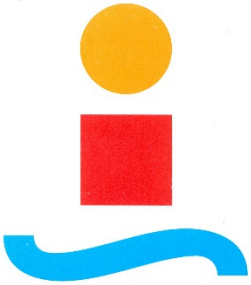


escuela superior de
INGENIEROS DE SEVILLA

DISEÑO DE UN SISTEMA AUTOMÁTICO DE IDENTIFICACIÓN DE NUBES BASADO EN VISIÓN ARTIFICIAL

Eduardo Fernández-Cantalejo Padial

DISEÑO DE UN SISTEMA AUTOMÁTICO DE IDENTIFICACIÓN DE NUBES BASADO EN VISIÓN ARTIFICIAL

Memoria presentada como parte de los requisitos para la obtención del título de Máster en Automática, Robótica y Telemática por la Universidad de Sevilla.

30 de noviembre de 2016

Tutelada por:

D. Manuel Ruiz Arahal

Índice general

Índice de Figuras

Índice de Tablas

Resumen / Abstract

Este documento describe el diseño e implementación de un Sistema Automático de Identificación de Nubes basado en visión artificial mediante el análisis cromático y de texturas de las imágenes tomadas del cielo. La identificación de nubes aporta una información importante a sistemas de meteorología u otros que se encuentren influidos por la posición y tipo de nubes, como puede ser un campo solar. A lo largo del documento se analizan las mejores características cromáticas y texturales basadas en la matriz de co-ocurrencia de grises que describa de la forma más precisa posible cada tipo de nube. Estas características se emplearán para realizar una segmentación a nivel de píxel de imágenes, tanto de un banco de imágenes de nubes como, en la fase final del proyecto, de imágenes tomadas "en vivo". Los aportes clave de este trabajo radican en el desarrollo de una técnica mixta cromático-textural que mejoran los resultados de cada tipo por separado y en la implementación del sistema en una plataforma hardware operativa.

1 | Introducción

Este documento describe el diseño e implementación de un Sistema Automático de Identificación de Nubes basado en visión artificial mediante el análisis cromático y de texturas de las imágenes tomadas del cielo, conforme al esquema básico de funcionamiento mostrado en la Figura 1.

La visión artificial como valor de entrada a los sistemas automatizados es un componente cada vez más extendido en la actualidad. En un mercado capitalista y competitivo como el actual supone un importante valor añadido en productos de última tecnología y sistemas productivos en general, abaratando los costes debido al uso de instrumentación más especializada.

La carrera tecnológica en la que nos encontramos inmersos se encamina hacia la implementación de tecnologías que emulan, cada vez con mayor fidelidad y fiabilidad, el comportamiento propio de las personas, complementándolas con funcionalidades adicionales.

Este proyecto recorre las tecnologías disponibles en la actualidad y analiza la mejor solución posible para la implementación de un sistema de identificación automática de nubes basado en visión artificial. El desarrollo de este sistema parte de la necesidad propuesta para optimizar el control y la operación de plantas solares, tratando de evitar los efectos perniciosos de los periodos de sombras de nubes que proyectan sobre los paneles solares.

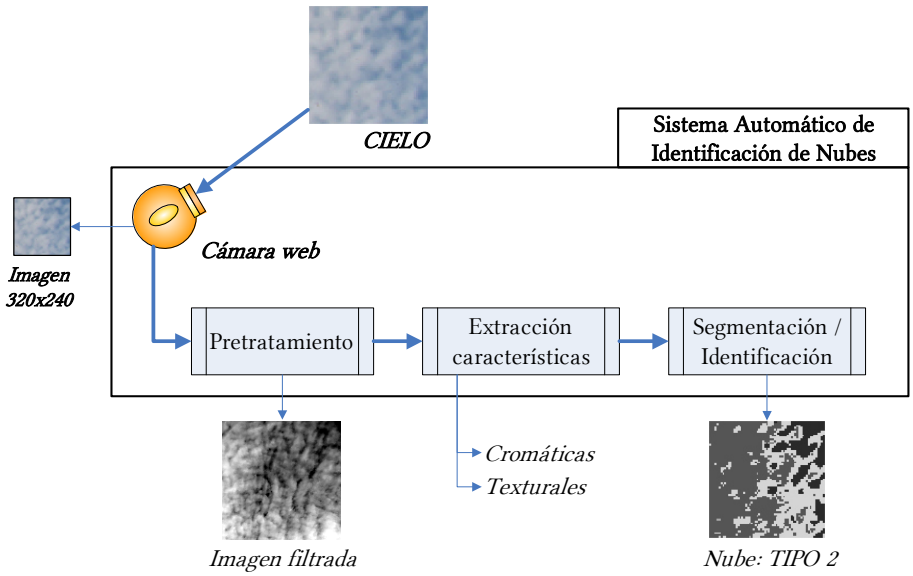


Figura 1. Arquitectura general del Sistema Automático de Identificación de Nubes

La caracterización y clasificación de imágenes, que suponen la base de este proyecto, es un problema conocido y ampliamente documentado en el mundo de la ingeniería. Muchos autores han abordado de forma específica el problema de clasificación de nubes, tanto aquellas tomadas desde el satélite como desde una estación terrestre. Las técnicas empleadas en dichos estudios son muy dispares dada la variada naturaleza las características escogidas para realizar la clasificación.

Este documento repasa las técnicas más relevantes y selecciona las técnicas más prometedoras, tomando como principio básico el tratamiento de texturas de las imágenes. Así, por un lado, se abordará una técnica cromática basada en la distribución de colores de cada píxel en el espacio RGB y por el otro una técnica textural basada en el análisis de texturas mediante matrices de co-ocurrencia de nivel de gris.

Adicionalmente se propone una técnica inédita que emplea conjuntamente características cromáticas y texturales, con el objetivo de mejorar los resultados que cada una de las técnicas de manera individual.

El método seguido en este proyecto consiste en una primera fase de análisis de las técnicas cromática y textural de forma independiente para la obtención de las mejores características descriptoras de cada tipo de nube, seguido de una fase de segmentación de imágenes almacenadas para, posteriormente, implementar un sistema de identificación automática de nubes a partir de un streaming de video "en vivo".

1.1 Antecedentes

Durante las últimas décadas se han desarrollado numerosas técnicas para la clasificación de patrones basados en diferentes características como la forma, el color y la textura. Cada una de estas aproximaciones muestra su conveniencia en función del problema y el tipo de patrón que se aborde.

En el caso del reconocimiento e identificación de nubes la forma no será determinante debido a su alta variabilidad, por lo que se descartará este tipo de características. El problema se afronta, por tanto, como la búsqueda de las mejores características cromáticas y texturales que, primero de forma independiente y posteriormente de forma conjunta, mejor describa cada tipo de nube.

El interés de la detección y clasificación de nubes es más notorio en algunas aplicaciones que en otras. Por ejemplo, en el caso de plantas fotovoltaicas las nubes pueden producir una cada instantánea de la radiación solar directa que se traduce en una gran perturbación en la producción que ha de ser absorbida por el sistema eléctrico. En el caso de plantas termosolares el efecto puede ser aliviado por la inercia térmica del sistema, sobre todo si se usa acumulación de calor en aceite, sales u otro medio. Para otras aplicaciones la radiación solar indirecta juega un papel importante por lo que el paso de nubes dispersas puede tener escasas consecuencias, por lo que basta con conocer la nubosidad general. En todos los casos la predicción de la nubosidad y el estudio de su impacto tienen un gran interés para optimizar la operación de la planta.

1.2 Objetivos globales

Este proyecto se encuentra enmarcado dentro de un proyecto global, Estimación y Predicción Distribuida de la Radiación para Control de Campos Solares (CODISOL), que busca el ajuste automático de una planta solar ante la llegada de nubes de manera que se sea capaz de modificar el flujo de agua para adaptarse a la falta de radiación del sol.

El proyecto comprende el estudio y el desarrollo de un sistema piloto de detección de nubes y predicción de la radiación solar a corto plazo para el control de campos solares. La realización del mismo involucra desde el estudio detallado del estado del arte hasta la fase de desarrollo del prototipo. Esto resulta de gran interés y aplicación en plantas solares.

El sistema permitirá la detección de cúmulos nubosos y podrá predecir la fracción de campo solar que se verá afectada por la disminución de radiación consecuente del paso de las nubes. De esta forma, el sistema de control podrá anticipar el efecto de estas perturbaciones, lo cual tiene un impacto en la optimización y la eficiencia de la operación, permitiendo la manipulación de variables fundamentales y la previsión y planificación correctas de uso de fuentes de energía auxiliares para mantener la producción lo más cerca posible de la generación diaria prevista. La Figura 2 muestra el esquema P&ID básico del funcionamiento del sistema.

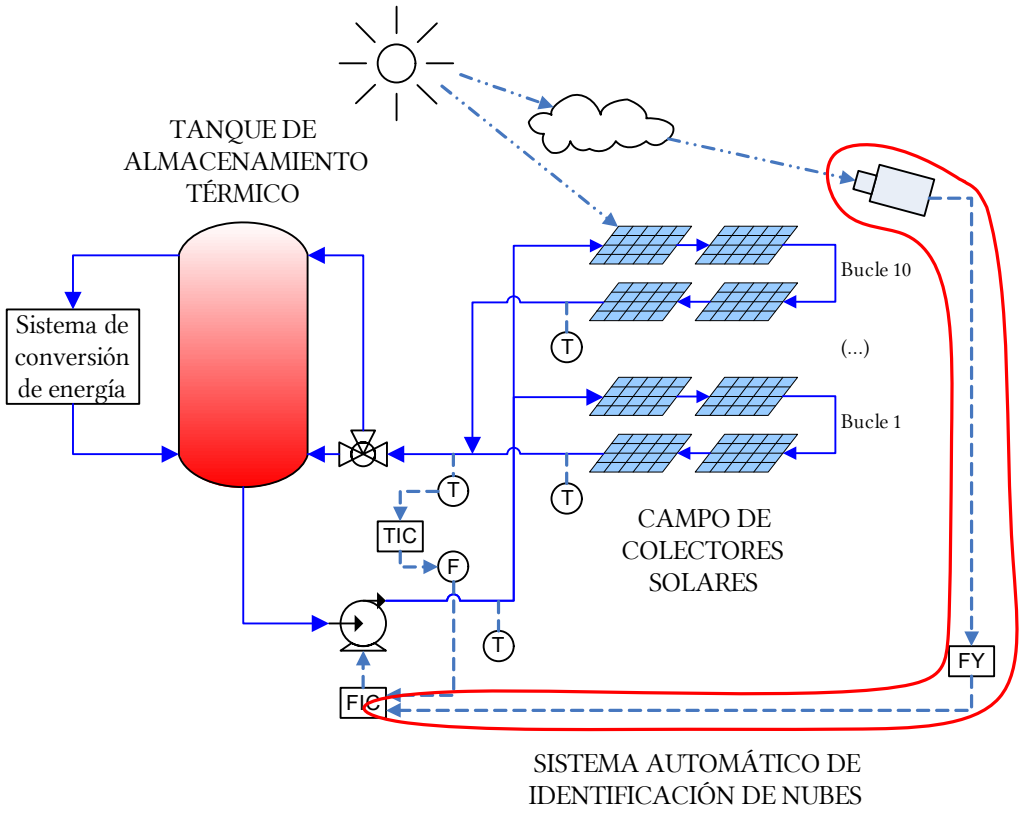


Figura 2. Esquema P&ID de un campo solar del proyecto CODISOL

El desarrollo de este sistema tendrá un doble impacto en el sector. Por un lado, contribuirá al aumento de la competitividad de la energía solar frente a la generación de energía a partir de carbón y otros combustibles fósiles. Adicionalmente, contribuirá también a la mejora en la integración fiable de las plantas de energía solar en la red eléctrica, ya que este sistema hará posible una mejor regulación de la producción y un ajuste a la generación diaria prevista a pesar de las perturbaciones nubosas.

Se desarrollarán modelos de predicción a partir de medidas meteorológicas, así como a partir de imágenes recogidas por un sistema de cámaras de ojo de pez. Una vez desarrollado el modelo, este sistema de cámaras se utilizará como una red de sensores distribuidos para predecir la radiación solar en la planta. Las distintas cámaras estarán conectadas entre ellas y con el sistema de control a través de una red inalámbrica, y utilizarán algoritmos de consenso para mejorar las predicciones. Se aplicarán técnicas de estimación distribuidos en red que tengan en cuenta los posibles retardos y pérdidas de paquetes en las comunicaciones.

Figura 3. Arquitectura general de un campo solar del proyecto CODISOL

Para validar los algoritmos desarrollados, se pretende aplicar los resultados a dos instalaciones solares: una de ellas está disponible en los laboratorios del departamento de Ingeniería de Sistemas y Automática de la Escuela Técnica Superior de Ingeniería de Sevilla, con una arquitectura general similar a la mostrada en la Figura 3. La segunda batería de pruebas se realizará en las instalaciones de un socio industrial de amplísima experiencia en el sector, lo que permitirá que participe en las propuestas, valide los resultados y se difundan e implementen los resultados científicos de este proyecto de investigación, aumentando así su impacto nacional e internacional.

Los subproyectos básicos asociados a Codisol se podrían asimilar a los siguientes:

- Módulo de cálculo de la sombra proyectada.
  - Extracción de imágenes rectificadas a partir de la cámara con lente de ojo de pez.
  - **Identificación del tipo de nube**, que aportará en última instancia una estimación de la altura de la nube.
  - Cálculo de la distancia de la nube.
  - Cálculo de la sombra proyectada a partir de la distancia, la altura y la posición del sol.
- Módulo de control de la planta solar.

1.3 Objetivos específicos

Este proyecto aborda el problema de la identificación del tipo de nube, recorriendo las fases de análisis y segmentación sobre una plataforma de simulación para lograr una implementación sobre sistemas reales y funcionales.

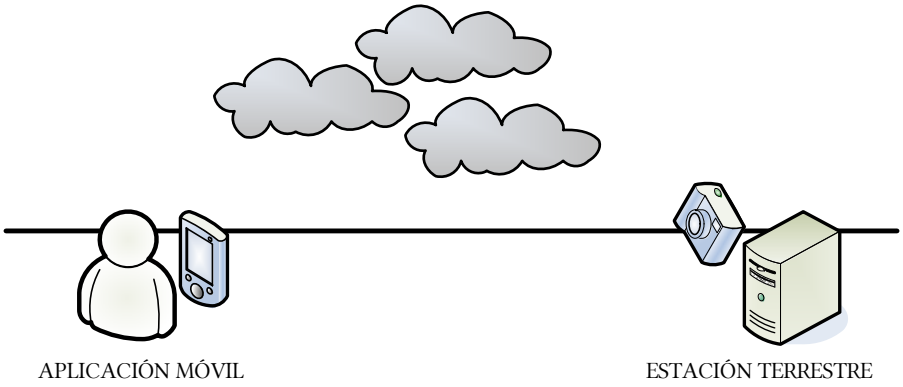


Figura 4. Objetivos específicos del proyecto

Se proponen como objetivos específicos el desarrollo de dos aplicaciones reales a los resultados del análisis:

- Una estación terrestre, sobre una plataforma hardware de reducidas dimensiones y económicamente viable para un despliegue notable de estaciones que cumplan con los objetivos del objetivo global que cumpla con los requerimientos del proyecto CODISOL.
- Una aplicación móvil que haga disponible la aplicación a un mayor número de usuarios pero con funcionalidades limitadas.

Este documento aborda cada asunto en secciones independientes. Así, la base teórica y fundamentos se recogen en la sección 3 del documento.

La metodología de cada una de las etapas del proyecto y de cada implementación realizada se explican en detalle en la sección 4.

Los resultados del Análisis y la Segmentación se recogen en las secciones 5 y 6 del documento, mientras que el diseño y desarrollo del Interfaz de Usuario (en MATLAB), la estación terrestre y la aplicación móvil ocupan las secciones 7, 8 y 9.

Finalmente, la sección 10 aborda las conclusiones del proyecto en su conjunto, dando paso en las secciones 11 y 12 a las referencias y el glosario.

2 | Alcance

El presente proyecto posee un objetivo triple, en el marco de la identificación de patrones de nube:

- Análisis descriptivo de las características cromáticas y texturales que mejor describen cada tipo de nube;
- Segmentación de imágenes almacenadas; y
- Desarrollo de un clasificador que caracterice cada píxel de las imágenes en vivo.

El objetivo último es, por tanto, el desarrollo de una aplicación que sea capaz de clasificar cada tipo de nube presente en el cielo en cada instante a partir de las imágenes captadas por una cámara colocada a la intemperie.

La Figura 5 muestra las fases del proyecto, donde la Fase I representa el alcance original del presente TFM y las Fases II y III corresponden a ampliaciones realizadas "motu propio" para la obtención de un sistema plenamente funcional en estaciones terrestres y móviles. De hecho, se ha realizado una publicación en las XXXVII Jornadas de Automática de los resultados obtenidos del Análisis Básico (ver Anexo 4).

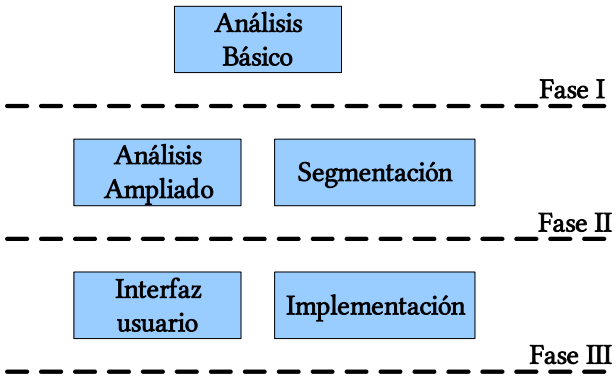


Figura 5. Fases del proyecto

El alcance detallado del proyecto se muestra en la Figura 6. El alcance original consistía en el análisis de las imágenes del banco de imágenes del SWIMCAT y un muestreo de las mismas para la aplicación de las técnicas cromática, textural y mixtas y finalmente la implementación de un clasificador de mínima distancia para la comparación de dichos resultados.

La Ampliación I ha consistido en el análisis de imágenes adicionales, derivadas siempre del banco de imágenes SWIMCAT, en búsqueda de preprocesamiento que potencialmente pudiera mejorar los resultados. Así mismo, se introdujeron una nueva técnica y dos clasificadores adicionales para, más allá del análisis, obtener imágenes segmentadas en las diversas combinaciones.

La Ampliación II introduce una Interfaz de Usuario en MATLAB para facilitar la aplicación del método descrito en este documento al análisis y segmentación de imágenes que pudieran proceder de otros bancos de imágenes.

Finalmente, la Ampliación III convierte el análisis teórico en un sistema real sobre dos plataformas separadas con desarrollos independientes: la primera basada en el computador de placa reducida Raspberry Pi y la segunda una plataforma software para su ejecución en dispositivos con sistema operativo Android.

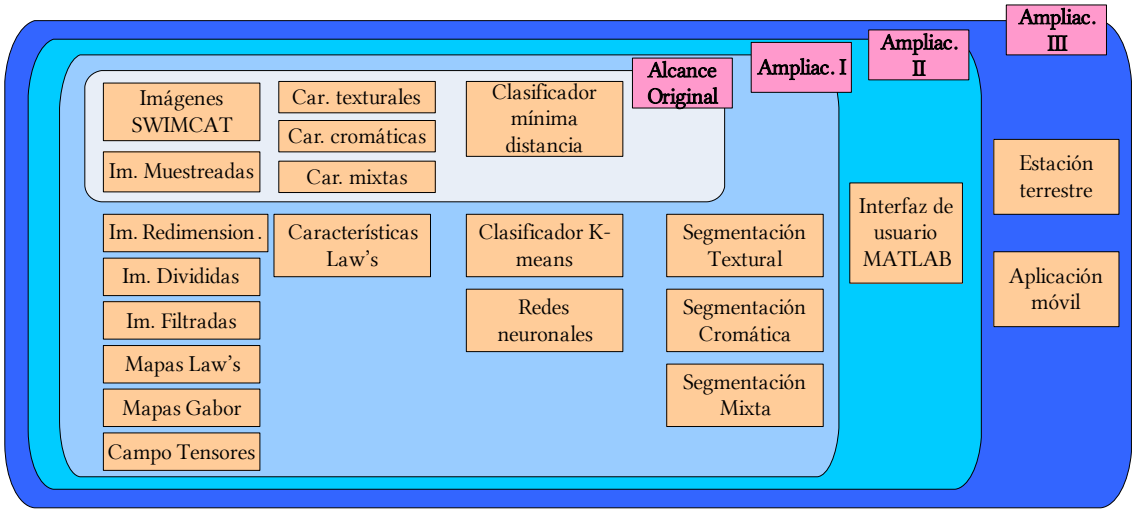


Figura 6. Ampliaciones en el alcance del proyecto

Este proyecto empleará un grupo de 784 fotografías diferentes tomadas con la cámara desde la que posteriormente se realizará el análisis en vivo y abordará el problema desde un punto de vista de la invariabilidad y robustez ante cambios de iluminación o tipo de cámara y, cuando esto no sea posible, emplear técnicas que minimicen en lo posible esta dependencia.

3 | Fundamentos

3.1 Principios generales

Las nuevas generaciones de sensores aportan información e imágenes de diferente naturaleza o redundante que en la actualidad puede tratarse y fusionarse para la consecución de resultados mejorados respecto a los métodos clásicos que aún se emplean. Las imágenes multivariadas son un medio con el que propagar información compleja, masiva y heterogénea para la alimentación de sistemas de análisis de datos.

El término genérico "imagen multivariada" incluye medios de información diversa (imagen en color, datos multimodales, datos multiespectrales e hiperespectrales, imágenes multi-fecha, datos heterogéneos y observaciones multifuente).

En las imágenes multivariadas los datos pueden ser de diferentes tipos:

- Datos multibanda: desde el caso más sencillo de imagen en color (p. ej. imágenes en RGB) a los casos más complejos de imágenes multiespectral (de 10 a 100 bandas espectrales), hiperespectrales (de 100 a 1000 bandas espectrales) o superespectrales (de más de 1000 bandas espectrales).
- Datos multimodales: la noción de modalidad se refiere a la adquisición de datos describiendo la misma escena pero por diferentes protocolos.
- Datos multivista: la misma escena se observa por el mismo sensor desde diferentes posiciones o por varios sensores usando la misma modalidad.
- Datos heterogéneos: tenemos acceso a matrices por las que la comprensión de los modelos físicos asociados con estos parámetros es indispensable.

Este estudio aborda el problema de la identificación de nubes desde un punto de vista de tratamiento de imágenes multibanda considerando todas las posibles bandas cromáticas extraídas a partir de las bandas RGB que se reciben directamente del sensor.

3.2 Tipos de nube

En 1956 la Organización Meteorológica Mundial (OMM) clasificó los tipos de nubes que existen por géneros, os cuales se subdividen en especies y en variedades. A continuación, resumiremos los 10 géneros principales de nubes pertenecientes a la clasificación recopilada en el Atlas Internacional de Nubes de la OMM.

1. Cirrus (Ci): nubes formadas por filamentos que adoptan un aspecto fibroso y delicado. Adquieren formas estiradas o estilizadas a modo de fajas estrechas o ramificadas. Su constitución física es de cristales de hielo.
2. Cirrostratus (Cs): nubes traslúcidas en forma de velo, casi transparente, que dejan entrever el disco solar o lo pueden acabar cubriendo completamente. A estas nubes se asocian fenómenos ópticos como halos solares (o lunares), iridiscencias o parhelios. Están constituidas por cristales de hielo.

3. **Cirrocumulus (Cc)**: nubes en forma de capa o manto delgado que adoptan formas de onda, rizos o gránulos, sin sombras propias. Están constituidas por cristales de hielo.

4. **Altrostratus (As)**: nubes de tipo medio-alto, estratiformes y mixtas, es decir, compuestas por gotas de agua o cristales de hielo. Presentan partes bastante delgadas que permiten entrever el sol y adoptan una forma de capa nubosa grisácea. Este tipo de nubosidad puede llegar a dejar en la alta montaña alguna llovizna débil o algún copo de nieve.

5. **Altocumulus (Ac)**: capa de nubes de aspecto fibroso o difuso en forma de anco o manto, de color grisáceo o blanco y gris mezclados, que tiene sobras propias y está compuesto por láminas o rodillos.

6. **Nimbostratus (Ns)**: capa nubosa gris, frecuentemente sombría, de aspecto borroso debido a las precipitaciones intermitentes de lluvia o nieve que de ella se producen. Por debajo de este manto nuboso, aparecen nubes bajas desgarradas o soldadas o no con ella. Esta capa nubosa puede llegar a tapar completamente el sol debido a su espesor.

7. **Stratocumulus (Sc)**: capa de nubes o manto gris o blanquecino compuesto por rodillos, losas o masas globulares alargadas, de aspecto no fibroso, soldados o no, que pueden llegar a cubrir el cielo en su totalidad.

8. **Stratus (St)**: capa nubosa generalmente gris, con base bastante uniformes que puede dejar alguna llovizna débil o algún prisma de hielo. Cuando el sol es visible a través de la capa, su contorno es claramente discernible.

9. **Cumulus (Cu)**: nubes densas, separadas entre sí, con contornos bien delimitados, que se desarrollan verticalmente, adoptando formas redondeadas, algodonosas o en forma de torres. En su parte superior se conserva una protuberancia a modo de coliflor. Cuando el sol las ilumina, adoptan una tonalidad blanca muy brillante, aunque su base acostumbra a ser oscura.

10. **Cumulonimbus (Cb)**: es la nube de tormenta por excelencia, densa y de considerable desarrollo vertical, en forma de montaña o de enormes torres. Su parte superior acostumbra a ser lisa, fibrosa o estriada y casi siempre aplastada, adoptando una forma de yunque. Por debajo de la base de esta nube –muy oscura o casi negra– aparecen nubes bajas, desgarradas, soldadas o no a ella, o precipitaciones en forma de virga.

La Figura 7 muestra ejemplos de cada uno de los 10 tipos de nube descritos anteriormente.

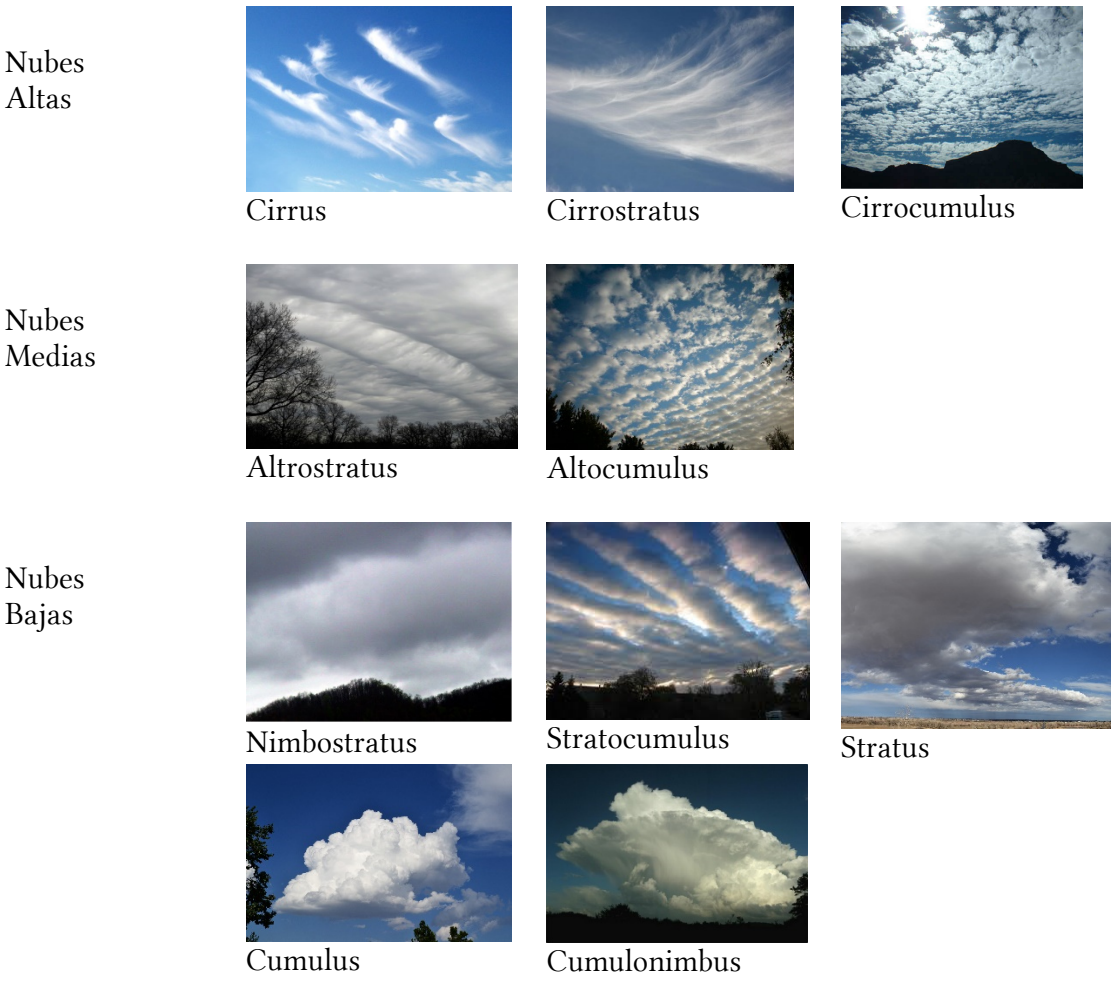


Figura 7. Tipos de nube

Según la altitud, las nubes se clasifican en altas, medias y bajas. Las nubes altas están situadas entre los 5 y los 13km de altura, las medias las encontramos entre los 2 y los 7 km y las bajas las veremos entre el nivel del suelo y los 2 km.

3.3 Generación de imágenes

3.3.1.1 Imágenes con filtro de contraste

Las imágenes son filtradas para realzar su textura de manera que se esperarían ligeras mejoras en la identificación.

La regla empleada para su conversión a escala de grises previa aplicación del filtro se realiza mediante la siguiente ecuación de ajuste:

$$ImagenGris = 0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B$$

Posteriormente se aplicará un filtro de ajustes que mapea los valores de intensidad de la imagen en color *I* a la imagen en escala de grises *J* de tal manera que el 1% de los datos saturan a las intensidades menores y mayores de *I*. De esta manera se incrementa el contraste de la imagen de salida *J*.

3.3.1.2 Mapas de Laws

Como se comenta en la sección 4.1.2.3, de los posibles mapas de Laws que se pueden generar se ha tomado el resultado de las convoluciones de las máscaras L5E5-E5L5.

3.3.1.3 Imágenes con filtro de Gabor

Un filtro de Gabor se obtiene mediante la modulación de una senoide con una gaussiana.

Para el caso de señales unidimensionales, una senoide 1D se modula con una gaussiana. Este filtro por tanto responderá a algunas frecuencias pero sólo en una parte localizada de la señal.

Las señales bidimensionales, como las imágenes, consideran la senoide en una orientación determinada que, combinada con la gaussiana, obtiene un filtro de Gabor.

Sea $g(x, y, \theta, \varphi)$ la función que define el filtro de Gabor centrada en el origen con θ como frecuencia espacial y φ como la orientación. Se pueden ver los filtros de Gabor como:

$$g(x,y,\theta,\varphi) = e^{-\left(\frac{x^2+y^2}{\sigma^2}\right)} \cdot e^{(2\pi\theta i(x\cdot cos\varphi + y\cdot sin\varphi))}$$

Se demuestra que σ, la deviación estándar del núcleo gaussiano depende de la frecuencia espacial a medir, es decir, θ.

La respuesta del filtro de Gabor a una imagen se obtiene mediante una operación de convolución 2D. Sea $I(x,y)$ la imagen y $G(x, y, \theta, \varphi)$ la respuesta del filtro de Gabor con frecuencia θ y orientación φ a una imagen en el punto (x,y) en la imagen del plano, entonces G(.) se obtiene como:

$$G(x,y,\theta,\varphi) = \int\int I(p,q) \cdot g(x-p, y-q, \theta, \varphi) dp\, dq$$

En la respuesta al filtro de Gabor, las zonas blancas indican una alta amplitud de respuesta, mientras que las zonas oscuras indican bajo nivel de respuesta. Una muestra del tipo de mapas creados por el filtro de Gabor en función de la orientación y la frecuencia se puede observar en la Figura 8.

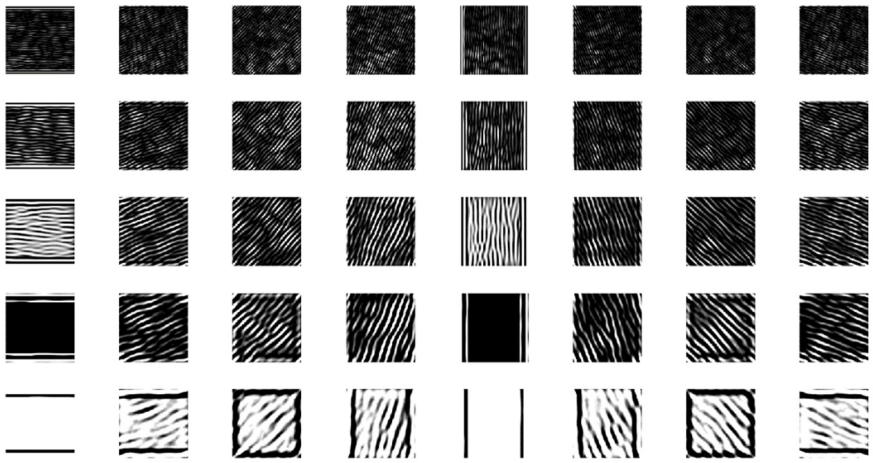


Figura 8. Imagen con filtro de Gabor aplicada

Esta imagen muestra una nube de tipo 'Patrón' en cinco escalas y ocho orientaciones diferentes, empleando un filtro de Gabor de dos dimensiones de 39 x 39. Las imágenes más claras denotan la orientación predominante de las nubes.

En este estudio, y con el único objetivo de realizar un primer acercamiento a esta técnica, se empleará la componente de mayor escala y orientación -45º como conjunto de imágenes que podrían servir para la clasificación de imágenes.

3.3.1.4 Campo tensorial

La votación tensorial es una técnica robusta cuyo objetivo es la extracción de información perceptual de nubes de puntos. Esta técnica es aplicable a nuestro proyecto mediante la transformación de las imágenes analizadas tratando de aislar las características texturales para una mejor detección.

En 3D, la votación tensorial estima medidas de prominencia de la probabilidad de un punto a pertenecer a una superficie, una curva o un nudo, o si se trata de un caso aparte.

Se basa en la propagación y agregación de las normales más probables mediante tensores de segundo orden que usan matrices simétricas semidefinidas positivas, a través de un proceso de votación similar a la convolución, asumiendo que los puntos vecinos pertenecen a la misma superficie suave. Esta técnica se ha mostrado versátil dado que ha sido adaptada a problemas más allá de los que originalmente se consiguieron excelentes resultados.

La principal hipótesis hecha por el tensor estructural es que la orientación de los gradientes cambia lentamente en regiones con bordes y rápidamente en regiones con esquinas. Además, asume que el tamaño del gradiente es pequeño en regiones planas y grande en regiones con bordes o esquinas. Por tanto, el tensor estructural estima la estructura local de la imagen mediante una suma ponderada de gradientes en una vecindad.

Para una imagen en escala de grises el tensor estructural J se define como la convolución de una función gaussiana con el gradiente tensorizado de la imagen.

$$J = G_p \cdot \nabla u \nabla u^T$$

donde G_p es la función gaussiana con media cero y desviación estándar ρ, ∇u el el gradiente de la imagen u y $\nabla u \nabla u^T$ representa el gradiente tensorizado en cada píxel. Una aproximación relacionada se basa en filtros de cuadratura. Además, se han propuesto extensiones usando derivativos de alto orden o estructuras curvadas.

En una primera fase se inicializa un tensor en cada punto de la nube o bien con una estimación de la normal o bien con un tensor con forma de bola si no hay información previa. Después cada tensor se descompone en tres componentes: un *palo*, un *plano* y una *bola*. Cada componen arroja votos que son los tensores que codifican la dirección más probable de la normal en un punto de la vecindad tomando en consideración la información codificada por el votante de ese componente. Finalmente, los votos se suman y analizan para estimar las medidas de superficie, curva y nodo en cada punto. Los puntos con baja prominencia se asumen como casos aparte.

Más formalmente el tensor de votación se expresa como:

$$TV(\boldsymbol{p}) = \sum_{\boldsymbol{q} \in vecindad(\boldsymbol{p})} SV(\boldsymbol{v}, S_q) + PV(\boldsymbol{v}, P_q) + BV(\boldsymbol{v}, B_q)$$

donde **q** representa cada uno de los puntos de la vecindad de **p**; SV, PV y BV son los votos de tensores *palo*, *plato* y *bola* de arrojan sobre p cada uno de los componentes de q; **v** = **p**-**q**; y Sq, Pq y Bq son los componentes *palo*, *plato* y *bola* del tensor en q respectivamente.

Estos componentes vienen dados por:

$$S_q = (\lambda_1 - \lambda_2)(e_1 e_1^T)$$

$$P_q = (\lambda_2 - \lambda_3)(e_1 e_1^T + e_2 e_2^T)$$

$$B_q = \lambda_3(e_1 e_1^T + e_2 e_2^T + e_3 e_3^T)$$

donde λ_i y e_i son los mayores autovalores y autovectores del tensor en **q**.

Lo que se obtiene cuando se aplica esta técnica a las imágenes de nubes son nuevas imágenes con las características cambiantes resaltadas, como la mostrada en la Figura 9.

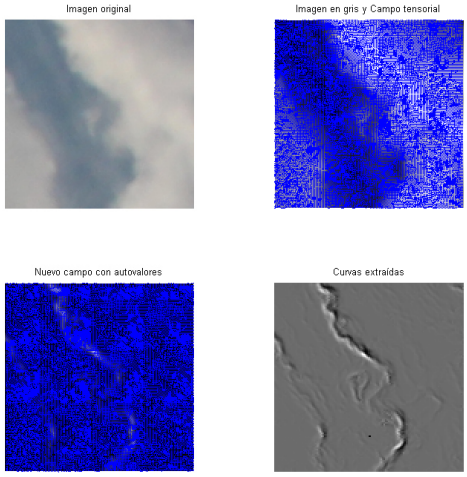


Figura 9. Descomposición conforme a campos tensoriales

3.4 Análisis cromático

3.4.1 Conversión entre espacios de color

Este apartado describe las ecuaciones para la transformación entre los diferentes espacios de color que son emplean habitualmente en los campos de visión artificial.

3.4.1.1 Teoría Tri-cromática

El color es extremadamente subjetivos y personal por lo que tratar de atribuir números a las reacciones del cerebro al estímulo visual es muy difícil. El objetivo de los espacios de color es ayudar en el proceso de descripción del color, ya sea en personas o entre máquinas o programas.

La retina del ojo percibe los colores usando tres bandas de color correspondientes a rojo, verde y azul claro, aunque biológicamente tenemos dos tipos de células que recogen la sensibilidad del color (conos) e intensidad (bastones) que posteriormente el cerebro procesa las diferentes "sensaciones" de color. Estas sensaciones se estandarizan conforme al CIE como las siguientes:

- Brillo: sensación humana por la que un área exhibe más o menos luz.
- Tonalidad: sensación humana de acuerdo a la cual un área se muestra similar a una o proporciones de dos, de los colores percibidos rojo, amarillo, verde y azul.
- "Coloridad": sensación humana por la que un área parece exhibir más o menos tonalidad.
- Luminosidad: sensación de brillo de un área relativo a una referencia blanca en la escena.
- Croma: coloridad de un área en relación al brillo de un blanco de referencia.
- Saturación: coloridad de un área en relación a su brillo.

La teoría tri-cromática describe la forma en que tres luces separadas (rojo, verde y azul) pueden generar cualquier color visible, basado en el uso de los sensores sensibles al color del ojo. Esta es la base sobre la que operan la fotografía y la impresión, esto es, empleando tres tintas diferentes para reproducir el color de una escena. Es también la forma en que operan los espacios de color de los ordenadores, mediante el uso de tres parámetros que definen un color.

3.4.1.2 Espacios de color

El espacio de color a emplear dependerá de la aplicación hacia la que se oriente el estudio o procesamiento. Podemos encontrar los siguientes espacios de color:

- RGB – Red Green Blue

  Se trata del sistema de color aditivo basado en la teoría tricromática. A menudo se encuentra en sistemas que usan monitores CRT para mostrar imágenes, así como sistemas de ordenador, televisión y video. RGB es fácil de implementar pero es no-

lineal respecto a la percepción visual. Es dependiente del dispositivo y la especificación de los colores es semi-intuitivo.

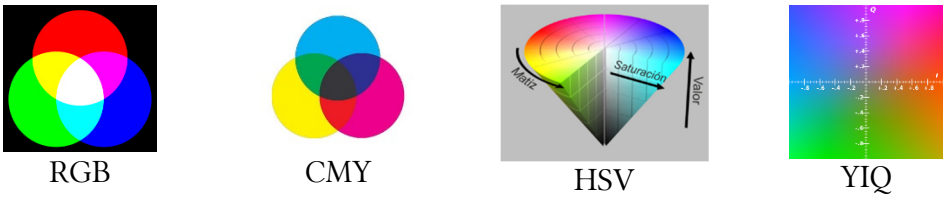


Figura 10. Espacios de color RGB, CMY, HSV e YIQ

- CMY(K) – Cyan Magenta Yellow (Black)

  Es un espacio de color sustractivo y se emplea principalmente en impresión y salidas de impresión en papel. El cuarto componente se incluye para mejorar el rango de densidad y la gama de colores. Es sencillo de implementar pero la conversión propia de RGB a CMY es compleja. CMY es dependiente del dispositivo, no-lineal respecto a la percepción visual y razonablemente no intuitivo.

- HSV – Hue Saturation Value

  Representa un conjunto de espacios de color con nombres alternativos coom HSI (intensidad), HSL (luminosidad), HCI (croma / coloridad), HVC, TSD (oscuridad), etc. La mayoría de estos espacios de color son transformaciones lineales desde el espacio RGB y por tanto son dependientes del dispositivo y no lineales. Su ventaja radica en un manera extremadamente intuitiva de especificar el color. Es muy sencillo seleccionar una tonalidad deseada y entonces modificarla ligeramente para ajustar su saturación e intensidad.

- YIQ – Luminancia – Crominancia

  Se trata de un conjunto de espacios de color de transmisión del color en televisión, junto con YUV, YbCr e YCC. Los espacios YIQ e YUV son espacios analógicos para sistemas NTSC y PAL respectivamente mientras que YCbCr es un estándar digital. Estos espacios de color separan el espacio RGB en información de luminancia y crominancia y son útiles en aplicaciones de compresión. Esos espacios son dependientes del dispositivo pero fueron diseñados para un uso bajo condiciones estrictamente definidas en sistemas cerrados. Son también bastante anti-intuitivas.

- CIELab

  Son dos los espacios de color CIE: CIELuv y CIELab. Son casi lineales con la percepción visual o al menos tan cercano como cualquier espacio de color puede llegar a ser. Como se trata de sistemas CIE de medida del color, esto es, basados en la misma visión humana, estos espacios son independientes del dispositivo pero tienen el problema de ser bastante anti-intuitivo a pesar de que el parámetro L tiene una buena correlación con el brillo percibido.

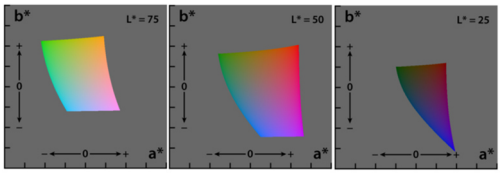


Figura 11. Mapa de color CIELab

La Figura 10 y Figura 11 muestras cómo se componen cada uno de los espacios de color descritos.

Adicionalmente, se han incluido en el Anexo 1 las conversiones matemáticas entre los diferentes espacios de color.

3.5 Análisis de texturas

Son varios los tipos de características que se han definido para el análisis de texturas:

- Métodos estadísticos

  La distribución espacial de valores de gris es una calidad que define la textura. Analizando la distribución espacial de los valores de gris, se computan características locales de la textura.

  - Primer orden: Basados en el histograma de los niveles de gris: media, desviación típica, varianza, etc. No se emplean en aplicaciones porque no funcionan de forma consistente.
  - Segundo orden:
    - Matriz de co-ocurrencia: estima la probabilidad de que dos píxeles a una distancia d tengan el mismo valor. A partir de ella se calculan ciertos estadísticos como la entropía.
    - Función de autocorrelación: La función de autocorrelación de una imagen puede usarse para evaluar tanto la regularidad como la finura/tosquedad de la textura. Puede determinarse en el dominio frecuencial, a través del espectro de la transformada de Fourier.

- Métodos geométricos

  Esta familia de métodos considera la textura compuesta por primitivas. Intentan describir las primitivas y las reglas que gobiernan la organización espacial. Tras identificar estas primitivas, la textura puede analizarse bajo dos perspectivas: computar propiedades estadísticas de los elementos extraídos o bien extraer la regla de colocación que describa la textura. Esta última aproximación podría mencionarse donde se propone la extracción de tokens usando las propiedades de la partición de Voronoi de una imagen. En cuanto a la segunda aproximación se repasan algunos métodos estructurales.

- Métodos basados en modelos

Se basan en la construcción de un modelo cuyos parámetros estimados describirían las cualidades de la textura. Esta familia incluye los campos aleatorios de Markov así como los fractales.

- Campos aleatorios de Markov

  Los métodos que utilizan los campos aleatorios de Markov gaussianos caracterizan la relación estadística entre un píxel y sus vecinos. El modelo estocástico resultante consta de un número de parámetros de igual al tamaño de la máscara de la vecindad. Los parámetros se pueden estimar mediante un algoritmo de mínimos cuadrados sobre cada posición de la máscara de la imagen. Una máscara que es bastante habitual es una máscara simétrica.

- Fractales

  El análisis de texturas basado en fractales se basa en lal correlación existente entre la dimensión fractal de una textura y su 'tosquedad' (coarseness). La descripción fractal de texturas se basa en gran medida en la determinación de la dimensión fractal. La propiedad de auto-similaridad implica que la dimensión fractal de una imagen es independiente de la escala. Un conjunto acotado A es auto-similar, si es la unión e Nr copias no solapadas de un conjunto similar a A, pero escalado por un factor r.

- Métodos basados en tratamiento de la señal

Esta familia de métodos se basa en representar la imagen en otro dominio, de forma que en este nuevo dominio se facilite la extracción de determinadas características. Muchas de las técnicas de este apartado consisten en calcular ciertas características de las imágenes tras haber sido filtradas, además algunas se apoyan en el hecho de que una primera etapa, el cerebro humano lleva a cabo un análisis frecuencial de la imagen.

- Filtros en el dominio espacial

  Dentro de esta categoría se incluyen desde métodos que miden la densidad de bordes a través de la magnitud de las respuestas de máscaras de Robert y laplaciana hasta métodos bastados en momentos espaciales y otros obtenidos mediante filtros espaciales y operadores no lineales, como la energía textural de Law's.

- Filtros en el dominio de Fourier

  Siguiendo los resultados psicovisuales se han desarrollado sistemas de análisis de texturas de filtrado en el dominio de Fourier para obtener distintas características. Cada filtro es selectivo tanto en frecuencia como en orientación.

- Filtros Gabor y ondículas (wavelets)

  La transformada de Fourier proporciona un análisis de la frecuencia global. Hay veces que es más apropiado que el análisis esté localizado en el dominio espacial. Una forma clásica de conseguirlo es a traés de la transformada de Fourier ventana. Cada filtro de Gabor es un filtro

pasabanda elíptico y orientado. Definen un muestreo del espacio frecuencial. Generan un vector de características de textura para cada píxel. Se pueden definir métodos de reconocimiento estadístico de patrones sobre estas características.

- Métodos basados en correlación y espectro de potencia

  Dado un patrón definido estudiar los picos de la correlación entre la imagen y el patrón. Sensible a la escala, rotación, ocultaciones parciales.

3.6 Tipos de clasificador

3.6.1 Métodos basados en distancias

El razonamiento basado en casos (CBR) es un tipo de algoritmo de aprendizaje automático basado en instancias que está relacionado con el aprendizaje basado en distancias. Este método supone que problemas similares tienen soluciones similares. En el aprendizaje automático es muy común utilizar la similitud entre objetos y existen muchos métodos que se fundamentan en establecer si un nuevo objeto es similar a uno previamente conocido. Una manera de hacerlo es cuantificando la similitud (o disimilitud) entre dos objetos por medio de una medida de distancia.

Una de las principales ventajas de los métodos basados en distancias es que el algoritmo se puede ajustar para un problema específico por medio de la definición de una distancia adecuada; es decir, la distancia es un parámetro del algoritmo de aprendizaje. Igualmente, un algoritmo puede ser usado para cualquier representación de datos si una función de distancia se define sobre él.

Las funciones de distancia más comunes utilizadas para representaciones basadas en truplas son la distancia Euclídea, la distancia de Manhattan, la distancia de Chebychev, la distancia Minkowski y la distancia de Mahalanobis.

Formalmente, definimos estas distancias de la siguiente manera:
Sea x una instancia arbitrariamente descrita por el vector de características:

$$\langle a_1(x), a_2(x), \dots, a_n(x) \rangle$$

donde $a_r(x)$ denota el valor de r-ésimo atributo de la instancia x; entonces la distancia entre dos instancias x_i y x_j, es definida por $d(x_i, x_j)$, para cada una de las siguientes distancias:

- Distancia Euclídea

$$d(x_i, x_j) = \sqrt{\sum_{i=1}^{n} \left(a_r(x_i) - a_r(x_j)\right)^2}$$

- Distancia de Manhattan

$$d(x_i, x_j) = \sum_{i=1}^{n} |a_r(x_i) - a_r(x_j)|$$

- Distancia de Chebychev

$$d(x_i, x_j) = \max_{i=1..n} |a_r(x_i) - a_r(x_j)|$$

- Distancia de Minkowski

$$d(x_i, x_j) = \left(\sum_{i=1}^{n} |a_r(x_i) - a_r(x_j)|^p \right)^{1/p}$$

- Distancia de Mahalanobis

$$d(x_i, x_j) = \left(\left(a_r(x_i) - a_r(x_j)\right)^t S^{-1} \left(a_r(x_i) - a_r(x_j)\right) \right)$$

La distancia de Minkowski es una generalización de las distancias Euclídea, Manhattan y Chebychev, donde un parámetro p debe ser definido. Si p = 1, es la distancia de Manhattan, si p = 2, es la distancia Euclídea y finalmente si p = ∞, es la distancia de Chebychev. Adicionalmente, la distancia Euclídea es un caso particular de la distancia de Mahalanobis: en la distancia Euclídea no se tiene en cuenta la correlación entre los atributos.

Los métodos basados en distancias consideran que cada ejemplo corresponde a un punto en el espacio métrico y el rendimiento de sus predicciones, mediante la comparación de la proximidad, difieren, en gran parte, en la manera en que son elegidos los ejemplos. A continuación se hace una breve descripción de los métodos más comunes; el k-vecinos más cercanos, el discriminante de Fisher y el aprendizaje por cuantificación vectorial son métodos supervisados y la agrupación jerárquica y k-medias son métodos no supervisados.

3.6.1.1 Clasificador de mínima distancia

Se ha optado por el clasificador más sencillo para validar los resultados ya que su funcionamiento será tanto más correcto cuanto más separados se encuentren los clusters entre s. De esta manera se está positivizando el distanciamiento entre las categorías de nubes.

Sea E un conjunto dotado de una (pseudo)métrica $d : E \times E \longrightarrow R$. Supongamos que se dan m prototipos $p_1, p_2, \ldots, p_m \in E$, tales que $p_i \in c_i$ representa a la clase c_i ($i = 1; 2;$;m), es decir, un prototipo por clase. Una clasificación por distancia mínima se define para cada $x \in E$ como $x \in c_i \Leftrightarrow d(x, p_i) \leq d(x, p_j); j = 1, 2, \ldots, m$ donde d es la (pseudo)métrica definida en E.

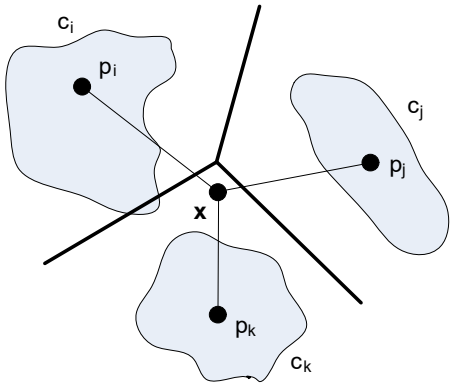


Figura 12. Clasificador de mínima distancia

En este tipo de clasificador, la fase de aprendizaje consiste únicamente en la elección de un buen representante o prototipo de cada clase en el conjunto de prototipos que pertenecen a la misma. Normalmente el representante elegido suele ser aquel que se encuentra más centrado dentro de la distribución de los prototipos en la clase. La Figura 12 muestra la clasificación de un punto en un espacio con tres clases diferentes.

3.6.1.2 Método kNN

También llamado método de k-vecinos más cercanos. Se trata de uno de los algoritmos más analizados en aprendizaje automático. Este algoritmo permite aproximar funciones de salida para valores discretos o continuos. Asume que las instancias corresponden a un punto en un espacio métrico n-dimensional. El valor de la función de salida para un nuevo ejemplo es estimado de los valores conocidos de los k ejemplos de entrenamiento más cercanos. La Figura 13 muestra el funcionamiento del clasificador kNN.

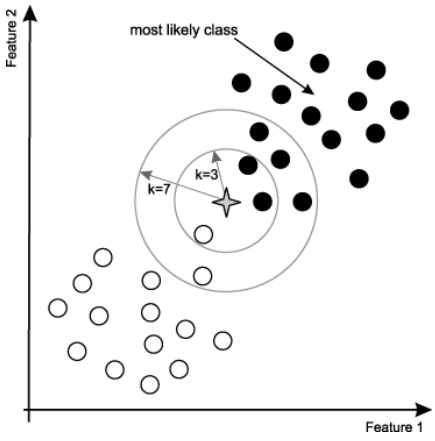


Figura 13. Clasificador kNN

3.6.1.3 K-medias

Los métodos de particionamiento son otro tipo importante de técnicas de agrupamiento. Los grupos son mejorados gradualmente de acuerdo con una función de optimización.

Entre ellos el más conocido es k-medias. Inicialmente un conjunto de centroides son elegidos al azar; luego, los ejemplos son asignados a su centroide más cercano formando grupos y el centriode de los grupos nuevos es calculado. Este proceso se repite hasta que los centroides no cambien. Aunque este algoritmo ha tenido éxito en aplicaciones industriales y científicas, tiene algunos inconvenientes; el más significativo es que su desempeño depende fuertemente de los supuestos iniciales y del número de centroides propuestos. La Figura 14 muestra la construcción del clasificador kNN.

