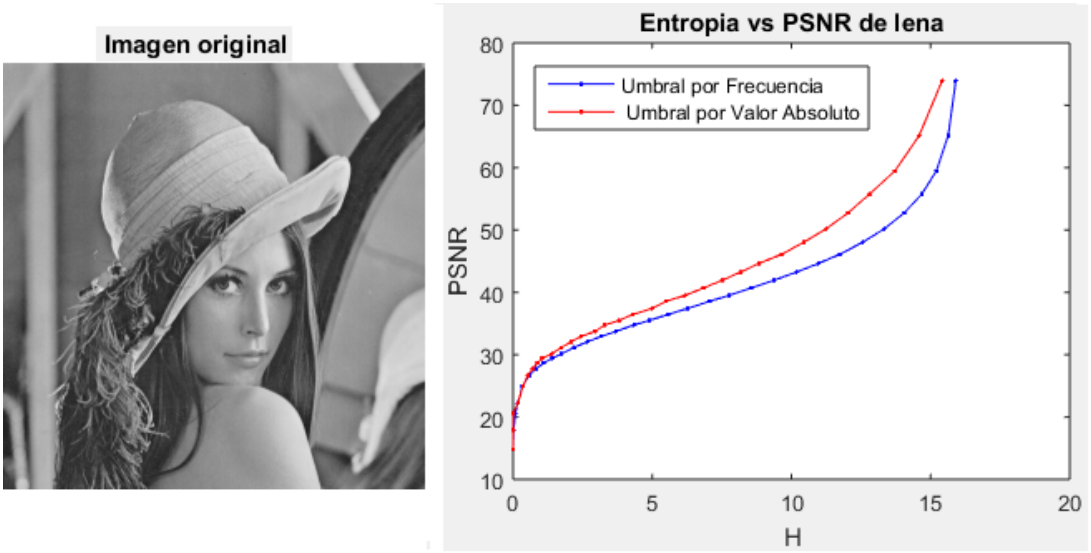
5.2.3. Compresión alcanzada de la transformada discreta del coseno DCT

Todo lo analizado en los ítems anteriores nos permite intuir que la compresión alcanzada utilizando la selección por valor absoluto sería mayor o igual que la alcanzada utilizando la selección por frecuencia, ya que, podemos considerar que todos los coeficientes de la DCT son del mismo orden (quizás el

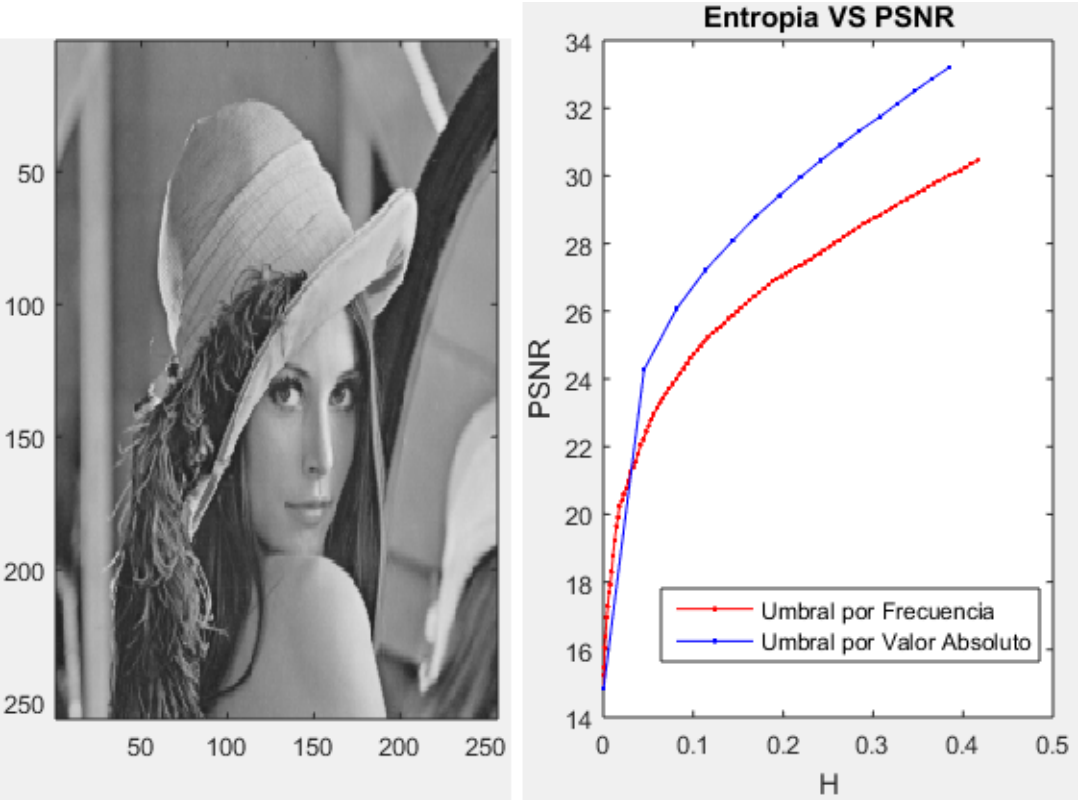
único coeficiente de mayor valor respecto a los demás es el primer coeficiente $D(0,0)$, es una de la razones principales por la que debe de ser incluido en cualquier selección que se realice, ya sea, tanto por frecuencia como por valor absoluto).

Por tanto, es **esperable** que cueste lo mismo guardarlos (no parece ser probable que los coeficientes elegidos por frecuencias puedan tener una codificación diferente a los elegidos por valor absoluto como por ejemplo una codificación diferencial); y cómo afirmamos que para el mismo porcentaje de coeficientes (lo que es equivalente a afirmar para la misma cantidad de coeficientes) obtenemos una mejor calidad visual utilizando la selección por valor absoluto, tendremos una mayor calidad visual para la misma tasa de compresión y los valores obtenidos del $PSNR$.

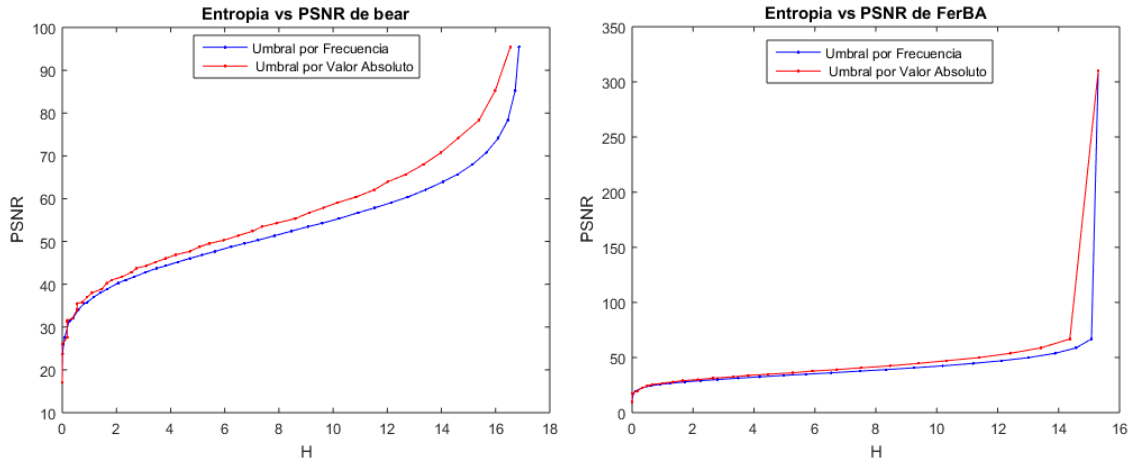
Podemos fundamentar esta intuición tomando buenas medidas de desempeño de calidad de la imagen y de comprensión. Presentamos los gráficos realizados para tres imágenes de diferentes tamaños. La imagen de Lena de 256×256, bear de 350×350 y una imagen propia de 350×350. Como se muestra a continuación:



En todos los gráficos se puede observar que el valor correspondiente al $PSNR$ de la imagen reconstruida utilizando la selección de los coeficientes de la DCT mediante valor absoluto es casi siempre mayor o está por encima que la utilizada por selección por frecuencia, para los mismos valores de entropía, salvo en los valores muy cercanos a cero (como se muestra en la gráfica de lena en los valores entre 0 y 0.1) y en el valor mayor de la entropía. Es decir, como se ve en la siguiente figura:



En los valores muy cercanos al cero lo que sucede es que se interpola con más valores diferentes de H (entropía) en la curva que representa la selección por frecuencia. Esto de debe a que **discretizamos** más los valores de H y del $PSNR$ para la selección por valor absoluto *(al utilizar la función round obtuvimos que el porcentaje de coeficientes elegidos era un número entero, por tanto obtuvimos 100 valores diferentes de H y de PSNR)*. Para el valor más alto del $PSNR$ (y de H) las curvas son iguales, dado que en ese punto los métodos seleccionan el 100 % de los coeficientes. Estos hechos se muestran a continuación:



5.2.4. Transmisión progresiva transformada discreta del coseno DCT

Para el análisis de este apartado anexamos seis archivos $.m$, es la extesión, que al ejecutar las implementaciones de los algoritmos producen vídeos. Estos archivos son: *TransProgresivAmberSurfc.m, TransProgresivbear.m, TransProgresivFerRudy.m, TransProgresivLena.m, TransProgresivMandril.m y TransProgresivPuente.m.*

En los primeros cinco experimentos logramos observar como la imagen se reconstruye mejor a medida que el porcentaje de coeficientes de la DCT aumenta. Como ya mencionamos, la reconstrucción es más rápida seleccionando los coeficientes de la DCT por valor absoluto, pero en ambos casos lo que observamos es que la imagen va siendo reconstruida desde los detalles más gruesos a los detalles más finos, es decir, al principio vamos obteniendo como un bosquejo de la imagen y a medida que se agregan coeficientes de la DCT van apareciendo los detalles, como se ve sobretodo en *TransProgresivMandril.m, TransProgresivLena.m* en donde los detalles más finos que son los pelos del bigote del mandril, así como las plumas del sombrero de Lena aparecen como últimos detalles en la reconstrucción de la imagen.

La justificación de esto es directa en la reconstrucción por frecuencia, ya que elegimos de las frecuencias más bajas a las frecuencias más altas y precisamente siendo los detalles más finos en la reconstrucción son aportados por las frecuencias más altas.

Otro punto de interés que observamos es que a partir de un porcentaje no tan alto de coeficientes los cambios en la reconstrucción de la imagen son imperceptibles a nuestro ojo debido a que a partir de cierto porcentaje no muy alto de coeficientes la imagen reconstruida nos resulta igual a la original. Sin embargo la imagen se va modificando aunque nuestro ojo no sea capaz de capturar estos detalles. Esto podemos visualizarlo con la función $surfc$ de $Matlab$ que gráfica la relación entre cada píxel de la imagen y su valor de gris. Por ejemplo, proponemos ver el vídeo generado por el archivo *TransProgresivAmberSurfc.m*. Al observar la reconstrucción de la imagen vemos que a partir de cierto momento nuestro ojo no capta diferencias entre las imágenes reconstruidas y la original, la *función surfc* permite observar que sigue habiendo cambios. Además, se pudo apreciar que al tomar un área de interés y am-

pliar la imagen(zoom), los detalles se observan a gran escala; pronunciándose en gran medida el efecto bloque según el tamaño de bloques elegidos. Así como, el efecto sombra.

5.3. Recuperación de Imágenes de Color en el Dominio de la Wavelet

En este apartado expondremos una sinopsis de los aspectos teóricos y conceptos básicos del artículo [53] desarrollados por (Utenpattanan Ariya, Orachat y Amnach, 2006), que serán las base para experimentación. Además, mostraremos la aplicación de esta técnica en una implementación específica, de este mismo artículo científico y se analizan los resultados obtenidos.

5.3.1. Introducción

Para comparar un par de imágenes o una serie de ellas, es necesario extraer una firma basada en sus píxeles y definir una regla para compararlas.

La firma en el proceso de compresión puede ser el color, la textura, la forma o cualquier otra información que permita llevar a cabo la comparación. Para la realización de nuestros experimentos haremos uso del histograma.

Un histograma de una imagen digital nos indica cómo están distribuidos los píxeles, mostrando la cantidad de píxeles en cada nivel de intensidad del color. Este nos muestra los detalles de las sombras en la parte izquierda, los tonos medios en el centro y las iluminaciones en la parte derecha, el histograma nos ayuda a determinar si la imagen contiene suficientes detalles para realizar una corrección correcta. Además, el histograma de color de una imagen se construye a partir del recuento de píxeles de cada color. La ventaja de aplicar el histograma, se debe a que es la forma más sencilla de describir sus propiedades o características de bajo nivel, asociadas al color de las imágenes y otra de las razones es que su cálculo es trivial. Sin embargo, una de las dificultades que presenta es que en general no son apropiado para la

realización de algunas tareas específicas en imágenes, tales como:

1. La ***búsqueda, filtrado y recuperación***, puesto que necesita un gran número de bits para representar la información.

2. ***Alta dimensionalidad,*** es decir la imagen puede ocupar más de 100 dimensiones.

3. ***Complejidad en el cómputo*** de la distancia entre histogramas.

4. Insensible a ***rotaciones, zoom*** y cambios de ***resolución***.

5. Sensible a cambios de ***iluminación***.

Existen diversas formas de medir la distancia o métricas entre dos histogramas de color, tales como: la distancia euclídea, la distancia de intersección, la distancia cuadrática y la distancia de Hamming, entre otras técnicas.

El algoritmo empleado para calcular el descriptor binario de Haar, consiste una serie de pasos que se mencionan a continuación:

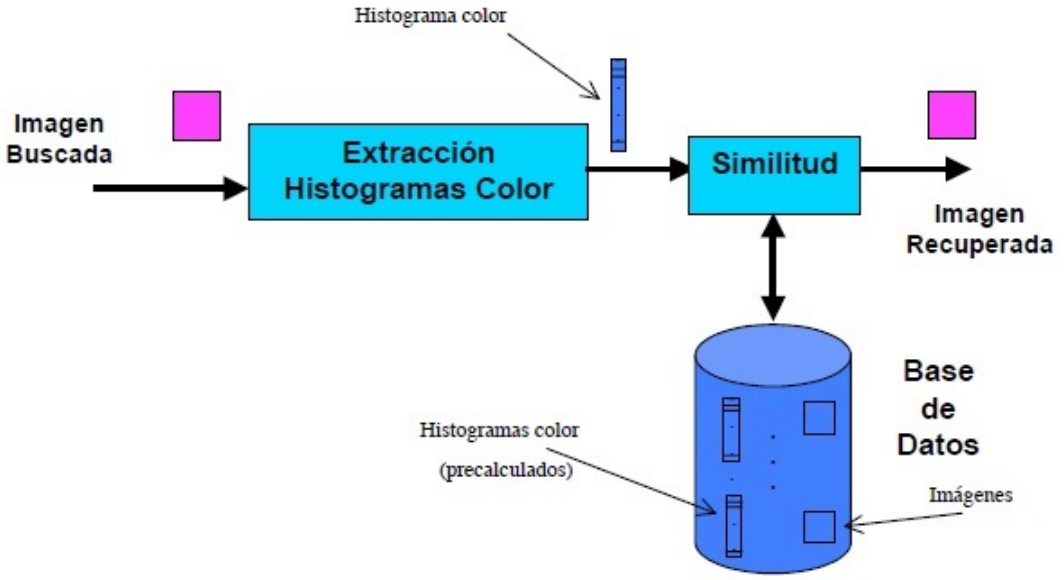
1. Se obtiene el ***histograma color*** utilizando espacio de color ***RGB***.

2. Se determinan los ***coeficientes de Haar*** del histograma mediante el uso de la transforma wavelet.

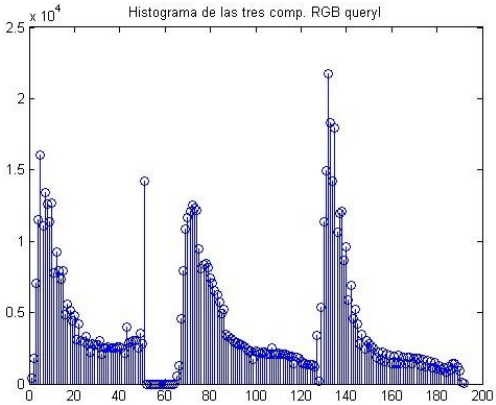
3. Se ***Binariza*** los coeficientes de Haar.

4. Se aplica la distancia de ***Hamming***.

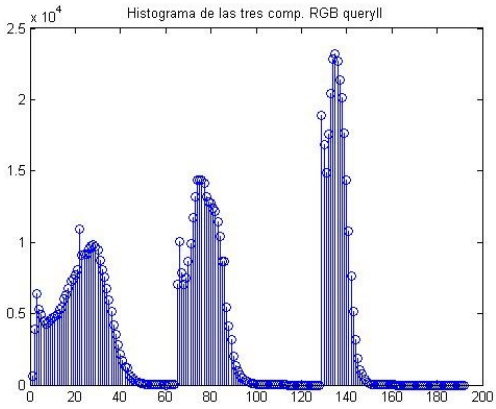
El proceso que seguimos para la recuperación las imágenes de prueba en la experimentación se basan en el siguiente esquema básico:

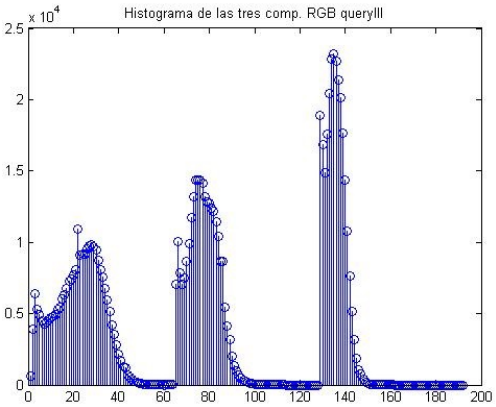


En el caso de la utilización de los histogramas de color de la query[1], en los diferentes experimentos se logró ver que los tonos claros de la imagen corresponden a los valores de la derecha histograma. A continuación, se muestran tres las imágenes de consulta, es decir las query I, II y III con sus respectivos histogramas:



[1]Es una técnica de consulta que implica dotar al sistema CBIR con una imagen de ejemplo, donde ésta servirá para realizar la búsqueda. Donde, CBIR (Content-based Image Retrieval), es decir la Consulta de imágenes mediante ejemplo es un sistema de búsqueda para recuperar imágenes basándose en su contenido, refiriéndose en su contexto a colores, figuras , texturas o cualquier otra información que pueda derivarse de la propia imagen.





A partir de los histogramas de las imágenes anteriores, podemos decir que el detalle de la imagen presenta una tonalidad dominante oscura, puesto que,se concentra en las sombras. También, podemos notar que existe un detalle de una imagen con tonalidad dominante clara, en las partes con más iluminaciones y el detalle de una imagen con tonalidad media, que corresponde a tonos medios. Una imagen con una gama tonal completa tiene píxeles en todas las áreas. Al identificar la gama tonal de la imagen nos ayuda a determinar las correcciones tonales correspondientes.

5.3.2. Resultados de las muestras de las imágenes recuperadas

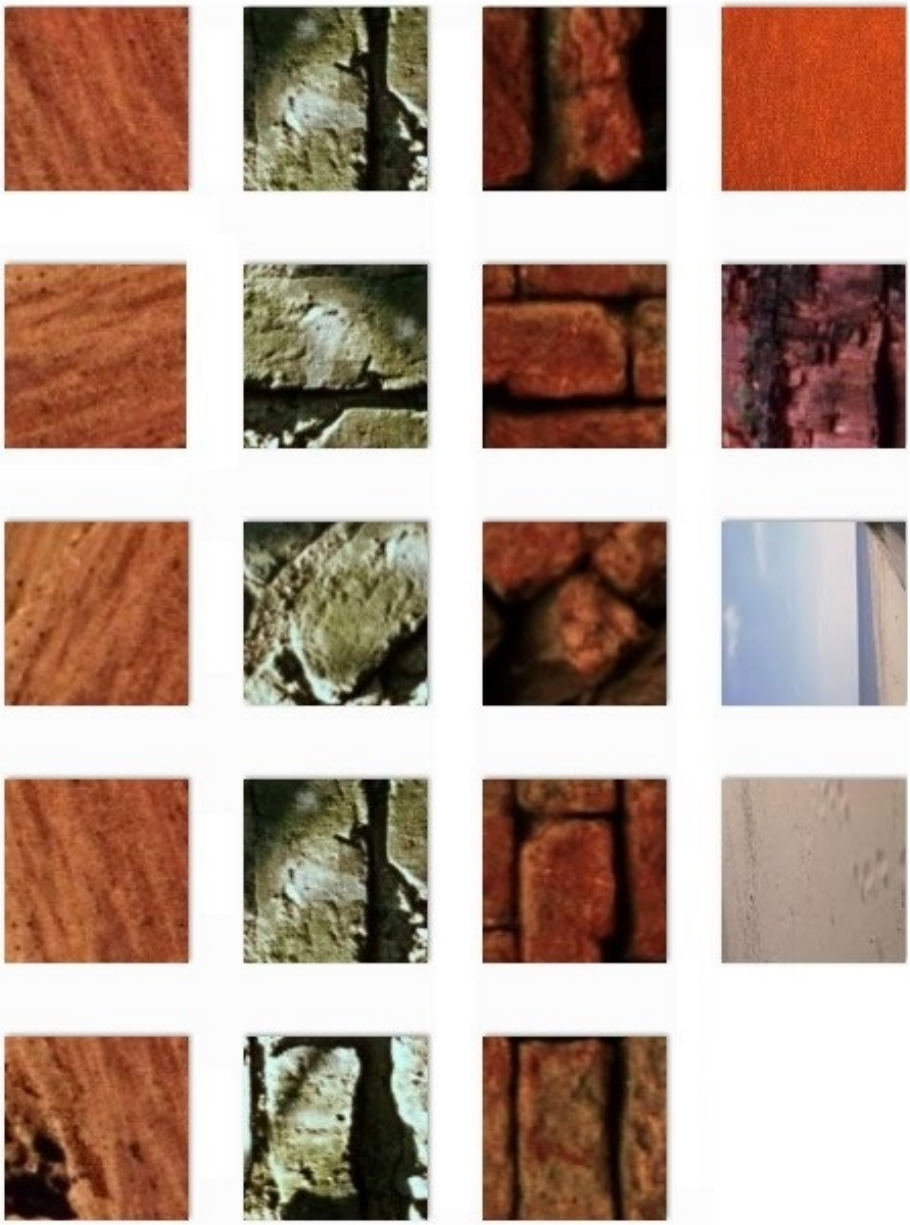
Para la realización de los experimentos, primero tratamos de obtener algunas imágenes digitales, con el fin de obtener la muestra, seleccionamos 3 imágenes grandes de texturas del conjunto de imágenes que se encuentran en [57] página web del grupo de investigación (JAMS Z. WANG) con las siguientes características:

- Que 2 de ellas tengan el mismo tipo de textura, y 2 el mismo color.

