Universidad Carlos III de Madrid

Escuela Politécnica Superior

Anotación Automática de Imágenes

Proyecto Fin de Carrera

Ingeniería Técnica de Telecomunicación Especialidad en Imagen y Sonido

Autor: Juan Rodríguez Povedano
Tutor: Julio Villena Román
Madrid, Octubre 2010

Proyecto: Anotación Automática de Imágenes

Autor: Juan Rodríguez Povedano

Tutor: Julio Villena Román

TRIBUNAL

Presidente: Manuel Urueña Pascual

Secretario: Mª Carmen Fernández Panadero

Vocal: Sara Pino Povedano

Realizado el acto de defensa del Proyecto Fin de Carrera el día 15 de Octubre de 2010 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

Fdo. Presidente Fdo. Secretario Fdo. Vocal

Nuestra recompensa se encuentra en el esfuerzo y no en el resultado. Un esfuerzo total es una victoria completa.

Mahatma Gandhi

Lo que parecía imposible ayer quizás sea posible hoy. Y lo que hoy parece imposible será posible mañana. ¡Nada es imposible!

Anónimo

Agradecimientos

La presentación de este proyecto me supone una realización personal, pero es consecuencia tanto de mi esfuerzo e interés como del apoyo que he recibido por parte de muchas personas allegadas. Por ello quiero mostrarles mi agradecimiento a todos aquellos que me han ayudado, me han soportado en algunas ocasiones o han estado a mi lado durante esta etapa que ahora se acaba.

En especial, a mi novia Jennifer por la ayuda, paciencia y compresión que ha tenido conmigo siempre y en concreto durante toda mi etapa como universitario.

A mis padres y mi hermana por su apoyo y su incombustible paciencia, sobre todo a la hora de la realización de este proyecto.

A todos mis amigos de El Hoyo de Pinares, los cuales siempre me han apoyado, me han prestado la ayuda necesaria e incluso se han interesado sobre el tema y la evolución que llevaba la ejecución de este proyecto.

A mi tutor Julio Villena, por la gran facilidad que siempre me ha mostrado y por supuesto por su inestimable ayuda.

Y por último, a mis profesores durante toda mi etapa como universitario, los cuales me han proporcionado los conocimientos necesarios para la realización de este proyecto y fundamentalmente para poder establecerme en la vida laboral.

A todos, ¡GRACIAS!

Índice:

LISTADO DE FIGURAS

LISTADO DE TABLAS

LISTADO DE ECUACIONES

PARTE I:

INTRODUCCIÓN

1. Introducción

1.1 Motivación y marco del proyecto

El presente trabajo tiene por objetivo presentar una metodología adecuada para el reconocimiento de determinados conceptos visuales en imágenes digitales, utilizando para ello diversas técnicas de Visión Artificial.

El progreso tecnológico y el consecuente avance de los servicios multimedia han incrementado de forma exponencial la cantidad de información digital. El gran volumen existente de este tipo de información ha provocado que sean necesarios sistemas que de modo eficiente y sencillo permitan adquirir, gestionar, buscar y recuperar la información. Los encargados de esto son los sistemas de *Recuperación de información* (IR, *Information Retrieval*). Desde un punto de vista práctico, dada una necesidad de información por parte del usuario, un proceso de IR produce como salida un conjunto de objetos cuyo contenido satisface dicha necesidad. Este último punto es importante, pues la función de un sistema IR no es devolver la información deseada por el usuario, sino la de indicar qué objetos son relevantes para dicha necesidad de información. El sistema más popular de recuperación de información es el de los motores de búsqueda en Internet, como por ejemplo *Google* o *Yahoo*.

La información que el usuario requiere puede ser de diversos tipos digitales, tales como texto, imágenes, audio y vídeo, entre muchos otros. De todos estos tipos de información, en este proyecto se trabajará con un sistema de recuperación de información en imágenes.

Los **Sistemas de Recuperación de Información en Imágenes** utilizan para su propósito técnicas de Visión Artificial. La Visión Artificial también conocida como Visión por Computadora, es un sub-campo de la Inteligencia Artificial. Su propósito principal es la de realizar un programa que sea capaz de "entender" una escena o las características de una imagen.

Las técnicas de Visión por Computadora han madurado a gran velocidad en los últimos veinte años. Los primeros sistemas prácticos se basan en imágenes binarias (blanco y negro) que se procesaban por bloques, ventanas o píxels. Rápidamente los algoritmos alcanzaron un nivel de desarrollo en el cual era posible reconocer el contorno de un objeto y su posición en la imagen.

El siguiente paso en el desarrollo fue la introducción de los sistemas de niveles de gris. Con esta tecnología cada elemento de la imagen, o píxel, se representa con un número proporcional a la intensidad de gris del píxel. La principal ventaja de esta técnica es que se pueden corregir las variaciones locales de iluminación.

Los sistemas de Visión Artificial de vanguardia operan sobre estructuras en lugar de píxels y requieren de poderosos procesadores de imágenes para procesar la gran cantidad de datos recibidos debido al aumento de la calidad digital de las imágenes.

Los sistemas de Visión por Computadora emplean técnicas de reconocimiento de imágenes para interpretar las propiedades de la imagen, compararlas con la base de datos e identificarla como objeto reconocido. Estas técnicas y otras del procesamiento de imágenes se han ampliado y extendido de forma exponencial debido a la gran cantidad de aplicaciones actuales que las solicitan. Algunos ejemplos prácticos de aplicaciones que las utilizan son: el reconocimiento de caracteres, diagnósticos médicos, tecnología del espacio, reconocimiento de huellas digitales, análisis geográficos y satelitales, además de muchas otras, incluidas el entretenimiento casero, álbumes fotográficos digitales o aplicaciones para profesionales del arte y el diseño.

La recuperación de información en las imágenes se divide en dos ramas: recuperación de imágenes basadas en texto (*text-based*) y basadas en contenido (*content-based*).

Los sistemas basados en texto, describen en documentos de texto el contenido de las imágenes, y aplican los conceptos procedentes de técnicas de recuperación de texto para recuperar el contenido de la imagen. Estos sistemas se han estancado debido a que no son precisos, pues la/s palabra/s claves que se utilicen para la búsqueda no serán suficientes para describir la variedad de conceptos de una imagen.

Los sistemas basados en contenido son los que más han evolucionado, pues resuelven en cierta forma los problemas de los anteriores. Estas técnicas se basan que mediante la extracción automática de características de bajo nivel a las imágenes (como pueden ser colores, formas, texturas, etc.) sea posible describir y reconocer el contenido de las mismas. La ventaja que proporcionan es la sencillez en la extracción de características, pues son operaciones básicas sobre los píxels. Su principal inconveniente es que las descripciones carecen de semántica y no se pueden realizar consultas a niveles superiores.

1.2 Objetivos

Este proyecto sigue la línea de la recuperación de información en imágenes mediante sistemas basados en contenido. En concreto se basa en la tarea *Visual Concept Detection Task (VCDT)* que se define en ImageCLEF 2008 [35]. Se pretende programar un sistema capaz de clasificar imágenes procedentes de una determinada base de datos mediante la aparición o ausencia de unos determinados conceptos en ellas.

En concreto, el objetivo es el de desarrollar un sistema de reconocimiento de imágenes que para una base de datos dada sea capaz de determinar si en las imágenes aparecen o no los siguientes conceptos:

- Interior o exterior, personas, día o noche, agua, caminos, vegetación, árboles, montañas, edificios, playa, animales, cielo y tipo de cielo (soleado, parcialmente nublado o nublado).

La base de datos que se proporciona contiene un total de 2803 fotografías a color de temática general dividas en dos grupos. El primero de ellos contiene 1809 imágenes que se utilizarán para realizar el entrenamiento del sistema y el segundo consta de 994 imágenes que serán utilizadas para la realización del test y la posterior evaluación del sistema desarrollado. En el capítulo 6.2 se amplía la información aquí detallada sobre el objetivo de la tarea VCDT propuesta por ImageCLEF 2008.

A parte del objetivo principal se ha marcado en este proyecto un objetivo secundario de forma particular. Como más adelante se detalla, todo sistema de reconocimiento de imágenes consta de una etapa, denominada *Extracción de características*, en la cual se extraen una serie de características o parámetros a las imágenes que a posteriori servirán para determinar la aparición o ausencia de los conceptos en ellas. La elección de estos parámetros proporcionará en cierta medida la eficacia del sistema en la detección de los conceptos. De esta forma el segundo objetivo marcado consiste en determinar cuál de los siguientes métodos de extracción de características es más o menos eficaz:

- Extracción de características general, extrae una serie de parámetros a las imágenes de forma general y a partir de ellas determina la aparición/ausencia de todos los conceptos en ellas.

- Extracción de características específica, para cada concepto se extrae un conjunto de parámetros diferente que permita determinar la aparición/ausencia de dicho concepto en las imágenes.

- Extracción de características combinada, combina los dos métodos anteriores, utilizando tanto características generales como específicas para determinar la aparición/ausencia de los conceptos.

Para el desarrollo de ambos objetivos se utilizan técnicas básicas del procesamiento de imágenes que se detallarán en capítulos posteriores.

1.3 Estructura del proyecto

El contenido de la memoria de este proyecto se encuentra dividido en las siguientes partes:

- **Parte I: Introducción**
  Motivación y objetivos del proyecto, y descripción de la organización de la memoria.

- **Parte II: Estado del Arte**
  Introducción al análisis de imágenes, descripción de las etapas del reconocimiento de imágenes y los algoritmos que se emplean en ellas, definición de la tarea *Visual Concept Detection Task* (*VCDT*) de ImageClef 2008 y exposición de los diferentes métodos empleados por los participantes en dicha tarea.

- **Parte III: Implementación**
  Arquitectura del sistema, técnicas de preprocesado, extracción de características y clasificación empleadas para la solución de la tarea *VCDT*.

- **Parte IV: Evaluación**
  Presentación de los resultados y evaluación de los diversos métodos empleados.

- **Parte V: Presupuesto**
  Presentación desglosada del presupuesto del proyecto.

- **Parte VI: Conclusiones y Líneas Futuras**
  Estudio de los objetivos propuestos y los resultados obtenidos. Futuras mejoras de las técnicas empleadas.

- **Parte VII: Anexo**
  Código de las funciones implementadas en Matlab para la elaboración de la tarea propuesta.

- **Bibliografía**
  Referencias bibliográficas utilizadas en este documento.

PARTE II:

ESTADO DEL ARTE

2. Introducción al análisis de imágenes

El problema del reconocimiento de objetos consiste en buscar objetos previamente conocidos en una imagen. Los seres humanos realizan esta tarea de una forma instantánea y sin esfuerzo, sin embargo, la realización por ordenador de esta tarea es muy compleja y costosa. A continuación se presentan y describen diversas técnicas que se aplican en diferentes situaciones para solucionar este problema.

Formalmente se define un sistema de reconocimiento de objetos como [9] "dada una imagen que puede contener uno o varios objetos de interés, y dado un conjunto de modelos conocidos por el sistema, el sistema ha de ser capaz de asignar a cada región la etiqueta correcta". Es decir, se puede definir como un problema de etiquetado de objetos basado en modelos conocidos.

2.1 Elementos de un sistema de análisis de imágenes

Un sistema de reconocimiento de objetos se divide en tres áreas básicas:

- Procesado de bajo nivel
- Procesado de nivel intermedio
- Procesado de alto nivel

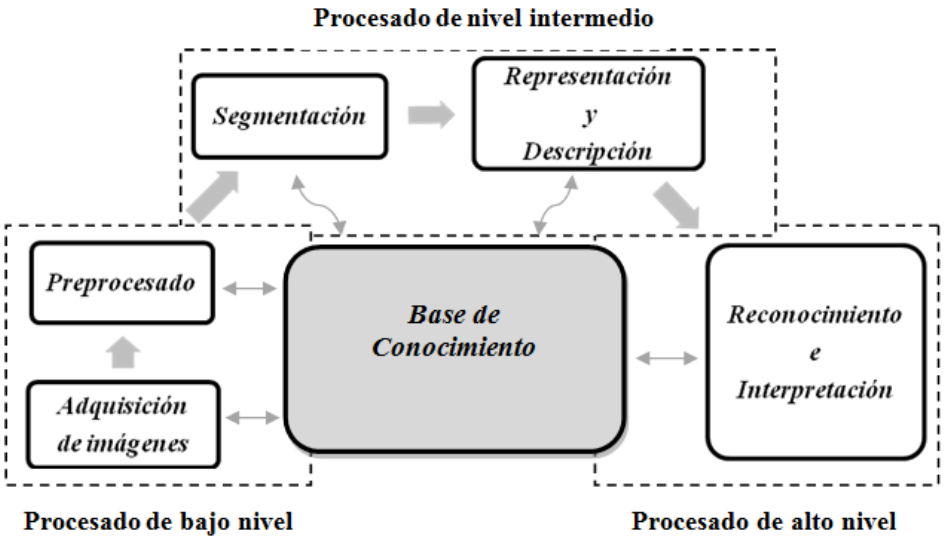

Figura 2.1 – Elementos de un sistema de análisis de imágenes

En la figura anterior se observa que cada área interactua con la "base de conocimiento". Es aquí donde se tienen los modelos conocidos y las características que los representan, ambos son la parte fundamental del reconocimiento. Por esto, para conseguir un buen sistema habrá que disponer de unos buenos modelos y, sobre todo, de unas carácterísticas que identifiquen unívocamente a cada objeto.

Las tres áreas se encuentran marcadas con líneas discontinuas y se solapan. Esto se debe a que no existen límites definidos entre los procesos, ya que existen técnicas que pueden ser usadas en varios procesos o realizan varios procesos a la vez.

El procesado de bajo nivel son técnicas que se aplican a la imagen para restaurar, realzar o mejorar ciertas características que interesarán posteriormente, además de la adquisición de las mismas. Estas técnicas no requieren inteligencia por parte del sistema por lo que se considera que operan a bajo nivel.

El procesado de nivel intermedio incluye la segmentación y la descripción. Es aquí donde se realiza una extracción de características de las regiones, las cuales servirán para poder describir e identificar al objeto.

El procesado de alto nivel comprende el reconocimiento y la interpretación. Estas técnicas no son tan precisas como las anteriores, sino que requieren de algoritmos de decisión. Para poder realizar este procesado y obtener un sistema eficaz es necesario haber realizado el procesado de bajo e intermedio nivel.

2.2 Componentes del sistema

La entrada a un sistema de reconocimiento es una imagen digital junto con los modelos conocidos, y la salida es una etiqueta que clasifica al objeto. Un sistema de reconocimiento de objetos debe tener los siguientes componentes [9] para poder realizar la tarea:

- Base de modelos o base de conocimiento
- Extractor de características
- Creador de hipótesis
- Verificador de hipótesis

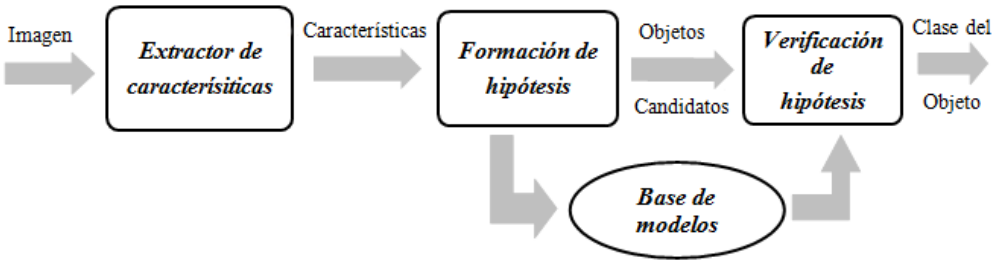

Figura 2.2 – Componentes de un sistema de reconocimiento de objetos

La base de modelos contiene todos los modelos conocidos por el sistema. La información existente depende de la precisión que se pretenda alcanzar con el sistema. Normalmente los modelos son vectores de características que describen e identifican al modelo y lo distinguen del resto.

El extractor de características aplica ciertas operaciones a las imágenes para obtener características que intenten identificar unívocamente cada objeto. Las características usadas por el sistema dependen fundamentalmente del objeto a reconocer, según el tipo de objeto serán útiles unas u otras características, como la forma, el color, el tamaño, etc.

El creador de hipótesis es un paso intermedio que es recomendable implementar (sobre todo en sistemas con un elevado número de objetos reconocibles) y sirve para realizar un descarte previo. Es decir, asigna probabilidades a los posibles objetos en función del valor de las características que se han detectado, reduciendo así el área de búsqueda. De esta forma el coste computacional se reduce (sobre todo cuando el número de objetos es alto).

La verificación de hipótesis compara con los posibles modelos y establece probabilidades de que sea el correcto. Finalmente el sistema seleccionará el objeto del modelo con la probabilidad más alta como el objeto correcto.

2.3 Complejidad

Las imágenes pueden ser muy diferentes en función de la iluminación con la que se tomen, la posición de la cámara, las características de ésta… Por esto la complejidad de un sistema de reconocimiento de objetos es muy alta, ya que un mismo objeto puede estar representado de diversas formas en imágenes diferentes. Los factores más comunes que influyen son:

- ***Punto de vista:*** Los objetos pueden ser observados con diferentes ángulos de visión y a diferentes distancias, estas vistas llevan a la representación de un mismo objeto en imágenes muy diferentes. Por esto se utilizan características de los objetos que sean independientes del punto de vista (lo que es muy complicado) o utilizar modelos que incluyan diferentes vistas del mismo objeto.

- ***Iluminación:*** Según el tipo de iluminación que tenga la escena, tanto en su dirección (iluminación frontal, difusa, omnidireccional), como en su amplitud, la distribución de luces y sombras va a ser diferente. Por tanto la complejidad del sistema variará según las diferencias que haya en las condiciones de iluminación entre imágenes que contengan el mismo objeto.

- ***Disposición de los objetos:*** En escenas reales normalmente los objetos no se encuentran aislados, sino que suele haber un fondo no uniforme y además otros objetos. Si el objeto que se quiere reconocer está superpuesto por otro, se producen oclusiones parciales, lo que es un gran problema porque se pierden las características que se querían extraer. Cuando existe un fondo u otro objeto detrás, con características similares al del objeto en cuestión, es probable que no se pueda diferenciarlos. Esto ocurrirá fundamentalmente en la segmentación.

- ***Diferentes características de un mismo objeto:*** Normalmente los objetos cambian su forma, color, tamaño, etc. pero mantienen su identidad. Por ejemplo se puede querer reconocer un objeto bolígrafo, pero éste puede ser rojo, negro, alargado, achatado… Esto hace que un mismo objeto pueda ser representado con características muy diferentes, lo que complica el reconocimiento. En estos casos habrá que buscar rasgos identificativos que se encuentren en la mayor cantidad posible de representaciones de un mismo objeto.

A continuación se muestra más a fondo el tipo de técnicas que se utilizan en las tres etapas del análisis de imágenes.

3. Procesado de bajo nivel

En esta etapa se realiza la adquisición de las imágenes y el preprocesado de las mismas. No se comenta nada acerca de la adquisición puesto que en la mayoría de este tipo de sistemas las imágenes ya han sido adquiridas y digitalizadas, como será en el caso del proyecto.

El cuanto al preprocesado, se estudiarán diferentes técnicas que se utilizan para el realce y mejora de la imagen. El objetivo es facilitar las tareas de segmentación y reconocimiento de la imagen.

3.1 Operaciones puntuales

Son independientes de la posición, es decir, cada píxel resultante es la transformación de ese mismo píxel en la imagen original, es decir, no se involucran píxeles vecinos en la transformación de cada píxel.

3.1.1 Mejora y realce del contraste

Estas técnicas buscan aumentar el contraste de las imágenes de forma que se mejoren algunas de sus características visuales para las etapas posteriores del análisis [3] [25]. Estos algoritmos se aplican cuando la iluminación en la escena es uniforme o cuando se quiere aumentar el contraste entre los diferentes objetos de la misma. Se basan en reasignar los valores de intensidad de una imagen. Se da un mayor rango de valores a al rango de intensidad que nos interese destacar más. Normalmente se aplica a la luminancia (rango de grises).

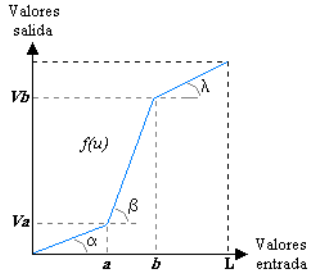

Figura 3.1 – Transformación general de la modificación del contraste

En la figura anterior se observa que lo que se hace es asignar a cada valor de luminancia un nuevo valor '*v*', de esta forma un valor '*a*' pasa a ser tras la transformación un nuevo valor '*Va*' y así para todos los valores de luminancia de la imagen. Esto se puede representar como una función entre los valores de luminancia de entrada y los de salida.

A continuación se muestran algunos ejemplos de las funciones más utilizadas y su resultado tras aplicarse a la imagen de '*Lenna*' [36], la cual se utiliza típicamente como prueba para algoritmos de compresión de imágenes debido a que contiene una buena mezcla de detalles, de zonas planas y una textura que hace trabajar bien a los sistemas de compresión de imagen.

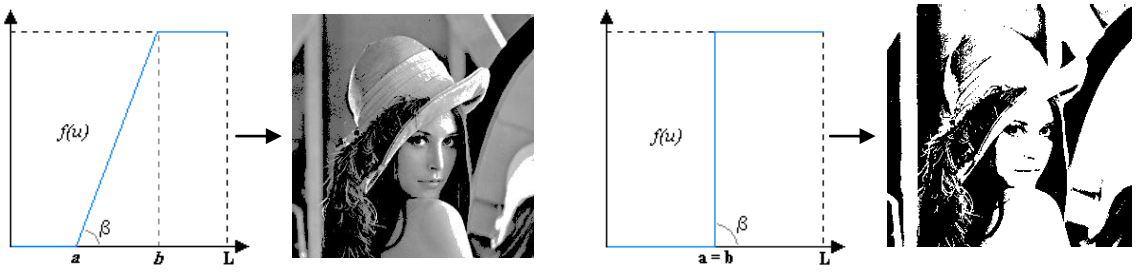

Figura 3.2 – Recorte de niveles con a = 100 y b = 200, y binarización con a = b = 120

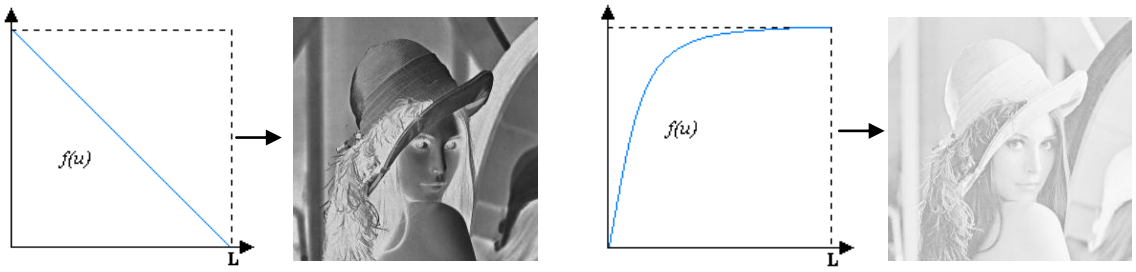

Figura 3.3 – Negativo y compresión del margen dinámico

3.1.2 Operaciones basadas en el Histograma: Igualación

Histograma: El histograma es una función discreta que contabiliza el número de ocurrencias de cada nivel de gris presentado en una imagen [25]. En el eje de abscisa está el nivel de gris y en el de ordenadas la frecuencia de cada nivel de gris en la imagen.

Histograma normalizado: Si al histograma se le divide por el número de píxeles de la imagen se obtendrá la función de probabilidad de cada nivel de gris en la imagen, lo que se denomina como histograma normalizado.

$$p(i) = \frac{h(i)}{M \cdot N} \qquad (3.1)$$

Siendo *h(i)* el número de ocurrencia del nivel de gris *i* en la imagen, *M* y *N* el número de filas y columnas de la imagen y *p(i)* la probabilidad de que ocurra el nivel de gris *i* en la imagen.

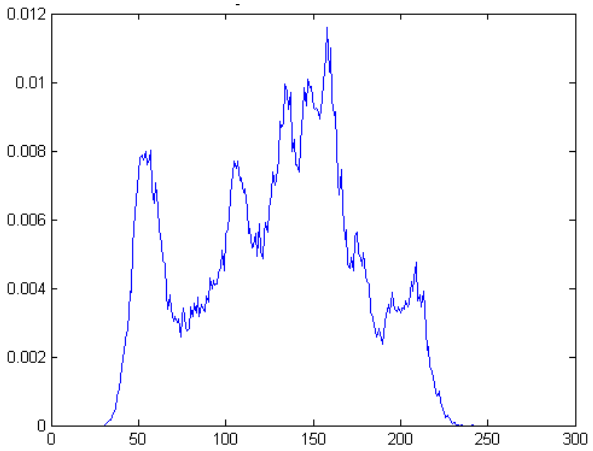

Figura 3.4 – Histograma normalizado de Lenna

Un histograma no dice nada sobre la disposición espacial de las distintas intensidades y dos imágenes diferentes pueden tener igual histograma. Sin embargo, el histograma habla sobre el tipo de adquisición que ha sido realizado y da información sobre el contraste de la imagen.

Igualación o ecualización del histograma: Esta técnica [25] se basa en tratar de transformar el histograma de una imagen para obtener un histograma uniforme, es decir, que la probabilidad de cualquier nivel de gris en la imagen sea la misma. Para obtener esta técnica se parte de la función de distribución, que para variables aleatorias discretas se define como la acumulación de todas las probabilidades, véase la Ec. (3.2)

Las características que ofrece son:

- Aprovecha mejor el número de niveles disponibles.
- Aumenta el contraste.
- En algunas ocasiones releva detalles ocultos por un bajo contraste.

$$F(i) = \sum_{i=0}^{r} p(i) \qquad (3.2)$$

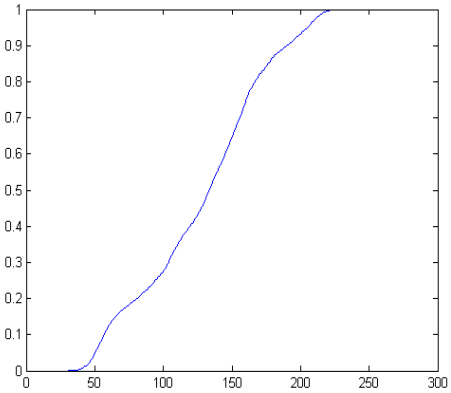

Figura 3.5 – Función de distribución de Lenna

Lo que se hace es aplicar a la imagen, como una transformación, la función de distribución, obteniendo como resultado la imagen ecualizada.

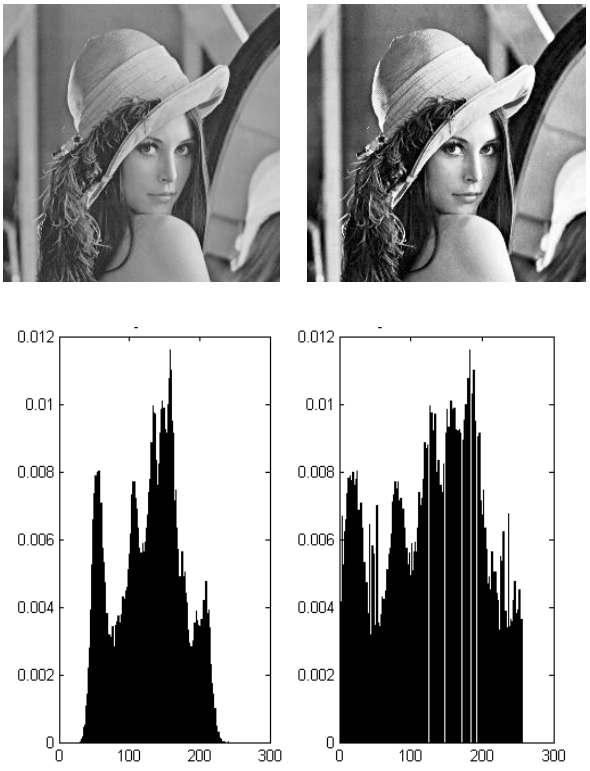

Figura 3.6 – Imagen original e imagen ecualizada mediante la ecualización del histograma

Una forma de implementarlo, al igual que cualquier transformación de intensidades (como las vistas anteriormente) es mediante un LUT (Look-Up-Table) [3]. Es una tabla que contiene la relación entre los niveles de intensidad originales y los que queremos después de la transformación, por tanto, solo hay que hacer una reasignación de niveles.

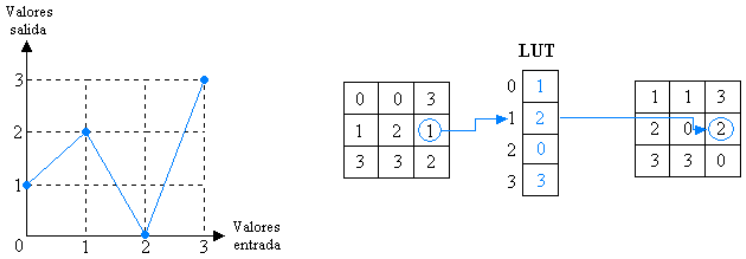

Figura 3.7 – Reasignación de niveles mediante un LUT

3.1.3 Operaciones entre imágenes

En ciertos problemas interesa trabajar en vez de con una imagen aislada, con una pareja o un conjunto de imágenes. Por ejemplo para fusión de varias imágenes o para detección de movimiento en secuencias de video. En estos casos se utilizan diversas operaciones entre las imágenes para conseguir los resultados deseados. Dichas operaciones son una extensión directa de las operaciones punto a punto. Las operaciones más utilizadas son la resta, producto, suma, división de imágenes y las operaciones lógicas.

3.2 Operaciones espaciales

Estas operaciones se aplican sobre las imágenes con el fin de realzar la imagen. Se denomina también como filtrado espacial puesto que consiste en aplicar filtros para, por ejemplo, detectar bordes o eliminar patrones de ruido. Es una operación "local" puesto que el valor de salida de cada píxel después de la transformación depende de su valor anterior y de los valores de los píxeles que lo rodean.

El filtrado espacial se realiza trasladando una matriz rectangular de dos dimensiones (llamada máscara, filtro, kernel o ventana), de tamaño mucho menor en relación con la imagen (3x3, 5x5, 7x7, etc.), que contiene "pesos" o coeficientes que determinarán el resultado de la transformación [18]. Esta se centra sobre el píxel a evaluar y se calcula su valor como el sumatorio del producto de los píxeles que se superponen entre la imagen y la máscara. Cuando se ha calculado, se desplaza la ventana sobre el siguiente píxel, realizando la misma operación. Este proceso de evaluar la vecindad ponderada del píxel se denomina "filtrado espacial". En general, el filtrado de una imagen 'f' de '$M\ x\ N$' con una máscara 'h' de '$m\ x\ n$' está dada por la siguiente expresión:

$$g(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} f(i,j) \cdot h(x+i, y+j) \quad con\ a = \frac{m-1}{2}\ \ y\ \ b = \frac{n-1}{2} \qquad (3.3)$$

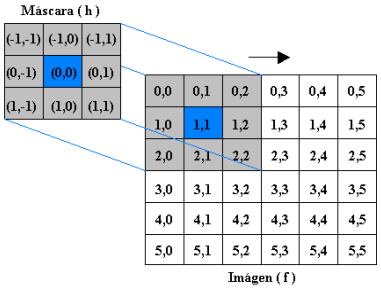

Figura 3.8 – Operación de filtrado espacial basado en máscara

La frecuencia espacial define la magnitud de cambios en el nivel de gris por unidad de distancia en una determinada zona de la imagen. Las áreas de la imagen con pequeños cambios o con transiciones graduales en los valores de los datos se denominan áreas de bajas frecuencias. Las áreas de grandes cambios o rápidas transiciones se conocen como áreas de altas frecuencias (por ejemplo, los bordes). Así, los filtros lineales espaciales se pueden dividir en:

3.2.1 Filtros Paso Bajo

Eliminan o atenúan las componentes de alta frecuencia (detalles marcados o finos de la imagen), proporcionando imágenes borrosas. Se utilizan para desenfocar, suavizar y fundamentalmente eliminar ruido. Todos los coeficientes son positivos y suman entre sí uno y lo que hacen es un promediado de los píxeles contenidos en el entorno de la ventana [18] [34] [6]. Para que no se produzca un alto desenfoque se da un mayor peso a los píxeles centrales que a los de su alrededor, véase la segunda máscara de la figura 3.9. Es posible hacer filtros direccionales, para ello en la máscara se pone cero en la dirección que no se quiere promediar. Así se preservan los bordes que pueden verse difuminados al aplicarse un filtro paso bajo.

$$\frac{1}{9}\cdot\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \qquad \frac{1}{16}\cdot\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

Figura 3.9 – Ejemplo de máscaras de Filtros Paso Bajo

3.2.2 Filtros Paso Alto

Eliminan o atenúan las componentes de baja frecuencia, resultando en imágenes principalmente con bordes acentuados. Por tanto se utilizan en detección de bordes, para enfocar imágenes borrosas, detección de piezas,… Los coeficientes deben sumar 0, por lo que aparecen números negativos. Como tras aplicarlo se obtienen valores desde -128 a +128, lo que se hace es considerar el 0 como un gris intermedio, siendo -128 el negro y +128 el blanco. Los más utilizados son las mascaras basadas en derivación direccional, como son los filtros de “Roberts” [18], “Prewitt” [34] y “Sobel” [6].

$$\begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \quad a) \qquad \begin{bmatrix}-1 & 0 & 1\\ -1 & 0 & 1\\ -1 & 0 & 1\end{bmatrix}\begin{bmatrix}-1 & -1 & -1\\ 0 & 0 & 0\\ 1 & 1 & 1\end{bmatrix} \quad b) \qquad \begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix} \quad c)$$

Figura 3.10 – Máscaras de derivación: a) Roberts, b) Prewitt, c) Sobel

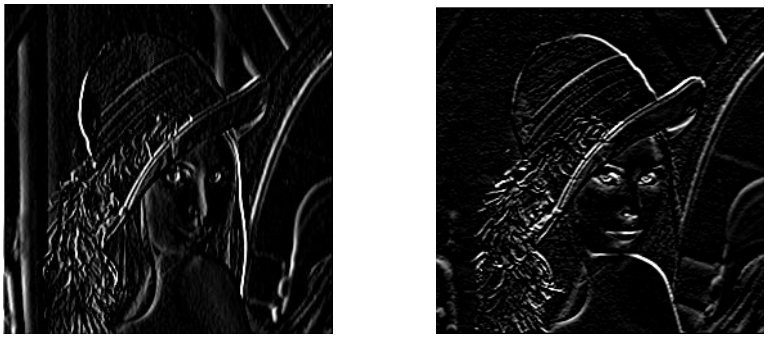

Figura 3.11 – Lenna filtrada con Prewitt para detección vertical y con Sobel para horizontal

3.2.3 Filtros Paso Banda

Principalmente se utilizan para el realce de bordes. La suma de los coeficientes ha de ser 0. Se pueden implementar como la diferencia de un filtro paso alto y un filtro paso bajo. De este tipo es el "Laplaciano", que está basado en la segunda derivada [34] [6].

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Figura 3.12 – Máscaras de tipo Laplaciano y Lenna filtrada con máscara Laplaciana

3.2.4 Filtros de estadísticos no ordenados

Los filtros estadísticos son filtros espaciales "no lineales" cuya respuesta está basada en ordenar los píxeles abarcados por una máscara y luego reemplazar el valor del píxel central con un valor determinado proveniente del resultado del ordenamiento [18]. Se suelen utilizar los filtros de máximo y mínimo.

Figura 3.13 – Lenna filtrada mediante el mínimo y el máximo

Sin embargo el más conocido de estos filtros es el filtro de medianas, el cual reemplaza el valor del píxel central por la mediana de los niveles de gris del vecindario de ese píxel. Es muy usado debido a que, para ciertos tipos de ruidos aleatorios, proveen una excelente reducción de ruido y un borroneado considerablemente menor que los filtros lineales de suavizado del mismo tamaño. Son particularmente efectivos cuando el ruido es del tipo impulso, también llamado ruido sal y pimienta [18] debido a que aparece como puntos negros o blancos sobrepuestos en la imagen. Además preserva los bordes de la imagen.

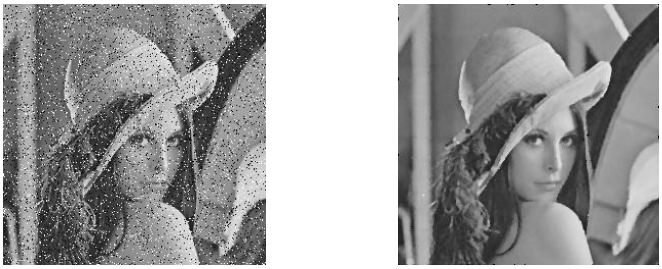

Figura 3.14 - Lenna con ruido sal y pimienta y resultado tras aplicar el filtro de medianas

3.3 Transformadas

Al igual que en el procesado unidimensional, las transformadas permiten un cambio de dominio y una simplificación, ya que expresan las imágenes como una combinación lineal de unas determinadas funciones base por unos coeficientes. Se utilizan para diversas aplicaciones, como por ejemplo: extracción de características, procesado, compresión, reconocimiento, segmentación, etc.