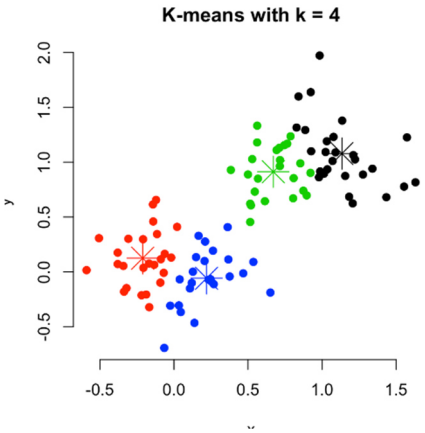


Figura 14. Clasificador K-Medias

3.6.1.4 Discriminante de Fisher

Cada clase es representada por medio un centroide que es un punto que minimiza la suma de la distancia para los elementos que pertenecen a una clase; un nuevo ejemplo es marcado de acuerdo con la etiqueta de su centroide más cercano. Este método es ansioso y los ejemplos son removidos de la memoria una vez que el modelo (conjunto de centroides) haya sido obtenido. Dado que el espacio de representación está compuesto por truplas de datos nominales o numéricos, se puede derivar un modelo más expresivo. Los centriodes de las clases adyacentes son unidos por líneas rectas y la mediana de estas líneas son las reglas discriminantes.

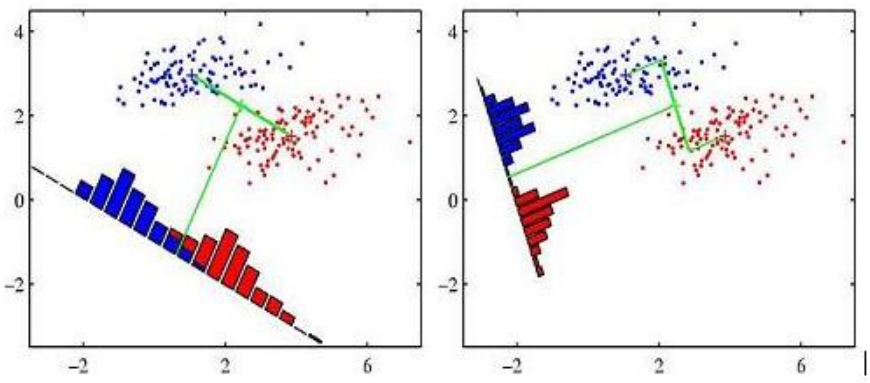


Figura 15. Clasificador mediante discriminante de Fisher

3.6.1.5 Aprendizaje por cuantificación vectorial (LVQ)

El modelo aprendido es una colección de prototipos donde un ejemplo es clasificado de acuerdo con la proximidad a estos prototipos. Inicialmente los prototipos son elegidos al azar, y sus posiciones se actualizan hasta que se alcanza un umbral. Este proceso se realiza de manera iterativa. Primero, un ejemplo del conjunto de entrenamiento es elegido y el prototipo más cercano es determinado. Dependiendo de las etiquetas, tanto del ejemplo como las del prototipo, la posición del prototipo cambia. Si las ambas etiquetas emparejan, el prototipo se acerca más al ejemplo.

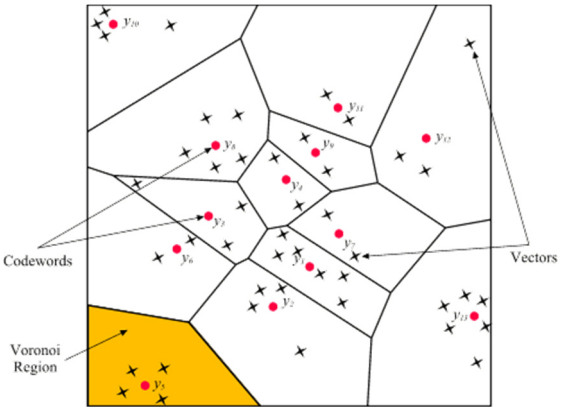


Figura 16. Clasificador por cuantificación vectorial

3.6.1.6 Agrupamiento jerárquico

Este algoritmo se basa en la construcción de un árbol en el que las hojas son los elementos del conjunto de ejemplos, el resto son nodos con subconjuntos de ejemplos que pueden ser utilizados como particionamiento del espacio. Este árbol de grupos es también llamado dendrograma. Existen dos métodos para construir el árbol:

1) Aglomerativo. El árbol se va construyendo empezando por las hojas, hasta llegar a la raíz. Inicialmente, cada ejemplo es un grupo y se van aglomerando los grupos para formar conjuntos más numerosos hasta la raíz, que contiene todos los ejemplos.

2) Desaglomerativo. Se parte de la raíz, que es un solo grupo conteniendo a todos los ejemplos, y se hacen divisiones paulatinas hasta llegar a las hojas que representa a la situaci´on en que cada ejemplo es un grupo.

La forma en que los grupos son divididos o fusionados está guiada por el principio en que los elementos más cercanos deben permanecer en el mismo grupo. En consecuencia, es necesario conocer si dos grupos son cercanos. Teniendo en cuenta que la función de distancia es definida solo para un par de elementos, es necesario extenderla para medir la distancia entre grupos.

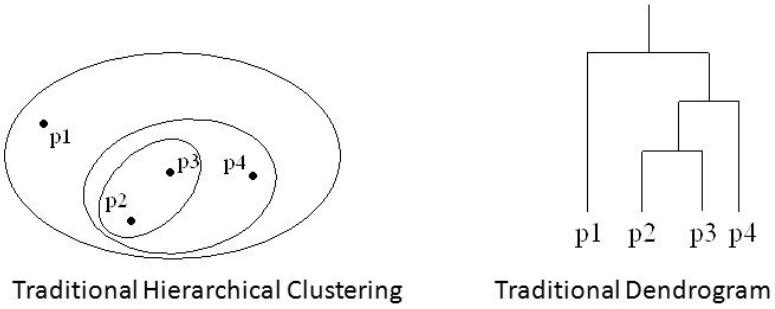


Figura 17. Clasificador mediante agrupamiento jerárquico

3.6.2 Medidas de similaridad para clustering de píxel

3.6.2.1 Ángulo espectral

El ángulo espectral es la extensión d-dimensional del ángulo 2D. Si se consideran los píxeles como puntos de un espacio d-dimensional y si, para cada punto, se dibuja un vector al origen, entonces el ángulo espectral mide el ángulo entre parejas de tales vectores. Pequeños valores del ángulo espectral corresponden a buenas similaridad entre píxeles.

Este método computa el ángulo espectral entre cada píxel y un espectro objetivo. Nótese que el ángulo espectral no es sensible a la diferencia entre iluminaciones de píxeles; esto es una importante desventaja dado que píxeles con diferentes intensidades pero buena correlación podrán reagruparse dentro de la misma clase. Desde otro punto de vista, el ángulo espectral es una herramienta importante para la medición de correlaciones lineales entre los espectros de píxeles. No es por tanto posible distinguir entre las correlaciones positivas y negativas.

3.6.2.2 Coeficiente de correlación de Pearson

El coeficiente de correlación de Pearson consigue superar la desventaja de píxeles con diferentes intensidades, ya que considera la media de los datos extraídos. Se define como:

El coeficiente de correlación de Pearson no puede distinguir entre píxeles con espectros similares pero diferentes intensidades.

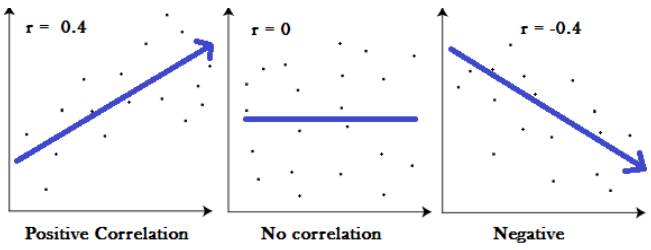


Figura 18. Clasificación mediante Coeficiente de correlación de Pearson

3.6.3 Técnicas de reducción de dimensión

3.6.3.1 Análisis de Componente Principal

El análisis de Componente Principal (PCA) es la mejor técnica de reducción dimensional lineal en términos de error medio cuadrático. Se trata de un método de segundo orden basado en la matriz de covarianza de las variables. En otros campos también es conocido como Descomposición del Valor Singular (SVD), la transformada Karhunen-Loève o transformada Hotelling.

En esencia, PCA busca reducir la dimensión de los datos hallando combinaciones lineales ortogonales (las Componentes Principales) de las variables originales con la mayor varianza.

Su interpretación puede ser compleja ya que, aunque las variables incorreladas se construyen como combinaciones lineales de las variables originales, éstas no se corresponden necesariamente con magnitudes físicas dotadas de significado.

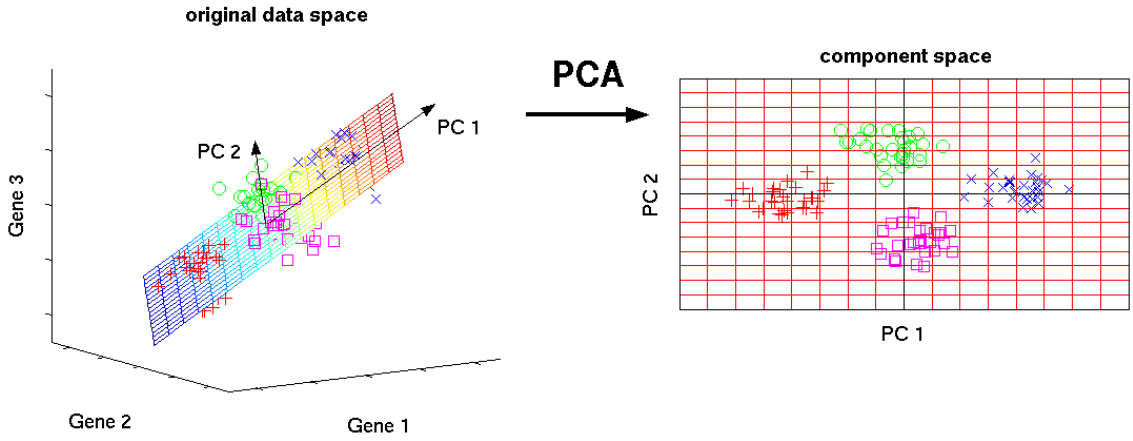


Figura 19. Análisis de Componente Principal

3.6.3.2 Análisis de Factor

El Análisis de Factor (FA), al igual que el PCA, es un método lineal, basado en sumatorios de datos de segundo orden. El FA asume que las variables medidas dependen de factores desconocidos comunes, a menudo no-medibles. El objetivo del FA es descubrir tales relaciones y por tanto pueda usarse para reducir la dimensión de los datos siguiendo el modelo de factor.

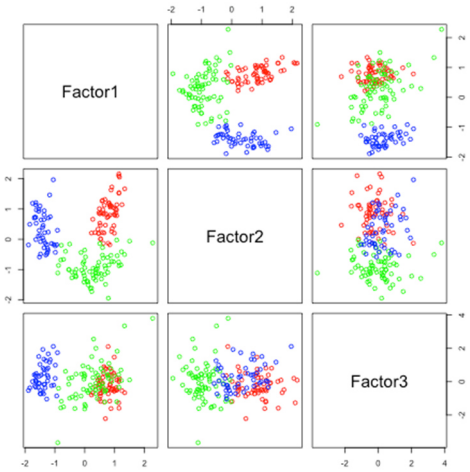


Figura 20. Análisis de Factor

3.6.3.3 Búsqueda de Proyección

La Búsqueda de Proyección (PP, Projection Pursuit) es un método lineal que, a diferencia de PCA y FA, puede incorporar información de mayor de segundo orden, y por tanto es útil para conjuntos de datos no-gausianos. Es computacionalmente más intensivo que los métodos de segundo orden.

Dado un índice de proyección que define el "interés" de una dirección, PP busca las direcciones que optimizan ese índice. Como la distribución gaussiana es la de menor interés (ya que tiene la menor estructura) los índices de proyección generalmente miden algunos aspectos no-gaussianos.

3.6.3.4 Análisis de Componente Independiente

El Análisis de Componente Independiente (ICA) es un método de orden elevado que busca proyecciones lineales, no necesariamente ortogonales unas a otras, que sean tan estadísticamente independientes como sea posible. Mientras que PP sólo tiene en cuenta estadísticas de segundo orden, éste depende de todas las estadísticas de orden superior.

ICA se puede considerar una generalización de conceptos de PCA y PP. Mientras que PCA busca variables no correladas, ICA busca variables independientes. Además ICA libre de ruido es un caso especial de PP. El modelo con ruido ICA es equivalente al modelo FA asumiendo datos no-gausianos.

Las posibilidades de definición de la función objetivo son diversas:

- Máxima probabilidad y entropía de red
- Información mutual y divergencia Kullback-Leibler
- Correlaciones cruzadas no lineales
- PCA no-lineal
- Tensores acumulados de segundo orden

- Negentropía
- Acumulados de orden superior
- Funciones de contraste general

3.6.4 Métodos no lineales

3.6.4.1 Curvas principales

Las curvas principales son curvas suaves que pasan por el "centro" de los conjuntos de datos multidimensionales. Las curvas principales lineales son en realidad componentes principales, mientras que las curvas principales no-lineales generalizan el concepto.

El concepto puede extenderse a superficies de dimensiones superiores pero el procedimiento de estimación se vuelve muy complejo.

3.6.4.2 Escalado multidimensional

El escalado mutidimensional (MDS) se trata de una representación de los elementos en un nuevo espacio que mantiene las proximidades originales de los datos. Las proximidad mide las similaridades entre los elementos y, en general, es una medida de distancia: cuanto más similares sean dos elementos, menos será la distancia entre ellos. Las medidas de distancia populares incluyen la distancia Euclídea (norma L_2), la distancia Manhanttan (L_1, norma absoluta) y la norma máxima. Los resultados de MDS son indeterminados con respecto a la traslación, rotación y reflexión.

MDS se ha usado típicamente para transformar los datos a dos o tres dimensiones y visualizando el resultado para descubrir estructuras ocultas en los datos.

3.6.4.3 Mapas topológicamente continuos

Hay diversos métodos basados en hallar un mapa continuo para transformar datos de altas dimensiones a entramados de bajas dimensiones de dimensiones fijas. Se pueden diferenciar dos tipos principales:

- Mapas auto-organizados de Kohonen (KSOM)
- Redes de densidad, con el mapeado topográfico generativo (GTM) como caso especial.

3.6.4.4 Redes neuronales

Las redes de neuronas artificiales (denominadas habitualmente como RNA) son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso biológico. Se trata de un sistema de interconexión de neuronas que colaboran entre sí para producir un estímulo de salida.

3.6.4.5 Algoritmos genéticos y evolucionarios

Los algoritmos genéticos y evolucionarios (GEA) son técnicas de optimización basados en la teoría darwiniana de la evolución, empleando la selección natural y genética para hallar la mejor solución entre los miembros de una población compitiendo. Hay muchas

referencias de cómo los GEA pueden usarse para la reducción de dimensión. En esencia, dado un conjunto de soluciones candidatas, una función objetivo evalúa la idoneidad de los candidatos y los valores para los parámetros del algoritmo escogido, GEA busca el espacio candidato para el miembro con el ajuste óptimo.

3.7 Segmentación

El objetivo de la segmentación es dividir una imagen en regiones, esto es, descomponer una imagen en sus partes constituyentes separando los objetos de interés entre sí. Cada una de las regiones tienen características homogéneas y constituye generalmente un objeto.

Los métodos de segmentación se pueden agrupar en cuatro clases diferentes:

a) Métodos basados en píxeles, que a su vez pueden ser:

- locales (basadas en las propiedades de los píxeles y su entorno)
- globales (basadas en la información global obtenida, por ejemplo, con el histograma de la imagen).

b) Métodos basados en bordes.

c) Métodos basados en regiones, que utilizan las nociones de homogeneidad y proximidad geométrica, como las técnicas de crecimiento, fusión o división.

d) Métodos basados en modelos. Antes de pasar a estudiar cada uno de estos modelos vamos a ver técnicas para la detección de puntos, rectas, bordes y contornos, como herramientas previas.

4 | Metodología

El proyecto se ha planteado como un proceso de análisis sistemático y exhaustivo de las mejores características para la descripción de cada tipo de nube, mediante un método que obtenga el mayor grado de independencia del tipo de imagen y las condiciones de luminosidad en el momento de la obtención de estas.

Se proponen los siguientes principios sobre los que se cimentará el presente proyecto:

- Colección de imágenes extensa, de forma que se incluya el mayor número de escenarios posibles en el estudio, dentro de una normalidad de funcionamiento, esto es, en los momentos entre el amanecer y el atardecer, pero sin incluir estos.
- Normalización de las características obtenidas, para que todas se encuentren en el mismo entorno.
- Clasificación empleando diferentes clasificadores para evaluar la bondad de cada uno de ellos y poder así seleccionar el que muestre el mejor comportamiento.

La Figura 21 muestra los puntos principales de la metodología seguida en este proyecto.

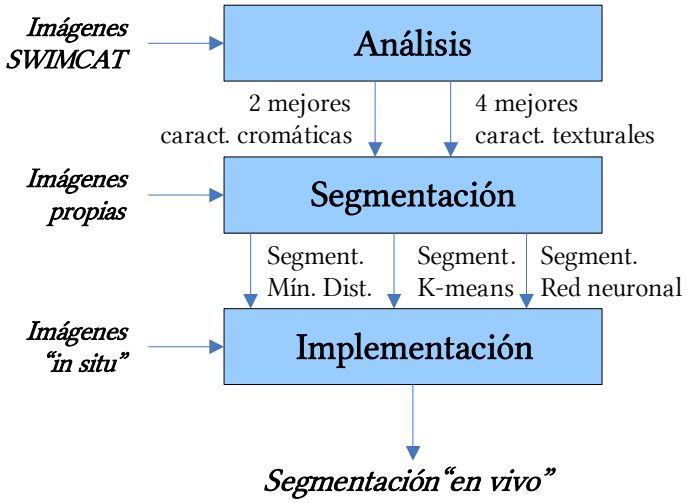


Figura 21. Metodología del proyecto

El Análisis tiene como entrada las imágenes del banco de datos SWIMCAT y busca la obtención de las 2 mejores características cromáticas y las 4 mejores características texturales.

Posteriormente, se emplearán las características seleccionadas durante la fase de Análisis para segmentar diversas imágenes, incluyendo algunas del SWIMCAT pero también otras realizadas con medios propios para verificar un comportamiento coherente ante fuentes diferentes de aquellas con las que se entrenó el algoritmo.

Finalmente, se implementará una de las segmentaciones en un sistema real con análisis de imágenes "en vivo".

4.1 Análisis

La fase de análisis tiene como objeto la selección de las mejores características para la descripción de los diferentes tipos de nube. Una vez establecida la metodología del análisis pueden incorporarse nuevas imágenes, procedimientos o clasificadores para ampliar el campo de búsqueda de características.

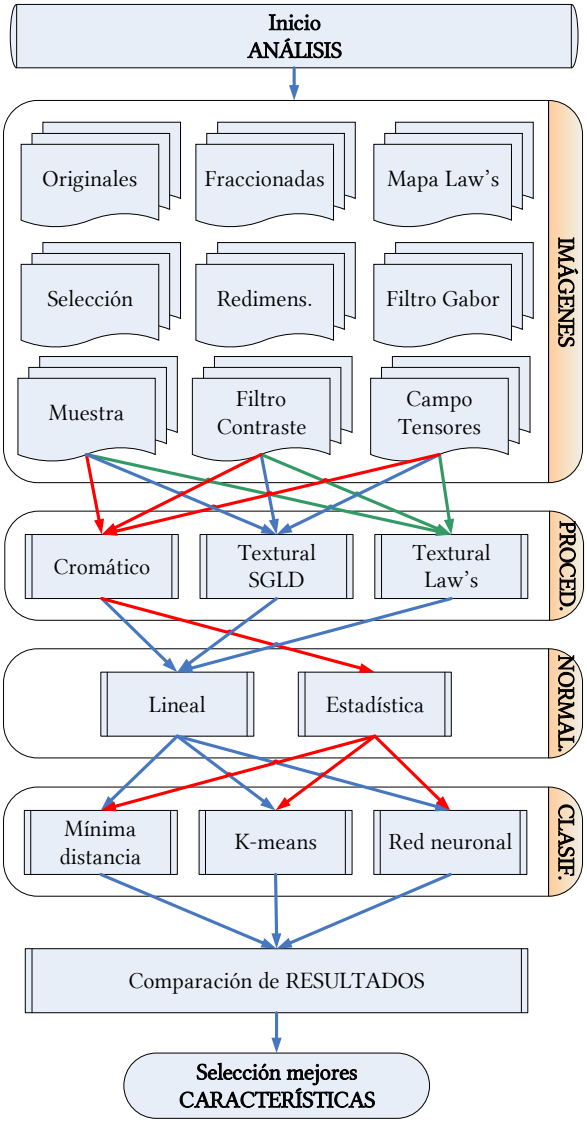


Figura 22. Esquema del Análisis

Este proyecto ha consistido en el análisis de 9 tipos diferentes de imágenes generados a partir de las pertenecientes al banco de imágenes SWIMCAT. Estas imágenes se procesan conforme a tres procedimientos diferentes, según se muestra en la Figura 22, para la obtención de características, que deberán ser normalizadas y clasificadas conforme a tres tipos diferentes de clasificadores.

La sección 5 | de este documento recoge los resultados intermedios y finales del análisis.

4.1.1 Colección de Imágenes

Los estudios presentados en la literatura son frecuentemente difíciles de reproducir debido a la alta dependencia de los resultados hacia la limitada y específica librería de fotografías empleada, así como la variabilidad de las características frente al mismo tipo de nubes.

La calidad y dimensión de la muestra de imágenes debe ser suficientemente extensa y significativa para que los resultados puedan considerarse válidos y reproducibles. Se busca la universalización de los resultados mediante la exhaustividad en el tipo y cantidad muestras en el conjunto de estudio.

Asimismo, debe escogerse un conjunto de tipos de nube que sea capaz de satisfacer la funcionalidad para la que se realiza el estudio. Este estudio tiene como objetivo último la estimación de la altura de las nubes en el cielo por lo que la clasificación de la que se parte responde a aquellas nubes que tengan un rango de altura específico.

Este estudio ha empleado la base de datos SWIMCAT de nubes (Singapore Whole sky IMaging CATegories). Esta colección de fotografías contiene 784 imágenes clasificadas por personal experto del Servicio Meteorológico de Singapur en cinco categorías como las mostradas en la Figura 1 y conforme a lo recogido en la Tabla 1. Este conjunto se considera como muestra válida para un estudio en profundidad.

El conjunto de nubes que forman parte de la base de datos SWIMCAT es suficientemente extensa y se divide en cinco categorías. Sin embargo, esta base de datos no considera todas las categorías definidas por la World Meteorological Organization (WMO) en su Atlas Internacional de Nubes.

Mediante la identificación de estas clases de nubes se podrá realizar una estimación de la altura de las nubes.

4.1.1.1 Imágenes del banco de imágenes SWIMCAT

Las imágenes tienen dimensiones 125x125 píxeles y fueron tomadas usando WAHRSIS, una estación terrestre calibrada de cielo completo. Esta estación se encuentra en la Universidad Tecnológica de Singapur (1.34°N, 103.68°E). Las imágenes se capturaron a lo largo de un período de 17 meses desde enero de 2013 a mayo de 2014. Se seleccionaron basadas en características visuales y se clasificaron conjuntamente con expertos del Servicio Meteorológico de Singapur.

Este proyecto ha considerado cinco tipos de nubes, siguiendo la clasificación SWIMCAT. La Figura 23 recoge una muestra de cada tipo.

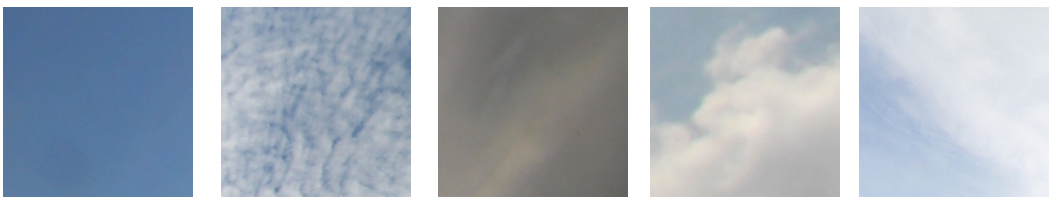


Figura 23. Tipos de nube conforme SWIMCAT

4.1.1.2 Selección de imágenes

Las imágenes SWIMCAT incluyen imperfecciones, tales como reflejos, aviones o gotas de lluvia, tal como se muestra en la Figura 24. Puesto que este proyecto trata de extraer las características de las propias nubes, se ha realizado una selección de las imágenes que excluye las imperfecciones en la medida de lo posible.

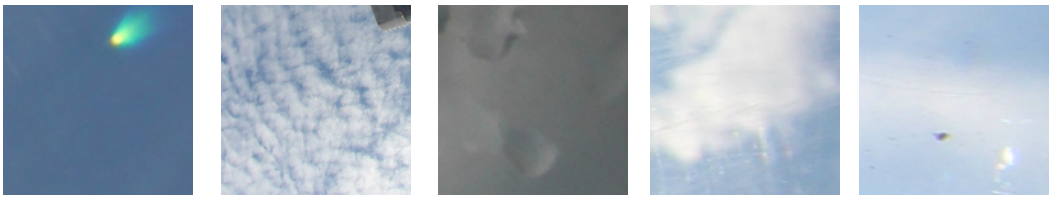


Figura 24. Ejemplos de nubes descartadas

4.1.1.3 Muestra de imágenes

Se ha realizado un muestreo de imágenes que contengan únicamente píxels que pertenezcan a un determinado tipo de nube, excluyendo otras que puedan contener cielo o velo. La Figura 25 muestra la metodología de muestreo.

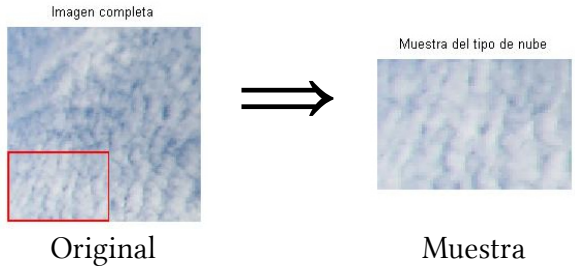


Figura 25. Ejemplo de muestreo de imágenes

4.1.1.4 Imágenes fraccionadas

Se he realizado un fraccionamiento de las imágenes para obtener nuevas imágenes de nubes pertenecientes al mismo tipo y enriquecer el número de imágenes con que se entrena el algoritmo.

Sin embargo, como se puede comprobar en la Figura 26, los resultados no se espera que mejoren los obtenidos con las imágenes originales.

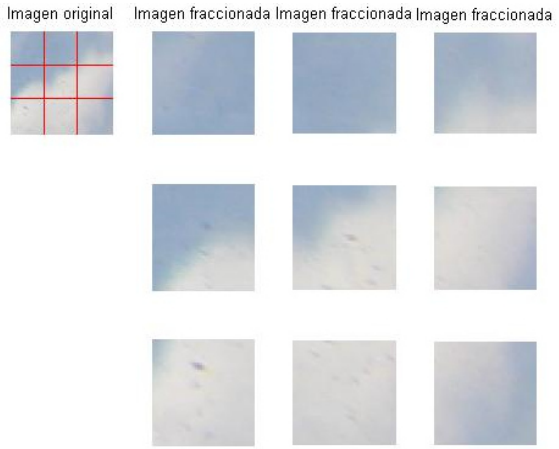


Figura 26. Ejemplo de fraccionamiento de imágenes

4.1.1.5 Imágenes redimensionadas

Así mismo, se ha analizado el efecto de la reducción de la resolución de las imágenes, como muestra la Figura 27. Pese a lo que cabe esperar, se han obtenido resultados ligeramente superiores al análisis de las imágenes originales.

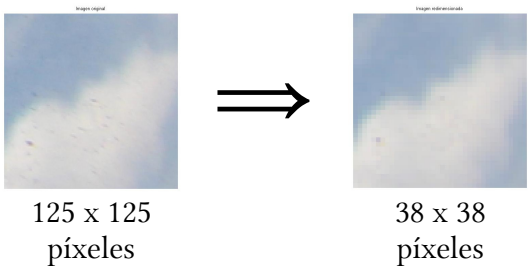


Figura 27. Ejemplo de redimensionamiento de imágenes

4.1.1.6 Imágenes con filtro de contraste

Se ha aplicado un filtro de contraste para estudiar si este tipo de pre-procesamiento pudiera mejorar los resultados obtenidos. La Figura 28 recoge una muestra de cada tipo de nube filtrada.

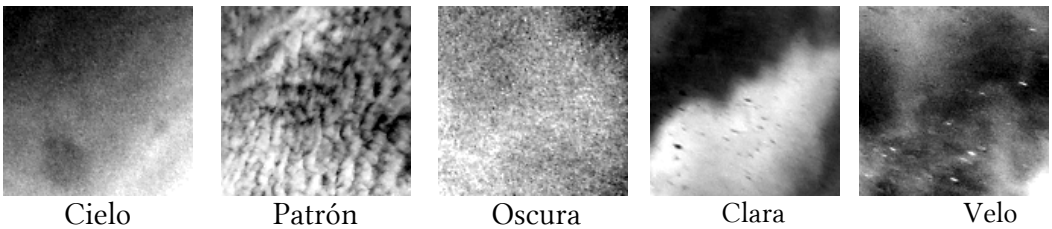


Figura 28. Tipos de nube con filtro de contraste

4.1.1.7 Mapas de Laws

La Figura 29 muestra un mapa modelo de cada uno de los tipos de nubes en estudio.

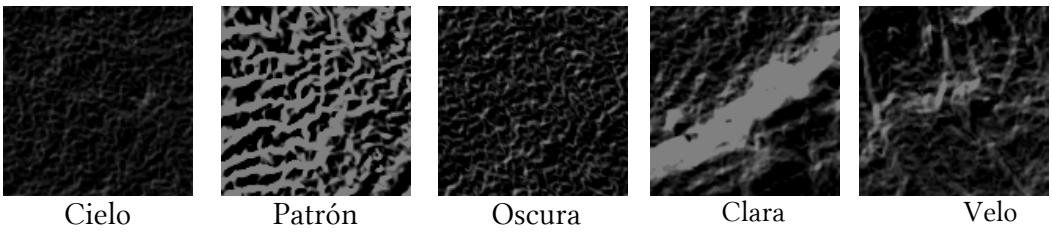


Figura 29. Tipos de nube con filtro de Laws

4.1.1.8 Imágenes con filtro de Gabor

La Figura 30 muestra resultados típicos para cada uno de las cinco clases de nubes en estudio.

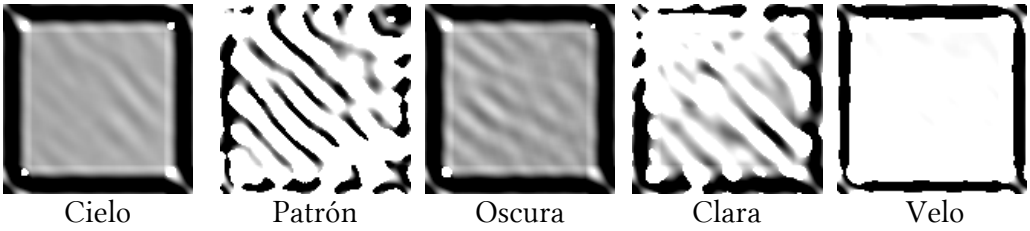


Figura 30. Tipos de nube con filtro de Gabor

4.1.1.9 Campo tensorial

El cálculo del campo tensorial de las imágenes SWIMCAT da lugar a nuevas imágenes con texturas diferentes para cada tipo de nube, como muestra la Figura 31.

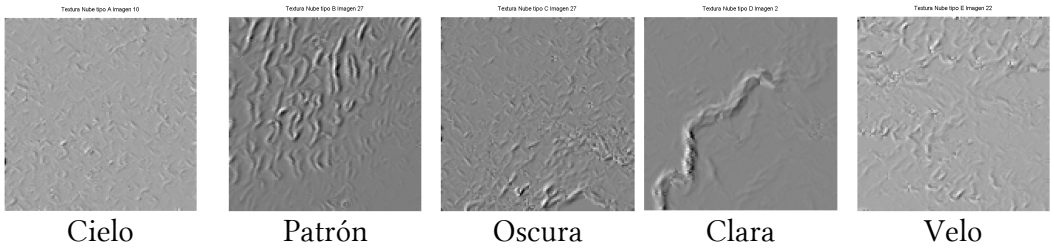


Figura 31. Tipos de nube filtrado mediante campo tensorial

4.1.1.10 Imágenes independientes

Finalmente se han empleado imágenes tomadas desde un dispositivo propio para la verificación "a posteriori" de los resultados obtenidos durante el análisis. La Figura 32 es una muestra de estas imágenes.

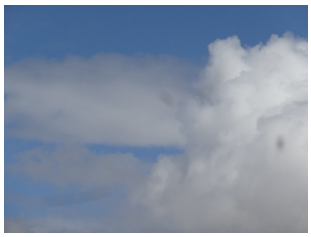


Figura 32. Ejemplo de imagen independiente

4.1.2 Procedimiento

4.1.2.1 Características cromáticas

El objetivo del procedimiento de extracción de características cromáticas consiste en la obtención de un vector de características con tantas líneas como píxeles tiene la imagen y tantas columnas como características se quieran estudiar. En nuestro caso estamos buscando las dos mejores características cromáticas, como se muestra en la Figura 33.

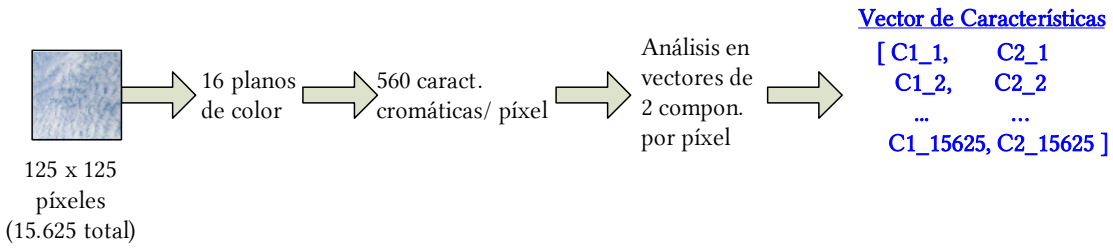


Figura 33. Esquema de extracción de características cromáticas

Las características cromáticas son a menudo consideradas como poco robustas debido que son sensibles a cambios en las condiciones lumínicas y de escala, entre otros aspectos. Sin embargo, hay estudios que emplean estas técnicas para la identificación de nubes con éxito y que se revisara en este estudio con una base de datos de imágenes de gran población y diversidad.

Toda imagen en color puede expresarse en diversos espacios de color de forma que el color se distribuye de forma diferente en cada uno de ellos. Cada uno de los planos de un espacio de color se expresa como escala de grises.

Un estudio exhaustivo debe cubrir todas las posibilidades. Por ello, se emplearan todos los espacios de color descritos y empleados en estudios similares, esto es, se consideraran los siguientes planos de color: RGB, HSV, YIQ, L*a*b* y los derivados (R-B), (R/B), (B-R/B+R) y croma, donde este último se calcula como C = max(R,G,B) -mín(R,G,B). Cada uno de estos 16 planos de color constituye una de las 16 características cromáticas de pixel en

cada imagen. La Tabla 1 muestras los espacios de color descritos en la sección 3.4 de este documento y cuáles se han seleccionado para el presente estudio.

Tabla 1. Espacios de color incluidos en el análisis

Familia	Espacio	Empleado	Características	Observación
Tría tri-cromática	RGB	X	3	
	R-B	X	1	
	R/B	X	1	
	(B-R)/(B+R)	X	1	
Impresión	CMY			
Tonalidad	HSL			
Saturación	HSV	X	3	
	HSI			
	HCI	x	1	(Croma)
Televisión	YIQ	X	3	
	YUV			
	YCbCr			
	YCC			
CIE	CIELab	X	3	
	CIELuv			

A continuación se muestran todas las conversiones seleccionadas para el estudio cromático aplicado a una misma imagen de nubes.

La transformación de una nube en los diferentes espacios y posteriormente mostrados en espacio RGB se muestran en la Figura 34.

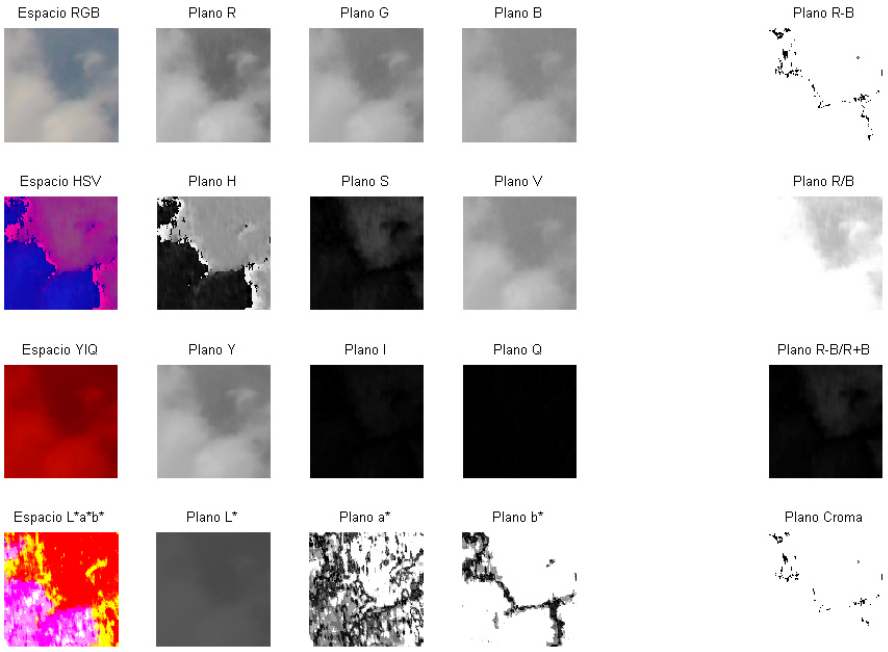


Figura 34. Planos de color empleados

Entre las diferentes técnicas (índice de Pearson, Análisis de Componentes Principales, ...) se ha empleado el análisis de la Distancia Geométrica Euclídea (DGE) a la diagonal del espacio de color como descriptor cromático, pues se busca evitar generalizaciones que puedan distorsionar las características individuales de cada punto de la imagen.

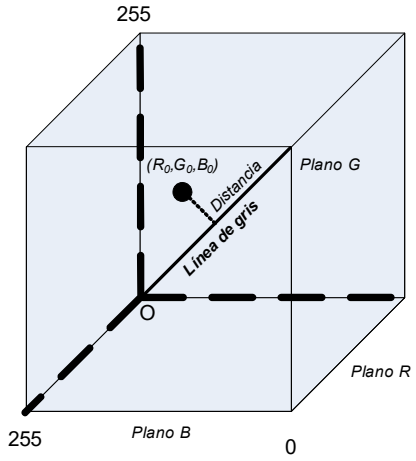


Figura 35. Cubo de truplas de color

El análisis de la DGE consiste en calcular la distancia de un píxel para cada trupla de las 16 características previamente definidas a la diagonal de un cubo formado por el máximo valor en cada uno de estas características, tal como muestra la Figura 35.

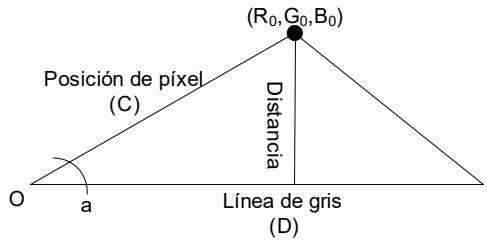


Figura 36. Distancia de un píxel a la línea de gris en una trupla de colores

El cálculo de la distancia se basa en sencillas reglas geométricas, obteniendo la formula siguiente:

$$Dist = |\bar{D}| \sin\left(arccos\left(\frac{|\bar{C} - \bar{D}| - |\bar{D}| - |\bar{C}|}{-2|\bar{D}||\bar{C}|}\right)\right)$$

donde de la Figura 36 se obtiene que los módulos:

$$|\bar{D}| = \sqrt{255^2 + 255^2 + 255^2} = 441{,}67$$

$$|\bar{C}| = \sqrt{R^2 + G^2 + B^2}$$

La distancia de cada pixel a la línea de gris para cada trupla de planos de color puede tratarse como una característica independiente.

Las distancias normalizadas constituirán las características del vector de características que servirán de entrada al clasificador.

$$Vect = [Dist_{RGB}, Dist_{RGH}, \dots, Dist_{HSV}, \dots]$$

Cuando se emplean gran cantidad de datos estadísticamente distribuidos, es usual descartar los datos inferiores al cuartil 25% y superiores al 75 %, sobre todo cuando se emplearán estos datos para generar los prototipos de un clasificador basado en distancias.

Los límites intercuartiles se han calculado como:

- Distancia intercuartil: $IQD = Q_3 - Q_1$
- Límite inferior: $IQI = Q_1 - 1{,}5 \cdot IQD$
- Límite superior: $IQS = Q_3 + 1{,}5 \cdot IQD$

En este estudio se han considerado las 560 truplas de características cromáticas posibles para una muestra de 213.600 píxeles extraídos de las imágenes de la base de datos SWIMCAT. Por tanto, cada trupla supone una característica diferente en el clasificador de mínima distancia. Agrupando varios pares de truplas se mejoran los resultados de clasificación. La Figura 5 muestra las regiones de clasificación generadas por una sola trupla y agrupadas en parejas de truplas.

4.1.2.2 Características texturales – Matriz de Co-ocurrencia

El objetivo del procedimiento de extracción de características texturales consiste en la obtención de un vector de características con una fila por imagen y tantas columnas como características se quieran estudiar. En nuestro caso estamos buscando las cuatro mejores características texturales, como se muestra en la Figura 37.

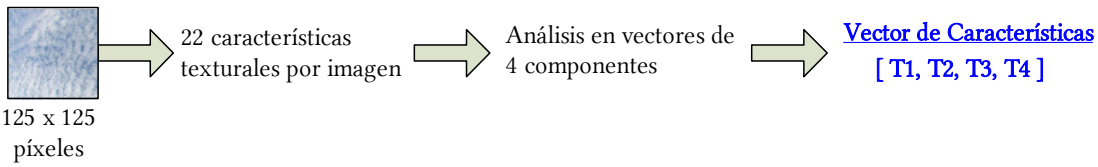


Figura 37. Esquema de extracción de características texturales

Las características que se incluyen en el estudio, aunque no siempre de forma explícita, de todas las características descritas en la sección 3.5 se recogen en la Tabla 2.

Tabla 2. Características incluidas en el análisis

Método	Tipo	Subtipo	Empleado	Características
Estadístico	Primer orden	Media		
		Desviación típica		
		Varianza		
	Segundo orden	Matriz co-ocurrencia	X	22
		F. autocorrelación		
Geométricos	Primitivas			
Modelos	Campos Markov			
	Fractales			
Tratamiento de señal	Dominio tiempo	Momentos		
		Law's	X	9
	Dominio Fourier			
	Gabor y ondículas	Wavelets		
		Filtro Gabor	X	16
Espectro de potencia	Espectros			

Las características texturales son magnitudes estadísticas extraídas del valor del entorno de un punto, a diferencia de las características cromáticas que solo depende del valor tonal de cada píxel de manera individual.

Así, se asume que la información de contexto textural en una imagen se encuentra definida por la relación espacial general o media con la que los tonos de gris tienen unos con otros. Esta información textural se considera adecuadamente especificada por la matriz de frecuencias relativas P_{ij} con la cual dos celdas vecinas separadas una distancia d tienen lugar en la imagen, una con un tono gris i y la otra con un tono de gris j. Tales matrices de frecuencias de tonos de gris con dependencia espacial son función de la relación angular entre las celdas vecinas como de la distancia entre ellas.

Las características texturales que se han considerado son las definidas por Haralick (1973), Soh (1999) y Clausi (2002) que se construyen como Matriz de Coocurrencia de Niveles de Gris (GLCM), presentadas en la Tabla 3 y cuya formulación matemática se recoge en los Anexos 2 y 3.

Tabla 3. Características texturales de Haralick, Soh y Clausi

Nro.	Abrev.	Característica	Haralick, 1973	Soh, 1999	Clausi, 2002
1	CON	Contraste (Inercia)	2	2	5
2	COR	Correlación	3	3	8
3	ENE	Segundo momento angular (Energía)	1	1	2
4	HOM	Momento de la diferencia inversa (Homogeneidad)	5	4	7
5	ACR	Autocorrelación		6	
6	CMC	Coeficiente de máxima correlación	14		
7	RDG	Relevancia de grupo		9	

Nro.	Abrev.	Característica	Haralick, 1973	Soh, 1999	Clausi, 2002
8	TDG	Tono de grupo		8	
9	DIS	Disimilaridad		7	4
10	ENT	Entropía	9	5	3
11	DIN	Diferencia inversa			6
12	PDM	Probabilidad máxima		10	1
13	SCV	Suma de cuadrados: Varianza	4		
14	MDS	Media de la suma	6		
15	VDS	Varianza de la suma	7		
16	ENS	Entropía de la suma	8		
17	VDD	Varianza de la diferencia	10		
18	EDD	Entropía de la diferencia	11		
19	MIC1	Medidas informativas de correlación (1)	12		
20	MIC2	Medidas informativas de correlación (2)	13		
21	INN	Diferencia inversa normalizada			9
22	IDN	Momento de la diferencia inversa normalizada			10

Debe tenerse en cuenta que sólo aquellas características puramente estadísticas son independientes a cambios en el nivel de gris, esto es, no varían con el tono de gris. Son las características consideradas por Clausi en su artículo de 2002.

Como se ha indicado, el resultado durante el tratamiento de textura mediante la matriz de coocurrencia depende de dos parámetros: distancia entre vecinos y ángulo entre vecinos. Este estudio considerara las vecindades de 1 píxel y cuatro ángulos: 0º, 90º, 135º y 270º, como se indica en la Figura 38.

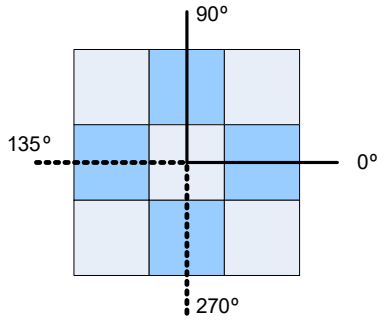


Figura 38. Distancia de niveles de gris entre píxeles

En este estudio se han considerado todas las imágenes de la base de datos, siendo el entorno del punto la imagen completa, de tamaño 192x192 píxeles.

Haralick recoge en su artículo que un componente esencial del marco conceptual de la textura es una medida, o más preciso, cuatro medidas muy relacionadas de las que todas las características derivan. Estas medidas son matrices denominadas matrices de tono de gris de vecinos angulares dependientes del espacio. Estas matrices son función de una relación angular entre las celdas adyacentes así como de la distancia entre ellas.

Este apartado analiza cada característica a una distancia d=1 en sus cuatro direcciones (0º, 45º, 90º y 135º).

De las 22 características analizadas, 7 de ellas son independientes del ángulo:

- Suma de cuadrados: Varianza
- Media de la suma
- Varianza de la suma
- Entropía de la suma
- Autocorrelación
- Tono de grupo (cluster shade)
- Relevancia de grupo (cluster prominence).

Ninguna de estas características es, conforme a lo expuesto por Clausi, independiente del nivel de gris.

En nuestro caso, el cálculo de la matriz de co-ocurrencia de grises da como resultado la generación de una matriz que puede expresarse como imagen en escala de grises, de manera que se pueda apreciar visualmente los diferentes tipos de 'huella' que cada tipo de nube imprime en dicha matriz, tal como se muestra en la Figura 39.

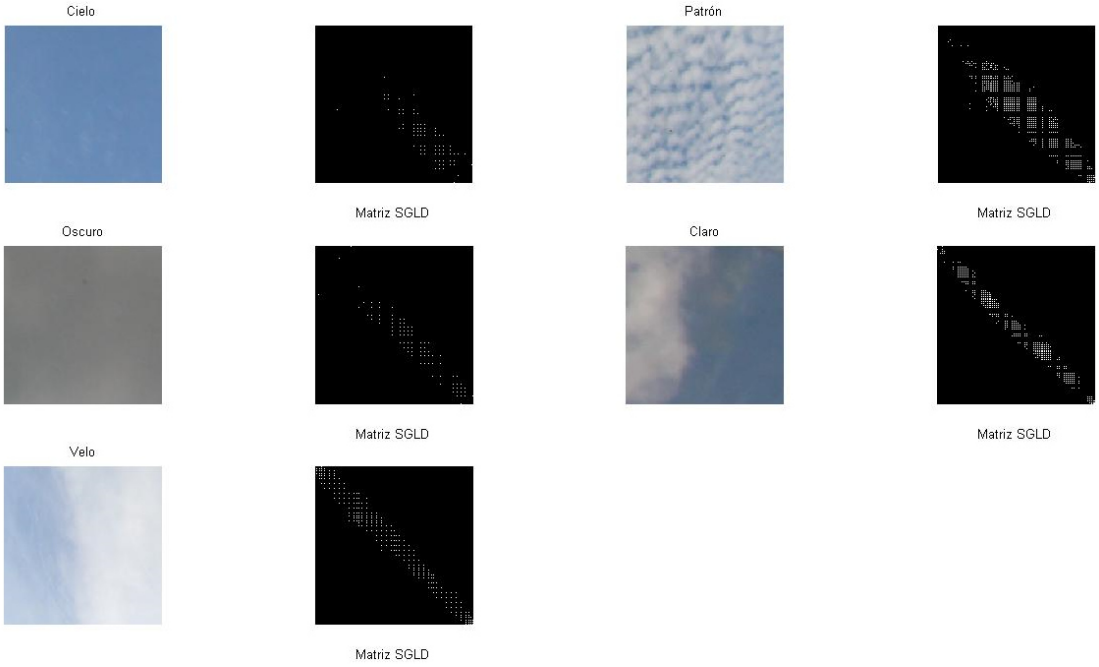


Figura 39. Matrices de co-ocurrencia de gris de los tipos de nube

Así mismo, cada operación que se realice como pre-filtro de la imagen afectará al comportamiento de dicha matriz. Se representan a continuación el efecto de algunas de las operaciones efectuadas.

- Reducción de escala

  Se observa que, al disminuir la escala, la ocurrencia de grises se difumina por lo que hay que tener controlada la escala.

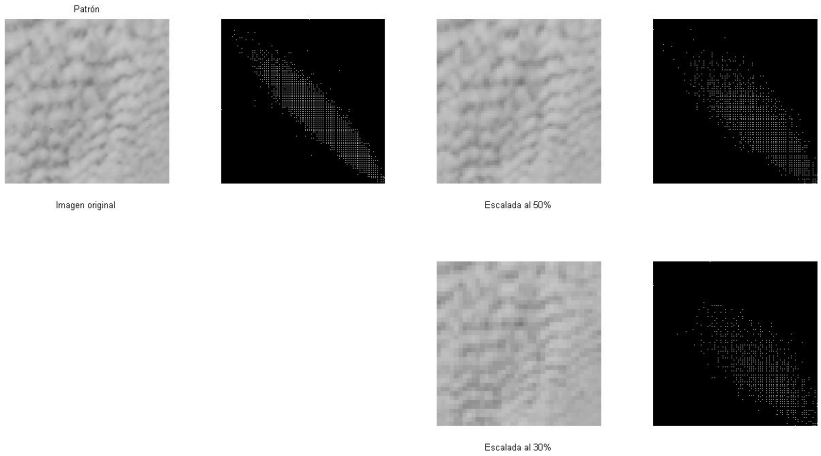


Figura 40. Efecto de la reducción de escala en la matriz GCLM

- Tratamiento de la imagen como canales independientes

  Puesto que la imagen es la superposición de tres canales de color, y puesto que la matriz de co-ocurrencia requiere como paso previo la transformación al espacio de grises, podría tomarse un solo canal de color y realizar el estudio prescindiendo del efecto del resto.

  Esta conversión, sin embargo, no produce ningún cambio en la forma y distribución de la huella. En toda una pérdida de densidad que, a efectos prácticos, no se traduce en cambios en los resultados obtenidos.

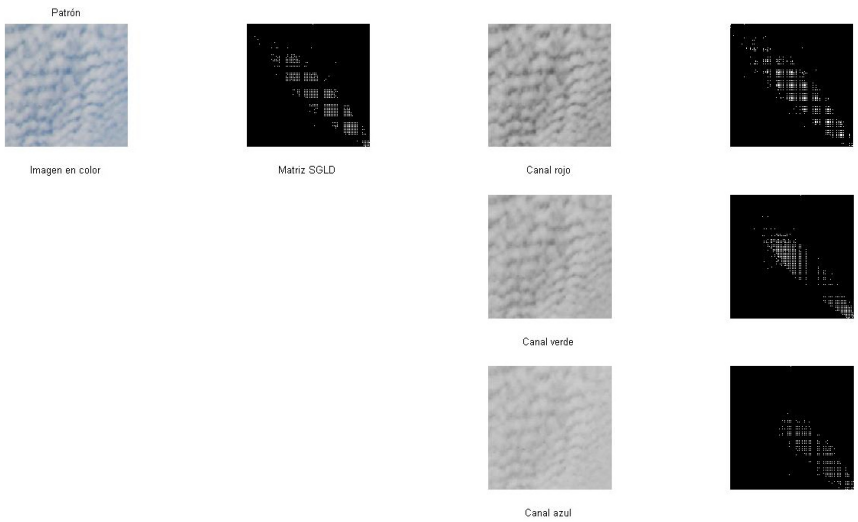


Figura 41. Efecto de los diferentes planos de color en la matriz GLCM

- División de la imagen

  Al dividir la imagen en varias subimágenes, el efecto que se produce es el mismo que la reducción de escala, si bien en otro tipo de nubes con bordes se produce la separación entre cielo y nube, por lo que las huellas pueden cambiar significativamente.