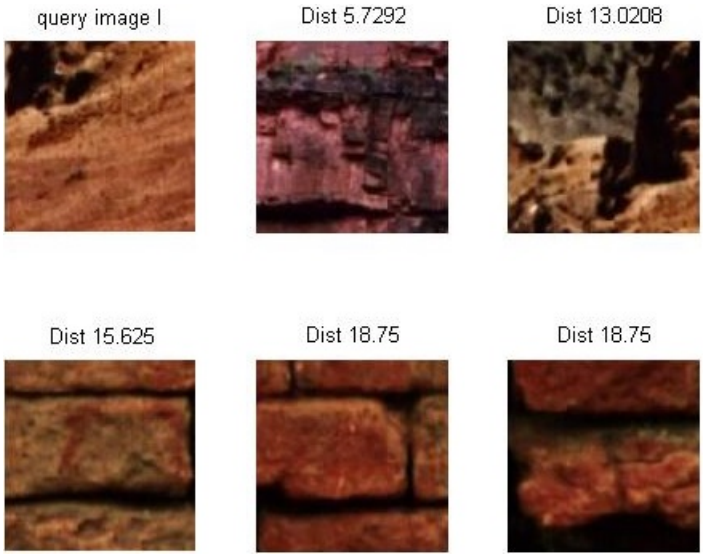
- Se quiere que 2 de las 3 tengan una textura similar, pero de distinto color.
- Y que 2 de las 3 tengan texturas diferentes de color similar. Son 3 imágenes de textura (i.e, pasto verde, pasto rojo, ladrillo rojo) ya que, cumplen con estas características.
- De cada una de las 3 imágenes se extraen 3 bloques de tamaño 128×128, es decir se obtiene 9 imágenes de menor tamaño, pero con características similares.
- Se agregan 4 imágenes pequeñas de tamaño 128×128, que tengan el mismo color predominante que las grandes, por ejemplo 2 de un color y 2 de otro.
- Se toma una imagen pequeña de cada imagen grande y se realiza una rotación con alguna aplicación o software como: photoshop o con MATLAB en 45 y 90 grados. O bien se rota la imagen grande y se extrae una de las imágenes pequeñas, es decir resultando 6 imágenes chicas. Con todo este proceso construimos nuestra base de datos de imágenes, las cuales son 19 en total, las que nos servirán para realizar los diferentes experimentos, y realizar las diferentes medidas y comparaciones hasta obtener la imagen recuperada más similar a la original.

A continuación, se muestra las imágenes Base de Texturas grandes de tamaño 512×512

Así como, el subconjunto de imágenes de prueba construidas según las características mencionadas:



Después, de haber realizado las diferentes corridas o simulaciones correspondientes a la *query image* *I*, nos quedamos con el siguiente grupo de imágenes las que presentan menor distancia entre las muestras construidas:



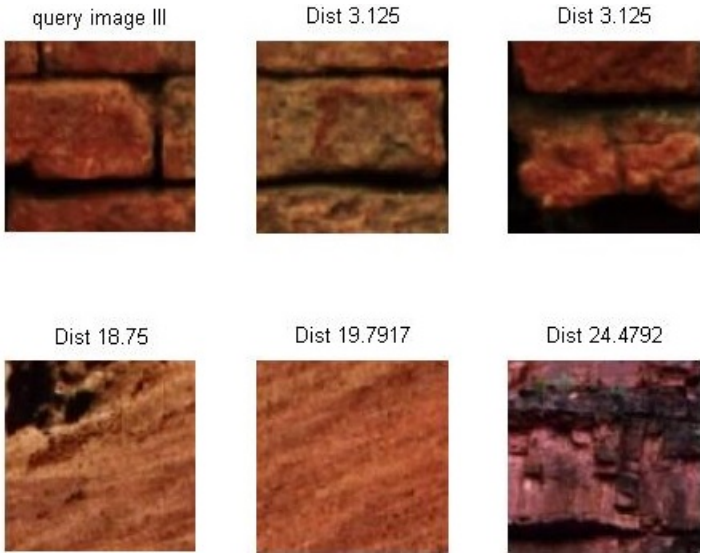
y de ellas vemos, qué algoritmo encuentra dos imágenes una imagen similar en el paso 2 con una distancia 13.0208, y que coincide con la imagen dividida de la esquina superior con respecto la imagen de consulta o de prueba.

En el segundo grupo de experimentación que corresponde a la *query image II*, obtenemos el siguiente subconjunto de imágenes recuperadas:

y de ellas observamos, qué algoritmo encuentra dos imágenes similares en el paso 1 y 2 con una distancia 0.52083 y 1.0417, y una de ellas coincide con la imagen de consulta o de prueba.

En el tercer grupo de experimentación, correspondiente a la *query image III* el algoritmo halla las imágenes similares en el paso 1 y 2 con una distancia 3.125 y 3.125, y que coincide con la imagen de

consulta o de prueba. Las que se muestran a continuación:



Lo cual podemos decir, que si incluimos la imagen original al grupo de prueba, el algoritmo encuentra la imagen original y ubica las más parecidas a ella. Además, afirmamos que:

1. Dos imágenes comparadas a través de sus histogramas globales de color pueden no estar semánticamente relacionadas a pesar de compartir una distribución similar de color.

2. Los descriptores binarios de la Wavelet de Haar de las imágenes de interés se compara con la de consulta utilizando la distancia de Hamming.

3. Las imágenes más similares son recuperadas y ordenadas según la distancia a la de consulta, como se observó en los diferentes experimentos realizados.

5.4. Algoritmos de Coincidencia para la Estimación de Movimientos

La implementación de algoritmos de estimación de movimiento es una de las tareas claves del proceso de compresión de secuencia de imágenes y de vídeo, y nos lleva un buen porcentaje en el tiempo de procesamiento, como se ha mencionado en el capítulo 3. Los algoritmos de coincidencia por bloques son desarrollados con la finalidad de reducir la complejidad computacional de los algoritmos, debido a

su eficiencia y el buen rendimiento. Las estrategias de búsqueda de ciertas características u objetos, es uno de los factores principales en el desarrollo de algoritmo de estimación de movimiento ya que tiene el potencial de proporcionar un buen rendimiento.

Este sección tiene como finalidad de implementar una serie de algoritmos de estimación de movimientos en el software $MATLAB$, seleccionados para lograr el mínimo número de cálculos y dar mejores valores de la medida de desempeño (señal de pico a ruido $PSNR$ desarrollado en la sección 2.2.10) utilizando las secuencias de imágenes proporcionadas.

Los algoritmos propuestos se modifican en base a la estrategia de búsqueda, según el enfoque de algoritmos de estimación de movimientos estándar. Estos se mejoran considerablemente mediante la implementación de algoritmos heurísticos y sus hibridos, proporcionando mejores resultados para PSNR sin aumentar el número de cálculos.

5.5. Secuencia de prueba para el conjunto de algoritmos de estimación de movimientos

Los experimentos realizados para el análisis del desempeño del conjunto de algoritmos descritos en el capítulo 3, consistieron en codificar una de secuencias de imágenes de prueba bus_cif_frame, selecciona del conjunto de imágenes que se encuentran en [47] página web de Video Trace Library (ARIZONA STATE UNIVERSITY), con ciertas características específicas, para realizar las comparaciones del rendimiento de los diferentes algoritmos, tomando las mediciones del número de puntos de búsqueda, del error en la estimación y compensación de movimientos generados.

Además, se describe las condiciones de los experimentos, en las que se utilizan 150 frame de las secuencias vídeo seleccionado de la misma página. Después, de las diferentes ejecuciones de los algoritmos, decidimos tomar un tamaño bloque de 16×16 píxeles y con un parámetro de búsqueda $p = \pm 7$, puesto que nos presenta mejores rendimientos y calidad visual en la reconstrucción. Además, se presen-

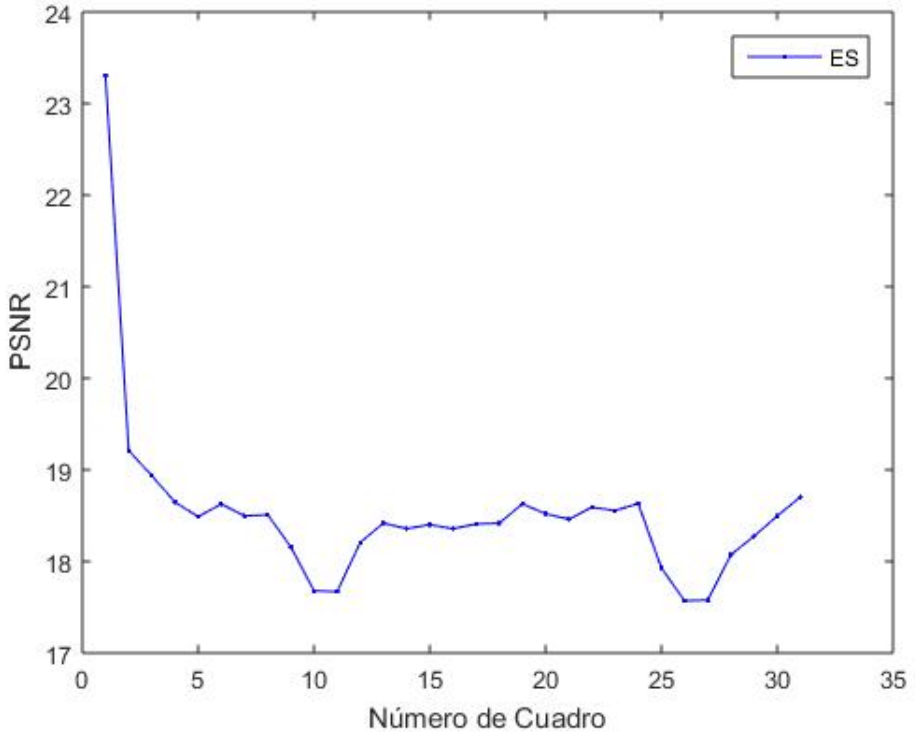
tan y comentan los resultados obtenidas tras la ejecución de los diferentes algoritmos desarrollados e implementados en el software $Matlab$. Los que se detallan a continuación:

- Algoritmo de búsqueda exhaustiva.
- Algoritmo de búsqueda en tres pasos.
- Algoritmo de búsqueda en cuatro pasos.
- Búsqueda en diamante.
- Algoritmo metaheurístico búsqueda en tres pasos simple y eficiente.
- Algoritmo metaheurístico mejorado en el patrón de búsqueda en tres pasos.

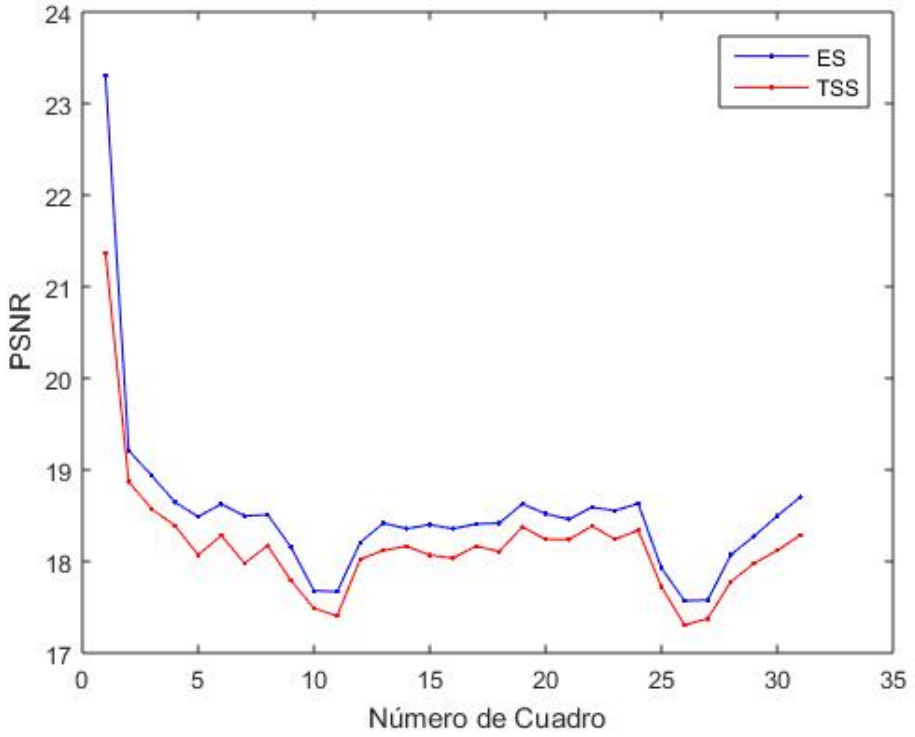
Los diferentes algoritmos se basan en las funciones de costo o métrica tratadas en el capítulo 3, que calculan el error entre dos macrobloques, de las que se eligieron MAD abordada en la sección 2.2.11 y la función de costos MSE abordada en 2.2.9, la elección de ellas se debe a que son las más conocidas y sencillas de implementar.

5.5.1. Resultados de Implementación de los Algoritmos Heurísticos de Coincidencia para la Estimación de Movimientos

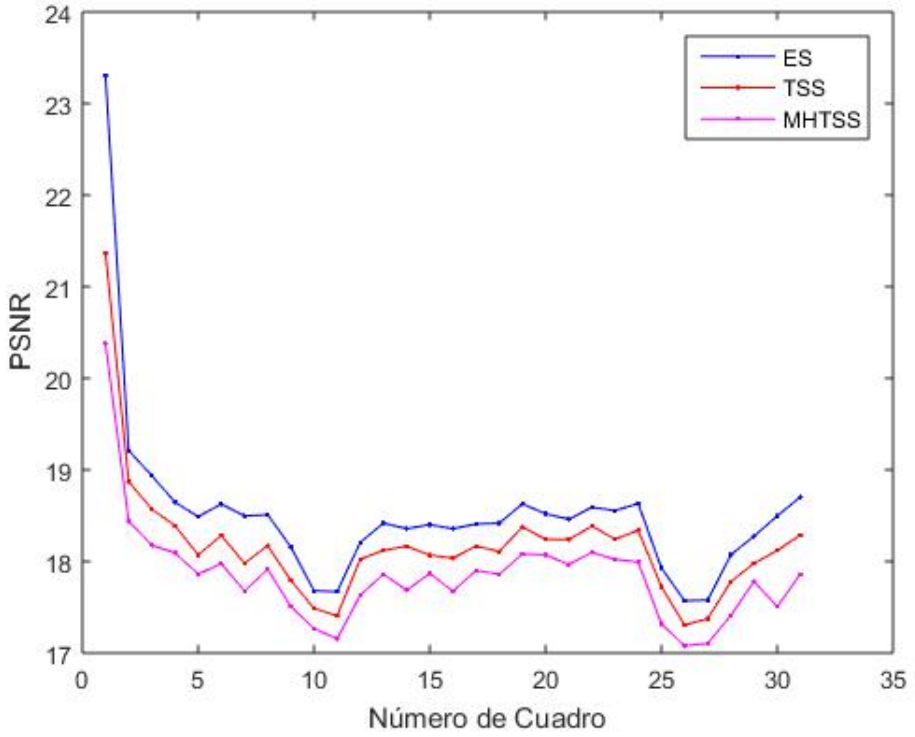
Para la precisión de los algoritmos de búsqueda en la estimación de movimientos se caracteriza por el valor del $PSNR$ contabilizando el promedio de los valores de PSNR. Cuanto más altos sean los valores obtenidos del PSNR, se obtendrá una mejor calidad de la imagen estimada y compensada. La relación de degradación de $PSNR$, también se utiliza en la comparación como se muestra en la siguiente figura. Esta relación se expresa en porcentaje ($\%$) entre la $PSNR$ de los algoritmos heurísticos propuestos como los de referencia, respectivamente. En la siguiente imagen vemos la ejecución del algoritmo búsqueda exhaustiva, lográndose un valor del $PSNR$ que oscila entre $0 - 19$ en la mayoría de los cuadros analizados, con un tamaño de bloque de 16×16 y un párametro de búsqueda de $p = 7$.



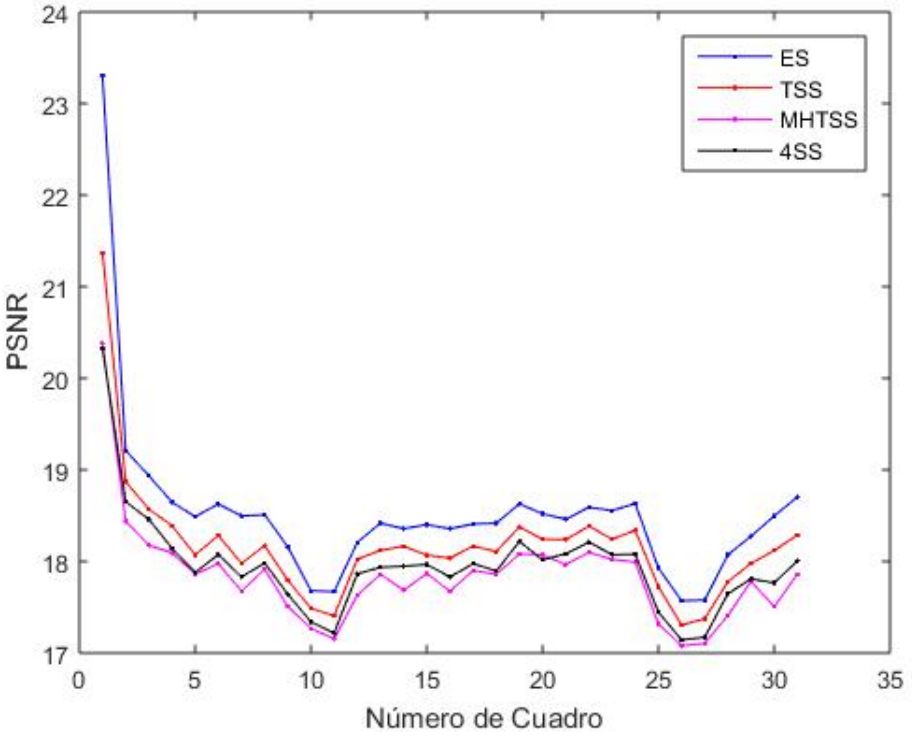
El análisis entre los números de cuadros vs su $PSNR$ se muestran en las siguiente figura, este nos explican que los valores obtenidos a partir del algoritmo búsqueda en tres pasos tiene mejor rendimiento computacional comparado con ES, pero no es capaz de detectar pequeños movimientos. Como se muestra a continuación



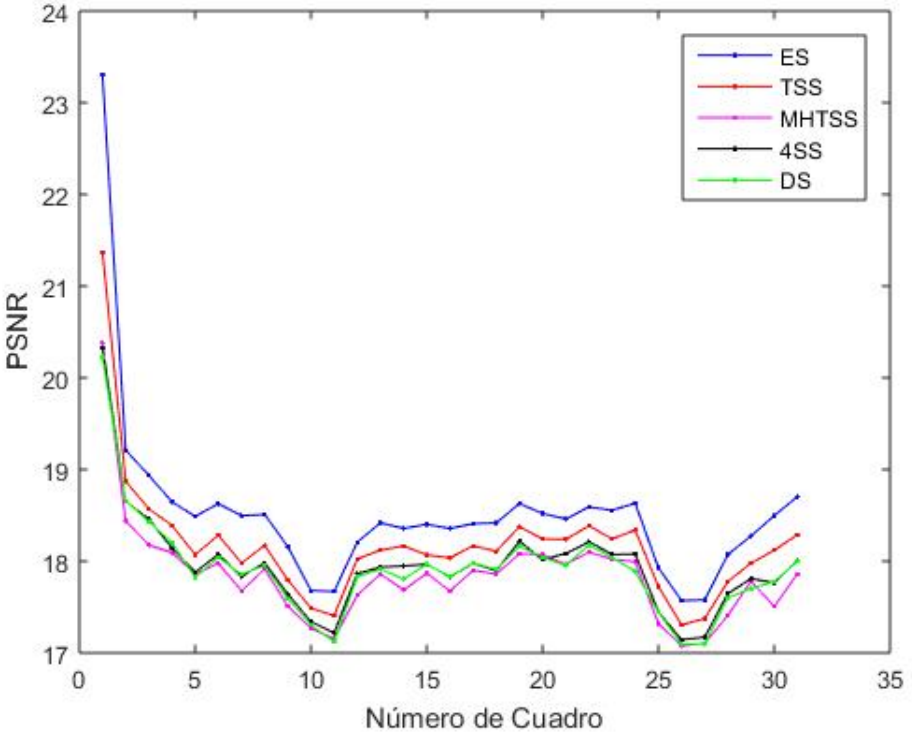
Al ejecutar el metaheurístico de búsqueda en tres pasos simple y eficiente $MHTSS$, vemos que este mejora considerablemente su rendimiento y hay buenos resultados para movimientos lentos, pero su complejidad computacional es mejor que la de TSS para el caso de movimientos rápidos, según la secuencia de imágenes de prueba. Esto detalles de ejecución se muestra a continuación:



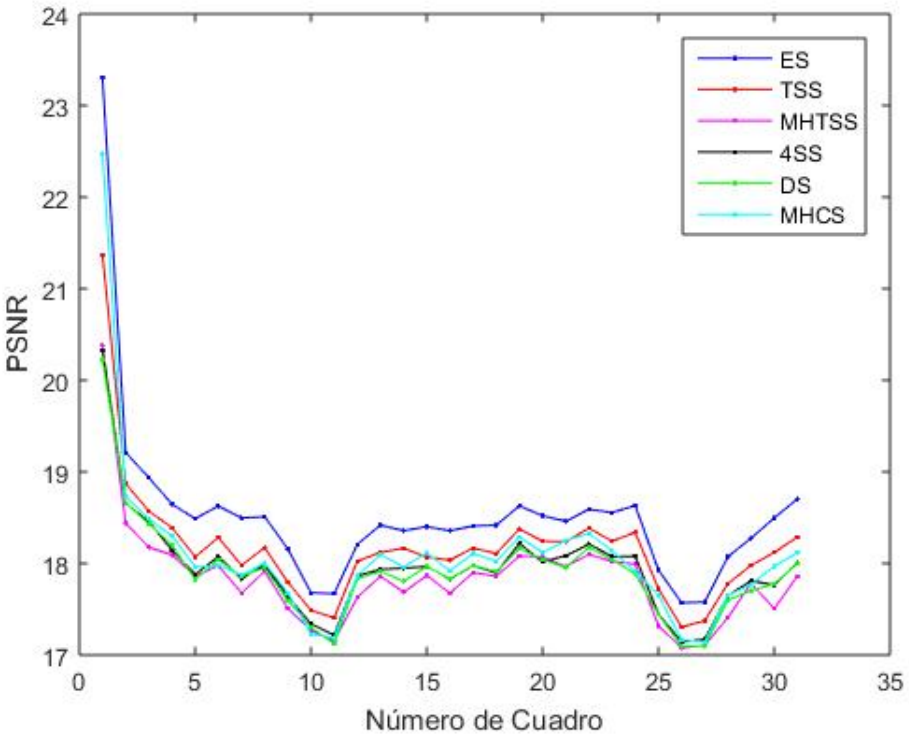
Las diferentes corridas del algoritmo búsqueda $4SS$ observamos que este presenta un ligero cambio significativo con respecto a la metaheurística simple, se logró detectar que la búsqueda en cuatro pasos es bueno para detectar movimientos lentos y su costo computacional es comparable con el metaheurístico simple y es menor que que algoritmo $MHTSS$. Los resultados de la ejecución se ve en la siguiente figura:



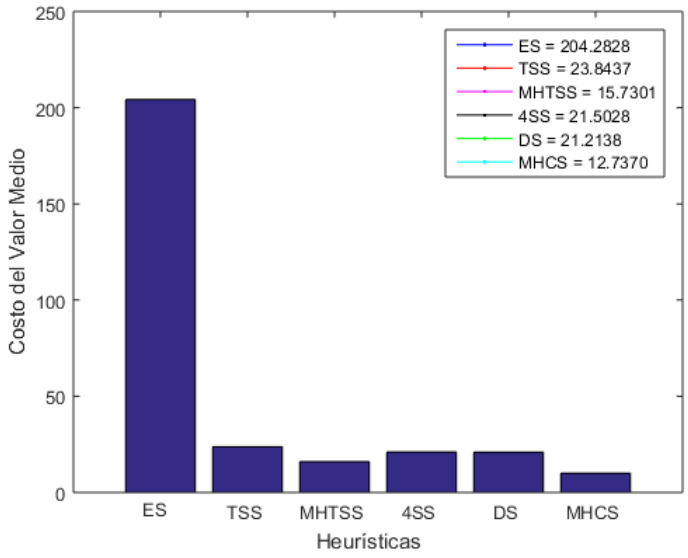
También, logramos apreciar que el algoritmo búsqueda DS tiene un ligero cambio significativo con respecto la metaheurística simple y la búsqueda en cuatro pasos, es bueno para detectar movimientos lentos. Estos hechos se ve en la siguiente figura:



La evaluación de la metaheurística mejorada según el patrón de búsqueda en cruz $MHCS$ implementada nos da un rendimiento medio, con respecto al resto de los algoritmos, y se comporta bien, tanto, computacionalmente cómo en términos de calidad visual, según las diferentes experimentaciones realizadas. Esto observa a continuación:



Todo lo anterior lo podemos fundamentar mediante la visualización del histograma de rendimiento de todos los algoritmos, notamos que el algoritmo ESes el que utiliza mayor costo computacional, según la métrica empleada. Los algoritmos heurísticos TSS, $4SS$ y DS utilizan aproximadamente el mismo tiempo de ejecución, es decir se comportan de forma similar, los metaheurístico $MHTSS$ y $MHCS$ trabaja de forma media y sus implementaciones son en promedio mejor con respectos a los demás algoritmos, debido a las mejoras de los parámetros variados. Como se muestra a continuación:



A continuación, mostramos el proceso de reconstrucción y de estimación de una imagen tras la ejecución del algoritmo de búsqueda en tres pasos:

Figura 5.1: Búsqueda en Tres Pasos

A continuación, se muestra el algoritmo de búsqueda en cuatro pasos:

Figura 5.2: Búsqueda en Cuatro Pasos

A continuación, se muestra el algoritmo de búsqueda en diamante o cruz:

Figura 5.3: Búsqueda en Diamante

A continuación, se muestra la mejora del heurístico de búsqueda en tres pasos simple y eficiente, que consiste en la mejora de los algoritmos de búsqueda en tres y cuatros pasos, resultado un poco más eficiente que ellos:

Figura 5.4: Mejora del heurísticos simple y eficiente de la búsqueda en tres pasos

En la siguiente figura vemos que al ejecutar heurístico propuesto, que consiste en la mejora de los algoritmos de búsqueda de patrones en tres, cuatros pasos y en combinación con el algoritmo de búsqueda en diamante, su resultados son un poco más eficiente, que los algoritmos bases y del heurístico.

Figura 5.5: Heurístico de mejora según el patrón de búsqueda en cruz

Capítulo 6

CONCLUSIONES Y PERSPECTIVA DE LA INVESTIGACIÓN

6.1. Conclusiones

En esta investigación, nos ocupamos de la estimación de movimiento $2-D$ disponible. Para este fin, se plantearon dos líneas principales de estudio: los que están desarrollados sobre los métodos de dominio de frecuencia, empleando texturas estáticas y la implementación de los métodos de dominio temporal utilizando texturas dinámicas, para su aplicación en la solución de problemas complejos de interés, como es la estimación de movimientos y compensación en la compresión de secuencias de imágenes o vídeo.

Se reviso y realizó un estudio unificado de todas las medidas de compresión, de la transformadas utilizadas con frecuencias y centrando nuestro interés en la transformada discreta del coseno y de la Wavelets, que son esenciales para la parte experimental de los algoritmos de estimación de movimientos, en especial las variantes en los heurísticos, ofreciendo un modelo formal de ellas. Se desarrollo el modelo formal, permitiendo caracterizar las heurísticas, observando los puntos en común que poseen los diferentes algoritmos de compresión y de búsqueda en los macrobloques de referencia, y centrándonos principalmente en la mejora de la calidad visual, la cantidad de información comprimida o reconstruida

y el comportamientos de los métodos en la búsqueda de similitudes entre píxeles o imágenes. Los algoritmos son suficientemente genéricos para abordar los modelos existentes y sus variantes, pero a la vez permiten controlar detalles de bajo nivel, que son especialmente interesantes en el tratamiento específico de las heurísticas. También, se estructuraron los resultados teóricos de la teoría matemática esencial para el diseño e implementación de los Algoritmos de Coincidencia para la Estimación de Movimiento en compresión de imágenes.

Se diseñaron e implementaron los códigos de la funciones, procedimientos fundamentales de los algoritmos del dominio de frecuencia (DCT, Wavelets) y Heurísticos (FS, TTS, 4SS, DS, MHTSS, MHCS) de Coincidencia para la Estimación de Movimiento en compresión de imágenes, escritos en el lenguaje de Programación Matemático $MATLAB$.

Los resultados generales muestran que al aplicar filtrado de los coeficientes de la DCT con base a valor es mejor que los coeficientes seleccionados por frecuencia, independiente del tamaño del macrobloque utilizado. En la que se realizaron diferentes corridas de ambos métodos, y sus posibles combinaciones, en las que se observó las mejoraras del desempeño, y la calidad visual de las imágenes reconstruidas. Este experimento nos sirve para afirmar que una selección por valor absoluto daría mejores resultados visuales ya que eliminan las altas frecuencias y dejan las más bajas.

Los experimentos de variar los parámetros y dividir la imágenes en bloques de diferentes tamaños, nos permitió decidir que puede ser un buen tamaño los bloques de 8×8, ya que obtuvimos un pixelado menor que al dividir la imagen en bloques de tamaño más pequeños. Para tamaños ≥ 8 mejora considerablemente el tamaño la calidad visual y el nivel de compresión de las imágenes analizadas.

En el caso de la parte experimental de la Wavelets pudimos apreciar que en las diferentes ejecuciones de los algoritmos implementados, que al incluir la imagen original a la base de datos de prueba el algoritmo implementado lo encuentra en el paso 1 o 2 con una distancia mínima, o muestra aquella que se encuentra a una distancia mínima. Además, dos imágenes comparadas a través de sus histogramas global de color pueden no estar semánticamente relacionadas a pesar de compartir una distribución

similar de color. Los descriptores binarios de la Wavelet de Haar de las imágenes de interés se compara con la imagen de consulta utilizando la distancia de Hamming. Por tanto, las imágenes más similares son recuperadas y ordenadas según la distancia a la de consulta.

El tercer conjunto de experimentos realizados en este trabajo, se centra en la implementación y evaluación del rendimiento del conjunto de algoritmos heurísticos de coincidencia para estimación de movimientos de búsqueda rápida, bajo el contexto de los codificadores normalizados $MPEG - JPEG$. Para concluir, los resultados demuestran que los algoritmos que presentan un mayor rendimiento son los heurístico mejorados $MHCS$ y $MHTSS$ debido a que son más eficiente y por su calidad visual, y en menor medida en orden creciente de complejidad computacional: DS, $4SS$, TS, y ES. En base al análisis realizado en este trabajo se considera estos algoritmos como los mejores en promedio.

6.2. Perspectiva de la Investigación

Como trabajo futuro, sería muy útil para ambas líneas tener una implementación más eficiente de los heurísticos, trabajar aún con la variación de los parámetros y las medidas de desempeño, pensar en el diseño de metaheurísticas para lograr mejores resultados. Además que se debe aprovechar el auge de los hardware en paralelo o la utilización de clúster de computadoras para agilizar los procesos de rendimiento, o en el caso que se trabaje con ordenadores de índole personal y con un buen poder de cálculo. Ya que una de las desventaja observadas de la líneas de investigación tratada, es que estos algoritmos de búsqueda son muy exigentes en términos de rendimiento computacional, desde el software utilizado que requiere de una buena capacidad de memoria caché y un buen procesador para su ejecución, ya que los hacen impracticable para aplicaciones en tiempo real, cuando la imagen o secuencia tiene mucho información que analizarse. Queda abierto el trabajo de los códigos en $C++$ para la mejora del rendimiento de los mismos.

Bibliografía

[1] Alegre, E., Sánchez, L., Fernández, R. Á., and Mostaza, J. C. (2003). *"Procesamiento Digital de Imagen: fundamentos y prácticas con Matlab"*. Universidad de León.

[2] Asmare, M. H., Asirvadam, V. S., and Iznita, L. (2009, April). *"Color space selection for color image enhancement applications"*. In Signal Acquisition and Processing, 2009. ICSAP 2009. International Conference on (pp. 208-212). IEEE. ISO 690

[3] Ayer, S., Sawhney, H. (1995). *"Layered representation of motion video using robust maximum-likelihood estimation of mixture models and MDL encoding"*. ICCV, pp. 777-784.

[4] Baker, S., & Matthews, I. (2004). Lucas-kanade 20 years on: *"A unifying framework"*. International journal of computer vision, 56(3), 221-255.

[5] Bakhtiari, S., Agaian, S., and Jamshidi, M. (2012, June). *"A color image enhancement method based on ensemble empirical mode decomposition and genetic algorithm"*. In World Automation Congress (WAC), 2012 (pp. 1-6). IEEE.

[6] Barron, I. L. (1994). D. I. Fleet, and SS Beauchemin, *"Performance of optical flow techniques"*. International Journal of Computer Vision, 12(2).

[7] Blelloch, G. E. (2001). *"Introduction to data compression"*. Computer Science Department, Carnegie Mellon University.

[8] Blum, C., and Roli, A. (2003). *"Metaheuristics in combinatorial optimization: Overview and conceptual comparison"*. ACM Computing Surveys (CSUR), 35(3), 268-308.

[9] Book, R. V. (1980). Michael R. Garey and David S. Johnson, *"Computers and intractability: A guide to the theory of NP completeness"*. Bulletin (New Series) of the American Mathematical Society, 3(2), 898-904.

[10] Bovik, A. C. (2010). *"Handbook of Image and Video Processing"*. Academic Press.

[11] Brox, T., Bruhn, A., Papenberg, N., and Weickert, J. (2004). *"High accuracy optical flow estimation based on a theory for warping"*. Computer Vision-ECCV, 25-36.

[12] Brox, T., & Malik, J. (2011). *"Large displacement optical flow: descriptor matching in variational motion estimation"*. IEEE transactions on pattern analysis and machine intelligence, 33(3), 500-513.

[13] CH, C. S., Ratnam, J. V. K., and Student, P. G. *"Comparison of Fast Block Matching Algorithms for Motion Estimation"*.

[14] Chen, W. H., Smith, C. H., and Fralick, S. C. (1977). *"A fast computational algorithm for the discrete cosine transform"*. IEEE Transactions on communications, 25(9), 1004-1009.

[15] Cheung, C. H., and Po, L. M. (2002). *"A novel cross-diamond search algorithm for fast block motion estimation"*. IEEE transactions on Circuits and Systems for Video Technology, 12(12), 1168-1177.

[16] Cook, S. A. (1971, May). *"The complexity of theorem-proving procedures"*. In Proceedings of the third annual ACM symposium on Theory of computing (pp. 151-158). ACM.

[17] Cuevas, E., Zaldívar, D., Pérez-Cisneros, M., Sossa, H., Osuna, V. (2013). *"Block matching algorithm for motion estimation based on Artificial Bee Colony (ABC)"*, Applied Soft Computing Journal 13 (6), pp. 3047-3059.

[18] Davis, L. (1991). *"Handbook of Genetic Algorithms Van Nostrand Reinhold New York Google Scholar"*.

[19] DUFAUX, F. (2010). *"Motion Estimation Techniques for Digital TV: A Review and a New Contribution"*.

[20] Ghanbari, M. (1990). *"The cross-search algorithm for motion estimation (image coding)"*. IEEE Transactions on Communications, 38(7), 950-953.

[21] Glover, F. W., and Kochenberger, G. A. (Eds.). (2006). *"Handbook of metaheuristics"* (Vol. 57). Springer Science & Business Media.

[22] Goldberg, D. E., and Holland, J. H. (1988). *"Genetic algorithms and machine learning. Machine learning"*, 3(2), 95-99.

[23] Goldberg, D. (1989). *"Genetic algorithms in optimization, search and machine learning"*. Reading: Addison-Wesley.

[24] González, R. C., and Woods, R. E. (1996). *"Tratamiento digital de imágenes"*. (Vol. 3). New York: Addison-Wesley.

[25] Evan Herbst, Xiaofeng Ren, and Dieter Fox. (2013) *"Rgb-d flow: Dense 3-d motion estimation using color and depth"*. In Robotics and Automation (ICRA), IEEE International Conference on, pages 2276-2282. IEEE, 2013.

[26] Holland, J. H. (1975). *"Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence"*. Ann Arbor, MI: University of Michigan Press.

[27] Horn, B. K., & Schunck, B. G. (1981). *"Determining optical flow"*. Artificial intelligence, 17(1-3), 185-203.

[28] Hueso, J. L., Riera, J., and Ginestar, D. (2013). *"El flujo óptico como herramienta para el vídeo-análisis de fenómenos físicos"*. Modelling in Science Education and Learning, 6, 97-111.

[29] Huffman, D. A. (1952). *"A method for the construction of minimum-redundancy codes"*. Proceedings of the IRE, 40(9), 1098-1101.

[30] Jagalingam, P., and Hegde, A. V. (2015). *"A review of quality metrics for fused image"*. Aquatic Procedia, 4, 133-142.

[31] Klette, R. (2014). *"Concise computer vision"*. Springer, London.

[32] Konigsberg, I. (2004). *"Diccionario técnico Akal de cine (Vol. 3)"*. Ediciones Akal.

[33] Kung, C. M., Cheng, W. S., and Jeng, J. H. (2014). *"Application of genetic algorithm to hexagon-based motion estimation"*. The Scientific World Journal.

[34] Lee, B. (1984). *"A new algorithm to compute the discrete cosine transform"*. IEEE Transactions on Acoustics, Speech, and Signal Processing, 32(6), 1243-1245.

[35] Lee, K. Y., and El-Sharkawi, M. A. (Eds.). (2008). *"Modern heuristic optimization techniques: theory and applications to power systems"* (Vol. 39). John Wiley and Sons.

[36] Lelewer, D. A., and Hirschberg, D. S. (1987). *"Data compression"*. ACM Computing Surveys (CSUR), 19(3), 261-296.

[37] Li, R., Zeng, B., and Liou, M. L. (1994). *"A new three-step search algorithm for block motion estimation"*. IEEE transactions on circuits and systems for video technology, 4(4), 438-442.

[38] Lu, J., and Liou, M. L. (1997). *"A simple and efficient search algorithm for block-matching motion estimation"*. IEEE Transactions on Circuits and Systems for Video Technology, 7(2), 429-433.

[39] Lucas, B. D., & Kanade, T. (1981). *"An iterative image registration technique with an application to stereo vision"*. In IJCAI, pages 674-679, 1981.

[40] Naveros, F., Díaz, T., Ralli, J., Ros, E., and Díaz, J. *"Flujo óptico variacional en plataformas paralelas GPU."*

[41] Nie, Y., and Ma, K. K. (2002). *"Adaptive rood pattern search for fast block-matching motion estimation"*. IEEE Transactions on image processing, 11(12), 1442-1449.

[42] Po, L. M., and Ma, W. C. (1996). *"A novel four-step search algorithm for fast block motion estimation"*. IEEE transactions on circuits and systems for video technology, 6(3), 313-317.

[43] Pourreza, H. R., Rahmati, M., and Behazin, F. (2002). *"Adaptive Pixel Difference Classification, an Efficient and Cost Effective Algorithm for Motion Estimation"*. Threshold, 3(4), 3.

[44] Redert, A., Hendriks, E., and Biemond, J. (1999). *"Correspondence estimation in image pairs"*. IEEE Signal Processing Magazine, 16(3), 29-46.

[45] Reeves, C. R. (1993). *"Modern heuristic techniques for combinatorial problems"*. John Wiley and Sons, Inc..

[46] Reeves, C. R. (1996). *"Modern heuristic techniques. Modern heuristic search methods"*, 1-25.

[47] Reisslein, M. (2005-2011). *"Video Trace Library "*. Arizona State University, [online] Available: http://trace.eas.asu.edu.

[48] Ruiz, V. González. (2000). *"Compresión reversible y transmisión de imágenes"*. PhD thesis, Universidad de Almería.

[49] Salomon David, (2007) *"Data compression: The Complete Reference"*, 4ta. Ed., Springer.

[50] Scarpino, F. (1997). *"VHDL and AHDL digital system implementation"*. Simon and Schuster Trade.

[51] Shahdad, M., Lipsett, R., Marschner, E., Sheehan, K., and Cohen, H. (1985). *"VHSIC hardware description language"*. Computer, 2(18), 94-103.

[52] Stiller, C., and Konrad, J. (1999). *"Estimating motion in image sequences"*. IEEE Signal Processing Magazine, 16(4), 70-91.

[53] Utenpattanant, A., Chitsobhuk, O., and Khawne, A. (2006, February). *"Color descriptor for image retrieval in wavelet domain"*. In Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference (Vol. 1, pp. 4-pp). IEEE.

[54] Verma, N., Sahu, T., and Sahu, P. (2012). *"Efficient motion estimation by fast three step search algorithms"*. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 1(5).

[55] Von Helmholtz, H. (1925). *"Treatise on Physiological Optics"*: Translated from the 3rd German Ed. J. P. C. Southall (Ed.). Optical Society of America.

[56] Wang, J., Adelson, E. (1994). *"Representing moving images with layers"*. In IEEE Transactions on Image Processing, vol. 3, no. 5, pp. 625-638.