Se define ***F(u,v)*** como la transformada de una imagen ***f(x,y)*** según la siguiente expresión:

$$F(u,v) = c \cdot \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} f(x,y) \cdot a^*_{u,v}(x,y) \qquad (3.4)$$

donde $a_{u,v}$ son las funciones base que utilice la transformada.

3.3.1 DFT (Discrete Fourier Transform)

Utiliza como funciones base imágenes sinusoidales complejas:

$$a_{u,v}(x,y) = \exp\left(ju\frac{2\pi}{M}x + jv\frac{2\pi}{N}y\right) \qquad (3.5)$$

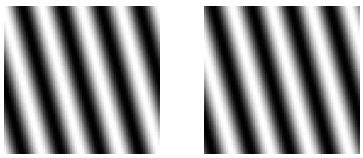

Figura 3.15 – Imagen base de la DFT (sinusoidal compleja). Parte real y parte imaginaria

La transformada discreta de Fourier [6] es por tanto compleja, aunque normalmente suele representarse únicamente el módulo. En su representación se sitúan las bajas frecuencias centradas y en los bordes las altas frecuencias.

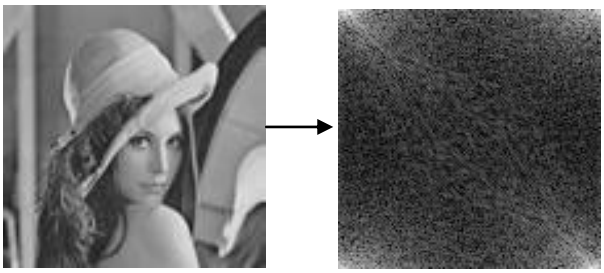

Figura 3.16 – Imagen de Lenna y módulo de su DFT

Propiedades de la DFT:

- Es compleja.
- La DFT de una imagen real es hermítica.
- *Separabilidad:* La DFT bidimensional se puede separar en dos DFTs unidimensionales, una aplicándola en el eje X y otra en el eje Y. Además se puede hacer primero la DFT en el eje X y luego en el eje Y o viceversa. Así se reduce la carga computacional.
- Desplazamiento circular.
- *Convolución circular:* Similar a la convolución pero realizando desplazamientos circulares en dos dimensiones.

Filtrado: La propiedad de convolución de la DFT es similar al procedimiento que se utiliza para implementar filtros basados en máscara, por lo que se pueden implementar filtros máscara como una convolución [6]. Para ello se realizan dos pasos:

1. Invertir la máscara con respecto al centro
2. Extender la máscara hasta el tamaño de la imagen rellenándola con ceros.

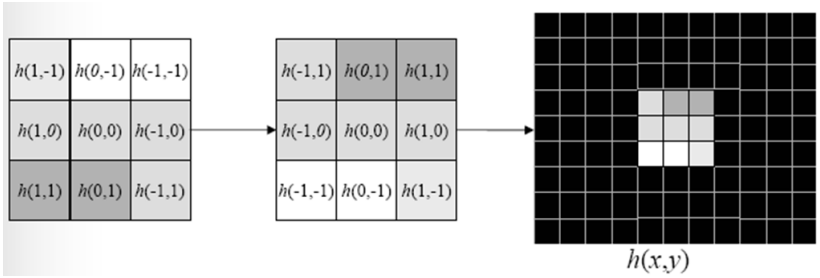

Figura 3.17 – Procedimiento para implementar un filtro máscara como un filtro DFT

Ambos procedimientos son prácticamente equivalentes, salvo posiblemente en la frontera de la imagen. La carga computacional sí es diferente, por lo que si la imagen es pequeña es mejor aplicar el filtrado basado en máscara, mientras que si la imagen es grande es mejor aplicar su equivalente DFT.

3.3.2 DCT (Discrete Cosine Transform)

Es una transformada ortogonal que se define por imágenes base reales basadas en el coseno. Se diferencia de la DFT en que sitúa las bajas frecuencias en la esquina superior izquierda, y en que interpreta como píxeles vecinos de los bordes, al mismo píxel, de forma que los cambios no son tan bruscos y así compacta más la energía en bajas frecuencias. Debido a esto se utiliza mucho para compresión en la codificación, por ejemplo en el estándar JPEG.

Propiedades de la DCT:
- Es real, ortonormal, completa y separable
- Tiene transformada rápida (Tipo FFT)
- Puede implementar filtros lineales

A partir de la DCT puede construirse la DFT de la siguiente forma [6]: se obtienen imágenes especulares (como un espejo, tanto en horizontal como en vertical y en diagonal), así se genera el desplazamiento circular que caracteriza a la DFT. Finalmente se puede aplicar la DFT a la nueva imagen formada por las imágenes especulares.

3.4 Espacio de color HSV

Aunque realmente no es un preprocesado sino que es un cambio del espacio de color de la imagen, se incluye en este apartado puesto que es muy útil para hacer más sencillo y eficaz el uso de las diferentes técnicas que se aplican sobre las imágenes. El modelo de color más utilizado es el RGB, que compone cada color mediante una combinación lineal de los tres colores primarios rojo, verde y azul.

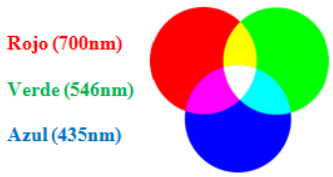

Figura 3.18 – Colores Primarios (R G B)

El problema es que cuando se aplica cualquier procesado sobre la imagen, hay que separar las tres componentes de color, aplicar dicho procesado individualmente a cada componente y finalmente combinar las 3 componentes resultantes para observar el resultado del procesado. Para solventar este problema en el tratamiento digital de imágenes se utiliza el modelo HSV.

El modelo **HSV** (del inglés *Hue, Saturation, Value* – Tono, Saturación, Valor), también llamado **HSB** (*Hue, Saturation, Brightness* – Tono, Saturación, Brillo), no es más que una transformación no lineal del espacio de color RGB, donde las coordenadas HSV se interpretan como coordenadas cilíndricas [39].

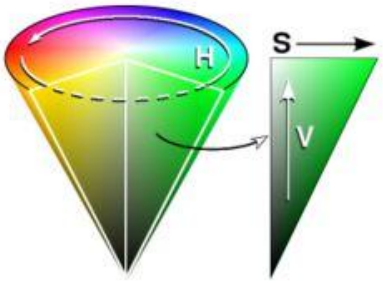

Figura 3.19 – Espacio de color HSV

Tono (H): se corresponde con el tono de color (rojo, verde, amarillo, etc.) Se representa como un ángulo, de 0º a 360º. Cada valor corresponde a un color. P.e. 0 es rojo, 60 es amarillo, 120 es verde,...

Saturación (S): indica el nivel de pureza del color, y varía entre 0 (todos los grises, incluyendo al negro y al blanco) y 1 (todos los colores puros). Cuanto menor sea la saturación de un color, mayor tonalidad grisácea habrá y más decolorado estará.

Brillo o Valor (V): representa la altura en el eje blanco-negro, es decir, la luminosidad del color. Los valores posibles van del 0 (negro) al 1 (no puede hacerse más brillante).

Sea *MAX* el valor máximo de las componentes R, G y B y *MIN* el mínimo de ellas, las componentes del modelo HSV se calculan como:

$$H = \begin{cases} no\ definido, & si\ MAX = MIN \\ 60° \times \dfrac{G-B}{MAX-MIN} + 0°, & si\ MAX = R\ y\ G \geq B \\ 60° \times \dfrac{G-B}{MAX-MIN} + 360°, & si\ MAX = R\ y\ G < B \\ 60° \times \dfrac{B-R}{MAX-MIN} + 120°, & si\ MAX = G \\ 60° \times \dfrac{R-G}{MAX-MIN} + 240°, & si\ MAX = B \end{cases} \quad (3.6)$$

$$S = \begin{cases} 0, & si\ MAX = 0 \\ 1 - \dfrac{MIN}{MAX}, & en\ otro\ caso \end{cases} \quad (3.7)$$

$$V = MAX \quad (3.8)$$

Con este modelo de color normalmente sólo se aplica el procesado que se vaya a realizar a la componente V (Valor), a no ser que se quiera variar el tono o la saturación. De esta forma se simplifica el problema, además de reducir el coste computacional del mismo.

4. Procesado de nivel intermedio

Este apartado consta de la segmentación de la imagen, que divide la misma en objetos o regiones, y de la descripción y representación de los objetos obtenidos. En el problema de reconocimiento que se plantea en este proyecto se puede obviar esta última parte y pasar directamente con las características extraídas a la clasificación.

4.1 Segmentación

Es la división de la imagen en regiones de similares características que se determinarán como objeto. Por tanto la idea de la segmentación es la agrupación de píxeles, que sean de cierta forma parecidos, en unidades (objetos) que tengan un significado dentro de la imagen. El resultado es por tanto diferentes objetos identificados en la imagen [41].

La segmentación se realiza a partir de unas características que determinen al objeto, por ello es necesario previamente definir los objetos que nos interesa identificar en la escena. Por tanto estas técnicas dependen del tipo de imágenes a analizar, y de lo que se trata es de realizar una buena segmentación de forma que se obtengan objetos con características discriminativas con respecto a los demás.

Tipos de segmentación:

1. ***Basada en características:***
   - *Segmentación por niveles de gris*
   - *Segmentación por agrupamiento*
   - *Segmentación por texturas*
2. ***Basada en transiciones:***
   - *Operadores de gradiente*
   - *Operadores de compás*
3. ***Basada en modelos:***
   - *Transformada de Hough*

4.1.1 Segmentación basada en características

Cada píxel se asigna a una región en función de un conjunto de características previamente conocidas que identifican dicha región [6]. Estas características se obtienen mediante el preprocesado previo y a partir de estas se realiza la segmentación. En función de las características elegidas se definen varios tipos, comentados a continuación.

4.1.1.1 Segmentación por niveles de gris

Es el caso en el que los diferentes objetos tienen diferentes valores de nivel de gris. El caso más sencillo sería aquel en el que el fondo es uniforme y el objeto es de un único valor de gris, de esta forma se establece un valor intermedio (umbral***)*** para separar el objeto del fondo. Por esto también se denomina ***"segmentación por umbral"*** [6].

Imagen Original *Imagen en niveles de gris* *Imagen segmentada por umbral*

Figura 4.1 - Segmentación por niveles de gris

Para este tipo de segmentación es muy útil utilizar el histograma, que representa el número de puntos de la imagen que posee cada valor de gris. Observando cualquier histograma, p.e. el de la figura 4.2, se puede ver que los lóbulos indican zonas homogéneas, por lo que los valles establecerán cuáles pueden ser los posibles umbrales de separación.

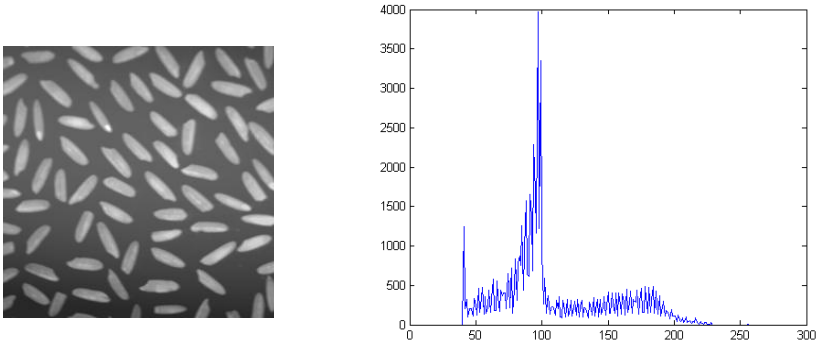

Figura 4.2 –Imagen de ejemplo "arroz" y su correspondiente histograma

Observando el histograma de la figura anterior, sería eficaz segmentar utilizando un valor entre 100 y 150, véase la figura 4.3, en la que se segmenta con un umbral igual a 130.

Existen diferentes problemas para aplicar este tipo de segmentación:

1. Objetos con un amplio rango de niveles de gris.
2. Fondo no uniforme.
3. Imagen con ruido.

Si se dan uno de estos casos o cualquier otro que impiden una buena segmentación por umbral, se puede mejorar realizando un preprocesado previo. Por ejemplo utilizar un filtro de mediana o de media.

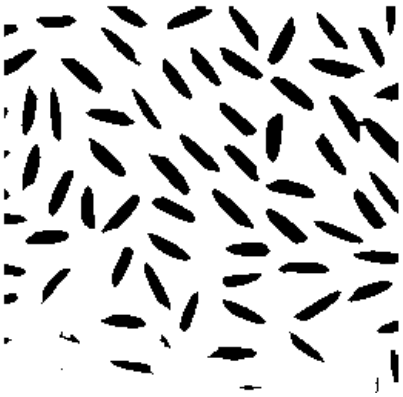

Figura 4.3 – Segmentación con umbral=130

4.1.1.2 Segmentación por agrupamiento

Se basa en que cada característica se define como un patrón o centroide y cada vector de características se asigna al centroide más próximo (mediante mínima distancia). Es muy útil cuando no se conocen concretamente las características discriminativas de las regiones a buscar o cuando no se conoce el número de objetos que hay. El agrupamiento puede realizarse mediante el algoritmo ***k-means*** [6]:

1. Se define una partición inicial con el número de centroides.
2. Los píxeles se asignan al centroide más próximo.
3. Se recalculan los centroides.
4. Se asignan de nuevo los píxeles al grupo cuyo centroide este más próximo
5. Se repite sucesivamente del paso 2 al 4 hasta que los centroides no varíen.

Si no se conocen el número de regiones, se comienza con una partición inicial de 2 centroides, se aplica el algoritmo de agrupamiento y se subdividen las regiones en dos nuevas. Se vuelve a aplicar el algoritmo y así sucesivamente hasta que se satisfaga un criterio de parada que se ha de establecer.

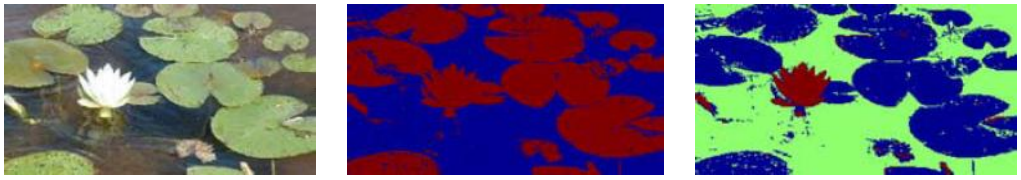

Figura 4.4 – Imagen "nenúfares". Resultado tras aplicar k-means con 2 y 3 regiones

4.1.1.3 Segmentación por textura

La textura [3] es un modelo de una parte de la imagen que queda caracterizada por la relación entre píxeles vecinos. Existen varios tipos de textura: con distintos niveles de brillo, con diferentes frecuencias espaciales, con diferentes orientaciones, etc. Para analizar la textura de la imagen se aplican filtros (operadores locales), siendo la salida de éstos la característica utilizada para segmentar.

Pueden obtenerse parámetros de textura a partir del histograma localizado (histograma de una cierta región de la imagen). Normalmente se extraen mediante una "ventana móvil", los más utilizados son:

- Dispersión, para extraer bordes finos.
- Media, para obtener bajas frecuencias.
- Varianza, para extraer bordes gruesos.
- Valor cuadrático medio o energía media, en bajas frecuencias.
- Asimetría o Kurtosis.
- Mediana, extrae un valor similar a la media pero más robusto.
- Moda, obtiene el valor de gris más probable.

Figura 4.5 – Aplicación de un filtro de varianza para segmentar por textura

4.1.2 Segmentación basada en transiciones

Estos métodos parten de la hipótesis de que las regiones a segmentar están separadas por una frontera claramente identificable. Se busca detectar transiciones entre regiones (bordes), y a partir de ellas definir las distintas regiones. Estos bordes se ven en la imagen como discontinuidades [42] y pueden ser de diferente naturaleza: intensidad, color, textura, etc.

En la figura siguiente se visualiza un borde y el resultado de aplicarle las dos primeras derivadas. La primera derivada presenta un máximo en el centro de la discontinuidad. En la segunda derivada aparece un paso por cero de la señal en el mismo punto. Por tanto la conclusión obtenida es que la detección de máximos o cruces por cero puede ser útil para la detección de bordes.

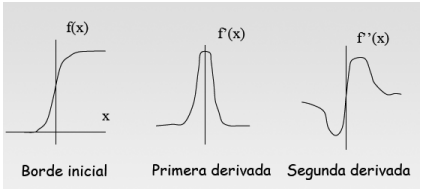

Figura 4.6 – Representación de las primeras derivadas de una frontera

Para segmentar la imagen en función de sus bordes se siguen los siguientes pasos:

- Aplicar un algoritmo de detección de bordes, obteniendo así una imagen de gradiente donde se visualizarán los bordes detectados.
- Umbralizar la imagen de gradiente para quedarse solo con los puntos que superan un cierto valor de gradiente.

Los puntos con valor alto de gradiente pueden reforzarse en función del valor de los píxeles vecinos, para así obtener bordes continuos y poder definir bien las diferentes regiones. Para esto se aplican operadores morfológicos como la *"dilatación"*. Existen dos tipos de detectores de bordes:

4.1.2.1 Operadores que se basan en calcular la primera derivada (máximos)

Son diversas máscaras espaciales que permiten calcular el gradiente o derivada de la imagen. Algunos ejemplos son los filtros de Roberts, Prewitt o Sobel [42] que se han visto en las operaciones espaciales del procesado de bajo nivel. Normalmente se utiliza una combinación de varios de ellos para obtener el gradiente en varias direcciones.

1	1	1
0	0	0
-1	-1	-1

-1	0	1
-1	0	1
-1	0	1

Figura 4.7 – Ejemplo de filtros para detección de bordes horizontal y vertical

4.1.2.2 Operadores basados en la segunda derivada (cruces por cero)

Un ejemplo del cálculo de la derivada de segundo orden es el Laplaciano [42]. La presencia de ruido en la imagen provoca en este tipo de filtro la aparición de 'falsos' bordes. Para solucionarlo se aplica previamente un filtrado gaussiano, que no es más que convolucionar la imagen con una función gaussiana. Esto da lugar a lo que se denomina filtro LOG [42], que es aplicar un filtrado gaussiano a la imagen y aplicar al resultado el operador laplaciano, obteniendo así los bordes, que serán los cruces por cero obtenido. Esto es equivalente a convolucionar la imagen con el laplaciano de la gaussiana.

0	1	0
1	- 4	1
0	1	0

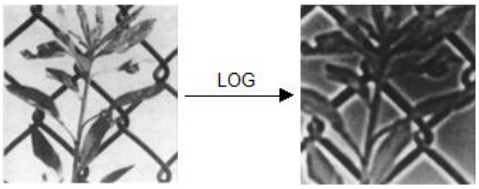

Figura 4.8 – Ejemplo de máscara Laplaciana y aplicación de un filtro LOG

4.1.3 Segmentación basada en modelos

Estas técnicas se basan en que se conocen unas determinadas características que conforman un modelo y tratan de identificarlo en la imagen [6]. Es decir, buscan zonas con formas determinadas (rectas, circunferencias, formas de bordes, etc.). No segmentan toda la imagen pero sí destacan la zona que se quieren buscar.

4.1.3.1 Proyecciones

Permite localizar objetos, se proyectan los valores de los píxeles en una dirección y se suman. Las proyecciones más utilizadas son las horizontales, las verticales y en algunos casos diagonales.

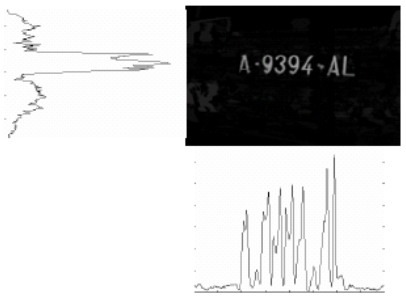

Figura 4.9 – Proyecciones horizontales y verticales de una imagen de ejemplo

4.1.3.2 Transformada de Hough

La Transformada de Hough [3] se utiliza en imágenes binarias o de escala de grises y consiste en encontrar curvas que puedan ser parametrizadas como líneas rectas, polinomios, círculos, elipses, etc.

Lo que hace el algoritmo es convertir los puntos significativos de la imagen a coordenadas polares, es decir, se convierten las coordenadas lineales (x,y) a coordenadas polares (ρ,θ) obteniendo así un espacio transformado que se denomina *"espacio de Hough"*. Por tanto cada punto pasa a ser una curva en el espacio de Hough. Observando este espacio se ve que los puntos donde cortan varias curvas representarán las posibles rectas en la imagen.

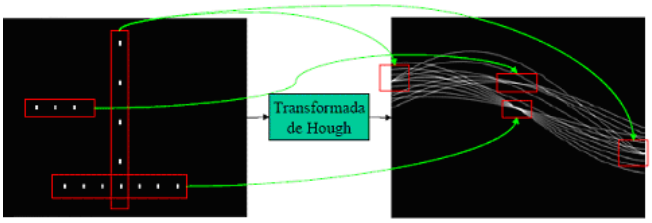

Figura 4.10 – Detección de rectas mediante la Transformada de Hough

Finalmente para obtener la imagen segmentada se realiza un filtrado teniendo en cuenta los valores más altos de la transformada de Hough, que representan las posibles rectas o curvas en la imagen.

4.1.4 Segmentación basada en homogeneidad

Se basan en asociación de píxeles, a partir de un píxel inicial o raíz, con los píxeles vecinos que tengan las mismas propiedades, por ejemplo, niveles de gris, color, etc.

4.1.4.1 Crecimiento de regiones o watersheed

El algoritmo que siguen es el siguiente:

1. Se parte de unos puntos o semillas como regiones
2. Se incluye en la misma región los píxeles vecinos con características similares (nivel de gris, textura, color,…)

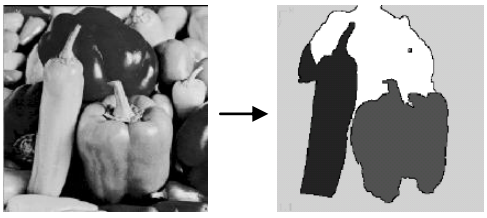

Figura 4.11 – Segmentación utilizando watersheed

Se continúa así hasta que todos los píxeles se han asignado a uno de los puntos de partida.

El problema que tiene este algoritmo es que la presencia de ruido o de texturas introduce sobre-segmentación, es decir, la imagen se segmenta en más regiones de las que en realidad hay.

4.2 Etiquetado

Tras la segmentación se procede a realizar un etiquetado de las regiones que se han obtenido para poder diferenciarlas. Como etiqueta se pueden utilizar diferentes números para cada región, aunque lo que normalmente se utiliza es un color distinto para cada objeto, para que sea más fácil diferenciarlos visualmente.

Figura 4.12 – Proceso de segmentación y etiquetado

5. Procesado de alto nivel

Tras la segmentación, extracción de características y descripción, cada objeto queda representado por una colección de descriptores (características) denominada patrón. En el reconocimiento cada patrón pertenece a una categoría o clase 'Ci'. El sistema debe asignar cada objeto a su categoría. Por tanto se considera la entrada de un sistema de reconocimiento de objetos una imagen digital, junto con algún tipo de conocimiento (patrones) acerca de uno o más modelos de objetos. La salida es una etiqueta que clasifica al objeto, en algunos casos, también se da la posición del objeto dentro de la imagen.

5.1 Patrones

Los patrones [9] son una descripción estructural o cuantitativa de un objeto o de alguna otra región de interés de la imagen. En general, un patrón está formado por un conjunto de características que han sido extraídas previamente de la imagen. Las tres representaciones de patrones principalmente utilizadas en la práctica son: los vectores (para representaciones cuantitativas), las cadenas (para representaciones estructurales) y los árboles (también para representaciones estructurales).

5.1.1 Patrones Vectoriales

Este tipo de patrones representan características cuantitativas de la imagen y se representan como:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad (5.1)$$

donde cada componente x_i es la *i-ésima* característica y *n* es el número de características. Se representan como vectores columna, pero también se pueden representar como vectores fila, véase la Ec. (5.2)

$$x = (x_1, x_2, \ldots, x_n)^T \qquad (5.2)$$

Representando las características en un espacio *n-dimensional*, si han sido bien elegidas (son discriminativas) se puede ver visualmente donde establecer las fronteras de decisión.

En el siguiente ejemplo [9] se quieren reconocer 3 tipos de lirios mediante dos características x_1 y x_2, que son la longitud y anchura de sus pétalos. En la figura 5.1 se muestra la representación de las características en un espacio bidimensional y se puede concluir que con este tipo de características se separan muy bien la "iris setosa" de las otras dos, pudiéndose establecer una frontera de decisión.

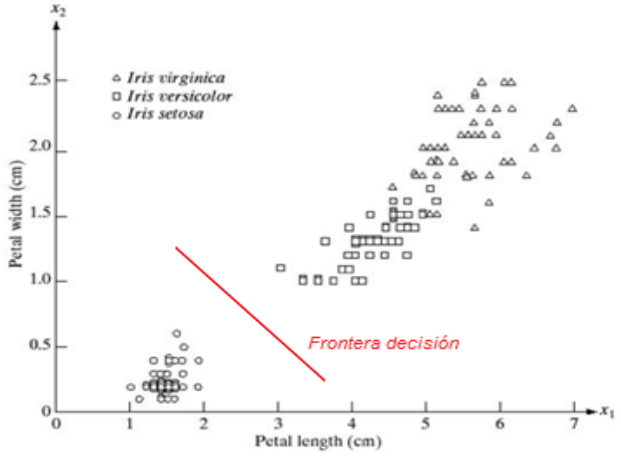

Figura 5.1 – Representación bidimensional de dos características para tres tipos de lirios

5.1.2 Patrones Estructurados

Codifican relaciones (espaciales o de otro tipo) entre componentes o características del objeto [9]. Por ejemplo el reconocimiento de huellas dactilares utiliza este tipo, basándose en las relaciones entre una serie de rasgos denominados *minucias* que describen las propiedades de los canales de las huellas dactilares.

La representación mediante cadenas no es más que una cadena de símbolos que representa una estructura. En el siguiente ejemplo se muestra un patrón en forma de escalera codificado junto a su estructura mediante dos símbolos *(a,b).*

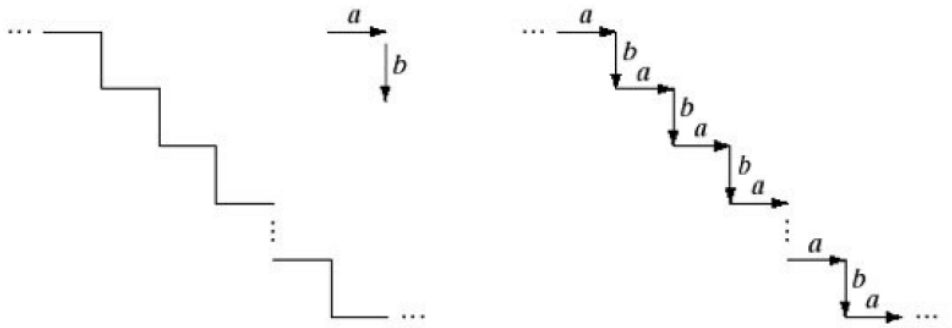

Figura 5.2 – Estructura en escalera y su codificación con dos símbolos

Se observa que sería más efectivo definir el patrón mediante una cadena de símbolos:

$$w = \ldots ababababab \ldots$$

Las representaciones en forma de cadena generan adecuadamente patrones de objetos cuya estructura se basa en conexiones sencillas, normalmente asociadas a formas de bordes o contornos. Una forma más potente que la anterior para muchas aplicaciones consiste en la utilización de descriptores en forma de árbol.

A continuación se muestra la representación en forma de árbol de una imagen aérea.

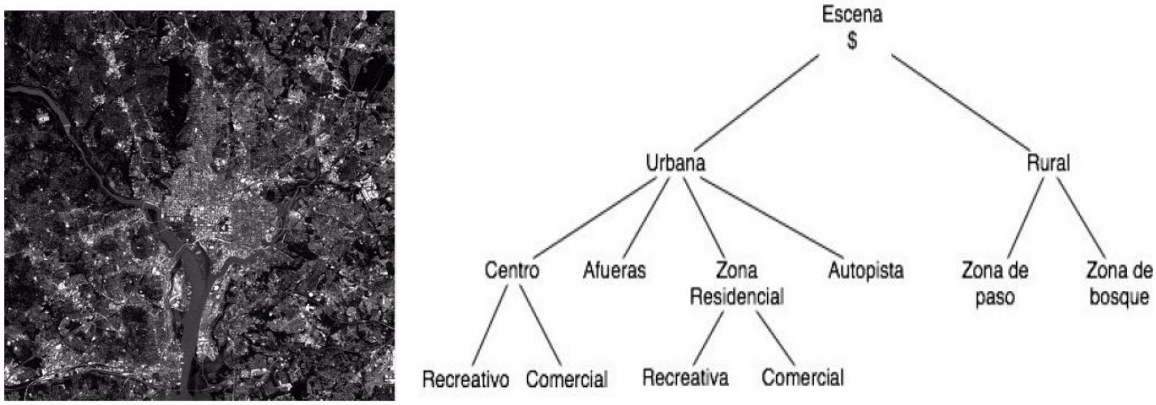

Figura 5.3 –Imagen aérea de zonas urbanas y rurales, y su representación en árbol

5.2 Reconocimiento de patrones mediante funciones discriminantes

En los siguientes apartados se comentan diversos métodos de decisión basados en la utilización de funciones de decisión o discriminantes. Se introduce un breve comentario sobre este tipo de funciones.

Tal y como se describe en [9], [2] y [15], sea un patrón $x = (x_1, x_2, \dots, x_n)^T$ de *n* dimensiones (características) y para *M* clases de patrones (objetos o regiones), $w_1, w_2, \dots, w_M$, el problema de decisión (reconocimiento) consiste en encontrar *M* funciones de decisión $d_1(x), d_2(x), \dots, d_m(x)$ que tengan la propiedad de que si un patrón *x* pertenece a la clase w_i, entonces:

$$d_i(x) > d_j(x), \quad donde\ j = 1,2, \dots, M \ \ y \ \ j \neq i \qquad (5.3)$$

Es decir, dado un patrón desconocido *x*, este pertenecerá a la clase *i-ésima* si al sustituir *x* en todas las funciones de decisión, $d_i(x)$ toma el máximo valor.

La frontera decisión que separa la clase w_i de la w_j viene dada por los valores de *x* para los que $d_i(x) = d_j(x)$, por lo que la frontera de decisión entre dos clases se define como $d_{ij}(x)$ y se puede expresar también como:

$$d_i(x) - d_j(x) = 0 \qquad (5.4)$$

De esta forma $d_{ij}(x) > 0$ para los patrones de la clase w_i; y $d_{ij}(x) < 0$ para los patrones de la clase w_j.

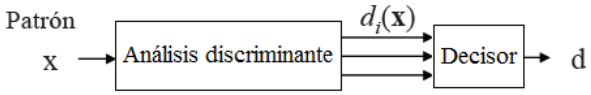

Figura 5.4 – Estructura de un sistema de reconocimiento de patrones mediante funciones discriminantes

Tras esta breve introducción se comentan algunos de los métodos que se utilizan para el reconocimiento:

1. ***Adaptación (Matching):***
   - *Clasificador de mínima distancia*
   - *Adaptación por correlación*
2. ***Clasificadores estadísticamente óptimos:***
   - *Clasificador Bayesiano*
3. ***Redes neuronales:***
   - *Perceptrón para dos clases de patrones*
   - *Perceptrón multicapa*

5.2.1 Adaptación

Representan cada clase mediante un patrón prototipo.

5.2.1.1 Clasificador de mínima distancia

Se caracteriza por la función discriminante:

$$d_i = \|x - m_i\|^2 \quad siendo\ m_i\ un\ patrón\ de\ la\ clase\ i \qquad (5.5)$$

El patrón m_i que caracteriza a cada clase puede obtenerse mediante extracción de características sobre una imagen previamente etiquetada o también puede obtenerse a partir de una colección de patrones etiquetados.

Las ventajas de este algoritmo son la simplicidad, la facilidad de ajuste y la baja carga computacional. Mientras que por desventaja tiene que solo es útil cuando las clases forman nubes poco dispersas y bien separadas.

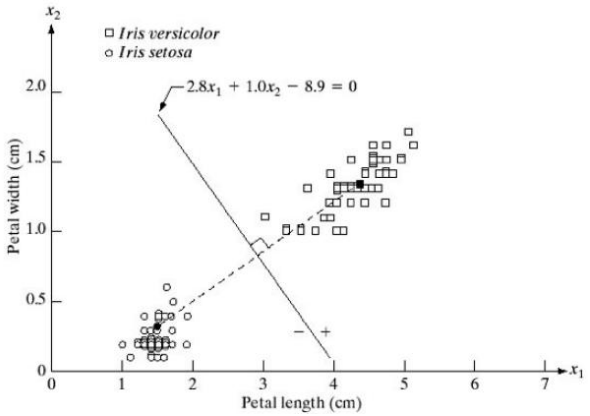

Figura 5.5 – Clasificador de mínima distancia para el ejemplo de los tres lirios

Existen diversas extensiones del clasificador de mínima distancia, que permiten obtener fronteras de clasificación más complejas y modelar categorías que no quedan adecuadamente representadas por su media. Un ejemplo de ello es el algoritmo *K-NN*.

K-NN (K Nearest Neighbours, los K vecinos más cercanos)

Cada clase *Ci* se caracteriza por una colección de prototipos m_{ij}. Cada patrón *x* se asigna a la clase mayoritaria de los *k* prototipos más próximos. Cada muestra de entrenamiento es un prototipo. Los prototipos son obtenidos mediante un algoritmo de agrupamiento, por ejemplo el algoritmo '*k-means*'

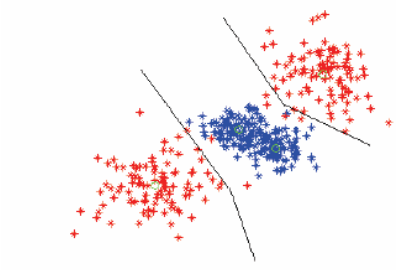

Figura 5.6 – Clasificador K-NN con K=1 y dos prototipos por cada clase

5.2.1.2 Adaptación por correlación

Este método se basa en la comparación de la imagen a clasificar, con una o varias imágenes patrón que caracterizan a cada clase. Para ello utiliza medidas de similitud como es el *coeficiente de correlación* [2], que proporciona una medida invariante frente a cambios de amplitud.

Figura 5.7 – Ejemplo de utilización del coeficiente de correlación

En la figura anterior se observan tres imágenes de águilas, donde la primera y la segunda pertenecen a la misma águila pero con otra disposición de la cámara y movimiento del animal; y la tercera es un águila de otra especie. Estableciendo como imagen patrón la primera se calcula el coeficiente de correlación de las otras dos con respecto a la imagen patrón, obteniendo:

- Coef. Correlación (Aguila1, Aguila1 (imagen movida)) = 0,53
- Coef. Correlación (Aguila1, Aguila2) = 0,12

Por tanto si se estableciese un umbral > 0,12 y < 0,53 se realizaría una buena clasificación, clasificando la segunda imagen como correspondiente a la imagen patrón, mientras que la tercera imagen se clasificaría como distinta al patrón.

Este método es muy útil para frames consecutivos como secuencias de vídeo, y aunque este ejemplo es sencillo, este algoritmo se puede utilizar para imágenes más complejas obteniendo el coeficiente de correlación entre regiones de las imágenes. Además existen otras normalizaciones que permiten obtener invariancia frente a cambios de escala u orientación [2].

5.2.2 Clasificadores estadísticamente óptimos

Las consideraciones probabilísticas tienen gran importancia en el reconocimiento de patrones, debido a la aleatoriedad a la que normalmente está sometida la generación de clases de patrones. De esta forma es posible obtener una técnica de clasificación óptima tal, que tenga asociada la probabilidad más baja de cometer errores de clasificación.

Parten del supuesto de que los patrones son realizaciones independientes de un modelo probabilístico. El más utilizado es el descrito a continuación.