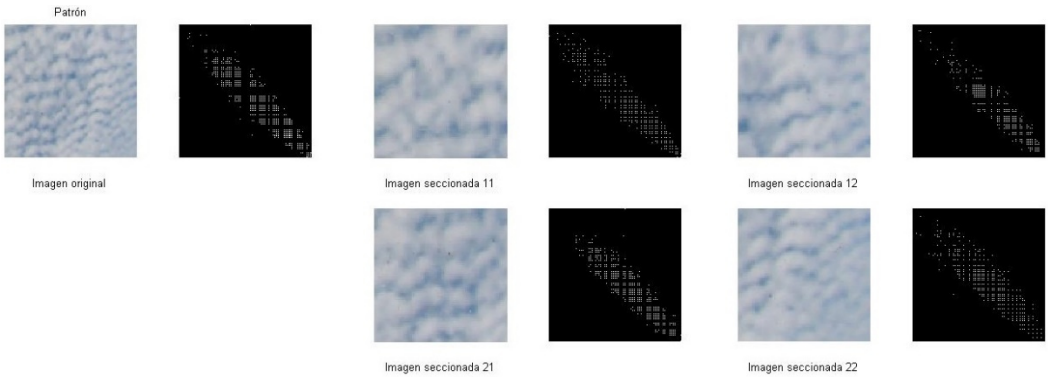


Figura 42. Efecto del fraccionamiento de imagen en la matriz GLCM

4.1.2.3 Características texturales de Laws

Laws observó que ciertos operadores de gradiente tales como los operadores laplaciano y de Sober acentuaban la microestructura subyacente de la textura en una imagen. Esta fue la base para un esquema de extracción de característica basado en una serie de matrices de respuesta impulsional a nivel de píxel obtenido de la combinación de los vectores que se muestran a continuación.

Cada matriz de una dimensión se asociacon una microestructura subyaciente y etiquetada usando un determinado acrónimo. Las matrices se convolucionan con otras matrices de forma combinacional para general un total de 25 máscaras, típicamente etiquetadas como L5L5 para la máscara resultante de la convolución de dos matrices L5.

Nivel L5 = [1 4 6 4 1]

Borde E5 = [-1 -2 0 2 1]

Punto S5 = [-1 0 2 0 1]

Ola W5 = [-1 2 0 -2 1]

Onda R5 = [1 -4 6 -4 1]

Estas máscaras son posteriormente convolucionadas con un campo textural para acentuar su microestructura obteniendo una imagen de la que se mide la energía de las matrices de la microestructura junto con otras variables estadísticas. La medida de la energía de una vecindad centrada en *F(j,k)*, *S(j,k)* se basa en la desviación estándar de la vecindad calculada a partir de la amplitud media de la imagen:

$$S(j,k) = \frac{1}{W^2}\left[\sum_{m=-w}^{w}\sum_{m=-w}^{w}[F(j+m,k+n) - M(j+m,k+n)]^2\right]^{1/2}$$

donde WxW es la vecindad del píxel y la amplitud media de la imagen *M(j,k)* se define como:

$$M(j,k) = \frac{1}{W^2}\sum_{m=-w}^{w}\sum_{m=-w}^{w} F(j+m, k+n).$$

En este documento se han calculado nueve mapas de Laws a partir de estas máscaras, obteniendo para cada imagen de nube, tal como se muestra en la Figura 43.

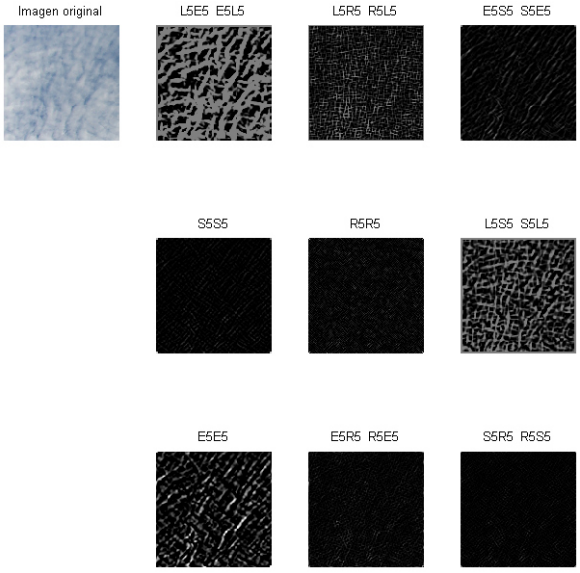


Figura 43. Diferentes mapas de Laws de una imagen

De todos ellos se ha seleccionado el correspondiente a las convoluciones L5E5-E5L5 para su procesamiento dentro de una colección de imágenes.

4.1.2.4 Técnica mixta cromático - textural

Las características texturales se han calculado a partir de 534 muestras de imágenes de la base de datos de SWIMCAT, lo que suponen 2136 características si tenemos en cuenta los cuatro ángulos y extraído cada una de las 22 características texturales, de manera que se obtiene un vector de 2136 x 22 características texturales. La construcción del vector de características se muestra en la Figura 44.

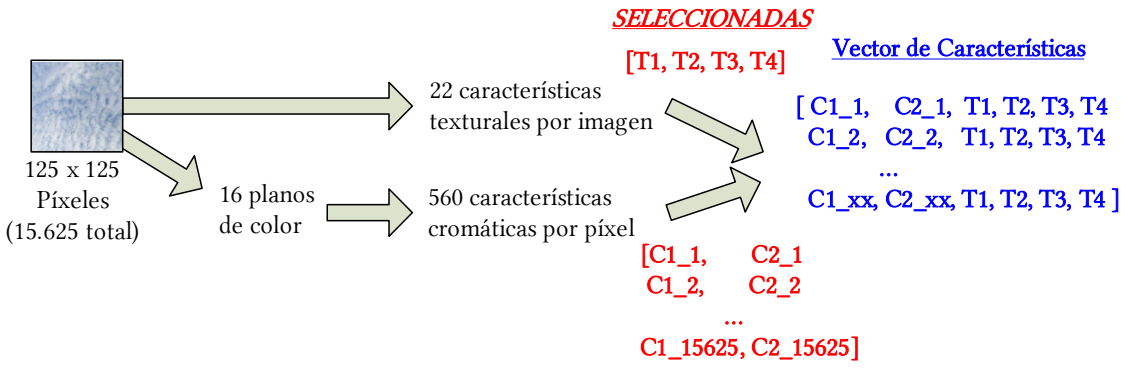


Figura 44. Esquema de extracción de características mixtas

Las características cromáticas se han extraídos de las mismas muestras que las características texturales, a razón de 1.000 píxeles por cada tipo de nube, por lo que se obtiene una muestra final de 213.600 píxeles y constituyendo un vector de características de dimensiones 213600 x 560 (las x columnas). Se puede identificar, por tanto, el valor cromático de cada píxel y el valor correspondiente de la textura asociada.

A partir de esta información se pueden crear matrices de características de tal manera que cada columna corresponde a un píxel y cada la puede significar indistintamente una característica cromática o textural indistintamente. A partir de estas matrices se pueden generar los vectores prototipos de cada que servirán para evaluar cada píxel de cualquier imagen de forma individual.

4.1.3 Normalización

Las características hasta el momento emplean una escala diferente para cada una de ellas. Para poder emplear prototipos y realizar comparaciones coherentes se hace necesario una normalización de las variables.

Por un lado, las características cromáticas se mueven en el intervalo [0,255] por lo que su normalización se puede realizar mediante la asignación al blanco (255) del valor 1, obteniendo un intervalo de [0,1].

Sin embargo, en la búsqueda de la obtención de características robustas e independientes de las condiciones externas durante la toma de imágenes se ha optado por realizar dos tipos de normalización para llevar a cabo una comparación de los resultados.

- Normalización lineal:

$$DistN_{estad} = \frac{Dist}{255}$$

- Normalización estadística:

$$DistN_{estad} = \frac{Dist - \mu}{\sigma^2}$$

4.1.4 Clasificador

La sección 3.6 de este documento describe los clasificadores más habituales. De todos ellos se han seleccionado tres para la realización de este análisis, conforme a la Tabla 4.

Tabla 4. Clasificadores incluidos en el análisis

Método	Tipo	Empleado
Basados en distancia	Mínima distancia	X
	kNN	
	K-medias	X
	Discriminante de Fisher	
	LVQ	
	Agrupamiento jerárquico	
Similaridad para clustering	Ángulo espectral	
	Coef. Correl. Pearson	
Reducción de dimensión	PCA	
	Análisis de Factor	
	Búsqueda de Proyección	
	ICA	
No lineales	Curvas principales	
	Escalado multidimensional	
	KSOM / GTM	
	Redes neuronales	X
	Algoritmos genéticos	

4.1.5 Herramienta de análisis

La herramienta empleada para el análisis es la plataforma matemática MATLAB (R2014a). En esta plataforma se han utilizado funciones de tres tipos, en función de la fuente:

- Funciones propias de MATLAB
- Funciones desarrolladas específicas para este proyecto
- Funciones de terceros.

En las secciones siguientes se presenta la arquitectura de la aplicación de análisis desarrollada y se presentan las funciones en detalle.

4.1.5.1 Arquitectura del software de análisis desarrollado

La arquitectura desarrollada para el análisis de las características cromáticas y texturales y su evaluación mediante la clasificación de las mismas imágenes con las que se entrenan los algoritmos se presenta en la Figura 45.

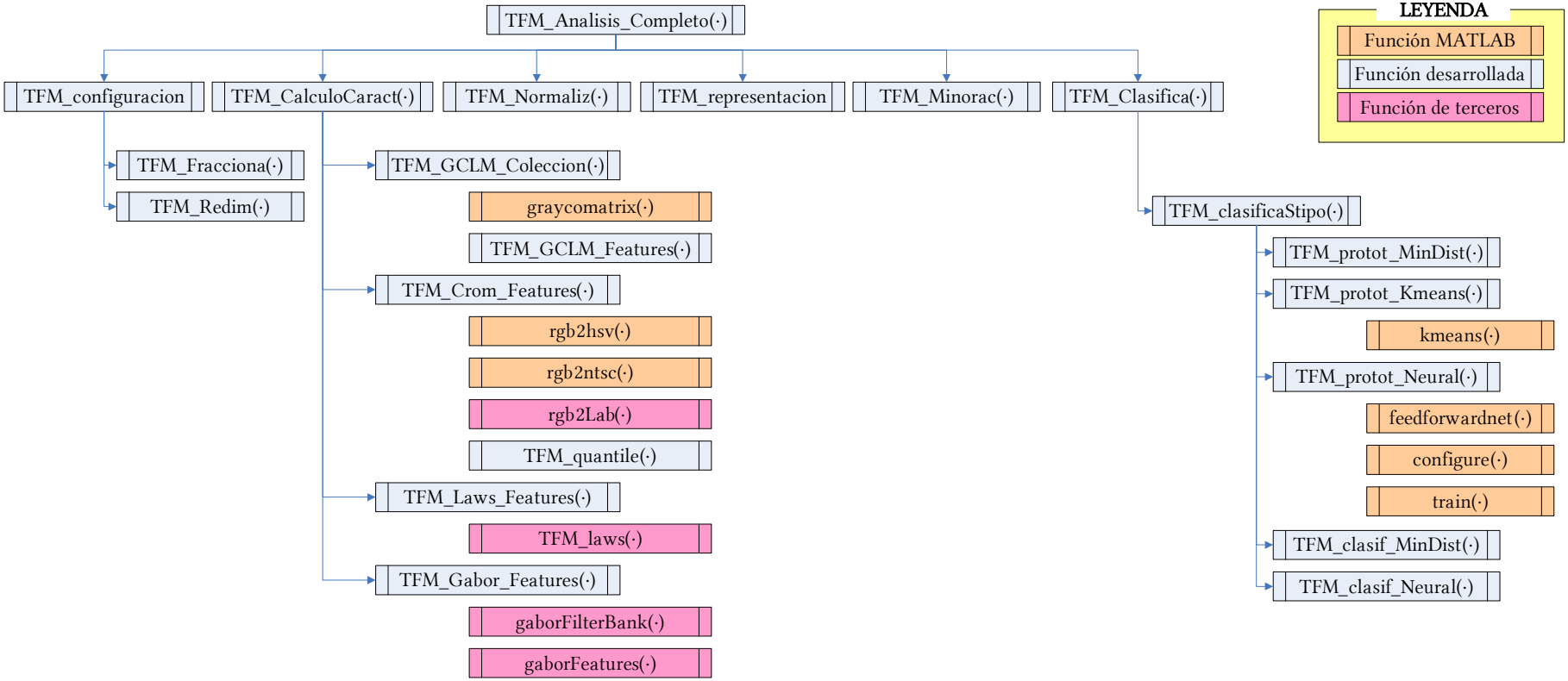


Figura 45. Arquitectura de la aplicación desarrollada para la fase de Análisis

Así mismo, para la fase de Segmentación se han desarrollado una serie de funciones que se ejecutan de manera independiente para, primero, la generación de los prototipos conforme a los emparejamientos más óptimos y posteriormente ejecutar la segmentación de las imágenes.

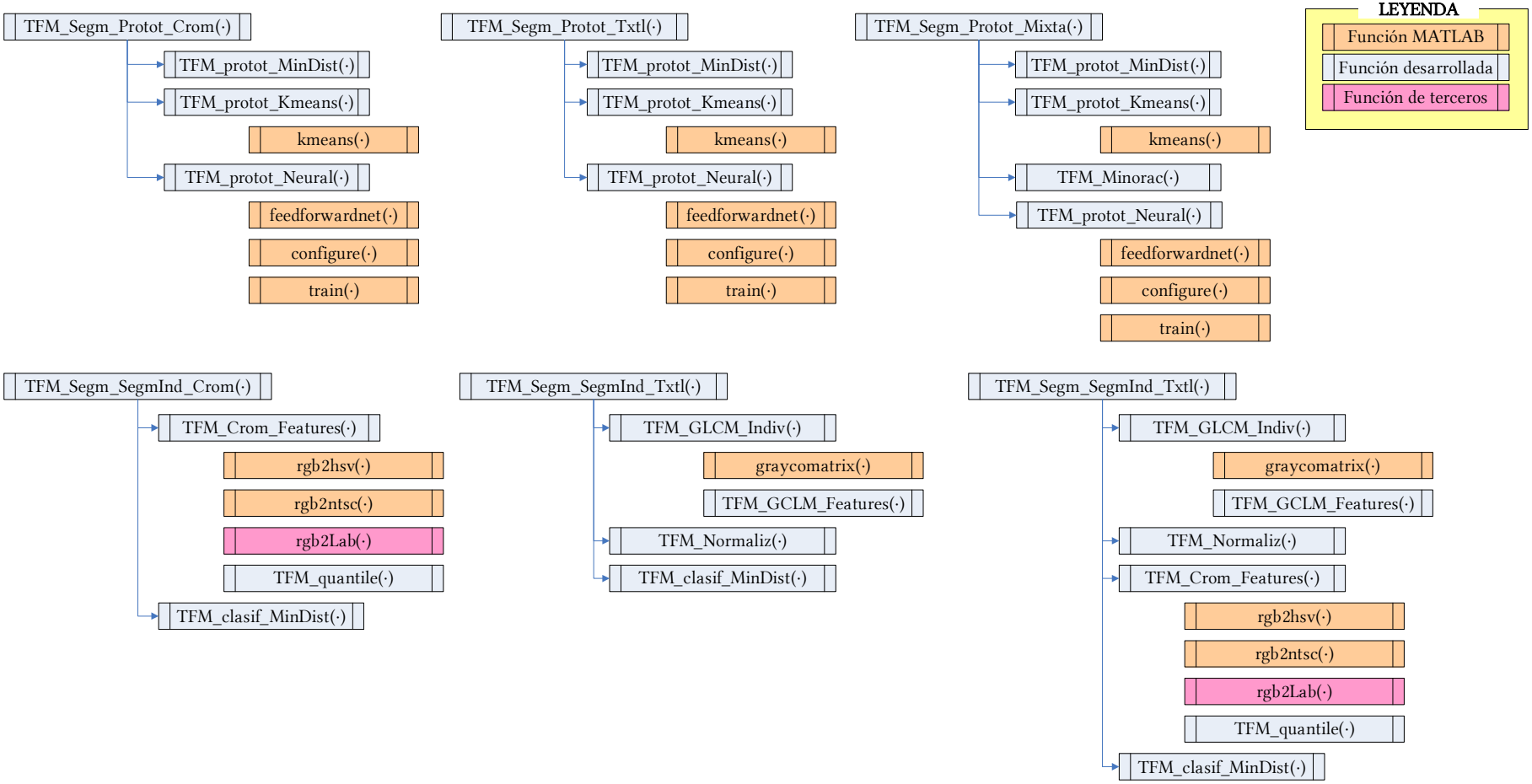


Figura 46. Arquitectura de la aplicación desarrollada para la fase de Segmentación

4.1.5.2 Funciones de MATLAB

El análisis de imágenes es el proceso de extracción de información relevante de las imágenes tales como formas, conteo de objetos, identificación de colores o propiedades de medida de objetos.

Las herramientas de Matlab proporcionan un conjunto de algoritmos estándar de referencia y funciones de visualización para el análisis de imágenes como el análisis estadístico y medidas de propiedades.

El análisis de imágenes se puede dividir en las siguientes categorías:

- Análisis de objetos

  El análisis de objetos de Matlab proporciona detección de bordes, círculos y líneas; traza fronteras; y realiza una descomposición 'quadtree' o de 'árbol cuaternario'.

- Propiedades de regiones e imagen

  Extrae información sobre los objetos de la imagen, tales como número de Euler, histogramas o distancia ponderada de gris.

- Análisis de texturas

  Incluye filtrado mediante la entropía, rango y desviación estándar, así como creación y cálculo de la matriz de co-ocurrencia de niveles de gris.

- Calidad de la imagen

  Incluye medidas de la calidad de imagen mediante: error de mínimos cuadrados, ratio señal de pico-a-ruido, e índice de similaridad estructural.

- Segmentación de la imagen

  Implementa segmentaciones de imagen basado en metodologías como el método de Otsu o mediante pesos de los píxel.

- Transformadas de imagen

  Realiza transformadas de Fourier, Coseno discreto, Radon y 'fan-beam' o 'colimador en abanico'.

- Conversiones entre espacios de color

  La plataforma Matlab representa los colores como valores RGB pero existen otros modelos de color y Matlab transforma las imágenes entre dichos espacios: RGB, L*a*b*, NTSC, XYZ, YCbCr y perfiles ICC.

Las funciones que MATLAB ofrece para el tratamiento de texturas de imágenes son las siguientes:

- Primer orden
  - Entropía de la imagen en escala de grises entropy()
  - Entropía local entropyfilt()
  - Rango local rangefilt()

  - Desviación estándar local stdfilt()

- Segundo orden
  - Matriz de co-ocurrencia de la imagen graycomatrix()
  - Propiedades de la matriz de co-ocurrencia graycoprops()
    - Contraste
    - Correlación
    - Energía
    - Homogeneidad

De todas ellas solamente se ha hecho uso de la función graycomatrix(·) puesto que las características texturales empleadas se han programado aparte, dando lugar a un total de 22 características, frente a las 4 que ofrece MATLAB.

4.1.5.3 Funciones propias

Se han desarrollado 20 funciones específicas para el análisis contenido en este trabajo, tal como se muestra en la Figura 45, si bien se han desarrollada algunas más para el tratamiento de imágenes individuales y el Interfaz de Usuario descritos en la sección 7 | de este documento.

A continuación se presenta una breve descripción de cada una de las funciones propias:

- TFM_Analisis_Completo

  Script que contiene las opciones de análisis básicas y ejecuta de forma secuencial el resto de funciones de Análisis.

- TFM_Configuracion

  Script que carga en el espacio de trabajos las variables auxiliares asociadas a las opciones de análisis básicas configuradas en la función TFM_Analisis_Completo.

  - TFM_Fracciona(·)

    Función que genera y guarda en el disco duro una nueva versión de las imágenes, fraccionadas en el número de imágenes que se especifique.

  - TFM_Redim(·)

    Función que genera y guarda en el disco duro una nueva versión de las imágenes, redimensionas conforme al ratio especificado.

- TFM_CalculoCarac(·)

  Función que determina el tipo de característica escogido para el análisis y ejecuta el proceso correspondiente.

  - TFM_GLCM_Coleccion(·)

    Función que determina el tipo de imágenes escogido y ejecuta el proceso de cálculo de características texturales.

        - TFM_GLCM_Features(·)

          Función de cálculo de las características texturales. Se corresponde a la versión matricial de una función de terceros que emplea métodos iterativos para el cálculo (y por tanto mucho más lento).

    - TFM_Crom_Features(·)

      Función de cálculo de las características cromáticas.

        - TFM_quantile(·)

          Función que filtra las muestras que sobrepasen un 75% y no alcancen el 25% de la distancia la centroide del prototipo.

    - TFM_Laws_Features(·)

      Función que calcula las características de Laws de una imagen.

    - TFM_Gabor_Features(·)

      Función que calcula unas características generadas a partir del filtro de Gabor. Su funcionamiento no es, sin embargo, suficientemente bueno para incluirse en el análisis.

- TFM_Normaliz(·)

  Función de normalización de las características para manejar datos en el intervalo [0,1].

- TFM_representacion

  Script que genera y guarda en disco duro una representación en 1D, ,2D y 3D del emparejamiento de características.

- TFM_Minorac(·)

  Función que minora el número de muestras en el análisis de características cromáticas, ya que en la fase anterior se generan demasiados datos para un tiempo de ejecución aceptable.

- TFM_Clasifica(·)

  Función que ejecuta la función TFM_ClasificaStipo(·) y guarda como archivo Excel los resultados.

    - TFM_ClasificaStipo(·)

      Función que determina el tipo de clasificador configurado en TFM_Analisis_Completo y ejecuta las funciones de prototipado y clasificación correspondientes.

        - TFM_protot_MinDist(·)

          Función de prototipado para un clasificador de Mínima Distancia.

        - TFM_protot_Kmeans(·)

          Función de prototipado para un clasificador K-means.

        - TFM_protot_Neural(·)

          Función de prototipado para un clasificador neuronal.

        - TFM_clasif_MinDist(·)

          Función de clasificación para un clasificador de Mínima Distancia o K-means.

        - TFM_clasif_Neural(·)

          Función de clasificación para un clasificador neuronal.

- TFM_Segm_Protot_Crom(·), TFM_Segm_Protot_Txtl(·), TFM_Segm_Protot_Mixta(·)

  Funciones que generan los prototipos para la segmentación de imágenes conforme al emparejamiento configurable.

- TFM_Segm_SegmIndiv_Crom(·), TFM_Segm_ SegmIndiv _Txtl(·), TFM_Segm_SegmIndiv _Mixta(·)

  Funciones para la segmentación de imágenes individuales o parte de una colección de imágenes.

4.1.5.4 Funciones de terceros

Además de las funciones que tiene MATLAB instaladas por defecto, se han empleado como punto de partida otros archivos que completan las herramientas:

- GLCM – aportación de Avinash Uppuluri:
  - Características GLCM GLCM_Features()
- CIELab – aportación de Yossi Rubner:
  - Conversión al espacio de color CIELab RGB2Lab()
- Gabor – aportación de Mohammad Haghighat:
  - Características de Gabor de una imagen gaborFeatures()
  - Banco de filtros de Gabor gaborFilterBank()
- Laws – aportación de Lukas Tencer:
  - Filtros de imagen de Laws laws()

Estas funciones han sido modificadas para cumplir con los propósitos de este proyecto; en especial la función GLCM_Features(·) que ha sido completamente replanteada para realizar el cálculo de forma matricial.

4.1.6 Optimización de la algoritmia

4.1.6.1 Formulación matricial frente a iterativa

Los tiempos de procesamiento son muy importantes en este tipo de análisis que manejan gran cantidad de información y sometido a un procesamiento iterativo y recurrente. En estos algoritmos cualquier tipo de optimización se traduce en un ahorro importante de tiempos de procesamiento.

Este proyecto está basado en algoritmos tomados de otros autores que han traducido de forma literal la formulación contenida en las definiciones teóricas.

Como parte de este proyecto se ha convertido esta formulación cuya ejecución se basa en bucles de cálculo es una formulación matricial que obtiene el mismo resultado numérico pero consigue un ahorro significativo en tiempo de resolución.

Sirva como ejemplo de optimización el caso de la Energía textural de una imagen:

- Planteamiento iterativo:

$$f_1 = \sum_i \sum_j p(i,j)^2$$

- Planteamiento matricial:

$$f_1 = [1 \quad \dots \quad 1] \cdot GLCM_{norm} \cdot GLCM_{norm} \cdot \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$$

La colección completa de variables texturales en su forma matricial puede encontrarse en el Anexo 2 de este documento.

Como resultado se han obtenido los siguientes tiempos de procesamiento de una imagen:

Tabla 5. Tiempos de procesamiento según su formulación

Algoritmo	Tiempo f. iteración	Tiempo f. matricial
22 carac. texturales considerando 1 ángulo	0.1050 s	3.0845 s
22 carac. texturales considerando 4 ángulos	0.1857 s	5.8927 s

Si se tiene en cuenta que la colección de imágenes SWIMCAT que se está considerando es de 560 imágenes (caso de imágenes seleccionadas), el ahorro tras un análisis completo puede ser de 54 minutos.

4.1.6.2 Muestreo de características cromáticas

Debido al elevado número de píxeles que se están considerando, y puesto que el número de características cromáticas se eleva a 560 cuando se consideran las distancias a la línea de gris tal como se explica en la sección 4.1.2.1 de este documento, se ha realizado un muestreo aleatorio de los píxeles a procesar.

En la generación de los prototipos de clasificación se han considerado únicamente un 5% sobre el total de las muestras. Estas muestras se eligen de manera completamente aleatoria dentro de la misma categoría de nubes, esto es, se mantiene la proporción de muestras para cada tipo de nube.

Los porcentajes de clasificación resultantes serán, por tanto, diferentes cada vez que se analicen las muestras, si bien se espera un comportamiento coherente con porcentajes de clasificación similares.

4.2 Segmentación

Este proyecto tiene como objetivo la segmentación basado en un método mixto que incluye al mismo tiempo una segmentación basado en propiedades locales de píxeles (descriptores cromáticos) y en propiedades basadas en regiones (descriptores texturales), conforme a la clasificación explicada en la sección 3.7 de este documento, si bien se obtienen segmentaciones cromáticas y texturales puras como pasos intermedios del proceso.

Las imágenes se evaluarán empleando los clasificadores de mínima distancia, K-means y redes neuronales y como resultado se obtendrá una matriz con elementos enteros en el entorno de [1,5], que se corresponderán con las cinco clases de nubes en estudio.

Tabla 6. Valores de píxel en la segmentación

Tipo de nube	Valor clasificación	Escala de grises
Negro	-	0,00
Cielo	1	42,67
Patrón	2	85,33
Nubes oscuras	3	128,00
Nubes claras	4	170,67
Velo	5	213,33
Blanco	-	255,00

La Tabla 6 incluye, además de los valores en escala de gris de cada nube, los colores negro y blanco para dar una referencia del tono de gris que aparecerán en las segmentaciones. De esta manera, el cielo tendrá un tono de gris muy oscuro y el velo será muy claro.

Se han implementado dos tipos de segmentación mixta diferentes:

1. Segmentación basada en el procesamiento textural con entornos cuyo centro varía píxel a píxel.

   En este caso, la matriz resultante tendrá unas dimensiones menores que la imagen original debido a que la segmentación textural está basado en regiones y se perderán, por tanto, los bordes de la imagen por no disponer de información completa sobre su clasificación real, como se muestra en la Figura 47. El entorno analizado dependerá de las dimensiones del entorno de región que se configure durante el análisis textural de la imagen.

Figura 47. Entorno textural completo considerado durante la segmentación

En este documento se han considerado entornos de 64x64 píxeles para las imágenes de 125x125 del banco de imágenes SWIMCAT. En este caso se estarían calculando 3.721 matrices GLCM, con lo que esto conlleva de procesamiento y tiempo de ejecución.

2. Segmentación basada en la división de la imagen en entornos separados e independientes entre ellos.

   En este caso, la matriz que se obtiene no presenta la pérdida de los bordes. Su resultado esperado será más impreciso que la segmentación textural completa, pero su ejecución será significativamente más rápida. La Figura 48 representa las áreas texturales consideradas durante la segmentación.

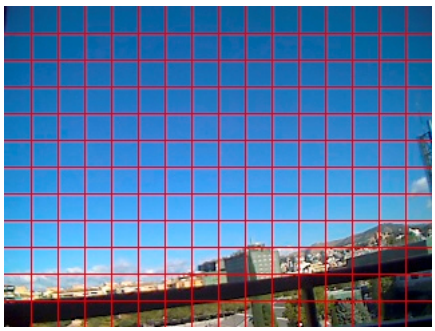


Figura 48. Entorno textural en cuadrícula considerado durante la segmentación

Las imágenes consideradas durante la ejecución de imágenes en vivo tienen dimensiones 320x240 píxeles y se emplean cuadros de 20x20 píxeles, con lo que se procesan únicamente 192 matrices GLCM.

La sección 6 | de este documento recoge los resultados de las diferentes segmentaciones.

4.3 Implementación

4.3.1 Interfaz de usuario

El interfaz de usuario recoge todas las técnicas empleadas durante las fases de Análisis y Segmentación bajo un mismo entorno de ejecución con el objetivo de presentar al usuario un interfaz amigable.

La Figura 49 muestra el "layout" del interfaz de usuario. Se encuentra dividida en 5 zonas, cuatro de las cuales corresponden a la fase de Análisis y la última a la de Segmentación.

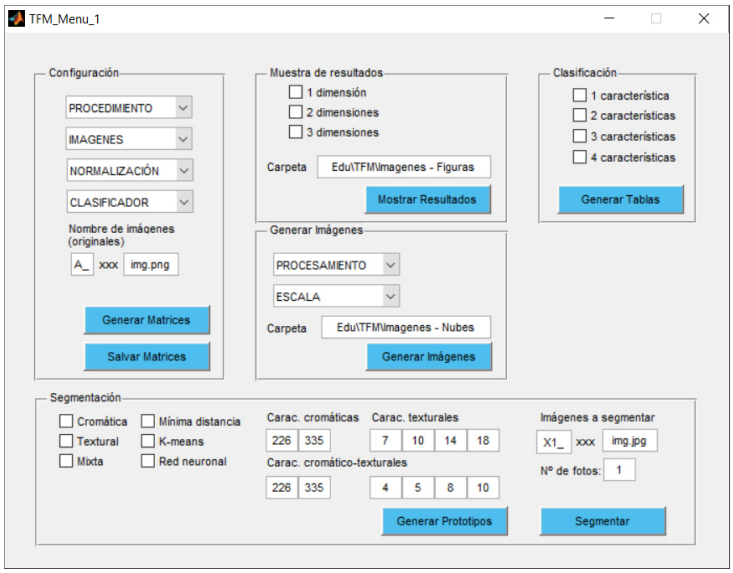


Figura 49. Layout del interfaz de usuario

La sección 7 | de este documento desarrolla los procesos detrás de cada opción propuesta en el interfaz de usuario.

4.3.2 Estación terrestre

La fase final del proyecto consiste en la implementación de los resultados sobre una plataforma real del sistema de identificación automática de nubes.

Se ha optado por el SBC (Single Board Computer) de bajo coste Raspberry Pi 3, cuya potencia de cálculo es limitada pero precisamente por ello supone un reto de optimización de los algoritmos.

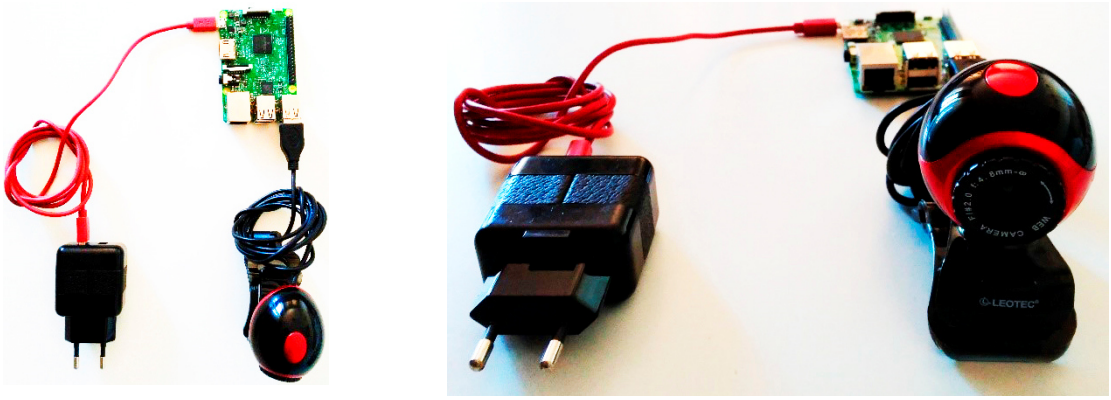
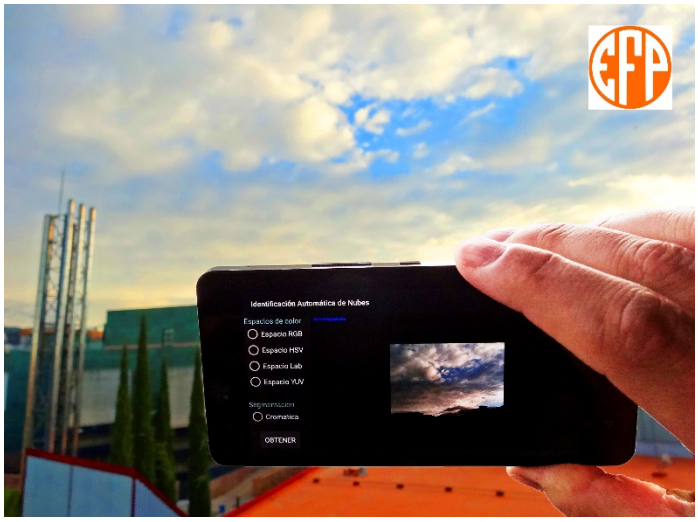


Figura 50. Hardware del Sistema Automático de Identificación de Nubes

Se han implementado las tres formas de segmentación con una resolución reducida para conseguir tiempos de respuesta razonables.

4.3.3 Aplicación móvil



5 | Resultado del Análisis

5.1 Clustering por tipo de característica

5.1.1 Características cromáticas

La aplicación del método de la Distancia Geométrica Euclídea a las 16 características cromáticas definidas anteriormente ofrece 560 combinaciones de distancias posibles, que es el número de características que se han estudiado por cada píxel. Debido al elevado número de variables y la dificultad de segmentar manualmente cada imagen se han tomado muestras significativas de cada tipo de nube para entrenar el clasificador.

En esta investigación se han calculado de forma exhaustiva todas las posibles combinaciones para el clasificador de mínima distancia y se han obtenido tasas de error en la clasificación aceptables para 1 sola dimensión y buenos en caracterizaciones con dos distancias, como se observa en la Tabla 7.

Tabla 7. Tasas de error globales de características cromáticas

Clasificador	1 Característica	2 Características
Mínima distancia	22,86%	10,52%
K-means	30,39%	16,82%
Red neuronal	19,46%	9,32% (**)

(**) *En el caso de las redes neuronales sólo se evaluaron 500 de las 156.520 muestras debido a los elevados tiempos de computación.*

La Figura 5 muestra la distribución de los píxeles caracterizados por lo que se puede observar las distancias que existen entre las cinco clases de nubes. Debe notarse que, debido al solapamiento se obtienen tasas de error altas para las clases B y E.

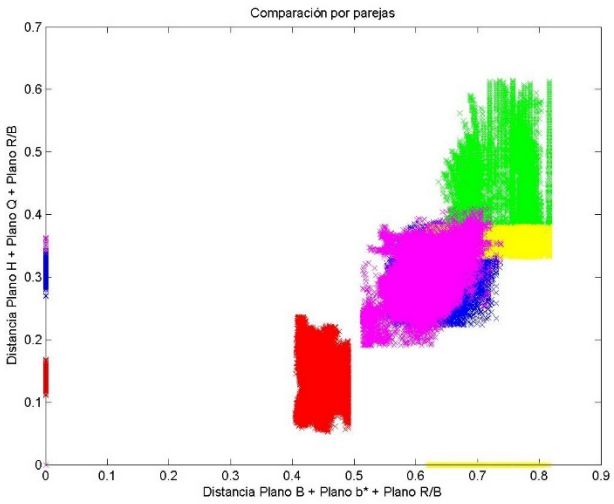


Figura 51. Ejemplo de clustering de características cromáticas

Algunas de las figuras obtenidas apuntan a la existencia de subclases de nubes, lo cual es positivo para clasificaciones más exigentes que requieras de categorías de nubes no incluidas en este estudio.

5.1.2 Características texturales

Las características texturales empleadas son 22 variables que se han estudiado en emparejamientos de hasta cuatro variables, mejorando el rendimiento global con cada nueva característica que se empleaba en el clasificador. La Tabla 4 muestra los resultados para los mejores grupos de dos, tres y cuatro características texturales.

Tabla 8. Tasas de error globales de características texturales

Clasificador	1 Característica	2 Caract.	3 Caract.	4 Caract.
Mínima distancia	39,46%	16,65%	12,23%	10,58%
K-means	36,43%	28,21%	21,21%	19,24%
Red neuronal	32,99%	10,09%	5,09%	4,32%

Se ha representado las nubes de dispersión de cada uno de los cinco tipos de escenarios para cada una de las parejas de parámetros y se ha evaluado los que presentan una mayor segregación y por tanto candidatos, así como aquellos emparejamientos de parámetros linealmente independiente o cercano a esta situación.

De las figuras obtenidas se observa que en la mayoría de los casos existe una dependencia entre estos parámetros ya que la representación en tres dimensiones presenta superficies o líneas en lugar de nubes. Ejemplos de la dependencia de variables se muestran en la Figura 52.

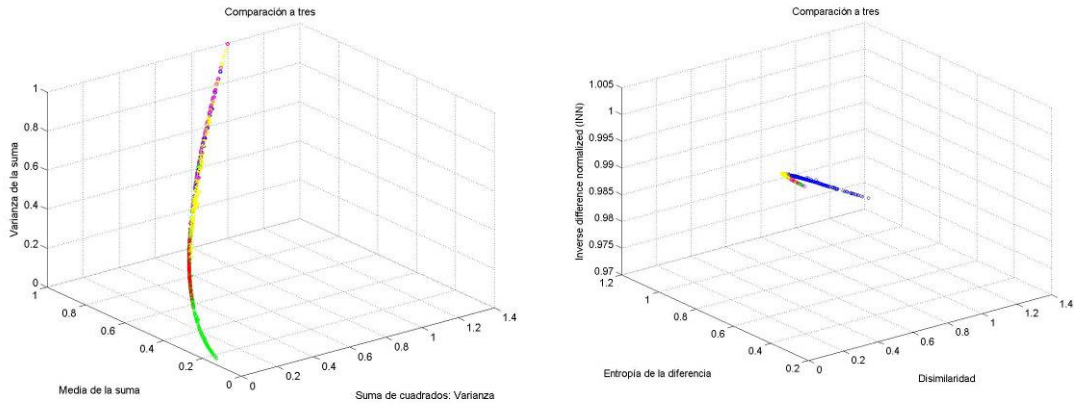


Figura 52. Ejemplo de clustering de variables linealmente dependientes

Por el contrario, otras agrupaciones presentan un gran potencial en cuanto a la segregación de parámetros, como se muestra en la Figura 53.

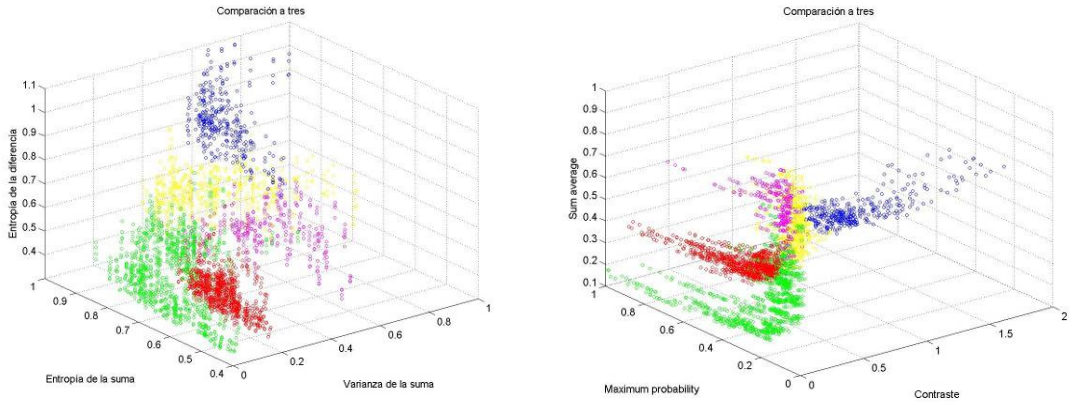


Figura 53. Ejemplo de clustering para segregaciones casi completas

Debido a la relación entre variables se obtienen representación de clustering similares para distintos parámetros.

Finalmente, la Figura 54 representa gráficamente la mejora en el clustering conforme se van introduciendo nuevas características. Esta figura parte de la Entropía de la Suma a la que añade la Varianza de la Suma y finalmente la Entropía de la Diferencia. Se observa que mientras que con una sola característica existe un fuerte acoplamiento entre las clases de nube, el clustering con tres características parecen estar completamente separadas unas clases de otras.

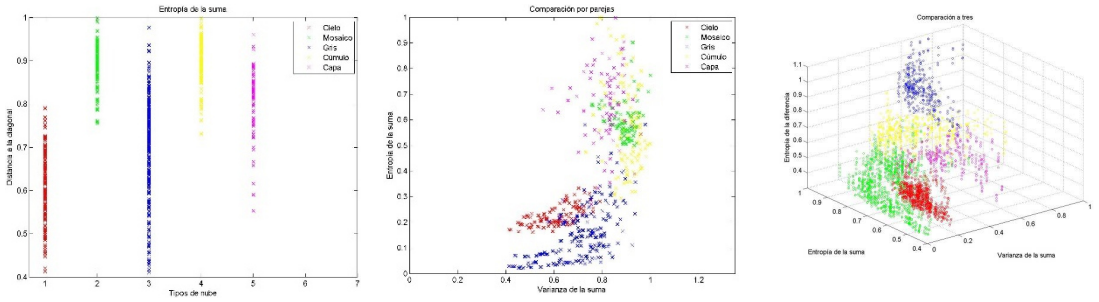


Figura 54. Agregación de características al clustering

5.1.3 Técnica mixta cromático-textural

La técnica mixta cromático-textural se basa en la clasificación de mediante el uso de cuatro características texturas y dos cromáticas. Puesto que las dimensiones del cluster es de 6 características su representación como forma de gráfico no es posible.

Además, debido al elevado número de posibles combinaciones entre características una búsqueda exhaustiva de las características más óptimas requeriría varias semanas de cálculo, por lo que se ha optado por tomar las mejores características individuales y componer el clasificador a partir de éstas.

Con las características escogidas para cada uno de los clasificadores y entrenando las características cromáticas con el 100% de las muestras, en lugar del 5%, los resultados obtenidos mejoran notablemente, como muestras la Tabla 9.

En el caso de las redes neuronales, además, se han empleado 50 neuronas, en lugar de las 25 en la fase de Análisis.

Tabla 9. Tasas de error del clasificador cromático-textural

Clasificador	A	B	C	D	E	Total
Mínima Distancia	0,00 %	5,48 %	0,94%	8,25%	4,46%	3,65%
K-means	0,00 %	43,34 %	1,27%	41,63%	44,02%	23,34%
Red Neuronal	0,00 %	0.38 %	0,08%	0,75%	0,56%	0,35%

5.2 Tasas de error de los clasificadores

5.2.1 Clasificador de mínima distancia

Las tasas de error obtenidas mediante el uso de clasificador de mínima distancia se presentan en la Tabla 10. Contiene también la evaluación de los otros clasificadores para los parámetros óptimos de éste.

Tabla 10. Tasas de error del clasificador de mínima distancia

Procedim.	Selección	Normalización	Car#1	Car#2	Car#3	Car#4	MínDist	K-means	Redes Neur
Cromático	Muestras	Lineal	80	226			**10,52%**	48,25%	-
SGLD	Seleccionadas	Lineal	7	10	14	18	**10,58%**	36,38%	10,85%
SGLD	Seleccionadas	Estadístico	7	10	14	18	**10,67%**	37,01%	10,18%
SGLD	Fraccionadas	Lineal	8	10	14	18	**20,34%**	34,76%	24,26%
SGLD	Redimensionadas	Lineal	7	10	14	18	**8,97%**	38,93%	10,54%
SGLD	Muestras	Lineal	4	6	14	22	**15,72%**		17,40%
SGLD	Mapas Law's	Lineal	1	11	17	22	**65,73%**	75,38%	4,38%
SGLD	Filtro contraste	Lineal	4	10	15	16	**31,89%**	55,80%	29,34%
SGLD	Mapas Gabor	Lineal	6	20	14	21	**33,44%**	35,94%	27,23%
SGLD	Campos tensoria-les	Lineal	8	10	14	13	**55,56%**	68,95%	39,44%
Law's	Seleccionadas	Lineal	2	4	7	8	**33,39%**	44,46%	-

Las características óptimas que se obtienen como resultado del análisis son:

- 7 CPROM Cluster Prominence
- 10 ENTRO Entropía
- 14 SAVGH Media de la suma
- 18 DENTH Entropía de la diferencia

Nótese que las características escogidas hacen óptima la mayoría de los tipos de imagen empleados.

5.2.2 Clasificador K-means

Al igual que en la sección anterior, la Tabla 11 muestra los resultados de clasificación mediante el clasificador K-means. Los resultados son peores que los obtenidos para el clasificador de mínima distancia.

Tabla 11. Tasas de error del clasificador K-means

Procedim.	Selección	Normalización	Car#1	Car#2	Car#3	Car#4	MínDist	K-means	Redes Neur
Cromático	Muestras	Lineal	226	335			12,72%	**16,82%**	-
SGLD	Seleccionadas	Lineal	1	9	13	15	15,71%	**19,24%**	11,88%
SGLD	Seleccionadas	Estadístico	21	15	10	18	16,47%	**20,18%**	13,71%
SGLD	Fraccionadas	Lineal	4	15	16	17	26,57%	**26,95%**	21,45%
SGLD	Redimensionadas	Lineal	8	11	13	18	18,48%	**16,79%**	18,71%
SGLD	Muestras	Lineal	11	14	16	20	18,17%	**22,21%**	15,05%
SGLD	Mapas Law's	Lineal	1	3	5	18	70,97%	**66,90%**	8,14%
SGLD	Filtro contraste	Lineal	7	15	10	18	32,59%	**34,41%**	31,25%
SGLD	Mapas Gabor	Lineal	8	16	17	20	36,56%	**31,56%**	27,23%
SGLD	Campos tensoriales	Lineal	8	7	9	13	60,24%	**59,35%**	42,74%
Law's	Seleccionadas	Lineal	2	6	7	8	44,82%	**38,57%**	-

Las características óptimas que se obtienen como resultado del análisis son:

- 1 CON Contraste
- 9 DIS Disimilaridad
- 13 SVC Suma de cuadrados: Varianza
- 15 VDS Varianza de la suma

5.2.3 Clasificador de Red Neuronal

La Tabla 12 muestras las tasas de error obtenidas para la red neuronal. Sin embargo, las características cromáticas no han podido ser procesadas debido al elevado tiempo de computación.

Tabla 12. Tasas de error del clasificador de red neuronal

Procedim.	Selección	Normalización	Car#1	Car#2	Car#3	Car#4	MínDist	K-means	Redes Neur
Cromático	Muestras	Lineal					-	-	-
SGLD	Seleccionadas	Lineal	4	10	14	22	14,60%	29,29%	**7,95%**
SGLD	Seleccionadas	Estadístico							
SGLD	Fraccionadas	Lineal							
SGLD	Redimensionadas	Lineal							
SGLD	Muestras	Lineal							
SGLD	Mapas Law's	Lineal							

Procedim.	Selección	Normalización	Car#1	Car#2	Car#3	Car#4	MínDist	K-means	Redes Neur
SGLD	Filtro contraste	Lineal							
SGLD	Mapas Gabor	Lineal							
SGLD	Campos tensoria-les	Lineal							
Law's	Seleccionadas	Lineal					-	-	-

Las características óptimas que se obtienen como resultado del análisis son:

- 4 HOM Homogeneidad
- 10 ENTRO Entropía
- 14 SAVGH Media de la suma
- 22 IDN Momento de la diferencia inversa normalizada

El clasificador de Red Neuronal presenta el problema de los tiempos de computación, ya que el tiempo de convergencia depende directamente del número de neuronas y la cantidad de muestras con que se entrena. Por ello, se han empleado diferente número de neuronas en función del emparejamiento de características:

Tabla 13. Tiempos de computación de las redes neuronales

Nº caract.	Nº neuronas	Tiempo medio (seg)	Tiempo máx. (seg)	Tiempo mín. (seg)
1	10	2,49	13,27	0,62
2	15	5,91	29,20	0,97
3	20	9,11	54,05	1,47
4	25	10,68	113,86	2,08

Debido al carácter aleatorio de las redes neuronales, su resultado no puede considerarse como definitivo, si bien se considera suficiente aproximado. La Figura 55 muestra la variabilidad en el entrenamiento de las redes neuronales.

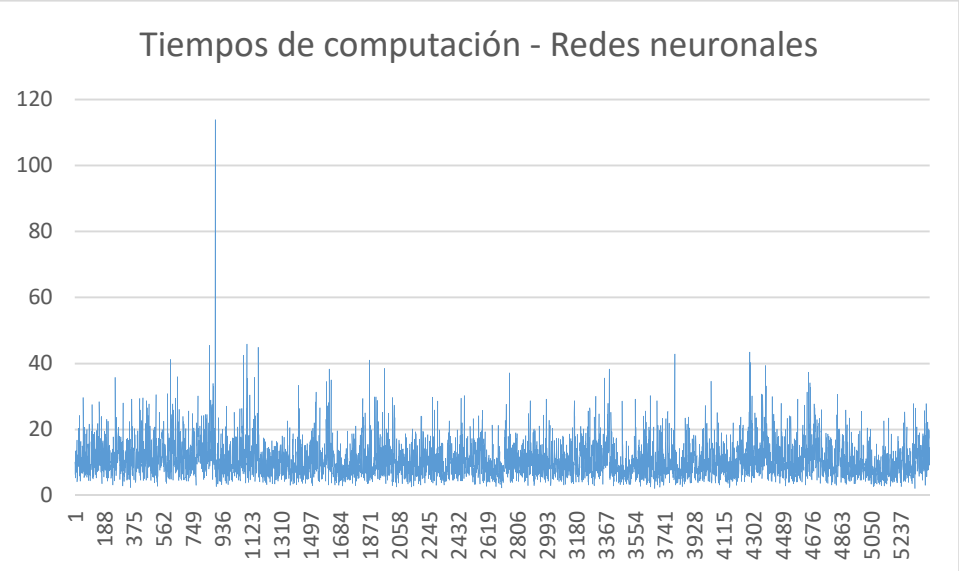


Figura 55. Tiempos de computación de redes neuronales

5.3 Resultado final

Las características empleadas son las recogidas en los apartados anteriores.

Tabla 14. Características empleadas en la técnica mixta

Clasificador	Características texturales				Carac. Cromáticas	
	Txtl_1	Txtl_1	Txtl_1	Txtl_1	Crom_1	Crom_2
Mín. Dist.	7	10	14	18	226	335
K-Means	1	9	13	15	226	335
Red Neur.	4	10	14	22	226	335

Las características cromáticas empleadas son las mismas debido a que en términos de clasificación su resultado es similar.

6 | Segmentación

6.1 Metodología

Este proyecto propone un proceso de segmentación consistente en las fases:

- Generación / carga de los prototipos textural, cromático y mixto conforme a los procedimientos definidos en la sección 4.1.2. Se excluye el uso de las características de Laws debido a su mejorable resultado comparado con los otros tres.
- Extracción de características texturales, cromáticas y mixtas definidas en la sección 5 | de este documento y posterior normalización.
- Clasificación de cada píxel de la imagen conforme a los tres clasificadores definidos en la sección 4.1.4 de este documento.
- Muestra de los resultados de los tres clasificadores conforme a los tres procedimientos; en total, por tanto, se obtienen nueve imágenes.

En la sección 6.2.1 se muestran los resultados como matriz de imágenes, donde la primera imagen corresponde a la imagen original, la segunda corresponde a la clasificación correcta (o categoría de nube SWIMCAT) y las nueve segmentaciones obtenidas.

La sección 6.2.2 recoge la clasificación de imágenes tomadas con medios propios, por lo que no existe una clasificación correcta.

