UCLV
Universidad Central
"Marta Abreu" de Las Villas

FIE
Facultad de
Ingeniería Eléctrica

Departamento de Telecomunicaciones y Electrónica

TRABAJO DE DIPLOMA

Título: "Obtención de patrones binarios locales en imágenes a color empleando álgebra de cuaterniones"

Autor: Raidel Labañino Medina

Tutor: Dr. C. Juan Valentín Lorenzo Ginori

Santa Clara, junio 2018

PENSAMIENTO

"Creo que, si haces algo y resulta bastante bueno, después deberías ir a hacer algo maravilloso y no pensar mucho en ello. Simplemente averigua qué sigue."

Steve Jobs

DEDICATORIA

A Tito, por parecer siempre tan cerca desde la distancia.

A mi mamá, la pieza irremplazable de mi rompecabezas.

AGRADECIMIENTOS

A mi mamá, por la proeza de ser súper mujer cuando faltaba un hombre. Porque su fuerza interior me vuelve invulnerable.

A Tata, mi hermana por convicción, porque los kilómetros no han mellado este amor profeso.

A Kiko, por ser incondicional para mí, porque afrontaremos juntos lo que nos depare la vida.

A Lau, mi álter ego, por llegar sin avisar y quedarse sin pedir permiso, por estar cuando más la necesito, y cuando no también.

A los amigos que compartieron conmigo cuarto, comida, fiesta, limpiezas y aprietos. Por ser los más democráticos en la convivencia. Que esta amistad sea eterna.

A Juan Lorenzo, mi tutor y guía, por la dedicación, el profesionalismo y la comprensión en todo momento. ¡Qué mejor paradigma!

A cada una de las personas que he conocido durante estos cinco años, que hicieron de mi universidad, la casa de la que nunca me iré del todo.

TAREA TÉCNICA

- Realizar una revisión bibliográfica sobre los métodos de representación y clasificación de imágenes a color propuestos hasta la fecha.
- Caracterizar la representación de imágenes a color en el dominio de los números hipercomplejos a partir de elementos determinados de su fundamentación algebraica.
- Resumir elementos generales sobre el contraste de Michelson y los patrones binarios locales, así como acerca del empleo de estos últimos en procesos de análisis y clasificación de texturas.
- Programar en Matlab un algoritmo de clasificación de imágenes basado en patrones binarios locales cuyos fundamentos sean el contraste de Michelson y el uso de cuaterniones para su representación.
- Evaluar experimentalmente la efectividad del algoritmo implementado mediante pruebas con imágenes a color teniendo en cuenta diferentes métodos.

RESUMEN

La clasificación de imágenes atendiendo a criterios de diversa índole ha sido un tema dentro del procesamiento digital de estas que ha cobrado especial relevancia en disímiles aplicaciones modernas. Para tal fin se han propuesto varios descriptores de imágenes desde distintas perspectivas, entre los cuales destaca sobre el que se basa este trabajo por su simplicidad y efectividad. Esencialmente aquí se implementa un descriptor local basado en patrones binarios locales cuya orientación va hacia el análisis de textura y que, a diferencia de otros descriptores, utiliza una combinación de elementos que no había sido antes empleada: el contraste de Michelson como medida estable del contenido de la imagen en regiones de interés y la representación de los píxeles de color mediante cuaterniones. Basado en ello, el algoritmo presenta dos métodos que utilizan en mayor o menor medida la relación entre los píxeles de color representados, pero que son igualmente efectivos y útiles. Los resultados obtenidos de experimentos con bases de datos de imágenes a color fundamentan la utilidad del algoritmo en cuestión para posteriores análisis y clasificaciones de texturas.

TABLA DE CONTENIDOS

PENSAMIENTO i

DEDICATORIA ii

AGRADECIMIENTOS iii

TAREA TÉCNICA iv

RESUMEN v

INTRODUCCIÓN 1

Objetivo general 2

Objetivos específicos 2

Organización del informe 4

CAPÍTULO 1. Álgebra de cuaterniones extendida al procesamiento digital de imágenes a color. Fundamentación teórica. 5

1.1 Introducción a los números cuaterniones. 5

1.1.1 Definición general. 6

1.1.2 Representación matemática. 6

1.1.3 Álgebra de cuaterniones. 7

1.1.3.1 Elementos básicos. 8

1.1.3.2 Multiplicación y suma de cuaterniones. Propiedades. 8

1.1.3.3 Rotaciones mediante cuaterniones y Traslación de Clifford. 9

1.2 Los cuaterniones en el procesamiento digital de imágenes. 11

1.2.1 Representación de imágenes en el dominio de los cuaterniones. 11

1.3 Clasificación de imágenes a color. Descriptores locales. 12

1.4 Conceptualización del contraste de Michelson. 14

1.4.1 Definición del contraste de Michelson para imágenes a color en el dominio de los cuaterniones. 14

1.5 Patrones Binarios Locales (LBPs). 15

1.5.1 Análisis de textura utilizando LBPs. 16

1.5.2 Derivación del operador LBP. 17

1.5.3 Invariancia a la rotación. 19

CAPÍTULO 2. Descriptor local *Quaternion-Michelson*. Materiales y métodos 22

2.1 Descriptor *Quaternion-Michelson* para la clasificación de imágenes RGB. 22

2.2 Procedimiento general para la implementación del algoritmo. 22

2.2.1 Propiedades de las imágenes a codificar. 24

2.3 Representación cuaterniónica de los planos de color de los píxeles. 25

2.4 Extracción del contraste de Michelson. 27

2.4.1 Ordenamiento de los cuaterniones atendiendo a su fase. 28

2.4.1.1 Cálculo de la Traslación de Clifford. 29

2.4.1.2 Obtención de la fase de los cuaterniones. 29

2.4.2 Métodos para la obtención del contraste. 30

2.4.2.1 Extracción del contraste de Michelson basado en cuaterniones. 30

2.4.2.2 Extracción del contraste de Michelson basado en la fase de cuaterniones. 31

2.5 Codificación de patrones. 32

2.5.1 Construcción de LBPs de imágenes a color empleando el contraste de Michelson calculado con empleo de cuaterniones. 33

2.6 Obtención del histograma normalizado para el resultado de la codificación. 34

2.7 Funciones de Matlab empleadas. 34

CAPÍTULO 3. Análisis y discusión de los resultados obtenidos 36

3.1 Resultados de la representación cuaterniónica de los píxeles. 36

3.1.1 Resultados de la traslación de Clifford aplicada a la matriz cuaternión. 38

3.1.2 Resultados de la obtención y ordenamiento de la fase de los cuaterniones. 39

3.2 Resultados de la extracción del contraste de Michelson. 40

3.3 Resultados de la codificación de patrones binarios locales. 45

3.4 Validación de los resultados obtenidos. 48

3.5 Tiempos de ejecución del algoritmo. 51

CONCLUSIONES Y RECOMENDACIONES 53

CONCLUSIONES 53

RECOMENDACIONES 54

REFERENCIAS BIBLIOGRÁFICAS 55

ANEXOS 59

Anexo I Código fuente de los programas de computación implementados. 59

Script QMD que contiene el algoritmo estructurado por pasos. 59

Código de la función *michelson* empleada para la extracción del contraste. 62

Código de la función *clifford* de uso en la ejecución de la traslación de Clifford (subfunción de *michelson*). 63

Código de la función *lbpgray_all* que calcula los patrones binarios locales a partir del mapa de Michelson obtenido. 64

Código de la función *nlqfilter* encargada del barrido de la imagen. 68

Código de la función *mkconstarray* (subfunción de *nlqfilter*). 70

Anexo II Mapas de Michelson de distintas imágenes utilizando el método basado en cuaterniones con vector unitario *i* 71

Anexo III Mapas de Michelson de distintas imágenes utilizando el método basado en la fase de cuaterniones con vector unitario *i* 72

INTRODUCCIÓN

En la rama de la clasificación de imágenes, objeto actual de investigación en los campos del procesamiento computacional y reconocimiento de patrones por sus múltiples adaptaciones, cobran especial relevancia los descriptores de imágenes como herramientas que procesan digitalmente las mismas en función de obtener resultados para una u otra aplicación específica. Estos se agrupan en dos modelos diferentes, globales y locales, teniendo en cuenta la porción de la imagen que analizan y, aunque ambos poseen ventajas y desventajas individuales, la presente investigación abordará básicamente sobre los descriptores locales a propósito del algoritmo que se pretende implementar.

Un gran número de descriptores de este tipo han sido propuestos desde diferentes perspectivas, entre ellos los Patrones Binarios Locales (LBP) [1] que, aunque han sido empleados considerablemente en imágenes en escala de grises, su implementación en imágenes a color es un tema de análisis en la actualidad. En contraposición a los descriptores globales, estos trabajan con características micro-estructurales de la imagen, por lo cual poseen un amplio espectro de utilidad en el reconocimiento de rostros, clasificación de texturas y recuperación de imágenes.

Como elemento crucial de esta rama se encuentra la representación de imágenes cuya mayor dificultad radica en que la captura de imágenes en ambientes realmente complejos generalmente sufre variaciones indeseadas como oclusiones y cambios de iluminación, intensidad y posicionamiento, lo cual se traduce en imposibilidad para el ser humano de diferenciar dichas características en clases iguales. Por ende, el objetivo principal de la representación de imágenes es encontrar la forma apropiada de describir las relaciones entre píxeles en una imagen o región local de la misma.

A partir de lo anteriormente relacionado, investigaciones novedosas han propuesto la representación de las imágenes a color mediante cuaterniones, logrando una concatenación de la información de color inmersa en los planos RGB de cada píxel; para lo cual se utilizan cuaterniones puros, donde las tres partes imaginarias del cuaternión representan cada canal, y la parte escalar del mismo se anula. Es importante mencionar que los estudios realizados utilizan esencialmente las características visuales de la imagen para representarla y manipularla.

Dichas características pueden clasificarse de acuerdo a su ámbito como locales o globales, y a su nivel de abstracción como lógicas y físicas [2]; estas últimas son aquellas que pueden expresarse cuantitativamente, y se extraen mediante la aplicación de técnicas de tratamiento digital de imágenes, también son llamadas características de bajo nivel [3]. Entre ellas destaca la textura, que se encuentra además en la categoría de característica local y puede definirse como una propiedad de homogeneidad en las regiones de la imagen [4]; y el color, utilizado para describir las distribuciones cromáticas de la imagen; que para el caso del espacio RGB, contiene la codificación de los canales rojo, verde y azul de acuerdo a su intensidad en tres componentes, canales o planos. El procesamiento en el dominio de las características permite eliminar información redundante y reducir la dimensionalidad de trabajo. Si este módulo funciona bien se producen dos cosas: una óptima clasificación y la reducción del tiempo de cómputo.

Por consiguiente, el presente trabajo prevé el desarrollo e implementación de un algoritmo de clasificación basado en el descriptor local *Quaternion-Michelson Descriptor (QMD)* [5], que muestra un enfoque diferente en torno a la evaluación de la textura en imágenes, básicamente por la representación de los píxeles como números hipercomplejos de cuatro dimensiones y la utilización del contraste de Michelson para la obtención de LBP extendidos a este dominio. A partir de ello, se definió como problema de investigación: ¿Cómo obtener patrones binarios locales de imágenes a color teniendo en cuenta la representación de las mismas a través de cuaterniones y la obtención de rasgos locales en regiones de interés?

Para darle cumplimiento a lo antes expuesto se propusieron los siguientes objetivos:

Objetivo general

- Implementar un algoritmo computacional que, a partir de la representación de imágenes a color mediante cuaterniones, obtenga el contraste de Michelson y los patrones binarios locales asociados a dichas imágenes, con orientación hacia el análisis de textura.

Objetivos específicos

1. Describir los principios básicos del álgebra de cuaterniones y el contraste de Michelson para la representación de imágenes a color.

2. Sintetizar elementos teóricos sobre la clasificación de imágenes a color a partir del uso de descriptores locales.
3. Resumir conceptos relativos a los patrones binarios locales y su aplicación en el análisis de textura.
4. Programar funciones en Matlab con empleo de *Matlab Image Processing Toolbox* relacionadas con el tratamiento de imágenes a color a nivel físico para el cálculo del contraste de Michelson mediante cuaterniones y de patrones binarios locales basados en dicho contraste.
5. Comparar los resultados de la extracción de características locales en imágenes a partir del algoritmo implementado.

De lo anteriormente relacionado se derivan las siguientes interrogantes científicas:

1. ¿Qué métodos han sido propuestos para la representación de imágenes a color y la obtención de rasgos locales orientados hacia el análisis de textura?
2. ¿Cuáles aspectos son imprescindibles para representar imágenes a color a partir de números hipercomplejos o cuaterniones de forma tal que se optimice el procesamiento digital de estas?
3. ¿Cuál es el procedimiento para implementar un algoritmo que obtenga patrones binarios locales asociados a imágenes a color teniendo en cuenta la representación de sus píxeles mediante cuaterniones y el contraste de Michelson?
4. ¿Qué ventajas posee un algoritmo de representación de imágenes basado en cuaterniones que reduce la complejidad computacional respecto a otros métodos?

Consecuentemente, la implementación del descriptor *Quaternion-Michelson* se espera reduzca la complejidad computacional de los actuales sistemas de procesamiento de imágenes en torno a la clasificación de las mismas, logrando una elegante y a la vez factible representación de ellas a partir del álgebra de los cuaterniones. Se pretende contribuir a la evolución del análisis de imágenes en nuestra universidad teniendo en cuenta rasgos físico-locales, como la textura, y que a su vez la investigación tribute al desarrollo, perfeccionamiento e

implementación de novedosos métodos de clasificación. Además, se estima que permita reducir el tiempo de procesado de imágenes a color y provea a especialistas e investigadores elementos para estudios ulteriores y análisis comparativos respecto a propuestas precedentes.

Con la ejecución del proyecto se da solución a la necesidad de poseer una herramienta que sirva de base para la clasificación de imágenes a color a través de software de difícil acceso para nuestro país e institución, que sean capaces de procesar las mismas en función de obtener resultados para disímiles aplicaciones prácticas modernas, entre las cuales cabe destacar, por ser tema de investigación abierto actual y al cual tributa el presente trabajo, el diagnóstico de imágenes microscópicas para la identificación y control de células infectadas con parásitos del género *Plasmodium*, responsables de enfermedades como la malaria.

Organización del informe

El informe de la investigación se estructura en introducción, capitulario, conclusiones, referencias bibliográficas y anexos. A continuación, se hace breve referencia a los temas y contenidos que se abordarán en cada capítulo.

Capítulo 1: En este capítulo se hace alusión a la fundamentación teórica de los temas que sirven de base a la investigación, tales como elementos generales sobre los cuaterniones y su álgebra, conceptos relativos a la clasificación y representación de imágenes, el análisis de textura y los diferentes tipos de descriptores de imagen, con énfasis en los Patrones Binarios Locales.

Capítulo 2: En esta sección se detallan las características de las imágenes utilizadas en el estudio y, sobre todo, se describe el procedimiento ejecutado por pasos para la implementación del algoritmo, indicando simultáneamente las funciones de Matlab que se emplean para tal finalidad.

Capítulo 3: En este apartado se recogen y analizan los resultados de una manera cualitativa y cuantitativa, ilustrados mediante tablas y figuras, se muestran pruebas estadísticas relacionadas y más, de todo lo cual se derivan importantes conclusiones.

CAPÍTULO 1. Álgebra de cuaterniones extendida al procesamiento digital de imágenes a color. Fundamentación teórica.

En este capítulo se exponen los fundamentos teóricos que sirven de base a la presente investigación, incluyendo temas del procesamiento digital de imágenes relacionados con ellos. De esta manera, se abordan cuestiones vinculadas a los números hipercomplejos, básicamente los cuaterniones. Se resumen las principales características de los mismos, así como elementos relevantes de su álgebra que tienen aplicación actual en la representación de imágenes a color. Además, se relata acerca de los patrones binarios locales, se sintetizan conceptos fundamentales relacionados con su aplicabilidad en el análisis de textura y se describe la obtención de los mismos a través del contraste de Michelson en regiones de interés dentro de una imagen de referencia.

1.1 Introducción a los números cuaterniones.

Los cuaterniones, o cuaternios, como indistintamente los cita la bibliografía referente al tema, son un conjunto numérico introducido por Sir William Rowan Hamilton[1] a mediados del siglo XIX, en 1843. Lo que Hamilton buscaba en la época era extender el concepto de número complejo a más de dos dimensiones, lo cual logró sacrificando la conmutatividad de la multiplicación.

Resulta que el conjunto de los cuaterniones, con las operaciones de suma y multiplicación definidas entre ellos, forma un sistema matemático conocido como anillo de división no conmutativo. Este título enfatiza el hecho de que el producto de cuaterniones, en general, no es conmutativo y también que para cada elemento diferente de cero existe un inverso multiplicativo [6].

[1] Físico-matemático irlandés (1805-1865) que hizo grandes aportaciones a los campos de la mecánica, óptica y astronomía. Entre sus trabajos más relevantes destacan la inserción de la mecánica hamiltoniana, la ecuación de Hamilton-Jacobi, el teorema de Cayley-Hamilton y el descubrimiento de los cuaterniones.

La no conmutatividad de la multiplicación entre cuaterniones imposibilitó el desarrollo de una teoría de funciones de variable cuaternia, hecho que llevó a que la aplicación de los cuaterniones en la matemática resultara poco significativa.

Los cuaterniones, sin embargo, probaron ser especialmente útiles para la representación de rotaciones en espacios tridimensionales, tarea en la que resultan más eficientes que las matrices de rotación habituales. Por esta razón han sido utilizados en aplicaciones relacionadas con el control de orientación de vehículos tales como aeronaves, submarinos, naves espaciales y similares, en animación tridimensional, realidad virtual y, como explica la presente investigación, en la representación de imágenes a color.

1.1.1 Definición general.

De manera resumida, los cuaterniones, referidos como números hipercomplejos de cuatro dimensiones, son una extensión de los números reales con similitud a los números complejos. Mientras que estos últimos son una extensión de los números reales por la adición de la unidad imaginaria i, tal que $i^2 = -1$, los cuaterniones son una extensión algo más generalizada.

Teniendo en cuenta que en 4 dimensiones hay 3 soluciones diferentes a la raíz de (-1), cada una de las dimensiones imaginarias tiene un valor múltiplo de dicha raíz, conocidos como i, j y k, elementos de la base canónica de $\mathbb{R}^3$ que cumplen las siguientes relaciones: [7]

$$i^2 = j^2 = k^2 = ijk = -1 \tag{1.1}$$

$$ij = -ji = k;\ jk = -kj = i; ki = -ik = j \tag{1.2}$$

Matemáticamente, se definen los cuaterniones como elementos de la forma:

$$\boldsymbol{q = a + bi + cj + dk} \tag{1.3}$$

tal que $a, b, c, d \in \mathbb{R}$, donde:

- a se denomina parte real o escalar.
- $bi + cj + dk$ se denomina parte imaginaria o vectorial.

1.1.2 Representación matemática.

El conjunto de cuaterniones se representa con la letra $\boldsymbol{H}$ o $\mathbb{H}$, en honor a su creador Hamilton; por consiguiente, su notación de conjunto queda formulada como:

$$\boldsymbol{H} = \mathbb{H} = \{q = a + bi + cj + dk : a, b, c, d \in \mathbb{R}\} \subset \mathbb{R}^4 \tag{1.4}$$

Aunque también se pueden representar como ampliación de los números complejos de la siguiente manera:

$$\boldsymbol{H} = \mathbb{H} = \{(a + bi) + (c + di)j : a + bi, c + di \in \mathbb{C}\} \subset \mathbb{C}^2 \tag{1.5}$$

La forma polar quedaría definida como: $q = |q|e^{z\theta} = |q|(cos\theta + zsin\theta)$ (1.6)

Donde $z = \frac{ib+jc+kd}{\sqrt{b^2+c^2+d^2}}$ (1.7) y $\theta = arctan\frac{\sqrt{b^2+c^2+d^2}}{a}$ (1.8)

Por otra parte, la **representación vectorial** de cualquier cuaternión se puede hacer considerando a $\{1, i, j, k\}$ como "base" y definiendo cualquier cuaternión como el producto interno de un vector $\vec{x} = (a, b, c, d)$ por el vector de las bases. A veces se puede apartar el elemento a, considerando entonces este otro vector: $\vec{x} = (a, \vec{a}) = (a, b, c, d)$.

La **representación matricial** puede hacerse, al menos, de dos formas distintas. La primera es usando matrices complejas de $2x2$.

$$q = \begin{pmatrix} a - di & -b + ci \\ b + ci & a + di \end{pmatrix} \tag{1.9}$$

La otra forma es utilizando matrices reales de $4x4$ como se muestra a continuación:

$$q = \begin{pmatrix} a & -b & d & -c \\ b & a & -c & -d \\ -d & c & a & -b \\ c & d & b & a \end{pmatrix} \tag{1.10}$$

Es válido destacar que, tanto en ambos casos como para los cuaterniones definidos como en (1.3), es posible verificar la identidad:

$$|q|^2 = a^2 + b^2 + c^2 + d^2 \tag{1.11}$$

1.1.3 Álgebra de cuaterniones.

El presente apartado está dedicado al estudio del álgebra de los cuaterniones partiendo de la definición antes expuesta, explicando a partir de ella algunas propiedades y operaciones algebraicas sencillas que sirven de base a operaciones más complejas de utilidad en el tratamiento de imágenes a color.