5.2.2.1 Clasificador Bayesiano de patrones gaussianos

Utiliza la teoría bayesiana [26], que considera la función de densidad de probabilidad ($p(x/w_j)$) de las clases como gaussianas. De esta forma se define la función de decisión de Bayes como:

$$d_j(x) = p(x|w_j) * P(w_j) \qquad (5.6)$$

En la figura siguiente se muestra un ejemplo con dos clases. La frontera entre las dos clases es un único punto, denominado x_0, tal que $d_1(x_0)=d_2(x_0)$. Si las dos clases son equiprobables, $P(w_1)=P(w_2)=1/2$, la frontera de decisión es x_0, para el cual $p(x_0/w_1)=p(x_0/w_2)$. Este punto es la intersección de las dos funciones de densidad de probabilidad. De esta forma, cualquier patrón situado a la derecha de x_0 se clasifica como perteneciente a la clase w_1 y cualquier patrón situado a la izquierda de x_0, se clasifica como perteneciente a la clase w_2. Cuando las clases no son equiprobables, x_0 se mueve a la izquierda si es más probable la ocurrencia de la clase w_1 o, recíprocamente a la derecha si la clase w_2 tiene más probabilidad de suceder. Este resultado es de esperar, porque el clasificador trata de minimizar el error de clasificación

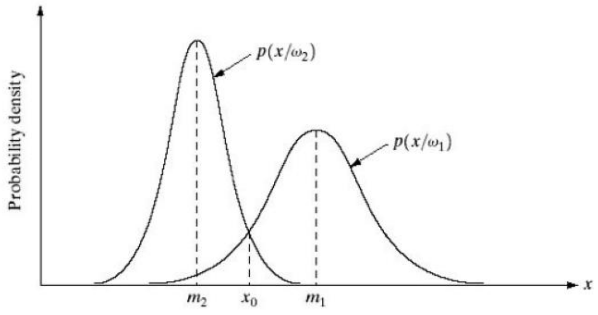

Figura 5.8 – Funciones de densidad de probabilidad para 2 clases de patrones unidimensionales

Por lo que para este problema solamente hay que estimar:

- $P(w_j)$: que puede suponerse constante, es decir, que las categorías son equiprobables.
- $P(x|w_j)$: se supone gaussiana, y por tanto la estimación se reduce a los parámetros de la media y la varianza de la gaussiana.

El gran inconveniente del modelo gaussiano es que en general no es realista, puesto que los datos normalmente no tienen una distribución gaussiana.

5.2.3 Redes Neuronales

Las redes de neuronales artificiales (denominadas habitualmente como **RNA** o en inglés como "ANN") son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales [40]. Se trata de un sistema de interconexión de neuronas en una red que colabora para producir un estímulo de salida. En inteligencia artificial es frecuente referirse a ellas como **redes neuronales**.

Las técnicas descritas anteriormente se basan en el uso de patrones ejemplo para la estimación de determinados parámetros estadísticos de cada clase de patrón. Estos patrones ejemplo se obtienen de la clasificación de imágenes conocidas (se conoce a qué clase pertenecen) que se denomina como conjunto de entrenamiento. El proceso de utilizar un conjunto de entrenamiento para obtener funciones de decisión se denomina aprendizaje o entrenamiento. En dichos métodos el entrenamiento es sencillo. Los patrones de entrenamiento de una determinada clase se utilizan de forma directa para calcular los parámetros de la función de decisión de dicha clase. Después de realizar la estimación de los parámetros en cuestión, se concreta la estructura del clasificador y su rendimiento dependerá de cómo se ajusten las distribuciones de los patrones reales a las suposiciones estadísticas realizadas en la obtención del método de clasificación.

A menudo, las propiedades estadísticas de las clases de patrones un problema son desconocidas, o no es posible realizar una estimación de las mismas. En la práctica, estos problemas de decisión se gestionan mejor utilizando métodos que obtienen directamente las funciones de decisión requeridas mediante el entrenamiento. En este apartado se muestran varias técnicas que siguen este criterio.

5.2.3.1 Perceptrón para dos clases de patrones

En su forma más simple, el perceptrón aprende una función de decisión lineal, que dicotomiza dos conjuntos de entrenamiento linealmente separables. En resumen, la respuesta de este dispositivo básico depende de la suma ponderada de sus entradas:

$$d(x) = \sum_{i=1}^{n} w_i x_i + w_{n+1} \qquad (5.7)$$

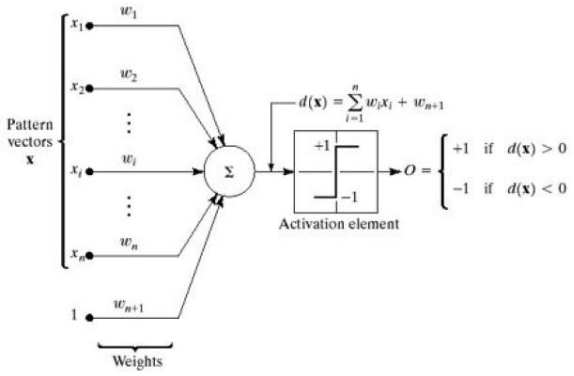

Figura 5.9 – Representación del modelo de perceptrón para dos clases de patrones

Los coeficientes w_i, denominados pesos, modifican las entradas antes de que se sumen y pasen al elemento de umbralización. La función que convierte la salida de la suma en la salida final del dispositivo se denomina función de activación.

Cuando $d(x)>0$, el elemento umbral hace que la salida del perceptrón pase a ser +1, lo que indica que se ha reconocido el patrón x como perteneciente a la clase w_1. Lo contrario es cierto si $d(x)<0$. Cuando $d(x)=0$, x se encuentra en la superficie de decisión que separa las dos clases de patrones, produciéndose una indeterminación.

La frontera de decisión implementada por el perceptrón se obtiene igualando a cero la ecuación:

$$d(x) = \sum_{i=1}^{n} w_i x_i + w_{n+1} = 0 \qquad (5.8)$$

Lo interesante de este algoritmo es que se aplican algoritmos iterativos que son los que cambian el peso de las entradas en función del éxito o fracaso de la clasificación de la entrada anterior.

5.2.3.2 Perceptrón multicapa

Se utiliza para problemas de reconocimiento más complicado, sobre todo cuando se tiene patrones multiclase.

Consta de numerosas capas de nodos de cálculo (neuronas) estructuralmente idénticas y dispuestas de modo que la salida de cada neurona de una determinada capa alimenta las entradas de todas las neuronas de la siguiente capa. El número de neuronas de la primera capa, denominada capa *A*, es N_A. Normalmente, $N_A=n$ (siendo '*n*' la dimensión de los patrones vectoriales que se introducen a la entrada). El número de neuronas de salida, denominada capa *Q*, se representa por N_Q. El número N_Q es igual a *M* (siendo '*M*' el número de clases de patrones que es capaz de reconocer la red neuronal, gracias al entrenamiento). La red reconoce un patrón vectorial x como perteneciente a la clase w_m si la m-ésima salida de la red está activada, mientras que las restantes salidas están desactivadas.

5.2.4 Máquinas de vectores de soporte (SVM)

Las máquinas de vectores de soporte (en inglés "*Support Vector Machines*", SVM), son un conjunto de algoritmos de aprendizaje supervisado desarrollados por Vladimir Vapnik y su equipo en los laboratorios AT&T en 1992 [37]. Están propiamente relacionados con problemas de clasificación y regresión. Dado un conjunto de ejemplos de entrenamiento (de muestras) podemos etiquetar las clases y entrenar una SVM para construir un modelo que prediga la clase de una nueva muestra. Intuitivamente, una SVM es un modelo que representa a los puntos de muestra en el espacio, separando las clases por un espacio lo más amplio posible. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de su proximidad pueden ser clasificadas a una u otra clase.

La idea básica es que dado un conjunto de datos de entrada o datos de entrenamiento en el que cada uno pertenece a una posible categoría se construye un hiperplano o un conjunto de hiperplanos en un espacio de dimensionalidad muy alta (o incluso infinita) que separe los datos. El objetivo final del algoritmo SVM es encontrar el hiperplano que separe los datos de manera óptima

En el ejemplo idealizado para 2-dimensiones, la representación de los datos a clasificar se realiza en el plano x-y. El algoritmo SVM trata de encontrar un hiperplano 1-dimensional que une a las variables predictoras y constituye el límite que define si un elemento de entrada pertenece a una categoría o a la otra.

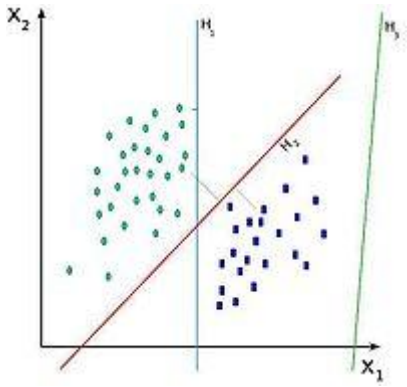

Figura 5.10 – Ejemplo de separación en 2 dimensiones mediante hiperplanos de un SVM

Existe un número infinito de posibles hiperplanos (líneas) que realicen la clasificación pero, se ha de encontrar la solución que permita un margen máximo entre los elementos de las dos categorías.

Se denominan vectores de soporte a los puntos que conforman las dos líneas paralelas al hiperplano, siendo la distancia entre ellas (margen) la mayor posible.

La manera más simple de realizar la separación es mediante una línea recta, un plano recto o un hiperplano N-dimensional. Pero desafortunadamente los universos a estudiar no se suelen presentar en casos idílicos de dos dimensiones como en el ejemplo anterior, sino que un algoritmo SVM debe tratar con más de dos variables predictoras, curvas no lineales de separación, casos donde los conjuntos de datos no pueden ser completamente separados o clasificaciones en más de dos categorías.

Debido a las limitaciones computacionales de las máquinas de aprendizaje lineal, éstas no pueden ser utilizadas en la mayoría de las aplicaciones del mundo real. La representación por medio de funciones Kernel ofrece una solución a este problema, proyectando la información a un espacio de características de mayor dimensión el cual aumenta la capacidad computacional de las máquinas de aprendizaje lineal. Es decir, se mapea el espacio de entradas X a un nuevo espacio de características de mayor dimensionalidad.

$$F = \{\varphi(x) | x \epsilon X\}$$
$$x = \{x_1, x_2, \dots, x_n\} \rightarrow \varphi(x) = \{\varphi(x)_1, \varphi(x)_2, \dots, \varphi(x)_n\} \qquad (5.9)$$

Los modelos basados en SVMs están estrechamente relacionados con las redes neuronales. Usando un función kernel, resultan un método de entrenamiento alternativo para clasificadores polinomiales, redes de base radial y perceptrón multicapa. A continuación se muestra una comparativa entre las redes neuronales y los algoritmos SVM.

Redes Neuronales

- Capas ocultas transforman a espacios de cualquier dimensión
- El espacio de búsqueda tiene múltiples mínimos locales
- El entrenamiento es costoso
- Se diseña el número de capas ocultas y nodos
- Muy buen funcionamiento en problemas típicos

SVMs

- Los kernels transforman a espacios de dimensión muy superior
- El espacio de búsqueda tiene un solo mínimo global
- El entrenamiento es muy eficiente
- Se diseña la función kernel y el parámetro de coste
- Muy buen funcionamiento en problemas típicos

5.3 Clasificación

El aprendizaje computacional consiste en crear sistemas capaces de optimizar un criterio de rendimiento usando datos de ejemplo o experiencias pasadas. Una cualidad para considerar a un sistema inteligente es la posibilidad de adaptar su comportamiento futuro respecto a experiencias obtenidas del pasado. El aprendizaje implica cambios de adaptación en el sistema, que permiten llevar a cabo las mismas tareas a partir de las mismas condiciones de un modo cada vez más eficiente. De forma general se puede decir que el aprendizaje computacional se realiza mediante dos fenómenos:

- *La perfección de una habilidad*: consiste en que el sistema produce respuestas más óptimas conforme se va ejecutando a través del tiempo.

- *La adquisición de conocimiento*: el sistema recoge y cataloga de forma supervisada o automática información externa para utilizarla en posteriores consultas.

La forma en la que el sistema de aprendizaje adquiere el conocimiento y la forma en la que proporciona su salida da lugar a los diferentes tipos de aprendizaje.

- ***Aprendizaje supervisado***. Genera una función que establece una correspondencia entre las entradas y las salidas deseadas del sistema. Un ejemplo de este tipo es el problema de "*clasificación*", donde el sistema de aprendizaje trata de etiquetar (clasificar) una serie de vectores utilizando una entre varias categorías (clases). La base de conocimiento del sistema está formada por ejemplos etiquetados anteriormente.

- ***Aprendizaje no supervisado***. Todo el proceso de modelado se lleva a cabo sobre un conjunto de ejemplos formado tan sólo por entradas al sistema. No se tiene información previa sobre las etiquetas de esos ejemplos. Ejemplos de este tipo de algoritmos son el agrupamiento y la compresión de datos.

- ***Aprendizaje semi-supervisado.*** Similar al no supervisado pero en este caso los ejemplos cuentan con etiquetas.

- ***Aprendizaje por refuerzo.*** El algoritmo aprende observando el mundo que le rodea. Su información de entrada se genera por retroalimentación que obtiene del mundo exterior como respuesta a sus acciones. Un ejemplo de este tipo es el Q-Learning.

- ***Aprendizaje multi-tarea.*** Utilizan conocimiento previamente aprendido por el sistema de cara a enfrentarse a problemas parecidos a los ya vistos.

El modelo de aprendizaje realizado en este proyecto es el aprendizaje supervisado, pues se trata de un problema de clasificación en el que se conocen previamente tanto las etiquetas de los ejemplos a clasificar como las categorías en las que serán clasificados.

Dentro del aprendizaje supervisado existen una gran cantidad de técnicas para los sistemas de clasificación, ya explicadas en apartados anteriores excepto el algoritmo K-NN que se detallará en el apartado de implementación al ser el utilizado en la clasificación del sistema desarrollado. Estas técnicas se dividen en:

- ***Técnicas estadísticas,*** discriminadores bayesianos, discriminación lineal, K-means, K-NN,…

- ***Técnicas computacionales y bio-inspiradas***, árboles de decisión, redes neuronales, máquinas de vectores de soporte (SVM´s),…

Tras los procesos de preprocesado, segmentación, extracción de características y descripción, cada objeto queda representado por una colección (posiblemente ordenada y estructurada) de descriptores denominada patrón.

La clasificación o reconocimiento de patrones [9] es el proceso por el cual se "etiqueta" (clasifica) cada objeto como perteneciente a una determinada clase. En la clasificación, al ser un sistema de aprendizaje supervisado, se conocen previamente las etiquetas de los objetos a clasificar y las clases en las que serán clasificados. Dicha información previa facilita la implementación del clasificador y servirá para evaluarlo.

Para diseñar un clasificador son necesarias dos tareas, el *Aprendizaje* y la *Validación*. Para ello el conjunto de ejemplos a clasificar se divide en dos:

- ***Conjunto de Entrenamiento.*** Se utiliza para el aprendizaje clasificador, determinando los patrones que pertenecerán a cada clase.

- ***Conjunto de Test.*** Se utiliza para validar el clasificador y obtener medidas de la calidad del mismo.

6. ImageCLEF 2008: Visual Concept Detection Task

6.1 Introducción

ImageCLEF es parte del *Cross Language Evaluation Forum* (CLEF), un evento para la evaluación de sistemas de recuperación de información multilingüe y multimedia realizado anualmente desde el año 2000. En estos momentos es uno de los congresos más importantes del mundo y en el que participan tanto grupos académicos como comerciales de todo el mundo. ImageCLEF fue propuesto por Mark Sanderson y Paul Clough de la Universidad de Sheffield [35], pero hoy en día se encuentra organizado voluntariamente por un elevado número de investigadores internacionales.

ImageCLEF [35] define anualmente varias tareas para que los grupos participantes concursen libremente sin límite de propuestas. Puesto que este proyecto se basa en una de las tareas de CLEF2008, se citan a continuación las tareas propuestas para ese año.

- Recuperación de fotografías (*Photographic Retrieval Task*)
- Recuperación de imágenes médicas (*Medical Retrieval Task*)
- Detección visual de conceptos en fotografías generales (*Visual Concept Detection Task*)
- Anotación automática de imágenes médicas (*Medical Automatic Image Annotation Task*)
- Recuperación de imágenes sobre una colección de imágenes de Wikipedia (*Image Retrieval Task from a collection of Wikipedia Images*)

6.2 Visual Concept Detection Task (VCDT)

Como se ha comentado en el capítulo 1, el proyecto está inspirado en esta tarea, que continúa con la Tarea de Anotación de Objetos de ImageCLEF 2006 y la Tarea de Recuperación de Objetos de ImageCLEF 2007. A diferencia de sus predecesoras, la VCDT de ImageCLEF 2008 tiene una fuerte interacción con la Recuperación de Fotografías de ImageCLEF 2008.

El objetivo de esta tarea es identificar visualmente conceptos de forma que posteriormente ayuden a la solución de la Recuperación de Fotografías de Image CLEF2008.

Para ello se proporcionan un conjunto de entrenamiento con aproximadamente 1800 imágenes, y un conjunto de test con aproximadamente 900 imágenes, de la base de datos "*IAPR TC-12 database"* [11]. Además se proporcionan dos archivos con las anotaciones tanto de las imágenes de entrenamiento como de test según la jerarquía de la figura 6.1.

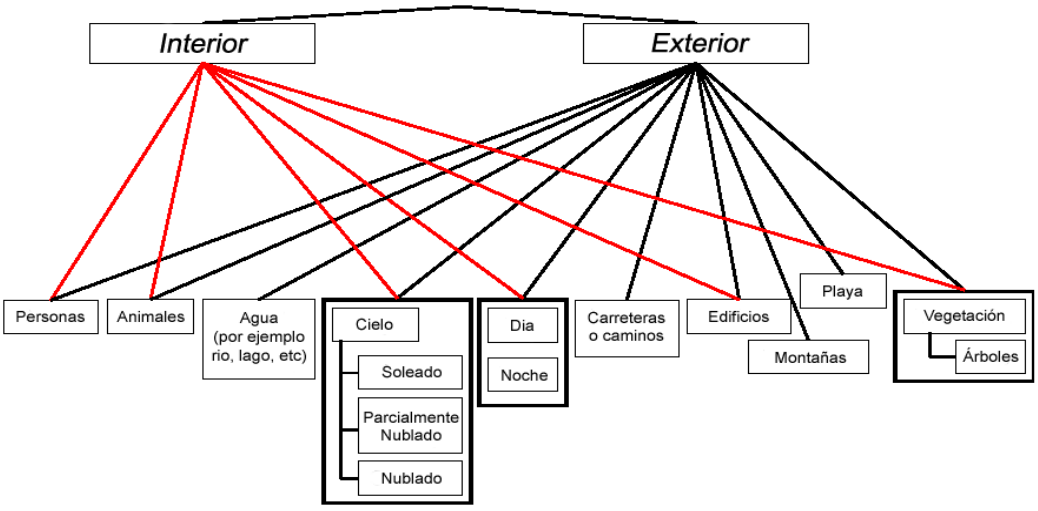

Figura 6.1 – Jerarquía de anotación de las imágenes de entrenamiento y test

El desarrollo de la tarea consiste en entrenar el sistema con las imágenes de la base de datos de entrenamiento junto a sus correspondientes anotaciones y posteriormente detectar la presencia/ausencia de los 15 conceptos anteriormente descritos, en las imágenes de la base de datos de test. Las anotaciones de las imágenes de test se utilizarán para evaluar la eficacia del sistema desarrollado, pero no se usarán lógicamente para el desarrollo del sistema.

Una vez descrito el contexto se van a detallar los datos utilizados en la implementación del proyecto.

Se utilizan los archivos de anotación de las imágenes de entrenamiento y de test, que siguen el siguiente formato:

```
27-27700.jpg 0 1 0 0 1 0 0 1 1 1 0 0 1 0 0 0 0
27-27704.jpg 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0 0
27-27704.jpg 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0 0
27-27705.jpg 0 1 0 0 1 0 0 1 0 1 0 0 1 0 0 1 0
27-27705.jpg 0 1 0 0 1 0 0 1 0 1 0 0 1 0 0 1 0
27-27706.jpg 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
27-27706.jpg 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
27-27709.jpg 0 1 0 1 0 0 1 0 0 0 0 0 1 0 0 1 0
27-27709.jpg 0 1 0 1 0 0 1 0 0 0 0 0 1 0 0 1 0
27-27712.jpg 0 1 0 0 0 0 0 1 0 1 0 0 1 0 1 0 0
27-27712.jpg 0 1 0 0 0 0 0 1 0 1 0 0 1 0 1 0 0
…
```

La primera columna indica el nombre de la imagen y las columnas sucesivas denotan la presencia *(1)* / ausencia *(0)* de los conceptos en este orden:

"Interior, exterior, personas, día, noche, agua, carreteras o caminos, vegetación, árboles, montañas, playa, edificios, cielo, soleado, parcialmente nublado, nublado y animales"

Para entrenar el sistema se utiliza la base de datos de entrenamiento, compuesta por 1809 imágenes de temática general. La cantidad de imágenes que contienen los distintos conceptos se muestra en la tabla a continuación.

Tabla 6.1 – Relación de aparición de los conceptos en las imágenes de entrenamiento

Interior	***Exterior***	***Personas***	***Día***	***Noche***	***Agua***	***Caminos***	***Vegetación***	***Árboles***
10%	90%	51%	93%	7%	25%	25%	65%	43%

Montañas	***Playa***	***Edificios***	***Animales***	***Cielo***	***Soleado***	***Parcialmente Nublado***	***Nublado***
20%	4%	49%	7%	70%	21%	23%	26%

Para evaluar el sistema se utiliza la base de datos de test, compuesta por 994 imágenes de temática general. La evaluación del sistema se realizará mediante el área bajo la curva ROC (AUC) que se explicará en el apartado posterior de evaluación. La tabla 6.2 muestra un resumen del número de imágenes que contienen cada concepto.

Tabla 6.2 – Relación de aparición de los conceptos en las imágenes de test

Interior	***Exterior***	***Personas***	***Día***	***Noche***	***Agua***	***Caminos***	***Vegetación***	***Árboles***
10%	90%	50%	94%	6%	23%	21%	62%	41%

Montañas	***Playa***	***Edificios***	***Animales***	***Cielo***	***Soleado***	***Parcialmente Nublado***	***Nublado***
19%	3%	48%	7%	72%	19%	23%	30%

A continuación y a modo de ejemplo se muestra una imagen por cada concepto de la base de datos.

Figura 6.2 – Ejemplo de imágenes para cada concepto

6.3 Métodos empleados en la VCDT

En total en esta tarea participaron 11 grupos que enviaron 53 métodos diferentes. A continuación se detallan los diferentes métodos y formas de trabajo que emplearon varios de los grupos participantes en el desarrollo de esta tarea.

6.3.1 Grupo HJFA

Jingtian Jiang, Xiaoguang Rui y Nenghai Yu.
MOE-Microsoft Key Laboratory of Multimedia Computing and Communication. Department of EEIS, University of Science and Technology of China [14].

Este grupo envió un único método llamado “Anotación de Características” para detectar los conceptos visuales ya predefinidos. El método aplicado selecciona para cada concepto tanto características locales como globales. Consiste fundamentalmente en tres procedimientos:

- Extracción de características locales y globales de las imágenes.
- Algoritmo de clustering para anotar las características.
- Clasificador K-NN para clasificar las imágenes de test de acuerdo a las imágenes de entrenamiento con las características anotadas.

6.3.1.1 Extracción de características

Se sabe que hay varios tipos de características en una imagen y que aparecen de forma variable en diferentes tareas. Después de examinar muchas características seleccionaron SIFT y características de color.

SIFT (Scale Invariant Feature Transformation) es una transformación a una imagen la cual la transforma en un conjunto de características con escala invariante. En primer lugar detecta los puntos característicos en el espacio y fija las posiciones de los puntos y la escala de las posiciones. Luego para cada punto hace uso del gradiente con los píxeles vecinos para calcular la dirección principal de dicho punto, logrando así la invarianza de escala y de dirección. Por último construye el descriptor característico para cada punto obteniendo así un conjunto de características de la imagen.

Las características de color se extraen de forma más sencilla. Se subdivide la imagen en 256 imágenes menores para extraerlas los descriptores de color. Se hace con el objetivo de que como la imagen original puede contener varios conceptos, extraer directamente sobre ella no va a representar bien a todos los conceptos. Dividiendo la imagen en pequeñas imágenes, cada una va a representar un concepto y por tanto las características van a representar los diferentes conceptos que contiene la imagen.

De esta forma se extrae el descriptor SIFT, que tiene 128 dimensiones para cada punto clave, y una imagen de unas 300 características, y el descriptor de color de 64 dimensiones con una imagen de 256 características.

6.3.1.2 Anotación de las características

Las características visuales solo proporcionan información de bajo nivel, dejando el alto nivel para la semántica en las imágenes. Por ello se introduce un método para asignar semántica a las características visuales llamando *"Función de Anotación"*, con el objetivo de reasignar a cada característica extraída un concepto de los ya predefinidos.

Sabiendo que dentro de una imagen pueden aparecer varios conceptos, que a su vez hay multitud de características que suelen ser cercanas entre sí las del mismo concepto, y que algunas de ellas incluso pueden ser ruido, pues no pertenecen a los conceptos que deseamos, se utilizó el algoritmo de agrupamiento *DBSCAN* ya que está basado en las densidades de las clases y el ruido.

Así aplicando este algoritmo de agrupamiento se obtiene que para cada concepto se tenga un conjunto de características que lo identifican.

6.3.1.3 Clasificador K-NN

Para clasificar las imágenes de entrenamiento se utiliza el algoritmo K-NN debido a que es fácil de implementar y a que es un algoritmo con aprendizaje recursivo. Se estudió que seguramente los clasificadores SVM serían más precisos pero también más difíciles de implementar.

Así para una imagen de test, se le extraen los descriptores SIFT y de color, se agrupan las características mediante el algoritmo DBSCAN y los dos conjuntos obtenidos se clasifican con el algoritmo K-NN mediante la distancia euclídea para saber a qué conceptos pertenecen. Finalmente se asignan los conceptos a la imagen de test obteniendo la clasificación de la misma.

6.3.1.4 Conclusiones

En el método empleado los parámetros de los dos algoritmos utilizados son muy importantes, los valores deben ser asignados con precisión para obtener buenos resultados. Así se seleccionó el nº de puntos del DBSCAN a 5 y el K-NN con un valor de K=25. Para la similitud de las características se utiliza la distancia euclídea, se evaluaron otras pero su rendimiento era menor. El promedio del EER es de 45,07 y el del AUC es de 19,96 por lo que el rendimiento no es bueno y algunos conceptos no se detectan por completo. El método tiene la tendencia de que se asignará más fácilmente el concepto más común.

La conclusión es que el rendimiento no es bueno debido a la simplicidad del método y se cree que con algunas modificaciones se pueden obtener mejores resultados.

6.3.2 Grupo LSIS

Herve Glotin y Zhongqiu Zhao.
Laboratoire des sciences de l'information et des systemes. UMR CNRS & Universite' Sud Toulon-Var France [13].

El método que han empleado para la tarea VCDT se basa solo en la información visual de las imágenes. En primer lugar se divide el conjunto de imágenes de entrenamiento en dos grupos: un grupo de entrenamiento y otro de validación. Se extrae a ambos un conjunto de características, donde proponen un nuevo tipo de características llamado "PEF (Profile Entropy Features – Características de Perfil de Entropía) que se añaden a las habituales medias y varianzas de color. A continuación con las características de las imágenes de entrenamiento se construyen unos clasificadores SVM, se seleccionan los mejores SVM´s con las características extraídas del grupo de validación y se utilizan para detectar los 17 conceptos visuales en las imágenes de test.

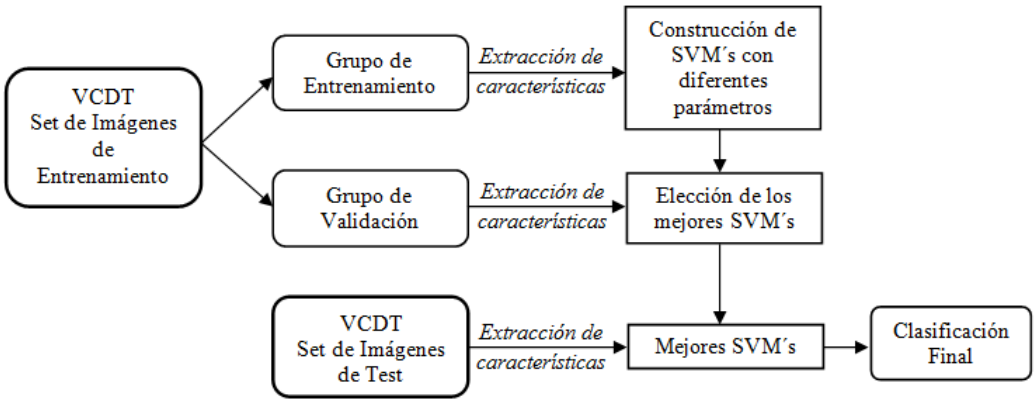

Figura 6.3 – Esquema de funcionamiento para cada concepto

6.3.2.1 Extracción de características

Proponen una nueva característica igual a la entropía del perfil del píxel. Un perfil de píxel puede ser por ejemplo una media aritmética simple en horizontal o vertical. La ventaja de esta característica es que combina las representaciones en forma y textura con un bajo coste de carga por parte del procesador. En este método se extienden este tipo de características con otra proyección para obtener el perfil de píxeles.

6.3.2.2 Máquinas de vectores de soporte (SVM)

Las SVM están basadas en una función de coste, por lo que para este problema utilizar una función de coste cuadrática optimizará el problema. Por esto utilizan máquinas de soporte con mínimos cuadrados (*LS-SVM*), que son reformulaciones de SVM´s que resuelven sistemas lineales.

Se formulan en total 100 SVM´s con valores de los parámetros diferentes para cada concepto y luego se seleccionan las mejores utilizando el conjunto de validación.

6.3.2.3 Conclusiones

En media, los resultados obtenidos para mediante las 126 características PEF que empleaba este método hacen que se sitúen los terceros en el ranking oficial del VCDT 2008. El grupo XRCE es el mejor utilizando características SIFT y las características habituales de color utilizadas por el grupo UPMC obtienen resultados similares o pocos menos discriminantes que los del grupo LSIS con PEF.

6.3.3 Grupo MMIS

Simon Overell[1], Ainhoa Llorente[2,3], Haiming Liu[2], Rui Hu[2], Adam Rae[2], Jianhan Zhu[2], Dawei Song[2] y Stefan Rüger[1,2] [24].
[1] Multimedia & Information Systems - Department of Computing, Imperial College London.
[2] Knowledge Media Institute - The Open University, Milton Keynes.
[3] Infotech Unit - Robotiker-Tecnalia, Bizkaia, Spain.

Este grupo participó en tres tareas del ImageCLEF2008: VCDT, ImageCLEF photo y ImageCLEF wiki. Para la tarea VCDT enviaron 4 métodos diferentes, los cuales utilizan el mismo sistema de extracción de características. Las características utilizadas en la tarea VCDT son una combinación de la característica de color CIELAB y la característica de textura Tamura.

CIE L * a * b (CIELAB) es el espacio de color más completo definido por el Organismo Internacional de Comisión de Iluminación (CIE). Sus tres coordenadas representan la luminosidad del color (L), su posición entre rojo, magenta y verde (a) y su posición entre el amarillo y el azul (b).

La característica de textura Tamura se calcula utilizando tres características principales de textura: "contraste", "aspereza" y "direccionalidad". El contraste tiene como objetivo captar el rango dinámico de los niveles de gris en una imagen. La aspereza tiene relación con la escala y las tasas de repetición de la textura. Y la direccionalidad revela la existencia de dirección en una imagen.

El proceso de extracción de las características es el siguiente, cada imagen se subdivide en nueve cuadros de igual tamaño y se calcula la media y el segundo momento central por canal en cada cuadro. Además se calculan el CIELAB y el Tamura para cada píxel. El vector de características resultante es la concatenación de todos los vectores extraídos en cada uno de los cuadros.

A continuación se detallan los cuatros métodos enviados.

6.3.3.1 Algoritmo de anotación automática de imagen

Este algoritmo corresponde a la labor llevada a cabo por Yavlinsky en [30]. Su método se basa en un modelo de aprendizaje supervisado que utiliza una aproximación Bayesiana junto con las técnicas de análisis de imágenes. El algoritmo utiliza características simples globales junto a la estimación de densidad no-paramétrica utilizando la técnica de suavizado de kernel con el fin de estimar la probabilidad de que las palabras que pertenecen a un vocabulario se presenten en las imágenes del conjunto.

6.3.3.2 Algoritmo de anotación automática de imagen mejorado

Es una versión mejorada de la descrita en la sección anterior. La entrada al sistema son las anotaciones obtenidas por el algoritmo desarrollado por Yavlinsky junto con una matriz que representa las probabilidades de todas las palabras del vocabulario aparezcan en las imágenes.

6.3.3.3 Algoritmo de medidas de disparidad

El tercer algoritmo sigue un enfoque basado en medidas de disparidad. Dada una imagen de test se calcula la distancia global a la media de las imágenes con la cual tiene conceptos en común. Cuanto menor sea el valor de esta distancia mayor será la probabilidad de que se etiquete dicha imagen con dicho concepto.

6.3.3.4 Algoritmo combinado

El último método es una combinación de los tres algoritmos anteriores. La entrada es el resultado de la salida del sistema de extracción de características con las características CIELAB y Tamura. Es un sistema automatizado que para cada concepto utiliza el algoritmo que proporciona un mejor resultado, siendo el resultado final la combinación de estos.

6.3.3.5 Conclusiones

Los resultados obtenidos por los cuatro algoritmos pueden verse en la Tabla 5.3. El mejor resultado corresponde al *"algoritmo de anotación automática de imagen mejorado"*, el cual fue probado anteriormente con el conjunto de datos de Corel (una colección de 5000 imágenes y 374 conceptos) obteniendo una significativa mejora del 5% al algoritmo de Yavlinsky. Sin embargo para el conjunto de imágenes de la tarea VCDT y los 17 conceptos a determinar no se obtiene una mejora tan significativa.

Tabla 6.3 – Resultados de los diversos algoritmos empleados por el grupo MMIS

Algoritmo	***EER***	***AUC***
Anotación automática de imagen mejorado	0.284425	0.779423
Anotación automática de imagen	0.288186	0.776546
Algoritmo combinado	0.318990	0.736880
Algoritmo basado en disparidad	0.410521	0.625017

6.3.4 Grupo UPMC/LIP6

Sabrina Tollari, Marcin Detyniecki, Ali Fakeri-Tabrizi, Massih-Reza Amini y Patrick Gallinari.
Université Pierre et Marie Curie-Paris et Laboratoire d'Informatique de Paris 6 [29].

El método del grupo UPMC se basa en un algoritmo de aprendizaje automático inductivo como son los bosques de árboles de decisión difusos, teniendo en cuenta las relaciones de exclusión e implicación que tienen los conceptos entre sí.

6.3.4.1 Extracción de características

Los descriptores visuales utilizados son exclusivamente basados en color. Con el fin de obtener información relacionada espacialmente se subdivide la imagen en nueve regiones superpuestas (véase la figura 6.4). Para cada región se calcula el histograma de color en el espacio HSV, el número de colores del histograma refleja la importancia de la región que será evaluada. El cuadro central representa el propósito de la imagen. Los cuadros de por encima y debajo de este se cree que darán información de ciertos conceptos como cielo, sol, vegetación,… Y el resto de cuadros se definen en términos de diferencia de color entre los de la izquierda y la derecha, para especificar si existe algún tipo de simetría.

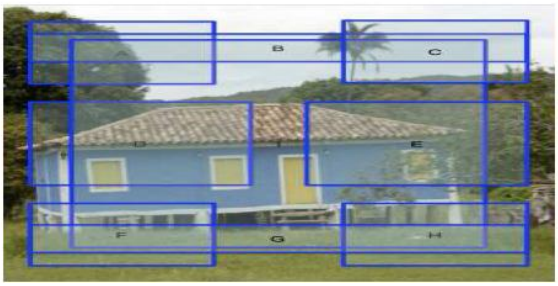

Figura 6.4 – Segmentación de la imagen para la extracción de características del grupo UPMC

6.3.4.2 Bosques de árboles de decisión difusos (FFDTs)

Una de las limitaciones de los árboles de decisión clásicos (DTs) es la robustez y los problemas que proporciona al trabajar con datos numéricos o imprecisos. La teoría de los árboles de decisión difusos (FDTs, Fuzzes Decision Trees) permite solventar estos problemas.

En la etapa de entrenamiento un bosque de FDTs (un FFDT está compuesto por varios FDTs) se construye para cada concepto X. Cada FDT F_i del bosque se construye basándose en un subconjunto T_i del conjunto de entrenamiento obtenido balanceada y aleatoriamente.

En la etapa de clasificación cada imagen *I* se clasifica por medio de cada F_i. Se obtiene un valor $d_i(I,X) \in [0,1]$ que representa el grado de que el concepto *X* aparezca en la imagen *I*. Para cada imagen *I* habrá *n* grados $d_i(I,X),\ i=1,\ldots,n$ obtenidos del bosque de FDTs. Entonces todos estos grados se agregan por simple votación, que matemáticamente corresponde con la suma de todos los grados. Finalmente se decidirá si aparece o no un concepto en una imagen estableciendo un umbral *t* (con $0 < t < n$).