6.2 Resultados de la segmentación

6.2.1 Resultados en imágenes SWIMCAT

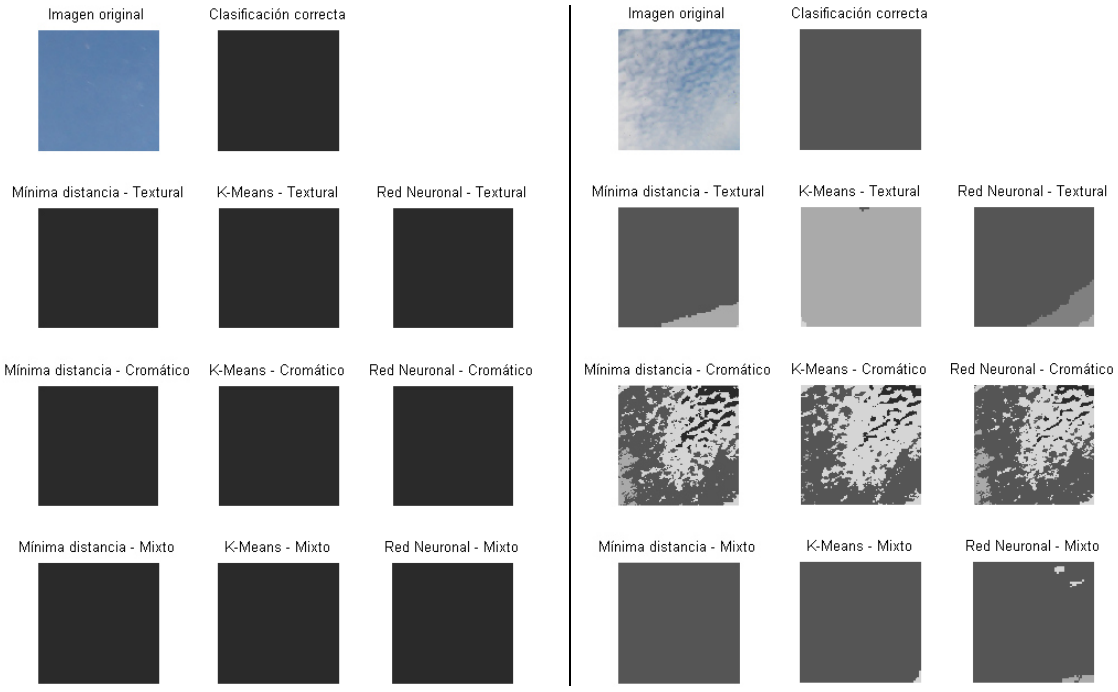


Figura 56. Segmentación de nubes tipo "Cielo" y "Patrón"

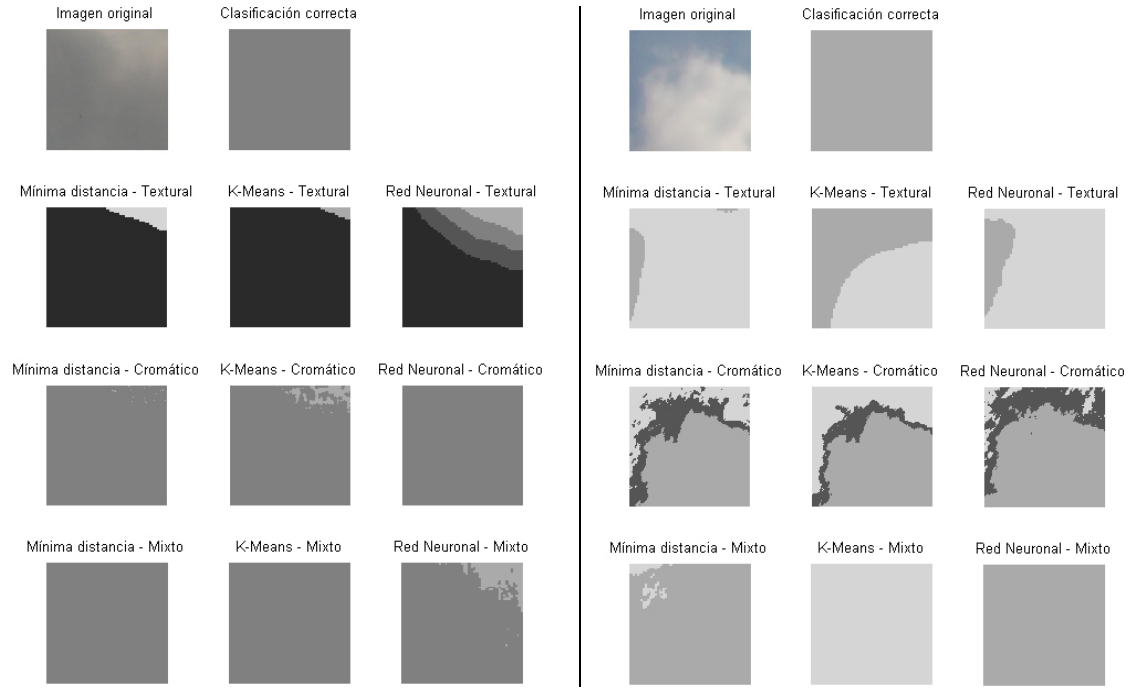


Figura 57. Segmentación de nubes tipo "Nubes Oscuras" y "Nubes Claras"

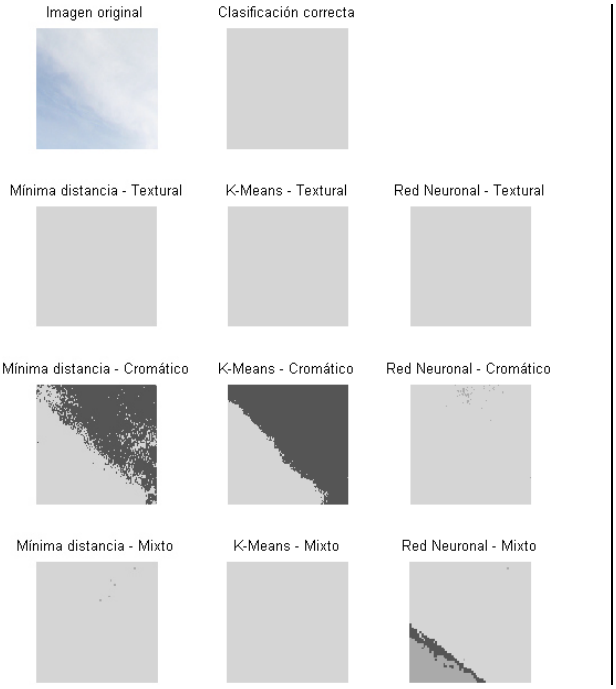


Figura 58. Segmentación de nubes tipo "Velo"

6.2.2 Segmentación de imágenes independientes

Se han aplicado dos métodos de segmentación a distintas imágenes tomadas en diferentes momentos del cielo de Granada.

La Figura 59 muestra un cielo con categoría B-Patrón. Cada una de las partes de la figura presentan las nueve segmentaciones explicadas anteriormente. Sin embargo, en todas ellas se observa una mejora sustancial en el caso de la clasificación mixta. En este caso el clasificador K-means parece obtener un mejor resultado, ya que identifica perfectamente el cielo y la zona de nubes sueltas. Esto se debe a que tiene una mayor tendencia a identificar el cielo sobre el resto de nubes.

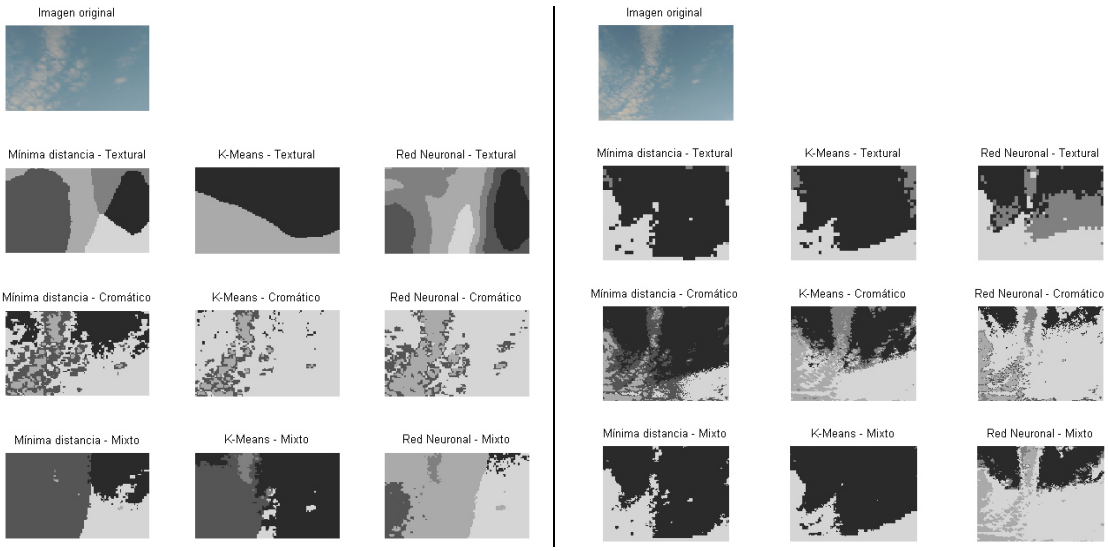


Figura 59. Segmentaciones de nubes independientes

La parte izquierda representa las segmentaciones empleando un procesamiento textural píxel a píxel y un entorno que descarta los bordes de la imagen original.

La parte derecha muestra las segmentaciones considerando entornos para el procesamiento textural basado en entornos separados no solapados, como se muestra en la Figura 48.

Como se observa las segmentaciones texturales de la parte derecha está muy pixelado, debido a las menores dimensiones de esta parte. Estas imágenes contrastan con las segmentaciones texturales de la parte izquierda, suaves y con zonas de nubes claramente diferenciadas.

El resultado de la segmentación textural completa es notablemente mejor que la anterior pero debido a los tiempos de procesamiento se descartan para aplicaciones en tiempo real.

El Anexo 3 recoge un conjunto de imágenes independientes segmentadas.

7 | Interfaz de usuario

El interfaz de usuario es un diálogo programado en MATLAB que integra los diferentes pasos del Análisis y la Segmentación. La Figura 60 muestra el "Layout" del interfaz.

Figura 60. Layout del interfaz de usuario por secciones

Este interfaz se ha dividido en cinco áreas:

1. Configuración

   Área donde se configuran los parámetros iniciales que se emplean en las áreas 1 a 4. Se debe escoger el procedimiento (cromático, textural o mixto), las imágenes, la normalización (lineal o estadística) y el clasificador (mínima distancia, K-means y red neuronal).

   Una vez configurado se pueden generar las matrices para trabajar con ellas y además salvarlas para un uso posterior.

2. Muestra de resultados

   Área para la generación de los diagramas de 1D, 2D y 3D de clustering para la comparación de resultados.

3. Generación de imágenes

Si se desea, se pueden generar imágenes fraccionadas o redimensionadas para su posterior estudio y clasificación, a las ratios dados de 0.5 y 0.33 en caso de redimensionamiento y de 4 y 9 en caso de fraccionamiento.

4. Clasificación

   Área para la generación de las tablas de clasificación para los parámetros configurados. Este proceso consiste en la generación de prototipos y clasificación de las mismas imágenes con las que se han entrenado los algoritmos. Estos datos aportan una importante información acerca de la bondad de los prototipos generados.

5. Segmentación

   Área para la generación de prototipos específicos y la segmentación de imágenes individuales.

7.1 Configuración

7.1.1 Variables de configuración

La Tabla 15 incluye todas las variables de configuración de este módulo. Las variables sombreadas corresponden a valores configurables desde los menús o textos.

Tabla 15. Variables MATLAB de Configuración

Variable	Sección	Valor Inicial	Tipo
handles.PROCED	Configuración	0	Entrada
handles.SELEC	Configuración	-1	Entrada
handles.NORMALIZ	Configuración	1	Entrada
handles.CLASIF	Configuración	0	Entrada
handles.Inic_foto	Configuración	'A_'	Entrada
handles.Fin	Configuración	'img.pgn'	Entrada
handles.nroFotos	Configuración	1	Interna
handles.ANG	Configuración	4	Interna
handles.NroClases	Configuración	5	Interna
handles.FraccionaEN	Configuración	0	Interna
handles.RedimensionaA	Configuración	0	Interna
handles.Seleccion	Configuración	[]	Interna
handles.carpetaNubes	Configuración	[]	Interna
handles.carpeta_nubes	Configuración	[]	Interna
handles.Inic_foto	Configuración	[]	Interna
handles.Fin	Configuración	[]	Interna
handles.nroFotos	Configuración	[]	Interna
handles.NroParam	Configuración	[]	Interna
handles.Procedim	Configuración	[]	Interna
handles.TitleParam	Configuración	[]	Interna
handles.MC	Configuración	[]	Salida

Variable	Sección	Valor Inicial	Tipo
handles.MCn	Configuración	[]	Salida
handles.MCn1	Configuración	[]	Salida
handles.MuestrasAcum	Configuración	[]	Salida
handles.MuestrasAcum1	Configuración	[]	Salida
handles.nroMuestras	Configuración	[]	Salida
handles.MAXe	Configuración	[]	Salida

7.1.2 Macro 'Generar Matrices'

La Figura 61 muestra el flujograma para la Generación de Matrices durante la fase de Configuración.

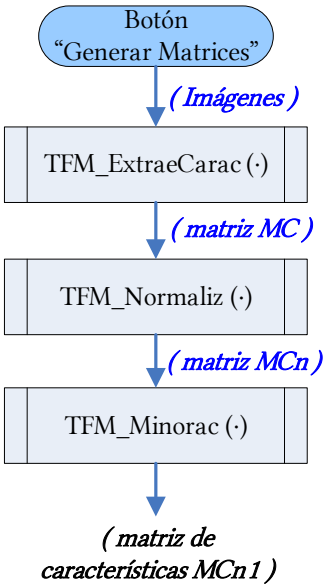


Figura 61. Flujograma del botón "Generar Matrices"

7.1.3 Macro 'Salvar Matrices'

La Figura 62 muestra el flujograma para salvar las matrices como archivo '.mat' durante la fase de Configuración.

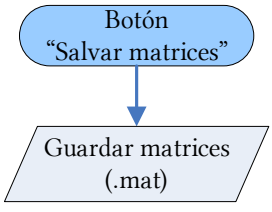


Figura 62. Flujograma del botón "Salvar Matrices"

7.1.4 Macro 'Cargar Matrices'

La Figura 63 muestra el flujograma para salvar las matrices como archivo '.mat' durante la fase de Configuración.

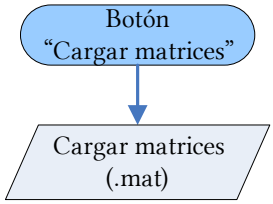


Figura 63. Flujograma del botón "Cargar Matrices"

7.2 Muestra de resultados

7.2.1 Variables de configuración

La Tabla 16 incluye todas las variables de configuración de este módulo.

Tabla 16. Variables MATLAB de Muestra de Resultados

Variable	Sección	Valor Inicial	Tipo
handles.VISUAL_1D	Muestra de resultados	0	Entrada
handles.VISUAL_2D	Muestra de resultados	0	Entrada
handles.VISUAL_3D	Muestra de resultados	0	Entrada
handles.carpeta_figuras	Muestra de resultados	[]	Entrada

7.2.2 Macro 'Mostrar Resultados'

La Figura 64 muestra el flujograma para la Muestra de Resultados durante la fase de Configuración, consistente en diagramas de 1D, 2D o 3D de las nubes de clustering de cada una de las imágenes de nubes SWIMCAT.

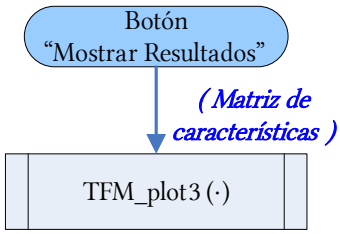


Figura 64. Flujograma del botón "Mostrar Resultados"

7.3 Generar Imágenes

7.3.1 Variables de configuración

La Tabla 17 incluye todas las variables de configuración de este módulo.

Tabla 17. Variables MATLAB de Generación de Imágenes

Variable	Sección	Valor Inicial	Tipo
handles.TYPE	Generar Imágenes	[]	Entrada
handles.ESCALA	Generar Imágenes	[]	Entrada

7.3.2 Macro 'Generar Imágenes'

La Figura 65 muestra el flujograma para la Generación de Imágenes durante la fase de Configuración.

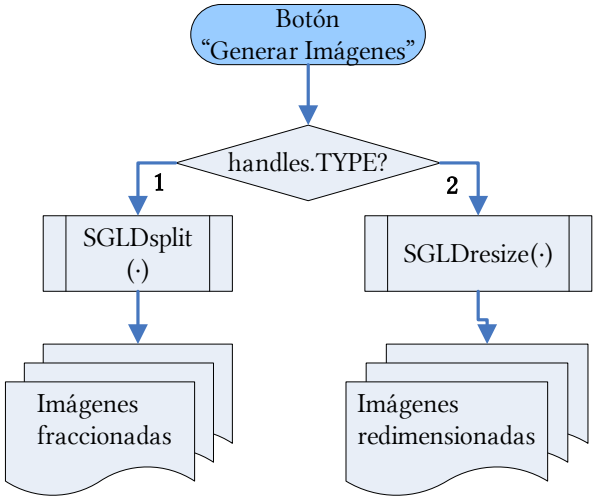


Figura 65. Flujograma del botón "Generar Imágenes"

7.4 Clasificación

7.4.1 Variables de configuración

La Tabla 18 incluye todas las variables de configuración de este módulo.

Tabla 18. Variables MATLAB de Clasificación

Variable	Sección	Valor Inicial	Tipo
handles.CLASIF_1D	Clasificación	0	Entrada
handles.CLASIF_2D	Clasificación	0	Entrada
handles.CLASIF_3D	Clasificación	0	Entrada
handles.CLASIF_4D	Clasificación	0	Entrada
handles.CLASIF_xD	Clasificación	[0,0,0,0]	Entrada

7.4.2 Macro 'Generar Tablas'

La Figura 66 muestra el flujograma para la Generación de Tabla de clasificación como archivo Excel durante la fase de Configuración.

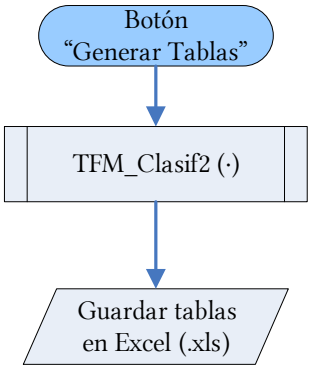


Figura 66. Flujograma del botón "Generar Tablas"

7.5 Segmentación

7.5.1 Variables de configuración

La Tabla 19 incluye todas las variables de configuración de este módulo.

Tabla 19. Variables MATLAB de Segmentación

Variable	Sección	Subsección	Valor Inicial
handles.SegmCrom	Segmentación	Prototipos	0
handles.SegmTxtl	Segmentación	Prototipos	0
handles.SegmMix	Segmentación	Prototipos	0
handles.SegmMinDist	Segmentación	Prototipos	0
handles.SegmKmeans	Segmentación	Prototipos	0
handles.SegmNeural	Segmentación	Prototipos	0
handles.CarCrom1	Segmentación	Prototipos	226
handles.CarCrom2	Segmentación	Prototipos	335
handles.CarTxtl1	Segmentación	Prototipos	7
handles.CarTxtl2	Segmentación	Prototipos	10
handles.CarTxtl3	Segmentación	Prototipos	14
handles.CarTxtl4	Segmentación	Prototipos	18
handles.CarMix_Crom1	Segmentación	Prototipos	226
handles.CarMix_Crom2	Segmentación	Prototipos	335
handles.CarMix_Txtl1	Segmentación	Prototipos	4
handles.CarMix_Txtl2	Segmentación	Prototipos	5
handles.CarMix_Txtl3	Segmentación	Prototipos	8
handles.CarMix_Txtl4	Segmentación	Prototipos	10
handles.Imag2segmentar	Segmentación	Segmentar	[]
handles.Imag2segmFin	Segmentación	Segmentar	[]

7.5.2 Macro 'Generar Prototipos'

La Figura 67 muestra el flujograma para la Generación de Prototipos durante la fase de Segmentación.

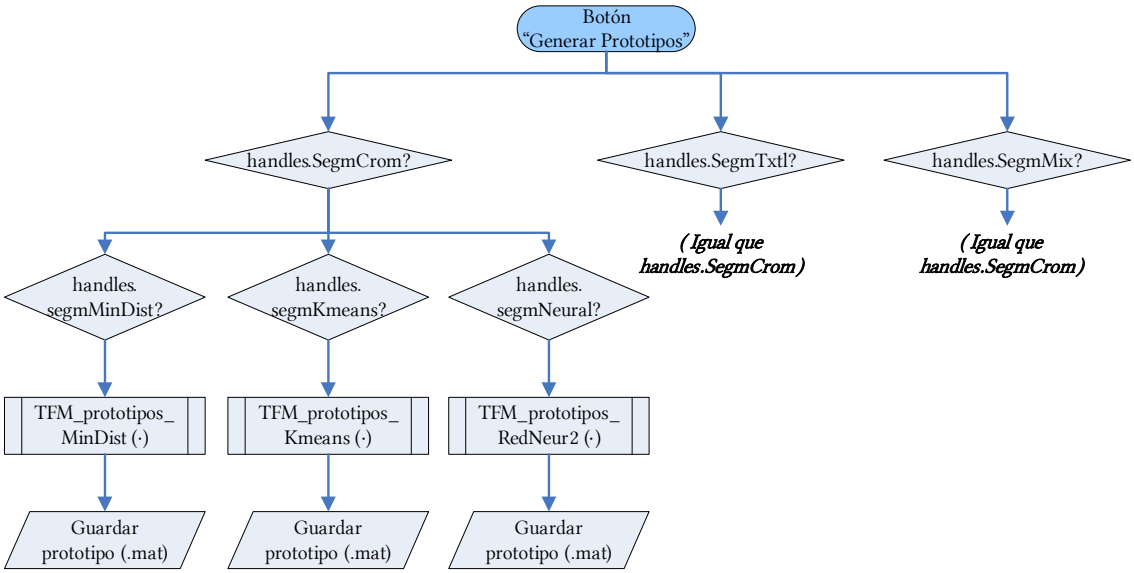


Figura 67. Flujograma del botón "Generar Prototipos"

7.5.3 Macro 'Segmentar'

La Figura 68 muestra el flujograma para la iniciar el proceso de segmentación durante la fase de Segmentación.

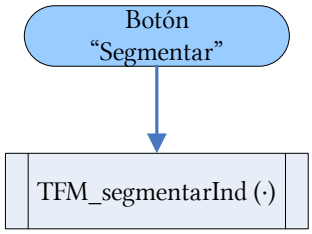


Figura 68. Flujograma del botón "Segmentar"

8 | Desarrollo de plataforma terrestre

8.1 Metodología

La implementación del Sistema Automático de Identificación de Nubes en una plataforma hardware que puede trabajar en la intemperie se ha realizado sobre el SBC de bajo coste Raspberry Pi.

Se han programado en Python los algoritmos de identificación textural en cuadrícula, así como la segmentación cromática y la segmentación mixta. No obstante, no se ha implementado la técnica mixta con procesamiento textural completa debido a falta de recursos de la plataforma hardware.

Durante el desarrollo de la aplicación se ha apreciado un óptimo manejo del cálculo matricial del sistema, pero extremadamente lento ante bucles recursivos, por lo que no se obtienen buenos resultados en términos de tiempos de procesamiento ante la implementación del sistema desarrollado en MATLAB durante las fases de Análisis y Segmentación.

8.2 Especificaciones técnicas

8.2.1 Plataforma hardware

Característica	Raspberry Pi 3, Modelo B
SoC	Broadcom BCM2837 (CPU + GPU + DSP + SDRAM + Puerto USB
CPU	1.2GHz 64-bit quad-core ARMv8
Juego de instrucciones	RISC de 32 bits
GPU	Broadcom VideoCore IV, OpenGL ES 2.0, MPEG-2 y VC-1 (con licencia), 1080p30 H.264/MPEG-4 AVC
SDRAM	1 GB (compartidos con la GPU)
Puertos USB 2.0	4
Entradas de vídeo	Conector MIPI CSI que permite instalar un módulo de cámara desarrollado por la RPF
Salidas de vídeo	Conector RCA (PAL y NTSC), HDMI (rev1.3 y 1.4), Interfaz DSI para panel LCD
Salidas de audio	Conector de 3.5 mm, HDMI
Almacenamiento integrado	MicroSD
Conectividad de red	10/100 Ethernet (RJ-45) vía hub USB, Wifi 802.11n, Bluetooth 4.1
Periféricos de bajo nivel	17 x GPIO y un bus HAT ID
Reloj en tiempo real	Ninguno
Consumo energético	800 mA, (4.0 W)
Fuente de alimentación	5 V vía Micro USB o GPIO header
Dimensiones	85.60mm × 53.98mm

Característica	Raspberry Pi 3, Modelo B
Sistemas operativos soportados	GNU/Linux: Debian (Raspbian), Fedora (Pidora), Arch Linux (Arch Linux ARM), Slackware Linux. RISC OS
Sistema operativo empleado	Debian

8.2.2 Entorno de programación

El entorno de programación empleado ha sido Python. Por lo que se ha programado todo el código desde cero para su migración desde MATLAB.

Adicionalmente se han empleado algunas librerías específicas para el tratamiento de imágenes y la obtención de características texturales:

- OpenCV: librería para el tratamiento de imágenes
- Numpy: librería de cálculo matricial
- SciKit: librería que contiene la función para el cálculo de la matriz SGLD.

Así mismo, se ha instalado la aplicación TightVNC para la conexión remota al dispositivo a través de la red inalámbrica.

8.3 Montaje

El sistema, mostrado en la Figura 69, se compone de tres elementos:

- Raspberry Pi 3, con WiFi integrado, con TightVNC operativo;
- Cámara web; y
- Fuente de alimentación.

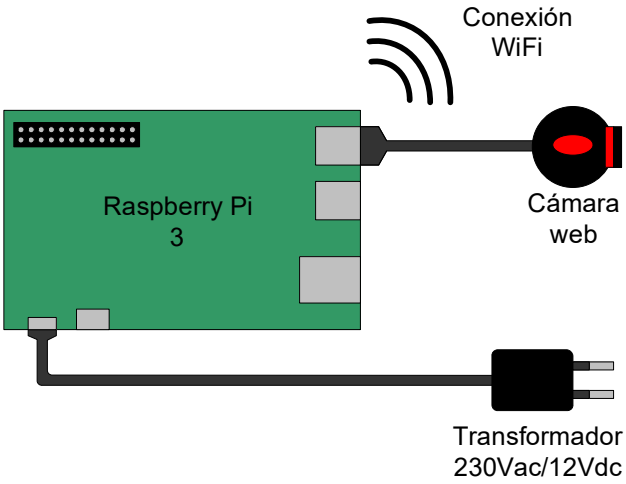


Figura 69. Montaje con Raspberry Pi 3

La cámara web es 'plug and play' por lo que no necesita más configuración que la librería OpenCV para tener acceso a ésta.

8.4 Resultados de la Identificación “en línea”

El objetivo final de la implementación en una plataforma hardware es la obtención de imágenes segmentadas “en vivo”.

Las imágenes segmentadas se representan en escala de grises siguiendo las tonalidades empleadas durante la segmentación en Matlab. La Figura 70 muestra las cinco clases de nube, obtenidas directamente de la Raspberry, como verificación de la configuración.

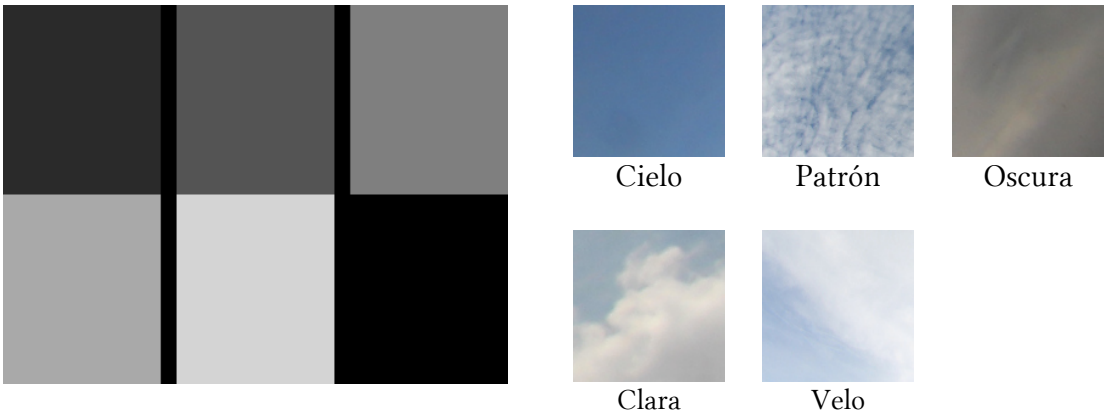


Figura 70. Escala de gris de los cinco tipos de nube

8.4.1 Ejemplo con cielo despejado

A continuación, se muestra la Figura 71 como resultado de la segmentación “en vivo” de un cielo despejado, con obstáculos y cielo degradado con diferentes tonalidades de azul hasta casi blanco en el horizonte.

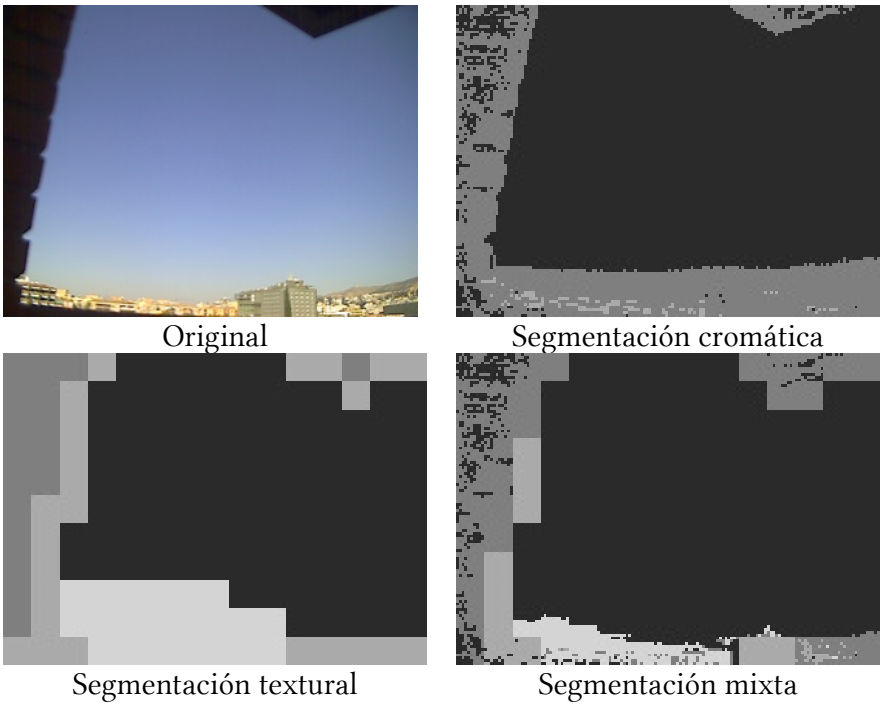


Figura 71. Segmentación mediante Raspberry Pi de cielo despejado

Como se observa, la segmentación cromática muestra una segmentación mayoritariamente de cielo, con la parte baja como nube oscura y clasificando la columna, el techo y los edificios con diferentes clases. El resultado es, por tanto, lo esperado.

La segmentación textural propone de igual manera un cielo despejado, identificando la columna y techo con textura de nubes oscuras y su borde como nubes claras. Asimismo identifica la parte baja del cielo despejado, casi blanco, como nube tipo "velo". El resultado es correcto.

Finalmente, el resultado combinado de la identificación mediante la técnica mixta cromático-textural se obtiene un cielo despejado con la columna y techo difuminados en cuanto a tipo de nube y un horizonte correctamente segmentado, dejando los edificios fuera de la calificación de cielo. Esta técnica mejora los resultado de las identificaciones cromática y textural por separado.

8.4.2 Ejemplo con nubes claras

La Figura 72 muestra la identificación de un cielo con nubes claras. Debido al tamaño de los cuadros de identificación, el resultado es una segmentación marcada por los cuadros texturales.

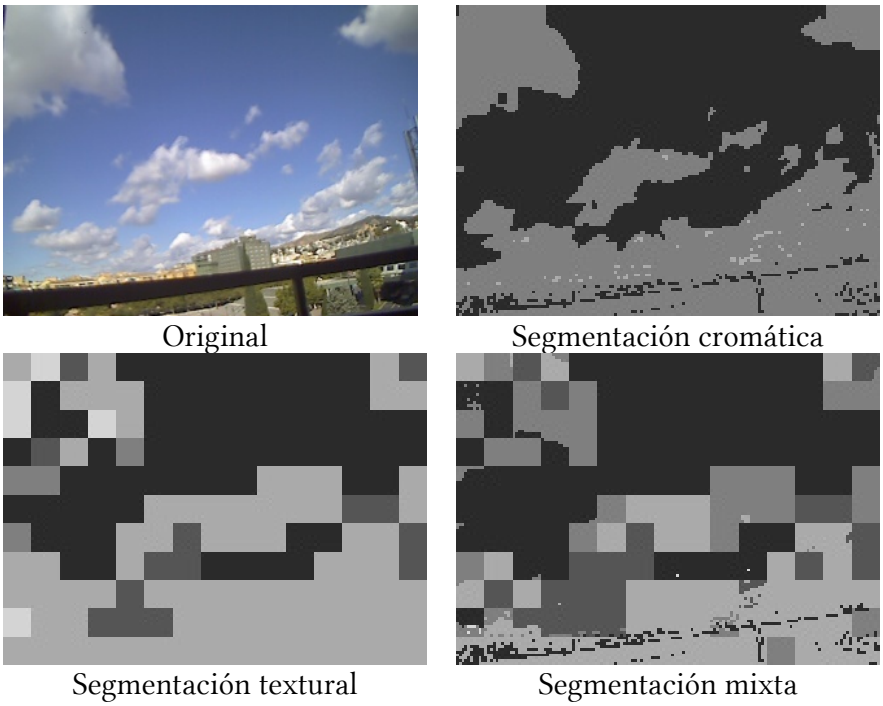


Figura 72. Segmentación mediante Raspberry Pi de nubes claras (I)

Nótese, sin embargo, que frente a la segmentación cromática como nubes oscuras, la caracterización textural redirige la identificación hacia nubes claras, que es lo esperado.

Por otro lado, se ha repetido la identificación empleando cuadros de características texturales más pequeños, 10 x 10 píxeles, en lugar de 20 x 20 en el resto de identificaciones. El resultado es una identificación más lenta pero más clara, como se puede ver en la Figura 73.

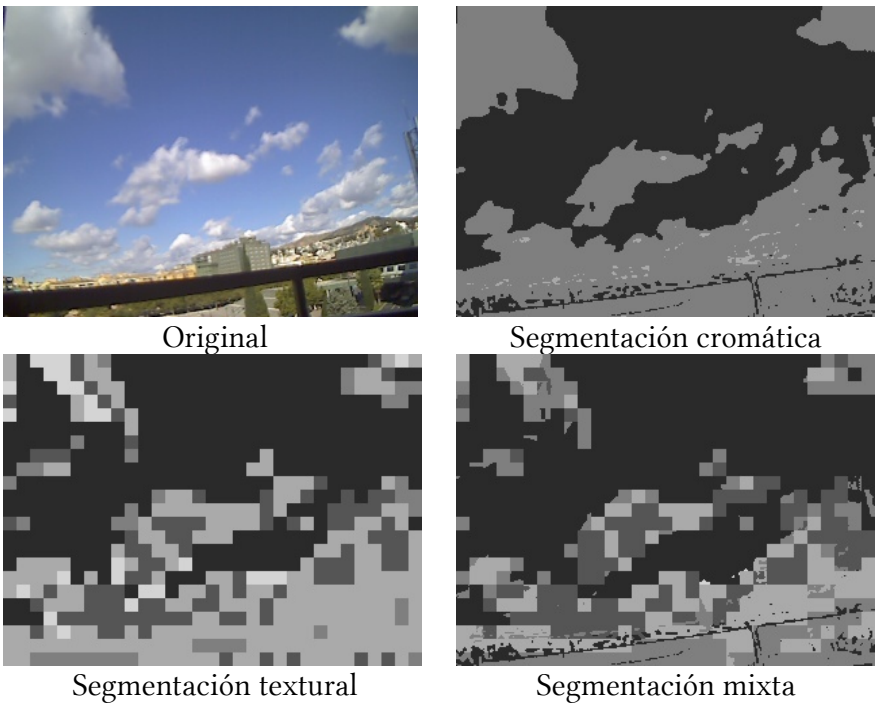


Figura 73. Segmentación mediante Raspberry Pi de nubes claras (II)

La segmentación cromática se mantiene invariable. Son las segmentaciones textural y mixta, a instancias de la primera, las que cambian. Como se observa, la segmentación textural acierta a identificar la mayor parte de la nube como nube clara, pero poco definida. Ya en la identificación mixta se observa la forma de la nube y una identificación aceptable dada la sombra que se forma en las nubes y que muestran un color oscuro.

8.4.3 Ejemplo con nubes tipo velo

Finalmente se ha tomado una muestra de cielo con nubes de tipo velo y se ha realizado una identificación con cuadros 10x10 píxeles. El resultado se muestra en la Figura 74.

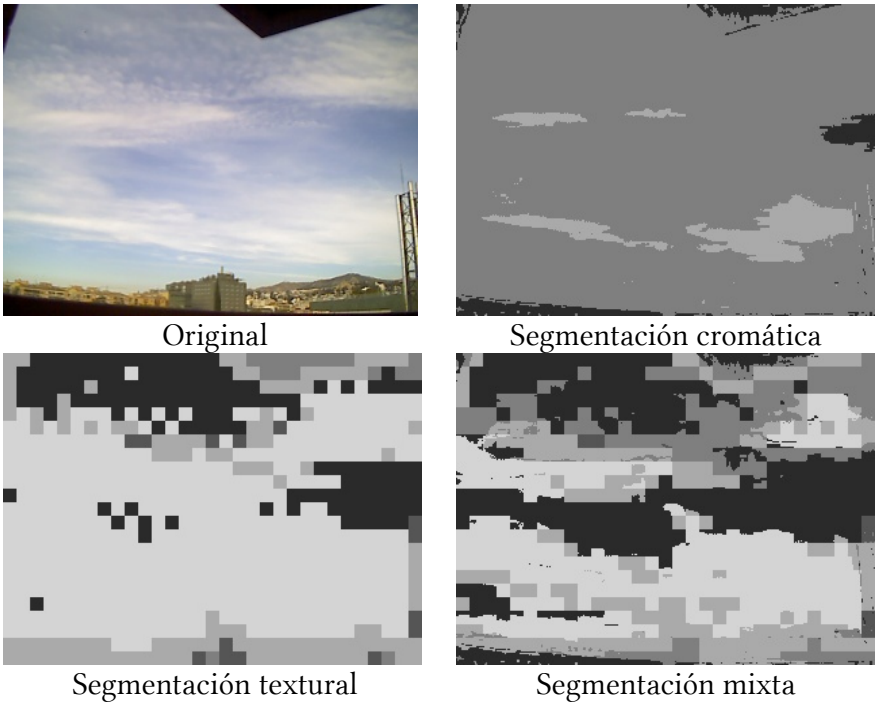


Figura 74. Segmentación mediante Raspberry Pi de nubes tipo velo

De nuevo la segmentación cromática marca bien los límites de las nubes y el cielo, pero la clasificación, si bien, correcta en el núcleo de las nubes, falla en su mayoría clasificándola como nube oscura. La segmentación textural está aún pixelada pero acierta en el tipo de nube, obteniendo como resultado una segmentación mixta que llega a identificar incluso los patrones de nubes que se forman en la parte superior de la imagen.

9 | Desarrollo de aplicación móvil

Como última fase del desarrollo se he implementado la segmentación cromática como aplicación móvil para dispositivos Android, junto con las diferentes vistas de los espacios cromáticos que se han empleado.

El objeto de esta aplicación es mostrar la aplicación práctica y comercial del desarrollo realizado. Al mismo tiempo cumple una función educativa por lo que queda como campos de desarrollo futuros la inclusión de una sección acerca del tratamiento cromático y otra que recoja los diferentes tipos de nube.

Se trata de un producto puramente académico que incluye elementos para una posible puesta en mercado. Los esfuerzos se han centrado en el correcto funcionamiento del filtro de tipos de nube.

9.1 Identidad visual

Si bien este no es un aspecto específico del proyecto, se ha realizado una labor de documentación y diseño como ejercicio adicional a la implementación misma de los algoritmos de segmentación cromática.

9.1.1 Icono de lanzamiento

El icono de lanzamiento es el primer elemento, junto con el nombre, que sirve de escaparate para la aplicación. Se trata del '*packaging*' *que lo envuelve.*

En primer lugar, este icono servirá para representar a la aplicación en las diferentes tiendas de aplicaciones —junto a las pantallas y textos promocionales— como elemento de venta para convencer al usuario de descargarla.

En este caso se ha optado por un icono de alto impacto visual, con colores vivos, para captar la atención del potencial cliente. Las letras se han escogido conforme a las iniciales del creador y autor del proyecto. En caso de comercialización este icono habrá de ser actualizado.

El icono de lanzamiento y su efecto en una pantalla de móvil se muestran en la Figura 75.

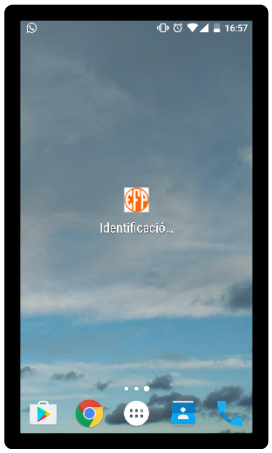


Figura 75. Aplicación móvil – Icono de lanzamiento

9.1.2 Pantalla de la aplicación

La pantalla de la aplicación se compone de los elementos:

- Grilla o retícula

  La retícula —grid en inglés— es la estructura invisible sobre la cual se apoyan todos los elementos visuales. Su función es la de separar cada uno de los componentes de la interfaz en un espacio ordenado, organizando los sitios que quedarán en blanco y aquellos que contendrán formas. Una retícula bien definida se transforma en una ayuda al diseño que, generando orden y simplicidad, mejora la usabilidad de la app.

  En el diseño de interfaces para móviles, la retícula permite establecer márgenes y determinar la ubicación de los botones, la separación de la tipografía y el espacio interior y exterior de los contenedores. Por supuesto, cada uno de los sistemas operativos tiene diferentes retículas y por tanto distintos módulos.

  En Android, el módulo base es de 48dp que equivale aproximadamente a nueve milímetros, tamaño mínimo recomendado para elementos interactivos. Basarse en este tamaño y respetar estas dimensiones para los botones, permite asegurar que estos podrán ser tocados con el dedo sin problemas, cuestión fundamental en el diseño para móviles.

  Para el espaciado y separación, en cambio, se usa un módulo de 8dp. Por ejemplo, el contenido de filas tiene una separación —superior e inferior— de 4dp, lo que hace que cuando dos filas están una sobre otra, se sumen conformando un espacio total de 8dp entre ellas. En los márgenes laterales, los diseños suelen tener 16dp o, dicho de otra forma, dos módulos de 8dp juntos.

- Tipografía

  El objetivo de la tipografía es conseguir que el texto se lea con claridad. Esto se logra no solo con una adecuada elección de la fuente, sino también gestionando su tamaño, separación entre líneas, ancho de columnas y contraste visual con el fondo.

  Este último punto, el contraste, es más importante de lo que puede creerse a simple vista. Un móvil es un dispositivo que muchas veces se usará fuera de casa, por ejemplo, en la calle. En algunos momentos el sol dará directamente sobre la pantalla y si no hay un buen contraste entre tipografía y fondo, la información en pantalla será imposible de leer.

  La tipografía es un componente que, como botones y gráficos, también se asienta en una retícula que definirá su ubicación y posición dentro del contexto general de la pantalla.

  Aunque en móviles no sea habitual una lectura de texto prolongada, la correcta legibilidad es una parte fundamental del diseño; por esta razón, la tipografía es tan importante como cualquier otro elemento visual que se incorpore en una interfaz y su elección no debería descuidarse.

  - Legibilidad y resolución

    Por ser un soporte digital, los móviles tienen características propias y algunas limitaciones ajenas a los medios impresos tradicionales. La pantalla

influye bastante en el comportamiento y desempeño tipográfico si se tiene en cuenta que, en algunos casos, es sumamente pequeña.

Actualmente, muchos dispositivos tienen pantallas con buena resolución, lo que elimina un factor de complejidad y una gran preocupación a la hora de elegir qué fuente se usará. Este tipo de pantallas suele encontrarse en móviles de alta gama, sin embargo, hay otros teléfonos con características no tan interesantes.

En este último caso, elegir la tipografía es una tarea bastante difícil que no se limita a la elección de familia y tamaño, ya que, mientras más pequeña sea la pantalla, hay más posibilidades de que la forma en que se pintan en ella los caracteres, sea más pobre.

- Tamaños mínimos

  En soportes digitales, como un móvil o una tableta, algo que condiciona el tamaño tipográfico es la distancia a la cual se sujeta el dispositivo. Los móviles suelen sostenerse de forma que la pantalla está más cerca del ojo del lector que en el caso de una tableta, lo que permite que el tamaño de la tipografía sea más pequeño.

  Al mismo tiempo, en los teléfonos el espacio en pantalla es mucho menor, lo cual obliga a ajustar el interlineado y la separación entre caracteres, para aprovechar el área disponible sin perjudicar la lectura.

  Los tamaños mínimos pueden variar dependiendo del sistema operativo, la resolución de la pantalla y la fuente que se elija. En todo caso, cuando se trata de tamaños pequeños, se aconseja elegir fuentes de formas simples y abiertas, con espaciado entre caracteres, líneas y márgenes para dar aire visual que facilite la lectura.

  La mejor forma de asegurar una correcta legibilidad no es otra que ponerla a prueba en el teléfono para el cual se diseña. Los diseños en pantalla de ordenador suelen ser engañosos y comprobarlos en el entorno más real posible, permite realizar ajustes y correcciones hasta conseguir el tamaño adecuado.

  En Android, el tamaño tipográfico se mide en sp —scaled pixels o píxeles escalados—, una forma de modificar la escala de las fuentes de acuerdo al tamaño de pantalla y a las preferencias definidas por el usuario en su configuración del teléfono. Los tamaños más comunes van desde 12sp hasta 22sp.

- Jerarquías

  Como elemento visual y componente de una interfaz, la tipografía también es susceptible de ser jerarquizada. Su importancia depende no solo de la función que cumple, sino también de la información que contiene y su posición en pantalla.

  Para definir diferentes niveles de protagonismo en un texto se puede apelar, además del tamaño, a las variantes —negrita, regular o light, por ejemplo— y al color.

Un texto con mayor jerarquía sería aquel que se ubica como título principal de sección. Por otro lado, en la información contenida dentro de una fila en una lista, puede haber varias jerarquías, por ejemplo, en el caso de un correo, podría verse el nombre de la persona que lo envía, un resumen del contenido y la fecha de envío; todos estos elementos tienen diferente importancia y definirla es el primer paso para saber qué estilo tipográfico aplicar.

- Color

  El color es un recurso vital en el diseño de una aplicación. Su uso abarca encabezados, textos, botones, fondos y muchos otros elementos que conforman la interfaz. En algunas ocasiones, está asociado a la identidad —color corporativo— y en otras, responde a criterios estéticos y decisiones de diseño.

  Un color por sí solo, salvo en el caso de colores reservados, no indica mucho. Como parte de un sistema cromático, el uso consistente, consciente y vinculado al contexto donde se aplica, es lo que lo llena de significado para el usuario.

  Hay ciertos colores que deben emplearse de forma cuidadosa porque tienen connotaciones que no pueden obviarse. Se llaman justamente «colores reservados» porque su uso debería limitarse especialmente a los nombrados a continuación:

  a. Rojo: Para errores y alertas importantes. Es un color que naturalmente indica peligro y llama la atención para centrarse inmediatamente en lo que está ocurriendo.

  b. Amarillo: Prevención. Señala que la acción que va a realizarse implica la toma de una decisión que ocasiona alguna consecuencia, por lo cual hay que estar alerta.

  c. Verde: Mensajes de éxito y confirmación de que una acción se ha realizado correctamente.

- Lenguaje

  En una aplicación hay infinidad de lugares donde usar textos: en encabezados y títulos, botones, mensajes de error y avisos en pantallas vacías. En cada caso, las palabras son tan importantes como el elemento gráfico que las contiene o acompaña. Lenguaje textual y visual van de la mano, unidos para lograr una experiencia de usuario consistente en todos los sentidos.

- Detalles visuales

  La diferencia entre aplicaciones radica en los detalles. Ellos pueden separar una aplicación regular de una genial. En una primera instancia no se presta atención a estos detalles que mejoran la experiencia del usuario e incluso, cambian su estado de ánimo; y es normal, al principio hay otros aspectos de diseño de los cuales hay que ocuparse. Pero a medida que la interfaz va llegando a su fin, es importante considerarlos para no pasar por alto algo que pueda arruinar el trabajo realizado.

Tras el recorrido sobre las principales características que debe cuidar una aplicación, se repasan los parámetros escogidos para esta aplicación, cuya pantalla se muestra en la Figura 76.

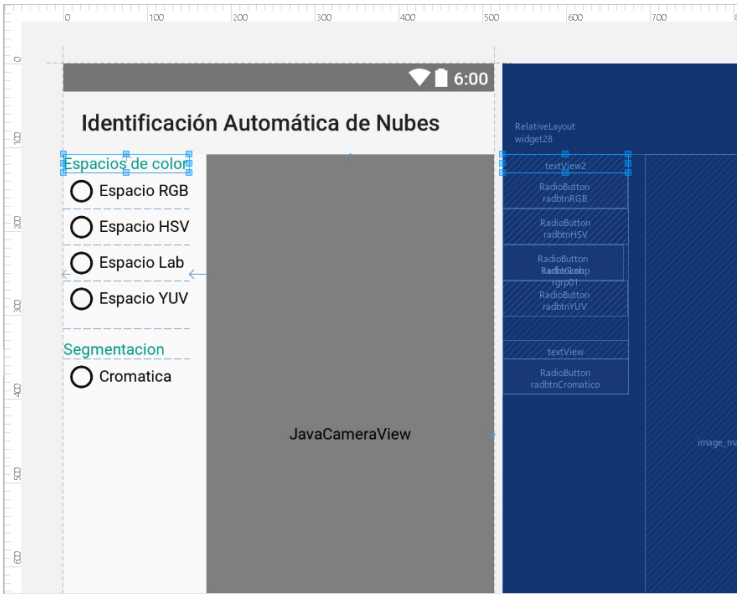


Figura 76. Aplicación móvil – Pantalla de la aplicación

Se observa que la retícula se distribuye de forma uniforme para las posibilidades y subtítulos.

La integración del concepto *jerarquía* se consigue mediante el color de las fuentes y el tamaño en el caso del nombre de la aplicación.

Las etiquetas del tipo "Radio" (opciones) tienen un tamaño mínimo de fuente de 14dp, mientras que el título de la aplicación se mantiene con un tamaño superior.

El lenguaje empleado, en este caso limitado a los tipos de espacios de color y segmentación, se enfocan a ser al mismo tiempo escuetos y específicos, sin que quepa lugar a confusión.

Finalmente, la atención a los detalles es ligeramente incorrecta, puesto que la imagen que muestra no llena la pantalla, conforme al diseño original que se hizo, sino que su resolución se limita a 640x480 píxeles por temas de estabilidad y robustez dada la potencia de cálculo de algunos móviles.

9.2 Interfaz de usuario

El interfaz de usuario muestra cinco elementos principales, tal como se muestra en la Figura 77:

- Barra de título,
- Imagen del cielo "en vivo",
- Sección "Espacios de color",
- Sección "Segmentación", y

- Botón “OBTENER”.

De estas opciones solo las tres últimas son interactivas. El funcionamiento de la aplicación consiste en la selección de un único espacio de color o segmentación para que el móvil muestre en pantalla el resultado del proceso de filtrado o conversión.

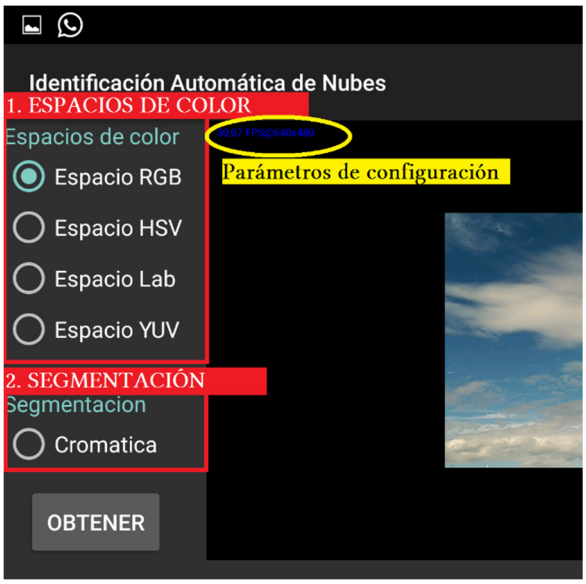


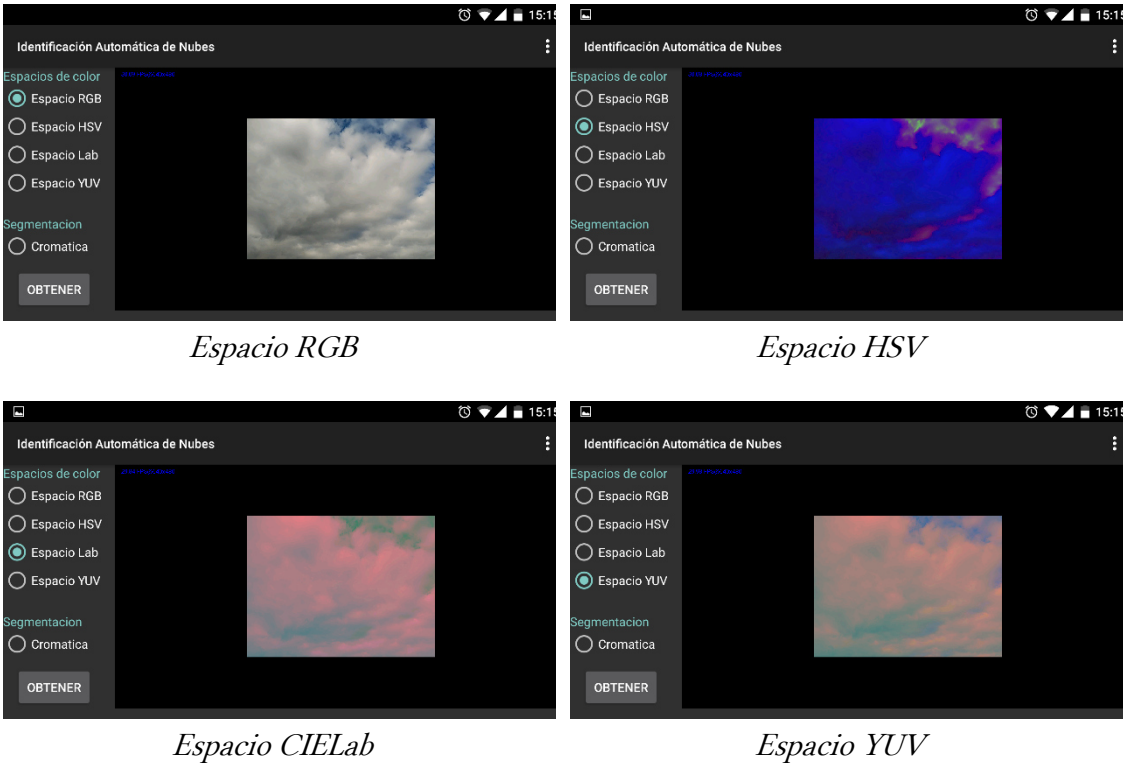
Figura 77. Aplicación móvil – Interfaz de usuario

9.2.1 Espacios de color

La primera sección de la aplicación recoge los cuatros grandes espacios de color empleados en el desarrollo de la aplicación, si bien tras la selección de los mejores parámetros como resultado del análisis recogido en la sección 5 | de este documento limita las características a los tres primeros.

La Figura 78 muestra los cuatro espacios de color disponibles:

- RGB,
- HSV,
- CIELab, y
- YUV, como equivalente al YIQ empleado en el análisis, ya que la plataforma de desarrollo no disponía de este segundo espacio.



Espacio RGB *Espacio HSV*

Espacio CIELab *Espacio YUV*

Figura 78. Aplicación móvil – Espacios de color

9.2.2 Segmentación cromática

Finalmente se ha implementado la segmentación cromática como funcionalidad principal de la aplicación como se muestra en la Figura 79.

Su funcionamiento es notablemente más lento que los espacios de color, teniendo los primeros una velocidad de refresco de unos 30 fps (frame per second) frente a los 1,5 fps de la segmentación.