1.1.3.1 Elementos básicos.

El concepto de cuaternión **conjugado** en números hipercomplejos, al igual que en los números complejos, se obtiene cambiando el signo de su componente imaginaria. Así, el conjugado de un cuaternión se define como:

$$\bar{q} = q^* = a - bi - cj - dk. \tag{1.12}$$

Puede apreciarse que la operación de conjugación hipercompleja es el único automorfismo que deja invariante el subconjunto de los números reales diferente de la identidad.

Por otra parte, el cuaternión nulo es aquel en que $a = b = c = d = 0$, mas si solo la parte real es cero, será un **cuaternión puro**, condición que es aprovechada en aplicaciones como la de esta investigación y que será pertinentemente explicada en epígrafes posteriores.

Mientras, la **norma** o valor absoluto se puede expresar como:

$$\|q\| = \sqrt{qq^*} = \sqrt{a^2 + b^2 + c^2 + d^2} \tag{1.13}$$

de manera tal que cuando su valor es uno, el cuaternión será de carácter unitario y, dado un cuaternión cuya norma no sea igual a uno se puede normalizar definiendo un nuevo cuaternión asociado al primero y determinado a partir de la siguiente operación:

$$q_1 = \frac{q}{|q|} \tag{1.14}$$

El cuaternión opuesto es: $$-q = -a - bi - cj - dk \tag{1.15}$$

y el inverso se define como: $$q^{-1} = \frac{q^*}{|q|^2} \tag{1.16}$$

1.1.3.2 Multiplicación y suma de cuaterniones. Propiedades.

La **multiplicación** de cuaterniones es una de las operaciones más importantes de la aritmética hipercompleja que sirve de base a numerosas aplicaciones y otras operaciones de mayor complejidad matemática. Se obtiene multiplicando componente a componente de forma tal que, si se tienen dos cuaterniones $q_1, q_2 \in \mathbb{H}$, $q_1 = a_1 + b_1 i + c_1 j + d_1 k$, $q_2 = a_2 + b_2 i + c_2 j + d_2 k$, el resultado del producto es:

$$q_i \cdot q_2 = (a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2) + (a_1 b_2 + b_1 a_2 + c_1 d_2 - d_1 c_2)\boldsymbol{i} + (a_1 c_2 - b_1 d_2 + c_1 a_2 + d_1 b_2)\boldsymbol{j} + (a_1 d_2 + b_1 c_2 - c_1 b_2 + d_1 a_2)\boldsymbol{k} \tag{1.17}$$

Sus propiedades más significativas se relacionan a continuación:

Propiedades del producto

• Ley interna: $\forall\, q_1, q_2 \in \mathbb{H} : (q_1 \cdot q_2) \in \mathbb{H}$

• Asociativa: $q_1 \cdot (q_2 \cdot q_3) = (q_1 \cdot q_2) \cdot q_3$

• Elemento neutro: $q_1 \cdot 1 = 1 \cdot q_1 = q_1$

• Elemento inverso: $\forall\, q_1 \neq 0 :\ q_1 \cdot {q_1}^{-1} = {q_1}^{-1} \cdot q_1 = 1$

• Distributiva del producto respecto a la suma: $q_1 \cdot (q_2 + q_3) = q_1 \cdot q_2 + q_1 \cdot q_3$

• No cumple la propiedad conmutativa, por lo cual se dice que $(\mathbb{H},\ \cdot)$ es un grupo no conmutativo (o no abeliano).

La **adición** de cuaterniones, por su parte, es igualmente análoga a la de los complejos, de manera tal que si existen los números q_1, q_2 (definidos anteriormente), la suma de ellos quedará de la siguiente forma:

$$q_1 + q_2 = (a_1 + a_2) + (b_1 + b_2)\boldsymbol{i} + (c_1 + c_2)\boldsymbol{j} + (d_1 + d_2)\boldsymbol{k}. \tag{1.18}$$

Destacando como propiedades:

Propiedades de la suma

• Ley interna: $\forall\, q_1, q_2 \in \mathbb{H} : (q_1 + q_2) \in \mathbb{H}$

• Conmutativa: $q_1 + q_2 = q_2 + q_1$

• Asociativa: $q_1 + (q_2 + q_3) = (q_1 + q_2) + q_3$

• Elemento neutro: $q_1 + 0 = 0 + q_1 = q_1$

Por consiguiente, es posible añadir que $(\mathbb{H}, +)$ es un grupo conmutativo (o abeliano).

1.1.3.3 Rotaciones mediante cuaterniones y Traslación de Clifford.

La importancia de los cuaterniones unitarios anteriormente citados reside en que a través de ellos se pueden representar rotaciones en tres dimensiones de manera muy sencilla. Si q es un cuaternión unitario, este puede pensarse como una esfera de radio 1 en el espacio 4D, y es posible representar una rotación en este espacio suponiendo que $\{b, c, d\}$ son las componentes de cualquier eje arbitrario y a el ángulo de rotación [8].

El cuaternión i representa una rotación de 180º alrededor del eje 'x', el j sobre el eje 'y', y el k sobre el 'z'. De esta manera, $i^2 = -1$, representa una rotación de 360º alrededor del eje 'x', igual sucede con el resto, de forma que tanto '+1', como '-1', representarán el mismo giro, aunque por caminos diferentes. Es decir, un giro de un objeto tridimensional respecto a un eje dos veces un valor de π que equivaldría, recordando el plano complejo, a multiplicar ese objeto por i^2 manteniendo el resto de dimensiones inalteradas, devuelve el opuesto del objeto. Esta curiosidad es lo que se denomina un espinor[2] [7].

Los cuaterniones pueden representar rotaciones 3D, escalados y reflexiones, sin embargo, no pueden representar traslaciones. Para representar una rotación mediante cuaterniones se ha de tener el punto a rotar en forma de cuaternión, lo que se consigue generando un cuaternión con la parte imaginaria a cero; así por ejemplo, el vector $v = \{1,2,3\}$ quedaría como el cuaternión $q = 0 + i + 2j + 3k$.

En este sentido, una importante propiedad incluida dentro de las llamadas álgebras de Clifford[3] es la Traslación Cuaterniónica de Clifford (CTQ) [9], la cual brinda soluciones diversas en aplicaciones con cuaterniones puros. Aquí, teniendo un vector unitario p, la traslación derecha se obtiene multiplicando el cuaternión en cuestión por dicho vector unitario (en ese orden), la izquierda sería al revés consecuentemente y, aunque el resultado de ambas es diferente, la fase de los cuaterniones resultantes es idéntica, propiedad que es aprovechada en el algoritmo sobre el cual versa la presente investigación. A continuación, se muestran las ecuaciones que definen tales operaciones:

$$CTQ_r(q,p) = qp \tag{1.19}$$

$$CTQ_l(q,p) = pq \tag{1.20}$$

Donde $CTQ_r(q,p) \neq CTQ_l(q,p)$ y $\theta_{CTQ_r} = \theta_{CTQ_r}$.

[2] Un espinor u objeto espinorial se define como un objeto que se transforma en su negativo u opuesto cuando sufre una rotación completa de 2π.

[3] Son álgebras asociativas de importancia en matemáticas, en particular en teoría de la forma cuadrática y del grupo ortogonal y en la física, que generalizan el cuerpo de los números complejos y los cuaterniones de Hamilton. Se nombran así por el matemático William Kingdon Clifford (1845-1879).

1.2 Los cuaterniones en el procesamiento digital de imágenes.

Expresar el contenido de una imagen de una forma en la cual los sistemas computacionales puedan procesarla y entenderla de la misma manera que lo hacen los seres humanos, es todavía un problema de investigación abierto. Sería necesario algo equivalente a un sistema de inteligencia artificial que trabajara como la mente humana con la habilidad de manipular ideas abstractas automáticamente para procesarlas. Esto, por supuesto, no está todavía al alcance de las capacidades de los sistemas actuales [10].

Como se corroboró anteriormente, los cuaterniones pueden interpretarse como vectores de cuatro componentes y, por tanto, como elementos que pertenecen a un espacio de cuatro dimensiones, por lo que en la actualidad se han insertado paulatinamente en el procesamiento y análisis de imágenes a color. Teniendo en cuenta estas características y dadas las ventajas de su uso, numerosos descriptores cuaterniónicos han sido lanzados, entre ellos: el descriptor cuaterniónico de momento Zernike [11], [12], el cuaterniónico de momento Fourier-Mellin [13] y su versión ortogonal [14], el descriptor de Fourier geométrico [15], el descriptor polar hipercomplejo de Fourier [16], el de momento cuaterniónico [17] y el descriptor de momento pseudo-Zernike cuaterniónico [14], que son de tipo global, sin embargo, descriptores locales como el cuaterniónico de Michelson, basamento de la presente investigación, usan dicha representación.

1.2.1 Representación de imágenes en el dominio de los cuaterniones.

Para manipular una imagen en el dominio de los cuaterniones, el primer paso es representar la misma a partir de ellos, pero resulta que la imagen de color está usualmente descrita en un espacio de color RGB que es un espacio 3D, para los planos rojo, verde y azul respectivamente, mientras el cuaternión es un sistema numérico de 4D. Para ocuparse de esta incompatibilidad entre el espacio de color y el dominio del cuaternión, las componentes imaginarias de este se usan para representar los píxeles de color, como se muestra a continuación [5]:

$$\dot{F}(x,y) = R(x,y)\boldsymbol{i} + G(x,y)\boldsymbol{j} + B(x,y)\boldsymbol{k} \tag{1.21}$$

Donde $R(x,y), G(x,y)\ y\ B(x,y)$ son dichas componentes. Por lo tanto, al utilizar esta herramienta se puede trabajar la imagen como una sola entidad, convirtiéndola en un cuaternión puro, y no con sus componentes separadas o en escala de grises.

A diferencia de los métodos tradicionales de representación de imágenes a color, la representación cuaterniónica (QR) codifica todos los canales de color de la imagen usando un cuaternión. Según la descripción previa, $\dot{F}(x, y)$ posibilita un mapeo individual entre el dominio del cuaternión y el espacio de color RGB, por consiguiente, las propiedades y operaciones antes expuestas de los cuaterniones, dígase módulo, fase, rotación, CTQ, entre otras, pueden ser utilizadas directamente para procesar imágenes de color en dependencia de la aplicación. Cualquier operación aplicada a $\dot{F}(x, y)$ tendrá efectos en todos los canales de color simultáneamente. Estas operaciones, por ende, traen perspectivas nuevas para la descripción de la imagen a color.

1.3 Clasificación de imágenes a color. Descriptores locales.

Dentro del procesamiento digital de imágenes los rasgos o valores de medición pueden describir propiedades globales o locales de las mismas y revelar información estadística, algebraica, geométrica, espacial, diferencial y espectral de una imagen o subimagen. Los rasgos son de gran importancia en el aislamiento de las regiones con propiedades comunes dentro de una imagen, la segmentación, y la subsecuente identificación o etiquetado de tales regiones, o sea, la clasificación de imágenes de acuerdo a su contenido [18], [19].

En el proceso de reconocimiento de patrones se utiliza un gran número de técnicas de clasificación. Algunas de estas se conocen como técnicas de decisión teórica, en las que la clasificación de un patrón desconocido se decide sobre la base de algunos principios teóricos como son los conjuntos deterministas, estadísticos o incluso difusos.

Normalmente existen dos métodos de clasificación: basados en aprendizaje supervisado, que trata casos de los cuales se conoce *a priori* la clase a la que pertenecen y que servirán para generar una signatura característica de cada una de las clases, pueden ser paramétricos y no paramétricos [20], [21]; por otra parte, están los basados en técnicas no supervisadas, los cuales utilizan algoritmos de clasificación automática multivariada donde los individuos más próximos se van agrupando formando clases [22].

En tal sentido, gran cantidad de esfuerzos se han encauzado y se han propuesto numerosos descriptores de imagen. Basados en la porción del contenido de la imagen utilizado, estos pueden ser clasificados generalmente en dos categorías: globales y locales. Los descriptores globales toman el contenido de toda la imagen en función de obtener una representación

general de la misma y, aunque han sido empleados en numerosas aplicaciones, son en exceso dependientes de los resultados de una segmentación previa y son bastante sensibles a variaciones de la imagen, por ende y dada la orientación de la presente investigación, se hace énfasis en los descriptores locales.

Reconocidos descriptores de esta índole han sido presentados a la comunidad científica, entre los cuales destacan: el Transformador de Rasgos de Escala Invariante (SIFT) [23], *affine*-SIFT (ASIFT) [24], Rasgos Robustos Acelerados (SURF) [25] y Patrones Binarios Locales (LBP) como anteriormente se mencionaba.

De igual modo y teniendo en cuenta que la información del gradiente[4] revela la tasa de cambio de la distribución del color en imágenes, este ha sido ampliamente empleado en el diseño de dichos descriptores. El Histograma de Gradiente Orientado (HOG) [26] es un ejemplo que lo corrobora, considerando su importante aporte a la teoría de la representación de imágenes al usar rasgos determinados del gradiente en cuestión y, siguiendo esta idea, numerosos descriptores basados en gradientes han sido propuestos [27], [28]. De manera análoga, los bordes de imagen también se han analizado para derivar descriptores locales. El descriptor local Histograma de Borde (EHD), el cual brinda estadísticas sobre los diferentes tipos de borde, fue diseñado para la evaluación de correspondencias entre imágenes [29], [30]. Estos son también utilizados en conjunto con los LBP para determinar bordes locales máximos en patrones binarios para aplicaciones de recuperación de imágenes y rastreo de objetos [31]. De igual manera, Satpathy *et al.* desarrollaron el Patrón Binario Local Discriminativo Robusto (DRLBP) y Patrón Ternario Local Discriminativo Robusto (DRLTP) para utilizar la información de los bordes y la textura en reconocimiento de objetos [32]. A diferencia de las estrategias anteriormente mencionadas, varios investigadores recientemente han desarrollado descriptores locales teniendo en cuenta el principio de percepción humana para los procedimientos de extracción de características. Por ejemplo, disímiles descriptores locales de Weber (WLDs) han sido lanzados siguiendo las leyes de Weber [33], que brindan una medida

[4] Campo vectorial cuyas funciones coordenadas son las derivadas parciales del campo escalar. Se representa con el operador diferencial *nabla* (∇). Indica la dirección en la cual un campo f varía más rápidamente y su módulo representa el ritmo de variación de f en la dirección de dicho vector. Es el único vector que, multiplicado por el vector unitario, da la derivada direccional del campo escalar.

especial de contraste del contenido de la imagen y, consecuentemente, han perfeccionado el desempeño de numerosos sistemas [34] – [37].

1.4 Conceptualización del contraste de Michelson.

El contraste, de manera general, se define como la diferencia relativa en intensidad entre un punto de una imagen y sus alrededores y se considera como un atributo perceptual fundamental de la misma. Un ejemplo simple es el contraste entre un objeto de brillo constante sobre un fondo de un brillo constante. Si ambas superficies tienen el mismo brillo, el contraste será nulo y si el conjunto está en tonos de gris, el objeto será tanto física como perceptiblemente indistinguible del fondo. Según se incrementa la diferencia en brillo el objeto será perceptiblemente distinguible del fondo una vez alcanzado el umbral de contraste, que se sitúa alrededor del 0,3 % de diferencia [38].

En el caso de figuras periódicas simples, como enrejillados que varían su brillo siguiendo una función sinusoidal, se emplean otros tipos de medidas del contraste. Entre las más sencillas destaca el contraste de Michelson que es una medida definida y estable del contraste en la imagen o cierta región de interés [39], [40].

Suponiendo que Ω es una región local de una imagen, el contraste de Michelson se define como:

$$\xi_{\Omega} = \frac{I_{max} - I_{min}}{I_{max} + I_{min}} \tag{1.22}$$

Donde I_{max} e I_{min} representan los valores máximos y mínimos de intensidad respectivamente dentro de la región Ω. El resultado de ξ_{Ω} depende de los valores de I_{max} e I_{min} como sigue:

$$\xi_{\Omega} = \begin{cases} 1, & si\ I_{max} > I_{min} = 0; \\ (0{,}1), & si\ I_{max} > I_{min} > 0; \\ 0, & si\ I_{max} = I_{min}. \end{cases} \tag{1.23}$$

1.4.1 Definición del contraste de Michelson para imágenes a color en el dominio de los cuaterniones.

Una vez efectuada la representación cuaterniónica (QR) de los píxeles, de manera que, para las tres dimensiones o valores de intensidad en los respectivos planos de color de cada uno

de ellos, exista un cuaternión según la ec. 1.21, es posible definir una pequeña región en $\dot{F}(x, y)$ como sigue:

$$\dot{S} = \{\dot{I}_n | n = 1, 2, \dots, N\} \tag{1.24}$$

Donde $\dot{I}_n$ es la QR del correspondiente píxel y N denota el número de píxeles en $\dot{S}$. De aquí, y teniendo en cuenta que la ec. 1.22 indica que se necesitan los valores de intensidad máximo y mínimo, se ha de tener un criterio de orden que la satisfaga como se muestra a continuación:

$$f(\dot{S}) = \{\dot{I}_n | t_n \in \{1, \dots, N\}, \dot{I}_{t1} \leq \dot{I}_{t2} \leq \cdots \leq \dot{I}_{tN}\} \tag{1.25}$$

En la cual $\leq$ indica la relación de orden entre los cuaterniones siguiendo el criterio dado; mas no existe en la actualidad un método para ordenarlos directamente. Para solucionar este problema, se ha planteado una simple forma de hacerlo, mediante sus módulos; sin embargo, esta aproximación no tiene en cuenta las relaciones entre los canales de color, lo que se traduce en falta de precisión en los resultados esperados. En el capítulo 2 se describe el procedimiento empleado para suprimir esta problemática.

Ahora, el contraste de Michelson para la región debería quedar formulado como sigue a continuación teniendo en cuenta la ec. 1.22:

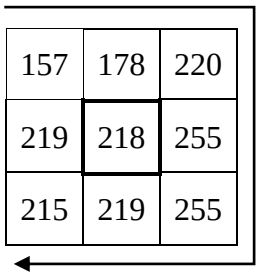
$$\xi_{\dot{S}} = \frac{\dot{I}_{tN} - \dot{I}_{t1}}{\dot{I}_{tN} + \dot{I}_{t1}} \tag{1.26}$$

Donde $\dot{I}_{tN}$ representa, según el criterio de orden establecido, el valor de intensidad del píxel representado por el último cuaternión, y consecuentemente $\dot{I}_{t1}$ del primero de ellos. Sin embargo, en la implementación del algoritmo esta fórmula por sí sola no es eficiente en la extracción del contraste.

1.5 Patrones Binarios Locales (LBPs).

LBP es un operador de textura simple pero muy eficiente que etiqueta los píxeles de una imagen por vecindad de umbral de cada píxel con el valor del píxel central, y considera el resultado como un número binario. Para comprender mejor este concepto se muestra un ejemplo sobre su principio de obtención y (o) funcionamiento:

Suponiendo que se tiene una matriz que representa las intensidades de una vecindad de píxeles dentro de una imagen, el LBP inicial que describe la misma se calcula de la siguiente manera:



218 < 157 …? 0 218 < 255 …? 1
218 < 178 …? 0 218 < 219 …? 1
218 < 220 …? 1 218 < 215 …? 0
218 < 255 …? 1 218 < 219 …? 1 LBP (1,1) = 00111101 = 61

Debido a su poder de discriminación y la simplicidad de cálculo, este operador de textura se ha convertido en un método popular que se usa en numerosas aplicaciones, de modo que puede ser visto como un enfoque unificador de los modelos tradicionalmente divergentes del análisis de texturas: los estadísticos y los estructurales [41].

Quizás la propiedad más importante del operador, para aplicaciones del mundo real, es su robustez frente a cambios en una escala de grises monótona, causados, por ejemplo, por las variaciones de iluminación, y frente a rotaciones en el caso de utilizar códigos circulares. Otra característica importante es su simplicidad computacional, lo que permite analizar las imágenes en tiempo real.

1.5.1 Análisis de textura utilizando LBPs.

LBP proporciona un operador de análisis de textura que se define como una medida de la textura en una escala de grises invariante, derivado de una definición general de textura mediante vecinos locales. La forma actual del operador LBP es muy diferente de su versión básica: la definición original se extiende a un conjunto de vecinos arbitrarios circulares, y se han desarrollado nuevas versiones del mismo, sin embargo, la idea principal es la misma: un código binario que describe el patrón de la textura local que es construido por el umbral de un conjunto de vecinos por el valor de gris de su centro. El operador tiene que ver con muchos otros métodos conocidos de análisis de texturas. En la *Fig.1.1* [41] se observan las relaciones del operador LBP con otros métodos.