Durante la etapa de clasificación se realiza un análisis de concurrencia, ya que cada árbol de decisión trata a cada concepto independientemente pero estos están interrelacionados entre sí. Por ello se tienen en cuenta los conceptos de exclusión (p.e. una imagen que contenga el concepto "día" excluye que contenga el concepto "noche") e implicación (p.e. si aparece el concepto "soleado", implica que aparezca el concepto "cielo"). Para solucionarlo se definen una matriz de co-ocurrencia y otra de simultaneidad, conociendo a priori los conceptos que excluyen o implican otros, y se definen unas reglas para determinar que concepto se determinará.

6.3.4.3 Conclusiones

Tras varias pruebas con distintos valores de umbral *t* en la clasificación, se observa que para una clasificación con simples FDTs se obtiene el mejor resultado con $t=25$, mientras que utilizando el uso combinado de la exclusión y la implicación se obtienen mejores resultados para cualquier umbral. Aún así la utilización de estas relaciones no mejora ni empeora la calidad de las etiquetas en la clasificación.

6.3.5 Grupo XRCE

J. Ah-Pine, C. Cifarelli, S. Clinchant, G. Csurka y J.M. Renders.
Xerox Research Centre Europe, France [1].

Su participación en la tarea VCDT sirvió como base para utilizar los resultados obtenidos para participar a su vez en la tarea ImageCLEF photo, que como ya se ha dicho utiliza las clasificaciones obtenidas por la tarea VCDT. Para ambas tareas del ImageCLEF 2008 se utilizó la representación de imágenes basadas en vectores Fisher. Estos son una extensión de la representación en *"bolsa visual de palabras"* (bag of visual words, BOV). La idea principal es caracterizar la imagen con el gradiente derivado del modelo de probabilidad del vocabulario. Esta representación puede ser posteriormente alimentada a un clasificador discriminativo para la categorización o utilizada para calcular las similitudes entre las imágenes para su recuperación.

Para la tarea VCDT se sabe que los conceptos se encuentran en una determinada jerarquía, que en este método no se tuvo en cuenta ni para el entrenamiento ni para la clasificación, considerándolo como un problema de clasificación multi-clase y multi-etiqueta. Usando la representación de las imágenes con vectores de Fisher se entrenaron clasificadores lineales y no lineales (basados en núcleo), donde cada clasificador detectaba la presencia/ausencia de cada uno de los 17 conceptos. Como clasificador lineal utilizaron su propio algoritmo de regresión logística "SLR" [16]. Para el entrenamiento de los clasificadores se utilizaron dos vectores de características de bajo nivel (color y textura). Se transformaron los resultados de los clasificadores en probabilidades, de tal forma que para decidir la presencia/ausencia de un concepto se compara la media de las dos probabilidades (color y textura) con un determinado umbral (0,65 en este caso).

Tanto el sistema lineal como el no lineal (basado en núcleo) ofrecen un buen resultado comparado con los demás métodos empleados por el resto de participantes. El basado en núcleo ofrece un poco más rendimiento que el lineal.

6.3.5.1 Conclusiones

Con los dos métodos enviados se obtienen los mejores resultados en todos los conceptos de los 11 grupos participantes en la VCDT. Se observa que con la representación de la imagen mediante vectores de Fisher aumenta visiblemente el rendimiento de la detección visual de conceptos en imágenes.

6.4 Resultados

Para cada método se evaluaron los resultados para cada concepto representando las curvas ROC (representación gráfica de la sensibilidad frente a 1-especificidad de un sistema de clasificación binario) y se obtuvieron los siguientes dos parámetros para evaluar el rendimiento de los clasificadores:

- Tasa de error cruzado (*EER, Equal Error Rate*), es la tasa en la cual se aceptan y rechazan los errores por igual, es decir, el punto en el que en la ROC coinciden la tasa de falsos positivos y la tasa de falsos negativos.

- Área bajo la curva ROC (*AUC, Area Under Curve*).

Además para cada método se obtuvo también las medias del EER y la AUC para todos los conceptos.

Tabla 6.4 – Resumen de los resultados de la tarea VCDT de ImageCLEF 2008

		Mejor Método			Media		
Grupo	Métodos	Ranking	EER	AUC	Ranking	EER	AUC
XRCE	2	1	16.7	90.7	1.5	18.0	89.7
RWTH	1	3	20.5	86.2	3.0	20.5	86.2
UPMC	6	4	24.6	82.7	11.0	27.2	65.2
LSIS	7	5	25.9	80.5	20.3	32.8	71.8
MMIS	4	13	28.4	77.9	23.3	32.6	73.0
CEA-LIST	3	17	29.0	73.4	26.3	33.4	59.7
IPAL-I2R	8	19	29.7	76.4	32.1	36.0	68.3
Budapest	13	20	31.1	74.9	31.8	35.2	68.6
TIA	7	24	32.1	55.6	39.6	39.9	36.3
HJ-FA	1	47	45.1	20.0	47.0	45.1	20.0
Makere	1	51	49.3	30.8	51.0	49.3	30.8

La tabla anterior muestra un resumen de los resultados obtenidos por cada grupo ordenada por el método que a cada grupo le dio un mejor resultado. Se puede observar que el grupo XRCE es el que mejor resultados obtiene.

En la tabla 6.5 se puede observar un resumen de los resultados obtenidos para cada concepto. Para cada concepto se muestra el mejor y el peor EER y AUC obtenido y la media de ambos para todos los métodos enviados. Los mejores resultados para todos los conceptos los obtuvo el grupo XRCE junto al grupo Budapest en el concepto *"noche"*. Entre estos datos cabe destacar:

- El mejor AUC de cada concepto es del 80% o superior.

- Los conceptos que obtienen mejores resultados son *"interior"* y *"noche"*.

- El concepto con peor resultado es *"carreteras o caminos"* debido a la variabilidad de aparición de este concepto.

- El concepto con más media, es decir, el que más ha sido detectado en la mayoría de los métodos es *"cielo"*.

- Al contrario, el concepto con menos media es "carreteras o caminos".

Tabla 6.5 – Resumen de los resultados por concepto

		Mejor			Media		Peor	
#	Concepto	EER	AUC	Grupo	EER	AUC	EER	AUC
00	Interior	8.9	97.4	XRCE	28.0	67.6	46.8	2.0
01	Exterior	9.2	96.6	XRCE	30.6	70.5	54.6	13.3
02	Personas	17.8	89.7	XRCE	35.9	62.2	53.0	0.4
03	Día	21.0	85.7	XRCE	35.4	64.9	52.5	9.7
04	Noche	8.7	97.5	XRCE/Budapest	27.6	72.5	73.3	0.0
05	Agua	23.8	84.6	XRCE	38.1	57.8	53.0	3.2
06	Caminos	28.8	80.0	XRCE	42.6	50.7	56.8	0.0
07	Vegetación	17.6	89.9	XRCE	33.9	67.4	49.7	30.7
08	Árboles	18.9	88.3	XRCE	36.1	62.8	59.5	1.0
09	Montañas	15.3	93.8	XRCE	33.1	61.2	55.8	0.0
10	Playa	21.7	86.8	XRCE	35.8	57.6	51.4	0.0
11	Edificios	17.0	89.7	XRCE	37.4	60.8	64.0	0.5
12	Cielo	10.4	95.7	XRCE	24.0	78.6	50.8	37.3
13	Soleado	9.2	96.4	XRCE	30.3	66.5	55.4	0.0
14	Parcialmente Nublado	15.4	92.1	XRCE	37.5	58.9	55.5	0.0
15	Nublado	14.1	93.7	XRCE	32.1	67.6	61.5	0.0
16	Animales	20.7	85.7	XRCE	38.2	54.2	58.4	0.0

Finalmente se muestra a continuación un resumen de todos los resultados obtenidos por cada método enviado por cada uno de los 11 grupos.

Tabla 6.6 - Resultados de todos los métodos enviados por todos los grupos para la tarea VCDT

Grupo	Método	EER(%)	AUC(%)
CEA_LIST	CEA_LIST_2	29.71	71.44
CEA_LIST	CEA_LIST_3	41.43	34.25
CEA_LIST	CEA_LIST_4	29.04	73.40
HJ_FA	HJ_Result	45.07	19.96
IPAL_I2R	I2R_IPAL_Cor	40.02	62.62
IPAL_I2R	I2R_IPAL_Edge	45.71	55.79
IPAL_I2R	I2R_IPAL_Hist	31.83	73.80
IPAL_I2R	I2R_IPAL_Linear	36.09	68.65
IPAL_I2R	I2R_IPAL_Texture	39.22	62.93
IPAL_I2R	I2R_IPAL_model	33.93	72.01
IPAL_I2R	I2R_IPAL_FuseMCE	31.17	74.05
IPAL_I2R	I2R_IPAL_FuseNMCE	29.71	76.44
LSIS	GLOT-metode23_LSIS	26.56	79.92
LSIS	New_kda_results	25.88	80.51
LSIS	FusionA_LSIS	49.29	50.84
LSIS	FusionH_LSIS	49.38	50.20
LSIS	MLP1_LSIS_GLOT	25.95	80.67
LSIS	MLP1_vcdt_LSISI	25.95	80.67
LSIS	Method2_LSIS	26.61	79.75
MMIS	MMIS_Ruihu	41.05	62.50
MMIS	Ainhoa	28.44	77.94
MMIS	Alexei	28.82	77.65
MMIS	CombinedReplacement	31.90	73.69
Makere	MAK	49.25	30.83
RWTH	PHME	20.45	86.19
TIA	INAOE-kr-00_HJ_TIA	42.93	28.90
TIA	INAOE-kr-04_HJ_TIA	47.12	17.58

Grupo	Método	EER(%)	AUC(%)
TIA	INAOE-1b-01_HJ_TIA	39.12	42.15
TIA	INAOE-psms-00_HJ_TIA	32.09	55.64
TIA	INAOE-psms-02_HJ_TIA	35.90	47.07
TIA	INAOE-rf-00_HJ_TIA	39.29	36.11
TIA	INAOE-rf-03_HJ_TIA	42.64	26.37
UPMC	B50trees100C5N5	27.32	71.98
UPMC	B50trees100C5N5T25	28.93	53.78
UPMC	B50trees100COOC5T25	28.83	54.19
UPMC	B50trees100pc	24.55	82.74
UPMC	B50trees100pc_COOC5	27.37	71.58
UPMC	B50trees100pc_T25	26.20	57.09
XRCE	TVPA_XRCE_KNN	16.65	90.66
XRCE	TVPA_XRCE_LIN	19.29	88.73
Budapest	Acad-acad-logreg1	37.36	66.39
Budapest	Acad-acad-logreg2	37.12	66.53
Budapest	Acad-acad-lowppnn	36.07	67.15
Budapest	Acad-acad-lowppnnpnn	32.46	73.05
Budapest	Acad-acad-medfi	32.47	73.57
Budapest	Acad-acad-mednofi	32.10	74.18
Budapest	Acad-acad-medppnn	37.01	59.30
Budapest	Acad-acad-medppnnpnn	32.47	73.61
Budapest	Acad-acad-mixed	38.34	63.80
Budapest	Acad-budapest-glob1	45.72	52.78
Budapest	Acad-budapest-glob2	31.14	74.90
Budapest	Acad-budapest-lowfi	32.48	73.03
Budapest	Acad-budapest-lownfi	32.44	73.32

PARTE III:

IMPLEMENTACIÓN

7. Arquitectura del sistema

Este proyecto tiene como objetivo la implementación de una tarea real, la clasificación de imágenes. En este caso concretamente se realiza la tarea VCDT que describe el foro de ImageCLEF 2008, ya expuesta anteriormente. El software elegido para el desarrollo de la tarea ha sido Matlab debido a que proporciona una gran cantidad de librerías para el tratamiento digital de imágenes y para multitud de operaciones matemáticas.

Para la implementación de dicha tarea se ha seguido el esquema descrito para el análisis de imágenes de la figura 1.1. No se han implementado ni la primera ni la tercera etapa del modelo, la adquisición de imágenes y la segmentación. La adquisición no se implementa debido a que como ya se ha mencionado se parte de una base de datos de imágenes ya digitalizadas, mientras que la segmentación tampoco es implementada debido a que la posibilidad de que en las imágenes puedan aparecer varios conceptos hace que una mala segmentación lleve a eliminar de la imagen la aparición de varios de estos conceptos. Para solucionar esto se debería realizar una segmentación específica para cada concepto o saltarse esta etapa y extraer las características directamente de la imagen sin segmentar. Se ha decidido por la segunda opción pues disminuye la complejidad del sistema y la carga computacional del mismo. El resto de etapas sí han sido implementadas, pues el objetivo final del sistema es la clasificación de las imágenes de la base de datos. En la figura 7.1 se muestra el modelo desarrollado para la tarea.

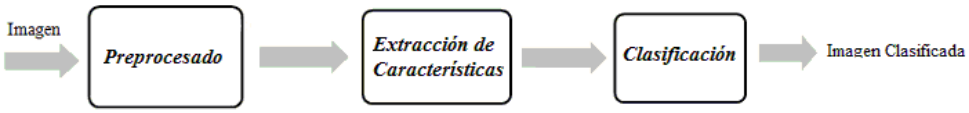

Figura 7.1 – Esquema de bloques del sistema

7.1 Preprocesado

Recibe como entrada una imagen digitalizada a la cual se le aplican varias técnicas de suavizado, que se presentan más adelante, en concreto: un filtro de medianas para eliminar el ruido impulsivo y un filtro gaussiano para eliminar el ruido gaussiano. Devuelve la misma imagen con sus cualidades mejoradas.

7.2 Extracción de Características

Recibe como entrada la imagen preprocesada y se le extraen una serie de características que puedan describir los conceptos a identificar. Este bloque se subdivide en dos sub-bloques, una extracción de características generales que se aplica a todas las categorías y una extracción de características específicas que se aplica a algunas de las categorías con el objetivo de mejorar los resultados que se obtendrían únicamente con el vector de características generales.

Extracción de características generales: se obtiene un vector de características tanto de color como de textura, como se describe más adelante. De color se obtienen los colores dominantes, las medianas y las desviaciones típicas de las componentes R, G y B del modelo de color RGB. De textura se obtienen la media, la desviación típica, el coeficiente de asimetría, el coeficiente de kurtosis, la uniformidad y la suavidad de los histogramas de las componentes R, G y B del modelo de color RGB y de la componente V del modelo de color HSV.

Extracción de características específicas: se obtiene un conjunto de características que dependen de la categoría. Es decir, para el concepto a identificar se extraen unas determinadas características que puedan describir la posible aparición o no dicho concepto. Se implementa solo para ciertas clases y el formato de estas características se detalla en páginas posteriores. Esta etapa devuelve dos vectores, el vector de características generales y el vector de características específicas.

7.3 Clasificación

Recibe como entrada los vectores de características extraídos en el proceso anterior y da como salida la clasificación de la imagen, devolviendo la aparición o no de cada uno de los siguientes 13 conceptos a determinar.

"Exterior/interior, personas, día/noche, agua, caminos, vegetación, árboles, montañas, playa, edificios, animales, cielo y tipo de cielo (soleado/parcialmente nublado/nublado)".

Se realiza una clasificación excluyente, pues según la jerarquía de anotación de las imágenes de la figura 6.1 la aparición o no de algunos conceptos implica o excluye la aparición de otros, por ejemplo la no aparición del concepto vegetación excluye que pueda aparecer el concepto árboles.

El algoritmo elegido para la clasificación es el K-NN. Basándose en él se han implementado 3 métodos diferentes, con el objetivo de evaluar si la clasificación es más eficiente utilizando características generales de las imágenes para todos los conceptos, utilizando características específicas para cada concepto o utilizando una combinación de las dos anteriores.

1. El primer método implementa un clasificador K-NN básico para cada concepto, de tal forma que dado el vector de características generales de una imagen de test a clasificar, se calculan las distancias al mismo vector de las imágenes de entrenamiento y se obtienen las *K* imágenes que proporcionen las menores distancias. Finalmente se clasifica la imagen a las clases mayoritarias en las *K* imágenes obtenidas.

2. El segundo método utiliza el vector de características específicas, por lo que se implementa únicamente para los 6 conceptos a los que anteriormente se extraen este tipo de características. Así se implementa para cada uno de los seis conceptos un clasificador K-NN básico utilizando las características específicas para obtener las *K* imágenes que proporcionen las menores distancias y por tanto el resultado de la clasificación.

3. El tercer y último método es una combinación de los dos anteriores. Se implementa de nuevo un K-NN básico que utiliza tanto los vectores de características generales como los vectores de características específicas. Se implementa de forma combinada solo para las seis categorías a las que se extraen las características específicas y de forma similar al primer método para el resto, pues las restricciones que impone la jerarquía establecida (figura 6.1) variará los resultados de diversos conceptos.

A continuación se detallan ampliamente cada una de las etapas del sistema implementado para el reconocimiento de imágenes según la tarea VCDT de ImageCLEF 2008.

8. Preprocesado

Las técnicas de preprocesado [25] pretenden mejorar o realzar las propiedades de la imagen para facilitar las siguientes operaciones de la clasificación de imágenes, tales como la segmentación, extracción de características y finalmente la interpretación automática de las imágenes.

Los métodos empleados en el preprocesado se basan bien en técnicas derivadas del procesamiento digital de señales o bien en un conjunto de procedimiento heurísticos que son combinación de técnicas del procesamiento digital de señales y otros tipos de manipulaciones matemáticas. En función de las pretensiones de sus transformaciones se pueden clasificar en:

- Realce o aumento del contraste (*enhancement*)
- Suavizado o eliminación del ruido (*denoising*)
- Detección de bordes (*edges*)

El desarrollo de estas técnicas en su mayoría se realiza en la información de luminancia de las imágenes, es decir, solo se analiza sobre imágenes en niveles de gris. Muchas de ellas pueden ser luego trasladas a imágenes a color, sin embargo no se puede generalizar.

En el apartado 2 del Estado del Arte se describen de forma general todas estas técnicas. En este capítulo y a continuación, se describen únicamente las técnicas desarrolladas acompañadas de una introducción teórica que amplía la información descrita en el Estado del Arte.

8.1 Suavizado

Las técnicas de suavizado intentan eliminar el nivel de ruido presente en la imagen. En las imágenes digitales se considera ruido a cualquier valor de un píxel de la imagen que no se corresponde exactamente con la realidad. Durante el proceso de adquisición de las imágenes se genera un ruido aleatorio que se suma a la señal. El origen del ruido es múltiple, la fuente más común se debe a la degradación de la señal en el proceso de captación (ruido de cuantificación o efecto de niebla en la imagen) y en el proceso de transmisión (posibles interferencias o errores de transmisión de los bits de información). En general se distinguen dos clases diferentes de ruido [25] [31]:

- *Ruido gaussiano*: Se caracteriza por tener un espectro de energía constante para todas las frecuencias. Tiene su origen en el ruido producido por los circuitos electrónicos o por el ruido de los sensores por falta de iluminación y/o altas temperaturas. Afecta a la intensidad de todos los píxeles.

- *Ruido impulsivo:* También denominado ruido "*sal o pimienta*" se caracteriza por la aparición de píxeles con valores arbitrarios normalmente detectables porque se diferencian mucho de los píxeles vecinos. Tiene su origen principalmente en la cuantificación que se realiza en el proceso de digitalización de la imagen, aunque también se genera al captar objetos calientes a consecuencia de la sensibilidad de las cámaras al infrarrojo. Su distribución viene dada por la siguiente ecuación.

$$g(x,y) = \begin{cases} 0 & ,si\ r(x,y) < p/2 \\ L-1 & ,si\ p/2 \leq r(x,y) < p \\ f(x,y), & si\ r(x,y) \geq p \end{cases} \qquad (8.1)$$

  Donde *r(x,y)* es un número aleatorio con distribución uniforme en [0,1) y *p* es la probabilidad de ocurrencia del ruido aleatorio, es decir, el porcentaje de píxeles que se verán afectados por el ruido impulsivo.

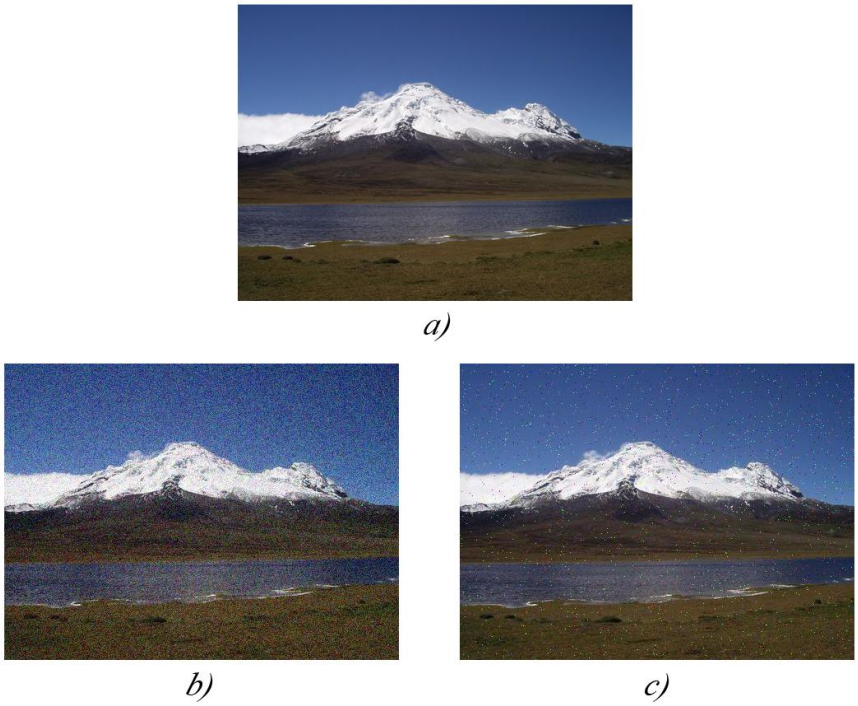

Figura 8.1 – a) Imagen original, b) Imagen con ruido gaussiano, c) Imagen con ruido impulsivo

8.1.1 Filtro Gaussiano

Los filtros gaussianos [25] son máscaras de convolución que emplean la discretización de las funciones de densidad normal de media cero y varianza dada, $N(0,\sigma^2)$:

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (8.2)$$

La respuesta en frecuencia de estas máscaras son ellas mismas, por tanto no presentan lóbulos secundarios en el módulo y son de fase lineal. Es separable, es decir, para aplicar un filtro gaussiano bidimensional se pueden aplicar dos unidimensionales, uno en sentido horizontal y otro en sentido vertical, reduciendo el número de operaciones de N^2 a $2N$.

Los filtros gaussianos se encuentran parametrizados en función de su varianza. Desde el punto de vista del dominio espacial, a medida que la varianza aumenta se tendrán en cuenta vecinos más alejados, por el contrario, a medida que la varianza disminuya se ponderará con los vecinos más próximos. Desde el punto de vista del dominio de la frecuencia la varianza determinar el ancho de banda del filtro paso bajo que supone la máscara de Gauss. El lóbulo principal del módulo de la respuesta en frecuencia viene determinado por la varianza.

En el proceso de discretización de la máscara gaussiana aparece el problema del número de coeficientes finitos a emular a la función continua. Para evitar la formación de lóbulos secundarios en la respuesta en frecuencia de la máscara se toma como regla que el tamaño del operador, w, debe ser al menos de: $w \geq 3c$, siendo c el tamaño del lóbulo central y dependiente de la varianza de la forma:

$$c = 2\sqrt{2\sigma^2} \qquad (8.3)$$

Nótese que se busca una máscara bidimensional que tenga simetría par y como además esta máscara posee simetría radial solo será necesario calcular un cuadrante de la máscara.

Implementación

Dada una imagen se le aplica un filtro gaussiano con los valores por defecto que implementa Matlab, concretamente una desviación típica de 0,5. Si se realizan los cálculos descritos anteriormente para una desviación típica de 0,5 correspondería una ventana *w* de:

$$c = 2\sqrt{2 \cdot 0{,}5^2} = \sqrt{2} \;\Rightarrow\; w = 3 \qquad (8.4)$$

Por tanto se aplica un filtro gaussiano con $\sigma = 0{,}5$ y una máscara de 3 x 3.

Entre las funciones empleadas cabe destacar la utilización de las siguientes.

- ***fspecial*** → Define un tipo de filtro entre varios dados, en este caso un filtro gaussiano con los valores por defecto que establece Matlab (máscara de 3 x 3 y $\sigma = 0{,}5$).

- ***imfilter*** → Filtra la matriz de datos (imagen) con el filtro que se le pasa como parámetro, en concreto se le pasa el filtro definido con la función anterior.

Resultados

Para una imagen dada de la base de datos se le aplica la implementación realizada y se obtiene la imagen con un pequeño suavizado. Es importante remarcar que debido al filtrado aparece un marco indeseado bordeando la imagen que posteriormente se tratará. En la siguiente figura se observa el resultado para una imagen de ejemplo de la base de datos de entrenamiento.

Figura 8.2 – Imagen original e imagen filtrada con un filtro gaussiano

8.1.2 Filtro de Medianas

El filtro de medianas [25] [34] es un filtro estadístico, por lo que realiza un filtrado espacial no lineal cuya respuesta está basada en reemplazar el valor del píxel central por la mediana de los niveles de gris de los píxeles vecinos (el valor original del píxel es incluido en el cálculo de la mediana). La mediana se define como el valor intermedio de la ordenación de los píxeles vecinos en función de su intensidad. Si los vecinos son *2M+1* píxeles, la mediana estará por encima de los *M* primeros valores y por debajo de los *M* últimos. Por ejemplo, en una vecindad de 3 x 3, la mediana será el nivel colocado en la quinta posición:

$$\begin{pmatrix} 89 & 87 & 14 \\ 92 & 75 & 56 \\ 99 & 78 & 90 \end{pmatrix}$$

$$\{89 \quad 87 \quad 14 \quad 92 \quad 75 \mid 56 \quad 99 \quad 78 \quad 90\} \Rightarrow \{14 \quad 56 \quad 75 \quad 78 \quad 87 \mid 89 \quad 90 \quad 92 \quad 99\}$$

Figura 8.3 – Ejemplo numérico de filtrado mediante el filtro de medianas

En este ejemplo la mediana será el nivel 87. En consecuencia, el ruido que tiene un valor atípico en el entorno quedará colocado en los extremos de la ordenación. Véase como el nivel 14 queda desplazado a uno de los extremos. De esta forma la mediana lo que consigue es hacer que los puntos con niveles de gris diferentes sean más parecidos a los de su vecindario.

Los filtros de mediana son muy utilizados debido a que, para ciertos tipos de ruidos aleatorios, proveen una excelente reducción de ruido y un borroneado considerablemente menor que los filtros lineales de suavizado del mismo tamaño. Son particularmente efectivos para la cancelación del ruido impulsivo (también llamado ruido sal y pimienta) debido a que este aparece como puntos negros o blancos superpuestos en la imagen.

El mayor inconveniente es su alto coste computacional, cuando la aplicación requiere de restricciones de tiempo se opta por el filtrado lineal.

Otro efecto adverso de la mediana es que produce un emborronado considerable en la imagen y provoca el desplazamiento de los bordes de la misma.

Implementación

Dada una imagen se le extraen las componentes R G B aplicándolas de forma independiente a cada una un filtro de medianas con una máscara de 3x3, pues el filtro de medianas trabaja a nivel bidimensional. A continuación se obtiene la imagen filtrada reconstruyéndola con las componentes que han sido filtradas.

Entre las funciones empleadas cabe destacar las funciones descritas a continuación.

- ***medfilt2*** → Realiza el filtro de medianas a una matriz de datos en dos dimensiones. Se ha utilizado el atributo *'symmetric'* para que durante el filtrado se interpreten los valores externos a los bordes de la imagen de forma simétrica.

- ***cat*** → Concatena arrays o matrices de datos. En la implementación realizada concatena las componentes R G B ya filtradas para obtener la imagen a color resultante tras aplicar el filtrado.

Resultados

Para una imagen dada de la base de datos se le aplica la implementación realizada y se obtiene la imagen suavizada. Al igual que ocurre con el filtrado gaussiano, se añade un borde indeseado además de unos píxeles negros en las esquinas de la imagen (figura 8.5) que se tratarán a continuación.

Figura 8.4 – Imagen original e imagen filtrada con filtro de medianas

8.2 Preprocesado en cascada

Durante la etapa de preprocesado se realizan las técnicas expuestas anteriormente en cascada. Así, a cada imagen de la base de datos se le aplica en primer lugar un filtro gaussiano para suavizar la imagen y eliminar el posible ruido gaussiano que pueda incluir. A continuación se le aplica un filtro de medianas que reduce los efectos de desdibujación de bordes y cancelación de pequeños detalles que introduce el filtrado gaussiano, además de eliminar el posible ruido impulsivo que pueda existir en la imagen.

La aplicación de dichos filtrados, como ya se ha comentado, introduce unos errores en la imagen (un marco indeseable y un valor de los píxeles de las esquinas diferentes a los reales) como se observa en la Figura 7.4. Esto se debe a que al aplicar ambas máscaras en los píxeles de los bordes, como las imágenes no son infinitas se deben dar valores a los píxeles vecinos de los bordes externos a la imagen (igual a cero en el caso del filtro gaussiano y con valor simétrico a los bordes en el caso del filtro de medianas).

Figura 8.5 – Detalle ampliado de los errores que introducen el filtrado gaussiano y de medianas

Este marco no se tendrá en cuenta en los siguientes procesos pues proporcionaría información irreal de la imagen, por ello se elimina implementando el siguiente método. Primero se recorta la imagen, eliminando así los bordes indeseables pero provocando una disminución del tamaño de la imagen, que se soluciona reajustando la imagen a su tamaño original. Finalmente para la eliminación de los píxeles de las esquinas, se recalculan sus valores como la media de sus vecinos más cercanos, obteniendo así unos valores más aproximados a los valores reales antes del filtrado.

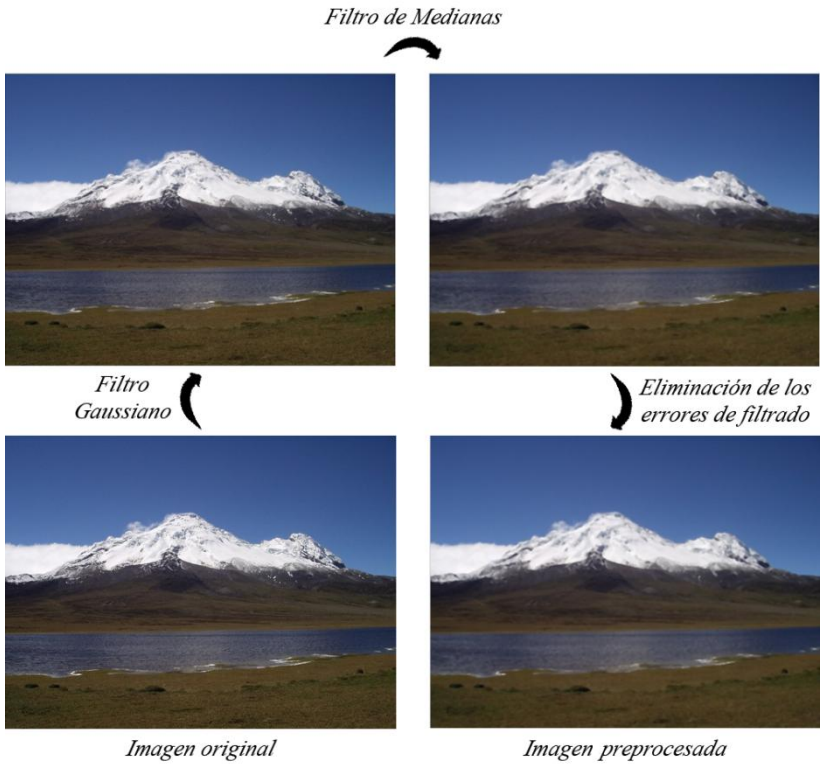

Figura 8.6 – Esquema de los procesos realizados durante el preprocesado

9. Extracción de características

La extracción de características pretende obtener una serie de rasgos característicos de una imagen, de tal forma que puedan describir en cierta forma el contenido de la misma. Así imágenes con contenidos similares pueden ser reconocidos y a su vez desechar la aparición de contenidos diferentes, siendo finalmente convenientemente clasificadas.

En los problemas de clasificación de información multimedia es necesario desarrollar un sistema capaz de describir el contenido de los vídeos o imágenes con las que se trata. Los encargados de esta descripción son los llamados *descriptores audiovisuales* [32]. Estos descriptores trabajan con características básicas que se presentan en los contenidos, y se basan principalmente en características de color, textura y forma. Con ello se proporciona un buen conocimiento de los objetos, que a su vez proporcionará una buena clasificación.

La principal tarea de este proyecto se basa en describir la aparición o ausencia de los conceptos descritos en el apartado 6.2. Para ello se intenta describir la información visual de estos conceptos en las imágenes a través de descriptores de color y de textura.

La forma más habitual de tratar este problema es la extracción de una serie de características que describan de forma general el contenido de la imagen. Así se pueden establecer similitudes y diferencias entre las imágenes a tratar. Sin embargo, en la tarea a realizar cabe la posibilidad de que en una misma imagen se presenten o no diversos de los conceptos a identificar. Debido a esto y con el fin de obtener una mejor clasificación en algunos de estos conceptos se ha optado por dividir esta etapa en dos sub-etapas.

En primer lugar se implementa la forma habitual, en la que se extrae un vector de características con el objetivo de describir el contenido de la imagen. A esta sub-etapa se le denomina *"Extracción de Características Generales"*.

En segundo lugar se obtiene un vector de características con descriptores que proporcionen información relevante sobre la aparición/ausencia de un determinado concepto. A esta se le denomina *"Extracción de Características Específicas"*. No se implementa para todos los conceptos debido a la alta complejidad y coste computacional que implica en algunos de ellos.

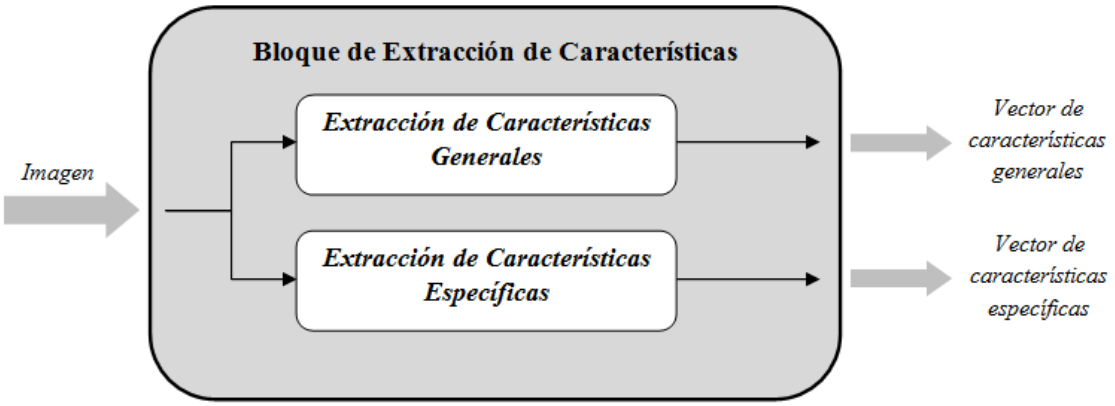

Figura 9.1 – Esquema de la etapa de Extracción de Características

A continuación se describen con más detalle cada una de estas dos sub-etapas, los descriptores que se obtienen en cada una de ellas y los detalles que se consideren oportunos de su implementación en Matlab.

9.1 Extracción de Características Generales

Dada una imagen de la base de datos se extrae un vector de características (vector de características generales) sin tener en cuenta ningún concepto a identificar. Se encuentran basados en características de color y textura.

- De color se extraen los colores dominantes, la cuantificación de color y las desviaciones típicas normalizadas.
- De textura se obtiene una serie de parámetros que caracterizan de forma estadística la textura de la imagen, estos son: la media, la desviación típica, el coeficiente de asimetría, el coeficiente de kurtosis, la uniformidad y la suavidad.

Se obtienen para cada imagen de la base de datos un total de 59 parámetros, que a continuación se detallan incluyendo la información teórica y los detalles que se consideran oportunos de su implementación en Matlab.

9.1.1 Colores Dominantes

El estándar MPEG-7 contiene varios descriptores de contenido multimedia, entre ellos el *Dominant Color Descriptor (DCD)* [27], el cual provee una descripción compacta de los colores en una imagen y cuyo objetivo es describir los colores más representativos en una imagen. A diferencia de los tradicionales descriptores basados en el histograma, los colores dominantes son calculados a partir de cada imagen en vez de cada componente del espacio de color, permitiendo así que la representación de color sea precisa y compacta.