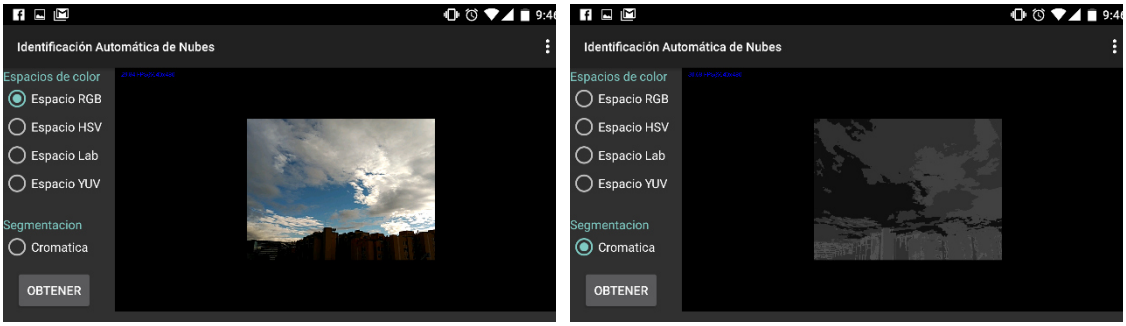


Figura 79. Aplicación móvil – Segmentación cromática

El proceso de cálculo de la segmentación sigue las pautas del empleado en el análisis y se recoge en la Figura 45 y Figura 46.

9.2.3 Otras segmentaciones

Ha resultado imposible la implementación de la segmentación textural debido a que la plataforma móvil sobre la que se ha programado la aplicación (Java) tolera mal los retrasos debido a iteraciones, obteniendo un producto inestable.

9.3 Especificaciones técnicas

9.3.1 Plataforma hardware

Se ha empleado para las pruebas un móvil bq Aquarius M5 con la versión Android 6.0.1 (Marshmallow).

El equipo de desarrolla se trata de un portátil Lenovo Z50-70, con procesador Intel Core i7-4510U y 16GB de RAM. El sistema operativo instalado es Windows 10.

9.3.2 Entorno de programación

El entorno de programación empleado ha sido Android Studio 2.2.2, donde la programación se hace en JAVA, con las versiones JRE 1.8.0 y versión JVM OpenJDK 64-Bit Server VM.

Adicionalmente se han empleado las librerías específicas para el tratamiento de imágenes:

- HAXM para la aceleración del emulador, y
- OpenCV 3.1.0: librería nativa para el tratamiento de imágenes.

10 | Conclusiones

Este proyecto abarca las fases de Análisis, Segmentación e Implementación orientado al desarrollo de un Sistema Automático de Identificación de Nubes. Cada una de estas fases representa un proceso independiente del resto, en cuanto se han establecido entradas y salidas a cada una de ellas y, por tanto, segregable.

La fase de Análisis busca las mejores características cromáticas y texturales que puedan emplearse para la identificación de nubes. Se han analizado 9 variantes diferentes de las imágenes tomadas del SWIMCAT, aplicando 3 procedimientos diferentes y evaluando las características obtenidas por 3 clasificadores separados.

El resultado obtenido es un conjunto de cuatro características texturales y dos características cromáticas que minimiza las tasas de error al evaluar las mismas imágenes con las que se entrenó el algoritmo.

Este análisis se ha realizado con la mayor exhaustividad posible, sumando más de 250 horas de procesamiento.

No obstante, la reproductibilidad de estos resultados es complejo incluso empleando las mismas imágenes, ya que existen factores que pueden alterar los resultados. La metodología empleada en este trabajo comienza con una pre-selección de imágenes, eliminando aquellas que contengan objetos extraños. Además, para el estudio de características cromáticas se ha extraído muestras de imágenes. Estos dos hechos invalidan la repetibilidad de los resultados. Adicionalmente se minoraron las características cromáticas al 5%, por lo que existe una variabilidad intrínseca dentro del mismo estudio.

Pese a las limitaciones, los resultados se consideran suficientemente generales. Las mejores características para los clasificadores de mínima distancia y de redes neuronales, cuyos resultados son significativamente mejores que los del clasificador K-means, contienen características en común. Esto refleja una coherencia en los resultados obtenidos.

La fase de Segmentación toma como entrada los resultados de la fase de Análisis para construir los prototipos de cada clasificador y realizar una segmentación de imágenes SWIMCAT, así como imágenes tomadas con medios propios.

Los resultados obtenidos son imágenes segmentadas con cierto criterio y por tanto se considera un buen resultado, si bien las segmentaciones se esperaba una mayor similitud entre los tres clasificadores mixtos.

La fase de Implementación define los procesos de usuario e incluye, por un lado, un interfaz de usuario y, por otro, dos aplicaciones diferentes de segmentación con imágenes "en vivo".

El interfaz de usuario incluye todos los procesos de análisis y segmentación bajo una misma ventana. Este proceso presentación el proyecto como un producto terminado que funciona y cumple los requisitos para los que fue diseñado.

El proyecto culmina con el Sistema Automático de Identificación de Nubes, esto es, la aplicación de segmentación "en vivo", en dos dispositivos de naturaleza muy diferente sobre plataformas separadas y alcances desiguales.

Así, la estación terrestre tiene implementadas las identificaciones con características textural, cromática y mixta con cuadrículas. Su funcionamiento es bueno, si bien la falta de procesamiento de la plataforma hardware es insuficiente para la implementación del procesamiento textural completo, resultado una identificación algo pixelada.

Finalmente, la aplicación móvil incluye únicamente la segmentación cromática, ya que el entorno de desarrollo JAVA no admite un procesamiento demasiado lento debido a su estabilidad. Por tanto, se han optimizado todos los algoritmos para trabajar sobre cálculos matriciales obteniendo al fin una aplicación funcional para la identificación de nubes sobre Android.

11 | Referencias

- Libros:
  - Gonzalez, W., & Woods, R. E. (2004). "*Digital Image Processing Using MATLAB*". Third New Jersey: Prentice Hall.
  - Shaefer, V. J. "*Guía de campo de la atmósfera. Con 336 ilustraciones en blanco y negro y 32 en color*". Publicador: Omega. Barcelona. 1983.
  - Laidlaw, D. H., & Vilanova, A. (Eds.). (2012). "*New developments in the visualization and processing of tensor fields*". Springer Science & Business Media.
  - Collet, C., Chanussot, J., & Chehdi, K. (2010). "*Multivariate image processing*".
  - Duda, R. O., Hart, P. E., & Stork, D. G. (2012). "*Pattern classification*". John Wiley & Sons.
  - Morales, R. R., & Azuela, J. H. S. (2012). "*Procesamiento y análisis digital de imágenes*". Alfaomega.
- Artículos:
  - Arahal, M.R., Cepeda, A., Camacho, E.F. (2002). "*Input variable Selection for Forecasting Models*". 15th IFAC World Congress on Automatic Control, Barcelone, Spain.
  - Arahal, M.R., Berenguel, M., Camacho, E.F., Pavón, F. (2009). "*Selección de variables en la predicción de llamadas en un centro de atención telefónica*". Revista Iberoamericana de Automática e Informática Industrial RIAI 6 (1), 94-104.
  - Clausi, D. A. (2002). "*An analysis of cooccurrence texture statistics as a function of grey level quantization*", Canadian Journal of remote sensing, 28(1), pp 45-62.
  - Dev, S., Lee, Y. H. y Winkler, S. (2015) "*Categorization of cloud image patches using an improved texton-based approach*", Image Processing (ICIP), 2015 IEEE International Conference on, pp 422-426.
  - Dev, S., Lee, Y. H., y Winkler, S. (2014) "*Systematic study of color spaces and components for the segmentation of sky/cloud images*", IEEE International Conference on Image Processing (ICIP), pp. 5102-5106.
  - Fodor, I. K. (2002). "*A survey of dimension reduction techniques*".
  - Ford, A. y Roberts, A. (1998). "*Colour Space Conversions*".
  - Haralick, R. M. (1979) "*Statistical and structural approaches to texture*", Proceedings of the IEEE, 67(5), pp 786-804.
  - Mantelli Neto, S. L., von Wangenheim, A., Pereira, E. B., y Comunello, E. (2010). "*The use of Euclidean geometric distance on RGB color space for the classification of sky and cloud patterns*". Journal of Atmospheric and Oceanic Technology, 27(9), pp 1504-1517.

- Medrano, M., Gil, A., y Martorell, I., y Potau, X., y Cabeza, L.F., (2010) "*State of the art on high-temperature thermal energy storage for power generation. Part 2-Case studies*", Renewable and Sustainable Energy Reviews, 14 (1), pp 56-72. Elsevier.
- Pawlowski, A., Mendoza, J.L., y Guzman, J.L., y Berenguel, M., y Acien, F.G., y Dormido, S., (2015) "*Selective pH and dissolved oxygen control strategy for a raceway reactor within an event-based approach*", Control Engineering Practice, 44, pp 209-218.
- Prasad, V. S. N., & Domke, J. (2005). "*Gabor filter visualization*". J. Atmos. Sci, 13.
- Soh, L. K., y Tsatsoulis, C. (1999). "*Texture analysis of SAR sea ice imagery using gray level co-occurrence matrices*", Geoscience and Remote Sensing, IEEE Transactions on, 37(2), pp 780-795.

- Base de datos de imágenes:
  - Singapore Whole sky IMaging CATegories (SWIMCAT) Database
- Páginas web:
  - http://appdesignbook.com/es/contenidos/diseno-visual-apps-nativas/
- Artículos relacionados con el proyecto global CODISOL:
  - Arahal, Cirre, C.M., Berenguel, M. (2007). "*Serial grey-box model of a stratified thermal tank for hierarchical control of a solar plant*". Solar Energy 82 (5), 441-451.
  - Limon, D., Alvarado, I., Alamo, T., Ruiz, M., Camacho, E.F. (2008). "*Robust control of the distributed solar collector field ACUREX using MPC for tracking*". IFAC Proceedings Volumes 41 (2), 958-963.
  - Schwartz, J.D., Arahal, M.R., Rivera, D.E., Smith, K.D. (2009). "*Control-relevant demand forecasting for tactical decision-making in semiconductor manufacturing supply chain management*". IEEE Transactions on semiconductor manufacturing 22 (1), 154-163.

12 | Glosario

Abreviatura	Descripción
CBR	Razonamiento Basado en Casos
CIE	Commission Internationale de l'Eclairage
CMY(K)	Cyan Magenta Yellow (Black)
CPU	Central Processing Unit
CRT	Cathode Ray Tube
DGE	Distancia Geométrica Euclídea
FA	Análisis de Factor
GEA	Algoritmos Genéticos y Evolucionarios
GLCM	Matriz de Coocurrencia de Niveles de Gris
GPIO	General Purpose Input/Output
GPU	Graphics Processing Unit
GTM	Mapeado Topográfico Generativo
HSV	Hue, Saturation, Value
ICA	Análisis de Componente Independiente
kNN	k-Vecino más cercano
KSOM	Mapas auto-organizados de Kohonen
LVQ	Aprendizaje por cuantificación vectorial
MDS	Escalado Mutidimensional
NTSC	National Television System Committee
OMM	Organización Mundial de Meteorología
PAL	Phase Alternating Line
P&ID	Process and Instrumentation Diagram
PP	Búsqueda de Proyección
RGB	Red, Green, Blue
RNA	Redes Neuronales Artificiales
SDRAM	Synchronous Dynamic Random Access Memory
SGLD	Spatial Grey Level Dependence
SoC	System on Chip
SWIMCAT	Singapore Whole sky IMaging CATegories
TFM	Trabajo Fin de Máster
USB	Universal Serial Bus
WMO	World Meteorological Organization

Anexo 1. Conversión entre espacios de color

CÁLCULO DE LA CARACTERÍSTICA CROMA

R' = R/255
G' = G/255
B' = B/255

C_{max} = max(R', G', B')
C_{min} = min(R', G', B')
$\Delta = C_{max} - C_{min}$

Siendo:

$Croma = \Delta$

CONVERSIÓN RGB a HSV

Cálculo del Tono (H):

$$H = \begin{cases} 0°, \Delta = 0 \\ 60° \cdot \left(\frac{G' - B'}{\Delta} mod 6\right), C_{max} = R' \\ 60° \cdot \left(\frac{B' - R'}{\Delta} + 2\right), C_{max} = G' \\ 60° \cdot \left(\frac{R' - G'}{\Delta} + 4\right), C_{max} = B' \end{cases}$$

Cálculo de la saturación (S):

$$S = \begin{cases} 0, C_{max} = 0 \\ \frac{\Delta}{C_{max}}, C_{max} \neq 0 \end{cases}$$

Cálculo del Valor (V):

$$V = C_{max}$$

CONVERSIÓN RGB a YIQ

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.595716 & -0.274443 & 0.321263 \\ 0.211456 & -0.522591 & 0.311135 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

CONVERSIÓN RGB a L*a*b*

No existen fórmulas sencillas para la conversión entre valores RGB o CMYK y L*a*b*, ya que los modelos de color RGB y CMYK dependen del dispositivo. Los valores RGB o CMYK deben ser transformados a un espacio de color absoluto específico, tal como sRGB o RGB de Adobe. Estos espacios serán dependientes del dispositivo, a diferencia de los datos resultantes de la transformación, permitiendo que estos datos sean transformados al espacio de color CIE 1931 y luego en L*a*b*.

$$R_{lineal} = \begin{cases} \dfrac{R_{standard}}{12{,}92}, R_{standard} \leq 0{,}04045 \\ \left(\dfrac{R_{standard} + 0{,}055}{1{,}055}\right)^{2{,}4}, R_{standard} > 0{,}04045 \end{cases}$$

La conversión de los planos G_{lineal} y B_{lineal} es similar a R_{lineal}.

Posteriormente se convierte al espacio intermedio XYZ.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0{,}4124 & 0{,}3576 & 0{,}1805 \\ 0{,}2126 & 0{,}7152 & 0{,}0722 \\ 0{,}0193 & 0{,}1192 & 0{,}9505 \end{bmatrix} \begin{bmatrix} R_{lineal} \\ G_{lineal} \\ B_{lineal} \end{bmatrix}$$

Finalmente el espacio de color L*a*b se calcula como función no lineal (siendo X_n, Y_n y Z_n los valores triestímulos CIE XYZ del punto blanco de referencia):

$$L^* = 116 \cdot f\left(\frac{Y}{Y_n}\right) - 16$$

$$a^* = 500 \cdot \left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right]$$

$$b^* = 200 \cdot \left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right]$$

donde:

$$f(t) = \begin{cases} t^{1/3}, para\ t > \left(\dfrac{6}{29}\right)^3 \\ \dfrac{1}{3} \cdot \left(\dfrac{29}{6}\right)^2 \cdot t + \dfrac{4}{29}, para\ otro\ valor \end{cases}$$

Anexo 2. Fórmulas de características texturales

Formulación iterativa

Tabla 20. Fórmulas de características texturales - Iterativo

Característica	Fórmula
Energía	$f_1 = \sum_i \sum_j p(i,j)^2$
Contraste	$f_2 = \sum_{i=1}^{N_g-1} n^2 \left\{ \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \right\}_{\lvert i-j \rvert = n}$
Correlación	$f_3 = \frac{\sum_i \sum_j (ij) p(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y}$
Suma de cuadrados: Varianza	$f_4 = \sum_i \sum_j (i-\mu)^2 p(i,j)$
Homogeneidad	$f_5 = \sum_i \sum_j \frac{1}{1+(i-j)^2} p(i,j)$
Media de la suma	$f_6 = \sum_{i=2}^{2 \cdot N_g} i \cdot p_{x+y}(i)$
Varianza de la suma	$f_7 = \sum_{i=2}^{2 \cdot N_g} (i - f_8)^2 \cdot p_{x+y}(i)$
Entropía de la suma	$f_8 = -\sum_{i=2}^{2 \cdot N_g} p_{x+y}(i) \cdot \log(p_{x+y}(i))$
Entropía	$f_9 = -\sum_i \sum_j p(i,j) \cdot \log(p(i,j))$
Varianza de la diferencia	$f_{10} = variance(p_{x-y})$
Entropía de la diferencia	$f_{11} = -\sum_{i=0}^{N_g-1} p_{x-y}(i) \cdot \log(p_{x-y}(i))$
Medidas informativas de correlación (1)	$f_{12} = \frac{HXY - HXY1}{\max(HX, HY)}$
	Donde: $HXY = -\sum_i \sum_j p(i,j) \cdot \log(p(i,j))$ $HXY1 = -\sum_i \sum_j p(i,j) \cdot \log(p_x(i) \cdot p_y(j))$

Característica	Fórmula
	$HXY2 = -\sum_i \sum_j p_x(i) \cdot p_y(j) \cdot \log(p(i,j))$
Medidas informativas de correlación (2)	$f_{13} = \left[1 - e^{-2 \cdot (HXY2 - HXY)}\right]^{1/2}$
Coeficiente de máxima correlación	$f_{14} = [segundo\ autovalor\ de\ Q]^{1/2}$
	Donde: $Q(i,j) = \sum_k \frac{p(i,k) \cdot p(j,k)}{p_x(i) \cdot p_y(j)}$
Autocorrelación	$f_{15} = \sum_i \sum_j (ij) \cdot p(i,j)$
Disimilaridad	$f_{16} = \sum_i \sum_j \lvert i-j \rvert \cdot p(i,j)$
Tono de grupo (Cluster shadow)	$f_{17} = \sum_i \sum_j \left(i + j - \mu_x - \mu_y\right)^3 \cdot p(i,j)$
Relevancia de grupo (Cluster Prominence)	$f_{18} = \sum_i \sum_j \left(i + j - \mu_x - \mu_y\right)^4 \cdot p(i,j)$
Probabilidad máxima	$f_{19} = MAX[p(i,j)]$
Diferencia inversa	$f_{20} = \sum_{i,j=1}^{G} \frac{C_{ij}}{1 + \lvert i-j \rvert}$
Diferencia inversa normalizada	$f_{21} = \sum_{i,j=1}^{G} \frac{C_{ij}}{1 + \lvert i-j \rvert^2 / G^2}$
Momento de la diferencia inversa normalizada	$f_{22} = \sum_{i,j=1}^{G} \frac{C_{ij}}{1 + (i-j)^2 / G^2}$

Formulación matricial

Notas previas:

- Como paso previo se definen dos tipos de multiplicación entre matrices: la multiplicación matricial, con el símbolo *, y la multiplicación elemento a elemento, con el símbolo ·
- En términos de notación se expresarán las potencias de multiplicaciones elemento a elemento como: $(A) \cdot (A) = (A)^{(.)2}$
- En términos de anotación, a partir de la definición de la matriz $GLCM_{norm}$ se entenderá GLCM como $GLCM_{norm}$.

Variables auxiliares:

- $glcm_{sum} = [1 \quad \dots \quad 1] * GLCM * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$
- $glcm_{media} = \frac{glcm_{sum}}{\dim(GLCM)^2}$

  donde dim(·) representa la dimensión de GLCM. Nótese que GLCM es cuadrada.

Matrices auxiliares:

- $GLCM_{norm} = \left(\frac{1}{glcm_{sum}}\right) \cdot GLCM$
- $GLCM_{ln} = \ln(GLCM)$

  donde ln(·) representa el logaritmo neperiano elemento a elemento. Desde el punto de vista computacional se trata de un cálculo matricial no recursivo.
- $colT = \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ \dim(GLCM) & \cdots & \dim(GLCM) \end{bmatrix}$
- $filT = \begin{bmatrix} 1 & \cdots & \dim(GLCM) \\ \vdots & \ddots & \vdots \\ 1 & \cdots & \dim(GLCM) \end{bmatrix}$
- $u_x = [1 \quad \dots \quad 1] * (colT \cdot GLCM) * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$
- $u_y = [1 \quad \dots \quad 1] * (filT \cdot GLCM) * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$
- $p_x = GLCM * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$ $\qquad p_y = ([1 \quad \dots \quad 1] * GLCM)^T$
- $h_x = -(p_x^T * \log(p_x))$ $\qquad h_y = -(p_y^T * \log(p_y))$

- $h_{xy1} = [1 \quad \dots \quad 1] * \left(\log(p_x * p_y^T) \cdot GLCM\right) * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$
- $h_{xy2} = [1 \quad \dots \quad 1] * \left(\log(p_x * p_y^T) \cdot (p_x * p_y^T)\right) * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$
- $s_x = \left([1 \quad \dots \quad 1] * \left((colT - u_x \cdot II) \cdot (colT - u_x \cdot II) \cdot GLCM\right) * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}\right)^{0.5}$
- $s_y = \left([1 \quad \dots \quad 1] * \left((filT - u_y \cdot II) \cdot (filT - u_y \cdot II) \cdot GLCM\right) * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}\right)^{0.5}$
- $cor_p = [1 \quad \dots \quad 1] * (colT \cdot filT \cdot GLCM) * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$
- $cor_m = [1 \quad \dots \quad 1] * \left((colT - u_x) \cdot (filT - u_y) \cdot GLCM\right) * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$

Las variables p_{x+y} y p_{x-y} se calculan de manera recursiva ya que no se ha podido encontrar una forma matricial sencilla para su cálculo.

$$plog_{x+y} = \ln(p_{x+y}) \qquad plog_{x-y} = \ln(p_{x-y})$$

Tabla 21. Fórmulas de características texturales - Matricial

Característica	Fórmula
Energía	$f_1 = [1 \quad \dots \quad 1] * GLCM \cdot GLCM * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$
Contraste	$f_2 = [1 \quad \dots \quad 1] * (colT - filT) \cdot (colT - filT) \cdot GCLM * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$
Correlación	$f_3 = \frac{cor_m}{s_x \cdot s_y}$
Suma de cuadrados: Varianza	$f_4 = [1 \quad \dots \quad 1] * \left(GLCM \cdot (colT - glcm_{media})^{(.)2}\right) * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$
Homogeneidad	$f_5 = [1 \quad \dots \quad 1] * (GLCM \cdot (1 + \lvert colT - filT \rvert)^{-1}) * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$
Media de la suma	$f_6 = [2 \quad \dots \quad \dim(GLCM)] * p_{x+y}$
Varianza de la suma	$f_7 = ([2 \quad \dots \quad \dim(GLCM)] - f_8)^{(.)2} * p_{x+y}$
Entropía de la suma	$f_8 = [-1 \quad \dots \quad -1] * p_{x+y} \cdot plog_{x+y}$

Característica	Fórmula
Entropía	$f_9 = [-1 \quad \dots \quad -1] * GLCM \cdot GLCM_{ln} * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$
Varianza de la diferencia	$f_{10} = [0 \quad \dots \quad (\dim(GLCM) - 1)]^{(.)2} * p_{x-y}$
Entropía de la diferencia	$f_{11} = [-1 \quad \dots \quad -1] * p_{x-y} \cdot plog_{x-y}$
Medidas informativas de correlación (1)	$f_{12} = \frac{f_9 - h_{xy1}}{\max([h_x, h_y])}$
Medidas informativas de correlación (2)	$f_{13} = \sqrt{1 - e^{-2 \cdot (h_{xy2} - f_9)}}$
Coeficiente de máxima correlación	$f_{14} = \frac{(cor_p - u_x \cdot u_y)}{s_x \cdot s_y}$
Autocorrelación	$f_{15} = cor_p$
Disimilaridad	$f_{16} = [1 \quad \dots \quad 1] * \lvert colT - filT \rvert \cdot GCLM * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$
Tono de grupo (Cluster shadow)	$f_{17} = [1 \quad \dots \quad 1] * (colT + filT - u_x - u_y)^{(.)3} \cdot GLCM * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$
Relevancia de grupo (Cluster Prominence)	$f_{18} = [1 \quad \dots \quad 1] * (colT + filT - u_x - u_y)^{(.)4} \cdot GLCM * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$
Probabilidad máxima	$f_{19} = \max(\max(GLCM))$
Diferencia inversa	$f_{20} = [1 \quad \dots \quad 1] * \left(GLCM \cdot \left(1 + (colT - filT)^{(.)2}\right)^{-1}\right) * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$
Diferencia inversa normalizada	$f_{21} = [1 \quad \dots \quad 1] * \left(GLCM \cdot \left(1 + \frac{\lvert colT - filT \rvert}{\dim(GLCM)}\right)^{-1}\right) * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$
Momento de la diferencia inversa normalizada	$f_{22} = [1 \quad \dots \quad 1] * \left(GLCM \cdot \left(1 + \frac{(colT - filT)^{(.)2}}{\dim(GLCM)^2}\right)^{-1}\right) * \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}$

Anexo 3. Resultados de la segmentación

A3.1 Segmentación textural completa

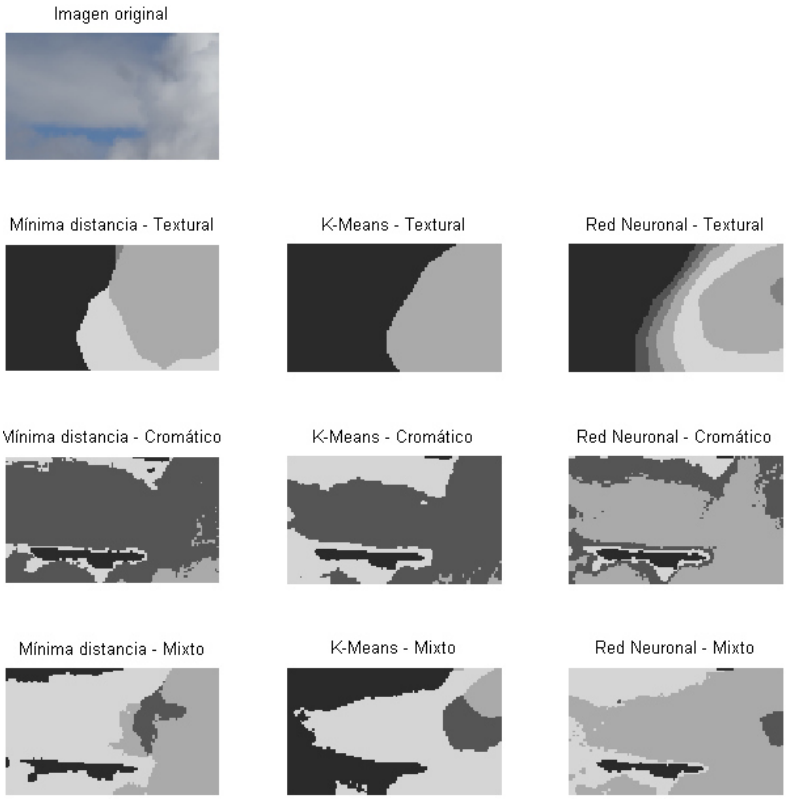


Figura 80. Segmentación con procesamiento textural completa (I)

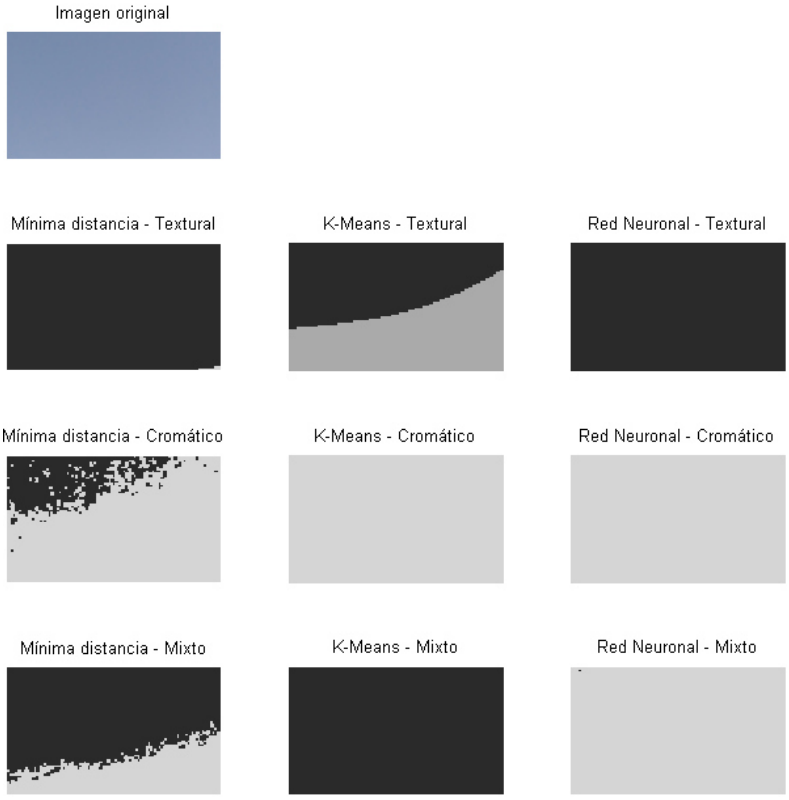


Figura 81. Segmentación con procesamiento textural completa (II)

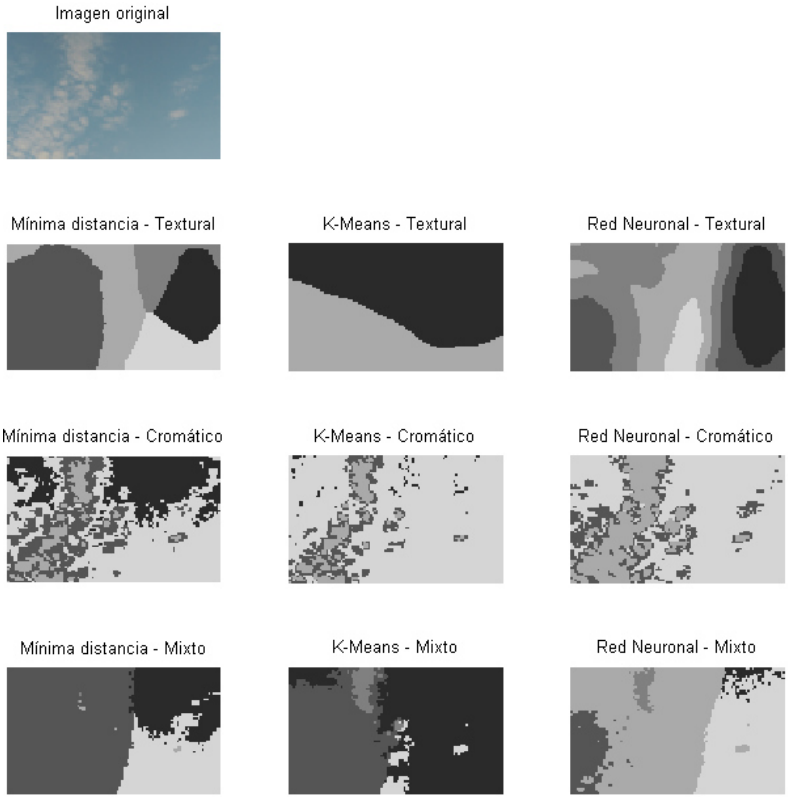


Figura 82. Segmentación con procesamiento textural completa (III)

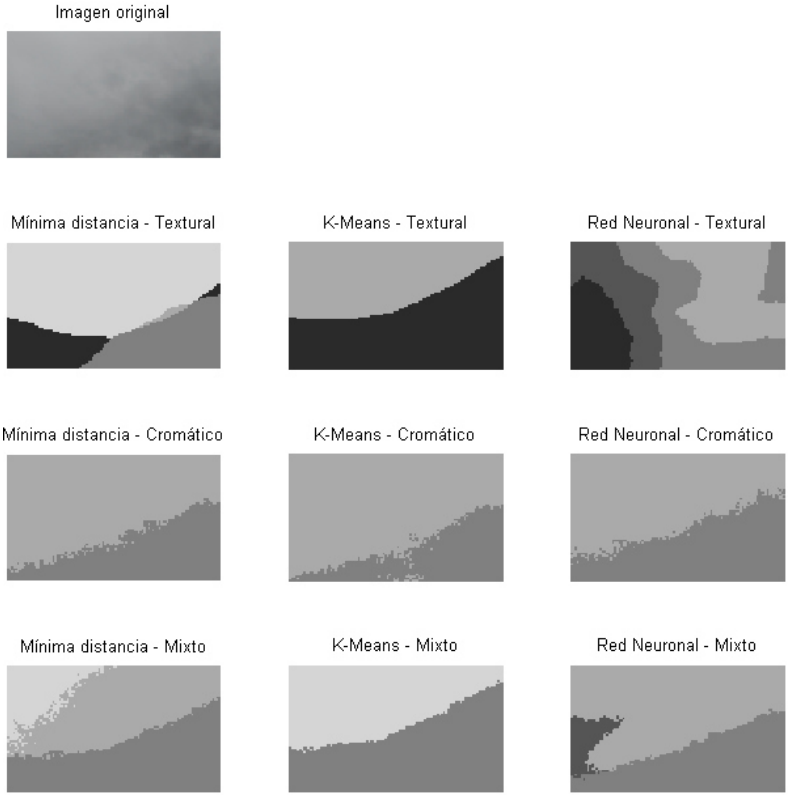


Figura 83. Segmentación con procesamiento textural completa (IV)

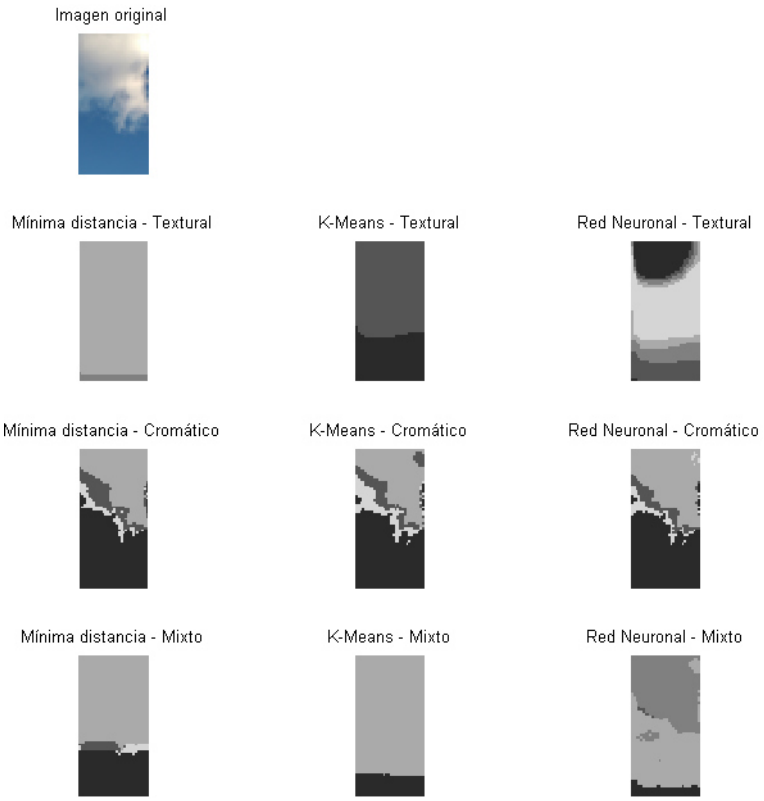


Figura 84. Segmentación con procesamiento textural completa (V)

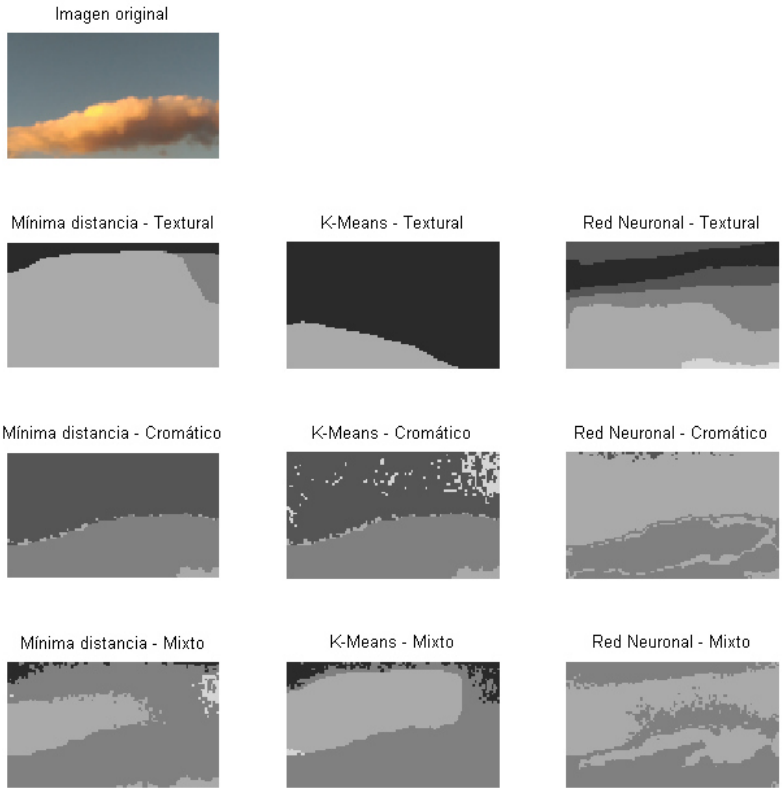


Figura 85. Segmentación con procesamiento textural completa (VI)

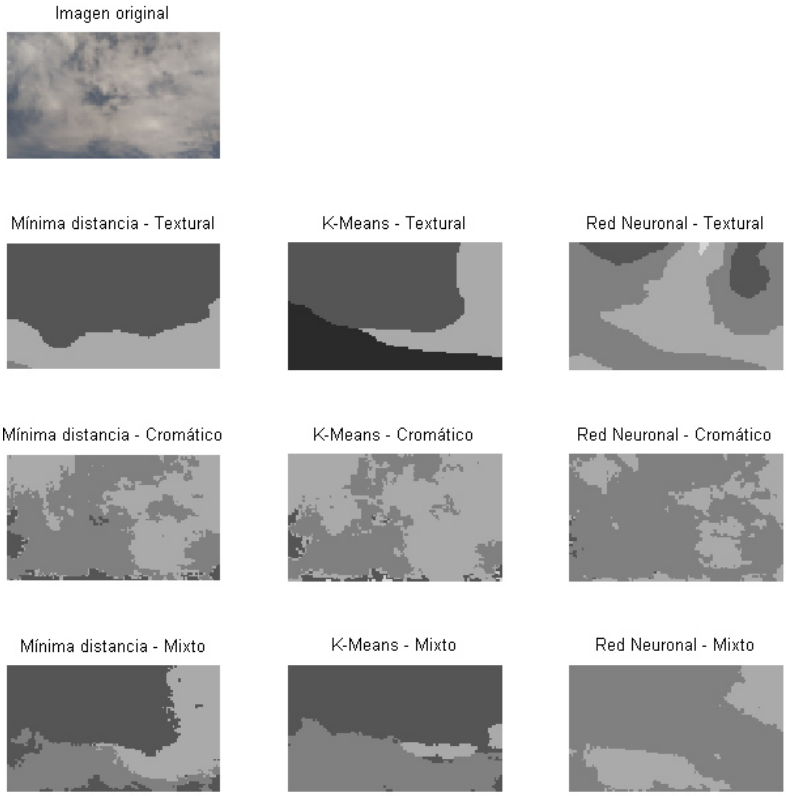


Figura 86. Segmentación con procesamiento textural completa (VII)

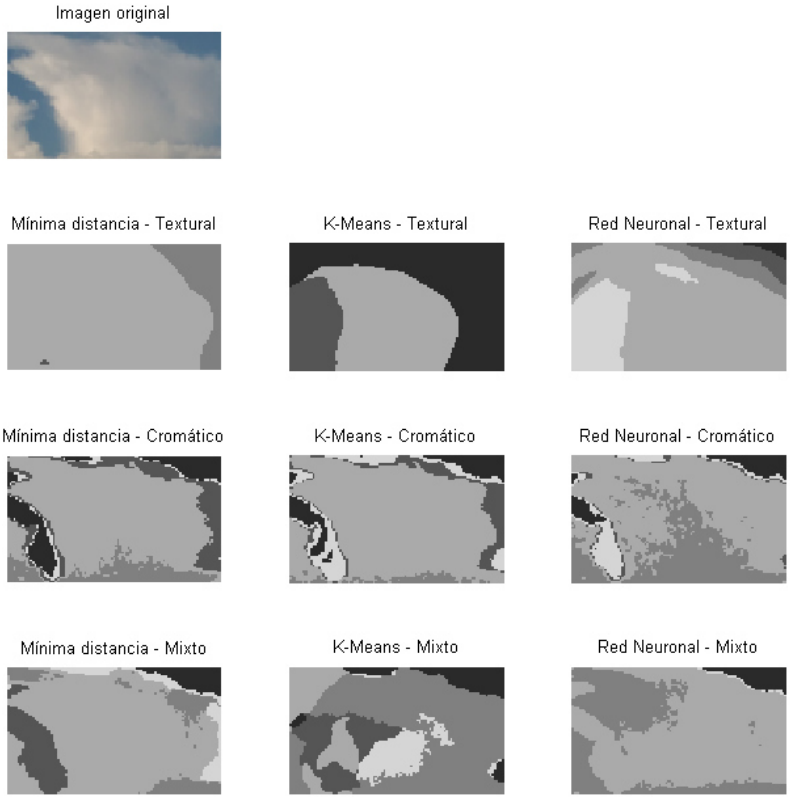


Figura 87. Segmentación con procesamiento textural completa (VIII)

A3.2 Segmentación textural en cuadrícula

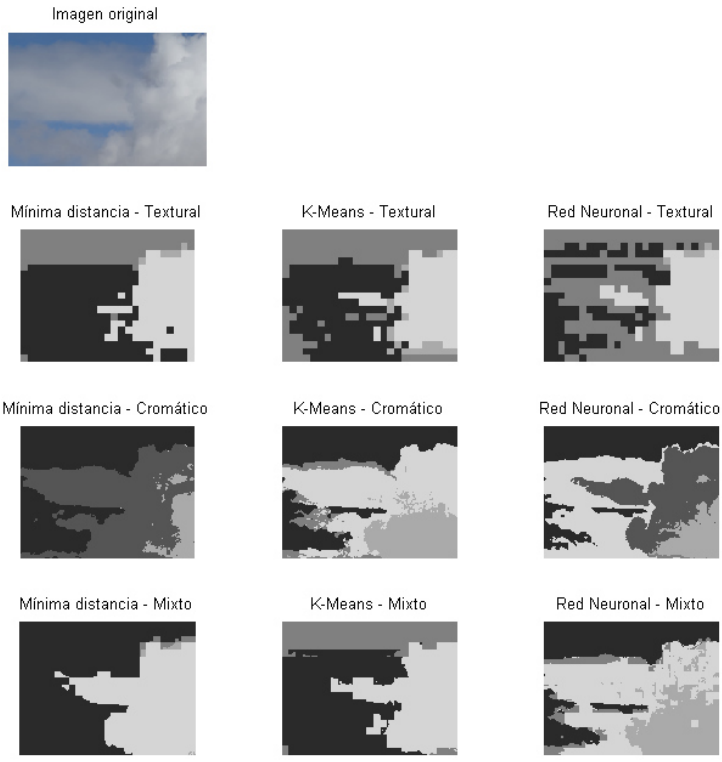


Figura 88. Segmentación con procesamiento textural en cuadrícula (I)

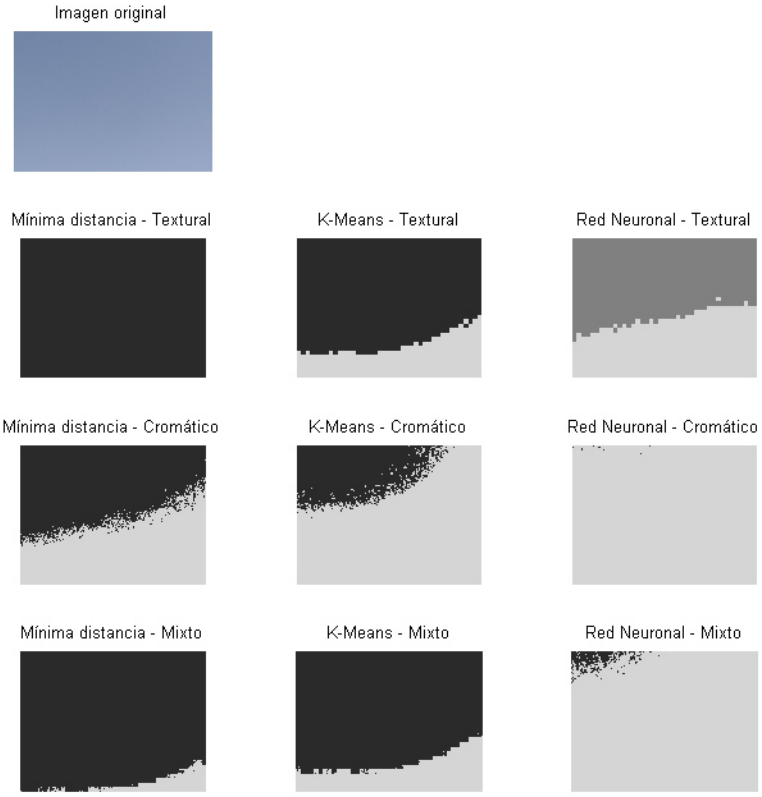


Figura 89. Segmentación con procesamiento textural en cuadrícula (II)

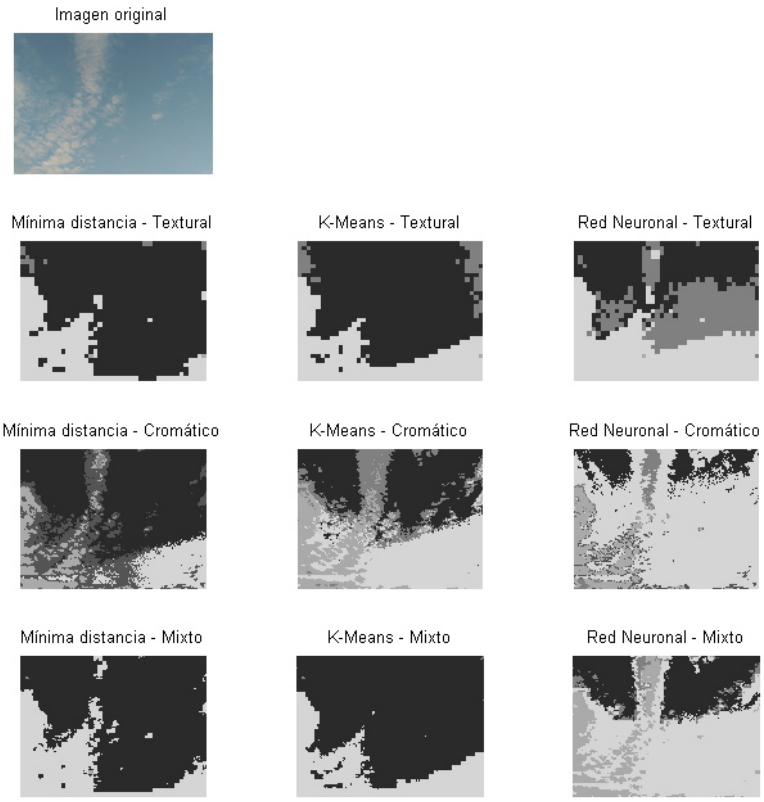


Figura 90. Segmentación con procesamiento textural en cuadrícula (III)

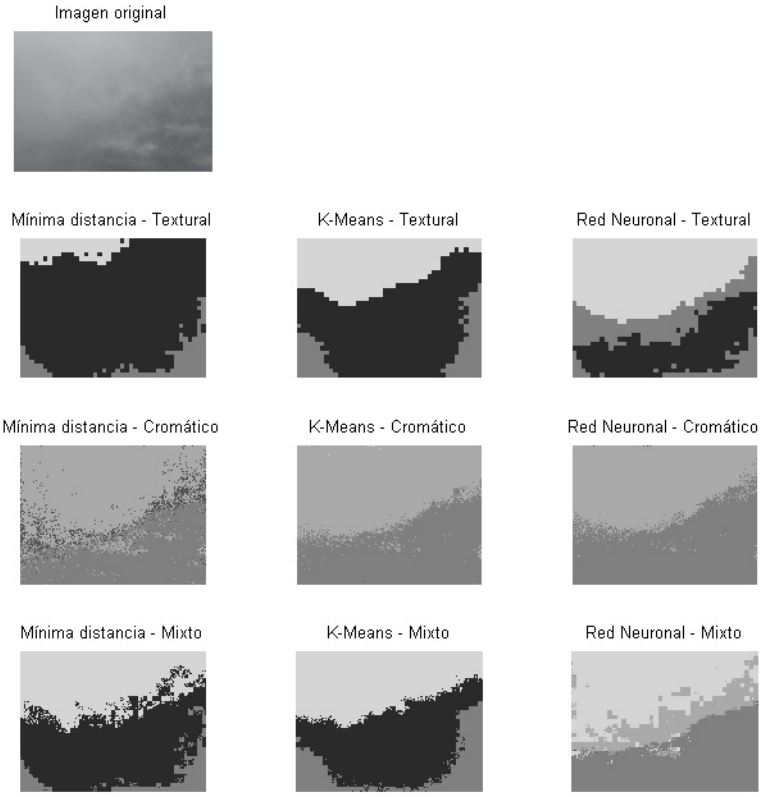


Figura 91. Segmentación con procesamiento textural en cuadrícula (IV)

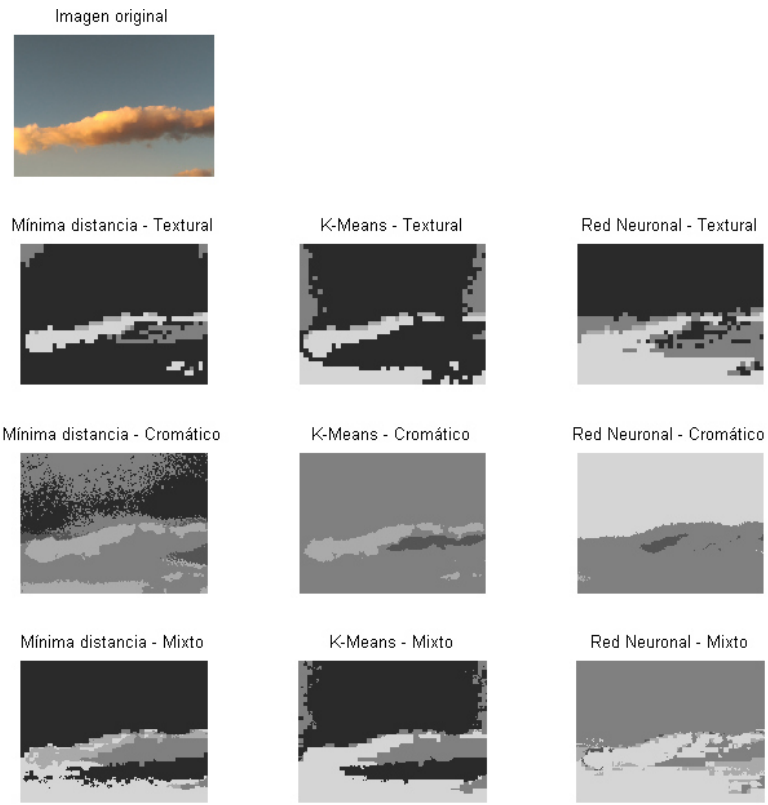


Figura 92. Segmentación con procesamiento textural en cuadrícula (V)

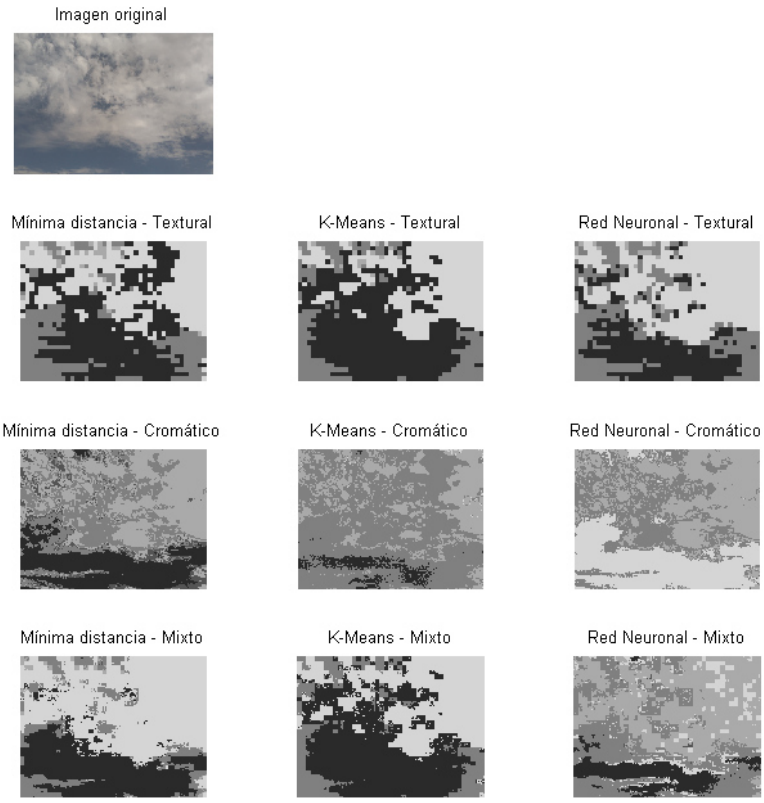


Figura 93. Segmentación con procesamiento textural en cuadrícula (VI)

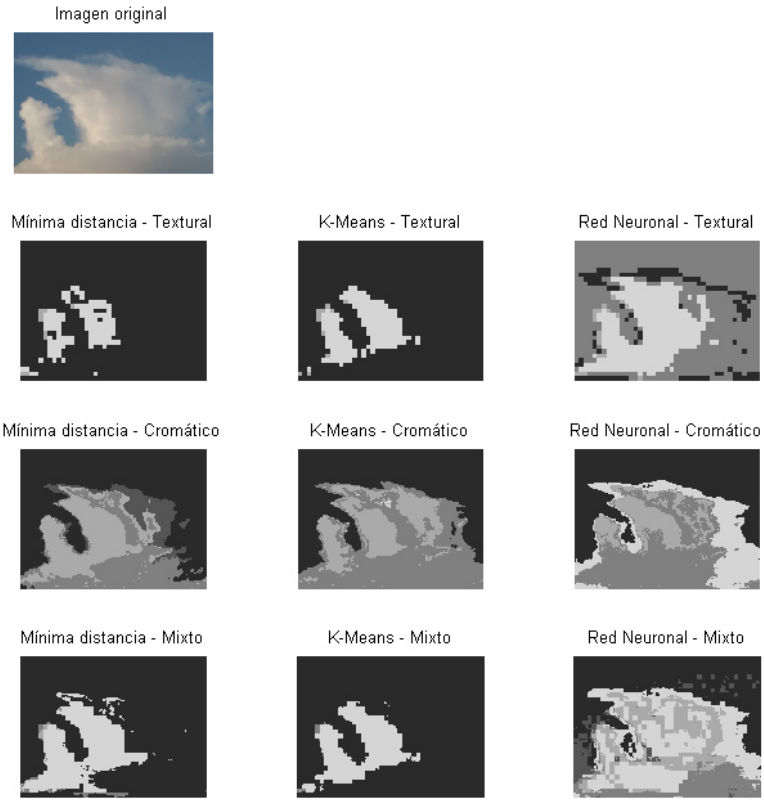


Figura 94. Segmentación con procesamiento textural en cuadrícula (VII)

Anexo 4. Publicaciones realizadas

Durante el desarrollo de este proyecto se ha realizado la publicación de los resultados del análisis con el clasificador de Mínima Distancia, localizable en Google Scholar:

- Padial, E. F. C., & Arahal, M. R. CARACTERIZACION CONJUNTA CROMATICA-TEXTURAL EN LA IDENTIFICACION DE NUBES.