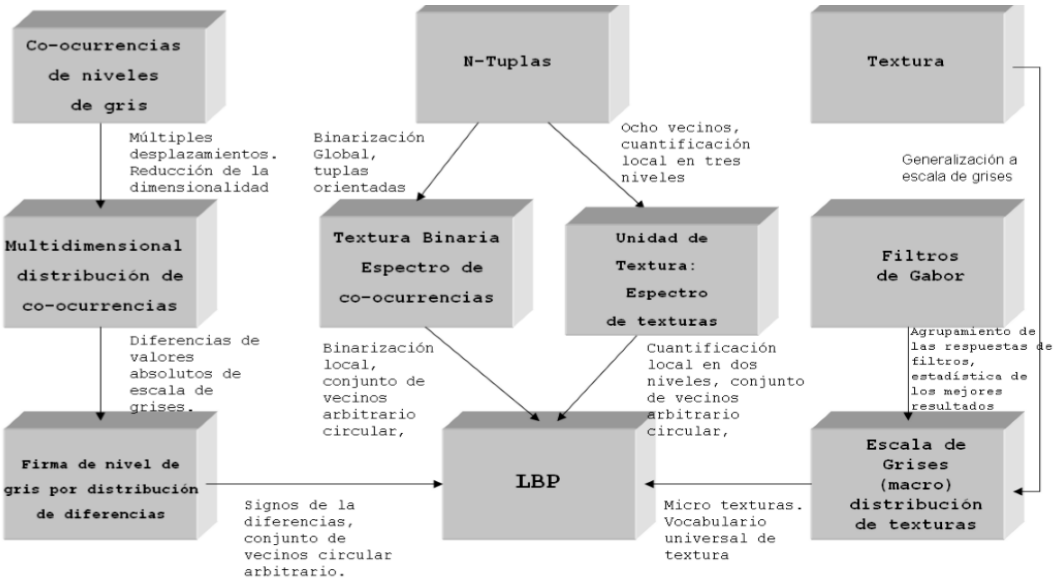


Fig.1.1 *Operador LBP en el campo de análisis de texturas.*

El operador original LBP, fue introducido por T. Ojala *et. al* [42] y es un método para describir texturas. El operador etiqueta cada uno de los píxeles de una imagen mediante el muestreo de matrices 3x3 comparando cada píxel con el valor central de la matriz y considerando el resultado como un número binario, como se explicaba anteriormente. A continuación, se calcula el histograma de las etiquetas y se concatenan los histogramas, lo cual se puede utilizar como un descriptor de textura.

La evolución del operador LBP fue desarrollada por M. Pietikäinen *et. al* [1] y posteriormente, el operador se extendió al uso de diferentes tamaños, no solo a ocho puntos, sino a muestreos circulares. Es válido destacar que la bilinealidad de estos se consigue con la interpolación de los valores de los píxeles, lo que permite utilizar cualquier radio y por lo tanto cualquier número de píxeles vecinos.

1.5.2 Derivación del operador LBP.

La derivación de este operador inicia con la definición de la textura T como la distribución conjunta de los niveles de gris de $P + 1, (P > 0)$ píxeles de una imagen de la siguiente manera:

$$T = t(g_c, g_{0,} g_1, \dots, g_{p-1}) \tag{1.27}$$

Donde g_c se corresponde con el nivel de gris del píxel central del muestreo de una matriz nxn y $g_p(p = 1, 2, \dots, p-1)$, corresponden a los valores de gris de los píxeles P equidistantes en un círculo de radio R, $(R > 0)$, que forman un conjunto circular y simétrico de los vecinos con respecto al centro. Si las coordenadas de g_c son (0, 0), entonces las coordenadas de g_p estarán dadas por $\left(-Rsin\left(\frac{2\Pi p}{P}\right), Rcos\left(\frac{2\Pi p}{P}\right)\right)$. En la *Fig. 1.2* [1] se pueden observar ejemplos para diferentes tamaños.

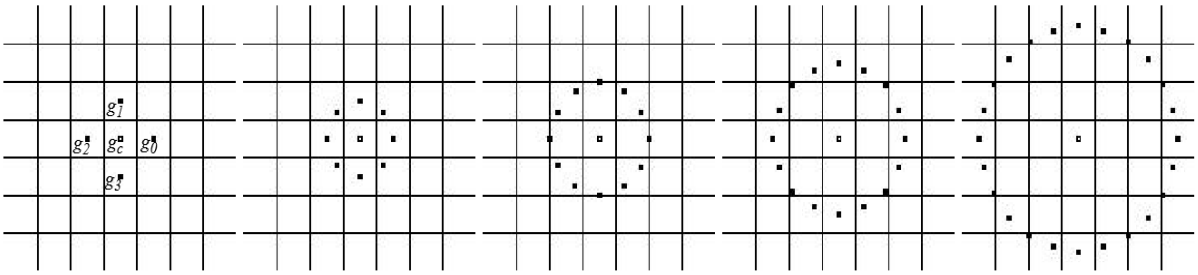


Fig. 1.2 *Simetría circular para conjuntos de vecinos.*

Luego, sin perder información se puede restar g_c de los puntos g_p, quedando:

$$T = t(g_c, g_0 - g_c, g_1 - g_c, \dots, g_{p-1} - g_c) \tag{1.28}$$

Donde, suponiendo que las diferencias son independientes la distribución, la expresión puede ser factorizada:

$$T \approx t(g_c)(g_0 - g_c, g_1 - g_c, \dots, g_{p-1} - g_c) \tag{1.29}$$

Ahora, como $t(g_c)$ describe la luminosidad general de una imagen, que no está relacionada a la textura de la imagen local, puede ser ignorada y queda:

$$T \approx (g_0 - g_c, g_1 - g_c, \dots, g_{p-1} - g_c) \tag{1.30}$$

Aunque invariante frente a cambios de escala de grises, las diferencias se ven afectadas por el cambio de escala. Para lograr invariancia con respecto a cualquier transformación monótona de esta escala, sólo es necesario considerar los signos de las diferencias como sigue:

$$T \approx t(s(g_0 - g_c), s(g_1 - g_c), \dots, s(g_{p-1} - g_c)) \tag{1.31}$$

$$\text{Donde } S(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{1.32}$$

Ahora, un peso de 2^p es asignado a cada signo $s(g_p - g_c)$, transformando la diferencia del muestreo en un único código LBP:

$$LBP_{P,R} = \sum_{p=0}^{p-1} s(g_p - g_c)^{2^p} \tag{1.33}$$

1.5.3 Invariancia a la rotación.

El operador $LBP_{P,R}$ produce 2^p valores de salida diferentes, correspondientes a los 2^p patrones binarios diferentes que se pueden formar con los P píxeles vecinos. Cuando se rota la imagen, los valores g_p de gris correspondientes se moverían a lo largo del perímetro del círculo alrededor de g_0. Puesto que g_0 está destinado a ser el valor de gris del elemento (0, R) a la derecha de g_c la rotación de un patrón binario en particular, naturalmente, da lugar a un $LBP_{P,R}$ diferente. Esto no se aplica a los patrones que constan de sólo 0s (o 1s), que permanecen constantes en todos los ángulos de rotación. Para eliminar el efecto de la rotación, es decir, para asignar un identificador único para cada rotación invariable LBP se define:

$$LBP_{P,R}^{ri} = \min\{ROR(LBP_{P,R}, i) \mid i = 0, 1, \dots P - 1\} \tag{1.34}$$

Donde $ROR(x, i)$ realiza un desplazamiento circular, a nivel de bit, hacia la derecha P-bits de x i -veces. En términos de imagen de bits la ecuación 1.34 corresponde simplemente a girar el conjunto de vecinos hacia la derecha tantas veces como el número que corresponde al bit más significativo a partir de g_{P-1}.

$LBP_{P,R}^{ri}$ cuantifica las ocurrencias de rotaciones invariables para los patrones correspondientes a ciertas micro características de la imagen, por lo que los mismos pueden ser considerados como detectores de rasgos. La *Fig. 1.3* [1] ilustra las 36 únicas rotaciones invariantes de patrones binarios locales que pueden ocurrir en el caso de P = 8, es decir, $LBP_{8,R}^{ri}$ puede tener 36 valores distintos.

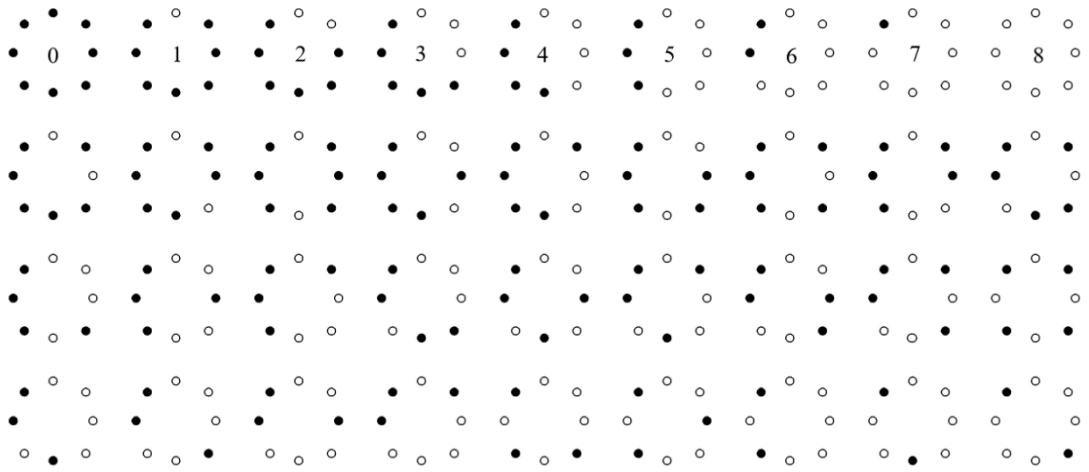


Fig. 1.3 *Rotaciones invariantes de patrones binarios que pueden ocurrir en el conjunto de ocho vecinos de la simetría circular* $LBP_{8,R}^{ri}$

Los círculos en blanco y negro se corresponden a los valores de bit de 0 y 1 en la salida de 8 bits del operador. La primera fila contiene los nueve patrones uniformes y los números dentro de ellos corresponden a su único $LBP_{8,R}^{riu2}$.

Se ha observado que existen ciertos patrones binarios que están presentes en más del 90% de las texturas generadas por muestreos locales de 3x3 píxeles. A estos patrones se les conoce con el nombre de patrones uniformes, que tienen una cosa en común, su estructura circular uniforme tiene muy pocas transiciones espaciales. Ellos funcionan como una micro plantilla, con un punto brillante (0), superficie plana o punto negro (8), y los bordes de diferente curvatura positiva y negativa (1-7).

Para definir formalmente los patrones uniformes, se introduce una medida de la uniformidad U (patrón), que se corresponde con el número de transiciones espaciales (a nivel de bits 0 / 1) cambios del patrón. Por ejemplo, los patrones 00000000_2 y 11111111_2 tienen un valor U de 0, mientras que los otros siete patrones en la primera fila de la *Fig. 1.3* tienen un valor U de 2, ya que hay exactamente dos 0 / 1 en el patrón de las transiciones. Del mismo modo, los otros 27 modelos tienen un valor U de al menos 4. Se designan a los patrones que tienen un valor de U al menos de 2 como patrones uniformes, derivándose otro operador invariante a escala de grises y rotaciones en lugar del usado hasta ahora $LBP_{P,R}^{ri}$:

$$LBP_{P,R}^{riu2} = \begin{cases} \sum_{P=0}^{P-1} s(g_P - g_c) \ \ si\ U(LBP_{P,R}) \leq 2 \\ P+1\ en\ cualquier\ otro\ caso \end{cases} \tag{1.35}$$

donde se tiene que:

$$U\left(LBP_{P,R}\right) = |s(g_{P-1} - g_c) - (g_0 - g_c)| + \sum_{P=1}^{P-1} |s(g_P - g_c) - s(g_{P-1} - g_c)| \tag{1.36}$$

En el capítulo siguiente se desglosa por pasos el procedimiento para la obtención de LBPs en imágenes a color empleando el contraste de Michelson obtenido mediante el uso de cuaterniones para la representación de las mismas; y estos patrones uniformes invariantes antes mencionados cobran especial relevancia en tal empeño.

CAPÍTULO 2. Descriptor local *Quaternion-Michelson*. Materiales y métodos.

2.1 Descriptor *Quaternion-Michelson* para la clasificación de imágenes RGB.

En el presente trabajo se pretende implementar, como anteriormente se indicó, un sencillo pero poderoso descriptor nombrado *Quaternion-Michelson Descriptor (QMD)* para la extracción de características locales útiles en la clasificación de imágenes a color. A diferencia de los descriptores locales anteriores, que extraen directamente los rasgos de la imagen en cuestión, este lo hace a partir del contraste de Michelson obtenido de la misma y la representación cuaterniónica (QR) de los planos de color. Si bien el contraste de Michelson es una medida estable del contenido de la imagen desde el punto de vista de la percepción humana, y la QR es capaz de manejar la información conjunta de color de la imagen preservando la interacción entre los diferentes canales cromáticos, QMD integra los méritos de ambos, de modo que experimentos realizados con bases de datos de imágenes demuestran su factibilidad en cada aspecto. A partir de ello y como resumen del trabajo realizado, se propone el descriptor Patrones Binarios del Contraste de Michelson Cuaterniónico (QMCBP) [5], que extrae desde dos perspectivas el contraste de Michelson y obtiene los patrones binarios locales de imágenes a color, estudio que habilita un posterior análisis de textura en imágenes de tales características.

2.2 Procedimiento general para la implementación del algoritmo.

Entrada: Teniendo la imagen a color F; L cuaterniones de referencia $\{\dot{p}_1, \dots, \dot{p}_L\}$ que representan una vecindad o región local de dicha imagen, y los parámetros P y R para la codificación de QMCBP, los pasos generales a tener en cuenta para la implementación del algoritmo son:

1- Lectura de la imagen a color F.

2- Obtención de $\dot{F}(x, y)$, representación cuaterniónica de F de acuerdo a la ec. 1.21.

3- Desde $n = 1$ hasta L:

3.1- Calcular la CTQ de $\dot{F}$ usando $\dot{p}_n$ y con ello obtener la fase correspondiente a cada uno de ellos.

3.2- Extraer el contraste de Michelson.

3.3- Realizar la codificación QMCBP para el mapa de contraste de Michelson obtenido teniendo en cuenta los parámetros $P = 8, R = 1$.

3.4- Obtener el histograma normalizado ρ_n para el resultado de la codificación QMCBP.

Salida: Mapa patrones binarios locales de F: $\{\rho_1, \dots \rho_L\}$.

A continuación, la *Fig. 2.1* muestra un esquema resumen de los pasos a seguir para la implementación del algoritmo:

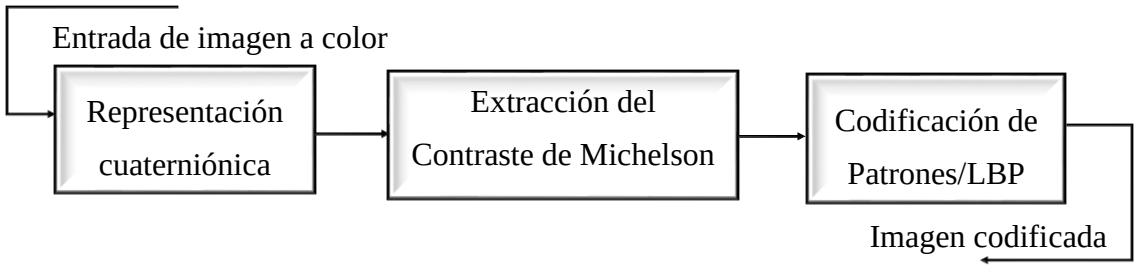
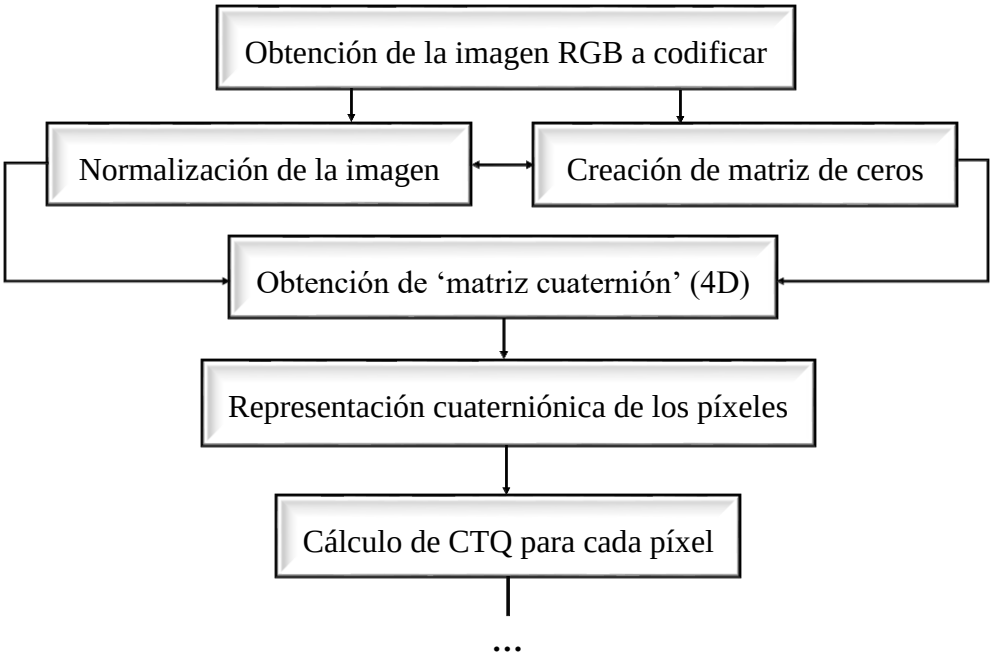


Fig. 2.1 *Esquema representativo del descriptor Quaternion-Michelson (QMD).*

Seguidamente, para mayor especificidad, se expone un diagrama de flujo que representa el algoritmo en detalles.



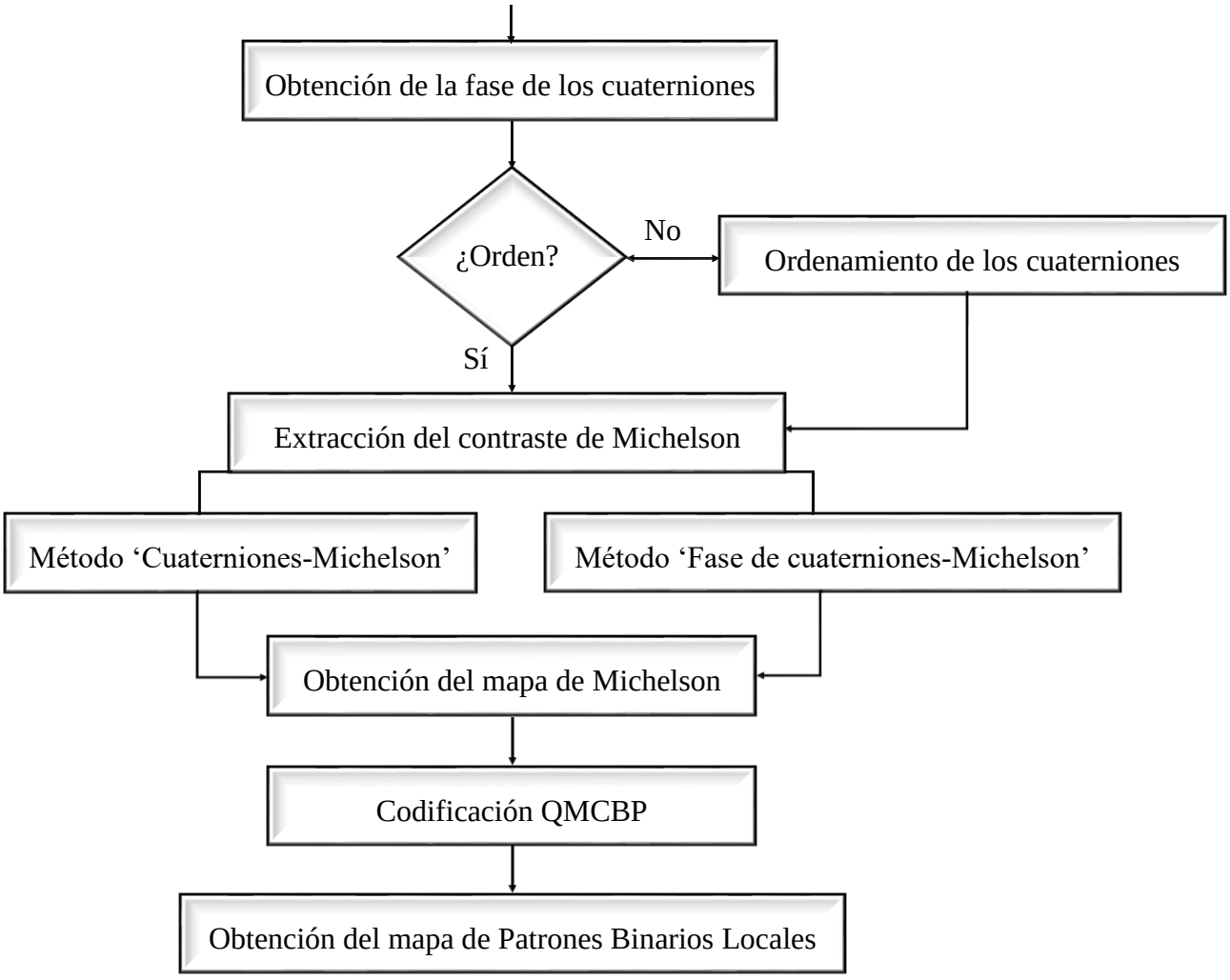


Fig. 2.2 *Diagrama de flujo del algoritmo para la implementación de QMD.*

2.2.1 Propiedades de las imágenes a codificar.

La mayoría de las imágenes utilizadas para la ejecución de experimentos y pruebas fueron obtenidas de [43]. Son imágenes RGB que poseen dimensiones de 768 píxeles de ancho por 576 de alto, con resolución horizontal y vertical de 96 ppp, profundidad en bits de 24 y extensión 'jpg'. Tales características fueron consultadas haciendo uso de la función *imfinfo* de Matlab y cargadas desde archivo para su procesamiento con *imread*. Seguidamente, se muestran algunas de estas imágenes:

Fig. 2.3 *Imágenes de texturas empleadas para los experimentos.*

Estas son imágenes de texturas cuyo contenido es bastante uniforme, útiles para estudios de clasificación; sin embargo, el resto de las imágenes empleadas tienen propiedades visiblemente diferentes, pero igualmente interesantes, entre ellas destacan [44]:

Fig. 2.4 *Otras imágenes empleadas para los experimentos.*

2.3 Representación cuaterniónica de los planos de color de los píxeles.

Primero, resulta interesante comentar que una imagen es esencialmente una función bidimensional $f(x, y)$, donde x e y representan las coordenadas espaciales, y el valor de f en cualquier par de coordenadas (x, y) representa la intensidad de la imagen en dicho punto. Una

imagen digital $f_d(x, y)$ descrita en un espacio 2D discreto se deriva de una imagen análoga $f(x, y)$ en un espacio 2D continuo a través de un proceso llamado digitalización. Al digitalizarla, la imagen continua en 2D es dividida en M filas y N columnas.