El DCD se define como:

$$F = \{(c_i, p_i, v_i), s\} \quad con\ i = 1,2, \dots, N \qquad (9.1)$$

donde N es el número de colores dominantes. Cada color dominante c_i es un vector compuesto por los valores del color dominante en cada componente del espacio RGB. El porcentaje p_i (normalizado entre 0 y 1) es la fracción de píxels de la imagen que corresponden al color c_i. La varianza de color v_i describe la variación de los píxeles a los colores dominantes a los que corresponden, este parámetro es opcional. Al igual que el parámetro anterior, la coherencia s es también opcional, y es un número que representa la homogeneidad espacial total de los colores dominantes en la imagen.

El número de colores representativos N puede variar, aunque se ha comprobado que con un máximo de 8 colores dominantes es suficiente para representar el color en una imagen.

Implementación

Se ha elegido por obtener $N=5$ colores dominantes, pues se consideran suficientemente significativos para describir el color de una imagen y se acercan al máximo de 8 que establece el estándar.

Dada una imagen de la base de datos se obtienen sus componentes RGB y para cada una de ellas se disponen los valores de sus píxeles en un vector columna. Se genera una matriz con los 3 vectores anteriores, disponiendo así en cada fila el valor de los píxels para cada componente. A esta matriz se le aplica un algoritmo de agrupamiento o clustering, en este caso se ha utilizado el algoritmo *Kmeans* con $K=5$ centroides. Los centroides obtenidos serán los cinco colores dominantes (c_i) y estarán compuestos por sus tres valores RGB.

Posteriormente se obtiene el porcentaje de píxels de la imagen que corresponde a cada centroide (p_i).

Se decide no calcular las varianzas de color (vi) y la coherencia (s) puesto que el estándar las establece como opcionales y aumentarían considerablemente la carga computacional del programa.

Como salida se proporciona un vector con los 5 colores dominantes obtenidos (indicando para cada uno su valor RGB) y el porcentaje de píxels que corresponde a cada color dominante.

Entre las funciones empleadas cabe destacar la utilización de las siguientes.

- ***im2col*** → Dispone los datos de una matriz en un vector columna.

- ***kmeans*** → Divide los datos de una matriz de entrada en un número de grupos igual al número de centroides. La división la realiza iterativamente mediante la distancia euclídea (la que establece Matlab por defecto) de los datos a los centroides, los cuales se recalculan durante cada iteración. Se ha utilizado el atributo '*singleton*', el cual cuando un grupo se queda vacio crea un nuevo grupo a partir del punto más lejano al resto de los centroides.

Resultados:

Tomando la misma imagen de ejemplo que en apartados anteriores, se muestra el agrupamiento de los píxels en los diferentes grupos con el valor de sus centroides (colores dominantes) directamente en la imagen. Se observa que 5 colores dominantes son suficientes para obtener una buena representación del color de una imagen.

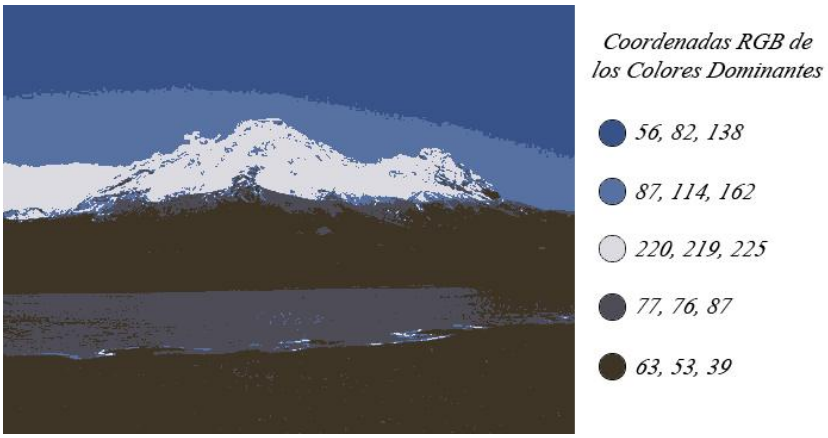

Figura 9.2 – Agrupamiento de los píxels en 5 Colores Dominantes en la imagen de ejemplo

Véase la tabla a continuación para visualizar los porcentajes de los cinco colores dominantes extraídos a la imagen de ejemplo anterior.

Tabla 9.1 – Colores Dominantes y sus correspondientes porcentajes de la imagen de ejemplo

R	***G***	***B***	***Porcentaje (0-1)***
87	114	162	0,1378
77	76	87	0,13
220	219	225	0,0825
56	82	138	0,2407
63	53	39	0,409

9.1.2 Cuantificación de Color

La implementación de este descriptor se ha inspirado en el descriptor *Color Quantization* [7] que establece el estándar MPEG-7, el cual define una cuantificación uniforme de un espacio de color determinado. Soporta cuantificadores lineales y no lineales como por ejemplo las Look-Up-Tables. Dentro de MPEG-7 este descriptor se combina con descriptores de color dominante para expresar el significado de los valores mediante un histograma de color.

Implementación

No se ha implementado tal y como lo describe el estándar, sino que se realiza una cuantificación en 5 niveles para cada una de las componentes RGB de la imagen. Se divide el rango de intensidades [0,255] en los siguientes 5 sub-rangos: [0,51] [52,102] [103,153] [154,204] [205,255] (elegidos estos para obtener en únicamente cinco niveles información de color global de toda la imagen) y para cada componente se cuantifican los píxels en los 5 niveles según al sub-rango al que pertenezcan, véase la figura 8.3.

Se puede decir que es como una simplificación del histograma de 256 niveles a 5 niveles. Con este descriptor se proporciona información acerca de la probabilidad de aparición de los píxels en determinados rangos de intensidad de la imagen. Como salida se proporciona un vector con la probabilidad de aparición o distribución de cada uno de los 5 niveles para cada componente.

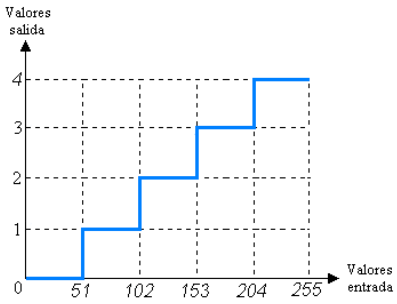

Figura 9.3 – Cuantificación en 5 niveles

Resultados

Continuando con la imagen tomada como ejemplo, las probabilidades de aparición de los 5 niveles de intensidad tras la cuantificación son los siguientes.

Tabla 9.2 – Distribución en 5 niveles de los píxels de la imagen cuantificada

Niveles	***Distribución en R***	***Distribución en G***	***Distribución en B***
0 – [0,51]	0,1596	0,1679	0,3561
1 – [52,102]	0,7299	0,6365	0,1708
2 – [103,153]	0,0240	0,1083	0,2614
3 – [154,204]	0,0235	0,0244	0,1439
4 – [205-255]	0,0629	0,0629	0,0677

En la figura 9.4 se observan las componentes RGB de la imagen, cuantificadas en 5 niveles. Para su correcta visualización se ha establecido como valor de intensidad para cada nuevo nivel el valor de intensidad medio del rango de niveles originales al que pertenezca cada píxel.

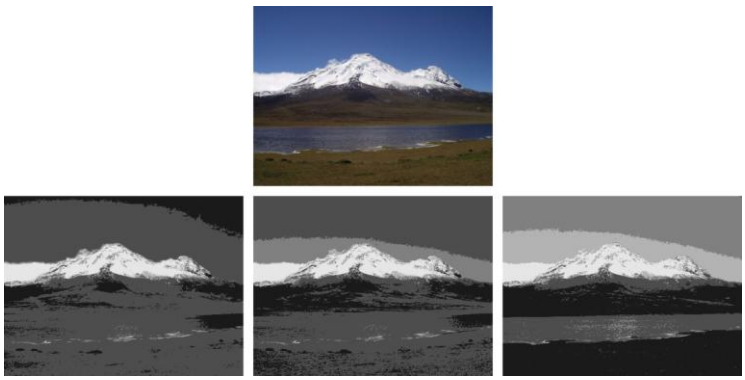

Figura 9.4 – Imagen de ejemplo y sus componentes R G B cuantificadas con 5 niveles

9.1.3 Parámetros estadísticos de Textura

La textura [33] es una característica importante utilizada en la identificación de objetos o regiones de interés en una imagen. El uso de la textura para identificar una imagen proviene de la habilidad innata de los humanos para reconocer diferencias texturales. Por consiguiente, la textura es una característica de difícil definición, siendo la más extendida la dada por Haralick [12]: ***"Una textura está definida por la uniformidad, densidad, grosor, regularidad, intensidad y direccionabilidad de medidas discretas del tono en los píxels y de sus relaciones espaciales"***.

En general la textura [19] es una propiedad asociada con las superficies tales como suavizado, rugosidad, granularidad, regularidad, etc. En sistemas de reconocimiento de imágenes la textura se considera como la repetición espacial de patrones en una superficie. Por ejemplo las puntadas de un jersey, las piedras de un camino o los granos de arena de una playa. La repetición de estos patrones puede ser periódica, como en el jersey, pero generalmente son estadísticos, como en las piedras y los granos de arena, la densidad de las piedras o los granos puede ser diferente en distintas regiones.

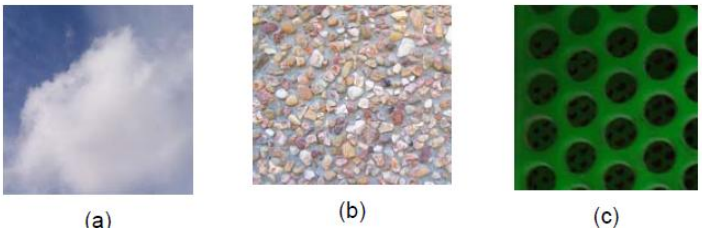

Figura 9.5 – Propiedades de Textura: (a) Suavidad, (b) Rugosidad, (c) Regularidad

Las texturas [28] ofrecen información espacial de los niveles de intensidad en una imagen. No pueden definirse para píxels aislados. Vienen dadas en su aspecto cuantitativo por la distribución espacial de los niveles de intensidad en una región. No se trata de una característica local y dependen en gran medida del grado de resolución de la imagen.

Como se ha visto, el concepto de textura depende del contexto en el que se estudie, pero para todos los contextos presenta los siguientes rasgos:

- La frecuencia de cambio de tono de los píxels.
- La dirección o direcciones de cambio.
- El contraste entre un píxel y los de su vecindad.

Sin embargo, es independiente del tono o color de la imagen. En el ejemplo de la figura 9.6 se muestran dos imágenes de una misma textura en diferentes tonos. Aunque una sea en tonos claros y otra en tonos oscuros se observa que se trata del mismo tipo de textura.

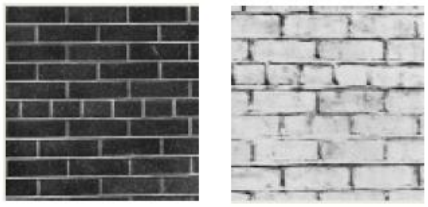

Figura 9.6 – Ejemplo de independencia del tono en la textura

Para caracterizar las texturas existen tres métodos de procesar las imágenes digitales y extraer su información:

- ***Descriptores estadísticos*** → se basan fundamentalmente en momentos estadísticos del histograma y obtienen características como la suavidad, rugosidad y granularidad.

- ***Descriptores estructurales*** → intentan obtener elementos básicos que se repiten en la imagen a través de propiedades invariantes de estos, como su forma, tamaño o nivel de gris.

- ***Descriptores espectrales o en frecuencia*** → se basan en el análisis del espectro de Fourier mediante la transformada del mismo nombre. Se utilizan principalmente para obtener la direccionalidad y la periodicidad de los patrones de textura.

En la implementación se ha utilizado una serie de descriptores estadísticos para caracterizar la textura de las imágenes.

Los descriptores estadísticos más simples para describir una textura son los momentos de 1º orden del histograma de niveles de gris. Sea *z* una variable aleatoria que indica la intensidad de una imagen discreta y sea $p(z_i),\ i=1,2,\ldots,L$ el correspondiente histograma, donde L es el número de niveles. El momento n-ésimo [22], μ_n de *z* respecto a la media es:

$$\mu_n(z) = \sum_{i=1}^{L} (z_i - m)^n p(z_i) \qquad (9.2)$$

donde *m* es el valor medio de *z* (la intensidad media) y se calcula como:

$$m = \sum_{i=1}^{L} z_i p(z_i) \qquad (9.3)$$

El segundo momento o varianza, $\mu_2(z) = \sigma^2(z)$, es de particular importancia para la descripción de la textura ya que proporciona una medida del contraste del nivel de gris, lo que se puede utilizar para establecer descriptores de suavidad.

El tercer momento o asimetría, $\mu_3(z)$, es una medida de la desviación del histograma, mientras que el cuarto momento o kurtosis, $\mu_4(z)$, mide la monotonía relativa.

El quinto momento así como los siguientes no son tan fácilmente relacionables con el histograma, pero proporcionan una elevada discriminación cuantitativa de la textura.

El descriptor de parámetros estadísticos de textura que se ha implementado incluye los siguientes seis parámetros:

- La ***intensidad media*** *m*.

- El ***contraste medio***, dado por la desviación típica σ.

$$\sigma = \sqrt{\mu_2(z)} = \sqrt{\sigma^2} \qquad (9.4)$$

- El ***coeficiente de asimetría*** γ_1, que mide la desviación del histograma y es igual a 0 para histogramas simétricos, positivo para histogramas desplazados a la izquierda y negativo para los que se encuentran desplazados a la derecha. Se obtiene a través de la asimetría μ_3 como:

$$\gamma_1 = \frac{\mu_3}{\mu_2^{3/2}} = \frac{\mu_3}{\sigma^3} \qquad (9.5)$$

- El ***coeficiente de kurtosis*** γ_2, que mide el aplanamiento del histograma respecto a una distribución normal. Toma valor igual a 0 cuando tiene una distribución normal, valor positivo para una distribución más esbelta que la normal y valor negativo cuando es más aplanada que la normal. Viene dado por el cuarto momento μ_4 como:

$$\gamma_2 = \frac{\mu_4}{\mu_2^2} - 3 = \frac{\mu_4}{\sigma^4} - 3 \qquad (9.6)$$

- La ***uniformidad U***, la cual da información de lo uniforme que es la textura. Esta es máxima cuando todos los niveles de gris son iguales y disminuye cuanto más diferentes sean. Se calcula como:

$$U = \sum_{i=0}^{L-1} p^2(z_i) \qquad (9.7)$$

- La ***suavidad de la intensidad R***, que proporciona información acerca de las variaciones de intensidad. Tiene valor igual a 0 en regiones de intensidad constante y se aproxima a 1 en regiones con intensidad extremas. A menudo la varianza usada en esta medida se normaliza para que esté en el rango [0,1], por lo que se divide por $(L-1)^2$. Se calcula a partir de la varianza de la siguiente forma:

$$R = 1 - \frac{1}{1 + \sigma^2(z)} \qquad (9.8)$$

Implementación

Para una imagen dada de la base de datos, se extraen los seis parámetros anteriormente descritos para cada una de sus componentes RGB y para su componente V del modelo de color HSV. En total se obtienen 24 parámetros para describir la textura en la imagen. El algoritmo consiste únicamente en la implementación de las fórmulas asociadas a los parámetros descritos.

Resultados

Se presentan los seis parámetros de textura descritos para cada una de las componentes de dos imágenes de la base de datos tomadas como ejemplo.

Figura 9.7 – Ejemplos de imágenes de la base de datos

Tabla 9.3 – Parámetros de Textura de las imágenes de ejemplo anteriores

	Paisaje				***Estadio de Fútbol***			
	R	***G***	***B***	***V***	***R***	***G***	***B***	***V***
Media	79,1916	85,2159	101,1840	112,0629	113,9266	115,7787	103,0748	121,9471
Contraste medio	58,2663	60,3309	75,9644	73,3374	84,7967	84,0160	85,3615	86,5689
Coef. Asimetría	1,8039	1,5632	1,2806	1,1519	1,2353	1,2069	1,3699	1,1794
Coef. Kurtosis	1,2028	0,1562	-1,2157	-1,5411	-1,3856	-1,4562	-0,9979	-1,5355
Uniformidad	0,0156	0,0107	0,0075	0,0078	0,0048	0,0045	0,0058	0,0045
Suavidad	0,0028	0,0032	0,0079	0,0069	0,0123	0,0118	0,0126	0,0133

9.2 Extracción de Características Específicas

Con el fin de mejorar los resultados obtenidos tras la clasificación, se extrae a las imágenes un conjunto de parámetros que forman el vector de características específicas. Se denominan características específicas porque intentar dar información, no del contenido general de la imagen, sino de la aparición o ausencia de un determinado concepto en ella. Para su implementación se ha realizado un análisis previo de las imágenes (tanto de las de entrenamiento como las de test), evaluando los rasgos identificativos de los conceptos a identificar y obteniendo así diferentes parámetros que los describan en cierta forma.

No se implementa para todos los conceptos especificados en la tarea, pues extraer información específica de algunos de ellos implica una gran complejidad, como por ejemplo para el concepto caminos. Los conceptos para los que se extraen las características específicas son:

- Interior/Exterior
- Personas
- Día/Noche
- Agua
- Vegetación
- Cielo

A continuación se presentan detalladamente las características extraídas y la información teórica que se considere necesaria para cada uno de los conceptos anteriores.

9.2.1 Parámetros de Interior/Exterior

Analizando las imágenes de la base de datos se observa que las fotografías realizadas en interiores suelen tener una luminosidad más uniforme que las realizadas en exteriores. Posiblemente esto se deba a que se tomen en espacios de menores dimensiones, en los cuales la luz natural o el flash de la cámara que iluminan la escena, lo hacen de forma uniforme prácticamente en toda la imagen.

De forma inversa las fotografías tomadas en exteriores suelen captar espacios de mayores dimensiones, en los cuales pueden aparecen zonas con diferentes iluminaciones, provocando que la luminosidad general de la imagen tenga una forma más irregular.

Tomando como referencia lo expuesto anteriormente se ha decidido que la información de luminancia en las imágenes puede ser relevante para diferenciar entre imágenes de interior y de exterior. Los parámetros que se cree que pueden proporcionar esta información son parámetros que describen la forma del histograma de la componente de luminancia *Y* del espacio de color *YUV*. Se utiliza este modelo de color en vez de otros debido a que se ajusta a los modelos fotográficos y al modelo humano de percepción.

El modelo de color YUV [23], es el empleado actualmente en la radiodifusión de la televisión, fundamentalmente en los estándares de televisión europeos PAL y SECAM. Está compuesto por una componente de luminancia (Y) y dos componentes de crominancia o de color (U,V). Tienen su origen en una recodificación del sistema RGB para responder a una característica de la visión humana, es más sensible a los cambios de luminancia que a las modificaciones en matices y saturaciones. Esta formulación logra una transmisión más eficiente de la señal, al destinar más ancho de banda a la señal de luminancia que a la crominancia, permitiendo su compatibilidad con los sistemas de TV en blanco y negro (otra razón de las razones fundamentales de la época que llevó a su creación). La conversión lineal entre el modelo RGB y el YUV es la siguiente:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0{,}229 & 0{,}587 & 0{,}114 \\ -0{,}147 & -0{,}289 & 0{,}436 \\ 0{,}615 & -0{,}515 & -0{,}100 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad (9.9)$$

Implementación

Se obtiene para cada imagen de la base de datos su componente Y del modelo de color YUV mediante la transformación desde RGB descrita en la Ec. 9.9. Se genera el histograma normalizado de la componente obtenida y se calculan seis parámetros estadísticos que definen la forma del histograma, en concreto los seis parámetros detallados en el apartado 9.1.3 que definían la textura de una imagen. El algoritmo consiste únicamente en la implementación de las fórmulas asociadas a dichos parámetros.

Resultados

Se muestran las componentes de luminancia y los correspondientes histogramas de dos imágenes, una tomada en interior y otra en exterior.

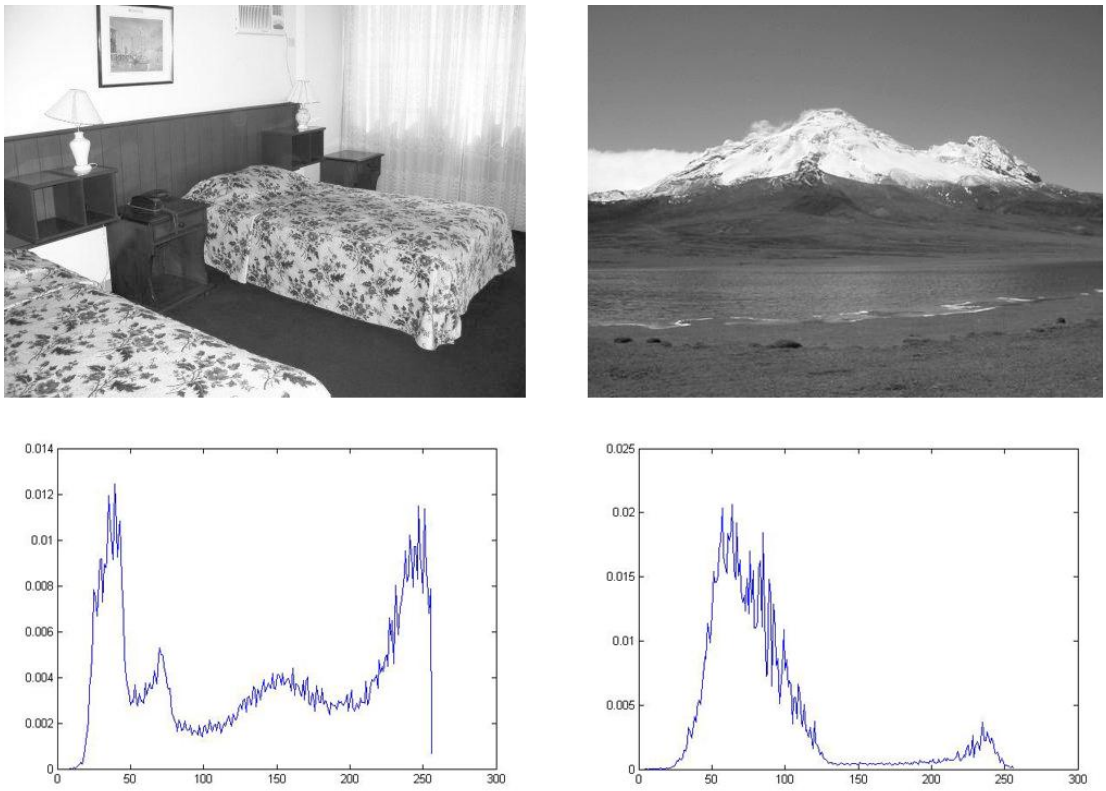

Figura 9.8 – Luminancias y sus correspondientes histogramas de imágenes ejemplo de interior y exterior

Se observa en el histograma de la imagen tomada interior que la luminancia está repartida de forma más uniforme que en la imagen tomada en exterior. En la tabla 9.4 se muestran los parámetros que se obtienen para las imágenes de ejemplo anteriores.

Tabla 9.4 – Parámetros extraídos del histograma de luminancia de las imágenes anteriores

	Imagen Interior	***Imagen Exterior***
Media	140,2962	85,2568
Contraste medio	80,5984	46,3458
Coeficiente de Asimetría	-0,0770	2,0657
Coeficiente de Kurtosis	-1,5161	3,9475
Uniformidad	0,0057	0,0115
Suavidad	0,0100	0,0011

9.2.2 Parámetros de Personas

Con la intención de describir la aparición/ausencia de personas en las imágenes se ha extraído información acerca de la textura de la imagen. En concreto se han utilizado algunos de los descriptores de Tamura.

Las características de Tamura [8] son en total seis: coarseness (no hay una traducción precisa para este término, se sugiere entenderlo como rugosidad), contraste, direccionalidad, parecido de las líneas, regularidad y rugosidad. Sus autores encontraron que las tres primeras son muy importantes, pues se encuentran fuertemente correlacionadas con la percepción humana. Estas tres características son definidas a continuación [5].

Coarseness:

El coarseness proporciona información acerca del tamaño de las texturas en la imagen. Cuanto más alto es el valor del coarseness, más rugosa es la textura. Cuando las texturas tienen un micro-patrón y un macro-patrón el procedimiento de Tamura considera el patrón más grande, y la forma de hacerlo se aplicar operadores de diferente tamaño de la siguiente manera:

Sea una imagen *I(x,y)* , el valor del píxel (*x*,*y*) se calcula como:

1. Para cada punto (*x*,*y*) se calcula el promedio sobre una vecindad definida que debe ser potencia de dos, por ejemplo 2 x 2, 4 x 4, … , 32 x 32:

$$A(x,y) = \frac{1}{2^{2k}} \sum_{i=1}^{2k} \sum_{j=1}^{2k} I(x - 2^{k-1} + i, y - 2^{k-1} + j) \qquad (9.10)$$

2. Se calcula la diferencia entre las vecindades que no se solapen a lados opuestos del punto en las direcciones horizontal y vertical.

$$D_k^h = |A_k(x + 2^{k-1}, y) - A_k(x - 2^{k-1}, y)| \qquad (9.11)$$
$$D_k^v = |A_k(x, y + 2^{k-1}) - A_k(x, y - 2^{k-1})| \qquad (9.12)$$

3. Para cada punto (*x*,*y*) se selecciona el valor con la diferencia más grande.

$$S(x,y) = Max_{d=h,v}\{E_k^d(x,y)\} \;\; con\; k = \{1, \dots, 5\} \qquad (9.13)$$

Este valor se asigna a cada píxel obteniendo el coarseness local de la imagen, lo cual permite calcular una distribución del coarseness en la imagen, es decir, un histograma coarseness que permite medir los tamaños de las texturas.

Contraste:

El contraste en sentido estricto es sinónimo de la calidad de la imagen. Más detalladamente se encuentra influenciado por cuatro factores: el rango dinámico de los niveles de gris, la polarización en los niveles de blanco y negro, la nitidez de los bordes y el periodo de repetición de los patrones. El contraste de Tamura se calcula utilizando la media y la varianza de los valores de intensidad en una vecindad dada de la siguiente manera:

$$F_{CON} = \frac{\sigma}{\alpha_4^z} \;\; con \;\; \alpha_4 = \frac{\mu_4}{\sigma^4} \;\; donde \;\; \mu_4 = \frac{1}{XY} \sum_{x=1}^{X} \sum_{y=1}^{Y} (I(x,y) - \mu)^4 \qquad (9.14)$$

μ_4 es el cuarto momento alrededor de la media μ, σ^2 es la varianza de los valores de gris en la vecindad y *z* ha sido configurada experimentalmente como 1/4.

Direccionalidad:

No la orientación como tal, sino la presencia de dirección en la imagen es interesante. Para calcular la direccionalidad, se calculan las derivadas horizontal y vertical de la imagen. Para ello se filtra la imagen con dos filtros de Prewitt de 3 x 3, uno horizontal y otro vertical.

$$\Delta_H = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{y} \quad \Delta_V = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Figura 9.9 – Filtros de Prewitt utilizados para calcular las derivadas horizontal y vertical

Luego para cada punto (*x*,*y*) se calcula el ángulo de orientación como:

$$\theta = \frac{\pi}{2} + tan^{-1} \frac{\Delta_V(x,y)}{\Delta_H(x,y)} \qquad (9.15)$$

Implementación

En primer lugar para cada imagen se obtiene la componente V del modelo de color HSV, a la cual se le elimina el 10% exterior de la misma, pues se considera que dicha información será redundante para la detección de personas en la imagen porque las personas suelen encontrarse encuadradas en las imágenes y las partes exteriores suelen pertenecer al fondo de las mismas.

A continuación se obtiene para la imagen resultante los parámetros de Tamura descritos, implementando en Matlab los algoritmos que los definen. Para el coarseness se ha utilizado una vecindad de 8 x 8. Debido a que tanto el coarseness como la direccionalidad son calculadas localmente (píxel a píxel), se obtienen a partir de ellas dos parámetros para poder caracterizarlas globalmente, la media y la desviación típica.

Así este algoritmo devuelve un vector con cinco parámetros, los cuales describen las características de Tamura en una imagen. Estos son: media y desviación típica del coarseness, media y desviación típica de la direccionalidad y el contraste.

Entre las funciones empleadas en Matlab cabe destacar:

- ***filter2*** → Filtra la matriz de datos de una imagen con el filtro que se le pasa como parámetro.

Resultados

Para dos imágenes de ejemplo de la base de datos, se muestran los cálculos locales de su coarseness y su direccionalidad.

Figura 9.10 – Imágenes de ejemplo, con presencia y ausencia de personas

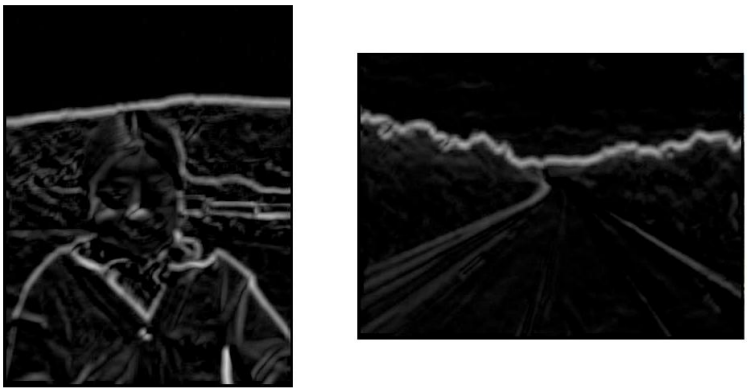

Figura 9.11 – Cálculo local del Coarseness de Tamura de las imágenes de ejemplo

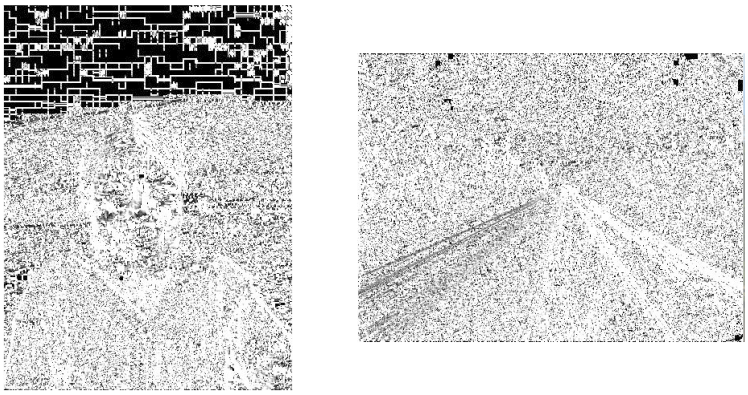

Figura 9.12 – Cálculo local de la Direccionalidad de Tamura de las imágenes de ejemplo

Mediante los cálculos locales del Coarseness y la Direccionalidad se obtienen las medias y la desviaciones que finalmente devuelve el algoritmo como parámetros descriptivos de Tamura. Para las imágenes de ejemplo anteriores, los parámetros que describen las características de Tamura son los siguientes.

Tabla 9.5 – Parámetros de Tamura de las imágenes de ejemplo

	Imagen con Personas	***Imagen sin Personas***
Media del Coarseness	10,7791	6,1502
Desviación del Coarseness	5,8452	2,4522
Contraste	72,7299	73,6716
Media de la Direccionalidad	1,3312	1,5381
Desviación de la Direccionalidad	0,1442	0,00890

9.2.3 Parámetros de Día/Noche

Para proporcionar información relevante sobre si la imagen se ha tomado de día o de noche, se ha extraído un conjunto de parámetros que dan información acerca de las probabilidades de aparición de los píxels en determinados rangos de niveles de gris de dos componentes de la imagen.

La primera componente que se ha utilizado es la componente de luminancia *Y* del modelo YUV, modelo que se encuentra detallado en el apartado 9.2.1. Claramente la luminancia de la imagen será muy diferente para imágenes de día que para imágenes de noche, por lo que en este caso será muy útil.

La segunda componente sobre la que se han extraído parámetros es la componente de amarillo *Y* del modelo de color CMY. El modelo CMY [38], acrónimo de cian, magenta y amarillo (yellow), se basa en la absorción de la luz y se utiliza fundamentalmente en la impresión en colores. El color que presenta un objeto corresponde a la parte de luz que incide sobre éste y que no es absorbida por el objeto. De esta forma es un modelo de color sustractivo (al contrario que el modelo RGB), en la que el negro se compone por la suma de todos ellos y el blanco es la ausencia de color.

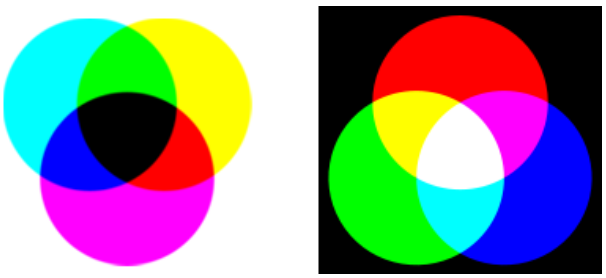

Figura 9.13 – Modelo de color sustractivo (CMY) y modelo de color aditivo (RGB)

El cian es opuesto al rojo, lo que significa que actúa como filtro que absorbe dicho color. Lo mismo ocurre con el magenta que es el opuesto al verde y el amarillo que es el opuesto al azul. Por tanto la conversión entre el modelo RGB y el CMY es muy simple y de la siguiente forma:

$$C = 1 - R \qquad (9.16)$$
$$M = 1 - G \qquad (9.17)$$
$$Y = 1 - B \qquad (9.18)$$

Aunque la componente Y representa el amarillo, proporciona información relevante para la discriminación entre día y noche, pues al ser un modelo sustractivo tendrá valor máximo para valores de negro y mínimo para valores claros. Se observa que prácticamente es la inversa de la luminancia de la imagen.

Con el fin de extraer información relevante de las componentes anteriormente descritas, particularmente se me ha ocurrido calcular la probabilidad de aparición de los píxels en 3 determinados rangos de los niveles de gris. Así se han definido estas probabilidades como:

- ***Probabilidad de oscuridad*** → probabilidad de aparición de los píxels en el rango [0,54]
- ***Probabilidad media*** → probabilidad de aparición de los píxels en el rango [55,199]
- ***Probabilidad de claridad*** → probabilidad de aparición de los píxels en el rango [200,255]

Como nota hay que destacar que para la componente de amarillo, la probabilidad de oscuridad se calcula en el rango de la probabilidad de claridad definida anteriormente y viceversa. Esto se debe a como se ha explicado el modelo CMY es un modelo de color sustractivo.

Implementación

Dada una imagen, se obtiene la componente *Y* del modelo CMY y la componente *Y* del modelo YUV que a partir de ahora se denominará como *L* (pues representa la luminancia de la imagen) para diferenciarlas. Para cada una de las componentes se obtienen 6 parámetros. Tres de ellos se extraen sobre toda la imagen, para obtener información global de la luminosidad. Los otros tres se obtienen sobre la parte superior de la imagen, ya que tras un análisis de las imágenes de la base de datos, esta zona seguramente pertenecerá a cielo y proporcionará así mayor información sobre si es de día o de noche. El algoritmo implementado para extraer los parámetros es similar para cada componente, y es de la siguiente manera:

1. Se obtiene el histograma normalizado
2. Se calculan 3 parámetros, que son las probabilidades descritas anteriormente. Estas se calculan como la suma acumulada de los valores del histograma en sus respectivos rangos.
3. Se recorta la imagen, quedándose con el 10% superior de la misma.
4. Se obtiene el histograma normalizado de la imagen recortada.
5. Se calculan otros 3 parámetros, de nuevo las probabilidades anteriores, pero en este caso sobre el histograma de la imagen recortada.

Finalmente se devuelve un vector con 12 parámetros, las seis probabilidades que caracterizan la componente *L* y las seis que caracterizan la componente *Y*.

Para su realización en Matlab se han utilizado comandos básicos, solo cabe destacar la utilización de la siguiente función.

- ***imhist*** → Genera el histograma de una imagen en niveles de gris que recibe como parámetro.

Resultados

En la siguiente figura se muestran dos imágenes de ejemplo de la base de datos, una tomada de día y otra tomada de noche, y sus correspondientes componentes *L* e *Y*.

Figura 9.14 – Componentes L e Y (de izquierda a derecha) de dos imágenes ejemplo de la base de datos

Tal y como se ha comentado, se observa que la componente *Y* es prácticamente la inversa de la componente *L*, lo que reforzará la calidad de los parámetros en las imágenes en las que estos sean significativos.

En la tabla a continuación se muestran los parámetros obtenidos para las imágenes anteriores. Se observa que las imágenes tomadas de día tienen mayores probabilidades en el rango medio y el de claridad, mientras que las imágenes tomadas de noche tienen mayores valores en el rango de oscuridad.