CARACTERIZACIÓN CONJUNTA CROMÁTICA-TEXTURAL EN LA IDENTIFICACIÓN DE NUBES

Eduardo Fernández-Cantalejo Padial
Escuela Técnica Superior de Ingeniería de Sevilla, eduardofpadial@gmail.com

Manuel R. Arahal
Escuela Técnica Superior de Ingeniería de Sevilla, arahal@us.es

Resumen

La identificación automática de nubes es un problema que se ha abordado desde diferentes puntos de vista en los últimos años. En particular mediante visión por ordenador usando clasificadores de formas que utilizan características espectrales, texturales y/o cromáticas. El presente estudio afronta el problema sin limitarse a un tipo particular de características. Se realiza una fusión de técnicas buscando el mejor emparejamiento de características abordando cuestiones cruciales como la invariabilidad respecto al ángulo y la diversidad de las muestras.

Palabras clave: GLCM, características cromáticas, clasificación, nubes.

1. INTRODUCCIÓN

La visión artificial es un valor añadido en auge en la industria y sociedad actuales. Sus campos de aplicación se amplían cada día hasta el punto de ser una parte vital en muchas áreas de la automatización.

El caso de la identificación de nubes tiene especial relevancia en diversas aplicaciones relacionadas con la industria solar (producción de energía, edificios, etc.) y en meteorología. En estas aplicaciones la identificación de nubes es un paso previo para la predicción del recurso solar a corto plazo. Al predecir posibles situaciones adversas se puede reaccionar con antelación para evitar o paliar los efectos más perniciosos.

En la literatura se han empleado diversas técnicas para la clasificación automática de nubes, que abarcan desde la distribución de colores de cada píxel individualmente en el espacio RGB [7], de manera sistemática [3] hasta el análisis de texturas mediante matrices de co-ocurrencia de nivel de gris [10]. Todos estos estudios realizan un anÃ¡lisis de cada una de cada técnica individualmente.

El presente artículo estudia la posibilidad de emplear técnicas diferentes y fusionar los resultados para la consecución de mejores resultados a los obtenidas de manera independiente.

Este nuevo enfoque se centra en la caracterización híbrida de características cromáticas conjuntamente con características texturales. Para ello se ha realizado un estudio individual de cada una de ellas para seleccionar las mejores características de cada técnica y posteriormente aplicar la fusión en la técnica mixta cromático-textural.

El objetivo final consiste en la descripción y clasificación de nubes desde una estación con base en la superficie terrestre, en contraposición a las imágenes tomadas desde los satélites. El tipo de nube será determinante para la estimación de la altura en que éstas se forman y desarrollan y de su posible evolución, en dirección y en altura.

2. ANTECEDENTES

Durante las últimas décadas se han desarrollado numerosas técnicas para la clasificación de patrones basados en diferentes características como la forma, el color y la textura. Cada una de estas aproximaciones muestra su conveniencia en función del problema y el tipo de patrón que se aborde. En el caso del reconocimiento e identificación de nubes la forma no será determinante debido a su alta variabilidad, por lo que se descartará este tipo de características. El problema se afronta, por tanto, como la búsqueda de las mejores características cromáticas y texturales que, primero de forma independiente y posteriormente de forma conjunta, mejor describa cada tipo de nube.

El interés de la detección y clasificación de nubes es más notorio en algunas aplicaciones que en otras. Por ejemplo, en el caso de plantas fotovoltaicas las nubes pueden producir una caída instantánea de la radiación solar directa (RSD) que se traduce en una gran perturbación en la producción que ha de ser absorbida por el sistema eléctrico [6]. En el caso de plantas termosolares el efecto puede ser aliviado por la inercia térmica del sistema, sobre todo si se usa acumulación de calor en aceite, sales u otro medio [8]. Para otras aplicaciones la radiación solar indirecta (RSI) juega un papel importante por lo que el paso de nubes dispersas puede tener escasas consecuencias [9], por

Cuadro 1: Clasificación de nubes SWIMCAT conforme WMO.

Etiqueta SWIMCAT	Nro. imágenes	Tipo de nube conforme WMO	Descripción
A	224	Cielo despejado	Sin nubes o nubosidad $< 10\,\%$
B	89	Cirrocúmulos y altocúmulos	Patrón de pequeñas nubes
C	251	Cúmulonimbus y nimboestratos	Nubes oscuras, cubiertas y densas
D	135	Cúmulos	Nubes bajas e hinchadas con bordes definidos, blancas o gris muy claras
E	85	Estrato y altoestratos	Capa de nubes medias y bajas uniformes usualmente cubierta y gris
		Estratocúmulos	Capa de nubes medias y bajas, rota o completa, blanca o gris
		Cirros y cirroestratos	Nubes altas y finas de humo o cubriendo el cielo

lo que basta con conocer la nubosidad general. En todos los casos la predicción de la nubosidad y el estudio de su impacto tienen un gran interés para optimizar la operación de la planta [4].

3. METODOLOGÍA

El análisis se planteará como una búsqueda exhaustiva y sistemática de las características de manera que tengan el mayor grado de independencia del tipo de imagen y las condiciones de luminosidad en el momento de la obtención de éstas. Por ello se empleará un método basado en los siguientes principios:

- Colección de imágenes extensa, de forma que se incluya el mayor número de escenarios posibles en el estudio, dentro de una normalidad de funcionamiento, esto es, en los momentos entre el amanecer y el atardecer, pero sin incluir éstos.
- Normalización estadística mediante la eliminación de la media y ponderado con la varianza del conjunto de características obtenidas.
- Clasificador, se aplicará únicamente el clasificador de mínima distancia, con un desempeño aceptable para la evaluación del potencial de mejora del método presentado. Estudios posteriores evaluarán el mejor clasificador a emplear.

Una vez definidos los elementos comunes se presentarán tres técnicas para la clasificación de nubes: las dos primeras conforme a técnicas incluidas en estudios pasados basados en tratar las características cromáticas y texturales de forma separada y posteriormente se presentará un método que integra ambos tipos de características para su uso en un mismo clasificador.

La evaluación de los resultados se ha realizado mediante el procesamiento de las imágenes de la base de datos SWIMCAT que previamente se han empleado para su entrenamiento, ya que el objetivo de este trabajo es el estudio de viabilidad del método de clasificación con características de diferente naturaleza y se considera suficiente con este tipo de validación.

A continuación se profundiza en cada uno de estos aspectos del análisis.

3.1. COLECCIÓN DE IMÁGENES

La calidad y dimensión de la muestra de imágenes debe ser suficientemente extensa y significativa para que los resultados puedan considerarse válidos y reproducibles. Se busca la universalización de los resultados mediante la exhaustividad en el tipo y cantidad muestras en el conjunto de estudio.

Asimismo debe escogerse un conjunto de tipos de nube que sea capaz de satisfacer la funcionalidad para la que se realiza el estudio. Los resultados de este estudio pretende la estimación de la altura de las nubes en el cielo por lo que la clasificación de la que se parte responde a aquellas nubes que tengan un rango de altura específico.

Este estudio ha empleado la base de datos SWIMCAT de nubes (Singapore Whole sky IMaging CATegories), empleada en [2]. Esta colección de fotografías contiene 784 imágenes clasificadas por personal experto del Servicio Meteorológico de Singapore en cinco categorÃas como las mostradas en la Figura 1 y conforme a lo recogido en la Tabla 1. Este conjunto se considera como muestra válida para un estudio en profundidad.

El conjunto de nubes que forman parte de la base de datos SWIMCAT es suficientemente extensa y se divide en cinco categorías. Sin embargo, esta base de datos no considera todas las categorías de-

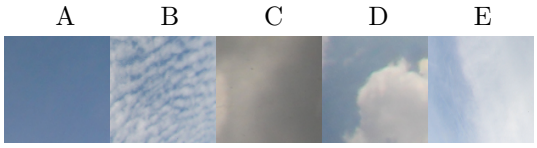


Figura 1: Tipos de nube A, B, C, D y E

finidas por la World Meteorological Organization (WMO) en su Atlas Internacional de Nubes.

Mediante la identificación de estas clases de nubes se podrá realizar una estimación de la altura de las nubes.

3.2. NORMALIZACIÓN

El tratamiento de estas distancias se realiza con sus valores en el intervalo [0,255] [3]. Sin embargo, para poder realizar la fusión posterior con las características texturales se necesitará la normalización de los valores entre [0,1]. Sin embargo, también se desea la obtención de características robustas e independientes de las condiciones externas durante la toma de imágenes, por lo que en este punto se realizarán dos tipos de normalización para estimar cuál resultará con un mejor clustering. Normalización estadística: $DistN_{est} = \frac{Dist-\mu}{\sigma}$

3.3. CLASIFICADOR DE MÍNIMA DISTANCIA

Se ha optado por el clasificador más sencillo para validar los resultados ya que su funcionamiento será tanto más correcto cuanto más separados se encuentren los clusters entre sí. De esta manera se está positivizando el distanciamiento entre las categorías de nubes.

Sea E un conjunto dotado de una (pseudo)métrica d : $E \times E \rightarrow R$. Supongamos que se dan m prototipos $p_1, p_2, \cdots, p_m \in E$, tales que $p_i \in c_i$ representa a la clase c_i $(i = 1, 2, \cdots, m)$, es decir, un prototipo por clase. Una clasificación por distancia mínima se define para cada $x \in E$ como $x \in ci \Leftrightarrow d(x, p_i) \leq d(x, p_j); j = 1, 2, \cdots, m$ (1.4) donde d es la (pseudo)métrica definida en E.

En este tipo de clasificador, la fase de aprendizaje consiste únicamente en la elección de un buen representante o prototipo de cada clase en el conjunto de prototipos que pertenecen a la misma. Normalmente el representante elegido suele ser aquel que se encuentra más centrado dentro de la distribución de los prototipos en la clase. La Figura 2 muestra la clasificación de un punto en un espacio con tres clases diferentes.

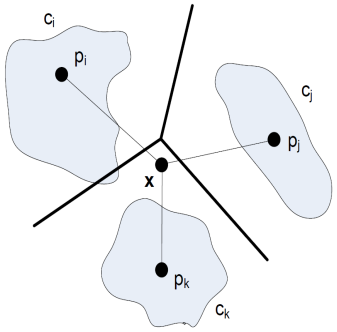


Figura 2: Clasificador de mÃnima distancia

3.4. FAMILIAS DE CARACTERÍSTICAS

3.4.1. Características cromáticas

Las características cromáticas son a menudo consideradas como poco robustas debido que son sensibles a cambios en las condiciones lumínicas y de escala, entre otros aspectos. Sin embargo, hay estudios que emplean estas técnicas para la identificación de nubes con éxito y que se revisará en este estudio con una base de datos de imágenes de gran población y diversidad.

Toda imagen en color puede expresarse en diversos espacios de color de forma que el color se distribuye de forma diferente en cada uno de ellos. Cada uno de los planos de un espacio de color se expresa como escala de grises.

Un estudio exhaustivo debe cubrir todas las posibilidades. Por ello, se emplearán todos los espacios de color descritos y empleados en estudios similares, esto es, se considerarán los siguientes planos de color: RGB, HSV, YIQ, L*a*b* y los derivados (R-B), (R/B), (B-R/B+R) y croma, donde éste último se calcula como C = máx(R,G,B) - mín(R,G,B). Cada uno de estos 16 planos de color constituye una de las 16 características cromáticas de píxel en cada imagen.

Entre las diferentes técnicas (índice de Pearson, Análisis de Componentes Principales, ...) se ha empleado el análisis de la Distancia Geométrica Euclídea (DGE) a la diagonal del espacio de color como descriptor cromático, pues se busca evitar generalizaciones que puedan distorsionar las características individuales de cada punto de la imagen.

El análisis de la DGE consiste en calcular la distancia de un píxel para cada trupla de las 16 características previamente definidas a la diagonal de un cubo formado por el máximo valor en cada uno de estas características, tal como muestra la Figura 3.

Cuadro 2: Relación y fuente de las características texturales.

Etiqueta	Abrev.	Característica	Haralick 1973	Soh 1999	Clausi 2002
1	ENE	Segundo momento angular (Energía) (Uniformidad)	1	1	2
2	CON	Contraste (Inercia)	2	2	5
3	COR	Correlación	3	3	8
4	SCV	Suma de cuadrados: Varianza	4		
5	HOM	Momento de la diferencia inversa (Homogeneidad)	5	4	7
6	MDS	Media de la suma	6		
7	VDS	Varianza de la suma	7		
8	ENS	Entropía de la suma	8		
9	ENT	Entropía	9	5	3
10	VDD	Varianza de la diferencia	10		
11	EDD	Entropía de la diferencia	11		
12	MIC1	Medidas informativas de correlación (1)	12		
13	MIC2	Medidas informativas de correlación (2)	13		
14	CMC	Coeficiente de máxima correlación	14		
15	ACR	Autocorrelación		6	
16	DIS	Disimilaridad		7	4
17	TDG	Tono de grupo (Cluster shadow)		8	
18	RDG	Relevancia de grupo (Cluster Prominence)		9	
19	PDM	Probabilidad máxima		10	1
20	DIN	Diferencia inversa			6
21	INN	Diferencia inversa normalizada			9
22	IDN	Momento de la diferencia inversa normalizada			10

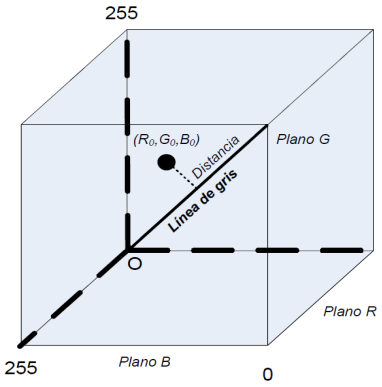


Figura 3: Distancia de un píxel en el espacio RGB

El cálculo de la distancia se basa en sencillas reglas geométricas, obteniendo la fórmula siguiente[7]:

$$Dist \quad = |\overline{D}| \sin\left(\arccos\left(\frac{|\overline{C}-\overline{D}|-|\overline{D}|-|\overline{C}|}{-2|\overline{D}||\overline{C}|}\right)\right) \quad (1)$$

donde de la Figura 4 se obtiene que los módulos:
$|\overline{D}| = \sqrt{255^2 + 255^2 + 255^2} = 441{,}67$

$|\overline{C}| = \sqrt{R^2 + G^2 + B^2}$

La distancia de cada pixel a la línea de gris para cada trupla de planos de color puede tratarse como una característica independiente.

Las distancias normalizadas constituirán las características del vector de características que servirán de entrada al clasificador.

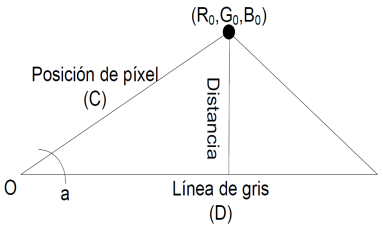


Figura 4: Distancia de un píxel en el espacio RGB

$$Vect = [Dist_{RGB}, Dist_{RGH}, ..., Dist_{HSV}, ...] \quad (2)$$

Cuando se emplean gran cantidad de datos estadísticamente distribuidos, es usual descartar los datos inferiores al cuantil 25 % y superiores al 75 %, sobre todo cuando se emplearán estos datos para generar los prototipos de un clasificador basado en distancias. Los límites intercuantiles se han calculado como:

- Distancia intercuantil: $IQD = Q_3 - Q_1$
- Límite inferior: $IQI = Q_1 - 1{,}5 \cdot IQD$
- Límite superior: $IQS = Q_3 + 1{,}5 \cdot IQD$

En este estudio se han considerado las 560 truplas de características cromáticas posibles para una muestra de 213.600 píxeles extraídos de las imágenes de la base de datos SWIMCAT. Por tanto, cada trupla supone una característica diferente en el clasificador de mínima distancia. Agrupando varios pares de truplas se mejoran los resultados de clasificación. La Figura 5 muestra las regiones de clasificación generadas por una sola trupla y agrupadas en parejas de truplas.

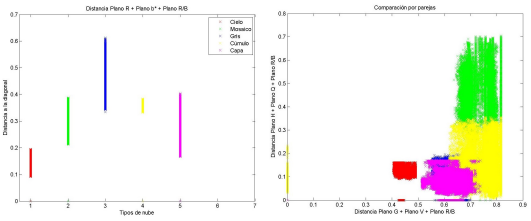


Figura 5: Distribución de clases en 1D y 2D

3.4.2. Características texturales

Las características texturales son magnitudes estadísticas extraídas del valor del entorno de un punto, a diferencia de las caracteristas cromáticas que sólo depende del valor tonal de cada píxel de manera individual.

Así, se asume que la información de contexto textural en una imagen se encuentra definida por la relación espacial general o "media" con la que los tonos de gris tienen unos con otros. Esta información textural se considera adecuadamente especificada por la matriz de frecuencias relativas P_{ij} con la cual dos celdas vecinas separadas una distancia d tienen lugar en la imagen, una con un tono gris i y la otra con un tono de gris j. Tales matrices de frecuencias de tonos de gris con dependencia espacial son función de la relación angular entre las celdas vecinas así como de la distancia entre ellas.

Las características texturales que se han considerado son las definidas por Haralick (1973)[5], Soh (1999) [10] y Clausi (2002) [1] que se construyen como Matriz de Coocurrencia de Niveles de Gris (GLCM), presentadas en la Tabla 2 y cuya formulación matemática se recoge en la Tabla 6.

Como se ha indicado, el resultado durante el tratamiento de textura mediante la matriz de coocurrencia depende de dos parámetros: distancia entre vecinos y ángulo entre vecinos. Este estudio considerará las vecindades de 1 píxel y cuatro ángulos: 0^o, 90^o, 135^o y 270^o, como se indica en la Figura 6.

En este estudio se han considerado todas las imágenes de la base de datos, siendo el entorno

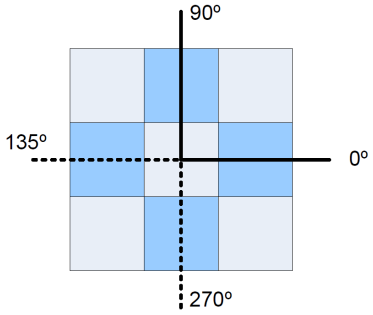


Figura 6: Ángulos de vecindad en la matriz de coocurrencia

del punto la imagen completa, de tamaño 192x192 píxeles.

3.5. TÉCNICA MIXTA DE CLASIFICACIÓN

Las características texturales se han calculado a partir de 534 muestras de imágenes de la base de datos de SWIMCAT, lo que suponen 2136 características si tenemos en cuenta los cuatro ángulos, tal como se muestra en la Figura 6, y extraído cada una de las 22 características texturales, de manera que se obtiene un vector de 2136 x 22 características texturales.

Las caracterÃsticas cromáticas se han extraídos de las mismas muestras que las características texturales, a razón de 1.000 píxeles por cada tipo de nube, por lo que se obtiene una muestra final de 213.600 píxeles y constituyendo un vector de características de dimensiones 213600 x 560 (filas x columnas). Se puede identificar, por tanto, el valor cromático de cada píxel y el valor correspondiente de la textura asociada.

A partir de esta información se pueden crear matrices de características de tal manera que cada columna corresponde a un píxel y cada fila puede significar indistintamente una característica cromática o textural indistintamente. A partir de estas matrices se pueden generar los vectores prototipos de cada que servirán para evaluar cada píxel de cualquier imagen de forma individual.

4. RESULTADOS

4.1. CARACTERÍSTICAS CROMÁTICAS

La aplicación del método de la Distancia Geométrica Euclídea a las 16 características cromáticas definidas anteriormente ofrece 560 combinaciones de distancias posibles, que es el número de características que se han estudiado

Cuadro 3: Resultado de la caracterización cromática.

Características		Tasa de error en clasificación					
Trupla 1	**Trupla 2**	**A**	**B**	**C**	**D**	**E**	**Global**
	R_b*_R/B	0,00 %	82,90 %	12,74 %	13,83 %	42,77 %	26,59 %
G_V_R/B	H_Q_R/B	0,00 %	22,64 %	5,17 %	12,54 %	22,53 %	11,34 %

Cuadro 4: Resultado de la caracterización textural.

Características				Tasa de error en clasificación					
T#1	**T#2**	**T#3**	**T#4**	**A**	**B**	**C**	**D**	**E**	**Global**
-	-	ENT	ACR	2,00 %	16,29 %	15,54 %	38,70 %	25,88 %	16,87 %
-	EDD	ENT	VDS	2,79 %	8,71 %	17,93 %	25,74 %	17,65 %	13,87 %
CON	RDG	ENS	MDS	2,01 %	2,25 %	15,94 %	24,07 %	7,35 %	10,87 %

por cada píxel. Debido al elevado número de variables y la dificultad de segmentar manualmente cada imagen se han tomado muestras significativas de cada tipo de nube para entrenar el clasificador.

En esta investigación se han calculado de forma exhaustiva todas las posibles combinaciones para el clasificador de mínima distancia y se han obtenido tasas de error en la clasificación aceptables para 1 sola dimensión y buenos en caracterizaciones con dos distancias, como se observa en la Tabla 3.

La Figura 5 muestra la distribución de los píxeles caracterizados por lo que se puede observar las distancias que existen entre las cinco clases de nubes. Debe notarse que, debido al solapamiento se obtienen tasas de error altas para las clases B y E.

Algunas de las figuras obtenidas apuntan a la existencia de subclases de nubes, lo cual es positivo para clasificaciones más exigentes que requieras de categorías de nubes no incluidas en este estudio.

4.2. CARACTERÍSTICAS TEXTURALES

Un componente esencial del marco conceptual de la textura es una medida, o más preciso, cuatro medidas muy relacionadas de las que todas las características derivan. Estas medidas son matrices denominadas matrices de tono de gris de vecinos angulares dependientes del espacio. Estas matrices son función de una relación angular entre las celdas adyacentes así como de la distancia entre ellas. El análisis se ha realizado para valores de distancia d=1 en sus cuatro direcciones (0^o, 45^o, 90^o y 135^o).

Las características texturales empleadas son 22 variables que se han estudiado en emparejamientos de hasta cuatro variables, mejorando el rendimiento global con cada nueva característica que se empleaba en el clasificador. La Tabla 4 muestra los resultados para los mejores grupos de dos, tres y cuatro características texturales.

4.3. TÉCNICA MIXTA CROMÁTICO-TEXTURAL

Finalmente se han realizado clasificaciones empleando conjuntamente una selección de características cromáticas y texturales para estudiar si una aplicación conjunta podría obtener mejores resultados que cada una de estas técnicas por separado.

Por ello se ha aplicado un vector de características en el que por cada subimagen I de tamaño $N_x \times N_y$ píxeles, en que se divide una imagen a segmentar, se ha asignado a cada uno de estos píxeles el mismo valor de característica textural correspondiente a la muestra completa.

Las características que se han seleccionado son las que obtuvieron mejor resultado de forma independiente. Este emparejamiento es un candidato firme a obtener un buen resultado, si bien la suma de dos resultados óptimos no garantiza el óptimo global. Del grupo cromáticas: B, L y R/B y del grupo texturales: Contraste, Relevancia de grupo, Media de la Suma y Entropía de la Suma. El resultado es la obtención de una tasa de error global del 8.70 %, con un buen rendimiento de clasificación para todas las clases salvo para los cúmulos que quedaría en un 21,44 %, tal como se muestra en la Tabla 5.

5. CONCLUSIONES

Este estudio pone de manifiesto que la aplicación de características descriptivas de diferentes familias, como son las cromáticas y las texturales, pueden fusionarse para obtener mejores resultados durante la clasificación.

La caracterización cromáticas depende únicamente del valor del píxel, mientras que la caracteriza-

Cuadro 5: Resultado de la técnica mixta.

Características				**Tasa de error en clasificación**					
T#1	**T#2**	**T#3**	**T#4**	**A**	**B**	**C**	**D**	**E**	**Global**
CON	RDG	ENS	MDS	0,25 %	6,05 %	5,05 %	21,44 %	10,61 %	8,70 %
B_L_R/B		H_I_R/B							

ción textural se obtiene de magnitudes estadísticas aplicadas a subimágenes de $N_x \times N_y$ píxeles. La integración de ambos tipos en una misma matriz de características ha posibilitado esta clasificación conjunta.

En este análisis se han revisado técnicas de manera exhaustiva conocidas a una base de datos extensa de imágenes como validación de los resultados previo a la aplicación de la técnica mixta.

Referencias

[1] Clausi, D. A. (2002). "An analysis of co-occurrence texture statistics as a function of grey level quantization", *Canadian Journal of remote sensing, 28(1)* , pp 45-62.

[2] Dev, S., Lee, Y. H. y Winkler, S. (2015) "Categorization of cloud image patches using an improved texton-based approach", *Image Processing (ICIP), 2015 IEEE International Conference on*, pp 422-426.

[3] Dev, S., Lee, Y. H., y Winkler, S. (2014) "Systematic study of color spaces and components for the segmentation of sky/cloud images", *IEEE International Conference on Image Processing (ICIP)*, pp. 5102-5106.

[4] Gallego, A.J., Fele, F., Camacho, E.F. y Yebra, L. (2013) "Observer-based Model Predictive Control of a parabolic-trough field", *Solar Energy, 97*, pp 426-435.

[5] Haralick, R. M. (1979) "Statistical and structural approaches to texture", *Proceedings of the IEEE, 67(5)*, pp 786-804.

[6] Limón, D., Alvarado, I., Álamo, T. Arahal, M.R., y Camacho, E.F., (2008) "Robust control of the distributed solar collector field ACUREX using MPC for tracking", *IFAC Proceedings Volumes, 41(2)*, pp 958-963.

[7] Mantelli Neto, S. L., von Wangenheim, A., Pereira, E. B., y Comunello, E. (2010). "The use of Euclidean geometric distance on RGB color space for the classification of sky and cloud patterns". *Journal of Atmospheric and Oceanic Technology, 27(9)*, pp 1504-1517.

[8] Medrano, M., Gil, A., y Martorell, I., y Potau, X., y Cabeza, L.F., (2010) "State of the art on high-temperature thermal energy storage for power generation. Part 2-Case studies", *Renewable and Sustainable Energy Reviews, 14 (1)*, pp 56-72. Elsevier.

[9] Pawlowski, A., Mendoza, J.L., y Guzmán, J.L., y Berenguel, M., y Acién, F.G., y Dormido, S., (2015) "Selective pH and dissolved oxygen control strategy for a raceway reactor within an event-based approach", *Control Engineering Practice, 44*, pp 209-218.

[10] Soh, L. K., y Tsatsoulis, C. (1999). "Texture analysis of SAR sea ice imagery using gray level co-occurrence matrices", *Geoscience and Remote Sensing, IEEE Transactions on, 37(2)* , pp 780-795.

Cuadro 6: Fórmula de características texturales.

Característica	**Fórmula**
Energía	$f_1 = \sum_i \sum_j p(i,j)^2$
Contraste	$f_2 = \sum_{i=1}^{N_g-1} n^2 \left\{ \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \right\}_{\lvert i-j \rvert = n}$
Correlación	$f_3 = \frac{\sum_i \sum_j (ij) p(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y}$
Suma de cuadrados: Varianza	$f_4 = \sum_i \sum_j (i-\mu)^2 \, p(i,j)$
Homogeneidad	$f_5 = \sum_i \sum_j \frac{1}{1+(i-j)^2} p(i,j)$
Media de la suma	$f_6 = \sum_{i=2}^{2N_g} i \cdot p_{x+y}(i)$
Varianza de la suma	$f_7 = \sum_{i=2}^{2N_g} (i - f_8)^2 \, p_{x+y}(i)$
Entropía de la suma	$f_8 = -\sum_{i=2}^{2N_g} p_{x+y}(i) \log\left(p_{x+y}(i)\right)$
Entropía	$f_9 = -\sum_i \sum_j p(i,j) \log\left(p(i,j)\right)$
Varianza de la diferencia	$f_{10} = variance\left(p_{x-y}\right)$
Entropía de la diferencia	$f_{11} = -\sum_{i=0}^{N_g-1} p_{x-y}(i) \log\left(p_{x-y}(i)\right)$
Medidas informativas de correlación (1)	$f_{12} = \frac{HXY - HXY1}{max\{HX, HY\}}$ donde: $HXY = -\sum_i \sum_j p(i,j) \log\left(p(i,j)\right)$ $HXY1 = -\sum_i \sum_j p(i,j) \log\left(p_x(i) \, p_y(j)\right)$ $HXY2 = -\sum_i \sum_j p_x(i) \, p_y(j) \log\left(p(i,j)\right)$
Medidas informativas de correlación (2)	$f_{13} = \left[1 - e^{-2 \cdot (HXY2 - HXY)}\right]^{1/2}$
Coeficiente de máxima correlación	f_{14} =[Segundo mayor autovalor de Q]$^{1/2}$ donde: $Q(i,j) = \sum_k \frac{p(i,k) \cdot p(j,k)}{p_x(i) \cdot p_y(j)}$
Autocorrelación	$f_{15} = \sum_i \sum_j (ij) p(i,j)$
Disimilaridad	$f_{16} = \sum_i \sum_j \lvert i-j \rvert \cdot p(i,j)$
Tono de grupo (Cluster shadow)	$f_{17} = \sum_i \sum_j (i + j - \mu_x - \mu_y)^3 \, p(i,j)$
Relevancia de grupo (Cluster Prominence)	$f_{18} = \sum_i \sum_j (i + j - \mu_x - \mu_y)^4 \, p(i,j)$
Probabilidad máxima	$f_{19} = \sum_{i,j} MAX\left[p(i,j)\right]$
Diferencia inversa	$f_{20} = \sum_{i,j=1}^{G} \frac{C_{ij}}{1+\lvert i-j \rvert}$
Diferencia inversa normalizada	$f_{21} = \sum_{i,j=1}^{G} \frac{C_{ij}}{1+\lvert i-j \rvert^2 / G^2}$
Momento de la diferencia inversa normalizada	$f_{22} = \sum_{i,j=1}^{G} \frac{C_{ij}}{1+(i-j)^2 / G^2}$

Anexo 5. Código de las funciones propias en MATLAB

TFM_Analisis_Completo

```
clear all;

%% OPCIONES de ejecución
%
PROCED = 1;      % Selecciona el procedimiento de extracción de carac-
terísticas
                 % 1 = Cromático (RGB, HSV, YIQ, Lab, R-B, R/B, R-
B/R+B, Chroma)
                 % 2 = Textural - GLCM, Haralick (22 parámetros: entro-
pía, ...)
                 % 3 = Law's
                 % 4 = Gabor (NO OPERATIVO)
                 % 6 = Clasificador K-means

SELECTOR = 4;    % Selecciona las imágenes que se emplearán
                 % 1 = Originales
                 % 2 = Imágenes fraccionadas
                 % 3 = Imágenes redimensionadas
                 % 4 = Muestra de imágenes (para análisis cromático)
                 % 5 = Mapas de Laws
                 % 6 = Filtro de contraste
                 % 7 = Filtro de Gabor
                 % 8 = Tensores

NORMALIZ = 1;    % Elige el tipo de normalización para los datos croma-
ticos
                 % 1 = Normalización lineal (Cn = C / 255)
                 % 2 = Normalización estadística (Cn = (C - media)/Va-
rianza)

CLASIFIC = 1;    % Elige el tipo de clasificador
                 % 1 = Mínima distancia
                 % 2 = K-means
                 % 3 = Redes neuronales

GENERARim = 0;   % Si se seleccionan imágenes fraccionadas o redimen-
sionadas
                 % las imágenes podrían exister o desear crearse nuevas

VISUAL_1D = 0;   % Gráficos de histograma individuales
VISUAL_2D = 0;   % Gráficos por parejas de parámetros
VISUAL_3D = 0;   % Gráficos por tríos de parámetros

CLASIF_1D = 1;
CLASIF_2D = 1;
CLASIF_3D = 0;
CLASIF_4D = 0;
CLASIF_xD = [ CLASIF_1D, CLASIF_2D, CLASIF_3D, CLASIF_4D ];
```

```
TFM_Config;

%% Extracción de características cromáticas o texturales
%
[ MC, nroMuestras, MuestrasAcum ] = TFM_CalculoCaract( PROCED, ...
    SELECTOR, NroClases, Inic_foto, Fin, nroFotos, FraccionaEN, Redi-
mensionaA, ANG);

%% Normalización
%
[ MCn, MAXe ] = TFM_Normaliz( MC, PROCED, NORMALIZ, [] );

%% Visualizaciones 1D, 2D y 3D
%
TFM_representacion;

%% Reducción del número de muestras para el procedimiento Cromático
[MCn1, MuestrasAcum1] = TFM_Minorac( PROCED, MCn, MuestrasAcum, mino-
rac );

%% Clasificador
[ MinE ] = TFM_Clasifica( PROCED, SELECTOR, NORMALIZ, CLASIFIC,...
    CLASIF_xD, MCn1, MuestrasAcum1 );
```

TFM_Config

```
%% TFM_Config
%
% Script que define la configuración asociada a los parámetros de se-
lección
% básicos del scrip 'TFM_Analisis_Completo'.

disp('Comenzando Configuración')

%% Ubicación de carpetas
carpeta_nubes = fullfile('Edu','TFM','Imagenes - Nubes');
carpeta_figuras = fullfile('Edu','TFM','Imagenes - Figuras');

NroClases = 5;
ANG = 1;
minorac = 1;

%% Declaraciones conforme al tipo de imágenes

% Base de datos SWIMCAT
switch SELECTOR
    case 4
        A_inic='A_'; A_nroFotos = 100;  % SKY
        B_inic='B_'; B_nroFotos = 80;   % PATTERN
        C_inic='C_'; C_nroFotos = 185;  % DENSE-DARK
        D_inic='D_'; D_nroFotos = 115;  % DENSE-LIGHT
        E_inic='E_'; E_nroFotos = 80;   % VEIL
    case 6
        A_inic='A_'; A_nroFotos = 224;  % SKY
```

```
        B_inic='B_'; B_nroFotos = 89;    % PATTERN
        C_inic='C_'; C_nroFotos = 251;   % DENSE-DARK
        D_inic='D_'; D_nroFotos = 135;   % DENSE-LIGHT
        E_inic='E_'; E_nroFotos = 85;    % VEIL
    case 7
        A_inic='Gabor39_A_selec ('; A_nroFotos = 100;  % SKY
        B_inic='Gabor39_B_selec ('; B_nroFotos = 80;   % PATTERN
        C_inic='Gabor39_C_selec ('; C_nroFotos = 185;  % DENSE-DARK
        D_inic='Gabor39_D_selec ('; D_nroFotos = 115;  % DENSE-LIGHT
        E_inic='Gabor39_E_selec ('; E_nroFotos = 80;   % VEIL
    case 8
        A_inic='test_structureA_'; A_nroFotos = 62;  % SKY
        B_inic='test_structureB_'; B_nroFotos = 62;   % PATTERN
        C_inic='test_structureC_'; C_nroFotos = 62;  % DENSE-DARK
        D_inic='test_structureD_'; D_nroFotos = 62;  % DENSE-LIGHT
        E_inic='test_structureE_'; E_nroFotos = 62;   % VEIL
    otherwise
        A_inic='A_selec ('; A_nroFotos = 100;  % SKY
        B_inic='B_selec ('; B_nroFotos = 80;   % PATTERN
        C_inic='C_selec ('; C_nroFotos = 185;  % DENSE-DARK
        D_inic='D_selec ('; D_nroFotos = 115;  % DENSE-LIGHT
        E_inic='E_selec ('; E_nroFotos = 80;   % VEIL
end

% Para el análisis cromático se usan siempre las muestras de imágenes
if PROCED == 1
    SELECTOR = 4;
    warning(['Las imágenes seleccionadas no están optimizadas para un
',...
        'análisis cromático. Se tomarán las muestras preparadas.'])
    minorac = 0.05;
end

switch SELECTOR
    case 1
        carpeta_nubes = fullfile(carpeta_nubes, 'IMAG_SWIMCAT');
        Fin = ').png';
        FraccionaEN = 0; RedimensionaA = 0;
        Seleccion = 'Originales';
    case 2
        FraccionaEN = 9;     % Número de partes en que se dividirán las
imágenes
        carpeta_nubes = fullfile(carpeta_nubes,
['SplitS',num2str(FraccionaEN)]);
        A_inic='A_selec (0'; B_inic='B_selec (0'; C_inic='C_selec (0';
        D_inic='D_selec (0';E_inic='E_selec (0';
        Fin = ').png';
        RedimensionaA = 1;
        Seleccion = ['Fraccion',num2str(FraccionaEN)];
    case 3
        RedimensionaA = 0.5;     % Fracción de redimensionado de las
imágenes
        carpeta_nubes = fullfile(carpeta_nubes, ['Fracti-
onS',num2str(RedimensionaA)]);
%         Fin = ').png';
        Fin = [num2str(RedimensionaA),').png'];
        FraccionaEN = 0;
        Seleccion = ['Redimens',num2str(RedimensionaA)];
    case 4
        carpeta_nubes = fullfile(carpeta_nubes, 'Im_Segment');
```

```
        A_nroFotos = 120; C_nroFotos = 120; D_nroFotos = 120;
        Fin = '_segimg.png';
        FraccionaEN = 0; RedimensionaA = 0;
        Seleccion = 'Muestras';
    case 5
        carpeta_nubes = fullfile(carpeta_nubes, 'Laws');
        Fin = ').png';
        FraccionaEN = 0; RedimensionaA = 0;
        Seleccion = 'Originales';
    case 6
        carpeta_nubes = fullfile(carpeta_nubes, 'FiltroContraste');
        Fin = '_FiltroContrasteimg.png';
        FraccionaEN = 0; RedimensionaA = 0;
        Seleccion = 'Filtradas';
    case 7
        carpeta_nubes = fullfile(carpeta_nubes, 'Gabor');
        Fin = ').png';
        FraccionaEN = 0; RedimensionaA = 0;
        Seleccion = 'Gabor';
    case 8
        carpeta_nubes = fullfile(carpeta_nubes, 'Tensores');
        Fin = 'img.png';
        FraccionaEN = 0; RedimensionaA = 0;
        Seleccion = 'Tensores';
end

% A_nroFotos=3; B_nroFotos=3; C_nroFotos=3; D_nroFotos=3; E_nroFo-
tos=3;
Inic_foto = {A_inic, B_inic, C_inic, D_inic, E_inic};
nroFotos = [ A_nroFotos, B_nroFotos, C_nroFotos, D_nroFotos, E_nroFo-
tos ];

%% Declaraciones conforme al tipo de procedimiento

switch PROCED
    case 1
        NroParam = 16;  % Número de parámetros cromáticos
        Proced = 'Cromatico';
    case 2
        NroParam = 22;  % Número de parámetros texturales por Haralick
        ANG = 4;        % Número de ángulos de parámetros GLCM
        Proced = 'Haralick';
    case 3
        NroParam = 9;  % Número de parámetros de Law's
        Proced = 'Laws';
end

if GENERARim == 1
    switch SELECTOR
        case 2
            for i=1:5
                TFM_Fracciona(Inic_foto{i}, Fin, nroFotos(i),...
                    FraccionaEN, carpeta_nubes);
            end
        case 3
            for i=1:5
                TFM_Redim(Inic_foto{i}, Fin, nroFotos(i),...
                    RedimensionaA, carpeta_nubes);
            end
```

```
    end
end

%% Características cromáticas
switch PROCED
    case 1
        Params ={'Plano R',...       %1
            'Plano G',...            %2
            'Plano B',...            %3
            'Plano H',...            %4
            'Plano S',...            %5
            'Plano V',...            %6
            'Plano Y',...            %7
            'Plano I',...            %8
            'Plano Q',...            %9
            'Plano L',...            %10
            'Plano a*',...           %11
            'Plano b*', ...          %12
            'Plano R/B',...          %13
            'Plano R-B',...          %14
            'Plano (R-B)/(R+B)',... %15
            'Chroma'};               %16

        i=0;
        TitleParam = {};
        for j1 = 1:16
            for j2 = j1+1:16
                for j3 = j2+1:16
                    i = i+1;
                    TitleParam{i} = ['Distancia ',Params{j1},' + ',...
                        Params{j2}, ' + ', Params{j3}];
                end
            end
        end
        clear Params
%% Características de Haralick
    case 2

        TitleParam ={'Contraste',...     %1
            'Correlación MATLAB',...     %2
            'Energía',...                %3
            'Homogeneidad MATLAB',...    %4
            'Autocorrelación',...        %5
            'Correlación [1,2]',...      %6
            'Cluster Prominence',...     %7
            'Cluster Shade',...          %8
            'Disimilaridad',...          %9
            'Entropía',...               %10
            'Homogeneidad',...                              %11
            'Probabilidad Máxima', ...                      %12
            'Suma de cuadrados: Varianza',...               %13
            'Media de la suma',...                          %14
            'Varianza de la suma',....                      %15
            'Entropía de la suma',...                       %16
            'Varianza de la diferencia',...                 %17
            'Entropía de la diferencia',...                 %18
            'Information measure of correlation1',...       %19
            'Information measure of correlation2',...       %20
            'Inverse difference normalized (INN)',...       %21
            'Inverse difference moment normalized'};        %22
```

```
%% Características de Law's
    case 3

        TitleParam ={'L5E5/E5L5',...     %1
            'L5R5/R5L5',...              %2
            'E5S5/S5E5',...              %3
            'S5S5',...                   %4
            'R5R5',...                   %5
            'L5S5/S5L5',...              %6
            'E5E5',...                   %7
            'E5R5/R5E5',...              %8
            'S5R5/R5S5'};                %9
end

disp('Fin de Configuración')
```

TFM_Fracciona(·)

```
function [ im_nroN ] = TFM_Fracciona( im_inic, im_fin, im_nro, Spli-
tInto, folder0 )
%TFM_Fracciona   Función que fracciona las imágenes en subimágenes

% Inicializaciones previas
S0 = floor(sqrt(SplitInto));
folder = folder0;
% Crea la carpeta
if (exist(folder,'dir') == 0)
   mkdir (folder);
end
% Genera las imágenes fraccionadas
for i=1:im_nro
    % Nombre de la imagen original
    im_i=[im_inic,num2str(i),im_fin];
    % Carga la imagen
    f1=imread(im_i);
    [a,b,~]=size(f1);
    for j=1:S0
        for k = 1:S0
            % Extrae cada subimagen
            f3=imcrop(f1, [round((j-1)*a/S0), round((k-1)*b/S0), a/S0,
b/S0]);
            file1 =
[im_inic,num2str(0),num2str(i),num2str(j),num2str(k),im_fin];
            file2 = fullfile(folder,file1);
            imwrite(f3,file2);
        end
    end
end
% Número de subimágenes totales
im_nroN = im_nro * SplitInto;

end
```

TFM_Fracciona(·)

```
function [ im_nroN ] = TFM_Redim( im_inic, im_fin, im_nro, fracc, fol-
der0 )
%TFM_Redim   Función que redimensiona y salva las imágenes

% Crea la carpeta para almacenar las imágenes redimensionadas
if (exist(folder0,'dir') == 0)
   mkdir (folder0);
end

for i=1:im_nro
    % Nombre de la imagen
    im_i=[im_inic,num2str(i),im_fin];
    % Carga la imagen y la redimensiona
    f1=imread(im_i);
    f2=imresize(f1,fracc);
    % Define el nombre de la nueva imagen y la salva
    file1 = [im_inic,num2str(i),num2str(fracc),im_fin];
    file2 = fullfile(folder0,file1);
    imwrite(f2,file2);
end
% Número de nuevas imágenes
im_nroN = im_nro;

end
```

TFM_CalculoCaract(·)

```
function [ MC, nroMuestras, MuestrasAcum ] = TFM_CalculoCaract( PRO-
CED, ...
    SELECTOR, NroClases, Inic_foto, Fin, nroFotos, Fracc, Redim, ANG)
%TFM_CalculoCaract   Función para el cálculo de características
%   Esta función extrae las características cromáticas y texturales en
%   función de los parámetros configurados.

MC = []; MuestrasAcum = 0;

% Selección del tipo de características en función del procedimiento
switch PROCED
    % Característcas cromáticas
    case 1
        disp('Procesando características cromáticas')
        % Definiciones previas
        nroMuestras = 400 * nroFotos;
        MuestrasAcum = [0, 400 * nroFotos];
        % Características para cada clase
        for i = 1:NroClases
            MuestrasAcum(i+1) = MuestrasAcum(i) + MuestrasAcum(i+1);
            % TFM_Crom_Features %%%%%%%%%
            aaa = TFM_Crom_Features(Inic_foto{i}, Fin, nroFotos(i));
            MC = [MC; aaa];
            MuestrasAcum(i) = length(aaa);
            clear aaa;
        end
    % Características texturales
    case 2
```

```
        disp('Procesando características texturales')
        % Definiciones previas para imágenes fraccionadas
        if Fracc ~= 0
            MC = zeros(ANG * sum(nroFotos) * Fracc, 22);
            nroMuestras = ANG * nroFotos * Fracc;
            MuestrasAcum = [0, ANG * nroFotos * Fracc ];
        else
            MC = zeros(ANG * sum(nroFotos), 22);
            nroMuestras = ANG * nroFotos;
            MuestrasAcum = [0, ANG * nroFotos ];
        end
        % Características para cada clase
        for i = 1:NroClases
            MuestrasAcum(i+1) = MuestrasAcum(i) + MuestrasAcum(i+1);
            % TFM_GLCM_Coleccion %%%%%%%%%
            MC(MuestrasAcum(i)+1:MuestrasAcum(i+1),:) = ...
                TFM_GLCM_Coleccion( Inic_foto{i}, Fin, nroFotos(i),
...
                Fracc, Redim, ANG, SELECTOR );
        end
    % Características de Laws
    case 3
        disp('Procesando características de Laws')
        % Características para cada clase
        for i = 1:NroClases
            % TFM_Laws_Features %%%%%%%%%
            MC(MuestrasAcum(i)+1:MuestrasAcum(i+1),:) = ...
                TFM_Laws_Features(Inic_foto{i}, Fin, nroFotos(i));
            nroMuestras(i) = length(MC) - MuestrasAcum;
            MuestrasAcum = length(MC);
        end
    % Características de Gabor
    case 4
        disp('Procesando características de Gabor')
        for i = 1:NroClases
            % TFM_Gabor %%%%%%%%%
            MC(MuestrasAcum(i)+1:MuestrasAcum(i+1),:) = ...
                TFM_Gabor_Features(Inic_foto{i}, Fin, nroFotos(i));
            nroMuestras(i) = length(MC) - MuestrasAcum;
            MuestrasAcum = length(MC);
        end
end
% Cálculo del vector de Muestras Acumuladas
MuestrasAcum=zeros(5,1);
MuestrasAcum(1)=nroMuestras(1);
for i=2:5
    MuestrasAcum(i) = MuestrasAcum(i-1) + nroMuestras(i);
end
end
```

TFM_Crom_Features(·)

```
function [ distancia ] = TFM_Crom_Features( im_inic, im_fin, im_nro )
%TFM_RGBAnalysis extrae las principales componentes (PCA) de una ima-
gen.
%   Los espacios de color considerados en este estudio son:
%   * R, G y B
%   * H, S y V
%   * Y, I y Q
```

```
%    * L, a y b
%    * R/B, B-R, (B-R)/(B+R)
%    * Chroma
%

% Inicializaciones previas
VectCaracCrom =[];
index = 1;

%% Análisis para extracción de CARACTERÍSTICA cromática por píxel
for i=1:im_nro
    % Definición del nombre de la imagen a procesar
    if im_nro == 1
        im_i = [ im_inic, im_fin ];
    else
        im_i = [ im_inic, num2str(i), im_fin ];
    end
    disp (['Procesando imagen ', im_i])
    % Carga de la imagen y configuración del tamaño de la imagen a es-
tudiar
    im0=imread(im_i);
    if im_nro == 1
        [n0, m0, ~] = size(im0);
        aux = 1;
        im1=imresize(im0, aux * [n0,m0]);
    else
        im1=imresize(im0,[20,20]);
    end
    % Cálculo de los planos de color
    im1_R = double(im1(:,:,1)); im1_G = double(im1(:,:,2)); im1_B =
double(im1(:,:,3));
    im1_HSV = rgb2hsv(im1);
    im1_H = im1_HSV(:,:,1); im1_S = im1_HSV(:,:,2); im1_V =
im1_HSV(:,:,3);
    im1_YIQ = rgb2ntsc(im1);
    im1_Y = im1_YIQ(:,:,1); im1_I = im1_YIQ(:,:,2); im1_Q =
im1_YIQ(:,:,3);
    im1_Lab = RGB2Lab(im1);
    im1_L = im1_Lab(:,:,1); im1_a = im1_Lab(:,:,2); im1_b =
im1_Lab(:,:,3);
    im1_RdB = double(im1_R)./double(im1_B);
    im1_RmB = abs(im1_B - im1_R); im1_RaB = im1_R + im1_B;
    im1_RcB = double(im1_RmB)./double(im1_RaB);
    im1_Chroma = max(max(im1_R,im1_G),im1_B)-
min(min(im1_R,im1_G),im1_B);
    % Normalización de los datos a la escala [0,255]
    [n1,m1]=size(im1_R);
    Im1_C1 = reshape(im1_R,n1*m1,1);
    Im1_C2 = reshape(im1_G,n1*m1,1);
    Im1_C3 = reshape(im1_B,n1*m1,1);
    Im1_C4 = reshape(im1_H,n1*m1,1).*255;
    Im1_C5 = reshape(im1_S,n1*m1,1).*255;
    Im1_C6 = reshape(im1_V,n1*m1,1).*255;
    Im1_C7 = reshape(im1_Y,n1*m1,1).*255;
    Im1_C8 = reshape(im1_I,n1*m1,1)+127;
    Im1_C9 = reshape(im1_Q,n1*m1,1).*255;
    Im1_C10 = reshape(im1_L,n1*m1,1);
    Im1_C11 = reshape(im1_a,n1*m1,1)+127;
    Im1_C12 = reshape(im1_b,n1*m1,1)+127;
    Im1_C13 = reshape(im1_RdB,n1*m1,1).*255;
```

```
    Im1_C14 = reshape(im1_RmB,n1*m1,1);
    Im1_C15 = reshape(im1_RcB,n1*m1,1).*255;
    Im1_C16 = reshape(im1_Chroma,n1*m1,1);
    % Generación del vector de características cromáticas
    Carac_Crom =
[Im1_C1,Im1_C2,Im1_C3,Im1_C4,Im1_C5,Im1_C6,Im1_C7,Im1_C8,Im1_C9,...
        Im1_C10,Im1_C11,Im1_C12,Im1_C13,Im1_C14,Im1_C15,Im1_C16];
    VectCaracCrom = [VectCaracCrom; Carac_Crom];

    index = index + 1;
    clear im_i
end

%% Análisis para extracción de DISTANCIA cromática por píxel
% Inicializaciones previas
k0 = 0;
VectCaracCrom2  = double(VectCaracCrom);
% Cálculo de las distancias cromáticas a la diagonal del cubo tricolor
for j1 = 1:16
    for j2 = j1+1:16
        for j3 = j2+1:16
            k0 = k0+1;
            modD2 = sqrt( VectCaracCrom2(:,j1).^2 + ...
                VectCaracCrom2(:,j2).^2 + VectCaracCrom2(:,j3).^2 );
            modX2 = sqrt((255-VectCaracCrom2(:,j1)).^2 + ...
                (255-VectCaracCrom2(:,j2)).^2 + ...
                (255-VectCaracCrom2(:,j3)).^2);
            alpha2 = acos( ( modX2 - modD2 - sqrt(3*255^2) ) ./ ...
                (-2*sqrt( 3*255^2 ) * modD2 ) );
            dist02(:,1) = modD2 .* sin(alpha2);
            % Eliminación de los valores alejados de los valores me-
dios
            if im_nro == 1
                distancia(:,k0) = dist02(:,1);
            else
                distancia(:,k0) = TFM_quantile( dist02(:,1) );
            end
            clear distA0
        end
    end
end
end
```