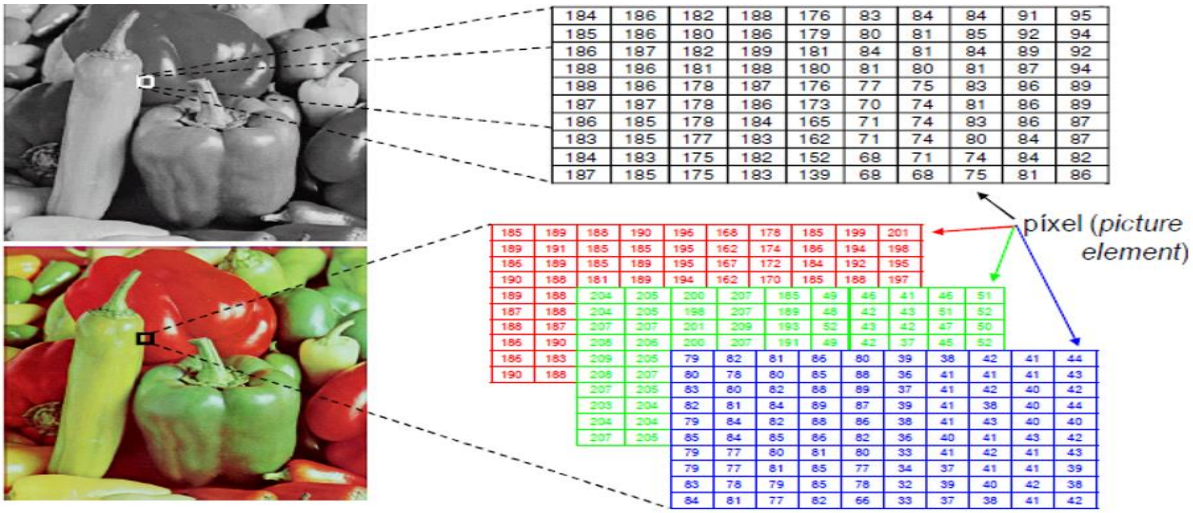


Fig. 2.5 *Representación matricial de imágenes en escala de grises y a color.*

Matlab almacena las imágenes como vectores bidimensionales (matrices), en el que cada elemento de la matriz corresponde a un solo píxel. En la *Fig. 2.5* se puede apreciar que las imágenes en escala de grises son representadas como matrices de MxN, mientras que las RGB son arreglos de dimensión MxNx3, o sea, 3 matrices de MxN. Es común, además, que una imagen contenga sub-imágenes llamadas regiones de interés (ROI) independientemente del tipo que sea.

Como parte fundamental de la implementación del algoritmo, para la extensión de las operaciones realizadas de manera local sobre la imagen se utilizó la función *nlqfilter* (Anexo I), adaptación en el dominio de los cuaterniones, hecha en un trabajo previo, de la función *nlfilter* que pertenece a la librería de Matlab. Esta barre la imagen completa en una ventana deslizante de dimensiones que se indiquen, en este caso de 3x3, de manera que permite realizar instrucciones determinadas sobre estas vecindades locales. Tales instrucciones corresponden a operaciones o funciones que se mencionarán posteriormente para el cálculo del contraste de Michelson y la codificación de patrones binarios locales.

Teniendo en cuenta lo anteriormente relacionado, para realizar la representación cuaterniónica de los planos de color se propone lo siguiente:

- Dado que según la ec. 1.21 los cuaterniones utilizados para la representación deben ser puros, es necesario obtener una 'matriz cuaternión' (por llamarla de alguna manera) de cuatro dimensiones, donde estén representados para cada píxel los niveles de intensidad de las 3 componentes de color y la parte escalar de los cuaterniones en cuestión, cuyo valor obviamente debe ser cero. Para este empeño se emplean las funciones *zeros* y *cat* de Matlab; la primera de ellas para crear una matriz de ceros de iguales dimensiones a la imagen a procesar, y la segunda para concatenar esta matriz con las matrices que representan los planos de color, obteniéndose un arreglo de MxNx4.

De esta manera, el vector que corresponde a cualquier píxel de la imagen estará representado por los elementos {0, b, c, d}, que pueden ser considerados como argumentos de cuaterniones para su uso subsecuente.

2.4 Extracción del contraste de Michelson.

La disyuntiva de la extracción del contraste de Michelson basada en la representación cuaterniónica de los píxeles de la imagen radica en obtener un ordenamiento de los cuaterniones implicados por vecindad de tal forma que sea posible definir un máximo y un mínimo y, con ello, precisar la medida de contraste que refiere cada uno a partir de sus respectivas intensidades teniendo en cuenta la ec. 1.26.

De ahí, se precisan los siguientes pasos para la obtención de dicho contraste una vez representada la imagen mediante cuaterniones.

1. Se establece un criterio de orden de los píxeles representados, en este caso atendiendo a la fase de los cuaterniones dado que no existe una relación que logre ordenarlos teniendo en cuenta simultáneamente la información de color de los planos R-G-B. Se obtiene la fase de cada uno de ellos y se ordenan ascendentemente utilizando la función *sort*.
2. Ya ordenados, se emplea uno de los métodos descritos en el epígrafe 2.4.2 para la extracción del contraste en dependencia de la aplicación sucesiva que se requiera, en este caso se utiliza el método basado en la fase de los cuaterniones (subep. 2.4.2.2), teniendo en cuenta su sencillez de implementación y la reducción del tiempo de

cómputo que representa según comparaciones realizadas. Para tal fin, se programó la función *michelson* cuyo código se encuentra disponible en el Anexo I.

Los parámetros que recibe la función son el tamaño de la región local que procesa, el cuaternión unitario que será empleado en la CTQ y el método a emplear. Esta trabaja con la matriz cuaternión antes creada y ejecuta su operación en una vecindad de dimensiones que se indiquen, en este caso de 3x3 como anteriormente se mencionó, y consecuentemente devuelve el contraste correspondiente a dicha región.

2.4.1 Ordenamiento de los cuaterniones atendiendo a su fase.

En el capítulo 1 se comentaba que no es posible ordenar los cuaterniones y a la vez lograr precisión en los resultados utilizando una transformación en el dominio de los reales ni a partir de sus módulos, puesto que estas aproximaciones desestiman la relación entre los canales cromáticos; sin embargo, trabajos precedentes [45] consideran que una solución viable tiene su esencia en utilizar la fase de los mismos. El procedimiento se basa en obtener la fase de cada cuaternión y establecer un orden entre ellos teniendo en cuenta este criterio.

Para el cálculo de la fase se emplean las cuatro componentes del cuaternión, incluyendo la parte real según indica la ec. 1.8 que se recuerda oportunamente a continuación:

$$\theta = arctan\frac{\sqrt{b^2 + c^2 + d^2}}{a}$$

Esto pudiera ser interpretado como una dificultad para el algoritmo dado que en la representación cuaterniónica de los píxeles de color se utilizan cuaterniones puros, donde $a = 0$, obteniéndose siempre el mismo resultado aunque b, c y d sean diferentes. A pesar de esto, es posible calcular dicha fase.

Seguidamente se muestran los pasos a seguir para el ordenamiento de los cuaterniones:

1. Teniendo una región representada mediante QR, se realiza la traslación de Clifford de cada píxel representado de manera que aparezca una cuarta componente a ($a \neq 0$) en el dominio del cuaternión y sea posible calcular la fase consecuentemente, dando solución a la problemática antes planteada.

2. Luego se obtiene la fase utilizando la ec. 1.8 y se crea un vector con los valores de los nueve cuaterniones (en vecindad de 3x3), de forma que sea posible visualizar la relación de orden entre ellos, si no están ordenados el paso inmediato es hacerlo.

2.4.1.1 Cálculo de la Traslación de Clifford.

CTQ es una simple transformación de un cuaternión y la fase del resultado de ella puede ser empleada para medir la similitud entre dos cuaterniones; por ende, se aplica CTQ previo al ordenamiento de la siguiente manera:

Teniendo que $\dot{S} = \{\dot{I}_n | n = 1, 2, \dots, N\}$ es una pequeña región en la imagen representada mediante cuaterniones, $\dot{\rho}$ un cuaternión unitario y $\dot{S}_{\dot{\rho}}$ el resultado de CTQ en $\dot{S}$, se pude obtener:

$$\dot{S}_{\dot{\rho}} = \{\dot{I}_n \dot{\rho} | n = 1, 2, \dots, N\} \tag{2.1}$$

En este caso fue utilizada la traslación derecha de Clifford, aunque un resultado de orden similar sería obtenido empleando la traslación izquierda. El cuaternión unitario define la dirección en la cual se hace la traslación, dígase i, j o k, siendo empleados vectores de la forma [0 1 0 0], [0 0 1 0] y [0 0 0 1] para cada caso respectivamente. En la función *michelson* es posible definir cuál de ellos utilizar; los resultados visiblemente varían.

Para el cálculo de la traslación de Clifford se programó la función *clifford* que se encarga de ejecutar la CTQ teniendo como parámetros de entrada un cuaternión extraído de la matriz cuaternión, uno de los cuaterniones unitarios antes ejemplificados y el sentido en que se realiza la traslación. El código que lo respalda se encuentra disponible en el Anexo I.

2.4.1.2 Obtención de la fase de los cuaterniones.

Seguido del paso antes descrito, se computa la fase de cada elemento en $\dot{S}_{\dot{\rho}}$ como sigue:

$$\phi = \{\phi_n | \phi_n = \theta(\dot{I}_n \dot{\rho}), n = 1, 2, \dots, N\} \tag{2.2}$$

Para su consumación se utilizan las funciones *sqrt* en el numerador de la ec. 1.8 y *atan2* para la obtención del ángulo.

Es importante aclarar aquí que se obtiene el mismo ϕ aplicando una traslación de Clifford izquierda haciendo uso de la ec. 1.20. Según se indicaba anteriormente, ϕ_n describe la relación entre $\dot{I}_n$ y $\dot{\rho}$, y ϕ puede ordenarse ascendente o descendentemente teniendo presente que

$\phi_n \in [0, \pi]$. Ahora, si se denota el arreglo de fase ordenado como ϕ', una representación del mismo quedaría como sigue:

$$\phi' = \{\phi_{tn} | \phi_{t1} \leq \phi_{t2} \leq \cdots \leq \phi_{tN},\ t_n \in \{1, 2, \ldots, N\}\} \tag{2.3}$$

Donde t_n es el elemento indicado en $\dot{S}$.

De esta manera, es posible establecer una relación entre la fase de cada cuaternión y los cuaterniones en cuestión. Así, las fases de orden ϕ_{t1} y ϕ_{tN} indican qué cuaternión es el primero y cuál es el último. Consecutivamente es necesario determinar qué cuaterniones corresponden a estos ángulos, para lo cual se utiliza la función *find* de Matlab, la cual en este caso encuentra la posición del primer y último cuaternión en la matriz inicial que los contiene (matriz cuaternión) teniendo en cuenta el orden de la fase.

2.4.2 Métodos para la obtención del contraste.

Una vez obtenido ϕ' se proponen dos métodos para la extracción del contraste de Michelson en el dominio de los cuaterniones, a continuación se describen detalladamente.

2.4.2.1 Extracción del contraste de Michelson basado en cuaterniones.

Cuando t_n esté listo para ser manejado según las condiciones que se precisan, se pueden ordenar los elementos en $\dot{S}$ de la siguiente forma:

$$\dot{S}' = \left\{\dot{I}_{tn} | t_n \in \{1, 2, \ldots, N\},\ \phi_{t1} \leq \phi_{t2} \leq \cdots \leq \phi_{tN}\right\} \tag{2.4}$$

Donde $\dot{I}_{t1}$ es considerado el mínimo en $\dot{S}$ e $\dot{I}_{tn}$ el máximo. Ahora, si directamente son sustituidos estos en la ecuación general del cálculo del contraste de Michelson, como quedó recogido en la ec. 1.26, $\xi_{\dot{S}}$ será todavía un cuaternión, lo cual es un inconveniente para otros procesos relacionados que sobrevienen, entre ellos la codificación de patrones binarios. Es necesario obtener, por ende, un número real como resultado.

Dado que la fase de los cuaterniones describe la relación entre sus componentes, es posible obtener los resultados esperados calculando la fase del contraste definido como $\xi_{\dot{S}}$. A continuación, se presenta la ecuación que lo reseña:

$$\xi_{\dot{S}} = \theta\left(\frac{\dot{I}_{tN} - \dot{I}_{t1}}{\dot{I}_{tN} + \dot{I}_{t1}}\right) \tag{2.5}$$

A propósito y en pos de ofrecer un análisis más profundo se analiza $\xi_{\dot{S}}$ a partir de todas las componentes de color. Si $\dot{\rho} = iR_{t0} + jG_{t0} + kB_{t0}$, $\dot{I}_{t1} = iR_{t1} + jG_{t2} + kB_{t3}$ e $\dot{I}_{tN} = iR_{tN} + jG_{tN} + kB_{tN}$, donde $\{R_{t0}, G_{t0}, R_{t0}\}$, $\{R_{t1}, G_{t1}, R_{t1}\}$ y $\{R_{tN}, G_{tN}, R_{tN}\}$ son las correspondientes intensidades de color de $\dot{\rho}, \dot{I}_{t1}$ e $\dot{I}_{tN}$, basado en la definición del contraste de Michelson, la fórmula quedaría descrita como:

$$\frac{\dot{I}_{tN} - \dot{I}_{t1}}{\dot{I}_{tN} + \dot{I}_{t1}} = \frac{i\Delta R + j\Delta G + k\Delta B}{i\Theta R + j\Theta G + k\Theta B} \tag{2.6}$$

Donde $\Delta R, \Delta G$ y ΔB representan las diferencias entre la última y primera componente de color de los píxeles tratados en cada caso, o sea, $\Delta R = R_{tN} - R_{t1}$, $\Delta G = G_{tN} - G_{t1}$ y $\Delta B = B_{tN} - B_{t1}$. Por otra parte, $\Theta R, \Theta G$ y ΘB son una medida de la intensidad total en cada dupla como sigue: $\Theta R = R_{tN} + R_{t1}$, $\Theta G = G_{tN} + G_{t1}$ y $\Theta B = B_{tN} + B_{t1}$. Ahora, sustituyendo la ec. 2.6 en la ec. 2.5, el resultado para la extracción del contraste de Michelson a través de este método se calcula como:

$$\xi_{\dot{S}} = arctan\left(\frac{\sqrt{(\Delta G\Theta B - \Delta B\Theta G)^2 + (\Delta B\Theta R - \Delta R\Theta B)^2 + (\Delta R\Theta G - \Delta G\Theta R)^2}}{(\Delta R\Theta R + \Delta G\Theta G + \Delta B\Theta B)}\right) \tag{2.7}$$

Es posible observar que $\xi_{\dot{S}}$ maneja la interacción entre Δ y Θ para describir el contraste local. Si Δ es similar a Θ, el numerador de $\xi_{\dot{S}}$ será pequeño y el denominador grande porque el producto efectuado es análogo al producto de vectores, siendo consecuentemente el contraste de pequeño valor en este caso y grande en caso contrario. Comparados con las intensidades de color, Δ y Θ son simples rasgos de la región local desde diferente perspectiva; y su combinación en la ec. 2.7 provee un punto de vista que involucra más información del contenido de la imagen.

2.4.2.2 Extracción del contraste de Michelson basado en la fase de cuaterniones.

Aunque el método anterior provee mayor información del contenido de la imagen, el método que se propone a continuación permite derivar el descriptor local de una manera más simplificada teniendo en cuenta que ϕ en la ec. 2.2 contiene igualmente información determinante sobre la imagen original. Dicha versión de $\xi_{\dot{S}}$ queda formulada como sigue:

$$\xi'_{\dot{S}} = \frac{\phi_{tN} - \phi_{t1}}{\phi_{tN} + \phi_{t1}} \tag{2.8}$$

En este caso el contraste de Michelson extraído en la región local es el resultado directo de insertar la fase obtenida en la ecuación original que define dicho contraste. $\xi'_{\dot{S}}$ brinda una medida similar a la del método antes descrito, sin embargo, puede ser considerada como una representación de nivel superior de las imágenes a color.

2.5 Codificación de patrones.

De las ecs. 2.5 y 2.8 no es difícil discernir que $\xi_{\dot{S}} \in [0, \pi]$ y $\xi'_{\dot{S}} \in [0, 1]$, razón que resulta interesante en el estudio de las distribuciones de contraste en imágenes naturales. Sin embargo, si directamente son utilizados los histogramas resultantes de cada caso como descriptores de imágenes a color, los rasgos obtenidos pueden obviar características esenciales que las distinguen. En función de resolver esta problemática se introduce un procedimiento de codificación del contraste de Michelson obtenido.

Dado que los patrones binarios locales (LBP) han probado ser una simple y efectiva herramienta en la clasificación de texturas y reconocimiento de rostros y tienen amplia aceptación en este ámbito, se propone implementar LBP en imágenes de contraste de Michelson representadas mediante cuaterniones para obtener un descriptor más poderoso llamado Patrones Binarios del Contraste de Michelson Cuaterniónico *(QMCBP)* como antes se citaba.

A continuación, teniendo en cuenta las ecuaciones enunciadas en el epígrafe 1.5, se presenta su descripción [5]:

$$QMCBP_{P,R} = \begin{cases} l(\sum_{n=0}^{P-1} \delta(\xi_n - \xi_c) 2^n), si\ H \leq 2; \\ P(P-1) + 2\ \ en\ otros\ casos. \end{cases} \tag{2.9}$$

Donde $H = |\delta(\xi_{P-1} - \xi_c) - \delta(\xi_0 - \xi_c)| + \sum_{n=1}^{P-1} |\delta(\xi_n - \xi_c) - \delta(\xi_{n-1} - \xi_c)|$

(2.10) con $\delta(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$

ξ_c es el píxel central de la región local de interés tomada; P es el número de píxeles alrededor de ξ_c, mientras R es la distancia entre ξ_n y ξ_c; $l(x)$ es una función que asigna un índice particular a cada patrón uniforme.

En el subepígrafe siguiente se resumen los aspectos más relevantes de las funciones utilizadas para este empeño.

2.5.1 Construcción de LBPs de imágenes a color empleando el contraste de Michelson calculado con empleo de cuaterniones.

Para la obtención de patrones binarios locales se emplearon dos funciones: *lbpgray_all* que fue programada anteriormente en un trabajo interno del departamento y cuyo código se encuentra en el Anexo I, y *extractLBPFeatures* que pertenece a la librería de Matlab.

La primera de ellas devuelve los patrones invariantes a la rotación, así como la clase a la que pertenecen empleando 8 puntos y radio 1 según la *Fig.1.3*. Las primeras nueve clases corresponden a los patrones uniformes y se designan de clase 10 todos aquellos que posean más de dos transiciones espaciales. De aquí, se escoge el menor patrón en notación decimal (que coincide con uno de los que menos transiciones de 1 a 0 o viceversa realizan). Teniendo en cuenta este valor, es posible extender los patrones calculados por vecindad a toda la imagen, obteniéndose una matriz de iguales dimensiones a la imagen original, un mapa de patrones binarios locales; en el capítulo siguiente se muestran algunos resultados.

La segunda devuelve, en dependencia de los parámetros que se le introduzcan, un vector de 1xN donde N representa el número de rasgos de los LBPs uniformes extraídos que contienen información de utilidad para tareas de detección, reconocimiento y clasificación de texturas normalmente en imágenes en escala de grises, sin embargo, en este trabajo se hace una formulación para la obtención de dichos rasgos en imágenes a color, en cuyo caso se emplea como argumento de entrada el mapa de contrastes de Michelson obtenido previamente, que es una distribución de los niveles de intensidad que caracterizan la imagen RGB inicial en términos de dicho contraste.

Entre los parámetros que se le pueden introducir están *NumNeighbors* y *Radius* que indican el número de puntos vecinos y el radio de los patrones con simetría circular a obtener; por defecto estos valores son de 8 y 1 respectivamente. También se encuentra *Uprigth*, que es una bandera de invariancia a la rotación, especificado por dos parámetros separados por coma: *Upright* y un escalar lógico de modo que, si este es indicado como *true*, los rasgos de los LBPs no codifican información de rotación y razonablemente ha de estar como *false* si son requeridos rasgos de invariancia a la rotación. En dependencia de ello será el valor de N

antes mencionado, devolviendo un vector de 1x59 en el primer caso y de 1x10 en el segundo. Otros parámetros pueden ser *Interpolation, Normalization y CellSize* que es posible consultarlos en la ayuda de Matlab a partir de su versión 2015[b].