[57] Wang. James Z. (2012-2017). *"MODELING OBJECTS, CONCEPTS, AESTHETICS AND EMOTIONS IN BIG VISUAL DATA "*. JAMES Z. WANG RESEARCH GROUP, [online] Available: http://wang.ist.psu.edu.

[58] Wintz, P. A. (1972). *"Transform picture coding"*. Proceedings of the IEEE, 60(7), 809-820.

[59] Yaakob, R., Aryanfar, A., Halin, A. A., and Sulaiman, N. (2013). *"A comparison of different block matching algorithms for motion estimation"*. Procedia Technology, 11, 199-205.

[60] Z. Michalewicz (1992). *"Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Berlin Heidelberg"*.

[61] Zitnick, C. L., Jojic, N., Kang, S. B. (2005) *"Consistent Segmentation for Optical Flow Estimation"*. In Proceeding ICCV proceedings of the tenth IEEE International Conference on Computer Vision, Vol. 2, pp. 1308 - 1315.

Capítulo 7

ANEXOS

7.1. Códigos de las funciones y Algoritmos Heurísticos de coincidencia para la Estimación de Movientos

A continuación se detallan cada uno de ellos, primero se presenta una imagen impresa del sotfware utilizado, después se presenta las principales funciones para la implementación de los diferentes algoritmos diseñados en los capítulos anteriores de la investigación.

7.1.1. Lectura, escritura de las imágenes y áreas de interés

A continuación mostramos como leer y escribir las diferentes imágenes, el proceso de obtenerla en escala gris, las extracciones de los diferentes planos, reducción en el número de colores, el filtrado, la composición de imágenes para obtener otras, imagen reducida es decir el área de interés y ampliación de la misma, y la lectura de cada una de los ficheros, y propiedades que se tomaron en cuenta en la implementación de los algoritmos heurísticos de la investigación.

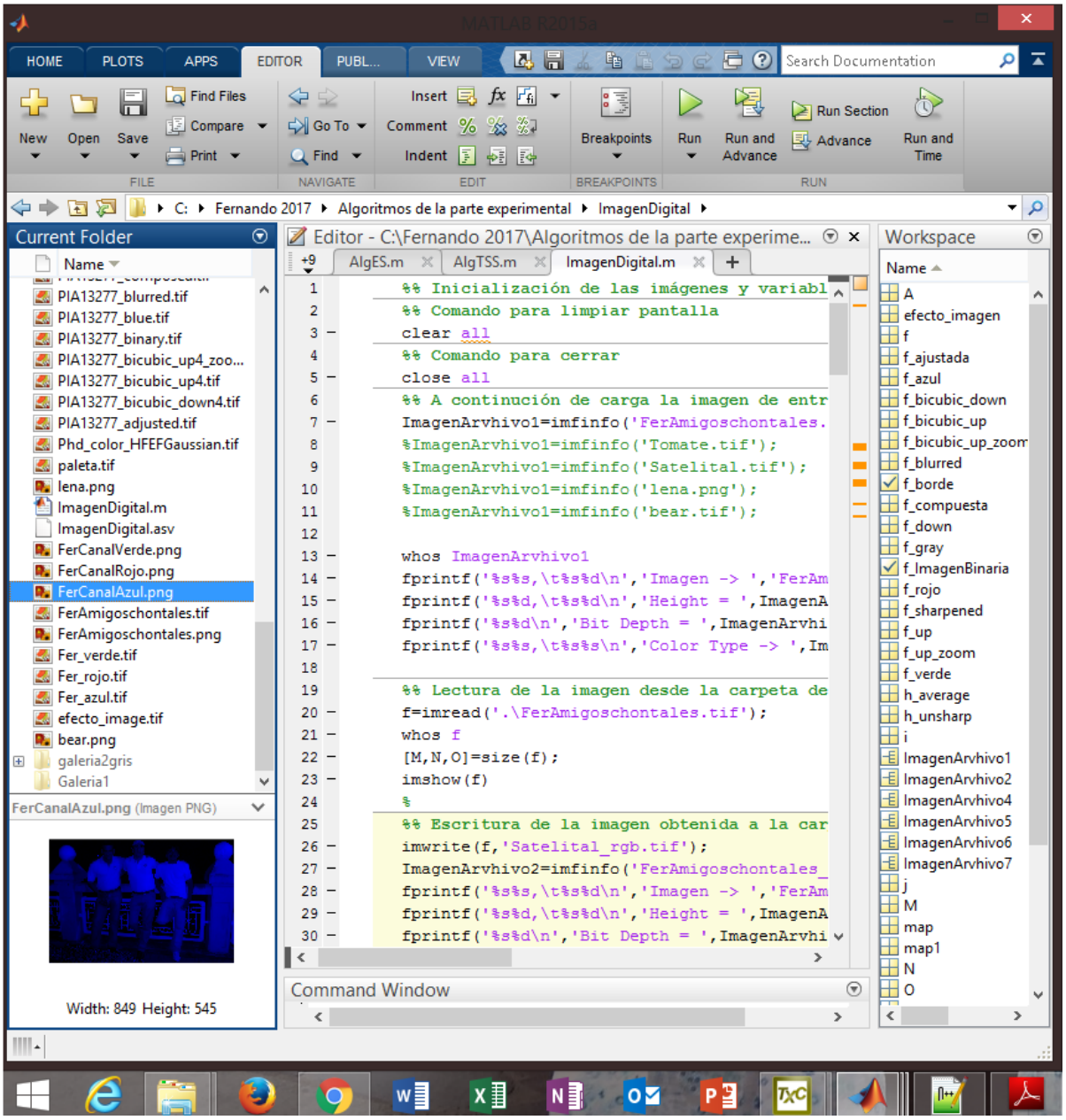


Figura 7.1: Preprocesamientos de imágenes digitales

```
%%Inicialización de las imágenes y variables de entradas
clear all %%Comando para limpiar pantalla
close all %%Comando para cerrar

%%A continución se carga las imágenes iniciales de entrada
ImagenArvhivo1=imfinfo('FerAmigoschontales.tif');
%%ImagenArvhivo1=imfinfo('Tomate.tif');
%%ImagenArvhivo1=imfinfo('FerAmigoschontales.tif');
%%ImagenArvhivo1=imfinfo('lena.png');
```

```
%%ImagenArvhivo1=imfinfo('bear.tif');

whos ImagenArvhivo1
fprintf('%s %s,\t %s %d\n','Imagen-->','FerAmigoschontales.tif','File
    Size(Bytes)=',ImagenArvhivo1.FileSize)
fprintf('%s %s,\t %s %d\n','Height=',ImagenArvhivo1.Height,'Width=',
    ImagenArvhivo1.Width)
fprintf('%s %d\n','Bit Depth=',ImagenArvhivo1.BitDepth)
fprintf('%s %s,\t %s %d\n','Color Type-->',ImagenArvhivo1.ColorType,'
    Compression-->',ImagenArvhivo1.Compression)

%%Lectura de una de las imagen prueba desde la carpeta de origen
f=imread('.\FerAmigoschontales.tif');
whos f
[M,N,O]=size(f);
imshow(f)

%%Escritura de la imagen obtenida a la carpeta de destino
imwrite(f,'FerAmigoschontales_rgb.tif');
ImagenArvhivo2=imfinfo('FerAmigoschontales_rgb.tif');
fprintf('%s %s,\t %s %d\n','Imagen-->','FerAmigoschontales_rgb.tif','
    File Size(Bytes)=',ImagenArvhivo2.FileSize)
fprintf('%s %s,\t %s %d\n','Height=',ImagenArvhivo2.Height,'Width=',
    ImagenArvhivo2.Width)
fprintf('%s %d\n','Bit Depth=',ImagenArvhivo2.BitDepth)
fprintf('%s %s,\t %s %s\n','Color Type--> ',ImagenArvhivo2.ColorType,'
    Compression--> ',ImagenArvhivo2.Compression)

%%Conversión de la Imagen a escala gris
f_gray=rgb2gray(f);
whos f_gray
figure, imshow(f_gray)
imwrite(f_gray, 'FerAmigoschontales_gray.tif');
ImagenArvhivo5=imfinfo('FerAmigoschontales_gray.tif');
fprintf('%s %s,\t %s %d\n','Imagen-->','FerAmigoschontales_gray.tif','
    File Size(Bytes)=',ImagenArvhivo5.FileSize)
fprintf('%s %s,\t %s %d\n','Height = ',ImagenArvhivo5.Height,'Width =
    ',ImagenArvhivo5.Width)
fprintf('%s %d\n','Bit Depth=',ImagenArvhivo5.BitDepth)
```

```
fprintf('% s %s,\t %s %s\n','Color Type -> ',ImagenArvhivo5.ColorType,'
    Compression -> ',ImagenArvhivo5.Compression)

%%Extracción de cada uno de los Planos Rojo, Verde, y Azul de la imagen
    original
f_rojo=f(:,:,1);f_verde=f(:,:,2);f_azul=f(:,:,3);
whos f_rojo f_verde f_azul
figure, imshow(f_rojo)
figure, imshow(f_verde)
figure, imshow(f_azul)
imwrite(f_rojo, 'FerAmigoschontales_rojo.tif');
imwrite(f_verde, 'FerAmigoschontales_verde.tif');
imwrite(f_azul, 'FerAmigoschontales_azul.tif');

A = imread('.\FerAmigoschontales.tif');
T_rojo  = A
T_rojo(:,:,2) = 0
T_rojo(:,:,3) = 0
%imshow(T_rojo)
figure, imshow(T_rojo)
imwrite(f_rojo, 'Fer_rojo.tif');
T_azul  = A
T_azul(:,:,1) = 0
T_azul(:,:,2) = 0
%imshow(T_rojo)
figure, imshow(T_azul)
imwrite(f_rojo, 'Fer_azul.tif');
T_verde  = A
T_verde(:,:,1) = 0
T_verde(:,:,3) = 0
%imshow(T_rojo)
figure, imshow(T_verde)
imwrite(f_rojo, 'Fer_verde.tif');

%%Conversión a una imagen indexada
[X map]=rgb2ind(f,256);
whos H map
figure
imshow(X,map)
```

```
%Reducción del Número de Colores
[Y map1]=imapprox(X,map,32);
figure
imshow(Y,map1)
paleta=zeros(512,128,3);
for j=1:32
        for i=1:16
                for z=1:3
                                paleta(i+(j-1)*16,:,z)=map1(j,z);
                end
        end
end
figure, imshow(paleta,[])

%%Escritura de la imagen indexada a 32 colores
imwrite(Y, map1, 'FerAmigoschontales_indexada.tif');
ImagenArvhivo4=imfinfo('FerAmigoschontales_indexada.tif');
fprintf('%s %s,\t %s %d\n','Imagen -> ','FerAmigoschontales_indexada.
    tif','File Size(Bytes)=',ImagenArvhivo4.FileSize)
fprintf('%s %s,\t %s %d\n','Height=',ImagenArvhivo4.Height,'Width=',
    ImagenArvhivo4.Width)
fprintf('%s %d\n','Bit Depth=',ImagenArvhivo4.BitDepth)
fprintf('%s %s,\t %s %s\n','Color Type -> ',ImagenArvhivo4.ColorType,'
    Compression -> ',ImagenArvhivo4.Compression)

%Escritura de la Paleta de 32 Colores
imwrite(paleta, 'paleta.tif');

%%Conversión a Imagen Binaria
f_ImagenBinaria = im2bw(f);
figure, imshow(f_ImagenBinaria)
%Escritura de la Imagen Binaria
imwrite(f_ImagenBinaria, 'FerAmigoschontales_ImagenBinaria.tif');
ImagenArvhivo5=imfinfo('FerAmigoschontales_ImagenBinaria.tif');
fprintf('%s %s,\t %s %d\n','Imagen -> ','
    FerAmigoschontales_ImagenBinaria.tif','File Size(Bytes)=',ImagenArvhivo5
    .FileSize)
fprintf('%s %s,\t %s %d\n','Height=',ImagenArvhivo5.Height,'Width=',
```

```
    ImagenArvhivo5.Width)
fprintf('%s %d\n','Bit Depth=',ImagenArvhivo5.BitDepth)
fprintf('%s %s,\t %s %s\n','Color Type -> ',ImagenArvhivo5.ColorType,'
    Compression -> ',ImagenArvhivo5.Compression)

%Escritura de la Imajen Ajustada
imwrite(f_ajustada, 'FerAmigoschontales_ajustada.tif');

%%Filtrado con Máscara de Desenfoque (Promediado) y Generación de la
    Máscara
h_average = fspecial('average',31);

%Filtrado
f_blurred = imfilter(f,h_average,'replicate');
figure, imshow(f_blurred)

%Escritura de la Imagen Filtrada
imwrite(f_blurred, 'FerAmigoschontales_ImagenFiltrada.tif');

%%Filtrado con Máscara de Enfoque (Destaca Detalles) y Generación de la
    Máscara
h_unsharp = fspecial('unsharp');
%Filtrado
f_sharpened = imfilter(f,h_unsharp,'replicate');
figure, imshow(f_sharpened)
%Escritura de la Imagen Filtrada
imwrite(f_sharpened, 'FerAmigoschontales_sharpened.tif');

%%Composición o combinación de Imágenes y Creación de Máscara-Efecto
radio=M/4 + M/8;
efecto_imagen=ones(M,N);
for i=1:M
        for j=1:N
                        if sqrt((abs(i-M/2)^2 + abs(j-N/2)^2)) > radio
                                efecto_imagen(i,j)=radio^2/((abs(i-M/2)^2 +
                                    abs(j-N/2)^2));
                        end
        end
end
```

```
%Creación de la Imagen Compuesta
f_compuesta=zeros(M,N,O);
for i=1:O
                f_compuesta(:,:,i)=immultiply(im2double(f(:,:,i)),
                    efecto_imagen);
end
figure, imshow(f_compuesta)
figure, imshow(efecto_imagen)

%Escritura de la Imagen Compuesta
imwrite(f_compuesta, 'FerAmigoschontales_compuesta.tif');

%Escritura de la Máscara-Efecto
imwrite(efecto_imagen, 'efecto_image.tif');

%%Redimensionamiento de la Imagen próxima y Reducción del Tamaño a 1/4
scale = 4;
f_down=imresize(f,1/scale, 'próxima');

%Recuperación del tamaño (Imagen Reducida x4)
f_up=imresize(f_down,scale, 'próxima');

%Detalle Central de la Imagen Recuperada
f_up_zoom=f_up(M/2-M/8:M/2+M/8-1,N/2-N/8:N/2+N/8-1,:);
figure, imshow(f_down)
figure, imshow(f_up)
figure, imshow(f_up_zoom)

%Escritura de la Imagen Reducida
imwrite(f_down, 'FerAmigoschontales_down4.tif');

%Escritura de la Imagen Recuperada
imwrite(f_up, 'FerAmigoschontales_up4.tif');

%Escritura del Detalle Central
imwrite(f_up_zoom, 'FerAmigoschontales_up4_zoom.tif');

%%Redimensionamiento de la Imagen (Bicubica)
```

```
%Reducción del Tamaño a 1/4
f_bicubic_down=imresize(f,1/scale,'bicubic');

%Recuperación del tamaño (Imagen Reducida x4)
f_bicubic_up=imresize(f_down,scale,'bicubic');

%Detalle Central de la Imagen Recuperada
f_bicubic_up_zoom=f_bicubic_up(M/2-M/8:M/2+M/8-1,N/2-N/8:N/2+N/8-1,:);
figure, imshow(f_bicubic_down)
figure, imshow(f_bicubic_up)
figure, imshow(f_bicubic_up_zoom)

%Escritura de la Imagen Reducida
imwrite(f_bicubic_down, 'FerAmigoschontales_bicubic_down4.tif');

%Escritura de la Imagen Recuperada
imwrite(f_bicubic_up, 'FerAmigoschontales_bicubic_up4.tif');

%Escritura del Detalle Central
imwrite(f_bicubic_up_zoom, 'FerAmigoschontales_bicubic_up4_zoom.tif');

%%Escritura de la Imagen con Compresión JPEG
imwrite(f,'FerAmigoschontales_Q_100.jpg','Calidad',100);
imwrite(f,'FerAmigoschontales_Q_75.jpg','Calidad',75);
imwrite(f,'FerAmigoschontales_Q_50.jpg','Calidad',50);
imwrite(f,'FerAmigoschontales_Q_25.jpg','Calidad',25);
imwrite(f,'FerAmigoschontales_Q_0.jpg','Calidad',0);
figure, imshow(imread('FerAmigoschontales.jpg'))
figure, imshow(imread('FerAmigoschontales_Q_75.jpg'))
figure, imshow(imread('FerAmigoschontales_Q_50.jpg'))
figure, imshow(imread('FerAmigoschontales_Q_25.jpg'))
figure, imshow(imread('FerAmigoschontales_Q_0.jpg'))

%Lectura de las Propiedades de un Fichero JPEG
ImagenArvhivo7=imfinfo('FerAmigoschontales_Q_75.jpg');
fprintf('%s %s,\t %s %d\n','Imagen --> ','FerAmigoschontales_Q_75.jpg
    ','File Size(Bytes)=',ImagenArvhivo7.FileSize)
fprintf('%s %s,\t %s %d\n','Height=',ImagenArvhivo7.Height,'Width=',
    ImagenArvhivo7.Width)
```

```
fprintf('%s %d\n','Bit Depth=',ImagenArvhivo7.BitDepth)
fprintf('%s %s,\t %s %s\n','Color Type-->',ImagenArvhivo7.ColorType,'
    Formato --> ',ImagenArvhivo7.Format)
```

7.1.2. Implementación de las funciones de los umbrales de la DCT

A continuación mostramos los códigos realizados en Matlab de las principales funciones, procedimientos utilizadas en el proceso de la transformada discreta del coseno, entre ellas tenemos:

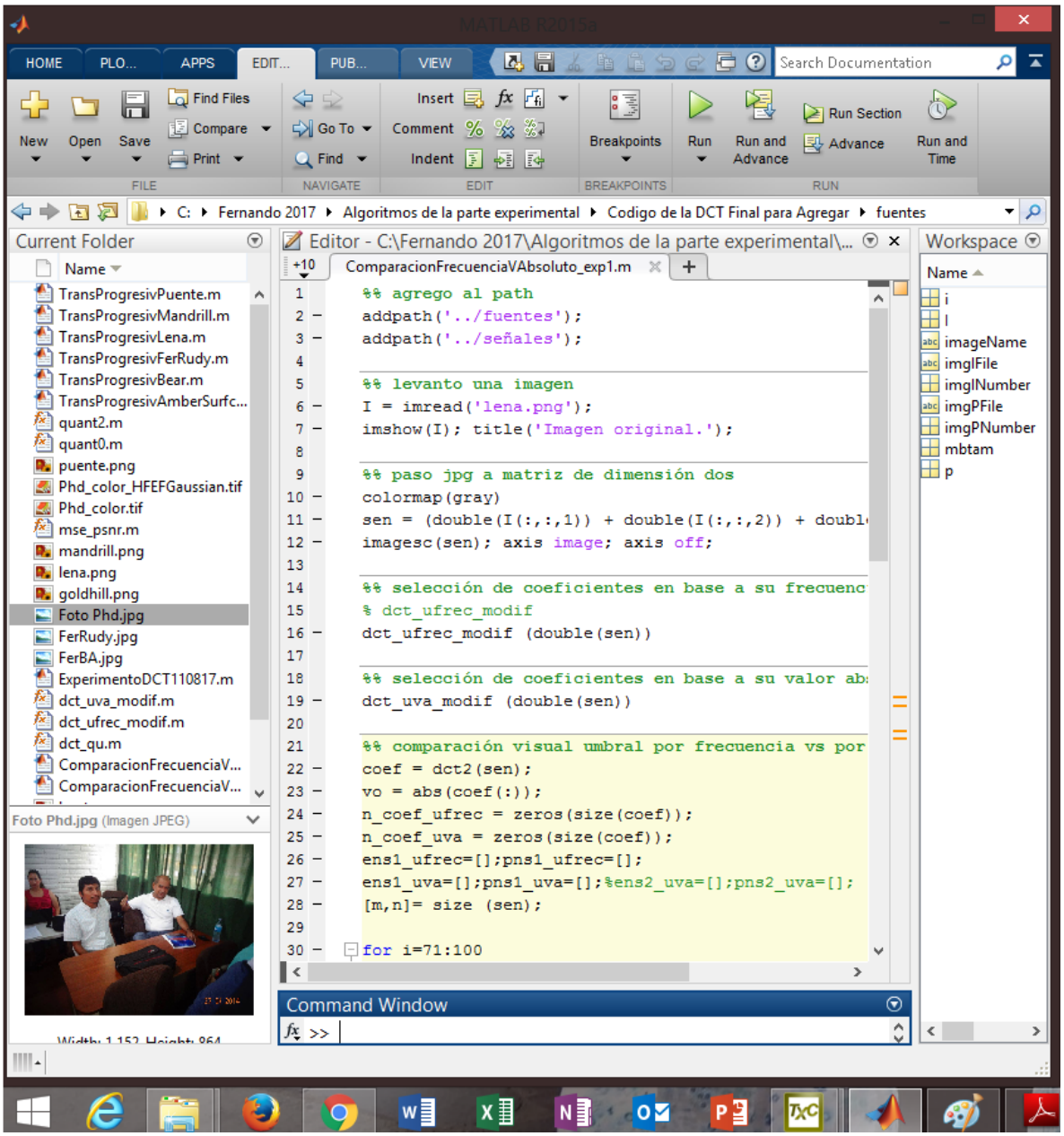


Figura 7.2: Comparación de la DCT

```
%%agrega la dirección o ruta de acceso
addpath('fuentes'); %son los procedimientos adicionales a utilizar
addpath('señales'); %el conjunto de imágenes de prueba

%Se lee la imágenes, en particular  "FerPhd"
I = imread('FerPhd.jpg');
%I = imread('Amber.png');
%I = imread('Mandril.png');
%I = imread('FerRudy.jpg');
%I = imread('FerPuente.png');

%Se paso a una matriz de dimensión dos y escala de gris
colormap(gray)
sen = (double(I(:,:,1)) + double(I(:,:,2)) + double(I(:,:,3)))/3;

%comparación visual de los umbrales: por frecuencia vs por valor absoluto
coef = dct2(sen); vo = abs(coef(:));
n_coef_ufrec = zeros(size(coef));
n_coef_uva = zeros(size(coef));
ens1_ufrec=[];pns1_ufrec=[];
ens1_uva=[];pns1_uva=[];
[m,n]= size (sen);

for i=1:m
        n_coef_ufrec(1:i,1:i) = coef(1:i,1:i); %umbral por frecuencia
        n_coef_uva = (abs(coef) >= prctile(vo, 100 - round (i*i*100/length(
            vo)))).*coef; %umbral por valor absoluto
        im_recup_ufrec = idct2(n_coef_ufrec);
        im_recup_uva = idct2(n_coef_uva);
        h_ufrec = entropy(n_coef_ufrec);
        h_uva = entropy(n_coef_uva);
        [mse_ufrec, psnr_ufrec] = mse_psnr(double(sen), im_recup_ufrec);
        [mse_uva, psnr_uva] = mse_psnr(double(sen), im_recup_uva);

        %volores de la entropía y del psnr
        ens1_ufrec = [ens1_ufrec h_ufrec];
        pns1_ufrec = [pns1_ufrec psnr_ufrec];
```

```
        ens1_uva = [ens1_uva h_uva];
        pns1_uva = [pns1_uva psnr_uva];

        subplot(131);image(im_recup_ufrec);colormap(gray(256));title(
            sprintf('Porc. de coef. por frec: %3.2f',100*i*i/(length(sen)*
            length(sen))));
        axis image; axis off;
        subplot (132); image(sen);colormap(gray(256));title(sprintf('Imagen
             original'));
        axis image; axis off;
        subplot(133);image(im_recup_uva);colormap(gray(256));title(sprintf
            ('Porc. de coef. por va:%3d',round (i*i*100/length(vo))));
        axis image; axis off;

        if (i < 80), pause(0.001);
        else pause (0.0001);
        end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Transformada DCT cuantizando los coeficientes uniformemente
function [ens1, pns1] = dct_qu(filename)
sen = readpgm(filename);
coef = dct2(sen); ens1=[];pns1=[];rangoM = 500;

vec1=2.^(-55:4:0.5);vec2=(0.5:0.0625:2.5);
vec3=3:2:32;vec4=33:4:rangoM;
vec=[vec1 vec2 vec3 vec4];
%for i=2.^(-20:2:sqrt(rangoM))
for i=vec
% se cuantiza los coeficientes uniformemente
n_coef = round(coef/i)*i;
im_recup = idct2(n_coef);
[h,t] = entropy(n_coef);
[mse, psnr] = mse_psnr(double(sen), im_recup);
ens1 = [ens1 h];
pns1 = [pns1 psnr];
%     porc = 100*t/simbDist;
%     cint = round(r2(2)/i) - round(r2(1)/i) +1;
```

```
disp(i);
%subplot(121);image(im_recup);colormap(gray(256));title(sprintf('Q: %2.5f
    %%, #ints %d, simb dist: %d',porc,cint,t));
%subplot(122);image(sen);colormap(gray(256));title('imagen original');
%pause;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% tranformada con umbrales por frecuencia
function [ens1, pns1] = dct_ufrec(sen)
% sen = readpgm('amber.pgm');
coef = dct2(sen);
n_coef = zeros(size(coef));
ens1=[];pns1=[];
[m,n]= size (sen);

for i=1:m
n_coef(1:i,1:i) = coef(1:i,1:i); % umbral por frecuencia
im_recup = idct2(n_coef);
h = entropy(n_coef);
[mse, psnr] = mse_psnr(sen, im_recup);
ens1 = [ens1 h];
pns1 = [pns1 psnr];

subplot(121);image(im_recup);colormap(gray(256));title(sprintf('cuadrado %d
    , porcentaje de coeficientes: %3.3f',i,100*i*i/(length(sen)*length(sen))
    ));
subplot(122);surfc(im_recup);colormap(gray(256));title(sprintf('cuadrado %d
    ',i));
pause;
end
% se pide un gráfico de la entropía y su calidad alcanzada
plot(ens1,pns1);title('Entropia VS Pnsr')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% tranformada con umbrales por valor absoluto
function [ens1, pns1] = dct_uva(sen)
coef = dct2(sen);
% vo = - sort(-abs(coef(:))); %ordena de mayor a menor los valores
    absolutos
vo = abs(coef(:));
```

```
n_coef = zeros(size(coef));
ens1=[];pns1=[];ens2=[];pns2=[];
[m,n]= size (sen);

for i=1:10:m
n_coef = (abs(coef) >= prctile(vo, 100 - round (i*i*100/length(vo)))).*coef
    ;
im_recup = idct2(n_coef);
h = entropy(n_coef);
%p = psnr(sen,im_recup,256);
[mse, psnr] = mse_psnr(double(sen), im_recup);
ens1 = [ens1 h];
pns1 = [pns1 psnr];

subplot(121);image(im_recup);colormap(gray(256));title(sprintf('porcentaje
    de coeficientes: %3.3f',round (i*i*100/length(vo))));
subplot(122);surfc(im_recup);colormap(gray(256));title(sprintf('porcentaje
    de coeficientes: %3.3f',round (i*i*100/length(vo))));
pause
end
plot(ens1,pns1);title('Entropia VS Pnsr')
```

7.2. Principales Funciones Implementadas para la Wavelts

A continuación se muestra los códigos de las principales funciones, procedimientos para la implementación de la Wavelets:

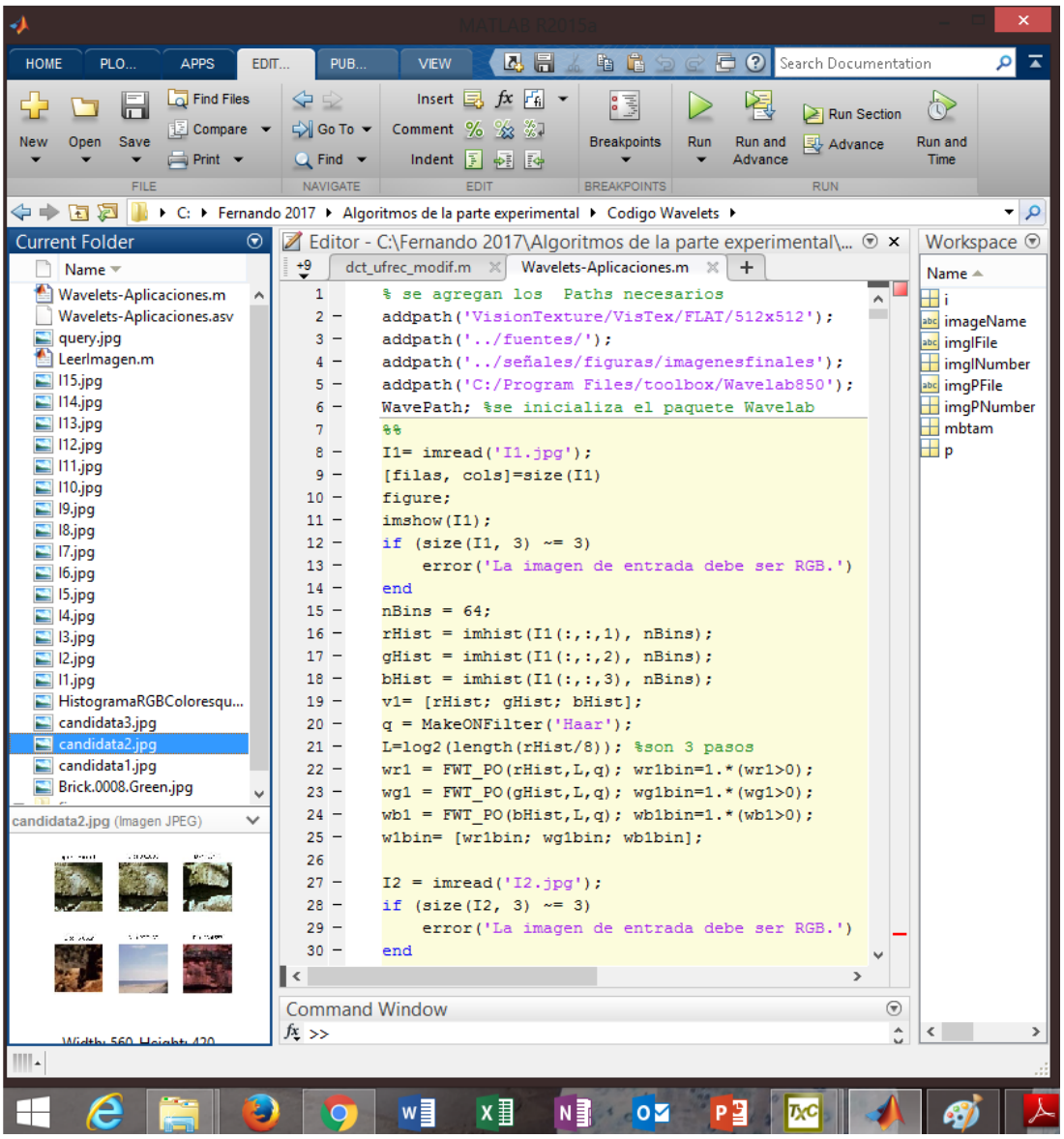


Figura 7.3: Aplicación de los coeficientes de la transformada Wavelets

```
%Agrego los Paths necesarios
addpath('VisionTexture/VisTex/FLAT/512x512');
addpath('input/');
addpath('../Fuentes/readwrite');
addpath('../Fuentes/rutinas');

%Levanto los archivos del directorio.
files = {'I1.jpg'; 'I2.jpg',...'I15.jpg'};

%Inicializamos las variables y estructuras.
rows = 512;nfiles = length(files);
nlevels = 3;natt = 3;
```

```
images = zeros(rows,rows,nfiles);
WS = zeros(rows,rows,nfiles);
attvalues = zeros(nfiles,natt*(3*nlevels + 1));

%se cargan las imágenes, se aplica la transformación y se guarda la capa o
    región de interes.
for i = 1:nfiles
        fprintf(['Leyendo ',char(files(i)),'\n']);
        rgb = imread(char(files(i)));
        ycbcr = rgb2ycbcr(rgb);
        images(:,:,i) = ycbcr(:,:,1);
end

%Se aplica la wavelet con la cantidad de niveles establecidos para cada
   imagen
qd = MakeONFilter('Daubechies',4);
L = log2(rows) - nlevels;

for i = 1:nfiles
fprintf(['Aplicando FWT2_PO ',char(files(i)),' -- ']);
WS(:,:,i) = FWT2_PO(double(images(:,:,i)),L,qd);
fprintf(['Procesando informacion...\n']);
%Se determina los valores de los atributos para cada nivel.
for j = 1:nlevels
mid = rows/(2^j) + 1;
lim = rows/(2^(j-1));

%Detalle vertical nivel j
V = WS(1:(mid-1),mid:lim,i);
x = zeros(1,3);
[x(1),x(2),x(3)] = calculateFeatures(V);
attvalues(i,(9*(j-1) + 1):(9*(j-1) + 3)) = x;

%Detalle horizontal nivel j
H = WS(mid:lim,1:(mid-1),i);
[x(1),x(2),x(3)] = calculateFeatures(H);
attvalues(i,(9*(j-1) + 4):(9*(j-1) + 6)) = x;

%Detalle diagonal nivel j
```

```
D = WS(mid:lim,mid:lim,i);
[x(1),x(2),x(3)] = calculateFeatures(D);
attvalues(i,(9*(j-1) + 7):(9*(j-1) + 9)) = x;
end

%Se observa el nivel de aproximación.
A = WS(1:(mid-1),1:(mid-1),i);
y = zeros(1,3);
[y(1),y(2),y(3)] = calculateFeatures(A);
attvalues(i,(9*nlevels + 1):(9*nlevels + 3)) = x;
end

%Se realiza el cálculo de las distancias entre todas las imagenes.
dist = zeros(nfiles,nfiles);
for i = 1:nfiles
        for j = 1:nfiles
                 if (i ~= j)
                dist(i,j) = norm(attvalues(i,:) - attvalues(j,:),2);
                 end
        end
end

%Se ordena los valores en forma creciente en cada caso.
sortedDist = zeros(size(dist));
sortedidx = zeros(size(dist));
[sortedDist, sortedidx] = sort(dist,2,'ascend');

%Se muestra el ordenamiento entre las imágenes en estudio, según la
   distancia seleccionada.
for i = 1:1
         figure;
         for j = 1:nfiles
        subplot(1,nfiles,j)
        image = imread(char(files(sortedidx(i,j))));
        imshow(image);
        %title([files(sortedidx(i,j)),' - Dist: ',num2str(sortedDist(i,j))
           ]);
        title(['Dist: ',num2str(sortedDist(i,j))]);
         end
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure;
imshow(I);
I1 = I(1:128,1:128,:);I2 = I(1:128,129:256,:);
I3 = I(129:256,1:128,:); I4 = I(374:502,129:256,:);
I5 = I(275:402,1:128,:); I6 = I(257:384,257:384,:);
size(I4), size(I5), size(I6)
imwrite(I1,'I1.jpg','jpg'); imwrite(I2,'I2.jpg','jpg');
imwrite(I3,'I3.jpg','jpg'); imwrite(I4,'I4.jpg','jpg');
imwrite(I5,'I5.jpg','jpg'); imwrite(I6,'I6.jpg','jpg');

figure;
subplot(221);image(I1);title('I1');
subplot(222);image(I2);title('I2');
subplot(223);image(I3);title('I3');
subplot(224);image(I4);title('I4');

figure;
subplot(221);image(I5);title('I5');
subplot(222);image(I6);title('I6');

I= imread('query2.ppm');
figure;
imshow(I);
I7 = I(1:128,1:128,:);    I8 = I(1:128,129:256,:);
I9 = I(129:256,1:128,:); I10 = I(129:256,1:128,:);
I11 = I(257:385,1:128,:); I12 = I(257:385,257:385,:);
imwrite(I7,'I7.jpg','jpg'); imwrite(I8,'I8.jpg','jpg');
imwrite(I9,'I9.jpg','jpg'); imwrite(I10,'I10.jpg','jpg');
imwrite(I11,'I11.jpg','jpg'); imwrite(I12,'I12.jpg','jpg');

figure;
subplot(221);image(I7);title('I7');
subplot(222);image(I8);title('I8');
subplot(223);image(I9);title('I9');
subplot(224);image(I10);title('I10');

figure;
```

```
subplot(221);image(I11);title('I11');
subplot(222);image(I12);title('I12');

I= imread('query.ppm');
figure;
imshow(I);
I13  = I(1:128,1:128,:);    I14 = I(1:128,129:256,:);
I15 = I(129:256,1:128,:); I16 = I(129:256,1:128,:);
imwrite(I13,'I13.jpg','jpg');   imwrite(I14,'I14.jpg','jpg');
imwrite(I15,'I15.jpg','jpg');   imwrite(I16,'I16.jpg','jpg');

figure;
subplot(221);image(I13);title('I13');
subplot(222);image(I14);title('I14');
subplot(223);image(I15);title('I15');
subplot(224);image(I16);title('I16');

figure;
subplot(221);image(I15);title('I17');
subplot(222);image(I16);title('I18');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%se agregan los  Paths necesarios
addpath('VisionTexture/VisTex/FLAT/512x512');
addpath('../fuentes/');
addpath('../señales/figuras/imagenesfinales');
addpath('C:/Program Files/toolbox/Wavelab850');
WavePath; %se inicializa el paquete Wavelab

I1= imread('I1.jpg');
[filas, cols]=size(I1)
figure;
imshow(I1);
if (size(I1, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
nBins = 64;
rHist = imhist(I1(:,:,1), nBins);
gHist = imhist(I1(:,:,2), nBins);
```

```
bHist = imhist(I1(:,:,3), nBins);
v1= [rHist; gHist; bHist];
q = MakeONFilter('Haar');
L=log2(length(rHist/8)); %on 3 pasos
wr1 = FWT_PO(rHist,L,q); wr1bin=1.*(wr1>0);
wg1 = FWT_PO(gHist,L,q); wg1bin=1.*(wg1>0);
wb1 = FWT_PO(bHist,L,q); wb1bin=1.*(wb1>0);
w1bin= [wr1bin; wg1bin; wb1bin];

I2 = imread('I2.jpg');
if (size(I2, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I2(:,:,1), nBins);
gHist = imhist(I2(:,:,2), nBins);
bHist = imhist(I2(:,:,3), nBins);
v2= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr2 = FWT_PO(rHist,L,q); wr2bin=1.*(wr2>0);
wg2 = FWT_PO(gHist,L,q); wg2bin=1.*(wg2>0);
wb2 = FWT_PO(bHist,L,q); wb2bin=1.*(wb2>0);
w2bin= [wr2bin; wg2bin; wb2bin];

I3 = imread('I3.jpg');
if (size(I3, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I3(:,:,1), nBins);
gHist = imhist(I3(:,:,2), nBins);
bHist = imhist(I3(:,:,3), nBins);
v3= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr3 = FWT_PO(rHist,L,q); wr3bin=1.*(wr3>0);
wg3 = FWT_PO(gHist,L,q); wg3bin=1.*(wg3>0);
wb3 = FWT_PO(bHist,L,q); wb3bin=1.*(wb3>0);
w3bin= [wr3bin; wg3bin; wb3bin];

I4 = imread('I4.jpg');
if (size(I4, 3) ~= 3)
```

```
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I4(:,:,1), nBins);
gHist = imhist(I4(:,:,2), nBins);
bHist = imhist(I4(:,:,3), nBins);
v4= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr4 = FWT_PO(rHist,L,q); wr4bin=1.*(wr4>0);
wg4 = FWT_PO(gHist,L,q); wg4bin=1.*(wg4>0);
wb4 = FWT_PO(bHist,L,q); wb4bin=1.*(wb4>0);
w4bin= [wr4bin; wg4bin; wb4bin];



I5 = imread('I5.jpg');
if (size(I5, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I5(:,:,1), nBins);
gHist = imhist(I5(:,:,2), nBins);
bHist = imhist(I5(:,:,3), nBins);
v5= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr5 = FWT_PO(rHist,L,q); wr5bin=1.*(wr5>0);
wg5 = FWT_PO(gHist,L,q); wg5bin=1.*(wg5>0);
wb5 = FWT_PO(bHist,L,q); wb5bin=1.*(wb5>0);
w5bin= [wr5bin; wg5bin; wb5bin];

I6 = imread('I6.jpg');
if (size(I6, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I6(:,:,1), nBins);
gHist = imhist(I6(:,:,2), nBins);
bHist = imhist(I6(:,:,3), nBins);
v6= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr6 = FWT_PO(rHist,L,q); wr6bin=1.*(wr6>0);
wg6 = FWT_PO(gHist,L,q); wg6bin=1.*(wg6>0);
wb6 = FWT_PO(bHist,L,q); wb6bin=1.*(wb6>0);
```

```
w6bin= [wr6bin; wg6bin; wb6bin];

I7 = imread('I7.jpg');
if (size(I7, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I7(:,:,1), nBins);
gHist = imhist(I7(:,:,2), nBins);
bHist = imhist(I7(:,:,3), nBins);
v7= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr7 = FWT_PO(rHist,L,q); wr7bin=1.*(wr7>0);
wg7 = FWT_PO(gHist,L,q); wg7bin=1.*(wg7>0);
wb7 = FWT_PO(bHist,L,q); wb7bin=1.*(wb7>0);
w7bin= [wr7bin; wg7bin; wb7bin];

I8 = imread('I8.jpg');
if (size(I8, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I8(:,:,1), nBins);
gHist = imhist(I8(:,:,2), nBins);
bHist = imhist(I8(:,:,3), nBins);
v8= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr8 = FWT_PO(rHist,L,q); wr8bin=1.*(wr8>0);
wg8 = FWT_PO(gHist,L,q); wg8bin=1.*(wg8>0);
wb8 = FWT_PO(bHist,L,q); wb8bin=1.*(wb8>0);
w8bin= [wr8bin; wg8bin; wb8bin];

I9 = imread('I9.jpg');
if (size(I9, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I9(:,:,1), nBins);
gHist = imhist(I9(:,:,2), nBins);
bHist = imhist(I9(:,:,3), nBins);
v9= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
```

```
wr9 = FWT_PO(rHist,L,q); wr9bin=1.*(wr9>0);
wg9 = FWT_PO(gHist,L,q); wg9bin=1.*(wg9>0);
wb9 = FWT_PO(bHist,L,q); wb9bin=1.*(wb9>0);
w9bin= [wr9bin; wg9bin; wb9bin];

I10 = imread('I10.jpg');
if (size(I10, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I10(:,:,1), nBins);
gHist = imhist(I10(:,:,2), nBins);
bHist = imhist(I10(:,:,3), nBins);
v10= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr10 = FWT_PO(rHist,L,q); wr10bin=1.*(wr10>0);
wg10 = FWT_PO(gHist,L,q); wg10bin=1.*(wg10>0);
wb10 = FWT_PO(bHist,L,q); wb10bin=1.*(wb10>0);
w10bin= [wr10bin; wg10bin; wb10bin];

I11 = imread('I11.jpg');
if (size(I11, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I11(:,:,1), nBins);
gHist = imhist(I11(:,:,2), nBins);
bHist = imhist(I11(:,:,3), nBins);
v11= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr11 = FWT_PO(rHist,L,q); wr11bin=1.*(wr11>0);
wg11 = FWT_PO(gHist,L,q); wg11bin=1.*(wg11>0);
wb11 = FWT_PO(bHist,L,q); wb11bin=1.*(wb11>0);
w11bin= [wr11bin; wg11bin; wb11bin];

I12 = imread('I12.jpg');
if (size(I12, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I12(:,:,1), nBins);
gHist = imhist(I12(:,:,2), nBins);
```

```
bHist = imhist(I12(:,:,3), nBins);
v12= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr12 = FWT_PO(rHist,L,q); wr12bin=1.*(wr12>0);
wg12 = FWT_PO(gHist,L,q); wg12bin=1.*(wg12>0);
wb12 = FWT_PO(bHist,L,q); wb12bin=1.*(wb12>0);
w12bin= [wr12bin; wg12bin; wb12bin];

I13 = imread('I13.jpg');
if (size(I13, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I13(:,:,1), nBins);
gHist = imhist(I13(:,:,2), nBins);
bHist = imhist(I13(:,:,3), nBins);
v1= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr13 = FWT_PO(rHist,L,q); wr13bin=1.*(wr13>0);
wg13 = FWT_PO(gHist,L,q); wg13bin=1.*(wg13>0);
wb13 = FWT_PO(bHist,L,q); wb13bin=1.*(wb13>0);
w13bin= [wr13bin; wg13bin; wb13bin];

I14 = imread('I14.jpg');
if (size(I14, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I14(:,:,1), nBins);
gHist = imhist(I14(:,:,2), nBins);
bHist = imhist(I14(:,:,3), nBins);
v14= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr14 = FWT_PO(rHist,L,q); wr14bin=1.*(wr14>0);
wg14 = FWT_PO(gHist,L,q); wg14bin=1.*(wg14>0);
wb14 = FWT_PO(bHist,L,q); wb14bin=1.*(wb14>0);
w14bin= [wr14bin; wg14bin; wb14bin];

I15 = imread('I15.jpg');
if (size(I15, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
```

```
end
rHist = imhist(I15(:,:,1), nBins);
gHist = imhist(I15(:,:,2), nBins);
bHist = imhist(I15(:,:,3), nBins);
v15= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %son 3 pasos
wr15 = FWT_PO(rHist,L,q); wr15bin=1.*(wr15>0);
wg15 = FWT_PO(gHist,L,q); wg15bin=1.*(wg15>0);
wb15 = FWT_PO(bHist,L,q); wb15bin=1.*(wb15>0);
w15bin= [wr15bin; wg15bin; wb15bin];

D1D1=pdist2(w1bin', w1bin', 'hamming')
D1D2=pdist2(w1bin', w2bin', 'hamming')
D1D3=pdist2(w1bin', w3bin', 'hamming')
D1D4=pdist2(w1bin', w4bin', 'hamming')
D1D5=pdist2(w1bin', w5bin', 'hamming')
D1D6=pdist2(w1bin', w6bin', 'hamming')
D1D7=pdist2(w1bin', w7bin', 'hamming')
D1D8=pdist2(w1bin', w8bin', 'hamming')
D1D9=pdist2(w1bin', w9bin', 'hamming')
D1D10=pdist2(w1bin', w10bin', 'hamming')
D1D11=pdist2(w1bin', w11bin', 'hamming')
D1D12=pdist2(w1bin', w12bin', 'hamming')
D1D13=pdist2(w1bin', w13bin', 'hamming')
D1D14=pdist2(w1bin', w14bin', 'hamming')
D1D15=pdist2(w1bin', w15bin', 'hamming')

vectordist = ([D1D1, D1D2, D1D3, D1D4, D1D5, D1D6, D1D7, D1D8, D1D9, D1D10,
    D1D11, D1D12, D1D13, D1D14,  D1D15].*100)
vectordistord=sort(vectordist,'ascend')

figure;imshow(I1);title(['query image 1']);
figure;imshow(I2);title([' I2: dist ' num2str(D1D2 * 100)]);
figure;imshow(I3);title([' I3: dist ' num2str(D1D3 * 100)]);
figure;imshow(I4);title([' I4: dist ' num2str(D1D4 * 100)]);
figure;imshow(I5);title([' I5: dist ' num2str(D1D5 * 100)]);
figure;imshow(I6);title([' I6: dist ' num2str(D1D6 * 100)]);
figure;imshow(I7);title([' I7: dist ' num2str(D1D7 * 100)]);
figure;imshow(I8);title([' I8: dist ' num2str(D1D8 * 100)]);
```

```
figure;imshow(I9);title(['I9: dist ' num2str(D1D9 * 100)]);
figure;imshow(I10);title(['I10: dist ' num2str(D1D10 * 100)]);
figure;imshow(I11);title(['I11: dist ' num2str(D1D11 * 100)]);
figure;imshow(I12);title(['I12: dist ' num2str(D1D12 * 100)]);
figure;imshow(I13);title(['I13: dist ' num2str(D1D13 * 100)]);
figure;imshow(I14);title(['I14: dist ' num2str(D1D14 * 100)]);
figure;imshow(I15);title(['I15: dist ' num2str(D1D15 * 100)]);

%%imagenes de consulta
query1 = imread('Brick.0008.Green.ppm');
figure;
imshow(query1);imwrite(query1,'query.jpg','jpg');

query2 = imread('BrickPaint.Red.ppm');
figure;
imshow(query2);imwrite(query2,'BrickPaint.Red.ppm','jpg');

query3 = imread('MtValley.0000.ppm');
figure;
imshow(query3);imwrite(query3,'MtValley.0000.ppm','jpg');



I1 = imread('Brick.0008.Green-1.ppm');
figure;imshow(I1);

%query en tam de 256x256
I2 = imread('Brick.0008.Green-2.ppm');
figure;imshow(I2);

I3 = imread('Brick.0008.Green-3.ppm');
figure;imshow(I3);

I4= imread('BrickPaint.Red-1.ppm');
figure;imshow(I4);

I5= imread('BrickPaint.Red-2.ppm');
figure;imshow(I5);

I6= imread('BrickPaint.Red-3.ppm');
```

```
figure;imshow(I6);

I7= imread('MtValley.0000-1.ppm');
figure;imshow(I7);

I8= imread('MtValley.0000-2.ppm');
figure;imshow(I8);

I9= imread('MtValley.0000-3.ppm');
figure;imshow(I9);

%%Histograma de valores RGBHIST de la primera query
I = imread('Brick.0008.Green.ppm'); %continuo
if (size(I, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end

nBins = 64;
rHist = imhist(I(:,:,1), nBins);
gHist = imhist(I(:,:,2), nBins);
bHist = imhist(I(:,:,3), nBins);
vHist= [rHist; gHist; bHist];
figure;
histograna = stem(1:3*nBins, vHist);
title('Histograma de las tres comp. RGB queryI');
q = MakeONFilter('Haar');
L=log2(length(rHist/8)); %son 3 pasos
%aplicó la transformada wavelet
wr = FWT_PO(rHist,L,q);
wrbin=1.*(wr>0);
wg = FWT_PO(gHist,L,q);
wgbin=1.*(wg>0);
wb = FWT_PO(bHist,L,q);
wbbin=1.*(wb>0);
wbin= [wrbin; wgbin; wbbin];
figure;
hold on;
h(1) = stem(1:nBins, rHist); title('Histograma banda R');
h(2) = stem(1:nBins + 1/3, gHist);  title('Histograma banda G');
```

```
h(3) = stem(1:nBins + 2/3, bHist);  title('Histograma de las tres comp. RGB
    queryI');
set(h, 'marker', 'none');
set(h(1), 'color', [1 0 0]);
set(h(2), 'color', [0 1 0]);
set(h(3), 'color', [0 0 1]);
D=pdist2(wbin', wbin', 'hamming')

%Histograma de valores RGBHIST de la segunda query
I = imread('BrickPaint.Red.ppm'); %continuo
if (size(I, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
nBins = 64;
rHist = imhist(I(:,:,1), nBins);
gHist = imhist(I(:,:,2), nBins);
bHist = imhist(I(:,:,3), nBins);
vHist= [rHist; gHist; bHist];
figure;
histograna = stem(1:3*nBins, vHist);
title('Histograma de las tres comp. RGB queryII');
q = MakeONFilter('Haar');
L=log2(length(rHist/8)); %son 3 pasos
%se aplica la transformada wavelet
wr = FWT_PO(rHist,L,q);
wrbin=1.*(wr>0);
wg = FWT_PO(gHist,L,q);
wgbin=1.*(wg>0);
wb = FWT_PO(bHist,L,q);
wbbin=1.*(wb>0);
wbin= [wrbin; wgbin; wbbin];
figure;
hold on;
h(1) = stem(1:nBins, rHist); title('Histograma banda R');
h(2) = stem(1:nBins + 1/3, gHist);  title('Histograma banda G');
h(3) = stem(1:nBins + 2/3, bHist);  title('Histograma de las tres comp. RGB
    queryII');
set(h, 'marker', 'none');
set(h(1), 'color', [1 0 0]);
```

```
set(h(2), 'color', [0 1 0]);
set(h(3), 'color', [0 0 1]);
D=pdist2(wbin', wbin', 'hamming')

%Histograma de valores RGBHIST de la tercera query
I = imread('BrickPaint.Red.ppm'); %continuo
if (size(I, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
nBins = 64;
rHist = imhist(I(:,:,1), nBins);
gHist = imhist(I(:,:,2), nBins);
bHist = imhist(I(:,:,3), nBins);
vHist= [rHist; gHist; bHist];
figure;
histograna = stem(1:3*nBins, vHist);
title('Histograma de las tres comp. RGB queryIII');
q = MakeONFilter('Haar');
L=log2(length(rHist/8)); %son 3 pasos
%se aplica la transformada wavelet
wr = FWT_PO(rHist,L,q);
wrbin=1.*(wr>0);
wg = FWT_PO(gHist,L,q);
wgbin=1.*(wg>0);
wb = FWT_PO(bHist,L,q);
wbbin=1.*(wb>0);
wbin= [wrbin; wgbin; wbbin];
figure;
hold on;
h(1) = stem(1:nBins, rHist); title('Histograma banda R');
h(2) = stem(1:nBins + 1/3, gHist);  title('Histograma banda G');
h(3) = stem(1:nBins + 2/3, bHist);  title('Histograma de las tres comp. RGB
    queryIII');
set(h, 'marker', 'none');
set(h(1), 'color', [1 0 0]);
set(h(2), 'color', [0 1 0]);
set(h(3), 'color', [0 0 1]);
D=pdist2(wbin', wbin', 'hamming')
```

```
I1= imread('I1');
if (size(I1, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
nBins = 128;
rHist = imhist(I1(:,:,1), nBins);
gHist = imhist(I1(:,:,2), nBins);
bHist = imhist(I1(:,:,3), nBins);
v1= [rHist; gHist; bHist];
q = MakeONFilter('Haar');
L=log2(length(rHist/8)); %on 3 pasos
wr1 = FWT_PO(rHist,L,q); wr1bin=1.*(wr1>0);
wg1 = FWT_PO(gHist,L,q); wg1bin=1.*(wg1>0);
wb1 = FWT_PO(bHist,L,q); wb1bin=1.*(wb1>0);
w1bin= [wr1bin; wg1bin; wb1bin];

I2 = imread('I2');
if (size(I2, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I2(:,:,1), nBins);