TFM_GLCM_Coleccion(·)

```
function StatsN =...
    TFM_GLCM_Coleccion( im_inic, im_fin, im_nro, Fracc, Redim, nroAng,
Sel)
%TFM_GLCM_Coleccion Función que extrae las características de Haralick
%
% Se especifican el nombre de las imágenes y si están fraccionadas o
% redimensionalizadas, así como el número de ángulos.

%% Inicializaciones
% Comprobación de validez de nombre de fotos
if and(Fracc~=0, Redim~=1)
    StatsN = NaN;
```

```
    warning('No se puede redimensionar y dividir en partes en el mismo
proceso');
    return;
end

% Inicializaciones
if nroAng == 4
    G1 = zeros(256,256,4);
else if nroAng == 1
        G1 = zeros(256,256,2);
    end
end

if Fracc~=0
    j0 = floor(sqrt(Fracc));
elseif Redim == 1
    j0 = 3;
else
    j0 = 1;
end

statsIm2(im_nro*j0*j0,1) = struct;
statsIm2(1,1).a=0;

%% Extracción de las características de Haralick
index = 1;
for i=1:im_nro
    for j=1:j0
        for k=1:j0
            % Definición del nombre de la imagen a procesar
            if Fracc~=0

im_i=[im_inic,num2str(i),num2str(j),num2str(k),im_fin];
            elseif Sel == 5
                im_i=[im_inic,num2str(i),'_',num2str((j-
1)*3+k),'_',im_fin];
            elseif im_nro == 1
                im_i=[im_inic,im_fin];
            else
                im_i=[im_inic,num2str(i),im_fin];
                index = i;
            end
            disp (['Procesando imagen ', im_i])
            % Carga de la imagen y conversión a escala de grises
            f1=imread(im_i);
            c0 = size(f1,3);
            if c0 == 1
                f2 = f1;
            else
                f2 = rgb2gray(f1);
            end
            % Definición de la distancia de píxeles
            D=1;
            % Cálculo de las matrices GLCM para cada ángulo
            if nroAng == 4
                G11 = graycomatrix(f2,'NumLevels',256,'offset',[0,D]);
                G12 = graycomatrix(f2,'NumLevels',256,'offset',[-
D,D]);
```

```
                G13 = graycomatrix(f2,'NumLevels',256,'offset',[-
D,0]);
                G14 = graycomatrix(f2,'NumLevels',256,'offset',[-D,-
D]);
                G1(:,:,1) = G11;
                G1(:,:,2) = G12;
                G1(:,:,3) = G13;
                G1(:,:,4) = G14;
            else if nroAng == 1
                    G11 = graycomatrix(f2,'NumLevels',256);
                    G1(:,:,1) = G11;
                    G1(:,:,2) = G11;
                end
            end
            % Extracción de las características texturales
            out = TFM_GLCM_Features(G1);
            statsIm2(index,:).a = out;
            index = index + 1;
        end
    end
end

%% Estructuración en una sola matriz
StatsN = zeros(length(statsIm2)*nroAng,22);
for j=1:length(statsIm2)
    for i=1:nroAng
        StatsN((i-1)*length(statsIm2)+j,1) = statsIm2(j).a.contr(i);
        StatsN((i-1)*length(statsIm2)+j,2) = statsIm2(j).a.corrm(i);
        StatsN((i-1)*length(statsIm2)+j,3) = statsIm2(j).a.energ(i);
        StatsN((i-1)*length(statsIm2)+j,4) = statsIm2(j).a.homom(i);

        StatsN((i-1)*length(statsIm2)+j,5) = statsIm2(j).a.autoc(i);
        StatsN((i-1)*length(statsIm2)+j,6) = statsIm2(j).a.corrp(i);
        StatsN((i-1)*length(statsIm2)+j,7) = statsIm2(j).a.cprom(i);
        StatsN((i-1)*length(statsIm2)+j,8) = statsIm2(j).a.cshad(i);
        StatsN((i-1)*length(statsIm2)+j,9) = statsIm2(j).a.dissi(i);
        StatsN((i-1)*length(statsIm2)+j,10) = statsIm2(j).a.entro(i);
        StatsN((i-1)*length(statsIm2)+j,11) = statsIm2(j).a.homop(i);
        StatsN((i-1)*length(statsIm2)+j,12) = statsIm2(j).a.maxpr(i);
        StatsN((i-1)*length(statsIm2)+j,13) = statsIm2(j).a.sosvh(i);
        StatsN((i-1)*length(statsIm2)+j,14) = statsIm2(j).a.savgh(i);
        StatsN((i-1)*length(statsIm2)+j,15) = statsIm2(j).a.svarh(i);
        StatsN((i-1)*length(statsIm2)+j,16) = statsIm2(j).a.senth(i);
        StatsN((i-1)*length(statsIm2)+j,17) = statsIm2(j).a.dvarh(i);
        StatsN((i-1)*length(statsIm2)+j,18) = statsIm2(j).a.denth(i);
        StatsN((i-1)*length(statsIm2)+j,19) = statsIm2(j).a.inf1h(i);
        StatsN((i-1)*length(statsIm2)+j,20) = statsIm2(j).a.inf2h(i);
        StatsN((i-1)*length(statsIm2)+j,21) = statsIm2(j).a.indnc(i);
        StatsN((i-1)*length(statsIm2)+j,22) = statsIm2(j).a.idmnc(i);

    end
end
end
```

TFM_GLCM_Indiv(·)

```
function StatsN = TFM_GLCM_Indiv( imagen )
%TFM_GLCM_Indiv Función que extrae las características de Haralick
```

```
%
% Cálculo de las 22 características de Haralick a partir de una ima-
gen,
% considerando un solo ángulo y una distancia de 1 píxel.

% Inicializaciones previas
f1 = imagen;
c0 = size(f1,3);
% Conversión de la imagen a escala de grises
if c0 == 1
    f2 = f1;
else
    f2 = rgb2gray(f1);
end

% Cálculo de la matriz GLCM y extracción de las características textu-
rales
G11 = graycomatrix(f2,'NumLevels',256);
out = TFM_GLCM_Features(G11);

% Integración de las características en un vector.
StatsN = zeros(1,22);

StatsN(1)  = out.contr; % Contraste
StatsN(2)  = out.corrm; % Correlación
StatsN(3)  = out.energ; % Energía
StatsN(4)  = out.homom; % Homogeneidad
StatsN(5)  = out.autoc; % Autocorrelación
StatsN(6)  = out.corrp; % Coeficiente de máxima correlación
StatsN(7)  = out.cprom; % Relevancia de grupo(Cluster Prominence)
StatsN(8)  = out.cshad; % Tono de grupo (Cluster Shade)
StatsN(9)  = out.dissi; % Disimilaridad
StatsN(10) = out.entro; % Entropía
StatsN(11) = out.homop; % Diferencia inversa
StatsN(12) = out.maxpr; % Probabilidad máxima
StatsN(13) = out.sosvh; % Suma de cuadrados: varianza
StatsN(14) = out.savgh; % Media    de la suma
StatsN(15) = out.svarh; % Varianza de la suma
StatsN(16) = out.senth; % Entropía de la suma
StatsN(17) = out.dvarh; % Varianza de la diferencia
StatsN(18) = out.denth; % Entropía de la diferencia
StatsN(19) = out.inf1h; % Medidas informativas de correlación (1)
StatsN(20) = out.inf2h; % Medidas informativas de correlación (2)
StatsN(21) = out.indnc; % Diferencia inversa normalizada (INN)
StatsN(22) = out.idmnc; % Momento de la diferencia inv. normal.

end
```

TFM_GLCM_Features(·)

```
function [out] = TFM_GLCM_Features(glcmin)
% TFM_GLCM_Features Función que extrae las características de Haralick
%
% Cálculo de las 22 características texturales de Haralick. Toma como
% variable de entrada la matriz GLCM calculada previamente.

% Definiciones previas
glcm = glcmin;
```

```
[size_glcm_1, size_glcm_2, size_glcm_3] = size(glcm);

% Inicialización de matrices
out.contr = zeros(1,size_glcm_3); % Contraste
out.corrm = zeros(1,size_glcm_3); % Correlación
out.energ = zeros(1,size_glcm_3); % Energía
out.homom = zeros(1,size_glcm_3); % Homogeneidad
out.autoc = zeros(1,size_glcm_3); % Autocorrelación
out.corrp = zeros(1,size_glcm_3); % Coeficiente de máxima correlación
out.cprom = zeros(1,size_glcm_3); % Relevancia de grupo(Cluster Promi-
nence)
out.cshad = zeros(1,size_glcm_3); % Tono de grupo (Cluster Shade)
out.dissi = zeros(1,size_glcm_3); % Disimilaridad
out.entro = zeros(1,size_glcm_3); % Entropía
out.homop = zeros(1,size_glcm_3); % Diferencia inversa
out.maxpr = zeros(1,size_glcm_3); % Probabilidad máxima
out.sosvh = zeros(1,size_glcm_3); % Suma de cuadrados: varianza
out.savgh = zeros(1,size_glcm_3); % Media    de la suma
out.svarh = zeros(1,size_glcm_3); % Varianza de la suma
out.senth = zeros(1,size_glcm_3); % Entropía de la suma
out.dvarh = zeros(1,size_glcm_3); % Varianza de la diferencia
out.denth = zeros(1,size_glcm_3); % Entropía de la diferencia
out.inf1h = zeros(1,size_glcm_3); % Medidas informativas de correla-
ción (1)
out.inf2h = zeros(1,size_glcm_3); % Medidas informativas de correla-
ción (2)
out.indnc = zeros(1,size_glcm_3); % Diferencia inversa normalizada
(INN)
out.idmnc = zeros(1,size_glcm_3); % Momento de la diferencia inv. nor-
mal.
glcm_sum  = zeros(size_glcm_3,1);
glcm_mean = zeros(size_glcm_3,1);
u_x = zeros(size_glcm_3,1);
u_y = zeros(size_glcm_3,1);
s_x = zeros(size_glcm_3,1);
s_y = zeros(size_glcm_3,1);
p_x = zeros(size_glcm_1,size_glcm_3);
p_y = zeros(size_glcm_2,size_glcm_3);
p_xplusy = zeros((size_glcm_1*2 - 1),size_glcm_3);
p_xminusy = zeros((size_glcm_1),size_glcm_3);
corm = zeros(size_glcm_3,1);
corp = zeros(size_glcm_3,1);
hx   = zeros(size_glcm_3,1);
hy   = zeros(size_glcm_3,1);
hxy  = zeros(size_glcm_3,1);
hxy1 = zeros(size_glcm_3,1);
hxy2 = zeros(size_glcm_3,1);

for k = 1:size_glcm_3 % number glcms
    % Normalización de la matriz GLCM
    glcm_sum(k) = sum(sum(glcm(:,:,k)));     % Calcula la suma de GLCM
    glcm(:,:,k) = glcm(:,:,k)./glcm_sum(k); % Normaliza cada GLCM
    % Cálculos auxiliares
    glcm_mean(k) = mean2(glcm(:,:,k));       % Computa la media de GLCM
    A=log(glcm(:,:,k) + eps*ones(size_glcm_1,size_glcm_2));
    % Elementos matriciales auxiliares
    colT = (1:1:size_glcm_1)' * ones(1, size_glcm_2);
    filT = colT';
    filP = 1+(1:1:(2*(size_glcm_1)-1));
    % Cálculos intermedios
```

```
    u_x(:,k) = sum(sum(colT.*glcm(:,:,k)));
    u_y(:,k) = sum(sum(filT.*glcm(:,:,k)));
    p_x(:,k) = sum(glcm(:,:,k)')';
    p_y(:,k) = sum(glcm(:,:,k) )';
    s_x(k) = (sum(sum(((colT - u_x(k)).^2 ).*glcm(:,:,k)))) .^0.5;
    s_y(k) = (sum(sum(((filT - u_y(k)).^2 ).*glcm(:,:,k)))) .^0.5;
    corp(k) = sum(sum(colT.*filT.*glcm(:,:,k)));
    corm(k) = sum(sum( (colT - u_x(k)) .* ( filT - u_y(k)) .*
glcm(:,:,k) ));
    hx(k) = -sum(p_x(:,k)'*log(p_x(:,k)+eps));
    hy(k) = -sum(p_y(:,k)'*log(p_y(:,k)+eps));
    hxy1(k) = -sum(sum(glcm(:,:,k) .* log(p_x(:,k) * p_y(:,k)' +
eps)));
    hxy2(k) = -sum(sum(p_x(:,k) * p_y(:,k)'  .* log(p_x(:,k) *
p_y(:,k)' + eps)));
    % Único cálculo no basado en matrices
    for k0 = 1:256
        p_xplusy(k0:256+k0-1,k) = p_xplusy(k0:256+k0-1,k)' +
glcm(k0,:,k);
        p_xminusy (k0,k) = sum(diag(glcm(:,:,k),-k0+1)) +
sum(diag(glcm(:,:,k),k0-1));
    end
    p_xminusy(1) = p_xminusy(1)/2;
    Am=log(p_xminusy(:,k)+eps*ones(size_glcm_1,1));
    Ap=log(p_xplusy(:,k) + eps*ones(2*(size_glcm_1)-1,1));

    % Cálculo de características texturales
    out.entro(k) = - sum(sum(glcm(:,:,k).*A));      hxy(k) = out.en-
tro(k);
    out.denth(k) = - sum(p_xminusy(:,k) .* Am);
    out.cprom(k) = sum(sum(((colT + filT - u_x(k) -
u_y(k)).^4).*glcm(:,:,k)));
    out.savgh(k) = filP * p_xplusy(:,k);
    out.contr(k) =   sum(sum((abs(colT-filT).^2).* glcm(:,:,k)));
    out.dissi(k) =   sum(sum((abs(colT-filT)   ).* glcm(:,:,k)));
    out.energ(k) =   sum(sum( glcm(:,:,k).^2) );
    out.entro(k) = - sum(sum( glcm(:,:,k).*A) );
    out.homom(k) =   sum(sum( glcm(:,:,k) ./ ( 1 + abs(colT-filT) )));
    out.homop(k) =   sum(sum( glcm(:,:,k) ./ ( 1 + (filT-colT).^2 )));
    out.sosvh(k) =   sum(sum( glcm(:,:,k) .* ( (colT -
glcm_mean(k)).^2 ) ));
    out.indnc(k) =   sum(sum( glcm(:,:,k) ./ ( 1 + abs(filT-
colT)/size_glcm_1 ) ));
    out.idmnc(k) =   sum(sum( glcm(:,:,k) ./ ( 1 + ((colT-
filT)/size_glcm_1 ).^2)));
    out.maxpr(k) =   max(max(glcm(:,:,k)));
    out.senth(k) = - sum(p_xplusy(:,k) .* Ap);
    out.svarh(k) = (((filP) - out.senth(k)).^2) * p_xplusy(:,k);
    out.dvarh(k) = sum(((0:(size_glcm_1-1)).^2)'.*p_xminusy(:,k));
    out.inf1h(k) = (out.entro(k) - hxy1(k)) / ( max([hx(k),hy(k)]) );
    out.inf2h(k) = ( 1 - exp( -2*( hxy2(k) - out.entro(k) ) ) )^0.5;
    out.cprom(k) = sum(sum(((colT + filT - u_x(k) - u_y(k)).^4).*
glcm(:,:,k)));
    out.cshad(k) = sum(sum(((colT + filT - u_x(k) - u_y(k)).^3).*
glcm(:,:,k)));
    out.autoc(k) = corp(k);
    out.corrp(k) = (corp(k) - u_x(k)*u_y(k))/(s_x(k)*s_y(k));
    out.corrm(k) = corm(k) / (s_x(k)*s_y(k));
end
```

TFM_Laws_Features(·)

```
function StatsN = TFM_Laws_Features( im_inic, im_fin, im_nro )
%TFM_Laws_Features Función que extrae las características de Laws
%
% Cálculo de las características de Laws.

%% Inicializaciones
Tamanyo = 2;
Pixel = 125 - (Tamanyo*2);
statsIm2 = zeros( im_nro, 9 );
out = zeros(1,9);

%% Extracción de las características de Laws
for i=1:im_nro
    im_i=[im_inic,num2str(i),im_fin];

    disp(['Analizando ', im_i])
    f1=imread(im_i);
    % Extracción de los mapas de Laws
    mapz = laws( f1, Tamanyo);
    for kk = 1:9
        out(kk) = sum(sum(mapz{kk}(1+Tamanyo:length(mapz{1})-Ta-
manyo,...
            1+Tamanyo:length(mapz{1})-Tamanyo)))/(Pixel*Pixel*128);
    end

    statsIm2(i,:) = out;
end
StatsN = statsIm2;

end
```

TFM_Gabor_Features(·)

```
function StatsN = TFM_Gabor_Features( im_inic, im_fin, im_nro )
%TFM_Gabor_Features Función que extrae las características de Gabor
%
% Cálculo de las características de Gabor.
%% Inicializaciones
Tamanyo = 2;
Pixel = 125 - (Tamanyo*2);
statsIm2 = zeros( im_nro, 9 );
out = zeros(1,9);

gaborArray = gaborFilterBank(5,8,39,39);
[u,v] = size(gaborArray);

%% Extracción de las características de Gabor
for i=1:im_nro
    im_i=[im_inic,num2str(i),im_fin];

    disp(['Analizando ', im_i])
    f1 = imread(im_i);
    [gaborResult, featureVector] = gaborFeatures(f1,gaborArray,4,4);
```

```
    statsIm2(i,:) = out;
end
StatsN = statsIm2;
end
```

TFM_Normaliz(·)

```
function [ MCn, MAXe ] = TFM_Normaliz( MC, PROCED, NORMALIZ, MAXe_o )
%TFM_Normaliz  Función que normaliza las características de clasifica-
ción

% Inicializaciones previas
[n,NroCarac] = size(MC);
% Definición de matrices
MAXe = zeros(NroCarac,1);
MCn = double(zeros(n,NroCarac));
% Para cada tipo de característica
for i=1:NroCarac
    if n == 1
        MAXe(i) = MAXe_o(i);
    else
        % Selección del parámetro de normalización
        switch PROCED
            % Procedimiento cromático
            case 1
                if NORMALIZ == 1
                    MAXe(i) = sqrt((255^2)*3);
                end
            % Procedimiento textural
            case 2
                MAXe(i) = max(abs(MC(:,i)));
            % Procedimiento basado en Laws
            case 3
                MAXe(i) = 1;
        end
    end
    % Proceso de normalización
    switch PROCED
        % Procedimiento cromático
        case 1
            % Normalización lineal
            if NORMALIZ == 1
                MCn(:,i) = double(MC(:,i))./double(MAXe(i));
            % Normalización estadística
            elseif NORMALIZ == 2
                MCn(:,i) = (double(MC(:,i)) - mean(dou-
ble(MC(:,i))))...
                    ./std(double(MC(:,i)));
            end
        % Procedimiento textural
        case 2
            MCn(:,i) = abs(double(MC(:,i)))./double(MAXe(i));
        % Procedimiento basado en Laws
        case 3
            MCn(:,i) = abs(double(MC(:,i)))./double(MAXe(i));
    end
end
end
```

TFM_Minorac(·)

```
function [ MCn1, MuestrasAcum1 ] = TFM_Minorac( PROCED, ...
    MCn, MuestrasAcum, minorac )
%TFM_Minorac   Función que reduce el número de muestras aleatoriamente

k = 0;
Muestras = [0; MuestrasAcum];
MCn1 = zeros(minorac * MuestrasAcum(length(Mues-
trasAcum)),size(MCn,2));
if PROCED == 1
    for i = 1:length(MuestrasAcum)
        for j = 1:(Muestras(i+1)-Muestras(i))*minorac
            k = k + 1;
            aux = rand();
            MCn1(k,:) = MCn(Muestras(i)+1+round(aux*(Muestras(i+1)-
Muestras(i))),:);
        end
    end
else
    MCn1 = MCn;
end
MuestrasAcum1 = MuestrasAcum * minorac;
end
```

TFM_representacion

```
%%%%% TFM_representacion %%%%%%
%
%% Visualizaciones 1D - Histogramas
close all;
ejes2=linspace(0,1,20);
MuestrasAcum0 = zeros(5,1); MuestrasAcum0(1) = nroMuestras(1);
for i=2:5
    MuestrasAcum0(i) = MuestrasAcum0(i-1) + nroMuestras(i);
end

if VISUAL_1D == 1
    k = 0;
    folder=fullfile(carpeta_figuras,Proced,Seleccion,'1D');
    for i=1:length(TitleParam)
%     for i=1:1
        k = k+1;
        figure(k);
        plot(1*ones(nroMuestras(1),1),MCn(1:MuestrasAcum0(1),i),'rx');
        hold on;
        plot(2*ones(nroMuestras(2),1),MCn(MuestrasAcum0(1)+1:Mues-
trasAcum0(2),i),'gx');
        plot(3*ones(nroMuestras(3),1),MCn(MuestrasAcum0(2)+1:Mues-
trasAcum0(3),i),'bx');
        plot(4*ones(nroMuestras(4),1),MCn(MuestrasAcum0(3)+1:Mues-
trasAcum0(4),i),'yx');
        plot(5*ones(nroMuestras(5),1),MCn(MuestrasAcum0(4)+1:Mues-
trasAcum0(5),i),'mx');
        xlim([0.7,7])
        title(TitleParam{i});xlabel('Tipos de nube')
        legend('Cielo','Mosaico','Gris','Cúmulo','Capa')
        ylabel('Distancia a la diagonal')
```

```
        zlabel('Nro de imágenes')

        file = fullfile(folder,[num2str(k),'.jpg']);
        saveas(k,file)
        close all
    end
end
clear hiMCA biMCA hiMCB biMCB hiMCC biMCC hiMCD biMCD hiMCE biMCE

%% Visualizaciones 2D - Emparejamiento a dos variables
k = 0;
if VISUAL_2D == 1
    folder=fullfile(carpeta_figuras,Proced,Seleccion,'2D')
    for i = 1:length(TitleParam)
        for j = i+1:length(TitleParam)
            k=k+1;
            figure(k);
            h1 = plot(MCn(1:MuestrasAcum0(1),j),...
                MCn(1:MuestrasAcum0(1),i),'rx',...
                MCn(MuestrasAcum0(1)+1:MuestrasAcum0(2),j),...
                MCn(MuestrasAcum0(1)+1:MuestrasAcum0(2),i), 'gx',...
                MCn(MuestrasAcum0(2)+1:MuestrasAcum0(3),j),...
                MCn(MuestrasAcum0(2)+1:MuestrasAcum0(3),i), 'bx',...
                MCn(MuestrasAcum0(3)+1:MuestrasAcum0(4),j),...
                MCn(MuestrasAcum0(3)+1:MuestrasAcum0(4),i),'yx',...
                MCn(MuestrasAcum0(4)+1:MuestrasAcum0(5),j),...
                MCn(MuestrasAcum0(4)+1:MuestrasAcum0(5),i), 'mx');
            h1 = round(h1);
            title('Comparación por parejas')
            xlim([0,1.35])
            legend('Cielo','Mosaico','Gris','Cúmulo','Capa')
            xlabel(TitleParam{i}); ylabel(TitleParam{j});
            file = fullfile(folder,[num2str(k),'.jpg']);
            saveas(k,file)
            close all
        end
    end
end

%% Visualizaciones 3D - Emparejamiento a 3 variables
if VISUAL_3D == 1
    folder=fullfile(carpeta_figuras,Proced,Seleccion,'3D');
    k = 0;
    for i1 = 1:length(TitleParam)
        for i2 = i1+1:length(TitleParam)
            for i3 = i2+1:length(TitleParam)
                k=k+1;
                figure(k);
                scatter3(MCn(1:MuestrasAcum0(1),i1),...
                    MCn(1:MuestrasAcum0(1),i2),...
                    MCn(1:MuestrasAcum0(1),i3),10,'r')
                hold on;
                scatter3(MCn(MuestrasAcum0(1)+1:Mues-
trasAcum0(2),i1),...
                    MCn(MuestrasAcum0(1)+1:MuestrasAcum0(2),i2),...
                    MCn(MuestrasAcum0(1)+1:Mues-
trasAcum0(2),i3),10,'g')
                scatter3(MCn(MuestrasAcum0(2)+1:Mues-
trasAcum0(3),i1),...
                    MCn(MuestrasAcum0(2)+1:MuestrasAcum0(3),i2),...
```

```
                    MCn(MuestrasAcum0(2)+1:Mues-
trasAcum0(3),i3),10,'b')
                scatter3(MCn(MuestrasAcum0(3)+1:Mues-
trasAcum0(4),i1),...
                    MCn(MuestrasAcum0(3)+1:MuestrasAcum0(4),i2),...
                    MCn(MuestrasAcum0(3)+1:Mues-
trasAcum0(4),i3),10,'y')
                scatter3(MCn(MuestrasAcum0(4)+1:Mues-
trasAcum0(5),i1),...
                    MCn(MuestrasAcum0(4)+1:MuestrasAcum0(5),i2),...
                    MCn(MuestrasAcum0(4)+1:Mues-
trasAcum0(5),i3),10,'m')
                title('Comparación a tres');
                xlabel(TitleParam{i1}); ylabel(TitleParam{i2});
                zlabel(TitleParam{i3});
                hold off;

                file = fullfile(folder,[num2str(k),'.jpg']);
                saveas(k,file)
                close all;
            end
        end
    end
end
```

TFM_Clasifica

```
function [ MinE ] = TFM_Clasifica( PROCED, SELECTOR, NORMALIZ, CLASI-
FIC,...
    CLASIF_xD, MCn, MAcum )
%TFM_Clasifica   Función que clasifica un vector de características

% Definicion de matrices
MinE = [];

% Proceso de clasificación
for numparam = 1:4
    if CLASIF_xD(numparam) == 1
        % Clasificación
        [~, Erel] = TFM_clasificaStipo( CLASIFIC, MCn, MAcum, numparam
);
        % Construcción del vector de errores mínimos
        MinE = [ MinE ; min(Erel(1:6,:)') ];
        % Nombre del archivo
        file = ['Clas_','Pr',num2str(PROCED),'Sel',num2str(SELEC-
TOR),...
            'Norm',num2str(NORMALIZ),'Clasif',num2str(CLASIFIC),
'_',...
            num2str(numparam),'D','.xls'];
        % Guardado parcial
        if length(Erel)>65000
            for i = 1:ceil(length(Erel)/65000)
                xlswrite(file,Erel(:, ...
                    int32((i-1)*65000+1) :...
                    int32(min([i*65000,length(Erel)])))',...
                    ['Error',num2str(numparam),'D_',num2str(i)]);
            end
        else
```

```
            xlswrite(file, Erel',['Error',num2str(numparam),'D']);
        end
        clear Erel file
    end
end
end
```

TFM_clasificaStipo

```
function [ ErrorCtd, ErrorRel ] = ...
    TFM_clasificaStipo( CLASIFIC, MCn, MAcum, NroParam )
% TFM_clasificaStipo evalúa todas las combinaciones de caracterísi-
cas.
%
% Esta función evalúa todas las posibles combinaciones de caracterís-
ticas
% de una colección de imágenes, generando los prototipos de cada clase
y
% generando un vector con los errores obtenidos, totales y relativos,
para
% cada clase.

% Número de neuronas del clasificador de neuronas
switch NroParam
    case 1
        nroNeuronas = 10;
    case 2
        nroNeuronas = 15;
    case 3
        nroNeuronas = 20;
    case 4
        nroNeuronas = 25;
end
% Número de emparejamiento de parámetros
TotalParam = size(MCn,2);
% Vector de emparejamientos posibles
VectCar = combnk(1:TotalParam,NroParam);
[ nroCombinaciones, ~ ] = size(VectCar);
% Definición de matrices
errores = zeros(6, 2, nroCombinaciones);
% Clasificación de cada posible combinación de emparejamientos
for i=1:nroCombinaciones
    disp(['CLASE ', num2str(VectCar(i,:))]);
    % Tipo de clasificador escogido
    switch CLASIFIC
        % Clasificador de mínima distancia
        case 1
            % Generación de prototipo
            [ M, Prot ] = TFM_prototipos_MinDist2(MCn, MAcum,Vect-
Car(i,:));
            % Clasificación y tasa de errores
            [~, error1] = TFM_clasif_MinDist( M, Prot, MAcum );
            errores(:,:,i) = error1;
        % Clasificador K-means
        case 2
            % Generación de prototipo
            [ M, Prot ] = TFM_prototipos_Kmeans2( MCn, MAcum,Vect-
Car(i,:));
            % Clasificación y tasa de errores
```

```
            [ ~, error1 ] = TFM_clasif_MinDist( M, Prot, MAcum );
            errores(:,:,i) = error1;
        % Red neuronal
        case 3
            % Generación de la red neuronal
            [ Y, T, ~, ~ ] = TFM_protot_Neural( MCn, MAcum,...
                VectCar(i,:), nroNeuronas );
            % Clasificación y tasa de errores
            errores(:,:,i) = TFM_clasif_Neural( Y, T, MAcum );
            % Debido a la lentitud, se salva cada 500 muestras
            if and(rem(i,500) == 0, NroParam == 2)
                file = ['Clas_','Pr',num2str(1),'Sel',num2str(4),...
                    'Norm',num2str(1),'Clasif',num2str(3), '_', ...
                    num2str(2),'D_',num2str(i/500),'.xls'];
                 ErrorRel1  = zeros(6+NroParam, nroCombinaciones);
                 ErrorRel1(1:6,:) = errores(:,2,:);
                 ErrorRel1(7:7+NroParam-1,:) = VectCar';
                 xlswrite(file,ErrorRel1(:,1:65000)',...
                    ['Error',num2str(NroParam),'D']);
            end
    end
end
% Vector de errores en términos de casos fallados
ErrorCtd  = zeros(6+NroParam, nroCombinaciones);
ErrorCtd(1:6,:) = errores(:,1,:);
ErrorCtd(7:7+NroParam-1,:) = VectCar';
% Vector de errores en términos relativos
ErrorRel  = zeros(6+NroParam, nroCombinaciones);
ErrorRel(1:6,:) = errores(:,2,:);
ErrorRel(7:7+NroParam-1,:) = VectCar';
end
```

TFM_protot_MinDist

```
function [ M, Prototipos ] = TFM_protot_MinDist( MCn, MAcum, VectCar )
% TFM_protot_MinDist genera el vector de prototipos de mínima distan-
cia.
%
% Construye la matriz de características M1 y la matriz de prototipos.
% Los parámetros de entrada son:
% - MCn: Matriz que contiene las características de la colección de
fotos.
% - MuestrasAcum: Vector que contiene la fila de comienzo de cada
clase.
% - VectCar: Vector que contiene las características específicas para
las
% que se generarán los prototipos.

%% Inicializaciones
NroParam = length(VectCar);
NroClases = length(MAcum);
NroCasos = MAcum(5);
M = zeros(NroParam+1,MAcum(5));

%% Comprobaciones previas
[n,m] = size(MCn);

if n ~= NroCasos
```

```
    warning('Las dimensiones de MCn y MuestrasAcum no son coherentes')
end
if m < NroParam
    warning('Exceso de características requerido en VectCar');
end
if NroClases < 2
    warning('El número de clases debe ser mayor de 1');
end

%% CONSTRUCCIÓN DE LA MATRIZ DE PARÁMETROS DE CLASIFICACIÓN
M(1:NroParam,:)=MCn(:,VectCar)';

M(NroParam+1,1:MAcum(1))=ones(1,MAcum(1));
for i = 1:NroClases-1
    M(NroParam+1,MAcum(i)+1:MAcum(i+1)) = i+1;
end

%% CÁLCULO DE PROTOTIPOS
%
% Se emplean los valores medios (centroides)

Prototipos  = zeros( NroParam, NroClases);

for clase = 1:NroClases
    abb = M(1:NroParam+1,find(M(NroParam,:) > 0));
    Prototipos(:,clase) = mean(abb(1:NroParam,find(abb(NroPa-
ram+1,:)==clase))');
end

end
```

TFM_protot_Kmeans

```
function [ M, Prototipos1 ] = TFM_protot_Kmeans( MCn, MAcum, VectCar )
% TFM_protot_Kmeans genera el vector de prototipos de K-means.
%
% Construye la matriz de características M1 y la matriz de prototipos.
% Los parámetros de entrada son:
% - MCn: Matriz que contiene las características de la colección de
fotos.
% - MuestrasAcum: Vector que contiene la fila de comienzo de cada
clase.
% - VectCar: Vector que contiene las características específicas para
las
% que se generarán los prototipos.

%% Inicializaciones
NroParam = length(VectCar);
NroClases = length(MAcum);
NroCasos = MAcum(5);
M = zeros(NroParam,MAcum(5));
Muestras = [ 0; MAcum ];

%% Comprobaciones previas
[n,m] = size(MCn);

if n ~= NroCasos
```

```
    warning('Las dimensiones de MCn y MuestrasAcum no son coherentes')
end
if m < NroParam
    warning('Exceso de características requerido en VectCar');
end
if NroClases < 2
    warning('El número de clases debe ser mayor de 1');
end

%% CONSTRUCCIÓN DE LA MATRIZ DE PARÁMETROS DE CLASIFICACIÓN

for i = 1:NroParam
    M(i,:) = MCn(:,VectCar(i));
end

%% CÁLCULO DE PROTOTIPOS
%
% Se emplean los valores medios (centroides)

[Clasif, Prototipos, SumD, Dist] = kmeans(M', NroClases);
Prototipos = Prototipos';

% Conteo de aciertos de cada clase por clase
conteo = zeros(length(MAcum),length(MAcum));
for i0 = 1:length(MAcum)
    for k0 = 1:length(MAcum)
        conteo(i0,k0) = length(find(Clasif(Muestras(i0) + 1 : Mues-
tras(i0+1))...
            == k0))/(Muestras(i0+1)-Muestras(i0));
    end
end

% Conteo agregado de aciertos
VectCar0 = perms(1:5);
for i0 = 1:length(VectCar0)
    conteo2(i0) = conteo(VectCar0(i0,1),1)+conteo(Vect-
Car0(i0,2),2)+...
        conteo(VectCar0(i0,3),3)+conteo(VectCar0(i0,4),4)+conteo(Vect-
Car0(i0,5),5);
end

% Selección del conjunto de características que maximiza los aciertos
% globales
seleccion   = min(find(conteo2 == max(conteo2)));

% Reordenamiento de las columnas de prototipos
[n_0,m_0]   = size(Prototipos);
Prototipos1 = zeros(n_0,m_0);
for i0 = 1:length(MAcum)
    Prototipos1(:,i0) = Prototipos(:,find(VectCar0(seleccion,:) ==
i0));
    M(NroParam+1,Muestras(i0)+1:Muestras(i0+1)) = i0;
end

end
```

TFM_protot_Neural

```
function [ Y, T0, MCn, Red_Clasif ] = TFM_protot_Neural( ...
    M1, MuestrasAcum, VectCar, nroNeuronas )
% TFM_protot_Neural genera el vector de prototipos de red neuronal.
%
% Construye la matriz de características M1 y la matriz de prototipos.
% Los parámetros de entrada son:
% - MCn: Matriz que contiene las características de la colección de
fotos.
% - MuestrasAcum: Vector que contiene la fila de comienzo de cada
clase.
% - VectCar: Vector que contiene las características específicas para
las
% que se generarán los prototipos.

%% Inicializaciones
NroParam  = length( VectCar );
MuestrasAcc = [0; MuestrasAcum];
% Definición de matrices
MCn = zeros( length(M1), NroParam+1);
% Extracción de los parámetros en estudio
for i = 1:NroParam
    MCn(:,i) = M1(:,VectCar(i));
end
MCn(MuestrasAcc(1)+1:MuestrasAcc(1+1),NroParam+1) = 1;
% Establemcimiento del tipo de nube
for i = 2:length(MuestrasAcum)
    MCn(MuestrasAcc(i)+1:MuestrasAcc(i+1),NroParam+1) = i;
end
MCn = MCn';
% Definición de matrices
P = MCn(1:NroParam,:);
T0 = MCn(NroParam+1,:);
NroCasos = size(T0, 2);
T = zeros(5,NroCasos);
Y0 = zeros(5,NroCasos,length(nroNeuronas));
Y = zeros(1,NroCasos,length(nroNeuronas));
% Definición de los valores de las neuronas de salida con valores
[0,1]
for j0 = 1:NroCasos
    switch T0(1,j0)
        case 1
            T(:,j0) = [1,0,0,0,0];
        case 2
            T(:,j0) = [0,1,0,0,0];
        case 3
            T(:,j0) = [0,0,1,0,0];
        case 4
            T(:,j0) = [0,0,0,1,0];
        case 5
            T(:,j0) = [0,0,0,0,1];
    end
end

for i = 1:length(nroNeuronas)
    % Creación de la Red Neuronal
    Red_Clasif=feedforwardnet( nroNeuronas(i) );
```

```
    % Configuración
    Red_Clasif = configure(Red_Clasif, P, T);

    % Establecimiento de los ratios de entrenamiento, validación y ve-
rific.
    Red_Clasif.divideParam.trainRatio=0.75;
    Red_Clasif.divideParam.valRatio=0.15;
    Red_Clasif.divideParam.testRatio=0.10;
    % Número de iteraciones máximas y algoritmo de entrenamiento
    Red_Clasif.trainParam.epochs=1000;
    Red_Clasif.trainFcn='trainlm';
    % Deshabilitar las ventanas secundarias de entrenamiento
    Red_Clasif.trainParam.ShowWindow=false;
    Red_Clasif.trainParam.ShowCommandLine=false;
    % Entrenamiento con datos de entrenamiento y validación
    Red_Clasif=train(Red_Clasif, P, T);
    % Evaluación de la red
    Y0(1:5,:,i) = Red_Clasif(P);
    % Conversión de la salida a un vector de valores enteros
    for j0 = 1:NroCasos
        Y(1,j0,i) = find(Y0(:,j0,i) == max(Y0(:,j0,i)));
    end
end

end
```

TFM_clasif_MinDist

```
function [ clasek, errores ] = TFM_clasif_MinDist( M1, Prototipos,
MAcum )
% TFM_clasif_MinDist evalúa las características en M1 vs. los prototi-
pos.
%
% Esta función evalúa las características de una colección de imágenes
% respecto a los prototipos dados y las clases dadas y genera un vec-
tor con
% los errores obtenidos, totales y relativos, para cada clase reque-
rida.
%
% Los parámetros de entrada son:
% - M1: Matriz que contiene las características de la colección de fo-
tos.
% - Prototipos: Matriz con los centroides de cada clase requerida.
% - MuestrasAcum: Vector que contiene la fila de comienzo de cada
clase.
% - minorac: Factor de minoración a la cantidad de fotos a evaluar, en
caso
% de que la colección de características sea muy grande.

%% Inicializaciones
[n,m]       = size(M1);
NroParam    = n-1;
NroClases   = length( MAcum );
errores     = zeros( NroClases+1,2 );
Muestras    = [ 0; MAcum ];
Prototipos2 = [];
M2 = [];
dist3 = zeros(5,MAcum(length(MAcum)));
```

```
% Reescribir las matrices M1 y Prototipos como vector para el cálculo
% matricial
for i=1:NroParam
    Prototipos2 = [Prototipos2,Prototipos(i,:)];
    M2 = [M2; M1(i,:); M1(i,:); M1(i,:); M1(i,:); M1(i,:)];
end

%% CLASIFICACIÓN
%
% Cálculo de las distancias de cada clase a cada clase
dist2 = (Prototipos2'*ones(1,m)-M2).^2;
% Suma para obtener a cada clase
for i = 1:NroParam
    dist3 = dist3+dist2((i-1)*5+1:(i-1)*5+5,:);
end
% Asume que la clase a la que pertenece es la que menor distancia acu-
mula
[~,clasek]=min(dist3);
clasek = clasek';

%% CÁLCULO DE ERRORES GLOBAL Y RELATIVOS
% Calcula la tasa de errores globales
for i=1:NroClases
    errores(i,1)=length(find((clasek'-M1(NroParam+1,:))~=0 & ...
        M1(NroParam+1,:)==i));
end
% Error total de los errores globales
errores (NroClases+1,1) = sum(errores(1:NroClases,1));
% Cálculos de los errores relativos
for i = 1:NroClases
    errores(i,2) = 100 * errores(i,1)/ceil((Muestras(i+1)-Mues-
tras(i)));
end
% Error total de los errores relativos
errores (NroClases+1,2) = 100 * errores (NroClases+1,1) / ...
    ceil(Muestras(NroClases+1));

end
```

TFM_clasif_Neural

```
function [ errores ] = TFM_clasif_Neural( Y, T, MuestrasAcum )
% TFM_clasif_Neural evalúa las características en M1 vs. los prototi-
pos.
%
% Esta función evalúa las características de una colección de imágenes
% respecto a los prototipos dados y las clases dadas y genera un vec-
tor con
% los errores obtenidos, totales y relativos, para cada clase reque-
rida.
%
% Los parámetros de entrada son:
% - M1: Matriz que contiene las características de la colección de fo-
tos.
% - Prototipos: Matriz con los centroides de cada clase requerida.
% - MuestrasAcum: Vector que contiene la fila de comienzo de cada
clase.
```

```
% - minorac: Factor de minoración a la cantidad de fotos a evaluar, en
caso
% de que la colección de características sea muy grande.

%% Inicializaciones
NroClases   = length( MuestrasAcum );
errores = zeros( 6, 2 );
ErrorAcum = zeros(1,length(MuestrasAcum));

%% CLASIFICACIÓN
% Error de clasificación de clase caso por caso
Error_cl = Y-T;
% Se descartan errores menores de 0.5, ya que se asumen bien clasifi-
cados
distancia = find(abs(Error_cl)>0.5);
% Contabilización de errores
for j = 1:length(MuestrasAcum)
    distancia2 = find(distancia<MuestrasAcum(j));
    ErrorAcum(j)=length(distancia2);
    if j==1
        errores(j,1) = ErrorAcum(j);
        errores(j,2) = errores(j,1)/MuestrasAcum(j);
    else
        errores(j,1) = ErrorAcum(j) - ErrorAcum(j-1);
        errores(j,2) = errores(j,1)/(MuestrasAcum(j)-MuestrasAcum(j-
1));
    end
end
errores(length(MuestrasAcum)+1,1) = sum(errores(1:NroClases,1));
errores(length(MuestrasAcum)+1,2) = ...
    errores(length(MuestrasAcum)+1,1) / MuestrasAcum(NroClases);

end
```

TFM_Segm_Protot_Crom(·)

```
function [ Protot ] = TFM_Segm_Protot_Crom( Carac, Clasif )
% TFM_Segm_Protot_Crom  Función para la generación de prototipos cro-
máticos
%
% La variable de entrada Carac es una lista de vectores del tipo:
% Car={[1,2]}, o bien,
% Car = {[1,2], [3,4], [5,6]}
%
% La función salva los prototipos en los archivos .MAT siguientes:
% P1S4N1_MinDist, P1S4N1_Kmeans y P1S4N1_Neural.

% Carga el vector de características cromáticas para imágenes mues-
treadas
load('MCn_P1S4N1')
% Inicializaciones
Prototipos_P1S4N1_MinDist = [];
Prototipos_P1S4N1_Kmeans = [];
Red_Clasif = [];

% Selección de las características
switch length(Carac)
    case 0
```

```
        Carac = [ 33, 552 ];
        CaracMD = Carac;        CaracKM = Carac;        CaracRN = Carac;
    case 1
        CaracMD = Carac{1};     CaracKM = Carac{1};     CaracRN = Ca-
rac{1};
    case 3
        CaracMD = Carac{1};     CaracKM = Carac{2};     CaracRN = Ca-
rac{3};
    otherwise
        warning('Tamaño de la variable de entrada "Carac" incorrec-
ta');
        Protot = [];
        return
end

if Clasif(1) == 1
    % Generación de los prototipos del clasificador de Mínima Distan-
cia
    [ ~, Prototipos_P1S4N1_MinDist ] = TFM_protot_MinDist( ...
        MCn1_P1S4N1, MuestrasAcum1_P1S4N1, CaracMD );
    save('P1S4N1_MinDist','Prototipos_P1S4N1_MinDist','CaracMD')
end
if Clasif(2) == 1
    % Generación de los prototipos del clasificador K-means
    clear M Prototipos
    [ ~, Prototipos_P1S4N1_Kmeans ] = TFM_protot_Kmeans( ...
        MCn1_P1S4N1, MuestrasAcum1_P1S4N1, CaracKM );
    save('P1S4N1_Kmeans','Prototipos_P1S4N1_Kmeans', 'CaracKM')
end
if Clasif(3) == 1
    % Generación de los prototipos del clasificador neuronal
    clear M Prototipos
    [ ~, ~, ~, Red_Clasif ] = TFM_protot_Neural( ...
        MCn1_P1S4N1, MuestrasAcum1_P1S4N1, CaracRN, 50 );
    save('P1S4N1_Neural','Red_Clasif', 'CaracRN')
end

Protot = {Prototipos_P1S4N1_MinDist, Prototipos_P1S4N1_Kmeans,
Red_Clasif};
```

TFM_Segm_Protot_Txtl(·)

```
function [ Protot ] = TFM_Segm_Protot_Txtl( Carac, Clasif )
% TFM_Segm_Protot_Txtl  Función para la generación de prototipos tex-
turales
%
% La variable de entrada Carac es una lista de vectores del tipo:
% Car={[1,2,3,4]}, o bien,
% Car = {[1,2,3,4], [3,4,5,6], [5,6,7,8]}
%
% La función salva los prototipos en los archivos .MAT siguientes:
% P2_MinDist, P2_KMeans y P2_Neural.

% Carga el vector de características cromáticas para imágenes mues-
treadas
load('MCn_P2S1N1'); MCn = MCn_P2S1N1; MuestrasAcum = Mues-
trasAcum_P2S1N1;
% Inicializaciones
```

```
Prototipos_P2_MinDist = [];
Prototipos_P2_KMeans = [];
Red_Clasif_P2 = [];

% Selección de las características
switch length(Carac)
    case 0
        CaracT_MD = [ 7, 10, 14, 18 ];
        CaracT_KM = [ 4,  8, 15, 18 ];
        CaracT_RN = [ 7, 10, 14, 16 ];
    case 1
        CaracT_MD = Carac{1}; CaracT_KM = Carac{1}; CaracT_RN = Ca-
rac{1};
    case 3
        CaracT_MD = Carac{1}; CaracT_KM = Carac{2}; CaracT_RN = Ca-
rac{3};
    otherwise
        warning('Tamaño de la variable de entrada "Carac" incorrec-
ta');
        Protot = [];
        return
end

if Clasif(1) == 1
    % Generación de los prototipos del clasificador de Mínima Distan-
cia
    [ ~, Prototipos_P2_MinDist ] =TFM_protot_MinDist( MCn, Mues-
trasAcum,...
        CaracT_MD );
    save('P2_MinDist','Prototipos_P2_MinDist','CaracT_MD')
end
if Clasif(2) == 1
    % Generación de los prototipos del clasificador K-means
    [ ~, Prototipos_P2_KMeans ] = TFM_protot_Kmeans( MCn, Mues-
trasAcum,...
        CaracT_KM );
    save('P2_KMeans' ,'Prototipos_P2_KMeans','CaracT_KM')
end
if Clasif(3) == 1
    % Generación de los prototipos del clasificador neuronal
    [ ~, ~, ~, Red_Clasif_P2 ] = TFM_protot_Neural( MCn, Mues-
trasAcum,...
        CaracT_RN, 50 );
    save('P2_Neural' ,'Red_Clasif_P2','CaracT_RN')
end
Protot = {Prototipos_P2_MinDist, Prototipos_P2_KMeans, Red_Clasif_P2};
```

TFM_Segm_Protot_Mixta(·)

```
function [ Protot ] = TFM_Segm_Protot_Mixta( nroNeuronas0, Carac, Cla-
sif )
% TFM_Segm_Protot_Txtl  Función para la generación de prototipos tex-
turales
%
% Las características cromáticas y texturales a emplear son las utili-
zadas
% en los entrenamientos específicos de cada una de éstas.
%
```

```
% La función salva los prototipos en los archivos .MAT siguientes:
% P1y2S4N1_MinDist, P1y2S4N1_KMeans y P1y2S4N1_Neural.

% Carga el vector de características cromáticas para imágenes mues-
treadas:
% Datos Cromáticos
load MCn_P1S4N1
load P1S4N1_MinDist
load P1S4N1_Kmeans
load P1S4N1_Neural
% Datos Texturales
load MCn_P2S4N1
load P2_MinDist
load P2_Kmeans
load P2_Neural
load ('MCn_P2S1N1','MAXe_P2S1N1')

% Inicializaciones
Prototipos_P1y2S4N1_MinDist = [];
Prototipos_P1y2S4N1_KMeans = [];
Red_Clasif_P1y2S4N1_Neural = [];

if exist('MCn_P2S1N1','var')
    MCn_P2 = MCn_P2S1N1;
elseif exist('MCn_P2S4N1','var')
    MCn_P2 = MCn_P2S4N1;
end
% Vector de características mixtas: [ 4 x Texturales; 2 x Cromáticas]
if isempty(Carac)
    Carac_P1y2_MD  = [ CaracT_MD, CaracMD ];
    Carac_P1y2_KM  = [ CaracT_KM, CaracKM ];
    Carac_P1y2_RN  = [ CaracT_RN, CaracRN ];
else
    switch length(Carac)
        case 1
            Carac_P1y2_MD  = Carac{1};
            Carac_P1y2_KM  = Carac{1};
            Carac_P1y2_RN  = Carac{1};
        case 3
            Carac_P1y2_MD  = Carac{1};
            Carac_P1y2_KM  = Carac{2};
            Carac_P1y2_RN  = Carac{3};
    end
    CaracT_MD = Carac_P1y2_MD(1:4);  CaracMD = Carac_P1y2_MD(5:6);
    CaracT_KM = Carac_P1y2_KM(1:4);  CaracKM = Carac_P1y2_KM(5:6);
    CaracT_RN = Carac_P1y2_RN(1:4);  CaracRN = Carac_P1y2_RN(5:6);
end
nroClases = length(MuestrasAcum_P1S4N1);
% Inicialiaciones
Muestras = [ 0; MuestrasAcum_P1S4N1 ];
[ ~, m1 ] = size(MCn_P1S4N1);
[ ~, m2 ] = size(MCn_P2);
if isempty(nroNeuronas0)
    nroNeuronas = 100;
else
    nroNeuronas = nroNeuronas0;
end

% Construcción de la matriz de características combinada:
```

```
% - Las primeras 560 columnas corresponden a características cromáti-
cas, y
% - las  últimas  22 columnas corresponden a las características tex-
turales
%
% Se asume que el análisis cromático y textural se ha realizado pre-
viamente
% sobre las mismas imágenes (redimensionada a 20x20 píxeles en el caso
% cromático y considerando 4 ángulos: 0,90,180 y 270 en el textural).
La
% relación resultante es 1:100 entre las características cromáticas y
% texturales.

MCn_Mixto0 = MCn_P1S4N1;
k = 0;
for i = 1:length(MCn_P2)
    for j = 1:100
        k = k+1;
        MCn_Mixto0(k,m1+1:m1+m2) = MCn_P2(i,:);
    end
end
% Construcción del vector de características mixtas específico
MCn_Mixto_MinDist = MCn_Mixto0(:,[CaracMD, 560 + CaracT_MD]);
MCn_Mixto_KMeans  = MCn_Mixto0(:,[CaracKM, 560 + CaracT_KM]);
MCn_Mixto_Neural  = MCn_Mixto0(:,[CaracMD, 560 + CaracT_RN]);
% Adición de la columna con las clases de nubes correctas
for i=1:nroClases
    for j = Muestras(i)+1 : Muestras(i+1)
        MCn_Mixto_MinDist(j,length(CaracMD)+length(CaracT_MD)+1) = i;
        MCn_Mixto_KMeans (j,length(CaracKM)+length(CaracT_KM)+1) = i;
        MCn_Mixto_Neural (j,length(CaracRN)+length(CaracT_RN)+1) = i;
    end
end
if Clasif(1) == 1
    % Generación de los prototipos del clasificador de Mínima Distan-
cia
    [ ~, Prototipos_P1y2S4N1_MinDist ] = TFM_protot_MinDist( ...
        MCn_Mixto_MinDist, MuestrasAcum_P1S4N1, [1,2,3,4,5,6] );
    save('P1y2S4N1_MinDist','Prototipos_P1y2S4N1_MinDist','Ca-
rac_P1y2_MD')
end
if Clasif(2) == 1
    % Generación de los prototipos del clasificador K-means
    [ ~, Prototipos_P1y2S4N1_KMeans ] = TFM_protot_Kmeans( ...
        MCn_Mixto_KMeans, MuestrasAcum_P1S4N1, [1,2,3,4,5,6] );
    save('P1y2S4N1_KMeans','Prototipos_P1y2S4N1_KMeans','Ca-
rac_P1y2_KM')
end
if Clasif(3) == 1
    % Minoración previa para minimizar el tiempo de cálculo de la red
neuronal
    [MCn1_Neural, MuestrasAcum1_Neural] = TFM_Minorac( 1,
MCn_Mixto_Neural,...
        MuestrasAcum_P1S4N1, 0.05 );
    % Generación de los prototipos del clasificador neuronal
    [ ~, ~, ~, Red_Clasif_P1y2S4N1_Neural ] = TFM_protot_Neural( ...
        MCn1_Neural, MuestrasAcum1_Neural, [1,2,3,4,5,6], nroNeuronas
);
    save('P1y2S4N1_Neural','Red_Clasif_P1y2S4N1_Neural','Ca-
rac_P1y2_RN')
```

```
end
Protot = {Prototipos_P1y2S4N1_MinDist, Prototipos_P1y2S4N1_KMeans,...
    Red_Clasif_P1y2S4N1_Neural};
```

TFM_Segm_SegmInd_Crom(·)

```
function [ ] = TFM_Segm_SegmInd_Crom( nroImag )
% TFM_Segm_SegmInd_Crom   Función que genera una segmentación cromá-
tica

% Carga de prototipos cromáticos
load P1S4N1_MinDist
load P1S4N1_Kmeans
load P1S4N1_Neural

% Inicializaciones
nroFotos    = 1;
Fin = 'img.png';
Letras = {'A_4', 'B_4', 'C_4', 'D_4', 'E_4'};

for i0 = 1:nroImag
    figure;
    for i=1:5
        % Carga de la imagen
        Inic_foto = [Letras{i},num2str(i0)];
        f1 = imread([Inic_foto,Fin]);
        [filas, columnas, ~] = size( f1 );

        % Extracción de características
        MC_crom = TFM_Crom_Features(Inic_foto, Fin, nroFotos);
        % Normalización
        MCn_crom = MC_crom ./ sqrt(3*(255^2));

        % Selección de características cromáticas
        M_crom = MCn_crom(:,CaracMD)';
        M_crom = 1.*[M_crom; zeros(1,length(M_crom))];
        % Clasificación / Segmentación
        [c0, ~] = TFM_clasif_MinDist( M_crom, Prototipos_P1S4N1_Min-
Dist,...
            length(M_crom)*ones(5,1) );
        aaa1 = (1/6)*reshape(c0,[filas,columnas]);

        % Clasificación K-means
        clear c0
        M_crom2 = MCn_crom(:,CaracKM)';
        M_crom3 = 1.*[M_crom2; zeros(1,length(M_crom2))];
        [c0, ~] = TFM_clasif_MinDist( M_crom3, Prototi-
pos_P1S4N1_Kmeans,...
            length(M_crom3)*ones(5,1) );
        aaa2 = (1/6)*reshape(c0,[filas,columnas]);

        % Clasificación Redes Neuronales
        clear c0
        clase_Neural = Red_Clasif(M_crom2);
        for j0 = 1:length(clase_Neural)
            c0(j0) = min(find(clase_Neural(:,j0) == max(clase_Neu-
ral(:,j0))));
```

```
        end
        aaa3 = (1/6)*reshape(c0(1,:),[filas,columnas]);

        % Representación
        subplot(5,5,1+(i-1));  imshow(f1);
        subplot(5,5,6+(i-1));  imshow(aaa1);
        subplot(5,5,11+(i-1)); imshow(aaa2);
        subplot(5,5,16+(i-1)); imshow(aaa3);
        subplot(5,5,21+(i-1)); imshow((i/6)*ones(filas,columnas));
        clear f1 aaa1 aaa2 aaa3 c0
    end
end
```