2.6 Obtención del histograma normalizado para el resultado de la codificación.

La función final empleada en el análisis de la textura es el histograma de los resultados del operador, o sea, las etiquetas del patrón acumulado a lo largo de una muestra de textura. La razón por la que el histograma de los patrones uniformes proporciona una mejor discriminación en comparación con el histograma de todos los patrones individuales, se reduce a las diferencias en sus propiedades estadísticas.

Para la obtención de este histograma se utiliza la función *hist*. Los rasgos derivados pueden ser empleados, como se añadió antes, para la clasificación de imágenes en cuanto a su textura. Para ello, sería pertinente exportar los resultados obtenidos y procesar los datos en un *software* especializado, entre los cuales destaca WEKA. Es importante mencionar que para el caso de la función *extractLBPFeatures* no es necesario pasar por el paso de obtener el histograma, dado que esta devuelve el vector de rasgos necesario.

2.7 Funciones de Matlab empleadas.

Seguidamente, la *Tabla 2.1* muestra una breve descripción de las funciones más importantes que fueron utilizadas en la implementación del algoritmo y pertenecen a la librería de Matlab. La información que corresponde a cada función fue obtenida de la ayuda de Matlab 2017[a].

Tabla 2.1 *Descripción de algunas funciones de Matlab utilizadas para la implementación del algoritmo.*

Función	**Breve descripción**
atan2	Devuelve la tangente inversa en el intervalo cerrado $[-\pi, \pi]$.
bi2de	Convierte un vector fila binario en un entero decimal no negativo.
cat	Concatena arreglos en la cantidad de dimensiones que se especifiquen.

extractLBPFeatures	Devuelve los rasgos derivados del histograma de los patrones binarios locales uniformes extraídos de una imagen en escala de grises. Los rasgos LBP codifican información local de textura.
fliplr	Retorna un arreglo con las columnas abatidas en la dirección izquierda-derecha (sobre un eje vertical).
histogram	Representa el histograma como una variable cuantitativa y continua en forma de barras, en este caso de ancho uniforme.
im2double	Convierte una imagen de intensidad a doble precisión, haciendo un reajuste de escala de ser necesario.
imread	Lee una imagen desde archivo especificada por su nombre; infiriendo el formato de la misma teniendo en cuenta su contenido.
length	Retorna la longitud de la mayor dimensión de un arreglo; de manera que, si este es un vector simplemente devuelve el número de elementos del mismo. En caso de ser un arreglo nulo la longitud es cero.
quatmod	Calcula el módulo de un cuaternión dado.
quatmultiply	Calcula el producto de dos cuaterniones.
size	Devuelve un vector fila cuyos elementos contienen información sobre las correspondientes dimensiones de un arreglo.
sqrt	Calcula la raíz cuadrada de cada elemento de un arreglo determinado.
squeeze	Crea un arreglo a partir de otro cuyos elementos pertenezcan a más de dos dimensiones, de manera que los elementos resultantes son los mismos y las dimensiones independientes quedan solapadas.
zeros	Retorna un arreglo donde todos los elementos son cero.

CAPÍTULO 3. Análisis y discusión de los resultados obtenidos.

En el presente capítulo se hace un análisis cualitativo y cuantitativo de los resultados obtenidos como consecuencia de la implementación del algoritmo y del tratamiento de imágenes a color a partir del mismo. Se muestran resultados parciales de cada paso y como resultado final, la obtención del mapa de patrones binarios locales para estas imágenes, haciendo comparaciones oportunamente entre los diferentes métodos empleados en función de arribar a conclusiones sobre ello.

3.1 Resultados de la representación cuaterniónica de los píxeles.

Para la representación de los píxeles mediante cuaterniones se creó una matriz que fue llamada ‘matriz cuaternión’, de cuatro dimensiones, en la cual cada dimensión representa los coeficientes de los planos de color respectivamente y la componente escalar del cuaternión es igual a cero.

Los resultados que se muestran en la Tabla 3.1 corresponden a la vecindad de 3x3 que contiene los 9 primeros píxeles de cada imagen de muestra tomada. Fueron seleccionadas al azar solo tres imágenes para exponer los resultados obtenidos en este y los pasos que prosiguen en aras de evitar redundancia en el informe, teniendo en cuenta que los resultados son similares para otros casos; a continuación se muestran las mismas:

Fig. 3.1 *Imágenes empleadas para la exposición de los resultados.*

En lo sucesivo, en el orden en que aparecen, serán nombradas como **Img1**, **Img2** e **Img3**; donde **Img1** identifica a la imagen *PeppersRGB*, **Img2** a *samplepippin_RS_YachtClub019* e

Img3 a *samplepippin_Peel065* según los nombres extraídos de los repositorios de imágenes fuentes [43], [44].

Sin más, se muestra la tabla que encierra cómo quedaron concebidas algunas de las matrices que representan los niveles de intensidad normalizados de los píxeles de color en cada plano.

Tabla 3.1 *Representación cuaterniónica de los píxeles de color en la primera vecindad de 3x3.*

Img	Componente nula: $a = 0$			Componente de Rojo: $R(x,y)\boldsymbol{i}$			Componente de Verde: $G(x,y)\boldsymbol{j}$			Componente de Azul: $B(x,y)\boldsymbol{k}$		
Img1	0	0	0	0.5333	0.4824	0.5059	0.7098	0.6667	0.7020	0.1686	0.1490	0.1373
	0	0	0	0.5176	0.5373	0.5333	0.6863	0.7059	0.6863	0.1647	0.1490	0.1490
	0	0	0	0.5373	0.5176	0.5294	0.7020	0.6745	0.6902	0.1412	0.1373	0.1333
Img2	0	0	0	0.1804	0.1020	0.1216	0.6471	0.4745	0.4784	0.8431	0.6745	0.6980
	0	0	0	0.1569	0.1608	0.1529	0.7059	0.6431	0.6039	0.9098	0.8431	0.8118
	0	0	0	0.1882	0.1725	0.1255	0.7176	0.7255	0.7137	0.9294	0.9294	0.9294
Img3	0	0	0	0.7569	0.6980	0.7373	0.7059	0.7137	0.7490	0.4118	0.4196	0.4627
	0	0	0	0.7529	0.6902	0.6980	0.6745	0.6902	0.7412	0.3882	0.4039	0.4471
	0	0	0	0.6431	0.6667	0.6588	0.6000	0.6706	0.7059	0.3333	0.3961	0.4157

Como es posible observar, los valores de intensidad se encuentran en el intervalo cerrado [0, 1], lo cual se debe a que la imagen original en cada caso fue normalizada de manera que los píxeles, antes representados por 1 Byte (enteros de 8 bits, en el rango [0, 255]), ahora lo están por números en punto flotante que utilizan 8 Bytes por elemento.

Durante el momento de programación del algoritmo se constató que este procedimiento es necesario para garantizar la efectividad de operaciones complejas con cuaterniones cuyo dominio o rango de valores resultantes esté dentro de lo permitido por el *software* empleado, de lo contrario pueden ocurrir errores en tiempo de ejecución.

El paso siguiente para el trabajo con los píxeles representados mediante cuaterniones fue extraer dichos valores de intensidad de forma tal que, si se desea operar, por ejemplo, con el píxel ubicado en la posición (1,1) de **Img1**, el cuaternión consecuente sería $0 + 0.5333i +$

$0.7098j + 0.1686k$, que interpretado por Matlab es una matriz (o vector) de dimensiones 1x4 como se muestra a continuación: [0 0.5333 0.7098 0.1686].

A propósito de ello y en función de facilitar y organizar el trabajo se almacenaron los cuaterniones en una misma matriz de solo dos dimensiones (4x9), donde cada columna representa los cuaterniones en cuestión situados de izquierda a derecha. Así, por ejemplo, la primera columna corresponde al píxel de la posición (1,1), la cuarta al de la ubicación (2,1) y la novena al elemento de la (3,3). Seguidamente se muestran los resultados que corresponden a la vecindad antes mostrada de **Img1** e **Img2.**

Tabla 3.2 *Matriz cuaternión en dos dimensiones para Img1 e Img2.*

Img	**Matriz cuaternión 2D**								
Img1	0	0	0	0	0	0	0	0	0
	0.5333	0.4824	0.5059	0.5176	0.5373	0.5333	0.5373	0.5176	0.5294
	0.7098	0.6667	0.7020	0.6863	0.7059	0.6863	0.7020	0.6745	0.6902
	0.1686	0.1490	0.1373	0.1647	0.1490	0.1490	0.1412	0.1373	0.1333
Img2	0	0	0	0	0	0	0	0	0
	0.1804	0.1020	0.1216	0.1569	0.1608	0.1529	0.1882	0.1725	0.1255
	0.6471	0.4745	0.4784	0.7059	0.6431	0.6039	0.7176	0.7255	0.7137
	0.8431	0.6745	0.6980	0.9098	0.8431	0.8118	0.9294	0.9294	0.9294

Aunque esta representación organiza los cuaterniones, aún no están ordenados de la forma en que el contraste de Michelson a extraer lo demanda (o no es posible saberlo en este punto); por consiguiente es necesario, como se describió en apartados precedentes, ordenarlos atendiendo a su fase.

3.1.1 Resultados de la traslación de Clifford aplicada a la matriz cuaternión.

Si bien la fase se emplea como criterio de orden, para su obtención se ha de aplicar la traslación de Clifford a cada cuaternión de la matriz que los contiene. A continuación, la Tabla 3.3 expone los resultados de computar esta operación sobre **Img1** en la vecindad mostrada, utilizando la dirección de los tres vectores unitarios.

Esta tabla muestra una traslación derecha de Clifford dado que fue esta la utilizada en la implementación del algoritmo, sin embargo, como antes se señaló, utilizar la traslación izquierda no haría variar el resultado sucesivo de la fase.

Tabla 3.3 *Traslación de Clifford derecha aplicada a la matriz cuaternión de la vecindad mostrada de Img1.*

Vector unitario	**Traslación derecha Clifford**								
$\boldsymbol{i} = [\mathbf{0\ 1\ 0\ 0}]$	-0.5333	-0.4824	-0.5059	-0.5176	-0.5373	-0.5333	-0.5373	-0.5176	-0.5294
	0	0	0	0	0	0	0	0	0
	0.1686	0.1490	0.1373	0.1647	0.1490	0.1490	0.1412	0.1373	0.1333
	-0.7098	-0.6667	-0.7020	-0.6863	-0.7059	-0.6863	-0.7020	-0.6745	-0.6902
$\boldsymbol{j} = [\mathbf{0\ 0\ 1\ 0}]$	-0.7098	-0.6667	-0.7020	-0.6863	-0.7059	-0.6863	-0.7020	-0.6745	-0.6902
	-0.1686	-0.1490	-0.1373	-0.1647	-0.1490	-0.1490	-0.1412	-0.1373	-0.1333
	0	0	0	0	0	0	0	0	0
	0.5333	0.4824	0.5059	0.5176	0.5373	0.5333	0.5373	0.5176	0.5294
$\boldsymbol{k} = [\mathbf{0\ 0\ 0\ 1}]$	-0.1686	-0.1490	-0.1373	-0.1647	-0.1490	-0.1490	-0.1412	-0.1373	-0.1333
	0.7098	0.6667	0.7020	0.6863	0.7059	0.6863	0.7020	0.6745	0.6902
	-0.5333	-0.4824	-0.5059	-0.5176	-0.5373	-0.5333	-0.5373	-0.5176	-0.5294
	0	0	0	0	0	0	0	0	0

Los resultados perceptiblemente varían, y la utilización de uno u otro vector depende exclusivamente de la aplicación práctica que se requiera.

En la tabla, además, es posible apreciar la aparición de la parte real de los cuaterniones (representada en la primera fila) que antes era cero y es imprescindible para el cálculo de la fase; arribado a este momento ya es posible calcularla.

3.1.2 Resultados de la obtención y ordenamiento de la fase de los cuaterniones.

Los resultados de la obtención de la fase para cada cuaternión se resumen en la Tabla 3.4 que se muestra seguidamente. Estos valores iniciales indican el ángulo en radianes que corresponde a cada cuaternión, teniendo en cuenta la misma organización establecida por la matriz cuaternión. En este momento se comprueba si los cuaterniones originalmente estaban ordenados, en caso de que no sea así, el paso inmediato es ordenarlos; sin embargo, según experimentos realizados, las probabilidades indican que así será en la mayoría de los casos.

Tabla 3.4 *Fase de los cuaterniones (radianes).*

Img	Vector de fase (ϕ)								
Img1	2.2019	2.1858	2.1862	2.2034	2.2107	2.2203	2.2146	2.2157	2.2161
Img2	1.7390	1.6937	1.7137	1.7064	1.7213	1.7210	1.7298	1.7160	1.6776
Img3	2.3177	2.2713	2.2679	2.3396	2.2828	2.2495	2.3235	2.2786	2.2479

Como se puede apreciar, los cuaterniones por defecto no se encuentran ordenados en ninguno de los casos; por consiguiente se procede a ordenarlos siguiendo cualquier dirección de orden, en este caso de menor a mayor, quedando como resultado la tabla siguiente:

Tabla 3.5 *Fase de los cuaterniones ordenada (radianes).*

Img	Vector de fase ordenado (ϕ')								
Img1	2.1858	2.1862	2.2019	2.2034	2.2107	2.2146	2.2157	2.2161	2.2203
Img2	1.6776	1.6937	1.7064	1.7137	1.7160	1.7210	1.7213	1.7298	1.7390
Img3	2.2479	2.2495	2.2679	2.2713	2.2786	2.2828	2.3177	2.3235	2.3396

De esta manera es posible conocer cuáles son los píxeles representados por cuaterniones que representan los valores extremos del criterio de orden establecido, entiéndase Imáx e Imín; necesarios para el cálculo del contraste de Michelson.

Para el primer caso, el mínimo corresponde al píxel de la posición (1,2) de la vecindad; para el segundo y el tercero al de la posición (3,3). Los máximos coinciden con las posiciones (2,3); (1,1) y (2,1) respectivamente para cada imagen.

Una vez en este punto se procede a calcular el contraste en la vecindad aplicando uno u otro método de los antes descritos y luego se barre la imagen completa en una ventana deslizante del tamaño de dicha vecindad.

3.2 Resultados de la extracción del contraste de Michelson.

El barrido de la imagen, como anteriormente se refirió, se hace utilizando la función *nlqfilter*. Ahora, como esta función devuelve el contraste que corresponde a la vecindad en el píxel

central de la misma, para los píxeles de los bordes de la imagen (que no tienen 8 píxeles alrededor), los puntos que complementan la vecindad en cuestión son considerados de valor cero, por lo tanto resulta evidente deducir que el contraste de Michelson en los extremos de la imagen es siempre igual a uno utilizando el método basado en la fase de los cuaterniones, dado que es el resultado de dividir el máximo por él mismo, teniendo en cuenta que el mínimo siempre es cero. En el caso del otro método dicho contraste será igual a cero ya que el argumento de la tangente inversa siempre será cero y, por ende, el resultado también. Sin embargo, teniendo en cuenta algunos análisis derivados de los resultados, es preferible despreciar la información que brindan los bordes dado que, como estos valores son consecuencia del trabajo con píxeles 'fantasmas' (que no forman parte del contenido de la imagen) y ocupan un considerable por ciento dentro de la misma, es posible que el resultado real se vea comprometido, condición que no debe permitirse en ninguna aplicación sucesiva. Para el resto de la imagen, los valores varían de acuerdo al método y al vector unitario que se utilice. Los histogramas que se muestran a continuación documentan este planteamiento:

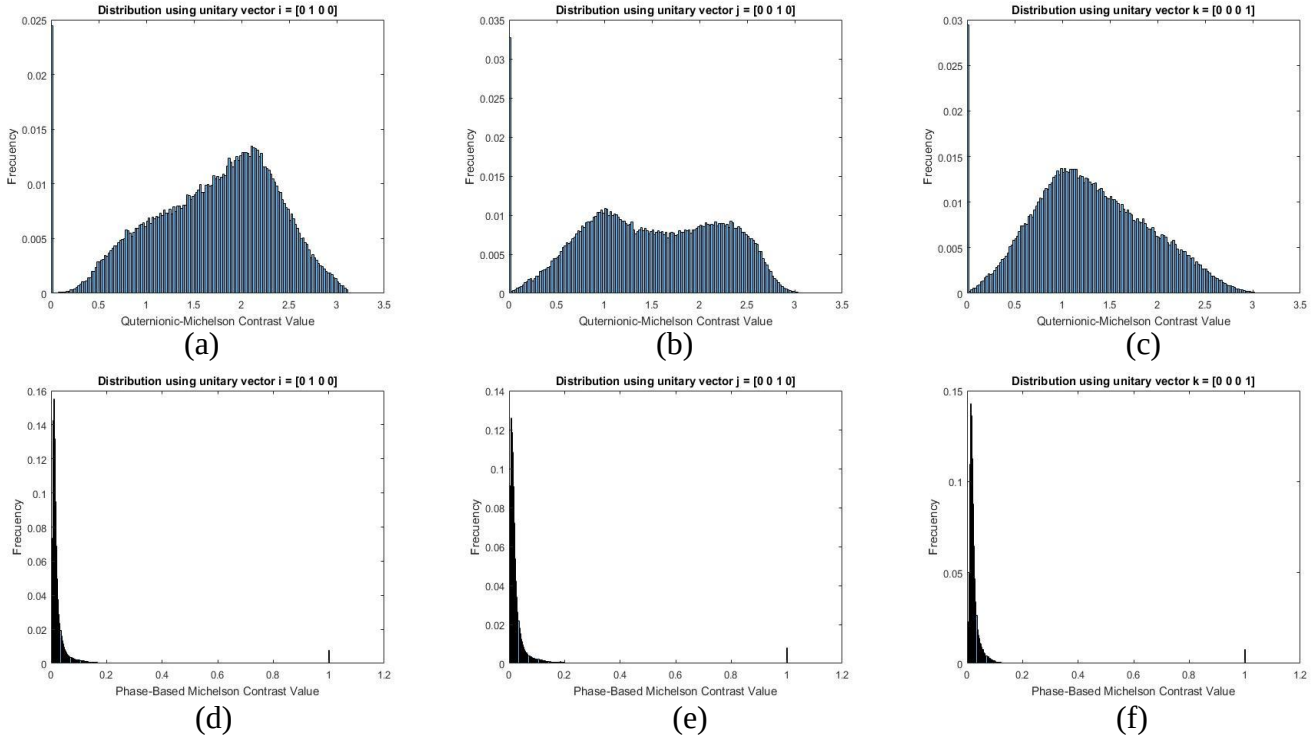


Fig. 3.2 *Distribuciones del contraste de Michelson para Img1 utilizando respectivamente los vectores unitarios i, j, k. (a)-(c): método basado en cuaterniones, (d)-(f): método basado en la fase de cuaterniones.*

En los tres primeros casos, la distribución resultante es similar a la mezcla de dos distribuciones Gaussianas[5]. Para el resto se observa cómo la mayoría de los valores fluctúan entre 0 y poco más que 0.1, casi 0.2, concentrándose más del 90% de la distribución en esa zona. El primer método, como ya se había pronosticado, brinda mayor información sobre el contenido de la imagen, sin embargo, ambos son igualmente precisos, la diferencia radica en la complejidad computacional de implementación y ejecución.

Los valores del contraste para los casos (a)-(c) a primera impresión es posible observar que oscilan entre 0 y un tanto más que 3 (teóricamente $[0, \pi]$) con una media próxima a 1.496. Para los casos (d)-(f) los valores de contraste se encuentran en el intervalo [0, 1] con media de 0.031.

Es en este momento que se ha obtenido el mapa de Michelson para las imágenes procesadas. Seguidamente se muestran las siete primeras filas y columnas de los mapas correspondientes a **Img2** utilizando ambos métodos y el vector unitario ***i***.

0	0	0	0	0	0	0
0	0.7124	0.8292	0.5159	0.4230	0.5630	0.4456
0	0.8612	0.8612	0.3958	0.4509	0.4612	0.3945
0	0.5526	0.5526	0.3958	0.5022	0.4066	0.4066
0	0.5283	0.4940	0.4385	0.4385	0.4624	0.4127
0	0.4724	0.4722	0.4700	0.4472	0.4308	0.5738
0	0.4200	0.4722	0.4671	0.4811	0.5414	0.5330

(a)

1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	0.0261	0.0268	0.0301	0.0312	0.0321	0.0321
1.0000	0.0148	0.0148	0.0240	0.0235	0.0177	0.0294
1.0000	0.0184	0.0184	0.0240	0.0291	0.0414	0.0414
1.0000	0.0256	0.0595	0.0579	0.0579	0.0535	0.0321
1.0000	0.0259	0.0621	0.0550	0.0521	0.0352	0.0269
1.0000	0.0380	0.0621	0.0425	0.0351	0.0398	0.0387

(b)

Fig. 3.3 *Fragmento de los valores del mapa de Michelson para Img2 utilizando el vector unitario i= [0 1 0 0]. (a): método basado en cuaterniones, (b): método basado en la fase de cuaterniones.*

[5] En estadística y probabilidad se llama **distribución normal**, **distribución de Gauss** o **distribución gaussiana** a una de las distribuciones de probabilidad de variable continua que con más frecuencia aparece aproximada en fenómenos reales. La gráfica de su función de densidad tiene una forma acampanada y es simétrica respecto a un determinado parámetro estadístico.

Importante mencionar que, aunque los ejemplos mostrados corresponden a una imagen en particular, los experimentos realizados con el resto de las imágenes arrojan resultados similares en cuanto a los valores de contraste y al comportamiento consecuente de los histogramas que los describen; la probabilidad de aparición de estos varía en dependencia del contenido de las imágenes propiamente. Las tablas 3.6 y 3.7 exponen algunos de estos resultados utilizando uno y otro método indistintamente.