Tabla 9.6 – Valores de los parámetros de "Día/Noche" extraídos a las imágenes anteriores

	Imagen de Día	***Imagen de Noche***
Prob. Oscuridad de L	10%	76%
Prob. Media de L	89%	24%
Prob. Claridad de L	1%	0%
Prob. Oscuridad del 10% superior de L	1%	86%
Prob. Media del 10% superior de L	99%	14%
Prob. Claridad del 10% superior de L	0%	0%
Prob. Oscuridad de Y	12%	92%
Prob. Media de Y	45%	7%
Prob. Claridad de Y	43%	1%
Prob. Oscuridad del 10% superior de Y	1%	98%
Prob. Media del 10% superior de Y	9%	2%
Prob. Claridad del 10% superior de Y	91%	0%

9.2.4 Parámetros de Agua

El agua [17] a pesar de ser incolora, cuando aparece en grandes cantidades toma diversos colores en tonos azules. Esto se debe a que cuando la luz incide sobre pequeñas cantidades de agua, todas las longitudes de onda lo atraviesan, haciéndolo incoloro (pues el color lo da la luz reflejada). Mientras que cuando la luz incide sobre cantidades mayores de agua, a ésta le cuesta más atravesarla, se absorbe selectivamente la parte roja del espectro y se reflejan las longitudes de onda de la parte azul, tomando el agua una cierta tonalidad azul a nuestra vista. Así cuanto mayor sea la cantidad de agua acumulada más intenso será el azul del agua, como ocurre por ejemplo en el mar.

Otra característica del agua en cuanto a la forma en la que nosotros lo observamos, es que presenta una textura bastante uniforme y en la que la repetición de sus patrones es de forma estadística.

Teniendo en cuenta que tras analizar las imágenes de la base de datos se observa que en la mayoría de ellas en las que aparece el concepto "*agua*" este lo hace en la zona inferior de la imagen. Y tomando los dos conceptos anteriores sobre la forma en la que visualizamos el agua, se extraen una serie de parámetros que intentan describir la presencia o no de agua en la imagen mediante la textura de la parte azul de la misma.

Implementación

Dada una imagen de la base de datos, se obtiene la componente de azul *B* del modelo RGB. A esta se le recorta el 40% superior, quedándose con el resto de la imagen. Se calcula el histograma normalizado de la imagen resultante y se extraen los 6 parámetros estadísticos de textura descritos en el apartado 9.1.3.

Para cada imagen se devuelve un vector con 6 parámetros estadísticos de textura del 60% inferior de la componente *B*, en concreto los siguientes parámetros: la media, el contraste medio, el coeficiente de asimetría, el coeficiente de kurtosis, la uniformidad y la suavidad.

En cuanto a la implementación en Matlab, esta consiste únicamente en la implementación de las fórmulas asociadas a los parámetros estadísticos que se extraen.

Resultados

Se muestran dos imágenes, uno con y otra sin la aparición de agua, y sus respectivas componentes de azul recortadas sobre la que se extraen los parámetros de textura. Se observa tal y como se ha comentado que el agua toma valores altos en la componente de azul.

Figura 9.15 – Imágenes de ejemplo y sus componentes B sobre la que se extraen los parámetros de agua

En la tabla a continuación se muestran los parámetros extraídos para las imágenes de ejemplo anteriores.

Tabla 9.7 – Valores de los parámetros de "Agua" extraídos a las imágenes de ejemplo anteriores

	Imagen con Agua	***Imagen sin Agua***
Media	137,1640	98,6916
Contraste Medio	28,5820	41,9312
Coeficiente de Asimetría	0,2547	0,2675
Coeficiente de Kurtosis	0,8380	0,3886
Uniformidad	0,0125	0,0075
Suavidad	0,0002	0,0007

Evaluando estos parámetros se puede concluir que las imágenes con agua:

- Tendrán valores mayores en la media, pues el nivel de azul es mayor.
- Menor contraste medio, pues a su vez la varianza será menor.
- Será más uniforme, puesto que los valores de gris de la componente B serán más parecidos.
- El valor de la suavidad será menor pues la textura del agua es más suave, ya que tiene valores de intensidad más constantes.

Aunque estas conclusiones dependerán de la forma en la que aparezca el agua en las imágenes.

9.2.5 Parámetros de Vegetación

Para la extracción de parámetros que describan la aparición/ausencia de vegetación en la imagen se ha realizado una conversión del modelo de color RGB al modelo C1C2C3 propuesto en [4] y que a continuación se detalla.

El modelo de color C1C2C3 es una modificación del sistema de color I1 I2 I3 propuesto por Ohta, *et al* [21]. Las componentes de este sistema modificado, C1, C2 y C3 se obtuvieron de forma experimental y permiten segmentar las imágenes digitales de color en regiones homogéneas de color mediante umbrales.

El sistema de color I1 I2 I3 propuesto por Yuichi Ohta se obtuvo de forma experimental, analizando los resultados de ocho imágenes a color. Ohta calculó los eigenvectores de cada una de las ocho imágenes a color, de donde surgieron las tres características ortogonales de color, también denominados atributos de color (I1 I2 e I3). Estas, de acuerdo a lo escrito por Ohta son componentes fundamentales para representar información de color. La transformación RGB al sistema I1 I2 I3 en forma matricial es la siguiente:

$$\begin{bmatrix} I1 \\ I2 \\ I3 \end{bmatrix} = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/2 & 0 & -1/2 \\ -1/4 & 1/2 & -1/4 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad (9.19)$$

Este sistema es equivalente a transformar el modelo RGB al modelo YUV representado en la Ec. (9.9) pero utilizando diferentes coeficientes en la transformación.

La componente I1 se obtiene del promedio de los tres canales rojo, verde y azul, conteniendo así información de los 3 canales. Por esto es similar a la componente Y del modelo YUV que tiene información de la luminancia de la imagen. Al hacer diferencias por pares de canales R – G, G – B y B – R, en los histogramas resultantes de cada diferencia se observa una distinción clara entre las diferentes zonas homogéneas de color. Estas diferencias permiten hacer una segmentación más adecuada mediante umbrales. Por esto se propuso el sistema modificado C1C2C3 en el cual sus componentes se obtienen mediante el sistema RGB como:

$$C1 = \frac{(R - G)}{2} \qquad (9.20)$$

$$C2 = \frac{(G - B)}{2} \qquad (9.21)$$

$$C3 = \frac{(B - R)}{2} \qquad (9.22)$$

Las componentes C1, C2 y C3 son ortogonales. Dependiendo de los valores en cada canal RGB los valores resultantes para las componentes transformadas pueden ser negativos, para evitarlo se hace un mapeo lineal de la forma:

$$y = x + 127 \qquad (9.23)$$

Con la transformación propuesta el valor del píxel se mapea a un intervalo [0,255], es decir los valores resultantes son todos positivos.

En la práctica la componente C1 muestra las zonas homogéneas con valores de rojo. Igual ocurre con la componente C2, que muestra las zonas homogéneas con valores de verde y de forma similar con la componente C3, mostrando las zonas homogéneas con de azul.

En concreto para la detección de la vegetación interesa la componente C2, que mostrará las zonas homogéneas con valores de verde. A partir de estas zonas se extraerán parámetros de textura que describan la aparición/ausencia de vegetación en la imagen.

Implementación

Dada una imagen se obtiene mediante sus componentes RGB la componente C2 del sistema de color C1C2C3 tal y como se expone en [4] y se describe anteriormente.

No se realiza el mapeo lineal de la Ec. (9.23) debido a que Matlab trata los valores de los píxels de las imágenes con formato *"uint8"*. Este formato establece los números con 8 bits y sin signo, por lo que siempre son positivos y se encuentran en el rango [0,255]. De esta forma al obtener la componente C2, los valores que serían negativos se igualan a cero. Así se obtiene una mayor información sobre el verde en la imagen.

Como se observa en la Ec. (9.21), cuando el valor de azul (*B*) sea mayor que el de verde (*G*) se obtendrán valores negativos, y por tanto al utilizar el formato *"uint8"* serán igual a cero. Esto conlleva que los valores que se obtengan en la componente C2 sean aquellos que tienen gran cantidad de la componente de verde, obteniendo así una imagen con las zonas homogéneas donde los valores eran altos en la componente verde. El único problema es que la imagen se encontrará ahora en el rango [0,127] y al mostrarla se observarán los píxels con alto contenido de verde como grises intermedios, pero la información que contiene seguirá siendo la misma. Este problema se tendrá en cuenta a la hora de obtener los parámetros de textura.

Una vez obtenida la componente C2, se genera su histograma normalizado y se extraen los 6 parámetros estadísticos de textura descritos en el apartado 9.1.3. Teniendo en cuenta el problema que se comentaba anteriormente, los parámetros de textura se deberían obtener sobre el rango [0,127] del histograma, que es donde la imagen tendrá valores. Pero como además el objetivo es obtener información sobre zonas de verde, no se tendrá en cuenta para el cálculo de los parámetros de textura el nivel 0, pues corresponderá con todo aquel píxel que no tenga un valor alto de la componente de verde (*G*).

Por tanto se devolverá para cada imagen un vector con los siguientes seis parámetros de textura: media, contraste medio, coeficiente de asimetría, coeficiente de kurtosis, uniformidad y suavidad. Siendo estos obtenidos del histograma de la componente C2 en el rango [1,127].

En cuanto a la implementación en Matlab, esta consiste únicamente en la implementación de la fórmula asociada a la componente C2 y de las fórmulas asociadas a los parámetros estadísticos que se extraen.

Resultados

Para dos imágenes de ejemplo de la base de datos, una con la aparición de vegetación y otra con ausencia de ella, se muestran las componentes C2 y los histogramas sobre los que se extraen los parámetros de textura.

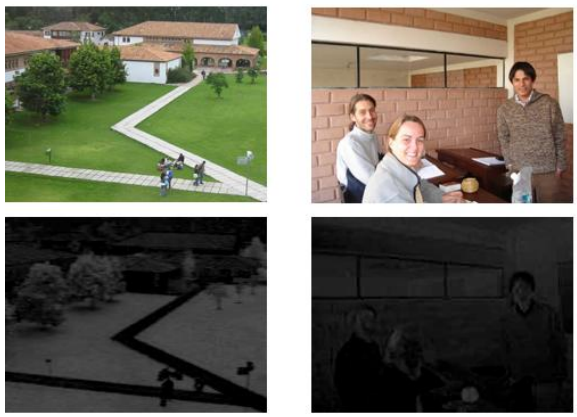

Figura 9.16 – Imágenes de ejemplo y sus componentes C2 del modelo C1C2C3

Se observa que en la imagen con vegetación, las zonas en las que se encuentra la vegetación aparecen en la componente C2 con valores de gris intermedio. A su vez, la componente C2 de la imagen en la que se ausenta la vegetación, tiene prácticamente en su totalidad valores en torno a cero. En los histogramas siguientes se muestra esta conclusión con más detalle.

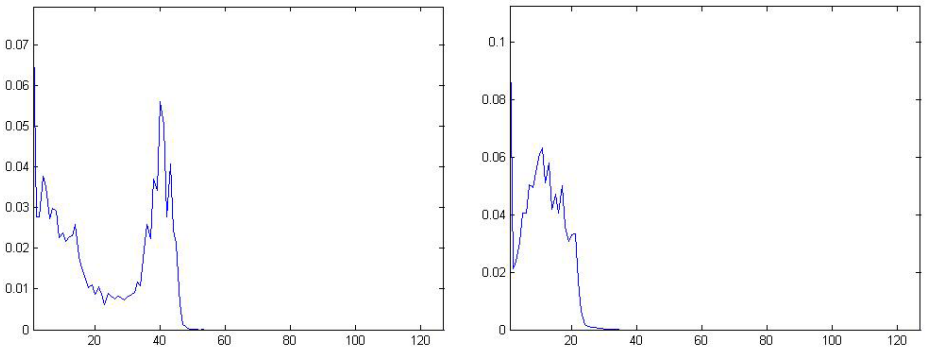

Figura 9.17 – Histogramas de las componentes C2 de las imágenes tomadas de ejemplo

Las diferencias en los histogramas de las imágenes con presencia/ausencia de vegetación se plasman en el vector con los parámetros de textura que devuelve este bloque.

Tabla 9.8 – Valores de los parámetros de "Vegetación" extraídos a las imágenes de ejemplo

	Imagen con Vegetación	***Imagen sin Vegetación***
Media	21,2909	9,7751
Contraste Medio	14,5693	5,2454
Coeficiente de Asimetría	0,3290	0,8470
Coeficiente de Kurtosis	-1,4852	-0,2174
Uniformidad	0,0254	0,0399
Suavidad	0	0,0001

9.2.6 Parámetros de Cielo

Una vez más se ha analizado la forma en la que aparece el cielo en las imágenes de la base de datos con el fin de extraer un vector de parámetros que describa la aparición/ausencia de este concepto en cada una de las imágenes. Con este análisis se ha llegado a la conclusión de que el cielo aparece en las escenas en la parte superior de las mismas, como era de prever. Además se ha observado que el cielo suele estar formado por zonas de luminosidad uniformes, independientemente de aparecer soleado, parcialmente nublado o nublado.

Teniendo en cuenta estos aspectos se han obtenido cuatro parámetros que proporcionan información acerca de la varianza de la luminosidad en la parte superior de la imagen. Con ellos se pretende diferenciar la presencia/ausencia de cielo en las imágenes.

Implementación

Dada una imagen se obtiene la componente *V* del modelo de color HSV, tal y como se describe en la Ec. (3.8). Se ha decidido por trabajar con esta componente porque es la que mejor detalla la luminosidad de una imagen [39]. A la componente de luminosidad obtenida, se le recorta la zona que para este caso es de interés, en concreto el 10% superior de la imagen.

Como primer parámetro, de los cuatro que se extraen, se obtiene la varianza de los píxels de la parte superior de la componente V.

En algunas de las imágenes se observa que el cielo no aparece en toda la zona superior, sino en determinadas zonas porque se encuentra oculto por diferentes objetos, como montañas, edificios u otros. Esto provocará que el valor de la varianza anterior sea muy alto y comparable con las imágenes en las que no aparece cielo. Para solucionar este problema se obtienen dos parámetros más.

Se divide la parte superior de la imagen en cinco cuadros de igual tamaño (si la imagen no es divisible por cinco se reajusta su tamaño para que así lo sea). Se calculan las varianzas de cada uno de los cuadros y los dos parámetros que serán devueltos serán las dos mínimas varianzas obtenidas, véase la figura 9.19. Así se soluciona el problema de poder detectar el cielo que pueda aparecer solo en determinadas zonas superiores de la imagen y no en toda la parte superior en general.

Observando los histogramas de la parte superior de la imagen de la componente de luminosidad, se ha llegado a la conclusión que las imágenes en las que aparece cielo, el histograma se encuentra más concentrado en torno a su valor máximo que en las imágenes en las que no hay cielo. Así se obtiene un parámetro que mide la dispersión del histograma y se ha definido como ***"coeficiente de dispersión"***. Este calcula el número de niveles del histograma que tienen un valor mayor a un determinado umbral, en concreto se escogido como umbral el 10% del valor máximo del histograma.

Por lo tanto se devuelve un vector con los parámetros de "cielo", que son los cuatro definidos anteriormente:

- La varianza de la parte superior de la componente *V* de la imagen.
- Las dos mínimas varianzas de 5 sub-cuadros de la zona superior de la componente *V*.
- El coeficiente de dispersión con un umbral del 10% del histograma de la parte superior de la componente *V*.

Para la implementación del código en Matlab se han utilizado comandos básicos, solo cabe destacar la utilización de la siguientes funciones.

- ***imresize*** → Reajusta el tamaño de una imagen al tamaño que se le pase como parámetro.
- ***imhist*** → Genera el histograma de una imagen en niveles de gris que recibe como parámetro.

Resultados

Se muestran los parámetros obtenidos para dos imágenes tomadas de ejemplo de la base de datos, una con la presencia de cielo y otra con ausencia del mismo. En la figura 9.18 claramente se observa que la varianza será menor en imágenes con cielo que en imágenes sin él.

Figura 9.18 – Imágenes de ejemplo, una con presencia y otra con ausencia de cielo

Figura 9.19 – Partes superiores de la componente V y los 2 sub-bloques con las mínimas varianzas

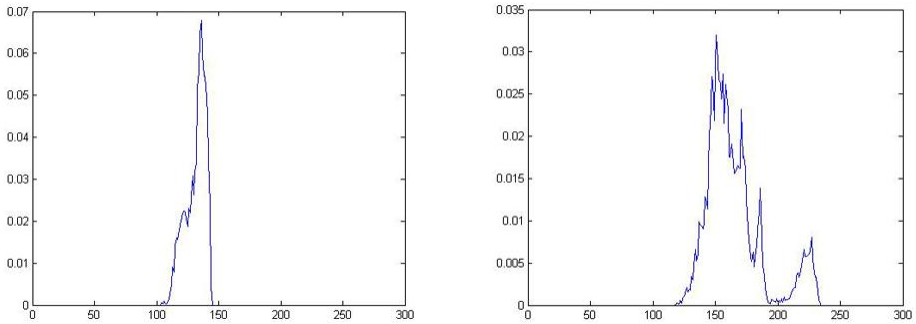

Figura 9.20 – Histogramas de las partes superiores de la componente V

Tal y como se ha comentado las imágenes con cielo presentan un histograma más concentrado en torno a su valor máximo, mientras que el resto presenta un histograma más disperso. En la tabla 9.9 se observa esta diferencia en el coeficiente de dispersión.

Tabla 9.9 – Parámetros de "cielo" extraídos a las imágenes de ejemplo anteriores

	Imagen con Cielo	***Imagen sin Cielo***
Varianza	$8{,}773 \cdot 10^{-10}$	$2{,}0542 \cdot 10^{-5}$
Varianza Mínima 1	$1{,}4439 \cdot 10^{-10}$	$1{,}5520 \cdot 10^{-6}$
Varianza Mínima 2	$2{,}6029 \cdot 10^{-10}$	$1{,}6731 \cdot 10^{-6}$
Coeficiente de Dispersión	31	75

9.3 Resumen

En la etapa de extracción de características se extrae un vector de características generales y un vector de características específicas. A continuación se muestra un resumen de los descriptores que contiene cada vector y la cantidad de parámetros o longitud de dichos vectores.

- **Vector de características generales (59 parámetros):**
  - Colores Dominantes (20)
  - Cuantificación de color (15)
  - Parámetros estadísticos de textura (24)

- **Vector de características específicas (39 parámetros):**
  - Parámetros de Exterior/Interior (6)
  - Parámetros de Personas (5)
  - Parámetros de Día/Noche (12)
  - Parámetros de Agua (6)
  - Parámetros de Vegetación (6)
  - Parámetros de Cielo (4)

10. Clasificación

10.1 Introducción

Todos los algoritmos de aprendizaje supervisado podrían haberse utilizado para la implementación de la clasificación en este proyecto. Se ha decidido implementar el algoritmo K-NN (K-Nearest Neighbours) debido a su simplicidad, lo que le hace uno de los más utilizados.

10.2 Clasificador K-NN

La idea básica sobre la que se fundamenta este algoritmo es que un nuevo caso se va a clasificar en la clase más frecuente a la que pertenezcan sus *K* vecinos más cercanos. Se fundamenta en una idea muy simple e intuitiva, lo que unido a su fácil implementación hace que sea un algoritmo de clasificación muy extendido [20].

10.2.1 Algoritmo K-NN básico

Se dispone de *N* casos (X_n), caracterizados por *n* variables o parámetros, $x_1, x_2, \dots, x_n$ y cuya clasificación es conocida como C_i, denotados como $(X_1, C_1), \dots, (X_N, C_N)$. Donde $X_i = (x_{i,1} \dots x_{i,n})$ para todo $i = 1, \dots, N$ y $C_i \in \{c^1, \dots, c^m\}$ siendo *m* las posibles clases de la clasificación.

Para un nuevo caso que se pretenda clasificar, denotado como $X = (x_1, \dots, x_n)$, se realiza el siguiente algoritmo.

1. Se calculan las distancias (d_i) de todos los casos ya clasificados al nuevo caso *X*.

$$D_x = (d_1, d_2, \dots, d_N) \ \ donde \ \ d_i = d(X_i, X) \qquad (10.1)$$

   La distancia que se utiliza típicamente es la distancia euclídea, pues nos encontramos en un espacio n-dimensional, y se calcula como:

$$d(X_i, X) = \sqrt{(X_{i,1} - X_1)^2 + (X_{i,2} - X_2)^2 + \cdots + (X_{i,n} - X_n)^2} = \sqrt{\sum_{j=1}^{n} (X_{i,j} - X_j)^2} \qquad (10.2)$$

2. Se obtienen las *K* menores distancias D_x^K y los *K* casos conocidos que las generan X^K.

3. Finalmente se clasifica el nuevo caso *X* a la clase más frecuente de los X^K.

En la figura 10.1 se muestra un ejemplo de clasificador K-NN con 25 casos ya clasificados en dos posibles valores (*m=2*), en un espacio bidimensional (con dos variables x_1 y x_2, *n=2*) y con *K=3*.

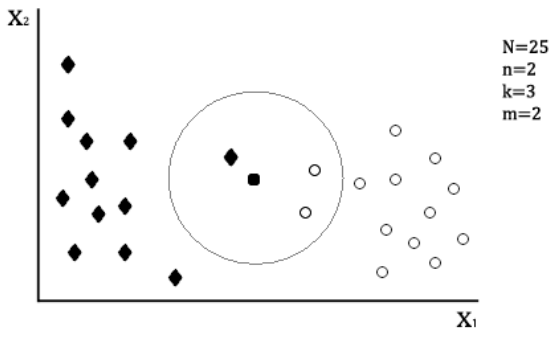

Figura 10.1 – Ejemplo de aplicación del algoritmo K-NN básico con K=3

Se observa que de los 3 casos ya clasificados más cercanos al nuevo caso a clasificar *X*, (representado por ●), dos de ellos pertenecen a la clase ○ y el otro pertenece a la clase ♦. Así el clasificador 3-NN predice la clase ○ para el nuevo caso.

En la figura 10.2 se muestra el mismo ejemplo pero con *K=1*. Nótese que ahora el caso más cercano a *X* pertenece a la clase ♦, por lo que el nuevo caso se asigna a la clase ♦.

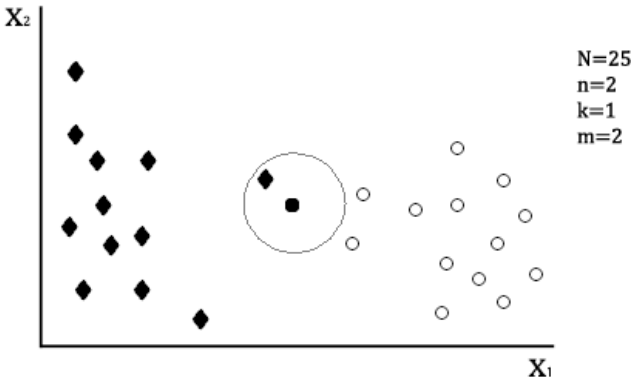

Figura 10.2 – Ejemplo de aplicación del algoritmo K-NN básico con K=1

Cabe destacar que la fase de entrenamiento consiste en almacenar los vectores característicos y las etiquetas de los casos de entrenamiento, mientras que es en la fase de evaluación o clasificación en la que se implementa el algoritmo anterior al conjunto de casos de test.

En caso de que se produzca un empate entre dos o más clases, conviene definir una regla para su ruptura. En los casos en los que las clases solo pueden tener dos valores se suele poner un valor de *K* impar para que no se produzca el empate. Para clases con más valores algunos ejemplos de reglas pueden ser: seleccionar la clase a la que pertenezca el vecino más próximo, seleccionar la clase con distancia menor, añadir nuevos vecinos más cercanos hasta que desaparezca el empate, etc.

Otra cuestión muy importante es la elección del valor de *K*. Está comprobado que el porcentaje de casos bien clasificados no depende linealmente del valor de esta variable. Sin embargo en casos similares a los aquí tratados se observa que los mejores resultados se obtienen para valores de *K* comprendidos entre 3 y 7 (véase la figura 10.3).

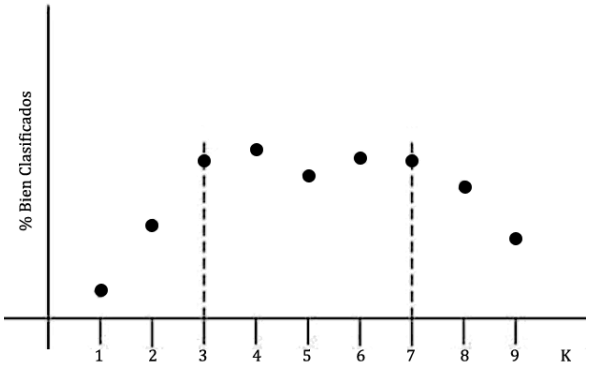

Figura 10.3 – Ejemplo de la no linealidad del porcentaje de casos bien clasificados en función de K

A continuación se describen algunas variantes del algoritmo K-NN básico.

10.2.2 K-NN con rechazo

La idea fundamental de esta variante es la de que para clasificar se deben tener ciertas garantías. Por ello puede ocurrir que un caso quede sin clasificar debido a que no se satisfacen las garantías determinadas.

Un ejemplo de este tipo podría ser que si *K=10* y *m=2* se establezca, para clasificar a una de las dos clases, que el número de vecinos más cercanos de la clase a clasificar sea mayor de un umbral, por ejemplo 6. Si esto no ocurriera el concepto quedaría sin clasificar.

10.2.3 K-NN con distancia mínima

En esta variante en primer lugar se selecciona un caso por clase, normalmente el más cercano al baricentro de todos los elementos. Esto produce una reducción de los casos conocidos de *N* a *m*. A continuación se asigna el nuevo caso a la clase cuyo representante este más cercano. Puede verse como un clasificador 1-NN aplicado a un conjunto de *m* casos (uno por cada clase).

Su coste computacional es bastante menor pero su efectividad está condicionada a la homogeneidad de las clases, cuanto más homogéneas sean más efectivo será el algoritmo.

10.2.3 K-NN con pesado de casos seleccionados

Se basa en la idea de que los *K* casos seleccionados no se contabilicen de igual forma, sino que se tenga en cuenta la distancia a la que se encuentran del caso a clasificar. Un ejemplo sería añadir un sistema de pesos en el que cada caso seleccionado tenga un peso inversamente proporcional a la distancia del nuevo caso.

10.2.3 K-NN con pesado de variables

En los algoritmos anteriores las distancias entre los casos conocidos y el nuevo caso se calculan por medio de la distancia euclídea dando el mismo peso a todas las variables. Esta variante propone dar diferentes pesos a las variables a la hora de calcular las distancias, ya que a veces existen variables que son irrelevantes para determinadas clases. Con este algoritmo las distancias entre los casos conocidos *Xi* y el caso nuevo *X* se calcularían introduciendo un peso w_j asignado a cada variable de *X*.

$$d(X_i, X) = \sqrt{\sum_{j=1}^{n} w_j \left(X_{i,j} - X_j\right)^2} \qquad (10.3)$$

10.3 Implementación

Anteriormente ya se ha comentado el porqué de la utilización del clasificador K-NN, su simplicidad, pero cabe destacar que aparte de esta característica, este algoritmo suele ser muy eficiente y proporcionar buenos resultados en tareas de reconocimiento de imágenes.

La etapa de entrenamiento del clasificador consiste únicamente en guardar en memoria los vectores de características generales y específicas del conjunto de imágenes de entrenamiento. Es en realidad en la etapa de test en la que se realiza la clasificación de las imágenes y para la cual se utilizará el conjunto de imágenes de test dado en la tarea.

Con el objetivo de evaluar si para tareas similares a las tratadas en este proyecto es más eficiente clasificar las imágenes mediante un gran número de características generales (extraídas sin tener en cuenta los objetos o conceptos a identificar) o mediante un número de características específicas considerablemente menor (extraídas de acuerdo a los conceptos u objetos a identificar en las imágenes) se han implementado los siguientes 3 métodos, basados todos ellos en el algoritmo K-NN. Cabe destacar que para la implementación en Matlab de cada uno de ellos se han utilizado únicamente comandos básicos.

10.3.1 Clasificador K-NN General

Se implementa un clasificador K-NN básico para cada uno de los 13 conceptos a identificar por parte del clasificador tal y como se describe a continuación.

Dada una imagen de test a clasificar, junto con su correspondiente vector de características generales extraído en la etapa anterior del sistema, se calculan las distancias euclídeas entre dicho vector y los vectores de características generales de las imágenes de entrenamiento guardados en memoria. Una vez calculadas todas las distancias se obtienen las *K* menores y las imágenes que las generan. Como es conocida la aparición/ausencia de los conceptos en las imágenes de entrenamiento, se clasificará la imagen con la aparición de cada concepto si en la mayoría de las *K* imágenes aparece dicho concepto. De forma inversa se clasificará con la ausencia del mismo si en la mayoría de las *K* imágenes se ausenta el concepto a determinar.

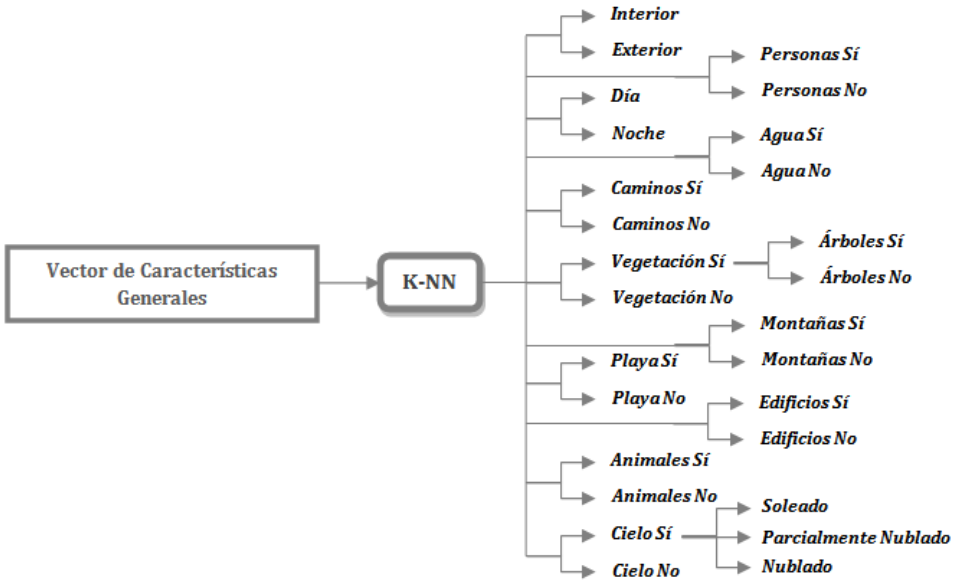

Figura 10.4 – Diagrama de bloques del Clasificador K-NN General

Como ejemplo se muestra la clasificación obtenida para una imágenes de test y las *K* imágenes que proporcionan las mínimas distancias entre los vectores de características generales con el método descrito anteriormente y con *K=3*.

Figura 10.5 – Imagen de ejemplo y las K=3 imágenes más cercanas con el clasificador K-NN General.

En la tabla a continuación se muestra la anotación real de la imagen de ejemplo a clasificar y la clasificación que proporciona este método.

Tabla 10.1 – Anotación real y clasificación utilizando el clasificador K-NN General con K=3

	Exterior/Interior	***Personas***	***Día/Noche***	***Agua***	***Caminos***	***Vegetación***
Anotación real	*Exterior*	*No*	*Día*	*No*	*No*	*Sí*
Clasificación	*Exterior*	*No*	*Día*	*No*	*Sí*	*Sí*

	Árboles	***Montañas***	***Playa***	***Edificios***	***Animales***	***Cielo***	***Tipo de Cielo***
Anotación real	*Sí*	*Sí*	*No*	*Sí*	*No*	*Sí*	*Nublado*
Clasificación	*Sí*	*No*	*No*	*Sí*	*No*	*Sí*	*Nublado*

10.3.2 Clasificador K-NN Específico

Se implementa únicamente para los seis conceptos sobre los cuales se obtuvo anteriormente una serie de características específicas que permitieran identificarlos, son los siguientes: *interior/exterior*, *personas, día/noche*, *agua*, *vegetación* y *cielo*.

El algoritmo utilizado sigue siendo el K-NN básico y es similar al del método anterior, la única diferencia es que en vez de un clasificador para todos los conceptos se implementan seis clasificadores independientes (uno para cada concepto de los que aquí se evalúan). Esto se debe a que para determinar la aparición/ausencia de cada uno de los seis conceptos en las imágenes se utiliza un conjunto de características diferentes, véase la figura 10.5. Por ejemplo para el concepto "*personas*" se utilizan los parámetros específicos de personas (apartado 9.2.2) contenidos en el vector de características específicas.

En conclusión, este método para cada concepto utiliza un clasificador K-NN básico que obtiene las *K* imágenes más cercanas, y por tanto genera la clasificación correspondiente, mediante los parámetros extraídos de forma específica para dicho concepto.

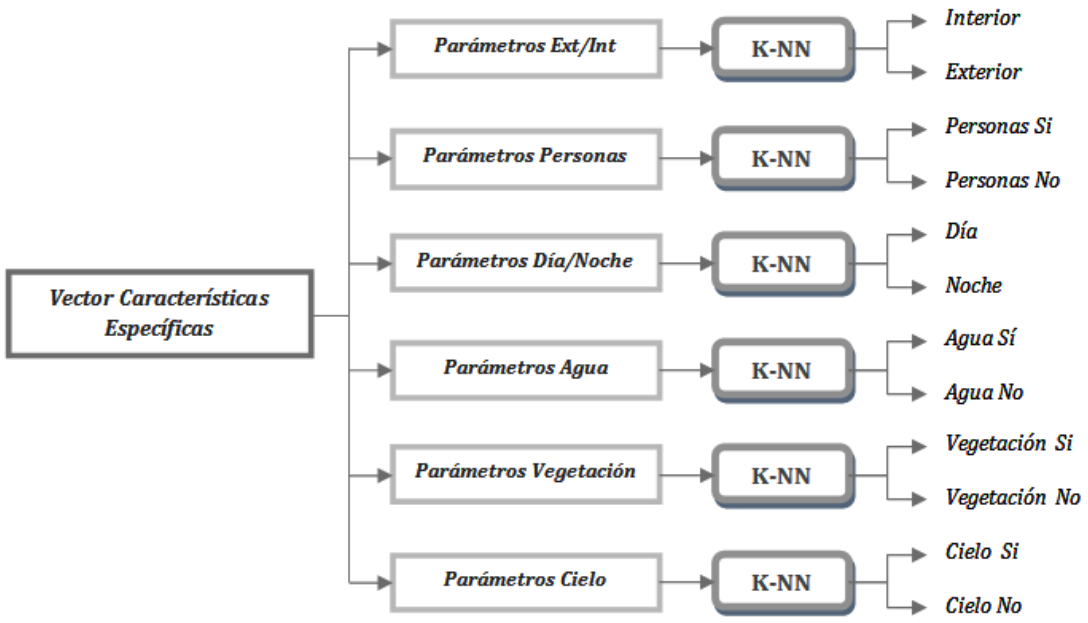

Figura 10.6 – Diagrama de bloques del Clasificador K-NN Específico

A modo de ejemplo se muestra para la imagen de ejemplo de test las *K=3* imágenes más cercanas obtenidas con el ***Clasificador K-NN Específico*** para cada uno de los seis conceptos evaluados.

Figura 10.7 – K=3 imágenes más cercanas por concepto con el K-NN Específico

En la tabla siguiente se observa la anotación real de la imagen utilizada para este ejemplo y la clasificación de la misma que proporciona este tipo de clasificador.

Tabla 10.2 – Anotación real y clasificación utilizando el clasificador K-NN Específico con K=3

	Exterior/Interior	***Personas***	***Día/Noche***	***Agua***	***Vegetación***	***Cielo***
Anotación real	*Exterior*	*No*	*Día*	*No*	*Sí*	*Sí*
Clasificación	*Exterior*	*No*	*Día*	*No*	*Sí*	*Sí*

10.3.3 Clasificador K-NN Combinado

El tercer y último método empleado para realizar la clasificación es una combinación de los anteriores, ya que utiliza tanto los vectores de características generales como los de características específicas extraídos a las imágenes.

Se implementa para los 13 conceptos propuestos en la tarea *VCDT*, pero en realidad solo es combinado para los seis sobre los cuales se extraen las características específicas. Para el resto solo se dispone de características generales, por lo que se implementan tal y como se describe en el primer método (***Clasificador K-NN General***). Estos se llevan a cabo debido a que sus resultados pueden variar en función de las restricciones que la tarea establece en la jerarquía de la figura 6.1.

Para los conceptos sobre los que se extraen las características específicas: *interior/exterior*, *personas*, *día/noche*, *agua*, *vegetación* y *cielo*; se realiza un clasificador K-NN básico independiente para cada uno de ellos, que recibe como entrada la combinación de los vectores de características generales y los parámetros específicos del concepto a determinar. Es decir, para cada concepto de los anteriores las

distancias euclídeas entre la imagen de test a clasificar y las imágenes de entrenamiento se calculan como la suma de las distancias euclídeas de los vectores de características generales y las distancias euclídeas de los parámetros específicos de dicho concepto. Una vez obtenidas las distancias se procede normalmente: se obtienen las *K* mínimas distancias, a continuación las *K* imágenes de entrenamiento que las proporcionan y finalmente se clasifica a la clase mayoritaria de dichas imágenes.