TFM_Segm_SegmInd_Txtl(·)

```
function [ ] = TFM_Segm_SegmInd_Txtl( nroImag )
% TFM_Segm_SegmInd_Crom   Función que genera una segmentación textural

% Carga de prototipos texturales
load P2_MinDist
load P2_Kmeans
load P2_Neural
load ('MCn_P2S1N1','MAXe_P2S1N1')

% Inicializaciones
Entorno = 64;
Fin = 'img.png';
Letras = {'A_4', 'B_4', 'C_4', 'D_4', 'E_4'};

for i0=1:nroImag
    figure;
    for i=1:5
        % Carga de la imagen
        Inic_foto = [Letras{i},num2str(i0)];
        f1 = imread([Inic_foto,Fin]);
        [filas, columnas, ~] = size( f1 );

        % Definición de matrices
        clase_MinDist = zeros(filas-Entorno, columnas-Entorno);
        clase_Kmeans = zeros(filas-Entorno, columnas-Entorno);
        clase_Neural = zeros(filas-Entorno, columnas-Entorno);

        % Extracción de características texturales de cada entorno
        for k1 = 1:filas-Entorno
            for k2 = 1:columnas-Entorno
                % Carga de la subimagen
                imagen = f1(k1:k1+(Entorno),k2:k2+(Entorno),:);
                % Extracción de características de la subimagen
                MC_txtl = TFM_GLCM_Indiv( imagen );
                % Normalización de características
                [ MCn_txtl, ~ ] = TFM_Normaliz(MC_txtl, 2, 1,
MAXe_P2S1N1);

                % CLASIFICADOR DE MÍNIMA DISTANCIA
                % Selección de características
                M_txtl = MCn_txtl( :, CaracT_MD )';
                M_txtl = [M_txtl;0];
```

```
                % Clasificación / Segmentación
                [c0, ~] = TFM_clasif_MinDist( M_txtl,...
                    Prototipos_P2_MinDist, ones(5,1) );
                clase_MinDist(k1,k2) = c0(1);
                clear M_txtl  c0;

                % CLASIFICADOR K-MEANS
                % Selección de características
                M_txtl = MCn_txtl( :, CaracT_KM )';
                % Clasificación / Segmentación
                [c0, ~] = TFM_clasif_MinDist( M_txtl,...
                    Prototipos_P2_MinDist, ones(5,1) );
                clase_Kmeans(k1,k2) = c0(1);
                clear M_txtl c0;

                % CLASIFICADOR NEURONAL
                % Selección de características
                M_txtl = MCn_txtl( :, CaracT_RN )';
                % Clasificación / Segmentación
                Aux = round(Red_Clasif_P2(M_txtl));
                clase_Neural(k1,k2) = min(find(Aux == max(Aux)));
                if clase_Neural(k1,k2) < 1; clase_Neural(k1,k2) = 1;
end
                if clase_Neural(k1,k2) > 5; clase_Neural(k1,k2) = 5;
end

            end
            disp([num2str(k1), ' de ',num2str(filas-Entorno)]);
        end

        % Representación
        subplot(5,5,1+(i-1));  imshow(f1);
        subplot(5,5,6+(i-1));  imshow(clase_MinDist * (1/6));
        subplot(5,5,11+(i-1)); imshow(clase_Kmeans  * (1/6));
        subplot(5,5,16+(i-1)); imshow(clase_Neural  * (1/6));
        subplot(5,5,21+(i-1)); imshow((i/6)*ones(filas,columnas));

        disp(['Foto ',Inic_foto, Fin])
    end
end
```

TFM_Segm_SegmInd_Mixta(·)

```
function [ ] = TFM_Segm_SegmInd_Mixta( nroImag, ColecOIndiv )
% TFM_Segm_SegmInd_Mixta   Función que genera segmentaciones cromá-
tica,
%                          textural y mixta.

% Carga de prototipos cromáticos y texturales
% Prototipos cromáticos
load P1S4N1_MinDist
load P1S4N1_Kmeans
load P1S4N1_Neural
% Prototipos texturales
load P2_MinDist
load P2_Kmeans
load P2_Neural
load ('MCn_P2S1N1','MAXe_P2S1N1')
```

```
% Prototipos mixtos
load P1y2S4N1_MinDist
load P1y2S4N1_KMeans
load P1y2S4N1_Neural

% Inicializaciones
if ColecOIndiv == 1
    Letras = {'X_1', 'X_1', 'X_1', 'X_1', 'X_1', 'X_1', 'X_1', 'X_1'};
    Fin = 'img.jpg';
    Cuadros = 100;
    Class = [];
else
    Letras = {'A_5', 'B_5', 'C_2', 'D_2', 'E_2'};
    Fin = 'img.png';
    Cuadros = 10;
    Class = [1,2,3,4,5];
end

for i0=1:nroImag
    for i=1:length(Letras)
        %% INICIALIZACIONES Y CONFIGURACIONES PREVIAS
        % Carga de la imagen
        Inic_foto = [Letras{i},num2str(i+i0-2)];
        f_1 = imread([Inic_foto, Fin]);
        [filas, columnas, ~] = size( f_1 );

        % Redimensionamiento de la imagen para el procesamiento textu-
ral
        f1 = imresize(f_1, [ 10*floor(filas/Cuadros),...
            10*floor(columnas/Cuadros) ] );
        [filas1, columnas1, ~] = size( f1 );
        % Se guarda la imagen intermedia para poder usarla en la fun-
ción
        % TFM_CromFeatures.
        imwrite(f1,'AuxIm.png');

        % Definición de matrices y variables
        clase_MinDist_C = zeros(floor(filas/100), floor(colum-
nas/100));
        clase_Kmeans_C = clase_MinDist_C;
        clase_Neural_C = clase_MinDist_C;
        M_txtl_MinDist1 = zeros(floor(filas/100) * floor(colum-
nas/100),...
            length(CaracT_MD));
        M_txtl_KMeans1  = M_txtl_MinDist1;
        M_txtl_Neural1  = M_txtl_MinDist1;
        index = 0;

        %% SEGMENTACIÓN TEXTURAL
        for k2 = 1:floor(columnas/Cuadros)
            for k1 = 1:floor(filas/Cuadros)
                index = index + 1;
                % Carga de la subimagen
                imagen = f_1((k1-1)*Cuadros+1:(k1-1)*Cuadros+Cua-
dros,...
                    (k2-1)*Cuadros+1:(k2-1)*Cuadros+Cuadros,:);
                % Extracción de características texturales y normali-
zación
                MC_txtl = TFM_GLCM_Indiv( imagen );
```

```
                [ MCn_txtl, ~ ] = TFM_Normaliz(MC_txtl, 2, 1,
MAXe_P2S1N1);

                % CLASIFICADOR DE MÍNIMA DISTANCIA
                % Selección de características
                M_txtl_MinDist = MCn_txtl(:,CaracT_MD)';
                % Almacenamiento separado para segmentación mixta
                M_txtl_MinDist1(index,:) = M_txtl_MinDist';
                M_txtl_MinDist = [M_txtl_MinDist;0];
                % Clasificación / Segmentación
                [c0, ~] = TFM_clasif_MinDist( M_txtl_MinDist,...
                    Prototipos_P2_MinDist, ones(5,1) );
                clase_MinDist_C(k1,k2) = c0(1);
                clear c0

                % CLASIFICADOR K-MEANS
                % Selección de características
                M_txtl_Kmeans = MCn_txtl(:,CaracT_KM)';
                % Almacenamiento separado para segmentación mixta
                M_txtl_Kmeans2 = MCn_txtl(:,Carac_P1y2_KM(1:4))';
                M_txtl_KMeans1(index,:) = M_txtl_Kmeans2;
                % Clasificación / Segmentación
                [c0, ~] = TFM_clasif_MinDist( [M_txtl_Kmeans;0],...
                    Prototipos_P2_KMeans, ones(5,1) );
                clase_Kmeans_C(k1,k2) = c0(1);
                clear c0

                % CLASIFICADOR NEURONAL
                % Selección de características
                M_txtl_Neural = MCn_txtl(:,CaracT_RN)';
                % Almacenamiento separado para segmentación mixta
                M_txtl_Neural1(index,:) = M_txtl_Neural;
                % Clasificación / Segmentación
                Aux= round(Red_Clasif_P2(M_txtl_Neural));
                clase_Neural_C(k1,k2) = min(find(Aux == max(Aux)));
                if clase_Neural_C(k1,k2) < 1; clase_Neural_C(k1,k2) =
1;end
                if clase_Neural_C(k1,k2) > 5; clase_Neural_C(k1,k2) =
5;end
            end
            disp([num2str(k2), ' de ', num2str(floor(columnas/Cua-
dros))]);
        end

        %% SEGMENTACIÓN CROMÁTICA
        clear c0;
        % Extracción de características cromáticas y normalización
        MC_crom = TFM_Crom_Features('AuxIm','.png', 1);
        MCn_crom = MC_crom ./ sqrt(3*255^2);
        nroCar = size(MCn_crom,2);
        % Duplicado para la segmentación mixta
        MCn_crom_2 = MCn_crom;

        % CLASIFICADOR DE MÍNIMA DISTANCIA
        % Selección de características cromáticas
        M_crom_MinDist0 = MCn_crom( :, CaracMD )';
        M_crom_MinDist0 = [M_crom_MinDist0; ...
            zeros(1,length(M_crom_MinDist0))];
        % Clasificación / Segmentación
```

```
        [c0, ~] = TFM_clasif_MinDist( M_crom_MinDist0,...
            Prototipos_P1S4N1_MinDist, length(M_crom_Min-
Dist0)*ones(5,1) );
        aaa1 = (1/6)*reshape(c0,[filas1,columnas1]);
        clear c0

        % CLASIFICADOR K-MEANS
        % Selección de características cromáticas
        M_crom_Kmeans = MCn_crom( :, CaracKM )';
        M_crom_Kmeans0 = [M_crom_Kmeans; zeros(1,len-
gth(M_crom_Kmeans))];
        % Clasificación / Segmentación
        [c0, ~] = TFM_clasif_MinDist( M_crom_Kmeans0,...
            Prototipos_P1S4N1_Kmeans, length(M_crom_Kmeans0)*ones(5,1)
);
        aaa2 = (1/6)*reshape(c0,[filas1,columnas1]);
        clear c0

        % CLASIFICADOR NEURONAL
        % Selección de características cromáticas
        M_crom_Neural = MCn_crom(:,CaracMD)';
        % Clasificación / Segmentación
        clase_Neural = Red_Clasif(M_crom_Neural);
        for j0 = 1:length(clase_Neural)
            c0(j0) = min(find(clase_Neural(:,j0) == max(clase_Neu-
ral(:,j0))));
        end
        aaa3 = (1/6)*reshape(c0(1,:),[filas1,columnas1]);

        %% SEGMENTACIÓN MIXTA CROMÁTICO-TEXTURAL
        %
        % CLASIFICADOR DE MÍNIMA DISTANCIA
        %
        % Integración de características cromáticas y texturales en un
        % solo vector de características
        M_crom_MinDist1 = MCn_crom_2( :, CaracMD )';
        % La información de las características texturales debe coin-
cidir
        % con la contenida por el vector cromático en cada pixel, y
tener
        % las mismas dimensiones. Por ello se debe expandir el vector
        % textural para tener el mismo n° de filas y columnas que el
        % cromático.
        % Primero se reestructura de vector a matriz-imagen
        M_txtl_MinDist01 = reshape(M_txtl_MinDist1,[floor(filas/Cua-
dros),...
            floor(columnas/Cuadros), length(CaracT_MD)]);
        % Segundo, cada pixel textural se corresponde con un entorno
10x10
        % del vector de características cromático
        for i2 = 1:filas1/10
            for i3 = 1:columnas1/10
                for i4 = 1:length(CaracT_MD)
                    M_txtl_MinDist02((i2-1)*10+1:(i2-1)*10+10,...
                        (i3-1)*10+1:(i3-1)*10+10,i4) = ...
                        M_txtl_MinDist01(i2,i3,i4)*ones(10);
                end
            end
        end
        % Finalmente se vuelve a estructurar como vector
```

```
        M_txtl_MinDist2 = reshape(M_txtl_MinDist02, ...
            [filas1 * columnas1,length(CaracT_MD)]);
        % Se integran en un solo vector
        M_Mixto_MinDist0 = [M_crom_MinDist1', M_txtl_MinDist2];
        % Se añade una columna vacía
        M_Mixto_MinDist = [M_Mixto_MinDist0,zeros((filas1)*(colum-
nas1),1)];
        % Clasificación / Segmentación
        [c0, ~] = TFM_clasif_MinDist( M_Mixto_MinDist', ...
            Prototipos_P1y2S4N1_MinDist,length(M_Mixto_Min-
Dist)*ones(5,1));
        bbb1 = (1/6)*reshape(c0,[(filas1),(columnas1)]);
        clear c0;

        % CLASIFICADOR K-MEANS
        %
        % Integración de características cromáticas y texturales
        M_txtl_KMeans03 = reshape(M_txtl_KMeans1,[floor(filas/Cua-
dros),...
            floor(columnas/Cuadros), length(CaracT_KM)]);
        for i2 = 1:filas1/10
            for i3 = 1:columnas1/10
                for i4 = 1:length(CaracT_KM)
                    M_txtl_KMeans04((i2-1)*10+1:(i2-1)*10+10,...
                        (i3-1)*10+1:(i3-1)*10+10,i4) = ...
                        M_txtl_KMeans03(i2,i3,i4)*ones(10);
                end
            end
        end
        M_txtl_KMeans4 = reshape( M_txtl_KMeans04,...
            [ filas1 * columnas1,length(CaracT_KM) ] );
        % Integración en un solo vector
        M_Mixto_KMeans0 = [M_crom_MinDist1', M_txtl_KMeans4];
        M_Mixto_KMeans  = [M_Mixto_KMeans0,zeros((filas1)*(colum-
nas1),1)];
        % Clasificación / Segmentación
        [c0, ~] = TFM_clasif_MinDist( M_Mixto_KMeans',...
            Prototipos_P1y2S4N1_KMeans, len-
gth(M_Mixto_KMeans0)*ones(5,1));
        bbb2 = (1/6)*reshape(c0,[(filas1),(columnas1)]);
        clear c0

        % CLASIFICADOR NEURONAL
        %
        % Integración de características cromáticas y texturales
        M_txtl_Neural01 = reshape(M_txtl_Neural1,[floor(filas/Cua-
dros),...
            floor(columnas/Cuadros), length(CaracT_KM)]);
        for i2 = 1:filas1/10
            for i3 = 1:columnas1/10
                for i4 = 1:length(CaracT_RN)
                    M_txtl_Neural02((i2-1)*10+1:(i2-1)*10+10,...
                        (i3-1)*10+1:(i3-1)*10+10,i4) = ...
                        M_txtl_Neural01(i2,i3,i4)*ones(10);
                end
            end
        end
        M_txtl_Neural2 = reshape(M_txtl_Neural02,[filas1 * colum-
nas1,length(CaracT_RN)]);
        % Integración en un solo vector
```

```
        M_Mixto_Neural0 = [M_crom_MinDist1', M_txtl_Neural2];
        % Clasificación / Segmentación
        claseP1y2_Neural = Red_Clasif_P1y2S4N1_Neural(M_Mixto_Neu-
ral0');
        for j0 = 1:length(claseP1y2_Neural)
            c0(j0) = min(find(claseP1y2_Neural(:,j0) == ...
                max(claseP1y2_Neural(:,j0))));
        end
        bbb3 = (1/6)*reshape(c0,[(filas1),(columnas1)]);

        %% Representación
        figure;
        subplot(4,3,1);   imshow(f1);
                          title('Imagen original');
        if ~isempty(Class)
            subplot(4,3,2);imshow((Class(i)/6)*ones(filas1,colum-
nas1));
                          title('Clasificación correcta');
        end
        subplot(4,3,4);   imshow(clase_MinDist_C * (1/6));
                          title('Mínima distancia - Textural');
        subplot(4,3,5);   imshow(clase_Kmeans_C * (1/6));
                          title('K-Means - Textural');
        subplot(4,3,6);   imshow(clase_Neural_C * (1/6));
                          title('Red Neuronal - Textural');
        subplot(4,3,7);   imshow(aaa1);
                          title('Mínima distancia - Cromático');
        subplot(4,3,8);   imshow(aaa2);
                          title('K-Means - Cromático');
        subplot(4,3,9);   imshow(aaa3);
                          title('Red Neuronal - Cromático');
        subplot(4,3,10);  imshow(bbb1);
                          title('Mínima distancia - Mixto');
        subplot(4,3,11);  imshow(bbb2);
                          title('K-Means - Mixto');
        subplot(4,3,12);  imshow(bbb3);
                          title('Red Neuronal - Mixto');

        clear aaa1 aaa2 aaa3 bbb1 bbb2 bbb3
        clear clase_MinDist_C clase_Kmeans_C clase_Neural_C
M_txtl_KMeans04
        clear M_txtl_MinDist02 M_txtl_MinDist2 M_txtl_KMeans4
    end
end
```

TFM_Segm_SegmInd_Mixta_PaP(·)

```
function [ ] = TFM_Segm_SegmInd_Mixta_PaP( nroImag, ColecOIndiv )
% TFM_Segm_SegmInd_Mixta   Función que genera segmentaciones cromá-
tica,
%                          textural y mixta, píxel a píxel

% Carga de prototipos cromáticos y texturales
% Prototipos cromáticos
load P1S4N1_MinDist
load P1S4N1_Kmeans
load P1S4N1_Neural
% Prototipos texturales
```

```
load P2_MinDist
load P2_Kmeans
load P2_Neural
load ('MCn_P2S1N1','MAXe_P2S1N1')
% Prototipos mixtos
load P1y2S4N1_MinDist
load P1y2S4N1_KMeans
load P1y2S4N1_Neural

% Inicializaciones
Entorno = 64;

if ColecOIndiv == 1
    Letras = {'X_1', 'X_1', 'X_1', 'X_1', 'X_1', 'X_1', 'X_1', 'X_1'};
    Fin = 'img.jpg';
    Class = [];
else
    Letras = {'A_2', 'B_2', 'C_2', 'D_2', 'E_2'};
    Fin = 'img.png';
    Class = [1,2,3,4,5];
end

for i0=1:nroImag
    for i=1:length(Letras)
        %% INICIALIZACIONES Y CONFIGURACIONES PREVIAS
        % Carga de la imagen
        Inic_foto = [Letras{i},num2str(i+i0-1)];
        f_1 = imread([Inic_foto, Fin]);
        [filas0, columnas0, ~] = size( f_1 );
        % Redimensionamiento de la imagen para el procesamiento textu-
ral
        f1=imresize(f_1,[125,125*columnas0/filas0]);
        [filas1, columnas1, ~] = size( f1 );
        filas = filas1-Entorno;
        columnas = columnas1-Entorno;
        % Se guarda la imagen intermedia para poder usarla en la fun-
ción
        % TFM_CromFeatures.
        imwrite(f1((Entorno/2)+1:filas1-(Entorno/2),...
            (Entorno/2)+1:columnas1-(Entorno/2),:),'AuxIm.png');

        % Definición de matrices y variables
        clase_MinDist_C = zeros(filas, columnas);
        clase_Kmeans_C  = zeros(filas, columnas);
        clase_Neural_C  = zeros(filas, columnas);
        M_txtl_MinDist1 = zeros(filas *columnas,length(CaracT_MD));
        M_txtl_KMeans3  = zeros(filas *columnas,length(CaracT_KM));
        M_txtl_Neural1  = zeros(filas *columnas,length(CaracT_RN));
        index = 0;

        %% SEGMENTACIÓN TEXTURAL
        for k1 = 1:columnas
            for k2 = 1:filas
                index = index + 1;
                % Carga de la subimagen
                imagen = f1(k2:k2+(Entorno),k1:k1+(Entorno),:);
                % Extracción de características texturales y normali-
zación
                MC_txtl = TFM_GLCM_Indiv( imagen );
```

```
                [ MCn_txtl, ~ ] = TFM_Normaliz(MC_txtl, 2, 1,
MAXe_P2S1N1);

                % CLASIFICADOR DE MÍNIMA DISTANCIA
                % Selección de características
                M_txtl_MinDist = MCn_txtl(:,CaracT_MD)';
                % Almacenamiento separado para segmentación mixta
                M_txtl_MinDist1(index,:) = M_txtl_MinDist';
                M_txtl_MinDist = [M_txtl_MinDist;0];
                % Clasificación / Segmentación
                [c0, ~] = TFM_clasif_MinDist( M_txtl_MinDist,...
                    Prototipos_P2_MinDist, ones(5,1) );
                clase_MinDist_C(k2,k1) = c0(1);
                clear c0

                % CLASIFICADOR K-MEANS
                % Selección de características
                M_txtl_Kmeans = MCn_txtl( :, CaracT_KM )';
                % Almacenamiento separado para segmentación mixta
                M_txtl_KMeans1(index,:) = M_txtl_Kmeans;
                M_txtl_Kmeans2 = MCn_txtl(:,Carac_P1y2_KM(1:4))';
                M_txtl_KMeans3(index,:) = M_txtl_Kmeans2;
                % Clasificación / Segmentación
                [c0, ~] = TFM_clasif_MinDist( [M_txtl_Kmeans;0],...
                    Prototipos_P2_KMeans, ones(5,1) );
                clase_Kmeans_C(k2,k1) = c0(1);
                clear c0

                % CLASIFICADOR NEURONAL
                % Selección de características
                M_txtl_Neural = MCn_txtl( :, CaracT_RN )';
                % Almacenamiento separado para segmentación mixta
                M_txtl_Neural1(index,:) = M_txtl_Neural;
                % Clasificación / Segmentación
                Aux= round(Red_Clasif_P2(M_txtl_Neural));
                clase_Neural_C(k2,k1) = min(find(Aux == max(Aux)));
                if clase_Neural_C(k2,k1) < 1; clase_Neural_C(k2,k1) =
1; end
                if clase_Neural_C(k2,k1) > 5; clase_Neural_C(k2,k1) =
5; end
            end
            disp([num2str(k1), ' de ', num2str(columnas)]);

        end


        %% SEGMENTACIÓN CROMÁTICA
        clear c0;
        % Extracción de características cromáticas y normalización
        MC_crom = TFM_Crom_Features('AuxIm','.png', 1);
        MCn_crom = MC_crom ./ sqrt(3*255^2);
        % Duplicado para la segmentación mixta
        MCn_crom_2 = MCn_crom;

        % CLASIFICADOR DE MÍNIMA DISTANCIA
        % Selección de características cromáticas
        M_crom_MinDist0 = MCn_crom( :, CaracMD )';
        M_crom_MinDist0 = [M_crom_MinDist0; ...
            zeros(1,length(M_crom_MinDist0))];
```

```
        % Clasificación / Segmentación
        [c0, ~] = TFM_clasif_MinDist( M_crom_MinDist0, ...
            Prototipos_P1S4N1_MinDist, length(M_crom_Min-
Dist0)*ones(5,1) );
        aaa1 = (1/6)*reshape(c0,[filas,columnas]);
        clear c0

        % CLASIFICADOR K-MEANS
        % Selección de características cromáticas
        M_crom_Kmeans = MCn_crom(:,CaracKM)';
        M_crom_Kmeans0 = 1.*[M_crom_Kmeans; zeros(1,len-
gth(M_crom_Kmeans))];
        [c0, ~] = TFM_clasif_MinDist( M_crom_Kmeans0,...
            Prototipos_P1S4N1_Kmeans, length(M_crom_Kmeans0)*ones(5,1)
);
        aaa2 = (1/6)*reshape(c0,[filas1-Entorno,columnas1-Entorno]);
        clear c0

        % CLASIFICADOR NEURONAL
        % Selección de características cromáticas
        M_crom_Neural = MCn_crom( :, CaracKM )';
        % Clasificación / Segmentación
        clase_Neural = Red_Clasif(M_crom_Neural);
        for j0 = 1:length(clase_Neural)
            c0(j0) = min(find(clase_Neural(:,j0) == max(clase_Neu-
ral(:,j0))));
        end
        aaa3 = (1/6)*reshape(c0(1,:),[filas1-Entorno,columnas1-En-
torno]);

        %% SEGMENTACIÓN MIXTA CROMÁTICO-TEXTURAL
        %
        % CLASIFICADOR DE MÍNIMA DISTANCIA
        %
        % Integración de características cromáticas y texturales en un
        % solo vector de características
        M_crom_MinDist1 = MCn_crom_2( :, CaracMD )';
        % Integración en un solo vector
        M_Mixto_MinDist0 = [M_crom_MinDist1', M_txtl_MinDist1];
        % Se añade una columna vacía
        M_Mixto_MinDist = [M_Mixto_MinDist0, zeros(filas*columnas,1)];
        % Clasificación / Segmentación
        [c0, ~] = TFM_clasif_MinDist( M_Mixto_MinDist', ...
            Prototipos_P1y2S4N1_MinDist,length(M_Mixto_Min-
Dist)*ones(5,1));
        bbb1 = (1/6)*reshape(c0,[(filas1-Entorno),(columnas1-En-
torno)]);
        clear c0;

        % CLASIFICADOR K-MEANS
        %
        % Integración de características cromáticas y texturales
        M_Mixto_KMeans0 = [M_crom_MinDist1', M_txtl_KMeans3];
        M_Mixto_KMeans  = [M_Mixto_KMeans0,zeros(filas*columnas,1)];
        % Clasificación / Segmentación
        [c0, ~] = TFM_clasif_MinDist( M_Mixto_KMeans',...
            Prototipos_P1y2S4N1_KMeans, len-
gth(M_Mixto_KMeans0)*ones(5,1));
```

```
        bbb2 = (1/6)*reshape(c0,[(filas1-Entorno),(columnas1-En-
torno)]);
        clear c0;

        % CLASIFICADOR NEURONAL
        %
        % Integración de características cromáticas y texturales
        M_Mixto_Neural0 = [M_crom_MinDist1', M_txtl_Neural1];
        % Clasificación / Segmentación
        claseP1y2_Neural = Red_Clasif_P1y2S4N1_Neural(M_Mixto_Neu-
ral0');
        for j0 = 1:length(claseP1y2_Neural)
            c0(j0) = min(find(claseP1y2_Neural(:,j0) == max(cla-
seP1y2_Neural(:,j0))));
        end
        bbb3 = (1/6)*reshape(c0,[(filas1-Entorno),(columnas1-En-
torno)]);

        %% Representación
        figure;
        subplot(4,3,1);   imshow(f1((Entorno/2)+1:filas1-(En-
torno/2),...
            (Entorno/2)+1:columnas1-(Entorno/2),:));
                          title('Imagen original');
        if ~isempty(Class)
            subplot(4,3,2);imshow((Class(i)/6)*ones(filas1,colum-
nas1));
                          title('Clasificación correcta');
        end
        subplot(4,3,4);   imshow(clase_MinDist_C * (1/6));
                          title('Mínima distancia - Textural');
        subplot(4,3,5);   imshow(clase_Kmeans_C * (1/6));
                          title('K-Means - Textural');
        subplot(4,3,6);   imshow(clase_Neural_C * (1/6));
                          title('Red Neuronal - Textural');
        subplot(4,3,7);   imshow(aaa1);
                          title('Mínima distancia - Cromático');
        subplot(4,3,8);   imshow(aaa2);
                          title('K-Means - Cromático');
        subplot(4,3,9);   imshow(aaa3);
                          title('Red Neuronal - Cromático');
        subplot(4,3,10);  imshow(bbb1);
                          title('Mínima distancia - Mixto');
        subplot(4,3,11);  imshow(bbb2);
                          title('K-Means - Mixto');
        subplot(4,3,12);  imshow(bbb3);
                          title('Red Neuronal - Mixto');

        clear aaa1 aaa2 aaa3 bbb1 bbb2 bbb3
        clear clase_MinDist_C clase_Kmeans_C clase_Neural_C
    end
end
```

TFM_Menu(·)

```
TFM_MENU_1, by itself, creates a new TFM_MENU_1 or raises the existing
%      singleton*.
%
%      H = TFM_MENU_1 returns the handle to a new TFM_MENU_1 or the
handle to
%      the existing singleton*.
%
%      TFM_MENU_1('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in TFM_MENU_1.M with the given input
arguments.
%
%      TFM_MENU_1('Property','Value',...) creates a new TFM_MENU_1 or
raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before TFM_Menu_1_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property ap-
plication
%      stop.  All inputs are passed to TFM_Menu_1_OpeningFcn via va-
rargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help TFM_Menu_1

% Last Modified by GUIDE v2.5 15-Nov-2016 19:30:37

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @TFM_Menu_1_OpeningFcn, ...
                   'gui_OutputFcn',  @TFM_Menu_1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before TFM_Menu_1 is made visible.
function TFM_Menu_1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
```

```
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to TFM_Menu_1 (see VARARGIN)

% Choose default command line output for TFM_Menu_1
handles.output = hObject;

% Inicializaciones
handles.VISUAL_1D   = 0;
handles.VISUAL_2D   = 0;
handles.VISUAL_3D   = 0;
handles.CLASIF_1D   = 0;
handles.CLASIF_2D   = 0;
handles.CLASIF_3D   = 0;
handles.CLASIF_4D   = 0;
handles.NroClases   = 5;
handles.PROCED      = 0;
handles.SELEC       = -1;
handles.NORMALIZ    = 1;
handles.CLASIF      = 0;
handles.CLASIF_xD   = [0,0,0,0];
% handles.Inic_foto   = 'A_';
% handles.Fin         = 'img.pgn';
handles.nroFotos    = 1;
handles.FraccionaEN = 0;
handles.RedimensionaA=0;
handles.ANG         = 4;
handles.MC          = [];
handles.MCn         = [];
handles.MuestrasAcum= [];
handles.NroNeuronas = 50;
% handles.carpeta_nubes=[];
% handles.carpeta_figuras=[];
handles.Seleccion   = [];
handles.minorac = 0.05;

disp('Inicialización OK')
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes TFM_Menu_1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = TFM_Menu_1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
% varargout{2} = handles.PROCED;


%% BOTÓN para la Evaluación Inicial
```

```
function Push_Gen_Clasif_Callback(hObject, eventdata, handles)

    %% Clasificador de mínima distancia

    handles.CLASIF_xD = [handles.CLASIF_1D, handles.CLASIF_2D,...
        handles.CLASIF_3D, handles.CLASIF_4D];
    [ handles.MinE ] = TFM_Clasifica( handles.PROCED, handles.SE-
LEC,...
        handles.NORMALIZ, handles.CLASIF, handles.CLASIF_xD,...
        handles.MCn1, handles.MuestrasAcum1 );

    guidata(hObject, handles);

%% MENÚ POPUP para seleccionar el tipo de imagen
function Menu_ConfInic_SELEC_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    handles.SELEC = -1;
    handles.FraccionaEN = 0;
    handles.RedimensionaA = 0;
    handles.NroClases = 5;
    guidata(hObject,handles);

function Menu_ConfInic_SELEC_Callback(hObject, eventdata, handles)

    val = get(hObject,'Value');
    handles.SELEC = val - 2;

    switch handles.SELEC
        case 1
            % Base de datos SWIMCAT - Selección
            A_inic='A_selec ('; A_nroFotos = 100;  % SKY
            B_inic='B_selec ('; B_nroFotos = 80;   % PATTERN
            C_inic='C_selec ('; C_nroFotos = 185;  % DENSE-DARK
            D_inic='D_selec ('; D_nroFotos = 115;  % DENSE-LIGHT
            E_inic='E_selec ('; E_nroFotos = 80;   % VEIL

            handles.carpeta_nubes = fullfile(handles.carpetaNubes,
'IMAG_SWIMCAT');
            handles.Fin = ').png';
            handles.FraccionaEN = 0;
            handles.RedimensionaA = 0;
            handles.Seleccion = 'Originales';
        case 2
            % Base de datos SWIMCAT - Selección
            A_inic='A_selec (0'; A_nroFotos = 100;  % SKY
            B_inic='B_selec (0'; B_nroFotos = 80;   % PATTERN
            C_inic='C_selec (0'; C_nroFotos = 185;  % DENSE-DARK
            D_inic='D_selec (0'; D_nroFotos = 115;  % DENSE-LIGHT
            E_inic='E_selec (0'; E_nroFotos = 80;   % VEIL

            handles.Fin = ').png';
            if handles.ESCALA == 1
                handles.FraccionaEN = 9;     % Número de partes en que
se dividirán las imágenes
```

```
            elseif handles.ESCALA == 2
                handles.FraccionaEN = 4;
            end
            handles.RedimensionaA = 0;
            handles.carpeta_nubes = fullfile(handles.carpetaNubes,...
                ['FraccS',num2str(handles.FraccionaEN)]);
            handles.Seleccion = ['Fraccion',num2str(handles.Fraccio-
naEN)];
        case 3
            % Base de datos SWIMCAT - Selección
            A_inic='A_selec ('; A_nroFotos = 100;  % SKY
            B_inic='B_selec ('; B_nroFotos = 80;   % PATTERN
            C_inic='C_selec ('; C_nroFotos = 185;  % DENSE-DARK
            D_inic='D_selec ('; D_nroFotos = 115;  % DENSE-LIGHT
            E_inic='E_selec ('; E_nroFotos = 80;   % VEIL

            if handles.ESCALA == 1
                handles.RedimensionaA = 0.3;    % Fracción de redimen-
sionado de las imágenes
            elseif handles.ESCALA == 2
                handles.RedimensionaA = 0.5;    % Fracción de redimen-
sionado de las imágenes
            end
            handles.Fin = [num2str(handles.RedimensionaA),').png'];
            handles.carpeta_nubes = fullfile(handles.carpetaNubes,...
                ['RedimS',num2str(handles.RedimensionaA)]);
            handles.FraccionaEN = 0;
            handles.Seleccion = ['Redimens',num2str(handles.Redimen-
sionaA)];
        case 4
            A_inic='A_'; A_nroFotos = 120;  % SKY
            B_inic='B_'; B_nroFotos = 80;   % PATTERN
            C_inic='C_'; C_nroFotos = 120;  % DENSE-DARK
            D_inic='D_'; D_nroFotos = 120;  % DENSE-LIGHT
            E_inic='E_'; E_nroFotos = 80;   % VEIL

            handles.carpeta_nubes = fullfile(handles.carpetaNubes,
'Im_Segment');
            handles.Fin = '_segimg.png';
            handles.FraccionaEN = 0;
            handles.RedimensionaA = 0;
            handles.Seleccion = 'Muestras';
        case 5
            % Base de datos SWIMCAT - Selección
            A_inic='A_selec ('; A_nroFotos = 100;  % SKY
            B_inic='B_selec ('; B_nroFotos = 80;   % PATTERN
            C_inic='C_selec ('; C_nroFotos = 185;  % DENSE-DARK
            D_inic='D_selec ('; D_nroFotos = 115;  % DENSE-LIGHT
            E_inic='E_selec ('; E_nroFotos = 80;   % VEIL

            handles.carpeta_nubes = fullfile(handles.carpetaNubes,
'Laws');
            handles.Fin = ').png';
            handles.FraccionaEN = 0;
            handles.RedimensionaA = 0;
            handles.Seleccion = 'MapaLaws';
        case 6
            A_inic='A_'; A_nroFotos = 224;  % SKY
            B_inic='B_'; B_nroFotos = 89;   % PATTERN
            C_inic='C_'; C_nroFotos = 251;  % DENSE-DARK
```

```
            D_inic='D_'; D_nroFotos = 135;   % DENSE-LIGHT
            E_inic='E_'; E_nroFotos = 85;    % VEIL

            handles.carpeta_nubes = fullfile(handles.carpetaNubes,
'FiltroContraste');
            handles.Fin = '_FiltroContrasteimg.png';
            handles.FraccionaEN = 0;
            handles.RedimensionaA = 0;
            handles.Seleccion = 'Filtradas';
        case 7
            A_inic='Gabor39_A_selec ('; A_nroFotos = 100;   % SKY
            B_inic='Gabor39_B_selec ('; B_nroFotos = 80;    % PATTERN
            C_inic='Gabor39_C_selec ('; C_nroFotos = 185;   % DENSE-
DARK
            D_inic='Gabor39_D_selec ('; D_nroFotos = 115;   % DENSE-
LIGHT
            E_inic='Gabor39_E_selec ('; E_nroFotos = 80;    % VEIL

            handles.carpeta_nubes = fullfile(handles.carpetaNubes,
'Gabor');
            handles.Fin = ').png';
            handles.FraccionaEN = 0;
            handles.RedimensionaA = 0;
            handles.Seleccion = 'Gabor';
        case 8
            A_inic='test_structureA_'; A_nroFotos = 62;   % SKY
            B_inic='test_structureB_'; B_nroFotos = 62;    % PATTERN
            C_inic='test_structureC_'; C_nroFotos = 62;   % DENSE-DARK
            D_inic='test_structureD_'; D_nroFotos = 62;   % DENSE-LIGHT
            E_inic='test_structureE_'; E_nroFotos = 62;    % VEIL

            handles.carpeta_nubes = fullfile(handles.carpetaNubes,
'Tensores');
            handles.Fin = 'img.png';
            handles.FraccionaEN = 0;
            handles.RedimensionaA = 0;
            handles.Seleccion = 'Tensores';
    end

    handles.Inic_foto = {A_inic, B_inic, C_inic, D_inic, E_inic};
    handles.nroFotos = [ A_nroFotos, B_nroFotos, C_nroFotos, D_nroFo-
tos, E_nroFotos ];

    guidata(hObject,handles);

%% MENÚ POPUP para seleccionar el PROCEDIMIENTO
function Menu_ConfInic_PROC_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    handles.PROCED = -1;
    handles.ANG = 1;
    guidata(hObject,handles);

function Menu_ConfInic_PROC_Callback(hObject, eventdata, handles)
```

```
    val = get(hObject,'Value');
    handles.PROCED = val - 2;
    if handles.PROCED == 1
        if handles.SELEC ~= 4
            handles.SELEC = 4;
            warning(['Las imágenes seleccionadas no están optimizadas
para un ',...
                'análisis cromático. Se tomarán las muestras prepara-
das.']);
        end
    end

    switch handles.PROCED
        case 1
            handles.NroParam = 16;  % Número de parámetros cromáticos
            handles.Procedim = 'Cromatico';

            Params ={'Plano R',...  %1
            'Plano G',...           %2
            'Plano B',...           %3
            'Plano H',...           %4
            'Plano S',...           %5
            'Plano V',...           %6
            'Plano Y',...           %7
            'Plano I',...           %8
            'Plano Q',...           %9
            'Plano L',...           %10
            'Plano a*',...          %11
            'Plano b*', ...         %12
            'Plano R/B',...         %13
            'Plano R-B',...         %14
            'Plano (R-B)/(R+B)',... %15
            'Chroma'};              %16

            i=0;
            handles.TitleParam = {};
            for j1 = 1:handles.NroParam
                for j2 = j1+1:handles.NroParam
                    for j3 = j2+1:handles.NroParam
                        i = i+1;
                        handles.TitleParam{i} = ['Distancia ',Pa-
rams{j1},' + ',...
                            Params{j2}, ' + ', Params{j3}];
                    end
                end
            end
            clear Params
        case 2
            handles.NroParam = 22;  % Número de parámetros texturales
por Haralick
            handles.ANG = 4;        % Número de ángulos de parámetros
GLCM
            handles.Procedim = 'Haralick';

            handles.TitleParam ={'Contraste',...        %1
            'Correlación MATLAB',...    %2
            'Energía',...               %3
            'Homogeneidad MATLAB',...   %4
            'Autocorrelación',...       %5
```

```
            'Correlación [1,2]',...     %6
            'Cluster Prominence',...    %7
            'Cluster Shade',...         %8
            'Disimilaridad',...         %9
            'Entropía',...              %10
            'Homogeneidad',...                          %11
            'Probabilidad Máxima', ...                  %12
            'Suma de cuadrados: Varianza',...           %13
            'Media de la suma',...                      %14
            'Varianza de la suma',....                  %15
            'Entropía de la suma',...                   %16
            'Varianza de la diferencia',...             %17
            'Entropía de la diferencia',...             %18
            'Information measure of correlation1',...   %19
            'Information measure of correlation2',...   %20
            'Inverse difference normalized (INN)',...   %21
            'Inverse difference moment normalized'};    %22
        case 3
            handles.NroParam = 9;  % Número de parámetros de Law's
            handles.Procedim = 'Laws';

            handles.TitleParam ={'L5E5/E5L5',...    %1
            'L5R5/R5L5',...             %2
            'E5S5/S5E5',...             %3
            'S5S5',...                  %4
            'R5R5',...                  %5
            'L5S5/S5L5',...             %6
            'E5E5',...                  %7
            'E5R5/R5E5',...             %8
            'S5R5/R5S5'};               %9
    end

    guidata(hObject,handles);


%% MENÚ POPUP para seleccionar el método de NORMALIZACIÓN
function Menu_ConfInic_NORMALIZ_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    handles.NORMALIZ = -1;
    guidata(hObject,handles);

function Menu_ConfInic_NORMALIZ_Callback(hObject, eventdata, handles)

    val = get(hObject,'Value');
    handles.NORMALIZ = val - 2;
    guidata(hObject,handles);

%% MENÚ POPUP para seleccionar el método de CLASIFICACIÓN

function Menu_ConfInic_CLASIF_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
```

```
    end
    handles.CLASIF = -1;
    guidata(hObject,handles);

function Menu_ConfInic_CLASIF_Callback(hObject, eventdata, handles)

    val = get(hObject,'Value');
    handles.CLASIF = val - 2;
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Menu_Procesam_TYPE_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    handles.TYPE = -1;
    guidata(hObject,handles);

% --- Executes on selection change in Menu_Procesam_TYPE.
function Menu_Procesam_TYPE_Callback(hObject, eventdata, handles)

    val = get(hObject,'Value');
    handles.TYPE = val - 2;
    switch handles.TYPE
        case 1
            switch handles.ESCALA
                case -1
                    aux = 00;
                case 1
                    aux = 9;
                case 2
                    aux = 4;
            end
            handles.carpeta_nubes2 = fullfile(handles.carpetaNubes,...
                ['FraccS',num2str(aux)]);
        case 2
            switch handles.ESCALA
                case -1
                    aux = 00;
                case 1
                    aux = 0.3;
                case 2
                    aux = 0.5;
            end
            handles.carpeta_nubes2 = fullfile(handles.carpetaNubes,...
                ['RedimS',num2str(aux)]);
    end

    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Menu_Procesam_ESCALA_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
```

```
    end
    handles.ESCALA = -1;
    guidata(hObject,handles);

% --- Executes on selection change in Menu_Procesam_ESCALA.
function Menu_Procesam_ESCALA_Callback(hObject, eventdata, handles)

    val = get(hObject,'Value');
    handles.ESCALA = val - 2;
    guidata(hObject,handles);

function check_1D_Callback(hObject, eventdata, handles)

    handles.VISUAL_1D = get(hObject,'Value');
    guidata(hObject,handles);

function check_2D_Callback(hObject, eventdata, handles)

    handles.VISUAL_2D = get(hObject,'Value');
    guidata(hObject,handles);

function check_3D_Callback(hObject, eventdata, handles)

    handles.VISUAL_3D = get(hObject,'Value');
    guidata(hObject,handles);

% --- Executes on button press in check_Carac_1.
function check_Carac_1_Callback(hObject, eventdata, handles)

    handles.CLASIF_1D = get(hObject,'Value');
    guidata(hObject,handles);

% --- Executes on button press in check_Carac_2.
function check_Carac_2_Callback(hObject, eventdata, handles)

    handles.CLASIF_2D = get(hObject,'Value');
    guidata(hObject,handles);

% --- Executes on button press in check_Carac_3.
function check_Carac_3_Callback(hObject, eventdata, handles)

    handles.CLASIF_3D = get(hObject,'Value');
    guidata(hObject,handles);

% --- Executes on button press in check_Carac_4.
function check_Carac_4_Callback(hObject, eventdata, handles)

    handles.CLASIF_4D = get(hObject,'Value');
    guidata(hObject,handles);

function Text_CarpetaNubes_Callback(hObject, eventdata, handles)

    handles.carpeta_nubes = get(hObject,'String');
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
```

```
function Text_CarpetaNubes_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
    handles.carpetaNubes  = str(1:length(str)-1);
    handles.carpeta_nubes = handles.carpetaNubes;
    guidata(hObject,handles);

function Text_CarpetaFiguras_Callback(hObject, eventdata, handles)

    handles.carpetaFiguras = get(hObject,'String');
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Text_CarpetaFiguras_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
    handles.carpetaFiguras = str(1:length(str));
    guidata(hObject,handles);

% --- Executes on button press in check_segment_Crom.
function check_segment_Crom_Callback(hObject, eventdata, handles)
    handles.SegmCrom = get(hObject,'Value');
    guidata(hObject,handles);

% --- Executes on button press in check_segment_Txtl.
function check_segment_Txtl_Callback(hObject, eventdata, handles)
    handles.SegmTxtl = get(hObject,'Value');
    guidata(hObject,handles);

% --- Executes on button press in check_segment_Mix.
function check_segment_Mix_Callback(hObject, eventdata, handles)
    handles.SegmMix = get(hObject,'Value');
    guidata(hObject,handles);

% --- Executes on button press in check_segment_MinDist.
function check_segment_MinDist_Callback(hObject, eventdata, handles)
    handles.SegmMinDist = get(hObject,'Value');
    guidata(hObject,handles);

% --- Executes on button press in check_segment_Kmeans.
function check_segment_Kmeans_Callback(hObject, eventdata, handles)
    handles.SegmKmeans = get(hObject,'Value');
    guidata(hObject,handles);

% --- Executes on button press in check_segment_Neural.
function check_segment_Neural_Callback(hObject, eventdata, handles)
    handles.SegmNeural = get(hObject,'Value');
    guidata(hObject,handles);

function Text_CarCrom_2_Callback(hObject, eventdata, handles)
```

```
    handles.CarCrom2 = get(hObject,'String');
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Text_CarCrom_2_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
    handles.CarCrom2 = str2double(str(1:length(str)));
    guidata(hObject,handles);

function Text_CarCrom_1_Callback(hObject, eventdata, handles)
    handles.CarCrom1 = get(hObject,'String');
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Text_CarCrom_1_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
    handles.CarCrom1 = str2double(str(1:length(str)));
    guidata(hObject,handles);

function Text_CarTxtl_1_Callback(hObject, eventdata, handles)
    handles.CarTxtl1 = get(hObject,'String');
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Text_CarTxtl_1_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
    handles.CarTxtl1 = str2double(str(1:length(str)));
    guidata(hObject,handles);

function Text_CarTxtl_2_Callback(hObject, eventdata, handles)
    handles.CarTxtl2 = get(hObject,'String');
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Text_CarTxtl_2_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
    handles.CarTxtl2 = str2double(str(1:length(str)));
    guidata(hObject,handles);
```

```
function Text_CarTxtl_3_Callback(hObject, eventdata, handles)
    handles.CarTxtl3 = get(hObject,'String');
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Text_CarTxtl_3_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
    handles.CarTxtl3 = str2double(str(1:length(str)));
    guidata(hObject,handles);

function Text_CarTxtl_4_Callback(hObject, eventdata, handles)
    handles.CarTxtl4 = get(hObject,'String');
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Text_CarTxtl_4_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
    handles.CarTxtl4 = str2double(str(1:length(str)));
    guidata(hObject,handles);

function Text_CarMix_Crom_2_Callback(hObject, eventdata, handles)
    handles.CarMix_Crom2 = get(hObject,'String');
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Text_CarMix_Crom_2_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
    handles.CarMix_Crom2 = str2double(str(1:length(str)));
    guidata(hObject,handles);

function Text_CarMix_Crom_1_Callback(hObject, eventdata, handles)
    handles.CarMix_Crom1 = get(hObject,'String');
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Text_CarMix_Crom_1_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
```

```
    handles.CarMix_Crom1 = str2double(str(1:length(str)));
    guidata(hObject,handles);

function Text_CarMix_Txtl_1_Callback(hObject, eventdata, handles)
    handles.CarMix_Txtl1 = get(hObject,'String');
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Text_CarMix_Txtl_1_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
    handles.CarMix_Txtl1 = str2double(str(1:length(str)));
    guidata(hObject,handles);

function Text_CarMix_Txtl_2_Callback(hObject, eventdata, handles)
    handles.CarMix_Txtl2 = get(hObject,'String');
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Text_CarMix_Txtl_2_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
    handles.CarMix_Txtl2 = str2double(str(1:length(str)));
    guidata(hObject,handles);

function Text_CarMix_Txtl_3_Callback(hObject, eventdata, handles)
    handles.CarMix_Txtl3 = get(hObject,'String');
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Text_CarMix_Txtl_3_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
    handles.CarMix_Txtl3 = str2double(str(1:length(str)));
    guidata(hObject,handles);

function Text_CarMix_Txtl_4_Callback(hObject, eventdata, handles)
    handles.CarMix_Txtl4 = get(hObject,'String');
    guidata(hObject,handles);


% --- Executes during object creation, after setting all properties.
function Text_CarMix_Txtl_4_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
```

```
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
    handles.CarMix_Txtl4 = str2double(str(1:length(str)));
    guidata(hObject,handles);

function Text_FotoInic_Callback(hObject, eventdata, handles)
    handles.Inic_foto = get(hObject,'String');
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Text_FotoInic_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
    handles.Inic_foto = str(1:length(str));
    guidata(hObject,handles);

function Text_FotoFin_Callback(hObject, eventdata, handles)
    handles.Fin = get(hObject,'String');
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Text_FotoFin_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
    handles.Fin = str(1:length(str));
    guidata(hObject,handles);


function Text_Imag2segmentar_Callback(hObject, eventdata, handles)
    handles.Imag2segmentar = get(hObject,'String');
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Text_Imag2segmentar_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
    handles.Imag2segmentar = str(1:length(str));
    guidata(hObject,handles);

function Text_Imag2segmFin_Callback(hObject, eventdata, handles)
    handles.Imag2segmFin = get(hObject,'String');
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Text_Imag2segmFin_CreateFcn(hObject, eventdata, handles)
```

```
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
    handles.Imag2segmFin = str(1:length(str));
    guidata(hObject,handles);


function Text_NroFotos_Callback(hObject, eventdata, handles)
    handles.NroFotos = get(hObject,'String');
    guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function Text_NroFotos_CreateFcn(hObject, eventdata, handles)

    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    str = get(hObject,'String');
    handles.NroFotos = str2double(str(1:length(str)));
    guidata(hObject,handles);

% --- Executes on button press in Push_show_results.
function Push_show_results_Callback(hObject, eventdata, handles)

    TFM_plot3( handles.VISUAL_1D, handles.VISUAL_2D, handles.VI-
SUAL_3D,...
        handles.MuestrasAcum, handles.carpetaFiguras, handles.Proce-
dim, ...
        handles.Seleccion, handles.TitleParam, handles.MCn );

% --- Executes on button press in Push_Gen_Matrices.
function Push_Gen_Matrices_Callback(hObject, eventdata, handles)
    % Extracción de características cromáticas o texturales
    [ handles.MC, handles.nroMuestras, handles.MuestrasAcum ] = ...
        TFM_CalculoCaract( handles.PROCED, handles.SELEC, handles.Nro-
Clases,...
        handles.Inic_foto, handles.Fin, handles.nroFotos, hand-
les.FraccionaEN,...
        handles.RedimensionaA, handles.ANG);

    % Normalización
    [ handles.MCn, handles.MAXe ] = TFM_Normaliz( handles.MC,...
        handles.PROCED, handles.NORMALIZ, []  );

    % Reducción del número de muestras para el procedimiento Cromático
    if handles.PROCED == 1
        minorac = 0.05;
    else
        minorac = 1;
    end
    [handles.MCn1, handles.MuestrasAcum1] = TFM_Minorac( handles.PRO-
CED,...
        handles.MCn, handles.MuestrasAcum, minorac );
```

```
    guidata(hObject,handles);

% --- Executes on button press in Push_Save_Matrices.
function Push_Save_Matrices_Callback(hObject, eventdata, handles)

    file = ['MCn_P',num2str(handles.PROCED),'S',num2str(handles.SE-
LEC),...
        'N',num2str(handles.NORMALIZ),'_GUI'];
    MCn  = handles.MCn;
    MCn1 = handles.MCn1;
    MAXe = handles.MAXe;
    MuestrasAcum  = handles.MuestrasAcum;
    MuestrasAcum1 = handles.MuestrasAcum1;
    save(file,'MCn','MCn1','MAXe','MuestrasAcum','MuestrasAcum1');
    disp(['Matrices MCn, MCn1, MAXe, MuestrasAcum y MuestrasAcum1',...
        ' guardadas en el archivo ', file,'.MAT'])
    guidata(hObject,handles);

% --- Executes on button press in Push_Load_Matrices.
function Push_Load_Matrices_Callback(hObject, eventdata, handles)

    file = ['MCn_P',num2str(handles.PROCED),'S',num2str(handles.SE-
LEC),...
        'N',num2str(handles.NORMALIZ),'_GUI'];
    load(file);
    handles.MCn = MCn;
    handles.MCn1 = MCn1;
    handles.MAXe = MAXe;
    handles.MuestrasAcum = MuestrasAcum;
    handles.MuestrasAcum1 = MuestrasAcum1;
    disp(['Archivo ', file,'.MAT cargado en el espacio de trabajo'])
    guidata(hObject,handles);

% --- Executes on button press in Push_Gen_Protot.
function Push_Gen_Protot_Callback(hObject, eventdata, handles)

    Cxx = [handles.SegmMinDist, handles.SegmKmeans, handles.SegmNeu-
ral];
    if handles.SegmCrom == 1
        Carac = {[ handles.CarCrom1, handles.CarCrom2 ]};
        TFM_Segm_Protot_Crom( Carac, Cxx );
        disp('Prototipo/s cromático/s generado/s')
    end
    if handles.SegmTxtl == 1
        CaracT_MD = {[ handles.CarTxtl1, handles.CarTxtl2,...
            handles.CarTxtl3, handles.CarTxtl4 ]};
        TFM_Segm_Protot_Txtl( CaracT_MD, Cxx );
        disp('Prototipo/s textural/es generado/s')
    end
    if handles.SegmMix == 1
        Carac_P1y2 = {[ handles.CarTxtl1, handles.CarTxtl2,...
            handles.CarTxtl3, handles.CarTxtl4,...
            handles.CarCrom1, handles.CarCrom2 ]};
        TFM_Segm_Protot_Mixta( 100, Carac_P1y2, Cxx );
        disp('Prototipo/s mixto/s generado/s')
    end

    % --- Executes on button press in Push_segmentar.
function Push_segmentar_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to Push_segmentar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    if handles.SegmCrom == 1
        TFM_Segm_SegmInd_Crom( 1 );
    end
    if handles.SegmTxtl == 1
        TFM_Segm_SegmInd_Txtl( 1 );
    end
    if handles.SegmMix == 1
        TFM_Segm_SegmInd_Mixta( 1, 2 );
    end

% --- Executes on button press in Push_Gen_Imagenes.
function Push_Gen_Imagenes_Callback(hObject, eventdata, handles)

    switch handles.TYPE
        case 1
            for i=1:5
                TFM_Fracciona(handles.Inic_foto{i}, handles.Fin, ...
                    handles.nroFotos(i), handles.FraccionaEN, hand-
les.carpeta_nubes);
            end
        case 2
            for i=1:5
                TFM_Redim(handles.Inic_foto{i}, handles.Fin, ...
                    handles.nroFotos(i), handles.RedimensionaA, hand-
les.carpeta_nubes);
            end
    end
```