Tabla 3.6 *Parámetros estadísticos del mapa de Michelson para Img1, Img2 e Img3 empleando el método basado en cuaterniones.*

	Img1			**Img2**			**Img3**		
	i	***j***	***k***	***i***	***j***	***k***	***i***	***j***	***k***
$\xi_{mín}$	0	0	0	0	0	0	0	0	0
$\xi_{máx}$	3.1344	3.1063	3.0965	3.1101	3.1204	3.1112	3.1077	3.1222	3.0842
Media	1.7038	1.4930	1.2907	0.7259	0.6095	2.5379	2.6412	0.6039	0.4856
Moda	1	1	1	1	1	1	1	1	1

En esta tabla se evidencia cómo los valores efectivamente oscilan entre 0 y π empleando este método, mientras la media varía ligeramente a la par de las imágenes. La siguiente tabla verifica este planteamiento para el otro método implementado.

Tabla 3.7 *Parámetros estadísticos del mapa de Michelson para Img1, Img2 e Img3 empleando el método basado en la fase de cuaterniones.*

	Img1			**Img2**			**Img3**		
	i	***j***	***k***	***i***	***j***	***k***	***i***	***j***	***k***
$\xi'_{mín}$	0	0	0	0	0.0003	0.00006	0.0007	0.0004	0.0011
$\xi'_{máx}$	1	1	1	1	1	1	1	1	1
Media	0.0300	0.0328	0.0314	0.0315	0.0143	0.0160	0.0253	0.0175	0.0279
Moda	0	0	0	0	0	0	0	0	0

De ambas tablas se puede deducir la tendencia a tonos claros de los mapas obtenidos a través del primer método y oscuros para el segundo si se hace una observación de los valores más frecuentes (moda). Inmediatamente se relacionan los mapas de Michelson obtenidos con distintas imágenes que certifican lo antes dicho; para más resultados consultar Anexo II.

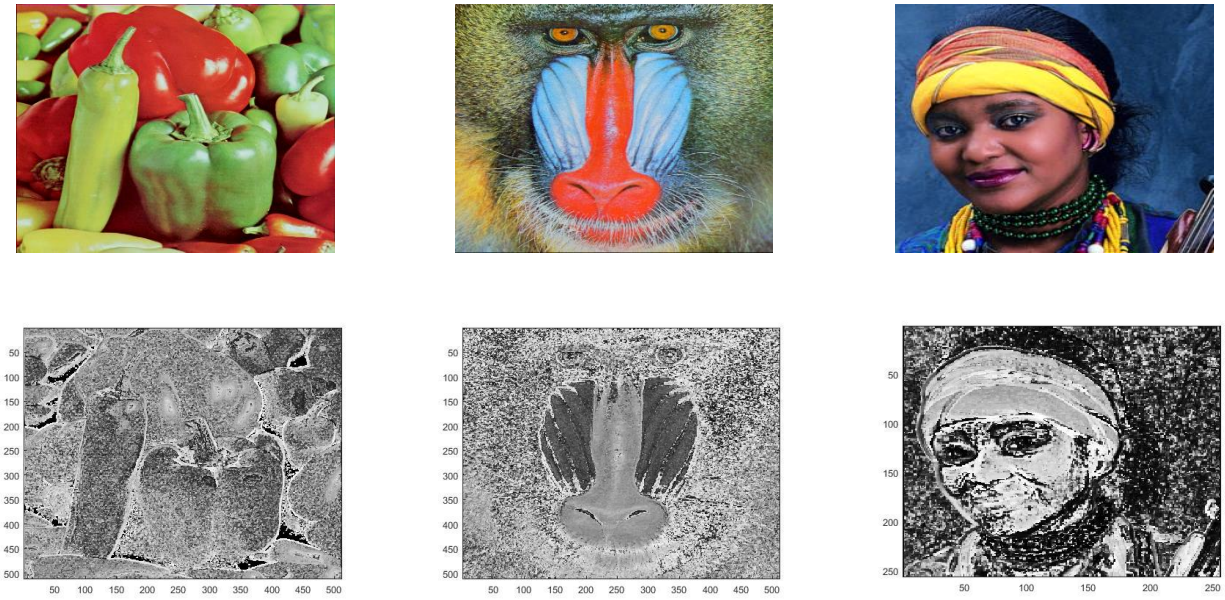


Fig. 3.4 *Mapa de Michelson (debajo) de distintas imágenes utilizando el método basado en cuaterniones con vector unitario i.*

Observar que el comportamiento en cada caso es similar utilizando el mismo método, lo cual se cumple igualmente en las imágenes que se mostrarán a continuación. Para el caso ya expuesto las imágenes se tornan entre blanco y varios tonos de gris, resultado que concuerda con los valores antes analizados.

En las imágenes siguientes sucede lo contrario. Estas se aprecian de valor negro con ligeras variaciones de gris dado que la diferencia de intensidad entre los píxeles es casi mínima en regiones contiguas, cosa que ocurre normalmente en imágenes de textura cuyo contenido suele ser bastante uniforme. Esto, por supuesto, está condicionado por el hecho de que en la fórmula que define este método esa propiedad influye proporcionalmente en el cálculo del contraste de Michelson; para otros resultados consultar Anexo III.

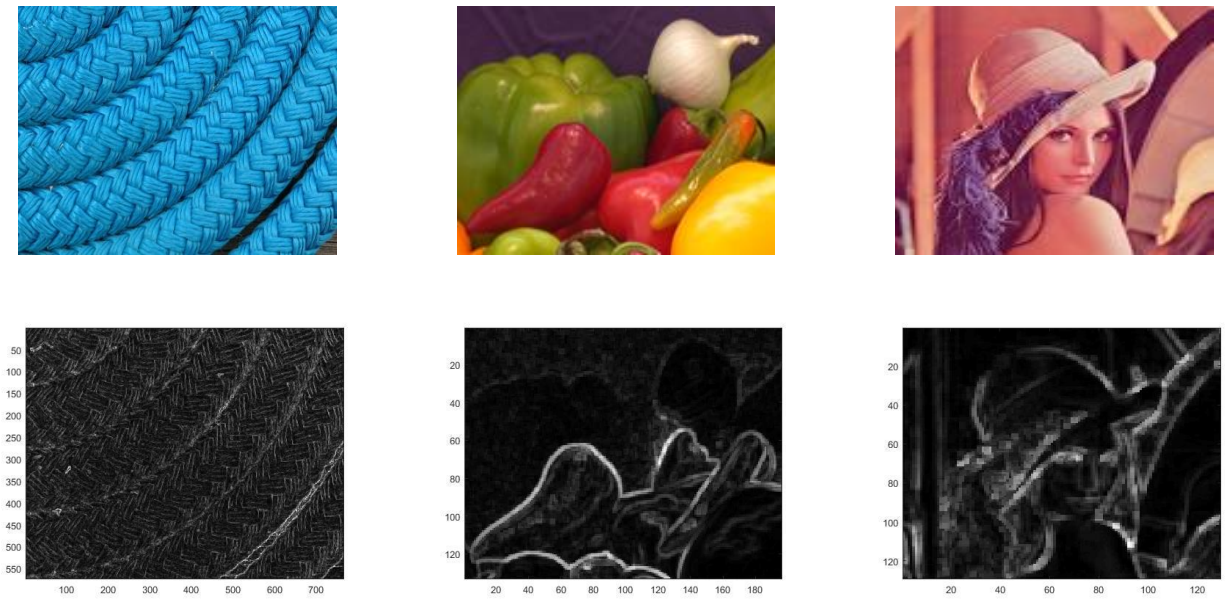
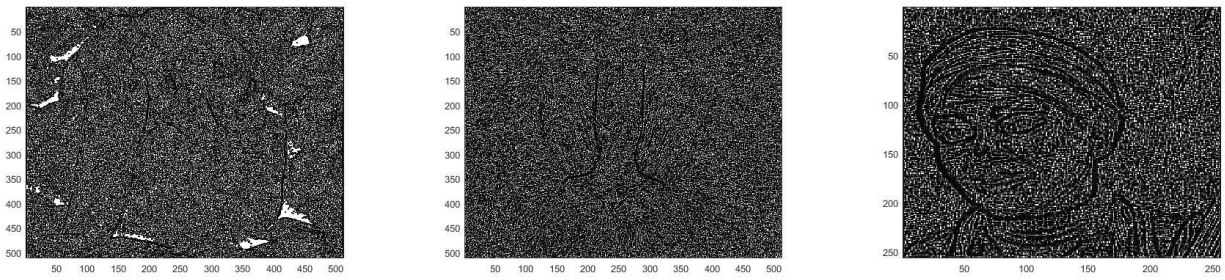


Fig. 3.5 *Mapa de Michelson (debajo) de distintas imágenes utilizando el método basado en la fase de cuaterniones con vector unitario i.*

3.3 Resultados de la codificación de patrones binarios locales.

El resultado de la codificación de patrones varía en dependencia de la función utilizada. Como se expuso en el capítulo anterior, *lbpgray_all* devuelve los patrones invariantes a la rotación y otros parámetros relacionados, brindando la posibilidad de utilizar estos valores para conformar un mapa de patrones binarios locales si se hace un barrido del mapa de Michelson antes obtenido.

De esta manera, experimentos realizados con las imágenes de prueba anteriores arrojaron los resultados que muestra la *Fig. 3.7*. Las imágenes a color originales a las cuales pertenece cada mapa LBP son las mismas que las mostradas en los mapas de Michelson del epígrafe anterior para cada método respectivamente.



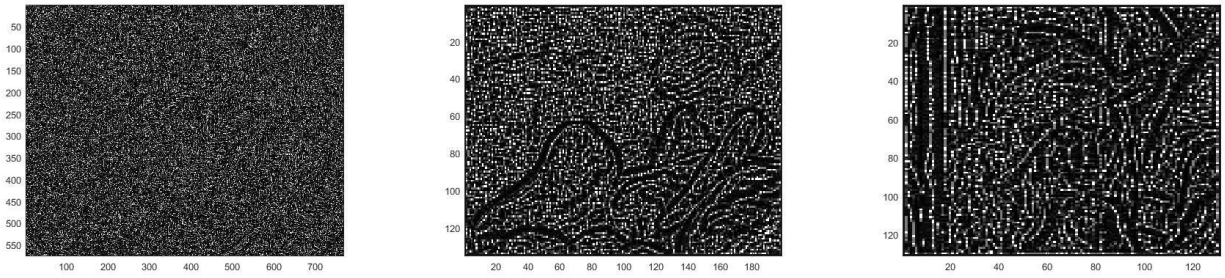


Fig. 3.7 *Mapas de patrones binarios locales utilizando el vector unitario i. (arriba): método basado cuaterniones, (abajo): método basado en la fase de cuaterniones.*

Estos mapas están conformados por píxeles cuyo valor decimal corresponde con el patrón binario invariante a la rotación de menor cantidad de transiciones espaciales de los 8 posibles para cada vecindad de 3x3 de la imagen, seleccionando el más pequeño en caso de existencia de más de uno con igual cantidad de variaciones y ubicando dicho valor en la posición del píxel central.

En aras de obtener una mejor idea sobre lo antes expuesto, la figura siguiente muestra las siete primeras filas y columnas del mapa de patrones binarios correspondientes a **Img3**.

255	255	255	255	255	255	255	5	25	17	17	17	17	17
255	7	31	31	3	31	3	19	255	255	255	127	31	23
255	0	25	1	5	127	31	17	31	31	15	119	39	7
255	9	127	255	55	127	3	17	7	15	15	7	255	127
255	5	0	9	23	5	1	17	15	3	1	1	7	55
255	23	95	31	15	3	31	17	255	127	127	55	63	255
255	1	255	63	1	1	3	17	15	27	3	31	15	15

(a) (b)

Fig. 3.6 *Fragmento de los valores del mapa de patrones binarios locales para Img3 utilizando el vector unitario i= [0 1 0 0]. (a): método basado en cuaterniones, (b): método basado en la fase de cuaterniones.*

Analizando estos resultados es sencillo deducir que para el caso del primer método los valores que corresponden a los bordes serán siempre 255 si se tiene en cuenta el principio de obtención de los LBP. Como antes había quedado demostrado el borde de los mapas de Michelson utilizando este método es 0, por consiguiente, al comparar este valor con sus píxeles circundantes la encuesta de que, si este es menor que cada valor que lo rodea siempre será

verdadera y, por tanto, el LBP resultante será 11111111, que representa el valor de intensidad 255; para el segundo caso esto obviamente no se cumplirá. A pesar de ello, para la manipulación posterior de los resultados estos bordes no fueron tomados en cuenta por la razón anteriormente comentada; aquí queda demostrado cómo estos pueden llegar a falsear los resultados reales. El resto de los valores dentro de la imagen depende exclusivamente del contenido de la misma.

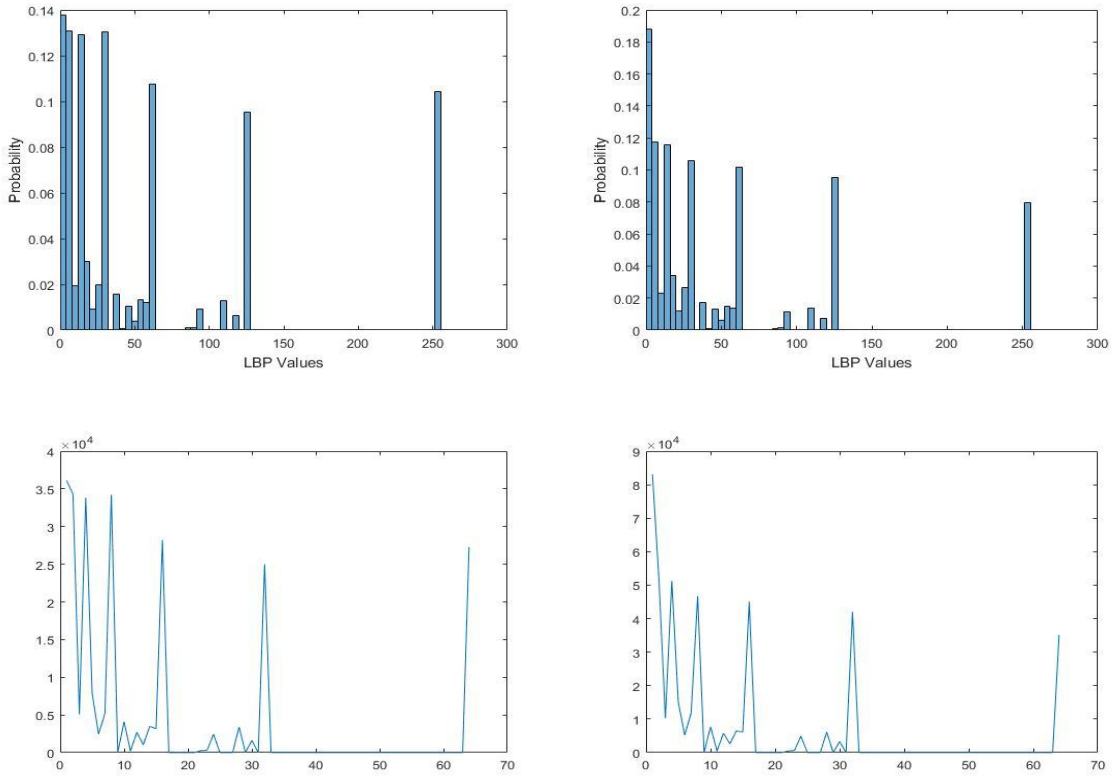


Fig. 3.7 *Representación del resultado de la codificación de patrones para Img1 e Img3 utilizando el método basado en la fase de cuaterniones, con vector unitario i. (izq.): Img1; (der.): Img3.*

En la figura se observa la distribución de los datos obtenidos del mapa de patrones binarios correspondiente a **Img1** e **Img3** utilizando el método basado en la fase de los cuaterniones, almacenados en 64 celdas. Las imágenes de arriba muestran la probabilidad de ocurrencia de los mismos, mientras las de abajo indican la cantidad de veces que aparecen estos resultados, quedando evidenciado que los valores más frecuentes y de mayor probabilidad son los más pequeños en estos y el resto de los casos examinados.

En otro orden, al utilizar la función de Matlab *extractLBPFeatures* los resultados son más acabados en cuanto al acercamiento a ciertas aplicaciones concretas se refiere, sin embargo, se obvian en el proceso algunos elementos que es posible manejar a partir de la función antes analizada y que son igualmente útiles para otros fines; la tabla que sigue muestra algunos de estos resultados. En este caso fue utilizado el parámetro *Upright* en modo *false* de forma que están recogidos solo rasgos derivados de patrones binarios invariantes a la rotación.

Tabla 3.8 *Vectores fila de rasgos LBP para los mapas de Michelson de Img1, Img2 e Im3.*

	Rasgos LBP									
Img1	0.0268	0.1383	0.0304	0.2834	0.1790	0.4056	0.1853	0.2160	0.7423	0.2651
Img2	0.0368	0.2422	0.0737	0.3083	0.2740	0.3777	0.2670	0.2625	0.5213	0.4581
Img3	0.0371	0.2061	0.0619	0.2992	0.1988	0.3691	0.2544	0.2423	0.6352	0.4001

Si bien los rasgos LBP resultantes de esta función codifican información de textura local útil en tareas de clasificación, detección y reconocimiento empleando imágenes en escala de grises, la utilización de imágenes representadas por el contraste de Michelson brinda un punto de vista diferente que puede cambiar la perspectiva y diversificar el espectro de aplicaciones desarrolladas hasta la fecha.

3.4 Validación de los resultados obtenidos.

En función de validar los resultados se hicieron varias pruebas experimentales sobre la obtención de patrones binarios locales en vecindades aleatorias de algunos de los mapas de Michelson de las imágenes procesadas. Las pruebas consistieron en conseguir, de forma manual, los LBP asociados a dichas vecindades teniendo en cuenta la descripción teórica que define su principio de obtención, para luego comprobar si el valor decimal correspondiente al píxel central respectivo en los mapas LBP que se concibieron de manera computacional coincide con el valor del patrón invariante a la rotación de menor tamaño y menos transiciones espaciales calculado en cada caso. La siguiente tabla representa las vecindades tomadas para el estudio:

Tabla 3.9 *Valores de contraste de vecindades aleatorias de los mapas de Michelson de las imágenes de muestra.*

Img1 (filas: 281-283, cols: 503-505)			**Img2** (filas: 304-306, cols: 129-131)			**Img3** (filas: 425-427, cols: 598-600)		
0.0236	0.0236	0.0187	0.0125	0.0121	0.0121	2.8323	2.7975	2.8071
0.0236	0.0236	0.0210	0.0202	0.0192	0.0108	2.8323	2.7233	2.8643
0.0144	0.0144	0.0163	0.0137	0.0192	0.0108	2.6866	2.5968	2.9311

En los dos primeros casos se muestran las vecindades extraídas de los mapas de Michelson obtenidos empleando el método basado en la fase con distintos vectores unitarios, mientras que el tercero corresponde al mapa obtenido por el método basado en cuaterniones. Inicialmente se halla un patrón comenzando la encuesta por un píxel determinado y el resto será el resultado de rotar los dígitos binarios del patrón obtenido como explica el epígrafe 1.5.3. En cada caso, comenzando por el píxel de la posición (1,1), se obtuvo:

Tabla 3.10 *LBP obtenido en cada vecindad comenzando la encuesta por el píxel de la posición (1,1).*

Píxel encuestado	**Img1**	**Img2**	**Img3**
(1,1)	0.0236 < 0.0236 …? 1	0.0192 < 0.0125 …? 0	2.7233 < 2.8323 …? 1
(1,2)	0.0236 < 0.0236 …? 1	0.0192 < 0.0121 …? 0	2.7233 < 2.7975 …? 1
(1,3)	0.0236 < 0.0187 …? 0	0.0192 < 0.0121 …? 0	2.7233 < 2.8071 …? 1
(2,3)	0.0236 < 0.0210 …? 0	0.0192 < 0.0108 …? 0	2.7233 < 2.8643 …? 1
(3,3)	0.0236 < 0.0163 …? 0	0.0192 < 0.0108 …? 0	2.7233 < 2.9311 …? 1
(3,2)	0.0236 < 0.0144 …? 0	0.0192 < 0.0192 …? 1	2.7233 < 2.5968 …? 0
(3,1)	0.0236 < 0.0144 …? 0	0.0192 < 0.0137 …? 0	2.7233 < 2.6866 …? 0
(2,1)	0.0236 < 0.0236 …? 1	0.0192 < 0.0202 …? 1	2.7233 < 2.8323 …? 1
LBP	$11000001_2 = 193_d$	$00000101_2 = 5_d$	$11111001_2 = 249_d$

Luego, los demás patrones de la vecindad quedarían como se indica seguidamente:

Tabla 3.11 *Patrones binarios locales de las vecindades de muestra.*

Img1			Img2			Img3		
En sistema de numeración binario								
11000001	11100000	01110000	00000101	10000010	01000001	11111001	11111100	01111110
10000011	g_0	00111000	00001010	g_0	10100000	11110011	g_0	00111111
00000111	00001110	00011100	00010100	00101000	01010000	11100111	11001111	10011111
En sistema de numeración decimal								
193	224	112	5	130	65	249	252	126
131	g_0	56	10	g_0	160	243	g_0	63
7	14	28	20	40	80	231	207	159

Por consiguiente, los valores que deben haberse obtenido en los mapas LBP son 7, 5 y 63 respectivamente. Para comprobarlo se ha de localizar en estos el valor decimal del LBP ubicado en el píxel central de cada vecindad probada. Las siguientes capturas muestran cuáles fueron los resultados computacionales previos:

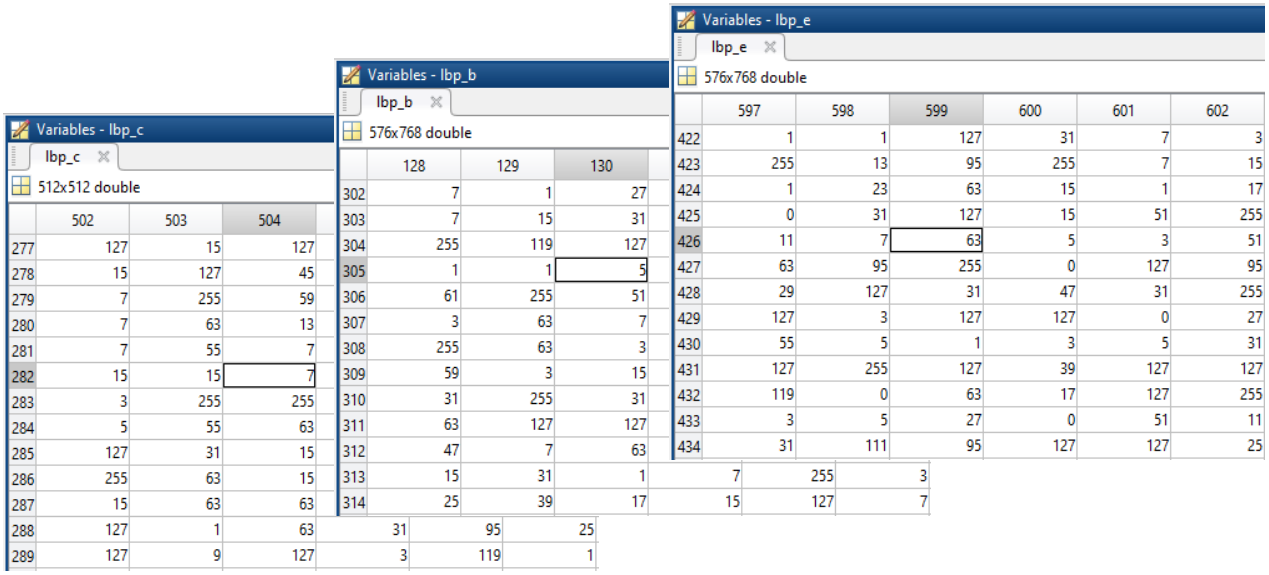


Fig. 3.8 *Capturas de pantalla de fragmentos de los mapas LBP correspondientes a (de izq. a der.) Img1, Img2 e Img3.*

Como es posible apreciar los valores esperados, que debían encontrarse en la posición (282,504) para **Img1**, en la (305,130) para **Img2** y para **Img3** en la (426,599) de sus respectivos mapas, coinciden con los calculados manualmente. Por todo ello y debido a pruebas con otras imágenes y vecindades se puede alegar que la obtención de patrones binarios locales a partir de la función *lbpgray_all* utilizada resulta plenamente viable.