gHist = imhist(I2(:,:,2), nBins);
bHist = imhist(I2(:,:,3), nBins);
v2= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr2 = FWT_PO(rHist,L,q); wr2bin=1.*(wr2>0);
wg2 = FWT_PO(gHist,L,q); wg2bin=1.*(wg2>0);
wb2 = FWT_PO(bHist,L,q); wb2bin=1.*(wb2>0);
w2bin= [wr2bin; wg2bin; wb2bin];

I3 = imread('I3');
if (size(I3, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I3(:,:,1), nBins);
gHist = imhist(I3(:,:,2), nBins);
bHist = imhist(I3(:,:,3), nBins);
v3= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
```

```
wr3 = FWT_PO(rHist,L,q); wr3bin=1.*(wr3>0);
wg3 = FWT_PO(gHist,L,q); wg3bin=1.*(wg3>0);
wb3 = FWT_PO(bHist,L,q); wb3bin=1.*(wb3>0);
w3bin= [wr3bin; wg3bin; wb3bin];

I4 = imread('I4.jpg');
if (size(I4, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I4(:,:,1), nBins);
gHist = imhist(I4(:,:,2), nBins);
bHist = imhist(I4(:,:,3), nBins);
v4= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %son 3 pasos
wr4 = FWT_PO(rHist,L,q); wr4bin=1.*(wr4>0);
wg4 = FWT_PO(gHist,L,q); wg4bin=1.*(wg4>0);
wb4 = FWT_PO(bHist,L,q); wb4bin=1.*(wb4>0);
w4bin= [wr4bin; wg4bin; wb4bin];

I5 = imread('I5.jpg');
if (size(I5, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I5(:,:,1), nBins);
gHist = imhist(I5(:,:,2), nBins);
bHist = imhist(I5(:,:,3), nBins);
v5= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %son 3 pasos
wr5 = FWT_PO(rHist,L,q); wr5bin=1.*(wr5>0);
wg5 = FWT_PO(gHist,L,q); wg5bin=1.*(wg5>0);
wb5 = FWT_PO(bHist,L,q); wb5bin=1.*(wb5>0);
w5bin= [wr5bin; wg5bin; wb5bin];

I6 = imread('I6.jpg');
if (size(I6, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I6(:,:,1), nBins);
gHist = imhist(I6(:,:,2), nBins);
```

```
bHist = imhist(I6(:,:,3), nBins);
v6= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr6 = FWT_PO(rHist,L,q); wr6bin=1.*(wr6>0);
wg6 = FWT_PO(gHist,L,q); wg6bin=1.*(wg6>0);
wb6 = FWT_PO(bHist,L,q); wb6bin=1.*(wb6>0);
w6bin= [wr6bin; wg6bin; wb6bin];

I7 = imread('I7.jpg');
if (size(I7, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I7(:,:,1), nBins);
gHist = imhist(I7(:,:,2), nBins);
bHist = imhist(I7(:,:,3), nBins);
v7= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr7 = FWT_PO(rHist,L,q); wr7bin=1.*(wr7>0);
wg7 = FWT_PO(gHist,L,q); wg7bin=1.*(wg7>0);
wb7 = FWT_PO(bHist,L,q); wb7bin=1.*(wb7>0);
w7bin= [wr7bin; wg7bin; wb7bin];

I8 = imread('I8.jpg');
if (size(I8, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I8(:,:,1), nBins);
gHist = imhist(I8(:,:,2), nBins);
bHist = imhist(I8(:,:,3), nBins);
v8= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr8 = FWT_PO(rHist,L,q); wr8bin=1.*(wr8>0);
wg8 = FWT_PO(gHist,L,q); wg8bin=1.*(wg8>0);
wb8 = FWT_PO(bHist,L,q); wb8bin=1.*(wb8>0);
w8bin= [wr8bin; wg8bin; wb8bin];

I9 = imread('I9.jpg');
if (size(I9, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
```

```
end
rHist = imhist(I9(:,:,1), nBins);
gHist = imhist(I9(:,:,2), nBins);
bHist = imhist(I9(:,:,3), nBins);
v9= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %son 3 pasos
wr9 = FWT_PO(rHist,L,q); wr9bin=1.*(wr9>0);
wg9 = FWT_PO(gHist,L,q); wg9bin=1.*(wg9>0);
wb9 = FWT_PO(bHist,L,q); wb9bin=1.*(wb9>0);
w9bin= [wr9bin; wg9bin; wb9bin];

I10 = imread('I10.jpg');
if (size(I10, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I10(:,:,1), nBins);
gHist = imhist(I10(:,:,2), nBins);
bHist = imhist(I10(:,:,3), nBins);
v10= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %son 3 pasos
wr10 = FWT_PO(rHist,L,q); wr10bin=1.*(wr10>0);
wg10 = FWT_PO(gHist,L,q); wg10bin=1.*(wg10>0);
wb10 = FWT_PO(bHist,L,q); wb10bin=1.*(wb10>0);
w10bin= [wr10bin; wg10bin; wb10bin];

I11 = imread('I11.jpg');
if (size(I11, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I11(:,:,1), nBins);
gHist = imhist(I11(:,:,2), nBins);
bHist = imhist(I11(:,:,3), nBins);
v11= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %son 3 pasos
wr11 = FWT_PO(rHist,L,q); wr11bin=1.*(wr11>0);
wg11 = FWT_PO(gHist,L,q); wg11bin=1.*(wg11>0);
wb11 = FWT_PO(bHist,L,q); wb11bin=1.*(wb11>0);
w11bin= [wr11bin; wg11bin; wb11bin];
```

```
I12 = imread('I12.jpg');
if (size(I12, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I12(:,:,1), nBins);
gHist = imhist(I12(:,:,2), nBins);
bHist = imhist(I12(:,:,3), nBins);
v12= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %son 3 pasos
wr12 = FWT_PO(rHist,L,q); wr12bin=1.*(wr12>0);
wg12 = FWT_PO(gHist,L,q); wg12bin=1.*(wg12>0);
wb12 = FWT_PO(bHist,L,q); wb12bin=1.*(wb12>0);
w12bin= [wr12bin; wg12bin; wb12bin];

I13 = imread('p13.jpg');
if (size(I13, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I13(:,:,1), nBins);
gHist = imhist(I13(:,:,2), nBins);
bHist = imhist(I13(:,:,3), nBins);
v1= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %son 3 pasos
wr13 = FWT_PO(rHist,L,q); wr13bin=1.*(wr13>0);
wg13 = FWT_PO(gHist,L,q); wg13bin=1.*(wg13>0);
wb13 = FWT_PO(bHist,L,q); wb13bin=1.*(wb13>0);
w13bin= [wr13bin; wg13bin; wb13bin];

I14 = imread('p14.jpg');
if (size(I14, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I14(:,:,1), nBins);
gHist = imhist(I14(:,:,2), nBins);
bHist = imhist(I14(:,:,3), nBins);
v14= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %son 3 pasos
wr14 = FWT_PO(rHist,L,q); wr14bin=1.*(wr14>0);
wg14 = FWT_PO(gHist,L,q); wg14bin=1.*(wg14>0);
```

```
wb14 = FWT_PO(bHist,L,q); wb14bin=1.*(wb14>0);
w14bin= [wr14bin; wg14bin; wb14bin];

 I15 = imread('p15.jpg');
 %figure;
 %imshow(I15);
 if (size(I15, 3) ~= 3)
                 error('La imagen de entrada debe ser RGB.')
 end
 rHist = imhist(I15(:,:,1), nBins);
 gHist = imhist(I15(:,:,2), nBins);
 bHist = imhist(I15(:,:,3), nBins);
 v15= [rHist; gHist; bHist];
 L=log2(length(rHist/8)); %son 3 pasos
 wr15 = FWT_PO(rHist,L,q); wr15bin=1.*(wr15>0);
 wg15 = FWT_PO(gHist,L,q); wg15bin=1.*(wg15>0);
 wb15 = FWT_PO(bHist,L,q); wb15bin=1.*(wb15>0);
 w15bin= [wr15bin; wg15bin; wb15bin];

D1D1=pdist2(w1bin', w1bin', 'hamming')
D1D2=pdist2(w1bin', w2bin', 'hamming')
D1D3=pdist2(w1bin', w3bin', 'hamming')
D1D4=pdist2(w1bin', w4bin', 'hamming')
D1D5=pdist2(w1bin', w5bin', 'hamming')
D1D6=pdist2(w1bin', w6bin', 'hamming')
D1D7=pdist2(w1bin', w7bin', 'hamming')
D1D8=pdist2(w1bin', w8bin', 'hamming')
D1D9=pdist2(w1bin', w9bin', 'hamming')
D1D10=pdist2(w1bin', w10bin', 'hamming')
D1D11=pdist2(w1bin', w11bin', 'hamming')
D1D12=pdist2(w1bin', w12bin', 'hamming')
D1D13=pdist2(w1bin', w13bin', 'hamming')
D1D14=pdist2(w1bin', w14bin', 'hamming')
D1D15=pdist2(w1bin', w15bin', 'hamming')

vectordist = ([D1D1, D1D2, D1D3, D1D4, D1D5, D1D6, D1D7, D1D8, D1D9, D1D10,
    D1D11, D1D12, D1D13, D1D14, D1D15].*100)
vectordistord=sort(vectordist,'ascend')
```

```
figure;imshow(I1);
title(['query2: dist ' num2str(D1D1 * 100)]);
imwrite(I1,'query1.jpg','jpg');

figure;imshow(I2);
title(['I2: dist ' num2str(D1D2 * 100)]);
imwrite(I2,'query2.jpg','jpg');

figure;imshow(I3);
title(['I3: dist ' num2str(D1D3 * 100)]);
imwrite(I3,'query3.jpg','jpg');

figure;imshow(I4);
title(['I4: dist ' num2str(D1D4 * 100)]);
imwrite(I4,'Part1query1.jpg','jpg');

figure;imshow(I5);
title(['I5: dist ' num2str(D1D5 * 100)]);
imwrite(I5,'Part2query1.jpg','jpg');

figure;imshow(I6);
title(['I6: dist ' num2str(D1D6 * 100)]);
imwrite(I6,'Part3query1.jpg','jpg');

figure;imshow(I7);
title(['I7: dist ' num2str(D1D7 * 100)]);
imwrite(I7,'Part1query2.jpg','jpg');

figure;imshow(I8);
title(['I8: dist ' num2str(D1D8 * 100)]);
imwrite(I8,'Part2query2.jpg','jpg');

figure;imshow(I9);
title(['I9: dist ' num2str(D1D9 * 100)]);
imwrite(I9,'Part3query2.jpg','jpg');

figure;imshow(I10);
title(['I10: dist ' num2str(D1D10 * 100)]);
imwrite(I10,'Part1query3.jpg','jpg');
```

```
figure;imshow(I11);
title([' I11: dist ' num2str(D1D11 * 100)]);
imwrite(I11,'Part2query3.jpg','jpg');

figure;imshow(I12);
title([' I12: dist ' num2str(D1D12 * 100)]);
imwrite(I12,'Part3query3.jpg','jpg');

figure;imshow(I13);
title([' I13: dist ' num2str(D1D13 * 100)]);
imwrite(I13,'imagenchica1.jpg','jpg');

figure;imshow(I14);
title([' I14: dist ' num2str(D1D14 * 100)]);
imwrite(I14,'imagenchica2.jpg','jpg');

figure;imshow(I15);
title([' I15: dist ' num2str(D1D15 * 100)]);
imwrite(I15,'imagenchica2.jpg','jpg');

%%todas con las chicas usando el filtro de Db4
I1 = imread('Brick.0008.Green.ppm');
if (size(I1, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
nBins = 128;
rHist = imhist(I1(:,:,1), nBins);
gHist = imhist(I1(:,:,2), nBins);
bHist = imhist(I1(:,:,3), nBins);
v1= [rHist; gHist; bHist];
q = MakeONFilter('Daubechies',4);
L=log2(length(rHist/8)); %son 3 pasos
wr1 = FWT_PO(rHist,L,q); wr1bin=1.*(wr1>0);
wg1 = FWT_PO(gHist,L,q); wg1bin=1.*(wg1>0);
wb1 = FWT_PO(bHist,L,q); wb1bin=1.*(wb1>0);
w1bin= [wr1bin; wg1bin; wb1bin];

I2 = imread('BrickPaint.Red.ppm');
```

```
if (size(I2, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I2(:,:,1), nBins);
gHist = imhist(I2(:,:,2), nBins);
bHist = imhist(I2(:,:,3), nBins);
v2= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr2 = FWT_PO(rHist,L,q); wr2bin=1.*(wr2>0);
wg2 = FWT_PO(gHist,L,q); wg2bin=1.*(wg2>0);
wb2 = FWT_PO(bHist,L,q); wb2bin=1.*(wb2>0);
w2bin= [wr2bin; wg2bin; wb2bin];

I3 = imread('MtValley.0000.ppm');
if (size(I3, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I3(:,:,1), nBins);
gHist = imhist(I3(:,:,2), nBins);
bHist = imhist(I3(:,:,3), nBins);
v3= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr3 = FWT_PO(rHist,L,q); wr3bin=1.*(wr3>0);
wg3 = FWT_PO(gHist,L,q); wg3bin=1.*(wg3>0);
wb3 = FWT_PO(bHist,L,q); wb3bin=1.*(wb3>0);
w3bin= [wr3bin; wg3bin; wb3bin];

I4 = imread('I1.jpg');
if (size(I4, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I4(:,:,1), nBins);
gHist = imhist(I4(:,:,2), nBins);
bHist = imhist(I4(:,:,3), nBins);
v4= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr4 = FWT_PO(rHist,L,q); wr4bin=1.*(wr4>0);
wg4 = FWT_PO(gHist,L,q); wg4bin=1.*(wg4>0);
wb4 = FWT_PO(bHist,L,q); wb4bin=1.*(wb4>0);
```

```
w4bin= [wr4bin; wg4bin; wb4bin];


I5 = imread('I2.jpg');
if (size(I5, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I5(:,:,1), nBins);
gHist = imhist(I5(:,:,2), nBins);
bHist = imhist(I5(:,:,3), nBins);
v5= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %son 3 pasos
wr5 = FWT_PO(rHist,L,q); wr5bin=1.*(wr5>0);
wg5 = FWT_PO(gHist,L,q); wg5bin=1.*(wg5>0);
wb5 = FWT_PO(bHist,L,q); wb5bin=1.*(wb5>0);
w5bin= [wr5bin; wg5bin; wb5bin];

I6 = imread('I3.jpg');
if (size(I6, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I6(:,:,1), nBins);
gHist = imhist(I6(:,:,2), nBins);
bHist = imhist(I6(:,:,3), nBins);
v6= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %son 3 pasos
wr6 = FWT_PO(rHist,L,q); wr6bin=1.*(wr6>0);
wg6 = FWT_PO(gHist,L,q); wg6bin=1.*(wg6>0);
wb6 = FWT_PO(bHist,L,q); wb6bin=1.*(wb6>0);
w6bin= [wr6bin; wg6bin; wb6bin];

I7 = imread('I4.jpg');
if (size(I7, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I7(:,:,1), nBins);
gHist = imhist(I7(:,:,2), nBins);
bHist = imhist(I7(:,:,3), nBins);
v7= [rHist; gHist; bHist];
```

```
L=log2(length(rHist/8)); %on 3 pasos
wr7 = FWT_PO(rHist,L,q); wr7bin=1.*(wr7>0);
wg7 = FWT_PO(gHist,L,q); wg7bin=1.*(wg7>0);
wb7 = FWT_PO(bHist,L,q); wb7bin=1.*(wb7>0);
w7bin= [wr7bin; wg7bin; wb7bin];

I8 = imread('I5.jpg');
if (size(I8, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I8(:,:,1), nBins);
gHist = imhist(I8(:,:,2), nBins);
bHist = imhist(I8(:,:,3), nBins);
v8= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr8 = FWT_PO(rHist,L,q); wr8bin=1.*(wr8>0);
wg8 = FWT_PO(gHist,L,q); wg8bin=1.*(wg8>0);
wb8 = FWT_PO(bHist,L,q); wb8bin=1.*(wb8>0);
w8bin= [wr8bin; wg8bin; wb8bin];

I9 = imread('I6.jpg');
if (size(I9, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I9(:,:,1), nBins);
gHist = imhist(I9(:,:,2), nBins);
bHist = imhist(I9(:,:,3), nBins);
v9= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr9 = FWT_PO(rHist,L,q); wr9bin=1.*(wr9>0);
wg9 = FWT_PO(gHist,L,q); wg9bin=1.*(wg9>0);
wb9 = FWT_PO(bHist,L,q); wb9bin=1.*(wb9>0);
w9bin= [wr9bin; wg9bin; wb9bin];

I10 = imread('I7.jpg');
if (size(I10, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I10(:,:,1), nBins);
```

```
gHist = imhist(I10(:,:,2), nBins);
bHist = imhist(I10(:,:,3), nBins);
v10= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr10 = FWT_PO(rHist,L,q); wr10bin=1.*(wr10>0);
wg10 = FWT_PO(gHist,L,q); wg10bin=1.*(wg10>0);
wb10 = FWT_PO(bHist,L,q); wb10bin=1.*(wb10>0);
w10bin= [wr10bin; wg10bin; wb10bin];

I11 = imread('I8.jpg');
if (size(I11, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I11(:,:,1), nBins);
gHist = imhist(I11(:,:,2), nBins);
bHist = imhist(I11(:,:,3), nBins);
v11= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr11 = FWT_PO(rHist,L,q); wr11bin=1.*(wr11>0);
wg11 = FWT_PO(gHist,L,q); wg11bin=1.*(wg11>0);
wb11 = FWT_PO(bHist,L,q); wb11bin=1.*(wb11>0);
w11bin= [wr11bin; wg11bin; wb11bin];

I12 = imread('I9.jpg');
if (size(I12, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I12(:,:,1), nBins);
gHist = imhist(I12(:,:,2), nBins);
bHist = imhist(I12(:,:,3), nBins);
v12= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr12 = FWT_PO(rHist,L,q); wr12bin=1.*(wr12>0);
wg12 = FWT_PO(gHist,L,q); wg12bin=1.*(wg12>0);
wb12 = FWT_PO(bHist,L,q); wb12bin=1.*(wb12>0);
w12bin= [wr12bin; wg12bin; wb12bin];

I13 = imread('p1.jpg');
if (size(I13, 3) ~= 3)
```

```
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I13(:,:,1), nBins);
gHist = imhist(I13(:,:,2), nBins);
bHist = imhist(I13(:,:,3), nBins);
v1= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr13 = FWT_PO(rHist,L,q); wr13bin=1.*(wr13>0);
wg13 = FWT_PO(gHist,L,q); wg13bin=1.*(wg13>0);
wb13 = FWT_PO(bHist,L,q); wb13bin=1.*(wb13>0);
w13bin= [wr13bin; wg13bin; wb13bin];

I14 = imread('p2.jpg');
if (size(I14, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I14(:,:,1), nBins);
gHist = imhist(I14(:,:,2), nBins);
bHist = imhist(I14(:,:,3), nBins);
v14= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr14 = FWT_PO(rHist,L,q); wr14bin=1.*(wr14>0);
wg14 = FWT_PO(gHist,L,q); wg14bin=1.*(wg14>0);
wb14 = FWT_PO(bHist,L,q); wb14bin=1.*(wb14>0);
w14bin= [wr14bin; wg14bin; wb14bin];

I15 = imread('p3.jpg');
if (size(I15, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I15(:,:,1), nBins);
gHist = imhist(I15(:,:,2), nBins);
bHist = imhist(I15(:,:,3), nBins);
v15= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr15 = FWT_PO(rHist,L,q); wr15bin=1.*(wr15>0);
wg15 = FWT_PO(gHist,L,q); wg15bin=1.*(wg15>0);
wb15 = FWT_PO(bHist,L,q); wb15bin=1.*(wb15>0);
w15bin= [wr3bin; wg3bin; wb15bin];
```

```
I16 = imread('p4.jpg');
if (size(I16, 3) ~= 3)
        error('La imagen de entrada debe ser RGB.')
end
rHist = imhist(I16(:,:,1), nBins);
gHist = imhist(I16(:,:,2), nBins);
bHist = imhist(I16(:,:,3), nBins);
v16= [rHist; gHist; bHist];
L=log2(length(rHist/8)); %on 3 pasos
wr16 = FWT_PO(rHist,L,q); wr16bin=1.*(wr16>0);
wg16 = FWT_PO(gHist,L,q); wg16bin=1.*(wg16>0);
wb16 = FWT_PO(bHist,L,q); wb16bin=1.*(wb16>0);
w16bin= [wr16bin; wg16bin; wb16bin];

D1D1=pdist2(w1bin', w1bin', 'hamming')
D1D2=pdist2(w1bin', w2bin', 'hamming')
D1D3=pdist2(w1bin', w3bin', 'hamming')
D1D4=pdist2(w1bin', w4bin', 'hamming')
D1D5=pdist2(w1bin', w5bin', 'hamming')
D1D6=pdist2(w1bin', w6bin', 'hamming')
D1D7=pdist2(w1bin', w7bin', 'hamming')
D1D8=pdist2(w1bin', w8bin', 'hamming')
D1D9=pdist2(w1bin', w9bin', 'hamming')
D1D10=pdist2(w1bin', w10bin', 'hamming')
D1D11=pdist2(w1bin', w11bin', 'hamming')
D1D12=pdist2(w1bin', w12bin', 'hamming')
D1D13=pdist2(w1bin', w13bin', 'hamming')
D1D14=pdist2(w1bin', w14bin', 'hamming')
D1D15=pdist2(w1bin', w15bin', 'hamming')
D1D16=pdist2(w1bin', w16bin', 'hamming')

vectordist = ([D1D1, D1D2, D1D3, D1D4, D1D5, D1D6, D1D7, D1D8, D1D9, D1D10,
    D1D11, D1D12, D1D13, D1D14, D1D15, D1D16].*100)
vectordistord=sort(vectordist,'ascend')

figure;imshow(I1);
title([' query1: dist ' num2str(D1D1 * 100)]);
imwrite(I1,'query1.jpg','jpg');
```

```
figure;imshow(I2);
title([' I2: dist ' num2str(D1D2 * 100)]);
imwrite(I2,'query2.jpg','jpg');

figure;imshow(I3);
title([' I3: dist ' num2str(D1D3 * 100)]);
imwrite(I3,'query3.jpg','jpg');

figure;imshow(I4);
title([' I4: dist ' num2str(D1D4 * 100)]);
imwrite(I4,'Part1query1.jpg','jpg');

figure;imshow(I5);
title([' I5: dist ' num2str(D1D5 * 100)]);
imwrite(I5,'Part2query1.jpg','jpg');

figure;imshow(I6);
title([' I6: dist ' num2str(D1D6 * 100)]);
imwrite(I6,'Part3query1.jpg','jpg');

figure;imshow(I7);
title([' I7: dist ' num2str(D1D7 * 100)]);
imwrite(I7,'Part1query2.jpg','jpg');

figure;imshow(I8);
title([' I8: dist ' num2str(D1D8 * 100)]);
imwrite(I8,'Part2query2.jpg','jpg');

figure;imshow(I9);
title([' I9: dist ' num2str(D1D9 * 100)]);
imwrite(I9,'Part3query2.jpg','jpg');

figure;imshow(I10);
title([' I10: dist ' num2str(D1D10 * 100)]);
imwrite(I10,'Part1query3.jpg','jpg');

figure;imshow(I11);
title([' I11: dist ' num2str(D1D11 * 100)]);
```

```
imwrite(I11,'Part2query3.jpg','jpg');

figure;imshow(I12);
title(['I12: dist ' num2str(D1D12 * 100)]);
imwrite(I12,'Part3query3.jpg','jpg');

figure;imshow(I13);
title(['I13: dist ' num2str(D1D13 * 100)]);
imwrite(I13,'imagenchica1.jpg','jpg');

figure;imshow(I14);
title(['I14: dist ' num2str(D1D14 * 100)]);
imwrite(I14,'imagenchica2.jpg','jpg');

figure;imshow(I15);
title(['I15: dist ' num2str(D1D15 * 100)]);
imwrite(I15,'imagenchica3.jpg','jpg');

figure;imshow(I16);
title(['I16: dist ' num2str(D1D16 * 100)]);
imwrite(I16,'imagenchica4.jpg','jpg');
```