En conclusión, para cada uno de los seis conceptos sobre los que se han extraído las características específicas se implementa un clasificador K-NN básico que utiliza para la clasificación la combinación de las características generales y específicas. Mientras que para el resto de conceptos se implementa un clasificador K-NN básico que utiliza solo las características generales.

De nuevo se muestra para la imagen de ejemplo las *K=3* imágenes más cercanas para cada uno de los conceptos en los que se implementa realmente el K-NN combinado. Para el resto de conceptos, al implementarse únicamente con las características generales, las imágenes más cercanas serán obviamente las mismas que las de la figura 10.5. También se muestra en la tabla a continuación la clasificación real de la imagen tomada de ejemplo y la clasificación de la misma tras evaluarla mediante este clasificador.

Tabla 10.3 – Anotación real y clasificación utilizando el clasificador K-NN Combinado con K=3

	Exterior/Interior	***Personas***	***Día/Noche***	***Agua***	***Caminos***	***Vegetación***
Anotación real	*Exterior*	*No*	*Día*	*No*	*No*	*Sí*
Clasificación	*Exterior*	*No*	*Día*	*No*	*Sí*	*Sí*

	Árboles	***Montañas***	***Playa***	***Edificios***	***Animales***	***Cielo***	***Tipo de Cielo***
Anotación real	*Sí*	*Sí*	*No*	*Sí*	*No*	*Sí*	*Nublado*
Clasificación	*Sí*	*No*	*No*	*Sí*	*No*	*Sí*	*Nublado*

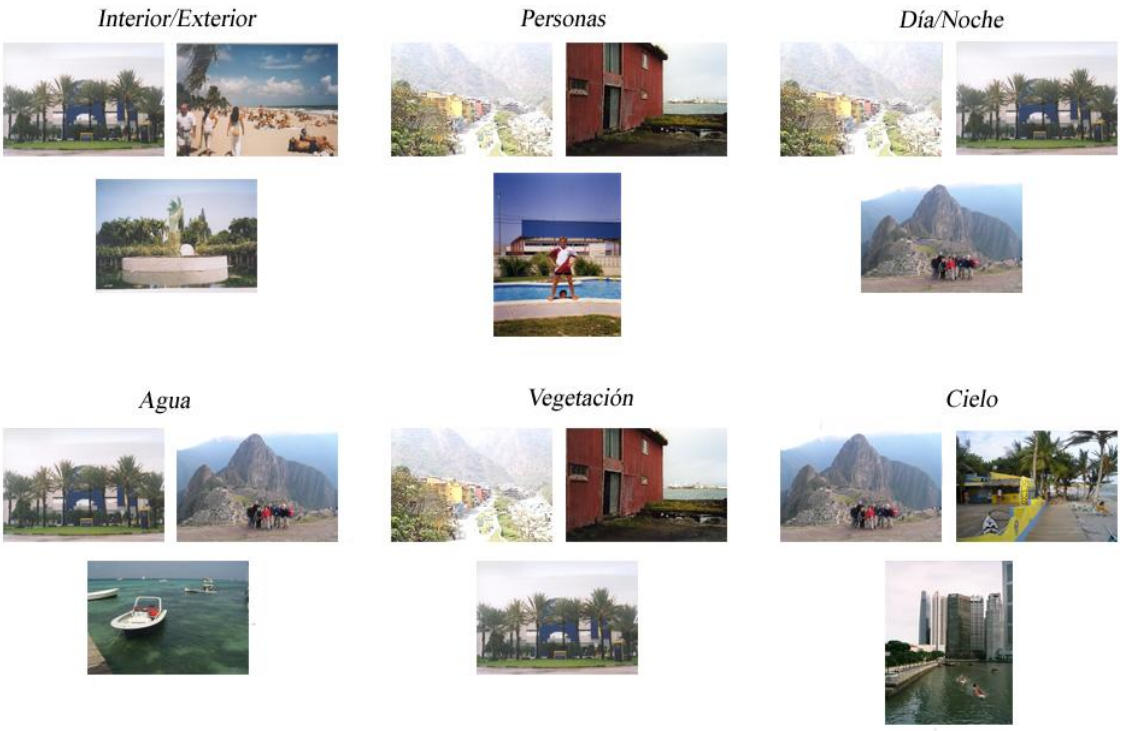

Figura 10.8 – K=3 imágenes más cercanas por concepto con el Clasificador K-NN Combinado

PARTE IV:

EVALUACIÓN

11. Evaluación

11.1 Introducción

En este capítulo se van a evaluar los diferentes métodos empleados para el desarrollo de la tarea propuesta. Para ello se presentan y analizan los resultados obtenidos durante la fase de test, ya que la fase de entrenamiento consiste únicamente en almacenar los vectores característicos y las etiquetas de los casos de entrenamiento.

Se mostrarán y evaluarán los resultados obtenidos para cada uno de los tres clasificadores implementados, analizando cada concepto en primer lugar para diferentes valores de *K* y en segundo lugar para todos los valores de *K* empleados.

Finalmente se compararán dichos resultados con los que obtuvieron los participantes de la tarea VCDT de ImageCLEF 2008.

A continuación y de modo introductorio se describen las medidas que se han utilizado para la evaluación de los distintos clasificadores.

11.2 Medidas de Evaluación

Para la evaluación de los clasificadores implementados se han utilizado una serie de medidas que se utilizan típicamente en esta clase de sistemas y son: matriz de confusión, precisión, cobertura, medida-F, curva ROC y el área bajo la curva ROC (*AUC*).

11.2.1 Matriz de Confusión

La matriz de confusión es una herramienta de visualización empleada en los sistemas de aprendizaje supervisado que contiene información acerca de la clasificación real y predicha dada por el clasificador. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa las instancias reales de la cada clase. Los beneficios de las matrices de confusión es que muestra gráficamente la calidad del clasificador y facilita ver si el sistema está confundiendo dos clases.

La tabla 11.1 muestra la matriz de confusión para un clasificador de dos clases, donde las entradas de la misma tienen los siguientes significados:

- ***TN*** (*True Negative*, Verdaderos Negativos) es el número de correctas predicciones que en realidad son negativas.
- ***FP*** (*False Positive*, Falsos Positivos) es el número de incorrectas predicciones que en realidad son negativas.
- ***FN*** (*False Negative*, Falsos Negativos) es el número de incorrectas predicciones que en realidad son positivas.
- ***TP*** (*True Positive*, Verdaderos Positivos) es el número de correctas predicciones que en realidad son positivas.

Tabla 11.1 – Matriz de Confusión para un Clasificador de dos clases

		Real	
		Positive	***Negative***
Predicho	***Positive***	***TP***	***FP***
	Negative	***FN***	***TN***

11.2.2 Precisión, Cobertura y Medida-F

La matriz de confusión proporciona una buena idea de la eficiencia del algoritmo pero no aporta detalles relativos a cómo de bien, o mal, se está llevando a cabo la clasificación. Por este motivo se miden los siguientes tres parámetros que se definen como:

Precisión:

Mide que cantidad de información de la que devuelve el sistema es correcta, expresada como el porcentaje de casos correctamente clasificados en una clase respecto al total de casos que son clasificados en esa misma clase. Es decir, es la proporción de casos predichos como correctos por el clasificador.

$$Precisión\ (\%) = \frac{TP}{TP + FP} \cdot 100 \qquad (11.1)$$

Cobertura o Recall:

Indica cuánta información relevante ha extraído el sistema, expresada como el porcentaje de casos correctamente clasificados respecto al total de casos que pertenecen a esa clase. Es decir, es la cantidad de casos clasificados correctamente como pertenecientes a una clase.

$$Cobertura\ (\%) = \frac{TP}{TP + FN} \cdot 100 \qquad (11.2)$$

Medida-F:

Entre la precisión y la cobertura existe un equilibrio, de tal forma que una precisión alta disminuye la cobertura y de forma inversa una cobertura alta disminuye en cierta forma la precisión. Por esto se obtiene esta medida que es una media armónica entre la precisión y la cobertura.

$$Medida - F\ (\%) = \frac{2 \cdot Precisión \cdot Cobertura}{Precisión + Cobertura} \qquad (11.3)$$

11.2.3 Curva ROC y Área bajo la curva

La curva ROC (acrónimo de *Receiver Operating Characteristic*, o Característica Operativa del Receptor) es una representación gráfica de la tasa de verdaderos positivos (TPR, *True Positive Rate*) frente a la tasa de falsos positivos (FPR, *False Positive Rate*) para un sistema clasificador binario según se varía el umbral de decisión. El análisis de la curva ROC proporciona información para seleccionar los modelos óptimos y descartar los que no lo son independientemente del coste de la distribución de las dos clases sobre las que se decide.

Para dibujar la curva ROC solo son necesarios los pares TPR y FPR para diversos valores del umbral de decisión. Cada par obtenido supone un punto de la curva ROC, pues se representan las FPR en el eje *x* y las TPR en el eje *y*. Estas tasas se obtienen a partir de la matriz de confusión y se describen como sigue.

- ***TPR, Tasa de Verdaderos Positivos***. Mide hasta qué punto es capaz el clasificador de clasificar los casos positivos correctamente de entre todos los casos positivos.

$$TPR = \frac{TP}{TP + FN} \qquad (11.4)$$

- ***FPR, Tasa de Falsos Negativos***. Define cuántos resultados positivos son incorrectos de entre todos los casos negativos disponibles durante la prueba.

$$FPR = \frac{FP}{FP + TN} \qquad (11.5)$$

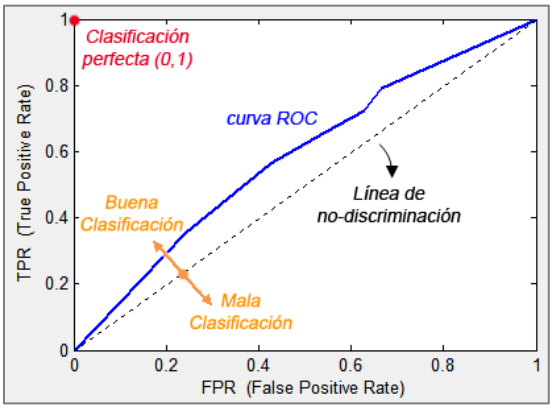

Figura 11.1 – Ejemplo de curva ROC

La mejor predicción posible será aquella que se sitúe en las coordenadas (0,1), donde la tasa de verdaderos positivos sería del 100% (ningún falso negativo) y la tasa de falsos positivos del 0% (ningún falso positivo). A este punto (0,1) también se le llama clasificación perfecta.

Por el contrario una clasificación totalmente aleatoria sería aquella que daría un punto en la diagonal desde el extremo izquierdo inferior al extremo derecho superior, la cual se llama línea de no-discriminación o aleatoriedad. Los puntos situados sobre de dicha recta determinarán que la clasificación es buena, mientras que los puntos situados bajo ella determinarán que los resultados no son buenos, siendo más eficaz invertir la clasificación

Existen diversos estadísticos que a partir de la curva ROC resumen el rendimiento del clasificador. El más utilizado es el ***AUC*** (*Area Under Curve*, o Área bajo la curva) debido a que este se interpreta como la probabilidad de que un clasificador puntuará una instancia positiva elegida aleatoriamente más alta que una negativa. Para obtener el AUC se utiliza un método básico que consiste en aproximar el valor de la integral de la curva mediante una serie de aproximaciones trapezoidales. Cabe destacar que para ello en Matlab se ha utilizado la siguiente función.

- ***trapz (X,Y)*** → Computa la integral de *Y* con respecto a *X* mediante el método de las aproximaciones trapezoidales.

11.3 Resultados

A continuación se exponen los resultados obtenidos para cada uno de los tres algoritmos implementados por categoría. En primer lugar se presentan y evalúan los resultados para el clasificador general, en segundo lugar los resultados del clasificador específico (se compararán con los resultados del clasificador general) y por último los resultados del clasificador combinado (los cuales se compararán con los métodos anteriores). Los resultados se presentan y analizan de forma individual para cada concepto.

Para los diferentes valores de *K* empleados, se muestra en primer lugar una gráfica con la variación en función de *K* de la precisión, cobertura y medida-F. En segundo lugar la matriz de confusión que proporciona una mejor proporción de la precisión y la cobertura, es decir, una mayor medida-F. Y finalmente la curva ROC del concepto junto al valor del AUC que le corresponda. Se incluirán además los comentarios que se consideren oportunos sobre cualquiera de estos resultados, y un resumen final de cada uno de los métodos implementados.

Cabe destacar que para la obtención de estos resultados se han utilizado diferentes valores de *K* en los clasificadores K-NN, en concreto, valores impares desde *K=1* hasta *K=9*. Se ha decidido establecer el límite en 9 debido a que para valores superiores los resultados obtenidos comienzan a ser deficientes.

11.3.1 Resultados del Clasificador K-NN General

Exterior/Interior:

Cabe notar que las imágenes habrán sido tomadas en exteriores o interiores, por lo que no es necesario evaluar cada concepto de forma individual ya que la aparición de uno conlleva la ausencia del otro y viceversa. De esta forma se simplifica el problema. Así se ha considerado en el clasificador binario como *'positivo'* el concepto Exterior y como *'negativo'* el concepto Interior, mostrando a continuación los resultados para el clasificador considerado.

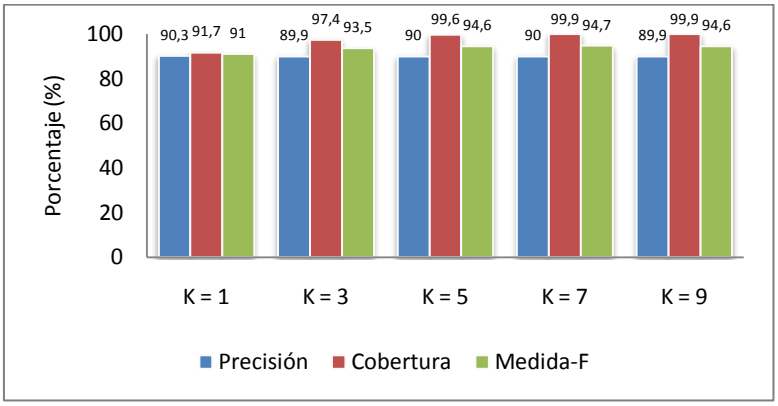

Figura 11.2 – Medidas de calidad en función de K del clasificador general para "Exterior/Interior"

Tabla 11.2 – Medidas de calidad del clasificador general con K=7 del concepto "Exterior/Interior"

Matriz de Confusión	
TP = 893	***FP = 99***
FN = 1	***TN = 1***

Precisión = 90%
Cobertura = 99,9%
Medida-F = 94,7%

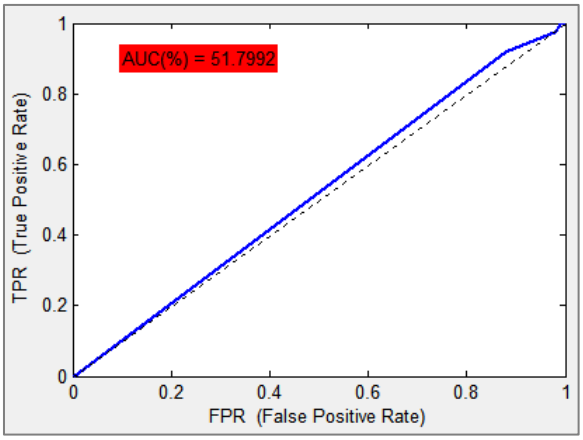

Figura 11.3 – Curva ROC y AUC del clasificador general para el concepto "Exterior/Interior"

Observando las medidas de calidad obtenidas se puede concluir que el clasificador proporciona excelentes resultados para detectar el concepto exterior, pero sin embargo los resultados para detectar el concepto inverso (interior) son francamente malos, tal y como muestran los TN y los FP de la matriz de confusión. Esto se debe a que la base de datos contiene muchas más imágenes tomadas en exteriores que en interiores, provocando que el clasificador tienda a clasificar las imágenes como exterior con mucha más probabilidad. Esta combinación de resultados es más visible en la curva ROC, donde esta es prácticamente igual a la línea de aleatoriedad, lo que genera que el AUC sea muy próximo al 50%.

Personas:

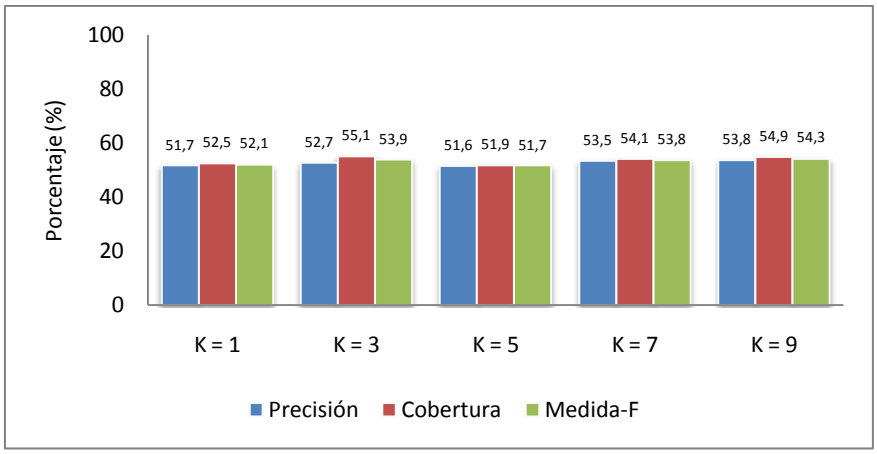

Figura 11.4 –Medidas de calidad en función de K del clasificador general para "Personas"

Tabla 11.3 – Medidas de calidad del clasificador general con K=9 del concepto "Personas"

Matriz de Confusión	
TP = 275	***FP = 236***
FN = 226	***TN = 257***

Precisión = 53,8%
Cobertura = 54,9%
Medida-F = 54,3%

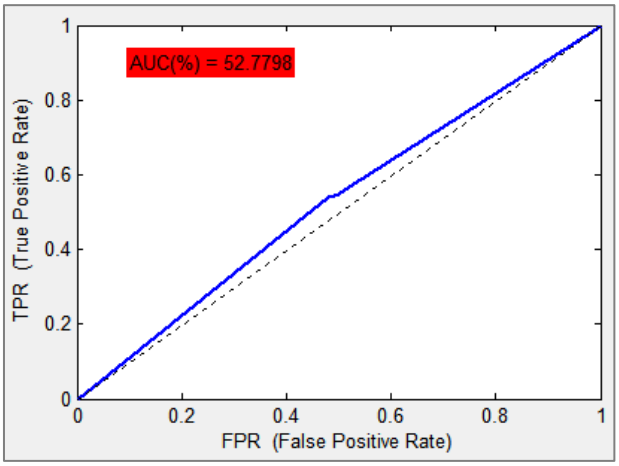

Figura 11.5 – Curva ROC y AUC del clasificador general para el concepto "Personas"

Dados los resultados anteriores se observa que se detectan tanto la aparición como la ausencia de personas en las imágenes con algo más del 50% de eficacia independientemente del valor de *K* elegido. Se podría decir que se detecta la presencia o ausencia de personas en las imágenes en una de cada dos imágenes evaluadas mediante este clasificador.

La curva ROC y el AUC (Figura 11.5) muestran lo anteriormente descrito, ya que la curva supera la línea de aleatoriedad y el AUC supera el 50%. Por tanto se puede concluir que los resultados para este concepto son aceptables pero no llegan a ser buenos o excelentes.

Día/Noche:

Al igual que ocurría con los conceptos "Exterior/Interior", el que una imagen haya sido tomada de día conlleva que no se haya tomado de noche y viceversa. Por ello y para simplificar el problema se han evaluado ambos conceptos como un solo, considerándose en el clasificador binario como *'positivo'* a "Día" y como *'negativo'* a "Noche".

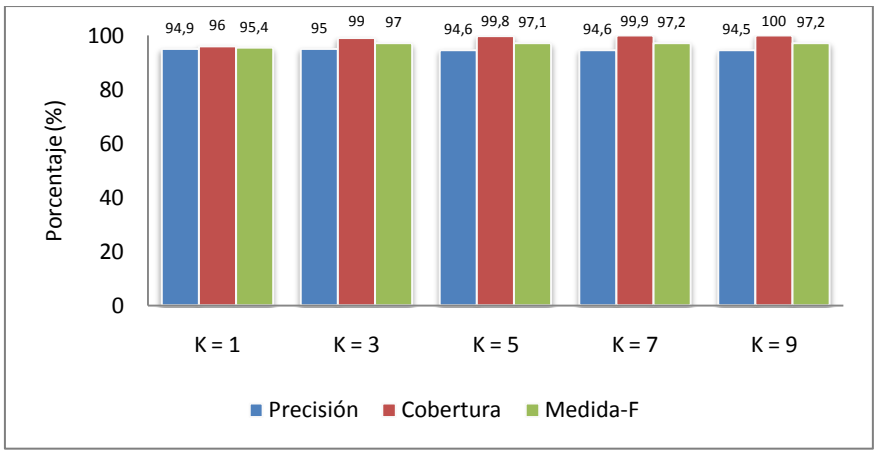

Figura 11.6 – Medidas de calidad en función de K del clasificador general para "Día/Noche"

Tabla 11.4 – Medidas de calidad del clasificador general con K=9 del concepto "Día/Noche"

Matriz de Confusión	
TP = 939	***FP = 55***
FN = 0	***TN = 0***

Precisión = 94,5%
Cobertura = 100%
Medida-F = 97,2%

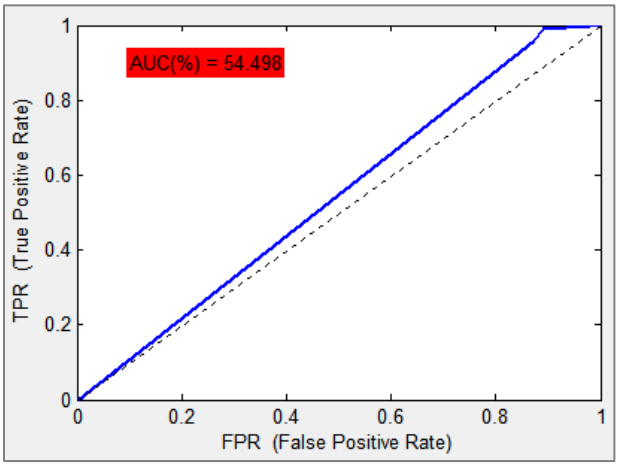

Figura 11.7 – Curva ROC y AUC del clasificador general para el concepto "Día/Noche"

Para *K=9* se obtiene una cobertura del 100%, es decir, todas las imágenes etiquetadas como día se predicen correctamente. Sin embargo ninguna de las imágenes etiquetada como noche es acertada. Esto se debe a que la proporción de imágenes etiquetadas como día en la base de datos es mucho mayor que las que son de noche, en concreto el 95% del total de imágenes de Test son de día y solamente el 5% son de noche. Esta desproporción conlleva que eligiendo valores altos de *K* sea mucho más probable predecir día que noche. Aún así, se realiza una aceptable clasificación tal y como muestra la curva ROC y el área bajo la misma.

Agua:

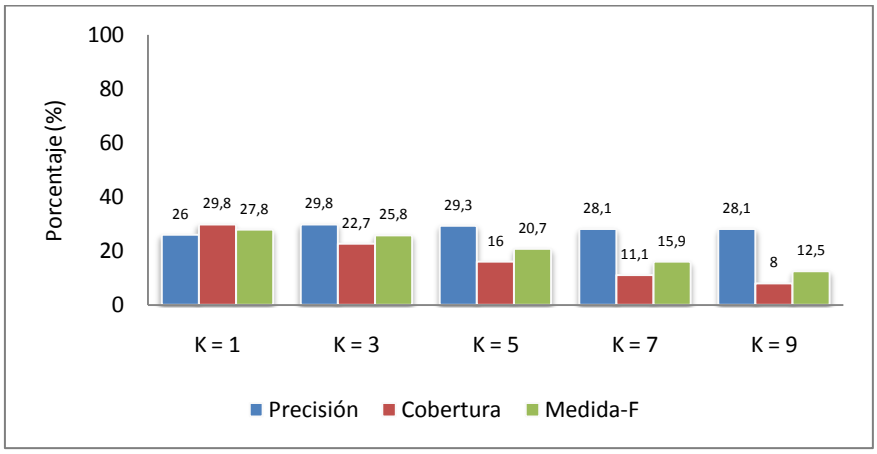

Figura 11.8 –Medidas de calidad en función de K del clasificador general para "Agua"

Tabla 11.5 – Medidas de calidad del clasificador general con K=1 del concepto "Agua"

Matriz de Confusión	
TP = 67	***FP = 191***
FN = 158	***TN = 578***

Precisión = 26%
Cobertura = 29,8%
Medida-F = 27,8%

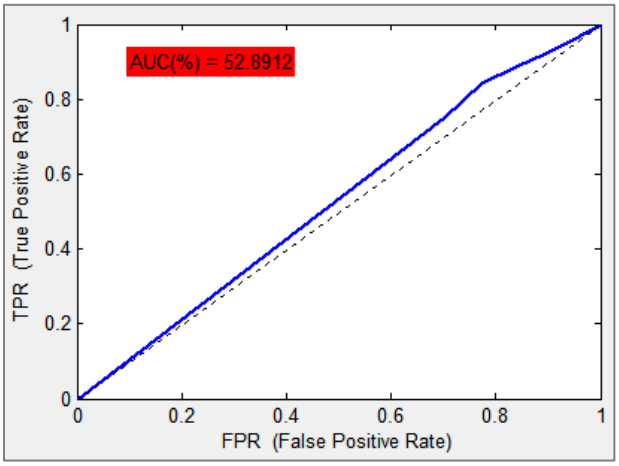

Figura 11.9 – Curva ROC y AUC del clasificador general para el concepto "Agua"

En el caso de detectar agua en las imágenes, el clasificador falla más veces que acierta. Esto provoca que se obtengan malos resultados tanto en la precisión, cobertura como medida-F.

Sin embargo para el caso en el cual el clasificador detecta la ausencia de agua, acierta un porcentaje de veces bastante mayor de las que falla.

La combinación de las dos conclusiones se observa en la curva ROC del clasificador. Así el área bajo la misma (AUC) tiene un valor del 52,9% aproximadamente, por lo que se concluye que se realiza una clasificación admisible.

Caminos:

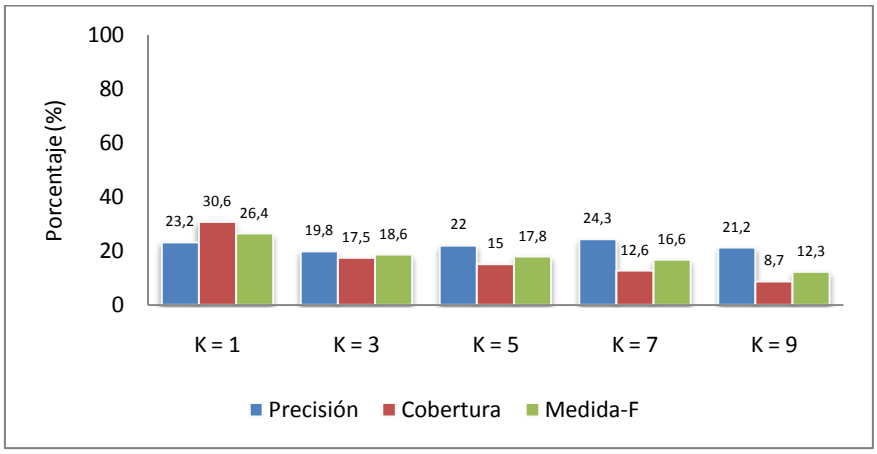

Figura 11.10 –Medidas de calidad en función de K del clasificador general para "Caminos"

Tabla 11.6 – Medidas de calidad del clasificador general con K=1 del concepto "Caminos"

Matriz de Confusión	
TP = 63	***FP = 209***
FN = 143	***TN = 579***

Precisión = 23,2%
Cobertura = 30,6%
Medida-F = 26,4%

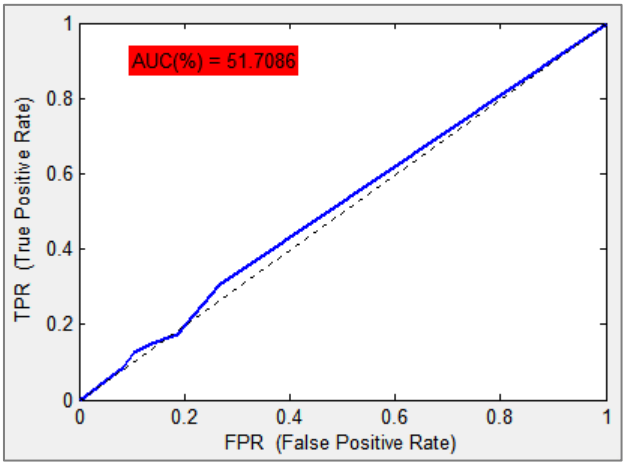

Figura 11.11 – Curva ROC y AUC del clasificador general para el concepto "Caminos"

Tal y como se observa en la figura 11.10 la medida-F disminuye en función del valor de *K* debido a que también lo hace la cobertura según *K* aumenta. Esto significa que a medida que se utiliza un mayor número de imágenes más parecidas a la evaluada el clasificador tiende a detectar que no hay caminos donde en realidad sí los hay. Esto se debe a la dificultad de encontrar características que puedan describir la aparición de caminos en las imágenes y a que el porcentaje de imágenes con caminos es mucho menor al porcentaje de imágenes sin caminos de la base de datos, en concreto hay una relación aproximada de 1 imagen con caminos contra 4 sin ellos.

Esto provoca que la clasificación que se realiza para la detección de caminos sea prácticamente aleatoria.

Vegetación:

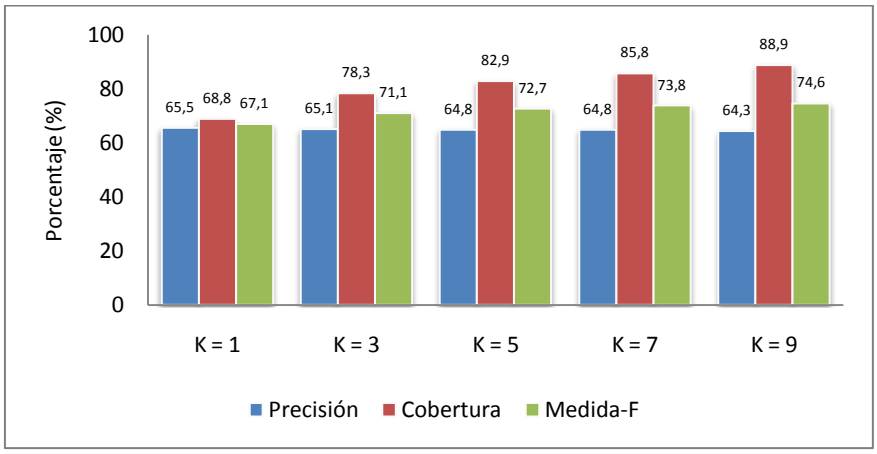

Figura 11.12 –Medidas de calidad en función de K del clasificador general para "Vegetación"

Tabla 11.7 – Medidas de calidad del clasificador general con K=9 del concepto "Vegetación"

Matriz de Confusión	
TP = 552	***FP = 307***
FN = 69	***TN = 66***

Precisión = 64,3%
Cobertura = 88,9%
Medida-F = 74,6%

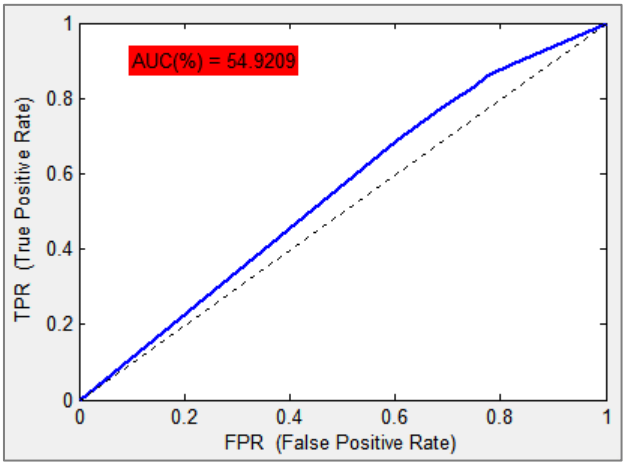

Figura 11.13 – Curva ROC y AUC del clasificador general para el concepto "Vegetación"

De forma inversa a la que ocurría con el concepto caminos, en este caso la cobertura aumenta según lo hace el valor de *K* utilizado. Por tanto el clasificador tiende a medida que aumenta *K* a detectar la presencia de vegetación en más imágenes. Esto provoca que el clasificador acierte más en detectar la presencia de vegetación (TP), pero también que falle más detectando vegetación cuando no la hay (FP), lo que en cierta forma es lógico puesto que la base de datos contiene prácticamente el doble de imágenes con vegetación que sin ella.

El valor obtenido de área bajo la curva ROC es bueno, aproximadamente del 55%, pero podría ser mejor si al aumentar los TP no lo hicieran también los FP.

Árboles:

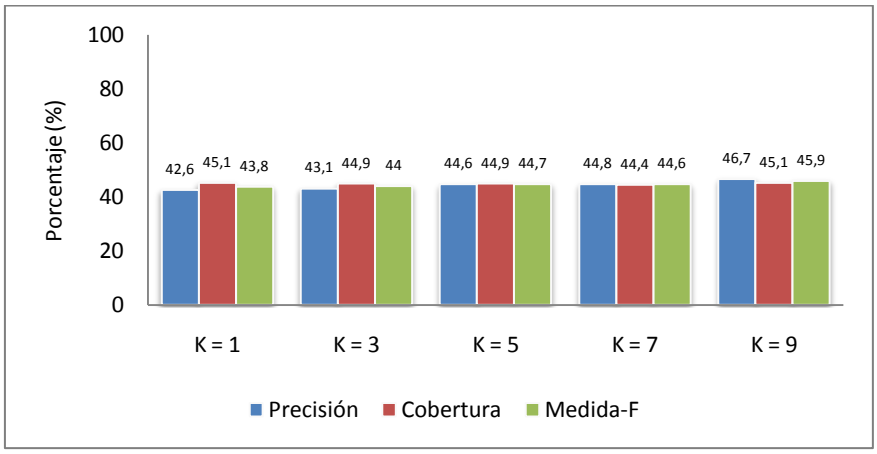

Figura 11.14 –Medidas de calidad en función de K del clasificador general para "Árboles"

Tabla 11.8 – Medidas de calidad del clasificador general con K=9 del concepto "Árboles"

Matriz de Confusión	
TP = 184	***FP = 210***
FN = 224	***TN = 376***

Precisión = 46,7%
Cobertura = 45,1%
Medida-F = 45,9%

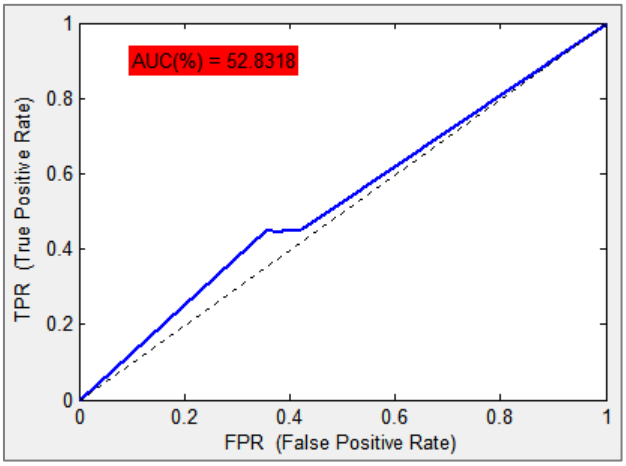

Figura 11.15 – Curva ROC y AUC del clasificador general para el concepto "Árboles"

La figura 11.4 muestra cómo a medida que aumenta el valor de *K* aumentan la precisión, la cobertura y consecuentemente la medida-F, aunque este aumento es prácticamente insignificante. Estas medidas se mueven para este concepto en torno a valores bajos, en concreto alrededor del 45%.

Observando la matriz de confusión para *K=9* se puede concluir que el clasificador no detecta lo suficientemente bien la presencia de árboles en las imágenes, por lo que lógicamente el clasificador detecta mejor la ausencia que la presencia de dicho concepto. Esto genera que el valor del área bajo la curva ROC obtenido tenga un resultado escasamente positivo, ya que el objetivo del clasificador es la detección de árboles en las imágenes.