3.5 Tiempos de ejecución del algoritmo.

Al inicio de este trabajo y en el transcurso del mismo se ha indicado la superioridad de este descriptor local respecto a otros teniendo en cuenta su facilidad de implementación y el ahorro que representa en tiempo de cómputo. Sin embargo, aunque en este informe no se pretende hacer tal comparación, se exponen los tiempos de ejecución del algoritmo teniendo en cuenta los dos métodos propuestos y con varias imágenes de prueba en pos de arribar a conclusiones sobre ello.

Tabla 3.12 *Tiempos de ejecución del algoritmo QMD (en segundos) para las tres imágenes de prueba.*

QMC	t1	t2	t3	t4	PMC	t1	t2	t3	t4
Img1	240.6037	756.9957	455.5389	1476.2566	**Img1**	237.8466	693.3549	435.4622	1343.1416
Img2	440.6975	1239.9031	792.8634	2428.0797	**Img2**	385.3376	1151.7404	734.9479	2275.4215
Img3	405.2053	1216.7134	749.8216	2324.3111	**Img3**	382.0919	1145.4601	725.8959	2169.9103

QMC: método basado en cuaterniones, **PMC:** método basado en la fase de cuaterniones, **t1:** tiempo promedio de obtención de 1 mapa de Michelson, **t2:** tiempo promedio de obtención de todos los mapas de Michelson (3 para cada método considerando los 3 vectores unitarios), **t3:** tiempo promedio de obtención de 1 mapa de Michelson y 1 mapa LBP. **t4:** tiempo promedio de obtención de todos los mapas de Michelson y todos los mapas LBP (igualmente teniendo en cuenta los 3 vectores unitarios en cada caso y ambos métodos).

Es conveniente comentar aquí que estos tiempos de ejecución dependen de varios factores independientemente de la complejidad computacional de cada método; entre ellos las propiedades y rendimiento de la PC en que se ejecuten, el contenido de las imágenes y las dimensiones de estas. Resulta trivial deducir que a mayor tamaño de imagen corresponde mayor tiempo de ejecución; basta pensar que **Img1** posee 262144 píxeles, mientras que **Img2**

e **Img3** están compuestas por 442368, lo cual hace la exploración más costosa para estas últimas en cuanto a tiempo de procesamiento se refiere.

Los resultados arrojados exhiben, además, el hecho de que a medida que las dimensiones de las imágenes procesadas aumenta, la diferencia de tiempo de cómputo por el uso de uno u otro método se acrecienta paralelamente, quedando evidenciado en todos los casos que el método basado en la fase es más ligero, sin embargo, los dos devuelven resultados de utilidad para aplicaciones de clasificación y análisis de textura.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

Una vez consumado el proceso investigativo y luego de analizados los resultados alcanzados se ha podido arribar a las siguientes conclusiones generales:

1. De la revisión bibliográfica relativa a los tres primeros objetivos específicos se desglosan las cuestiones que siguen:
   - La representación de imágenes a color mediante cuaterniones brinda una elegante y a la vez factible solución al problema de la concatenación de la información cromática de los planos del modelo RGB en el procesamiento digital de imágenes.
   - El descriptor local *Quaternion-Michelson* muestra un enfoque diferente en torno a la evaluación de la textura en imágenes, básicamente por la representación de los píxeles como cuaterniones y la utilización del contraste de Michelson para la obtención de LBPs extendidos a este dominio.
   - El uso del contraste de Michelson en el tratamiento del contenido de imágenes a color ofrece una simplificada y estable herramienta para la manipulación de rasgos locales en las mismas.
   - Los patrones binarios locales, tanto en imágenes en escala de grises como a color, constituyen un poderoso operador de texturas de utilidad en disímiles aplicaciones modernas.
2. El cumplimiento del objetivo específico 4 relativo a la programación en Matlab de funciones para el tratamiento de imágenes a color, demostró la efectividad y viabilidad del algoritmo implementado.
3. Los dos métodos descritos por el algoritmo son igualmente eficaces, sin embargo, el método basado en la fase de los cuaterniones puede ser considerado como una representación de nivel superior de las imágenes a color.
4. El costo computacional del algoritmo implementado es aceptable teniendo en cuenta las funciones que ejecuta y el extenso proceso de barrido de la imagen que efectúa.

RECOMENDACIONES

1. Potenciar el uso de patrones binarios en el procesamiento digital de imágenes a color en combinación con otros elementos de representación y caracterización de las mismas.
2. Para trabajos futuros, aumentar la cantidad de imágenes procesadas en función de obtener resultados estadísitcamente más rigurosos.
3. Considerar la obtención de patrones binarios locales en imágenes a color utilizando mayor cantidad de puntos por vecindad de referencia y mayor radio respecto al píxel central (P = 16, R = 2 y P = 24, R = 3).
4. Validar los resultados relativos a la obtención de los mapas de Michelson creando previamente imágenes con valores de intensidad en términos de contraste fijos, en aras de comparar estos con los que se obtienen y verificar la efectividad del método en este punto.
5. Realizar pruebas y experimentos teniendo en cuenta indistintamente el empleo de uno u otro método del algoritmo QMD y varios vectores unitarios, de manera que sea posible derivar resultados múltiples para una misma aplicación.
6. Extender el uso de los mapas de patrones binarios locales obtenidos utilizando el contraste de Michelson extraído mediante cuaterniones para experimentos de clasificación de texturas con diferentes tipos de clasificadores.

REFERENCIAS BIBLIOGRÁFICAS

[1] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[2] J. Vogel, A. Schwaninger, C. Wallraven, and H. H. Bülthoff, "Categorization of natural scenes: local vs. global information", *Proceedings of the Symposiumon Applied Perception in Graphics and Visualization (APGV06), ACM Press, New York*, pp. 33–40, 2006.

[3] Y. Liu, D. Zhang, and et al. "A survey of content-based image retrieval with high level semantics", *Pattern Recognition*, pp. 262-282, 2007.

[4] H. Tamura, S. Mori, and T. Yamawaki. "Texture features corresponding to visual perception", *IEEE Trans.on Sys. Man and Cyb*. SMC-8(6), 1978.

[5] R. Lan and Y. Zhou, "Quaternion-Michelson Descriptor for color image classification", *IEEE transactions on image processing,* vol.25, no.11 pp. 5281-5292, Nov. 2016.

[6] W. R. Hamilton, "On a new Species of Imaginary Quantities Conected with a Theory of Quaternions", *Proceedings of the Royal Irish Academy,* vol. 2, pp. 424-434, Dec. 1844.

[7] V. R. Bouza, "Sobre los cuaterniones, álgebras de Lie y matrices de Pauli. Teoría básica y aplicaciones físicas", Doble grado Física-Matemáticas, *Métodos matemáticos I*, pp. 5-10, 2012.

[8] A. Favieri, "Introducción a los cuaterniones", *Matemáticas aplicadas a la aeronáutica. Universidad Tecn. Nacional,* edUTecNe, pp. 21-30, nov. 2008.

[9] J. Weeks, R. Lehoucq, and J.-P. Uzan, "Detecting topology in a nearly flat spherical universe", Classical Quantum Gravity, vol. 20, no. 8, p. 1529, 2003.

[10] J. Vogel. "Semantic Scene Modeling and Retrieval PhD Thesis", *PhD thesis*, Swiss Federal Institute of technology Zurich. Zurich Germany, 2004.

[11] B. Chen, H. Shu, H. Zhang, G. Chen, and L. Luo, "Color image analysis by quaternion Zernike moments", *Proc. IEEE Int. Conf. Pattern Recognit. (ICPR),* pp. 625–628, Aug. 2010.

[12] B. J. Chen *et al.*, "Quaternion Zernike moments and their invariants for color image analysis and object recognition", *Signal Process.*, vol. 92, no. 2, pp. 308–318, 2012.

[13] L. Q. Guo and M. Zhu, "Quaternion Fourier–mellin moments for color images", *Pattern Recognit*., vol. 44, no. 2, pp. 187–195, Feb. 2011.

[14] B. Chen, H. Shu, G. Coatrieux, G. Chen, X. Sun, and J. L. Coatrieux, "Color image analysis by quaternion-type moments", *J. Math. Imag. Vis.*, vol. 51, no. 1, pp. 124–144, 2015.

[15] J. Mennesson, C. Saint-Jean, and L. Mascarilla, "New geometric Fourier descriptors for color image recognition", *Proc. 17th IEEE Int. Conf. Image Process. (ICIP)*, pp. 2685–2688, Sep. 2010.

[16] Z. Yang and S.-I. Kamata, "Hypercomplex polar Fourier analysis for color image", *Proc. 18th IEEE Int. Conf. Image Process. (ICIP)*, pp. 2117–2120, Sep. 2011.

[17] L. Guo, M. Dai, and M. Zhu, "Quaternion moment and its invariants for color object classification", *Inf. Sci.*, vol. 273, pp. 132–143, Jul. 2014.

[18] W. Pratt, "Digital Image Processing: PIKS Inside", 3rd ed. *John Wiley & Sons*, 2001

[19] A. A. Goshtasby, "Image registration principles, tools and methods", *Springer*, 2012.

[20] T. Acharya and Ajoy K. Ray, "Image processing. Principles and applications", John Wiley & Sons, 2005.

[21] S. Theodoridis and K. Koutroumbas, "Pattern recognition*"*, 2nd ed. *Elsevier*, 2003.

[22] R. Benítez, G. Escudero, and S. Kanaan, "Inteligencia artificial avanzada", *UOC*. 2011.

[23] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis*., vol. 60, no. 2, pp. 91–110, 2004.

[24] J.-M. Morel and G. Yu, "ASIFT: A new framework for fully affine invariant image comparison", *SIAM J. Imag. Sci*., vol. 2, no. 2, pp. 438–469, 2009.

[25] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features", *Proc. Eur. Conf. Comput. Vis. (ECCV*), pp. 404–417, May 2006.

[26] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection”, *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, pp. 886–893, Jun. 2005.

[27] C.-Y. Su and J.-F. Yang, “Histogram of gradient phases: A new local descriptor for face recognition”, *IET Comput. Vis.*, vol. 8, no. 6, pp. 556–567, 2014.

[28] K. Hara, K. Inoue, and K. Urahama, “Gradient operators for feature extraction from omnidirectional panoramic images”, *Pattern Recognit. Lett.*, vol. 54, pp. 89–96, Mar. 2015.

[29] D. K. Park, Y. S. Jeon, and C. S. Won, “Efficient use of local edge histogram descriptor”, *Proc. ACM Workshops Multimedia*, Nov. 2000, pp. 51–54.

[30] C. S. Won, D. K. Park, and S.-J. Park, “Efficient use of MPEG-7 edge histogram descriptor”, *Etri J.*, vol. 24, no. 1, pp. 23–30, Feb. 2002.

[31] M. Subrahmanyam, R. Maheshwari, and R. Balasubramanian, “Local maximum edge binary patterns: A new descriptor for image retrieval and object tracking”, *Signal Process.*, vol. 92, no. 6, pp. 1467–1479, Jun. 2012.

[32] A. Satpathy, X. Jiang, and H.-L. Eng, “LBP based edge-texture features for object recognition”, *IEEE Trans. Image Process.*, vol. 23, no. 5, pp. 1953–1964, May 2014.

[33] A. K. Jain, *Fundamentals of Digital Signal Processing*. Upper Saddle River, NJ, USA: Prentice-Hall, 1989.

[34] J. Chen, S. Shan, G. Zhao, X. Chen, W. Gao, and M. Pietikainen, “A robust descriptor based on Weber’s Law,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* (CVPR), pp. 1–7, Jun. 2008.

[35] J. Chen et al., “WLD: A robust local image descriptor”, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1705–1720, Sep. 2010.

[36] Y. Jiang, B. Wang, Y. Zhou, W. Li, and Q. Liao, “Patterns of Weber magnitude and orientation for uncontrolled face representation and recognition”, *Neurocomputing*, vol. 165, pp. 190–201, Oct. 2015.

[37] R. Lan, Y. Zhou, and Y. Tang, "Quaternionic Weber local descriptor of color images", *IEEE Trans. Circuits Syst. Video Technol.*, to be published, doi: 10.1109/TCSVT.2015.2492839.

[38] Legge and Kersten, "Contrast discrimination in peripheral vision", *J. Opt. Soc. Am.* pp.1594-1598, 1987.

[39] A. A. Michelson, "Studies in Optics", *North Chelmsford, MA: Courier Corporation*, 1995.

[40] E. Peli, "In search of a contrast metric: Matching the perceived contrast of Gabor patches at different phases and bandwidths", *Vis. Res.*, vol. 37, no. 23, pp. 3217–3224, Dec. 1997.

[41] E. J. Ramón Balmaseda, "Transformaciones basadas en el algoritmo Local Binary Pattern de imágenes capturadas con la *Kinect* para clasificación facial", pp. 58-64, Nov. 2011.

[42] T. Ojala, M. Pietikäinen, and D. Harwood. "A comparative study of texture measures with classification based on featured distributions", *Pattern Recognition,* pp. 51-59, 1996.

[43] "McGill calibrated colour image database", disponible en: http://tabby.vision.mcgill.ca/html/browsedownload.html, consultado: Abr. 20, 2018.

[44] Repositorio de imágenes asignatura PDI, Universidad Central "Marta Abreu" de Las Villas, disponible en ftp:\\10.12.1.67\docs\FIE\Carreras\Telecomunicaciones y Electrónica\PDI\ PDI_Tele_Imágenes_varias\.

[45] R. Lan, Y. Zhou, Y. Y. Tang, and C. L. P. Chen, "Person reidentification using quaternionic local binary pattern", *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2014, pp. 1–6.

ANEXOS

Anexo I Código fuente de los programas de computación implementados.

***Script QMD* que contiene el algoritmo estructurado por pasos.**

```
%% ==========Entrada de la imagen a color==========
i = imread('nombre_ImgRGB.formato_ImgRGB');
% Conversión de la imagen a clase 'double' normalizada, de manera que los
% valores de intensidad de los píxeles oscilarán entre [0,1].
imD = im2double(i);

%% ==========Representación cuaterniónica==========
% Creación de una matriz de ceros de iguales dimensiones que la imagen
% original. Se define la variable 'm_ceros': matriz de ceros.
m_ceros = zeros(size(imD(:,:,1)));
% Concatenación de la matriz de ceros con las matrices de los planos RGB
% de la imagen original, de forma que la extracción de los valores de
% intensidad de cada píxel devolverá el equivalente a un cuaternión puro.
% Se define la variable 'm_c': matriz cuaternión.
m_c = cat(3,m_ceros,imD);

%% ==========Extracción del contraste de Michelson==========
% Se procede a extraer el contraste de Michelson de los cuaterniones que
% representan la imagen a partir de la función 'michelson' que devuelve,
% para cada vecindad de 3x3, el valor del contraste extraído en el píxel
% central de la misma.
% Al mismo tiempo, se utiliza la función 'nlqfilter' que barre toda la
% imagen en una ventana deslizante de iguales dimensiones.
% Se definen las variables 'b','c','d' que usan respectivamente los vecto-
% res unitarios i,j,k en la CTQ(Traslación de Clifford), 'p' determina que
% el método a emplear es respecto a la fase de los píxeles representados
% por cuaterniones.
b=nlqfilter(m_c,[3 3],'michelson','i','p'); %Vector i=[0 1 0 0]
c=nlqfilter(m_c,[3 3],'michelson','j','p'); %Vector j=[0 0 1 0]
d=nlqfilter(m_c,[3 3],'michelson','k','p'); %Vector k=[0 0 0 1]

% Seguidamente se procede a extraer el contraste en torno a la información
% de color contenida en cada plano. Para ello, se utiliza como parámetro
% de la función 'michelson' la cadena 'q', que representa el método basado
% en cuaterniones. Para ello se definen, además, las variables 'e','f','g'.
e=nlqfilter(m_c,[3 3],'michelson','i','q'); %Vector i=[0 1 0 0]
f=nlqfilter(m_c,[3 3],'michelson','j','q'); %Vector j=[0 0 1 0]
g=nlqfilter(m_c,[3 3],'michelson','k','q'); %Vector k=[0 0 0 1]

% Seguidamente se muestran los mapas de Michelson obtenidos. PMC representa
% el primer método, QMC el segundo.
```

```
figure ('name','Michelson Map, PMC, i'); imagesc(b(:,:,1));
figure ('name','Michelson Map, PMC, j'); imagesc(c(:,:,1));
figure ('name','Michelson Map, PMC, k'); imagesc(d(:,:,1));
figure ('name','Michelson Map, QMC, i'); imagesc(e(:,:,1));
figure ('name','Michelson Map, QMC, j'); imagesc(f(:,:,1));
figure ('name','Michelson Map, QMC, k'); imagesc(g(:,:,1));

%% ==========Histogramas de los mapas de Michelson==========
% En este paso se obtiene una distribución de los valores de los contrastes
% de Michelson obtenido en cada caso.

              % Método basado en la fase de los cuaterniones:
figure('name','Distribution of Phase-Based Michelson Contrast (unitary
vector i)');
h1=histogram(b,'Normalization','probability');
xlabel('Phase-Based Michelson Contrast Value'); ylabel('Frecuency');
title('Distribution using unitary vector i = [0 1 0 0]');

figure('name','Distribution of Phase-Based Michelson Contrast (unitary
vector j)');
h2=histogram(c,'Normalization','probability');
xlabel('Phase-Based Michelson Contrast Value'); ylabel('Frecuency');
title('Distribution using unitary vector j = [0 0 1 0]');

figure('name','Distribution of Phase-Based Michelson Contrast (unitary
vector k)');
h3=histogram(d,'Normalization','probability');
xlabel('Phase-Based Michelson Contrast Value'); ylabel('Frecuency');
title('Distribution using unitary vector k = [0 0 0 1]');

              % Método basado en los cuaterniones:
figure('name','Distribution of Quternionic-Michelson Contrast (unitary
vector i)');
h4=histogram(e,'Normalization','probability');
xlabel('Quternionic-Michelson Contrast Value'); ylabel('Frecuency');
title('Distribution using unitary vector i = [0 1 0 0]');

figure('name','Distribution of Quternionic-Michelson Contrast (unitary
vector j)');
h5=histogram(f,'Normalization','probability');
xlabel('Quternionic-Michelson Contrast Value'); ylabel('Frecuency');
title('Distribution using unitary vector j = [0 0 1 0]');

figure('name','Distribution of Quternionic-Michelson Contrast (unitary
vector k)');
h6=histogram(g,'Normalization','probability');
xlabel('Quternionic-Michelson Contrast Value'); ylabel('Frecuency');
title('Distribution using unitary vector k = [0 0 0 1]');
```

```
%% ==========Obtención de los mapas de Patrones Binarios Locales==========
% Se declaran las variables N y R que representan la cantidad de puntos y
% el radio empleado en la obtención de los LBP de simetría circular.
N=8; R=1;
          % Método basado en la fase de los cuaterniones:
lbp_b=nlfilter(b(:,:,1),[3 3], 'lbpgray_all',N,R);
figure ('name','lbp_b'); imagesc(lbp_b(:,:,1)); colormap gray;
figure ('name','lbp_b_color'); imagesc(lbp_b(:,:,1));

lbp_c=nlfilter(c(:,:,1),[3 3], 'lbpgray_all',N,R);
figure ('name','lbp_c'); imagesc(lbp_c(:,:,1)); colormap gray;
figure ('name','lbp_c_color'); imagesc(lbp_c(:,:,1));

lbp_d=nlfilter(d(:,:,1),[3 3], 'lbpgray_all',N,R);
figure ('name','lbp_d'); imagesc(lbp_d(:,:,1)); colormap gray;
figure ('name','lbp_d_color'); imagesc(lbp_d(:,:,1));

         % Método basado en los cuaterniones:
lbp_e=nlfilter(e(:,:,1),[3 3], 'lbpgray_all',N,R);
figure ('name','lbp_e'); imagesc(lbp_e(:,:,1)); colormap gray;
figure ('name','lbp_e_color'); imagesc(lbp_e(:,:,1));

lbp_f=nlfilter(f(:,:,1),[3 3], 'lbpgray_all',N,R);
figure ('name','lbp_f'); imagesc(lbp_f(:,:,1)); colormap gray;
figure ('name','lbp_f_color'); imagesc(lbp_f(:,:,1));

lbp_g=nlfilter(g(:,:,1),[3 3], 'lbpgray_all',N,R);
figure ('name','lbp_g'); imagesc(lbp_g(:,:,1)); colormap gray;
figure ('name','lbp_g_color'); imagesc(lbp_g(:,:,1));

%%  ==========Histogramas de los mapas LBP==========
          % Método basado en la fase de los cuaterniones:
figure('name','hlbp_b');
hlbp_b = hist(lbp_b(:)); plot(hlbp_b);

figure('name','hlbp_c');
hlbp_c = hist(lbp_c(:)); plot(hlbp_c);

figure('name','hlbp_d');
hlbp_d = hist(lbp_d(:)); plot(hlbp_d);
          % Método basado en los cuaterniones:
figure('name','hlbp_e');
hlbp_e = hist(lbp_e(:)); plot(hlbp_e);

figure('name','hlbp_f');
hlbp_f = hist(lbp_f(:)); plot(hlbp_f);
```

```
figure('name','hlbp_g');
hlbp_g = hist(lbp_g(:)); plot(hlbp_g);
```