7.3. Algoritmos y Principales Funciones Implementadas para la Estimación de Movimientos en Matlab

A continuación, mostramos las principales funciones utilizadas en el proceso de implementación de los algoritmos heurísticos de coincidencia para la estimación de movimientos en comprensión de imágenes, entre ellas tenemos:

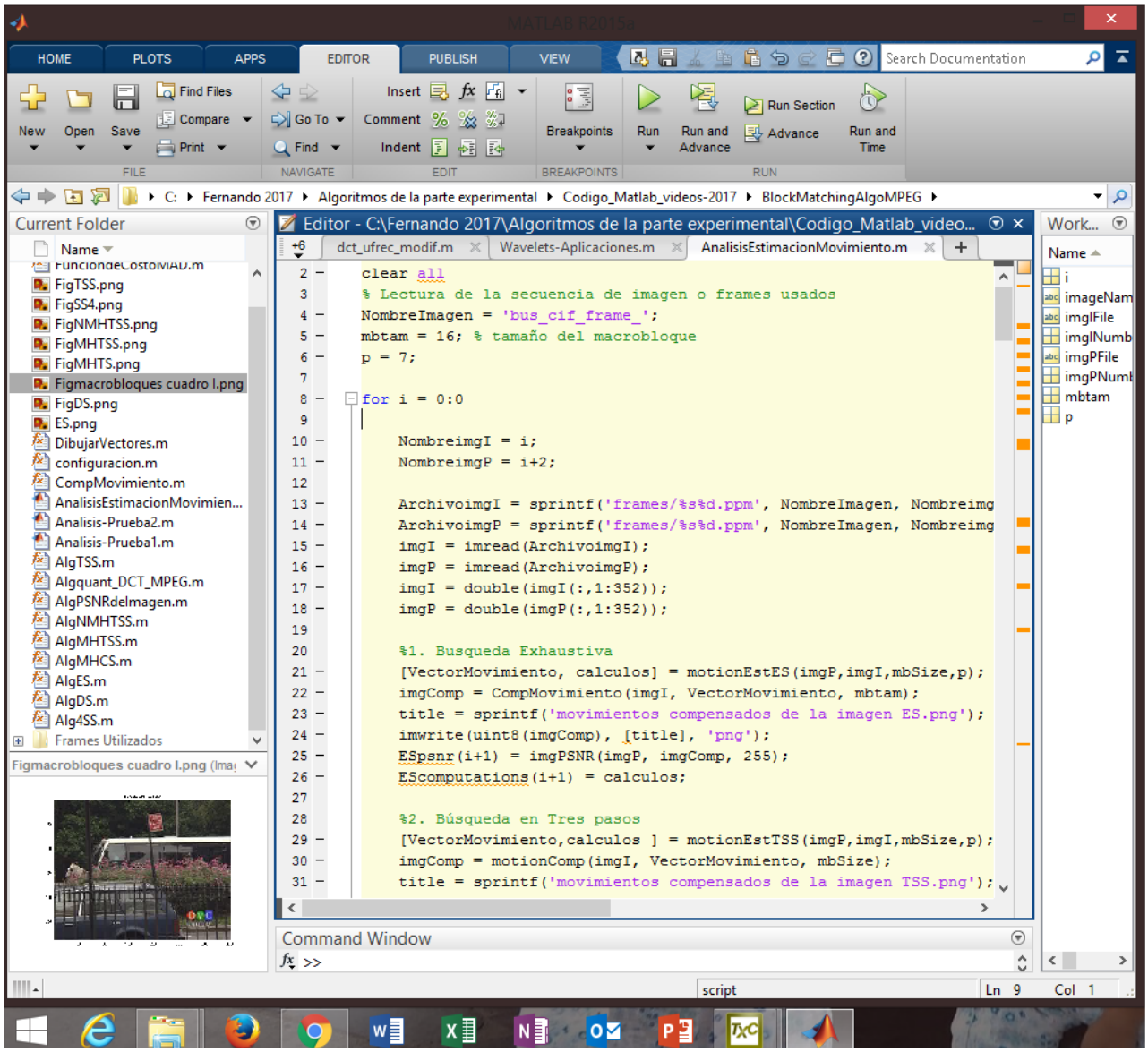


Figura 7.4: Implementación de los algoritmos heurísticos

```
close all
clear all
% Lectura de la secuencia de imagen o frames usados
NombreImagen = 'bus_cif_frame_';
mbtam = 16; % tamaño del macrobloque
p = 7;
for i = 0:0
    NombreimgI = i;
    NombreimgP = i+2;
        ArchivoimgI = sprintf('frames/%s%d.ppm', NombreImagen, NombreimgI);
        ArchivoimgP = sprintf('frames/%s%d.ppm', NombreImagen, NombreimgP);
    imgI = imread(ArchivoimgI);
```

```
imgP = imread(ArchivoimgP);
imgI = double(imgI(:,1:352));
imgP = double(imgP(:,1:352));

% Algoritmo Busqueda Exhaustiva
[VectorMovimiento, calculos] = EstMovimientoES(imgP,imgI,mbSize,p);
imgComp = CompMovimiento(imgI, VectorMovimiento, mbtam);
title = sprintf('movimientos compensados de la ??imagen ES.png');
imwrite(uint8(imgComp), [title], 'png');
ESpsnr(i+1) = imgPSNR(imgP, imgComp, 255);
calculosES(i+1) = calculos;

%2. Algoritmo Búsqueda en Tres pasos
[VectorMovimiento,calculos ] = EstMovimientoTSS(imgP,imgI,mbSize,p);
imgComp = motionComp(imgI, VectorMovimiento, mbSize);
title = sprintf('movimientos compensados ??de la imagen TSS.png');
imwrite(uint8(imgComp), [title], 'png');
TSSpsnr(i+1) = imgPSNR(imgP, imgComp, 255);
calculosTSS(i+1) = calculos;

% Algoritmo heurístico búsqueda en tres pasos simple y eficiente
[VectorMovimiento, calculos] = EstMovimientoMHTSS(imgP,imgI,mbSize,p);
imgComp = motionComp(imgI, VectorMovimiento, mbSize);
title = sprintf('movimientos compensados de la ??imagen SESTSS.png');
imwrite(uint8(imgComp), [title], 'png');
MHTSSpsnr(i+1) = imgPSNR(imgP, imgComp, 255);
calculosMHTSS(i+1) = calculos;

% Algoritmo de búsqueda en cuatro pasos
[VectorMovimiento, calculos] = EstMovimiento4SS(imgP,imgI,mbSize,p);
imgComp = motionComp(imgI, VectorMovimiento, mbSize);
title = sprintf('movimientos compensados de la ??imagen SS4.png');
imwrite(uint8(imgComp), [title], 'png');
4SSpsnr(i+1) = imgPSNR(imgP, imgComp, 255);
calculos4SS(i+1) = calculos;

% Algoritmo de búsqueda en diamante
[VectorMovimiento, calculos] = EstMovimientoDS(imgP,imgI,mbSize,p);
imgComp = motionComp(imgI, VectorMovimiento, mbSize);
```

```
    title = sprintf('movimientos compensados de la ??imagen DS.png');
    imwrite(uint8(imgComp), [title], 'png');
    DSpsnr(i+1) = imgPSNR(imgP, imgComp, 255);
    DScomputations(i+1) = calculos;

    %Algoritmo heurístico mejorado en el patrón de búsqueda en tres pasos
    [VectorMovimiento,calculos ] = EstMovimientoMHCS(imgP,imgI,mbSize,p);
    imgComp = motionComp(imgI, VectorMovimiento, mbSize);
    title = sprintf('movimientos compensados de la ??imagen NTSS.png');
    imwrite(uint8(imgComp), [title], 'png');
    MHCSpsnr(i+1) = imgPSNR(imgP, imgComp, 255);
    MHCScomputations(i+1) = calculos;

end
%%Gráfico del PSNR con los datos obtenidos
calculosES = S.calculosES; calculosTSS = S.calculosTSS;
calculosMHTSS = S.calculosMHTSS; calculos4SS = S.calculos4SS;
calculosDS = S.calculosDS; calculosMHCS = S.calculosMHCS;

ESpsnr = S.ESpsnr; TSSpsnr = S.TSSpsnr;
MHTSSpsnr = S.MHTSSpsnr; 4SSpsnr = S.4SSpsnr;
DSpsnr = S.DSpsnr; MHCSpsnr = S.ARPSpsnr;



figure; plot(ESpsnr, 'b.-'); legend('ES');

hold on; plot(TSSpsnr, 'r.-'); legend('TSS');

hold on; plot(MHTSSpsnr, 'm.-'); legend('MHTSS');

hold on; plot(4SSpsnr, 'k.-'); legend('4SS');

hold on; plot(DSpsnr, 'g.-'); legend('DS');

hold on; plot(MHCSpsnr, 'c.-'); legend('MHCS');

xlabel('Número de Cuadro');  ylabel('PSNR');
%title('Block Sike 16 x 16 search parameter p=7');
legend('ES', 'TSS', 'MHTSS', '4SS', 'DS', 'MHCS');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ALGORITMO DE BÚSQUEDA EXHAUSTIVA
function [vectorMovimeto, calculosES] = EstMovimientoES(imgP, imgI, mbSize,
    p)
[fila columna] = Tam(imgI);
vectores = ceros(2,fila*columna/mbTam^2);
costos = unos(2*p + 1, 2*p +1) * 65537;
calculos = 0;
ContadorMb = 1;
for i = 1 : mbTam : fila-mbTam+1
    for j = 1 : mbTam : columna-mbTam+1
        % la búsqueda completa inicia aquí
        % se evalua el costo de los bloques (2p + 1) verticalmente
        % y (2p + 1)  horizontal y
        %m es el índice de fila (vertical)
        %n es índice columna (horizontal)
        % es decir, que se esta escaneando en orden del frame de referencia

        for m = -p : p
            for n = -p : p
                refBlkVer = i + m;    % vector fila para el bloque de
                    referencia
                refBlkHor = j + n;    % coordenada de la columna/
                    Horizontal
                if ( refBlkVer < 1 || refBlkVer+mbSize-1 > fila ...
                        || refBlkHor < 1 || refBlkHor+mbSize-1 > columna)
                    continue;
                end
                costos(m+p+1,n+p+1) = FuncionCostoMAD(imgP(i:i+mbSize-1,j:j
                    +TamMB-1), ...
                      imgI(refBlkVer:refBlkVer+TamMB-1, refBlkHor:refBlkHor+
                        TamMB-1), TamMB);
                calculos = calculos + 1;

            end
        end
        % Se determina el vector de costo es mínimo
        % y se almacena ... es lo que devuelve como salida
```

```
        [dx, dy, min] = Costominimo(costos);  % se determina qué
            macrobloque en la imgI dió Costo mínimo
        vectores(1,ContadorMb) = dy-p-1;
        vectores(2,ContadorMb) = dx-p-1;
        ContadorMb = ContadorMb + 1;
        costos = ceros(2*p + 1, 2*p +1) * 65537;
    end
end

vectorMovimeto = vectores;
calculosES = calculos/(ContadorMb - 1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ALGORITMO HEURÍSTCO DE BÚSQUEDA EN TRES PASOS
function [VectorMovimiento, calculosTSS] = EstMovimientoTSS(imgP, imgI,
    TamMB, p)
[fila columna] = size(imgI);
vectores = ceros(2,fila*columna/mbTam^2);
costos = unos(3, 3) * 65537;
calculos = 0;

% se toma log a la base 2 de p para obtener el número de pasos requeridos
L = floor(log10(p+1)/log10(2));
MaxPasos = 2^(L-1);

% se inicia desde la esquina superior izquierda de la imagen, caminaremos
% en pasos según el tamaño del macrobloque
% por cada macrobloque que se observa  se busca una coincidencia cercana p
    píxeles
% a la izquierda, derecha, arriba y abajo de ella, es decir en cualquiera
    de
% las direcciones
contadorMB = 1;
for i = 1 : mbTam : fila-mbTam+1
    for j = 1 : mbTam : columna-mbTam+1
        % de los primeros pasos de búsqueda en los tres pasos
        % se evalua 9 elementos en cada uno de los pasos
        x = j;
```

```
y = i;
%Para evitar el calculo del punto central de la búsqueda
%una y otra vez siempre se almacena el valor anterior.
%Para la primera iteración almacenamos este valor fuera
%del ciclo for, pero para iteraciones posteriores almacenamos el
    costo en
%del punto donde vamos a cambiar raíz o patrón a seguir.
costos(2,2) = FuncionCostosMAD(imgP(i:i+mbTam-1,j:j+mbTam-1), ...
                               imgI(i:i+mbTam-1,j:j+mbTam-1),mbTam);
calculos = calculos + 1;
TamPaso = MaxPasos;

while(TamPaso >= 1)
    %m es el indice de la filas(vertical)
    %n es el indice de la columna(horizontal)
    %de esta forma, se está escaneando los frame en orden
    for m = -TamPaso : TamPaso : TamPaso
        for n = -TamPaso : TamPaso : TamPaso
            refBlkVer = y + m;
            refBlkHor = x + n;     %coordenadas de las columnas/
                Horizontal
            if ( refBlkVer < 1 || refBlkVer+mbTam-1 > fila ...
                || refBlkHor < 1 || refBlkHor+mbTam-1 > columna)
                continue;
            end
            costRow = m/TamPaso + 2;
            costCol = n/TamPaso + 2;
            if (costRow == 2 && costCol == 2)
                continue
            end
            costos(costosFila, costosColumna ) = FuncionCostosMAD(
                imgP(i:i+mbTam-1,j:j+mbTam-1), ...
                imgI(refBlkVer:refBlkVer+mbTam-1, refBlkHor:
                    refBlkHor+mbTam-1), mbTam);
            calculos = calculos + 1;
        end
    end

    %Se determina el vector de costo es mínimo
```

```
            % se almacena ... y se devuelve como salida

            [dx, dy, min] = costominimo(costos);   % determina que
                macrobloque en la imagen dió el costo mínimo

            %cambie el patrón de búsqueda en la ventana, con respecto el
                nuevo punto mínimo

            x = x + (dx-2)*TamPaso;
            y = y + (dy-2)*TamPaso;

            TamPaso = TamPaso / 2;
            costos(2,2) = costos(dy,dx);

        end
        vectores(1,ContadorMb) = y - i;
        vectores(2,ContadorMb) = x - j;
        ContadorMb = ContadorMb + 1;
        costos = ones(3,3) * 65537;
    end
end

motionVect = vectores;
calculosTSS = calculos/(ContadorMb - 1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ALGORITMO HEURÍSTCO DE BÚSQUEDA EN CUATRO PASOS
function [VectorMovimiento, calculos4SS] = EstMovimiento4SS(imgP, imgI,
    TamMB, p)
[fila columna] = Tam(imgI);

vectores = ceros(2,fila*col/TamMb^2);
costos = unos(3, 3) * 65537;
calculos = 0;
ContadorMb = 1;
for i = 1 : TamMb : fila-TamMb+1
    for j = 1 : TamMb : col-TamMb+1
        % inicia la búsqueda en cuatro pasos y imprime en orden los frame
        x = j;
```

```
y = i;
costos(2,2) = costFuncMAD(imgP(i:i+TamMb-1,j:j+TamMb-1), ...
                          imgI(i:i+TamMb-1,j:j+TamMb-1),TamMb);
calculos = calculos + 1;
        % se determina los nueves puntos como en el primer
                                                    % paso y se
                                                      evalúa
                                                    los
                                                    nueves
                                                    puntos
for m = -2 : 2 : 2
    for n = -2 : 2 : 2
        refBlkVer = y + m;
        refBlkHor = x + n;
        if ( refBlkVer < 1 || refBlkVer+TamMb-1 > fila ...
             || refBlkHor < 1 || refBlkHor+TamMb-1 > col)
             continue;
        end

        costosfila = m/2 + 2;
        costoscolumna = n/2 + 2;
        if (costosfila == 2 && costCol == 2)
            continue
        end
        costos(costosfila, costCol ) = costFuncMAD(imgP(i:i+TamMb
            -1,j:j+TamMb-1), ...
            imgI(refBlkVer:refBlkVer+TamMb-1, refBlkHor:refBlkHor+
                TamMb-1), TamMb);
        calculos = calculos + 1;

    end
end

[dx, dy, cost] = minCost(costos);       % se determina el
    macrobloque imgI que tiene costo mínimo asociado

if (dx == 2 && dy == 2)
    flag_4ss = 1;
else
```

```
    flag_4ss = 0;
    xLast = x;
    yLast = y;
    x = x + (dx-2)*2;
    y = y + (dy-2)*2;
end
costos = ones(3,3) * 65537;
costos(2,2) = cost;

stage = 1;
while (flag_4ss == 0 && stage <=2)
    for m = -2 : 2 : 2
        for n = -2 : 2 : 2
            refBlkVer = y + m;   % fila/Vert co-ordinate for ref
                block
            refBlkHor = x + n;   % col/Horizontal co-ordinate
            if ( refBlkVer < 1 || refBlkVer+TamMb-1 > fila ...
                || refBlkHor < 1 || refBlkHor+TamMb-1 > col)
                continue;
            end

            if (refBlkHor >= xLast - 2 && refBlkHor <= xLast + 2
                ...
                   && refBlkVer >= yLast - 2 && refBlkVer <= yLast
                       + 2 )
                continue;
            end

            costosfila = m/2 + 2;
            costCol = n/2 + 2;
            if (costosfila == 2 && costCol == 2)
                continue
            end

            costos(costosfila, costCol ) = costFuncMAD(imgP(i:i+
                TamMb-1,j:j+TamMb-1), ...
                                        imgI(refBlkVer:refBlkVer+TamMb
                                            -1, ...
                                           refBlkHor:refBlkHor+TamMb-1),
```

```
                                                    TamMb);
                calculos = calculos + 1;

            end
        end
        [dx, dy, cost] = minCost(costos);
        if (dx == 2 && dy == 2)
            flag_4ss = 1;
        else
            flag_4ss = 0;
            xLast = x;
            yLast = y;
            x = x + (dx-2)*2;
            y = y + (dy-2)*2;
        end
        costos = unos(3,3) * 65537;
        costos(2,2) = costos;
        stage = stage + 1;
    end
    for m = -1 : 1 : 1
        for n = -1 : 1 : 1
            refBlkVer = y + m;   % fila/Vert co-ordinate for ref block
            refBlkHor = x + n;   % col/Horizontal co-ordinate
            if ( refBlkVer < 1 || refBlkVer+TamMb-1 > fila ...
                 || refBlkHor < 1 || refBlkHor+TamMb-1 > col)
                 continue;
            end

            costosfila = m + 2;
            costCol = n + 2;
            if (costosfila == 2 && costCol == 2)
                continue
            end
            costos(costosfila, costoscolumna ) = costFuncMAD(imgP(i:i+
                TamMb-1,j:j+TamMb-1), ...
            imgI(refBlkVer:refBlkVer+TamMb-1, refBlkHor:refBlkHor+TamMb
                -1), TamMb);
            calculos = calculos + 1;
        end
```

```
        end
        % se determina el costo mínimo y se almacena aquí
        [dx, dy, cost] = costosminimo(costos);
        x = x + dx - 2;
        y = y + dy - 2;
        vectores(1,ContadorMb) = y - i;     % fila co-ordinate for the
            vector
        vectores(2,ContadorMb) = x - j;     % col co-ordinate for the vector
        ContadorMb = ContadorMb + 1;
        costos = ones(3,3) * 65537;
    end
end

VectorMovimiento = vectores;
calculos4SS = calculos/(ContadorMb - 1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ALGORITMO HEURÍSTCO DE BÚSQUEDA MHTSS
function [VectorMovimiento, MHTSS] = motionEstSESTSS(imgP, imgI, TamMb, p)
[fila columna] = Tam(imgI);
vectores = ceros(2,fila*columna/TamMb^2);
L = floor(log10(p+1)/log10(2));
pasoMax =  2^(L-1);
costos = unos(1,6)*65537;
calculos = 0;
contadorMb = 1;
for i = 1 : TamMb : fila-TamMb+1
    for j = 1 : TamMb : columna-TamMb+1
        TamPaso = PasoMax;
        x = j;
        y = i;
        while (TamPaso >= 1)
            refBlkVerPointA = y;
            refBlkHorPointA = x;
            refBlkVerPointB = y;
            refBlkHorPointB = x + TamPaso;
            refBlkVerPointC = y + TamPaso;
            refBlkHorPointC = x;
            if ( refBlkVerPointA < 1 || refBlkVerPointA+TamMb-1 > fila ...
```

```
        || refBlkHorPointA < 1 || refBlkHorPointA+TamMb-1 >
            columna)
    % no se hace nada %
else
    costos(1) = FuncionCostosMAD(imgP(i:i+TamMb-1,j:j+TamMb-1),
        ...
                imgI(refBlkVerPointA:refBlkVerPointA+TamMb-1,
                    ...
                refBlkHorPointA:refBlkHorPointA+TamMb-1),
                    TamMb);
    calculos = calculos + 1;
end
if ( refBlkVerPointB < 1 || refBlkVerPointB+TamMb-1 > fila ...
        || refBlkHorPointB < 1 || refBlkHorPointB+TamMb-1 >
            columna)
    % no se hace nada %
else
    costos(2) = FuncionCostosMAD(imgP(i:i+TamMb-1,j:j+TamMb-1),
        ...
            imgI(refBlkVerPointB:refBlkVerPointB+TamMb-1, ...
                refBlkHorPointB:refBlkHorPointB+TamMb-1), TamMb
                    );
    calculos = calculos + 1;
end
if ( refBlkVerPointC < 1 || refBlkVerPointC+TamMb-1 > fila ...
        || refBlkHorPointC < 1 || refBlkHorPointC+TamMb-1 >
            columna)
    % no se hace nada %
else
    costos(3) = FuncionCostosMAD(imgP(i:i+TamMb-1,j:j+TamMb-1),
        ...
            imgI(refBlkVerPointC:refBlkVerPointC+TamMb-1, ...
                refBlkHorPointC:refBlkHorPointC+TamMb-1), TamMb
                    );
    calculos = calculos + 1;
end

if (costos(1) >= costos(2) && costos(1) >= costos(3))
    quadrant = 4;
```

```
elseif (costos(1) >= costos(2) && costos(1) < costos(3))
    quadrant = 1;
elseif (costos(1) < costos(2) && costos(1) < costos(3))
    quadrant = 2;
elseif (costos(1) < costos(2) && costos(1) >= costos(3))
    quadrant = 3;
end
switch quadrant
    case 1
        refBlkVerPointD = y - TamPaso;
        refBlkHorPointD = x;
        refBlkVerPointE = y - TamPaso;
        refBlkHorPointE = x + TamPaso;
        if ( refBlkVerPointD < 1 || refBlkVerPointD+TamMb-1 >
            fila ...
                || refBlkHorPointD < 1 || refBlkHorPointD+TamMb
                    -1 > columna)
            %no se hace nada %
        else
            costos(4) = FuncionCostosMAD(imgP(i:i+TamMb-1,j:j+
                TamMb-1), ...
                        imgI(refBlkVerPointD:refBlkVerPointD+
                            TamMb-1, ...
                             refBlkHorPointD:refBlkHorPointD+
                                TamMb-1), TamMb);
            calculos = calculos + 1;
        end
        if ( refBlkVerPointE < 1 || refBlkVerPointE+TamMb-1 >
            fila ...
                || refBlkHorPointE < 1 || refBlkHorPointE+TamMb
                    -1 > columna)
            %no se hace nada %
        else
            costos(5) = FuncionCostosMAD(imgP(i:i+TamMb-1,j:j+
                TamMb-1), ...
                        imgI(refBlkVerPointD:refBlkVerPointD+
                            TamMb-1, ...
                             refBlkHorPointD:refBlkHorPointD+
                                TamMb-1), TamMb);
```

```
            calculos = calculos + 1;
        end

    case 2
        refBlkVerPointD = y - TamPaso;
        refBlkHorPointD = x;
        refBlkVerPointE = y - TamPaso;
        refBlkHorPointE = x - TamPaso;
        refBlkVerPointF = y;
        refBlkHorPointF = x - TamPaso;

        if ( refBlkVerPointD < 1 || refBlkVerPointD+TamMb-1 >
            fila ...
                || refBlkHorPointD < 1 || refBlkHorPointD+TamMb
                    -1 > columna)
            %no se hace nada %
        else
            costos(4) = FuncionCostosMAD(imgP(i:i+TamMb-1,j:j+
                TamMb-1), ...
                        imgI(refBlkVerPointD:refBlkVerPointD+
                            TamMb-1, ...
                             refBlkHorPointD:refBlkHorPointD+
                                TamMb-1), TamMb);
            calculos = calculos + 1;
        end
        if ( refBlkVerPointE < 1 || refBlkVerPointE+TamMb-1 >
            fila ...
                || refBlkHorPointE < 1 || refBlkHorPointE+TamMb
                    -1 > columna)
            %no se hace nada %
        else
            costos(5) = FuncionCostosMAD(imgP(i:i+TamMb-1,j:j+
                TamMb-1), ...
                        imgI(refBlkVerPointE:refBlkVerPointE+
                            TamMb-1, ...
                             refBlkHorPointE:refBlkHorPointE+
                                TamMb-1), TamMb);
            calculos = calculos + 1;
        end
```

```
        if ( refBlkVerPointF < 1 || refBlkVerPointF+TamMb-1 >
            fila ...
                || refBlkHorPointF < 1 || refBlkHorPointF+TamMb
                    -1 > columna)
            %no se hace nada %
        else
            costos(6) = FuncionCostosMAD(imgP(i:i+TamMb-1,j:j+
                TamMb-1), ...
                        imgI(refBlkVerPointF:refBlkVerPointF+
                            TamMb-1, ...
                             refBlkHorPointF:refBlkHorPointF+
                                TamMb-1), TamMb);
            calculos = calculos + 1;
        end
    case 3
        refBlkVerPointD = y;
        refBlkHorPointD = x - TamPaso;
        refBlkVerPointE = y - TamPaso;
        refBlkHorPointE = x - TamPaso;
        if ( refBlkVerPointD < 1 || refBlkVerPointD+TamMb-1 >
            fila ...
                || refBlkHorPointD < 1 || refBlkHorPointD+TamMb
                    -1 > columna)
            %no se hace nada %
        else
            costos(4) = FuncionCostosMAD(imgP(i:i+TamMb-1,j:j+
                TamMb-1), ...
                        imgI(refBlkVerPointD:refBlkVerPointD+
                            TamMb-1, ...
                             refBlkHorPointD:refBlkHorPointD+
                                TamMb-1), TamMb);
            calculos = calculos + 1;
        end
        if ( refBlkVerPointE < 1 || refBlkVerPointE+TamMb-1 >
            fila ...
                || refBlkHorPointE < 1 || refBlkHorPointE+TamMb
                    -1 > columna)
            %no se hace nada %
        else
```

```
                costos(5) = FuncionCostosMAD(imgP(i:i+TamMb-1,j:j+
                    TamMb-1), ...
                                imgI(refBlkVerPointD:refBlkVerPointD+
                                    TamMb-1, ...
                                     refBlkHorPointD:refBlkHorPointD+
                                         TamMb-1), TamMb);
                calculos = calculos + 1;
            end
        case 4
            refBlkVerPointD = y + TamPaso;
            refBlkHorPointD = x + TamPaso;
            if ( refBlkVerPointD < 1 || refBlkVerPointD+TamMb-1 >
                fila ...
                     || refBlkHorPointD < 1 || refBlkHorPointD+TamMb
                         -1 > columna)
                %no se hace nada %
            else
                costos(4) = FuncionCostosMAD(imgP(i:i+TamMb-1,j:j+
                    TamMb-1), ...
                                imgI(refBlkVerPointD:refBlkVerPointD+
                                    TamMb-1, ...
                                     refBlkHorPointD:refBlkHorPointD+
                                         TamMb-1), TamMb);
                calculos = calculos + 1;
            end
        otherwise
end
[cost, dxy] = minimo(costos);
switch dxy
    case 1
        x = x; y = y;
    case 2
        x = refBlkHorPointB;
        y = refBlkVerPointB;
    case 3
        x = refBlkHorPointC;
        y = refBlkVerPointC;
    case 4
        x = refBlkHorPointD;
```

```
                    y = refBlkVerPointD;
                case 5
                    x = refBlkHorPointE;
                    y = refBlkVerPointE;
                case 6
                    x = refBlkHorPointF;
                    y = refBlkVerPointF;
            end
            costos = unos(1,6) * 65537  ;
            TamPaso = TamPaso / 2;
        end
        vectores(1,contadorMb) = y - i;
        vectores(2,contadorMb) = x - j;
        contadorMb = contadorMb + 1;
    end
end
VectorMovimiento = vectores;
MHTSS = calculos/(contadorMb - 1);
```