Código de la función *michelson* empleada para la extracción del contraste.

```
function [m] = michelson (neigh, uquat, met)
% MICHELSON extrae el contraste de Michelson de una vecindad de múltiples
% dimensiones en el dominio de los cuaterniones.
% M = MICHELSON ([M N], 'UQUAT', 'MET') aplica el descriptor "Quaternion-
% Michelson" utilizando el método basado en cuaterniones y el método basado
% en la fase de los mismos a una vecindad de MxNxP, donde P debe ser 4 para
% garantizar las operaciones mediante cuaterniones.
% MICHELSON consecuentemente debe aceptar un bloque de MxNx4, una cadena
% de texto que representa el vector unitario utilizado 'i', 'j' o 'k' y
% otra que identifica el método usado 'p' o 'q' como entrada, y retorna un
% número real que brinda una medida del contraste en la vecindad.

% Evaluación de los argumentos de entrada.
if nargin<2
    error('Not enough input parameters')
end
if ischar(uquat)==0
     error('uquat has to be a string (i/j/k)')
end
if ischar(met==0)
    error('met has to be a string (p/q)')
end

% Creación de una matriz cuyas columnas representan el vector cuaternión
% que corresponde a cada píxel ('matriz cuaternión 2D').
s = size(neigh);
nrows = s(1);
vneigh = [];
for i = 1 : nrows
   D(:,:) = neigh(i,:,:);
   vneigh = [vneigh D'];
end

% Cálculo de la traslación de Clifford y obtención de la fase de cada
% vector cuaternión.
t = size (vneigh);
ncols = t(2);
vneigh2 = [];
vneigh3 = [];
for j = 1:ncols
% Traslación de Clifford.
     if uquat == 'i'
     ctq = clifford(vneigh(:,j)', [0 1 0 0], 'r');
```

```
     elseif uquat == 'j'
     ctq = clifford(vneigh(:,j)', [0 0 1 0], 'r');
     elseif uquat == 'k'
     ctq = clifford(vneigh(:,j)', [0 0 0 1], 'r');
     end
     vneigh2 = [vneigh2 ctq'];
% Obtención de la fase.
    md = sqrt (vneigh2(2,j)^2 + vneigh2(3,j)^2 + vneigh2(4,j)^2);
    f = atan2(md,vneigh2(1,j));
    vneigh3 = [vneigh3 f];
end

% Declaración de variables para la extracción del contraste.
         % Método basado en la fase de los cuaterniones:
fo = sort(vneigh3);

         % Método basado en los cuaterniones:
rt1= vneigh(2,(vneigh3==fo(1)));
rtn= vneigh(2,vneigh3==fo(end));

gt1= vneigh(3,vneigh3==fo(1));
gtn= vneigh(3,vneigh3==fo(end));

bt1= vneigh(4,vneigh3==fo(1));
btn= vneigh(4,vneigh3==fo(end));

dr=rtn(1)-rt1(1); tr=rtn(1)+rt1(1);
dg=gtn(1)-gt1(1); tg=gtn(1)+gt1(1);
db=btn(1)-bt1(1); tb=btn(1)+bt1(1);

switch met
    case 'p'
    % Extracción del contraste por el método basado en la fase:
    m = (fo(1,end) - fo(1,end-(end-1))) / (fo(1,end) + fo(1,end-(end-1)));
    case 'q'
    % Extracción del contraste por el método basado en los cuaterniones:
    qmy= sqrt( (dg*tb-db*tg)^2 + (db*tr-dr*tb)^2 + (dr*tg-dg*tr)^2 );
    qmx= dr*tr + dg*tg + db*tb;
    m = atan2(qmy,qmx);
end
end
```

Código de la función *clifford* de uso en la ejecución de la traslación de Clifford (subfunción de *michelson*).

```
function [ctq] = clifford(quat,uquat,par)
% CLIFFORD ejecuta la traslación de Clifford de un cuaternión multiplicando
% el cuaternión 'quat' por un cuaternión unitario (expresado como un vector
```

```
% fila). El parámetro 'par' puede ser 'r' (multiplicación derecha) o 'l'
% (multiplicación izquierda).
% El resultado será el cuaternión de entrada con los valores trasladados
% en dependencia del vector unitario empleado.

% Condicionamiento del cuaternión.
if length(size(quat))>2
    quat=(squeeze(quat))';
end

% Selección de la dirección de la traslación.
if quatmod(uquat)==1
switch par
    case 'r'
        ctq=quatmultiply(quat,uquat);
    case 'l'
        ctq=quatmultiply(uquat,quat);
end
else
    error('uquat should to be an unitary quaternion');
end
```

Código de la función *lbpgray_all* que calcula los patrones binarios locales a partir del mapa de Michelson obtenido.

```
function [lbpdecrinv,lbpclass, lbpbin, lbpdec, lbpbin-
rinv]=lbpgray_all(neigh, N, R);
% Function
% [lbpdecrinv,lbpclass, lbpbin, lbpdec, lbpbinrinv]=lbpgray_all(neigh, N,
% R);
% calculates the LBP values corresponding to the neighborhood neigh in a
% grayscale image.
% R = radius = 1, 2, 3.
% N = neighborhood size in terms of row/column length = 3, 5, 7. Only R =
% 1, N = 8 are considered firstly.
% lbpdecrinv = rotation invariant lbp decimal value.
% lbpclass = 10-class classification of rotation invariant lbp.
% lbpbin = raw lbp calculated for the neighborhoood, binary representation.
% lbpdec= decimal notation value corresponding to lbpbin.
% lbpbinrinv = binary representation of lbpdecrinv.
%
% Functionlbpgray_all uses the following nested functions to calculate the
% different parameters:
% function lbprinv=lbp_rinv(x, N, R); % Finds the rotation invariant lbp.
% function nlbp = lbp_class_value(x); % Evaluates the lbp class (10
%                                     % classes for N = 8).
%
% The result lbpbin is a binary vector which contains "1" or "0" in the
% positions of the 8 neighbors of the central pixel starting by North (upper
```

```
% pixel) and moving clockwise, depending upon whether their values are
% respectively higher or lower than that of the central pixel. The decimal
% value of lbpbin is lbpdec.
% N is the number of pixels considered in the lbp calculation.
% R is a number between 1 and 3 representing the radius of the neighborhood.
% In the current version only N=8, R=1 are accepted. In an extended version
% the values R=1, 2, 3 and N=8, 16, 24 are admitted,
%
% Ref. Ojala et al. Multiresolution gray-scale and rotation invariant tex-
% ture
% classification with local binary patterns. IEEE Transactions on pattern
% analysis and machine intelligence, Vol. 24 No. 7, 2002, pp. 971-987.
%
% Juan V. Lorenzo Ginori, Fredericton, September 2017.
%
if R~=1|N~=8
    error('Only R=1 amd N=8 are admitted at this moment')
end
% Depending on the value of N, neigh is [3X3], 5X5 or [7X7].
%
% Usually neigh will be [3 3] (or larger for LBP's with R>1), R==1 is the
% only value admitted in the current function.
%
% INTERPOLATION (MIGHT BE OPTIONAL)
% neigh=lbp_interp(neigh) % Interpolating the corner elements in the neigh-
% borhood.
%
% Now order the elements in neigh as a row vector and sort them to obtain
% a circular ordering. The order here is [123;456;789] and should be "cir-
% cular" 236958741 (5 at the center).
vmat=neigh(:)';
% Vector elements were ordered following neigh matrix ordering.
% Now sort values in circular order. Here we build the vector vcir.
vcir(1)=vmat(2);
vcir(2)=vmat(3);
vcir(3)=vmat(6);
vcir(4)=vmat(9);
vcir(5)=vmat(8);
vcir(6)=vmat(7);
vcir(7)=vmat(4);
vcir(8)=vmat(1);
ref=vmat(5);
% vcir
% Compare the 8-neighbor pixel values to that of the central pixel (fifth
% for
% a[3 3] neighborhood, 9 in LBP sorted ordering) and obtain an 8-elements'
% binary vector lbpbin having "1" where vcir(i)>ref=vmat(5)and zero else
% where;
%
% Sorted order.
```

```
lbpbin=zeros(1,8); % Initialize the binary lbp.
lbpbin(vcir>=ref)=1;
lbpdec=bi2de(fliplr(lbpbin)); % fliplr because bi2de takes right msb.

% Calculating the rotation invariant lbp which correspond to lbpbin.
%
lbpbinrinv=lbp_rinv(lbpbin, N, R); % See below function lbp_rinv.

lbpdecrinv=bi2de(fliplr(lbpbinrinv)); % Claculates decimal value of the
                                      % rotation invariant lbp, fliplr
                                      % because bi2de takes right msb.
%
% Classifying into regular (1 to 9) and non-regular (10) classes. See
% function lbp_class_value.
%
lbpclass=lbp_class_value(lbpbinrinv);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function nlbp = lbp_class_value(x);
% Calculating the value of the LBP considering uniform & non-uniform
% classes.
% This function must receive the rotation invariant lbp vector.
% x=LBP binary vector to be classified into one of the nine uniform classes
% or into class 10 (non-uniform).

function nlbp = lbp_class_value(x);
yy=ones(9,1); % Auxiliar vector for replications.
% Define AA as the matrix of uniform lbp vectors for 8-neighborhoods.
AA=[0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 1;
    0 0 0 0 0 0 1 1;0 0 0 0 0 1 1 1;
    0 0 0 0 1 1 1 1;0 0 0 1 1 1 1 1;
    0 0 1 1 1 1 1 1;0 1 1 1 1 1 1 1;
    1 1 1 1 1 1 1 1];
% AA and yy can either be saved and read or be generated at the beginning
% and then passed to the function. We chose here to define them inside the
% function.
xx=yy*x;      % This tensor product forms a matrix by replicating x (the
             % input binary lbp)in the 9 rows of xx.

aa=AA&xx|~AA&~xx; % Returns a matrix where the rows are the vectors which
                  % result from the logical operation that produces 1 where
                  % the bits coincide and 0 where they do not.

aas=sum(aa');      % Transpose and sum by columns the number of identical
                  % elements
                  % aas is a row vector containing the sum of columns in
                  % aa'.
```

```
th=sum(aas==8);    % Indicates the number of values equal to 8 in the sum.
                   % There could be only one (th==1), or none (th==0).

if th==0 % Non-uniform class.
    nlbp=10;
else
    nlbp=find(aas==8); % Finds the order of the element of the row vector
                       % aas which is ==8 indicating the row of aa that
                       % equals the % Uniform-class lbp to which x is equal.
end

% nlbp is the lbp class value for the rotation invariant lbp. Value  10 is
% used for the nonuniform class.
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Function lbp_rinv calculates the rotation-invariant lbp from the raw lbp
% in binary numbers. Here x is the raw binary lbp calculated for a certain
% neighborhood.
%
function lbprinv=lbp_rinv(x, N, R);
yy=[zeros(N)];% Allocating memory.
yy(1,:)=x;
for i=1:7
    yy(i+1,:)= circshift(x,i,2);
end
yy
yyd=bi2de(fliplr(yy)) % fliplr because bi2de by default takes right msb.
yym=min(yyd)
lbprinv=fliplr(de2bi(yym,8))
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Funtion lbp_interp interpolates the diagonal corners' elements in a 3X3
% neighborhood to calculate the lbp that corresponds to the center pixel.

function y=lbp_interp(neigh);
x=neigh;
y=x;
% lambda = sqrt(2)/2;
a=0.5; % a = (lambda)^2 = (sqrt(2)/2)^2;
b=0.207106781186548;% b = lambda(1-lambda)=(sqrt(2)-1)/2;
c=0.085786437626905;% c = (1-lambda)^2 = 3-2*sqrt(2))/2;
y(3,1)=a*x(3,1)+b*x(3,2)+c*x(2,2)+b*x(2,1);
y(3,3)=b*x(3,2)+a*x(3,3)+b*x(2,3)+c*x(2,2);
y(1,3)=c*x(2,2)+b*x(2,3)+a*x(1,3)+b*x(1,2);
y(1,1)=b*x(2,1)+c*x(2,2)+b*x(1,2)+a*x(1,1);
end
```

Código de la función *nlqfilter* encargada del barrido de la imagen.

```
function b = nlqfilter(varargin)
% NLQFILTER Perform general sliding-neighborhood operations on color images
% represented with quaternions.
% B = NLQFILTER(A,[M N],FUN) applies the function FUN to each M-by-N sliding
% block of A.  FUN may be a string containing the name of a function, a
% string containing an expression, or an inline function object.  FUN should
% accept an M-by-N-by-4 block as input, and return a vector result:
%
%       C = FUN(X)
%
% C is the output value for the center pixel in the M-by-N-by-4 block X.
% NLQFILTER calls FUN for each pixel in A. NLQFILTER zero pads the M-by-N
% block at the edges, if necessary.
%
% B = NLQFILTER(A,[M N],FUN,P1,P2,...) passes the additional parameters
% P1,P2,..., to FUN.
%
% B = NLQFILTER(A,'indexed',...) processes A as an indexed image, padding
% with ones if A is of class double and zeros if A is of class uint8.
%
% Class Support
% -------------
% The input image A can be of any class supported by FUN. The class of B
% depends on the class of the output from FUN.
%
%
% Example
% -------
% This call filters an RGB image using a median filter with a 3-by-3
% neighborhood:
%
% B = NLQFILTER(A,[3 3],'median(x(:))');
%
% The function declared as 'Fun' has the argument 'x' equal to the selected
% neighborhood.
%
%   See also NLFILTER and NL3FILTER.

%   Modified by Julian Cardenas. Toronto Nov 2000
%   Modified by Juan Lorenzo. Santa Clara Feb 2018
%   Adapted from the Matlab function NLFILTER created by
%   Clay M. Thompson 1-25-93
%   Copyright 1993-1998 The MathWorks, Inc.  All Rights Reserved.

[a, nhood, fun, params, padval] = parse_inputs(varargin{:});
```

```
% Expand A
[ma,na,c] = size(a);
aa = mkconstarray(class(a), padval, [ma+nhood(1)-1,na+nhood(2)-1,c]);
aa(floor((nhood(1)-1)/2)+(1:ma),floor((nhood(2)-1)/2)+(1:na),:) = a;

% Find out what output type to make.
rows = (0:nhood(1)-1); cols = (0:nhood(2)-1);
b    =    mkconstarray(class(feval(fun,aa(1+rows,1+cols,:),params{:})),    0,
size(a));

% Apply m-file to each neighborhood of a
f = waitbar(0,'Applying neighborhood operation...');
for i=1:ma
  for j=1:na
    x = aa(i+rows,j+cols,:);
    b(i,j,:) = feval(fun,x,params{:});
  end
  waitbar(i/ma)
end
close(f)

%%%
%%% Function parse_inputs
%%%
function [a, nhood, fun, params, padval] = parse_inputs(varargin)

switch nargin
case 0
    error('Too few inputs to NLQFILTER');
case 1
    error('Too few inputs to NLQFILTER');
case 2
    error('Too few inputs to NLQFILTER');
case 3
    if (strcmp(varargin{2},'indexed'))
        error('Too few inputs to NLQFILTER');
    else
        % NLQFILTER(A, [M N C], 'fun')
        a = varargin{1};
        nhood = varargin{2};
        fun = varargin{3};
        params = cell(0,0);
        padval = 0;
    end

otherwise
    if (strcmp(varargin{2},'indexed'))
        % NLFILTER(A, 'indexed', [M N], 'fun', P1, ...)
        a = varargin{1};
```

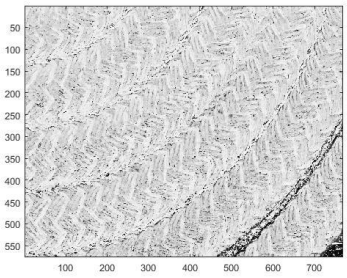
```
        nhood = varargin{3};
        fun = varargin{4};
        params = varargin(5:end);
        padval = 1;

    else
        % NLQFILTER(A, [M N], 'fun', P1, ...)
        a = varargin{1};
        nhood = varargin{2};
        fun = varargin{3};
        params = varargin(4:end);
        padval = 0;
    end
end

if (isa(a,'uint8') || isa(a,'uint16'))
    padval = 0;
end

% If we have been given an eval-able expression, remake it as an inline
% function object.
if (ischar(fun) && (any(fun<48) || any(fun>=91 & fun<=94)))
    fun = inline(fun,length(params));
end
```

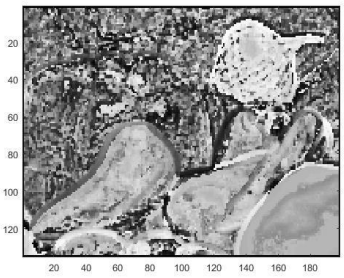
Código de la función *mkconstarray* (subfunción de *nlqfilter*).

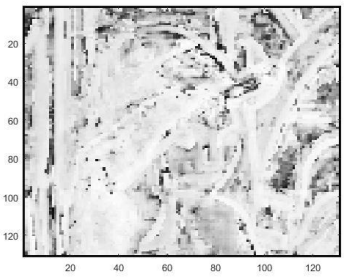
```
function out = mkconstarray(class, value, size)
%MKCONSTARRAY creates a constant array of a specified numeric class.
% A = MKCONSTARRAY (CLASS, VALUE, SIZE) creates a constant array of value
% VALUE and of size SIZE.

% Copyright 1993-1998 The MathWorks, Inc.  All Rights Reserved.

out = repmat(feval(class, value), size);
```

Anexo II Mapas de Michelson de distintas imágenes utilizando el método basado en cuaterniones con vector unitario *i*.

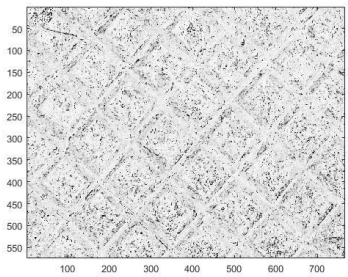


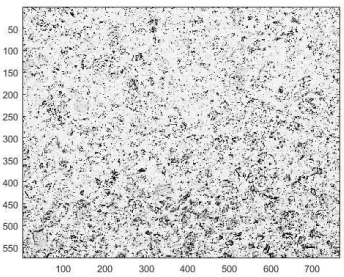


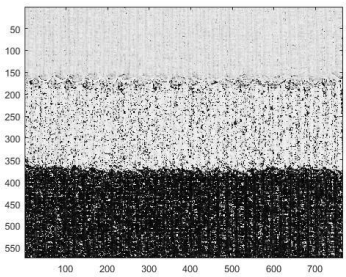












Anexo III Mapas de Michelson de distintas imágenes utilizando el método basado en la fase de cuaterniones con vector unitario *i*.