Montañas:

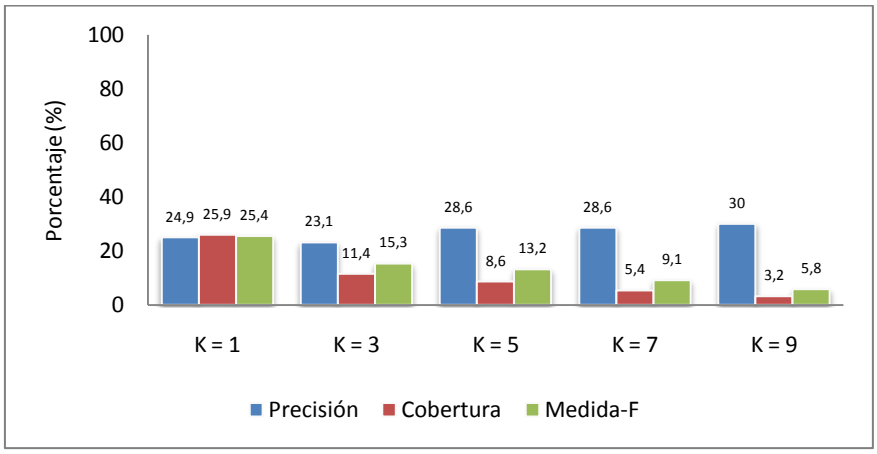

Figura 11.16 –Medidas de calidad en función de K del clasificador general para "Montañas"

Tabla 11.9 – Medidas de calidad del clasificador general con K=1 del concepto "Montañas"

Matriz de Confusión	
TP = 48	***FP = 145***
FN = 137	***TN = 664***

Precisión = 24,9%
Cobertura = 25,9%
Medida-F = 25,4%

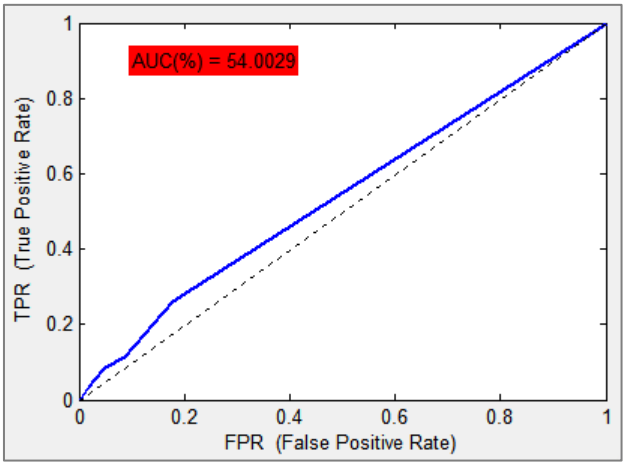

Figura 11.17 – Curva ROC y AUC del clasificador general para el concepto "Montañas"

Los resultados obtenidos detallan que la cobertura disminuye hasta valores insignificantes, lo que significa que el clasificador tiene tendencia a concluir que no existen montañas en las imágenes que se evalúan. Esto provoca que aumente la precisión, ya que prácticamente las pocas imágenes que se detectan con montañas realmente si aparecen en ellas.

Así en la curva ROC, la tasa de verdaderos positivos (TPR) es algo mayor que la tasa de falsos negativos (FPR), obteniendo un área bajo la misma con un valor aceptable. De esta forma se puede concluir que el clasificador no es malo, pero detecta mucho mejor la ausencia que la presencia de montañas en las imágenes, lo que no era el objetivo marcado.

Playa:

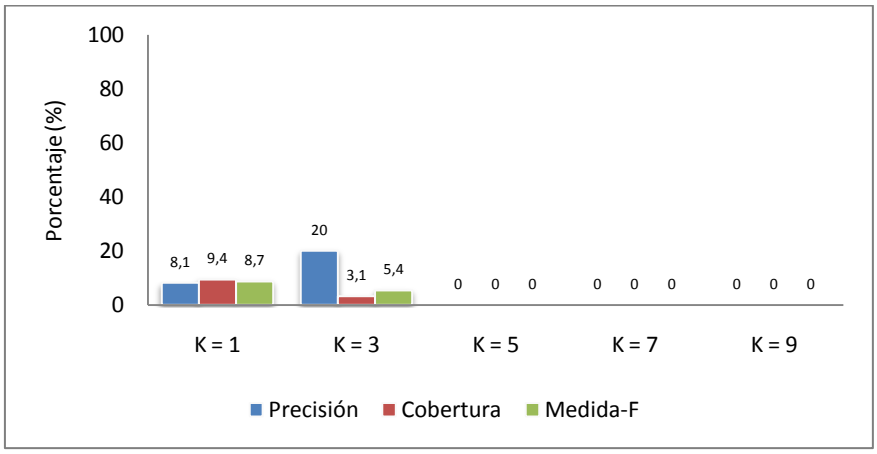

Figura 11.18 –Medidas de calidad en función de K del clasificador general para "Playa"

Tabla 11.10 – Medidas de calidad del clasificador general con K=1 del concepto "Playa"

Matriz de Confusión	
TP = 3	***FP = 34***
FN = 29	***TN = 928***

Precisión = 8,1%
Cobertura = 9,4%
Medida-F = 8,7%

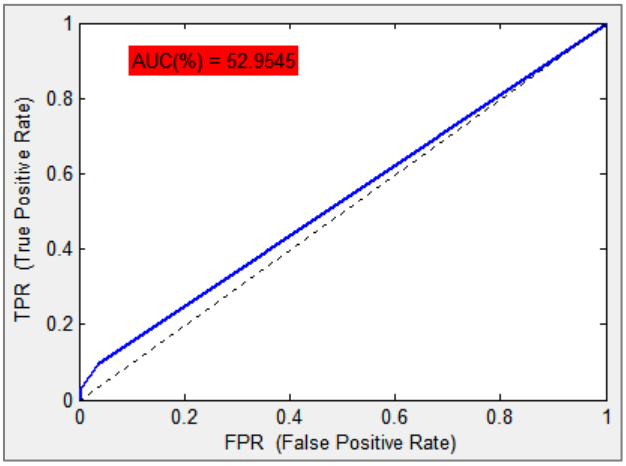

Figura 11.19 – Curva ROC y AUC del clasificador general para el concepto "Playa"

Sin duda este es uno de los peores conceptos evaluados. Como muestra la gráfica de la evolución de las medidas de calidad, para valores altos de *K*, estas medidas toman valor nulo. Es decir para valores de *K>3* el clasificador no detecta que en ninguna imagen aparezca playa.

La curva ROC puede ser engañosa, puesto que se obtiene un valor aceptable del AUC, pero se debe a que la base de datos contiene un porcentaje muy pequeño de imágenes con playa, y como clasifica todas con la ausencia de la misma acierta prácticamente en la mayoría de las imágenes.

En conclusión la clasificación no es buena y puede deberse a la poca cantidad de imágenes de test que contienen playa y a que los parámetros generales extraídos a las imágenes no describen de una forma adecuada la presencia de playa en ellas.

Edificios:

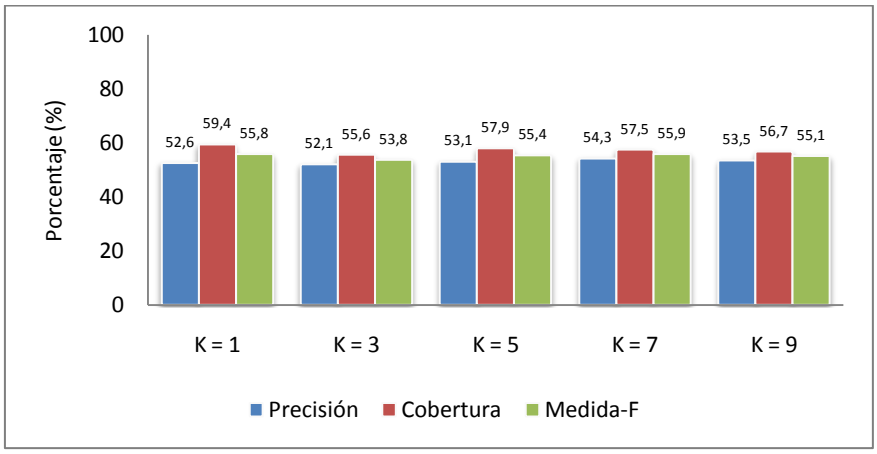

Figura 11.20 –Medidas de calidad en función de K del clasificador general para "Edificios"

Tabla 11.11 – Medidas de calidad del clasificador general con K=7 del concepto "Edificios"

Matriz de Confusión	
TP = 275	***FP = 231***
FN = 203	***TN = 285***

Precisión = 54,3%
Cobertura = 57,5%
Medida-F = 55,9%

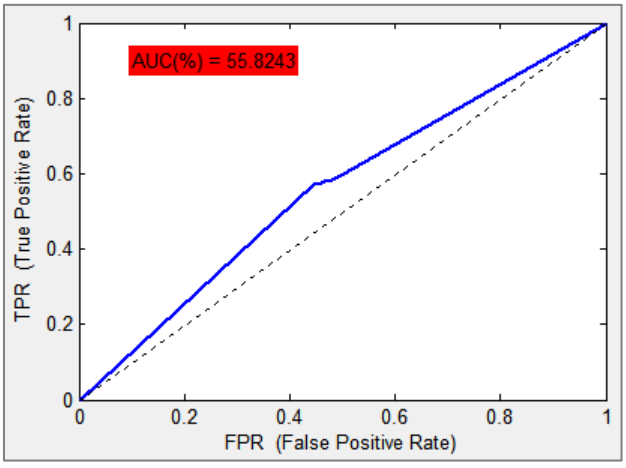

Figura 11.21 – Curva ROC y AUC del clasificador general para el concepto "Edificios"

Dado lo que se observa en la figura 11.20 se deduce que la precisión, cobertura y por tanto medida-F son prácticamente invariantes en función de *K*, ya que se mueven alrededor de valores próximos al 50% y 60%, siendo valores bastante aceptables.

La matriz de confusión muestra como el clasificador acierta más veces de las que falla, ya que tanto los true positive como los verdaderos negativos son mayores que los falsos positivos y falsos negativos. Así se genera una curva ROC bastante buena, donde el valor del AUC supera con creces el 50%, por lo que la clasificación para la detección de edificios es razonablemente aceptable.

Animales:

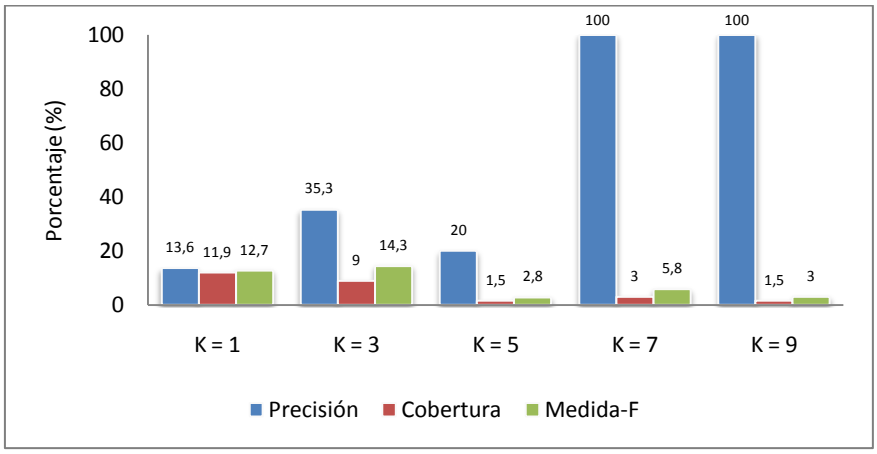

Figura 11.22 –Medidas de calidad en función de K del clasificador general para "Animales"

Tabla 11.12 – Medidas de calidad del clasificador general con K=3 del concepto "Animales"

Matriz de Confusión	
TP = 6	***FP = 11***
FN = 61	***TN = 916***

Precisión = 35,3%
Cobertura = 9%
Medida-F = 14,3%

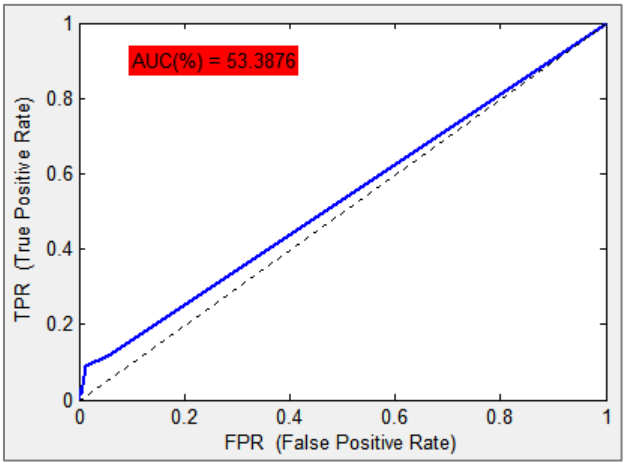

Figura 11.23 – Curva ROC y AUC del clasificador general para el concepto "Animales"

Para el caso de clasificar el concepto animales en las imágenes se obtienen unos resultados no demasiado buenos. Los valores de precisión y cobertura obtenidos son muy bajos, por lo que el clasificador no detecta bien la presencia de animales en las imágenes.

Cabe destacar que la precisión para *K=7* y *K=9* es un resultado algo engañoso puesto que en ambos casos es del 100%. Esto se debe a que a que no existe ningún falso negativo, pero es engañoso porque el número de true positive es muy pequeño.

La curva ROC muestra que para este concepto el clasificador es aceptable, y es así puesto que aunque el porcentaje de aciertos donde si hay animales es muy bajo, a su vez el porcentaje de aciertos donde no hay animales es muy alto.

Cielo:

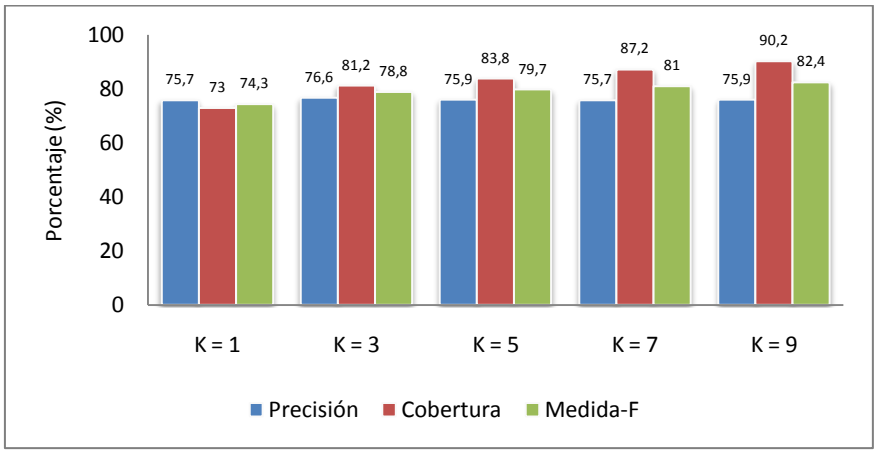

Figura 11.24 –Medidas de calidad en función de K del clasificador general para "Cielo"

Tabla 11.13 – Medidas de calidad del clasificador general con K=9 del concepto "Cielo"

Matriz de Confusión	
TP = 642	***FP = 204***
FN = 70	***TN = 78***

Precisión = 75,9%
Cobertura = 90,2%
Medida-F = 82,4%

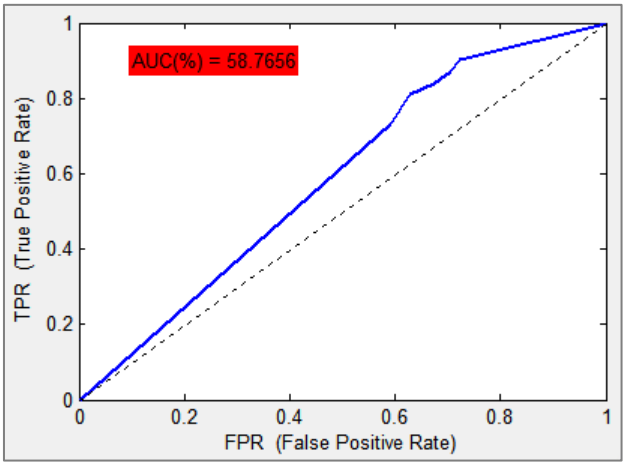

Figura 11.25 – Curva ROC y AUC del clasificador general para el concepto "Cielo"

El concepto cielo es sin duda para el que mejor resultados se obtienen. Tal y como muestra la evolución de las medidas de calidad en función de *K*, estás aumentan según aumenta el valor de *K*. Los valores obtenidos de precisión y cobertura son cada vez más altos, y se debe a que el porcentaje de imágenes que contienen cielo en el conjunto de entrenamiento es muy alto. De esta forma al tomar más imágenes de entrenamiento para clasificar cada imagen de test, es más posible que se clasifique con la aparición de cielo que con la ausencia del mismo. Así los mejores resultados de las medidas de calidad tomadas son para el mayor valor de *K* utilizado, 9.

Debido a las conclusiones anteriores se obtiene una buena curva ROC, que genera un AUC próximo al 59%. Así se concluye que se realiza una muy buena clasificación para este concepto.

Soleado:

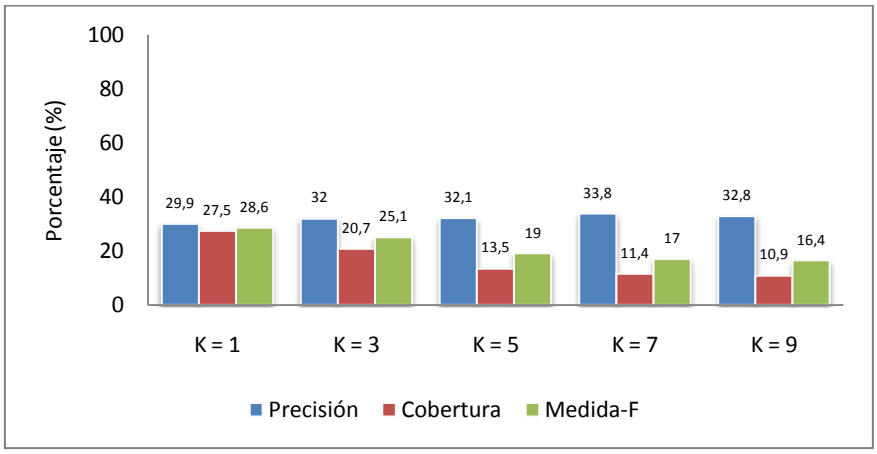

Figura 11.26 –Medidas de calidad en función de K del clasificador general para "Soleado"

Tabla 11.14 – Medidas de calidad del clasificador general con K=1 del concepto "Soleado"

Matriz de Confusión	
TP = 53	***FP = 124***
FN = 140	***TN = 677***

Precisión = 29,9%
Cobertura = 27,5%
Medida-F = 28,6%

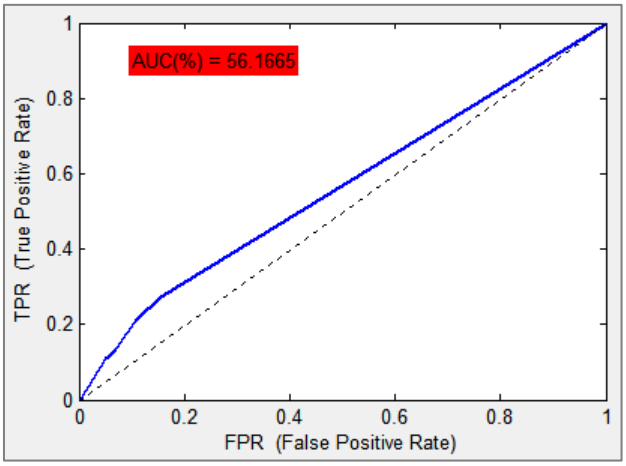

Figura 11.27 – Curva ROC y AUC del clasificador general para el concepto "Soleado"

A medida que aumenta el valor de K disminuye la cobertura del clasificador. El factor causante de esta disminución es que, como ya se ha comentado, el tipo de cielo tiene tres posibles valores: soleado, parcialmente nublado y nublado. A pesar de que se han evaluado cada uno de ellos como un concepto individual, clasificando su aparición o no, la posibilidad de aparición de cada uno de ellos en las imágenes hace que al aumentar el valor de K aumente la probabilidad de que las imágenes de entrenamiento que se utilizan para clasificar sean de los otros dos tipos de cielo al aquí evaluado (soleado). Debido a esto el clasificador tenderá a clasificar las imágenes con cielo soleado con la ausencia de este, aumentando por tanto los falsos positivos y por tanto disminuyendo la cobertura.

Aún así los resultados obtenidos en las medidas de calidad pueden considerarse adecuados, al igual que el valor del AUC, siendo por tanto el realizado un buen clasificador.

Parcialmente Nublado:

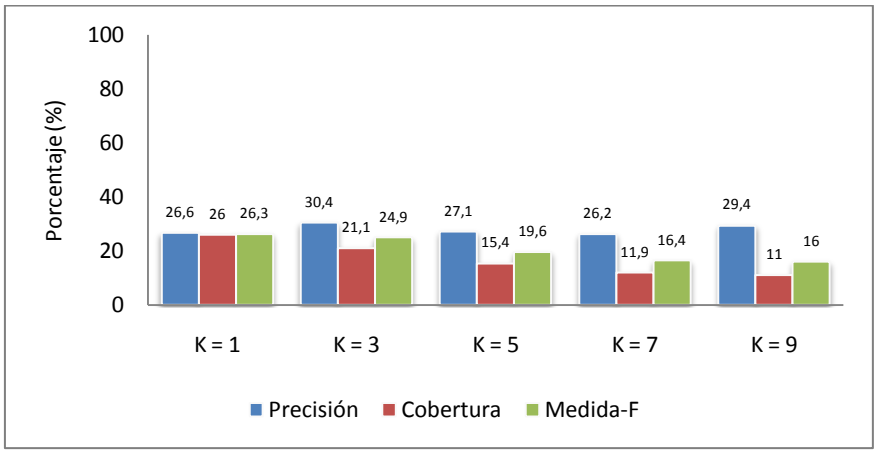

Figura 11.28 –Medidas de calidad en función de K del clasificador general para "P.Nublado"

Tabla 11.15 – Medidas de calidad del clasificador general con K=1 del concepto "P.Nublado"

Matriz de Confusión	
TP = 59	***FP = 163***
FN = 168	***TN = 604***

Precisión = 26,6%
Cobertura = 26%
Medida-F = 26,3%

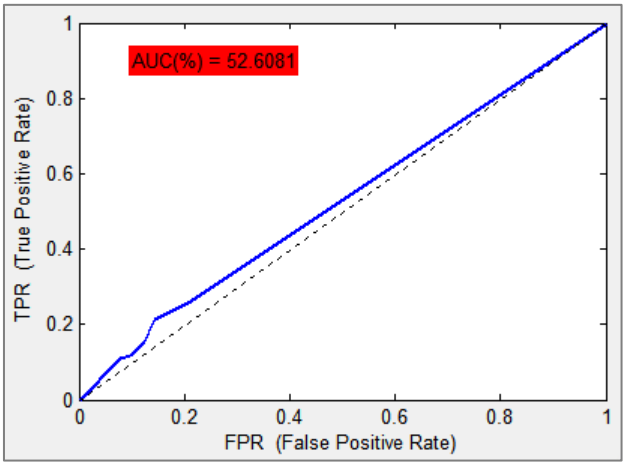

Figura 11.29 – Curva ROC y AUC del clasificador general para el concepto "P.Nublado"

En la figura 11.28 se observa que la precisión se mantiene pero la cobertura disminuye, lo que hace que disminuya también la medida-F. Además se observa que los valores de dichas medidas de calidad no son demasiado buenos. Esto se debe principalmente a que el clasificador detecta con más eficacia la ausencia que la presencia de cielo parcialmente nublado, como se puede ver en la matriz de confusión que proporciona la mejor medida-F (tabla 11.14).

Aún así la curva ROC y el área bajo la misma muestran un resultado positivo, por lo que puede considerarse que se realiza una clasificación aceptable para este concepto.

Nublado:

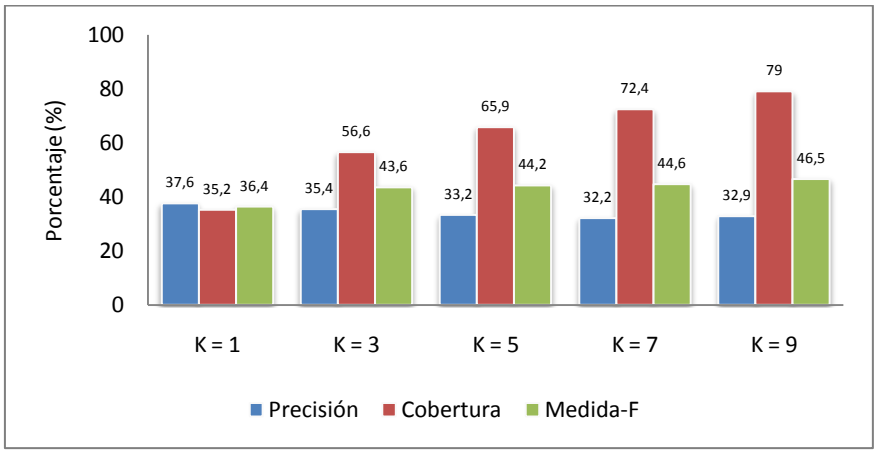

Figura 11.30 –Medidas de calidad en función de K del clasificador general para "Nublado"

Tabla 11.16 – Medidas de calidad del clasificador general con K=9 del concepto "Nublado"

Matriz de Confusión	
TP = 229	***FP = 468***
FN = 61	***TN = 236***

Precisión = 32,9%
Cobertura = 79%
Medida-F = 46,5%

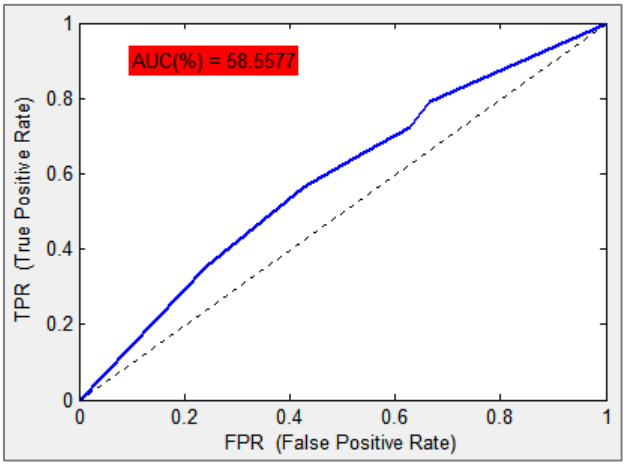

Figura 11.31 – Curva ROC y AUC del clasificador general para el concepto "Nublado"

La evolución de la cobertura es muy significativa, pues aumenta considerablemente según lo hace el valor de *K*. Esto significa que según aumenta *K* aumenta el número de verdaderos positivos y disminuyen los falsos negativos, realizándose por tanto una buena clasificación.

A su vez la buena clasificación realizada para el concepto nublado puede visualizarse también en la curva ROC, donde esta supera con creces la recta de aleatoriedad y se obtiene un valor del AUC superior al 58%.

Resumen:

Tabla 11.17 – Resumen de los valores de AUC por concepto del clasificador general

Exterior/Interior: 51,79992%	***Personas: 57,7998%***	***Día/Noche: 54,498%***
Agua: 52,8318%	***Caminos: 51,7086%***	***Vegetación: 54,9209%***
Árboles: 52,8318%	***Montañas: 54,0029%***	***Playa: 52,9545%***
Edificios: 55,8243%	***Animales: 53,3876%***	***Cielo: 58,7656%***
Soleado: 56,1665%	***Parcialmente Nublado: 52,6081%***	***Nublado: 58,5577%***

Máximo – Cielo: 58,7656% ***Mínimo – Caminos: 51,7086%***

$$\textbf{\textit{Media}} = (\sum_{i=1}^{15} AUC_i)/15 = 54{,}1172\%$$

El área bajo la curva ROC (*AUC*) para todos los conceptos del clasificador general supera el umbral del 50%, por lo que para todos ellos se puede decir que se realiza una clasificación aceptable.

Tal y como muestra la tabla anterior, el menor AUC corresponde al concepto "Caminos". En este caso el valor obtenido es poco superior al 50% lo que conlleva que la clasificación de dicho concepto en las imágenes se realice prácticamente de forma aleatoria. Esta misma conclusión puede determinarse para el concepto "Exterior/Interior", pues el valor obtenido del AUC es similar. Pero para este último caso, como ya se ha comentado anteriormente, cabe destacar que el clasificador tiene una gran precisión a la hora de detectar imágenes tomadas en exteriores pero no en interiores. Esto no ocurre con el concepto "Caminos", donde la precisión tanto de detectar la presencia como ausencia de caminos no es muy buena.

El máximo se obtiene para el concepto "Cielo", que al igual que para las imágenes con cielo nublado se obtienen valores del AUC próximos al 60%. Para estos casos la clasificación realizada en las imágenes es muy buena. De esta forma se puede concluir que las características generales extraídas a las imágenes detallan con gran capacidad la aparición/ausencia de cielo y la determinación de que, en caso de que este aparezca, sea nublado o no.

Para el resto de conceptos se obtienen valores de AUC en el rango 52-57%, realizándose por tanto una buena clasificación para todos ellos.

Con los resultados obtenidos para todos los conceptos del clasificador general se puede determinar que en la clasificación final de la imagen, realizando una media, se acertará aproximadamente en la mitad de los conceptos evaluados por el sistema.

De forma general se considera que las características extraídas a las imágenes para su posterior clasificación son suficientes para la obtención de buenos resultados mediante un clasificador K-NN. Se podría elevar el número de características a extraer, pero podría suponer un desequilibrio en los resultados de los conceptos, es decir, un número muy elevado de características provocaría que se detecten con gran precisión la aparición/ausencia de ciertos conceptos pero conllevaría a que muchos otros no sean bien identificados por dichas características y se obtuvieran malos resultados a la hora de clasificarlos. Además el coste computacional del sistema aumentará de forma elevada al tener que realizar más procesos para obtener las nuevas características, y cabe tener en cuenta que ya para el caso desarrollado el coste computacional no es demasiado bajo.

11.3.2 Resultados del Clasificador K-NN Específico

Exterior/Interior:

Al igual que en el Clasificador K-NN General, los conceptos de exterior e interior se han considerado como un único concepto, simplificando así el problema.

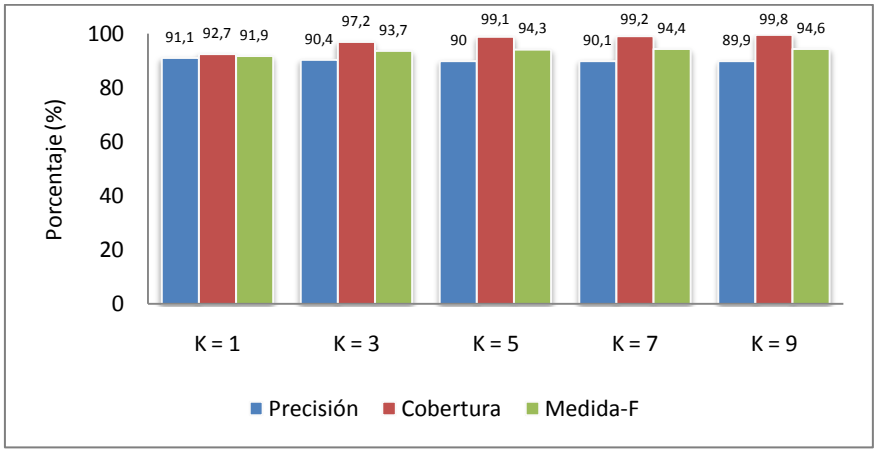

Figura 11.32 –Medidas de calidad en función de K del clasificador específico para "Exterior/Interior"

Tabla 11.18 – Medidas de calidad del clasificador específico con K=9 del concepto "Exterior/Interior"

Matriz de Confusión	
TP = 892	***FP = 100***
FN = 2	***TN = 0***

Precisión = 89,9%
Cobertura = 99,8%
Medida-F = 94,6%

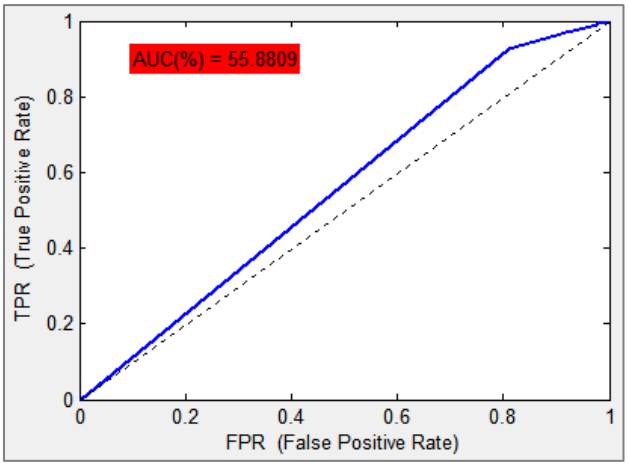

Figura 11.33 – Curva ROC y AUC del clasificador específico para el concepto "Exterior/Interior"

Las medidas de calidad obtenidas son excelentes, alrededor del 90% para los diferentes valores de K utilizados, al igual que ocurría en el clasificador general.

Sin embargo la curva ROC mejora, obteniéndose un AUC prácticamente del 56% mientras que en el método anterior era aproximadamente del 52%. Así se puede concluir que las características específicas extraídas para exterior/interior son muy buenas y proporcionan una clasificación con buena precisión.

Personas:

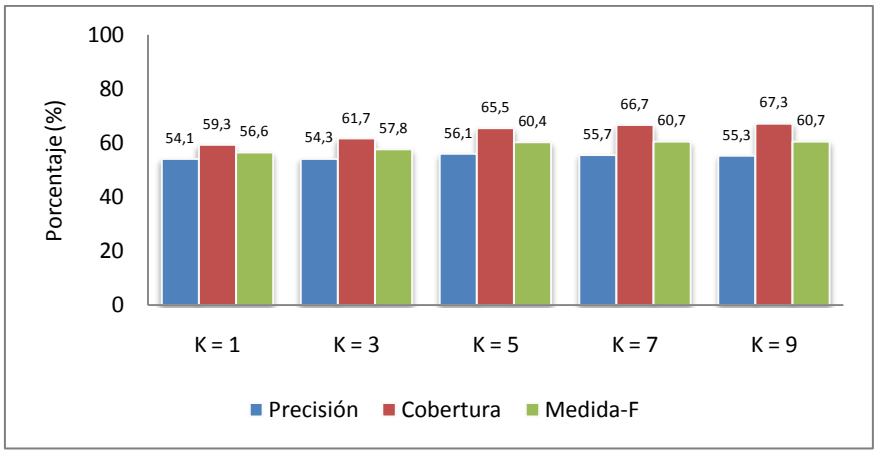

Figura 11.34 –Medidas de calidad en función de K del clasificador específico para "Personas"

Tabla 11.19 – Medidas de calidad del clasificador específico con K=7 del concepto "Personas"

Matriz de Confusión	
TP = 334	***FP = 266***
FN = 167	***TN = 227***

Precisión = 55,7%
Cobertura = 66,7%
Medida-F = 60,7%

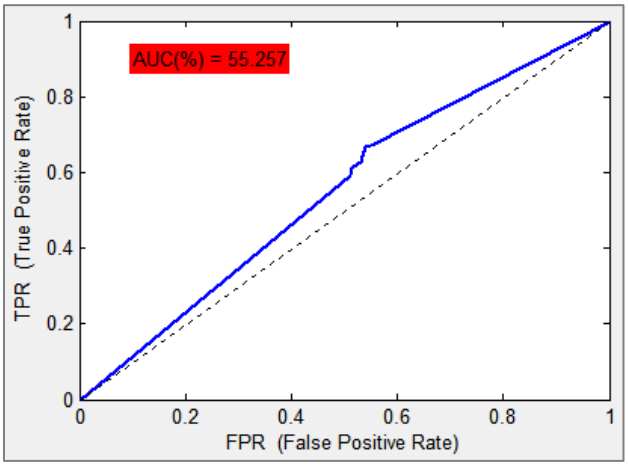

Figura 11.35 – Curva ROC y AUC del clasificador específico para el concepto "Personas"

Se obtienen mejores resultados tanto en las medidas de calidad como en la curva ROC utilizando las características específicas en vez de las generales. El aumento en la precisión y la cobertura no es demasiado grande pero sí es considerable. Y dicho aumento se hace más constatable, tal y como muestra la figura 11.34, según aumenta el valor de *K*. Es decir, a medida que se utilizan más imágenes de entrenamiento para clasificar cada imagen de test, aumenta la probabilidad de que el clasificador acierte detectando la presencia de personas.

Por tanto se puede determinar que el concepto personas se detecta en las imágenes con mayor precisión utilizando las características específicas en vez de las características generales.

Día/Noche:

Al igual que en el clasificador general, los conceptos ahora evaluados se han considerado como un único concepto, considerándose como *'positivo'* a "Día" y como *'negativo'* a "Noche".

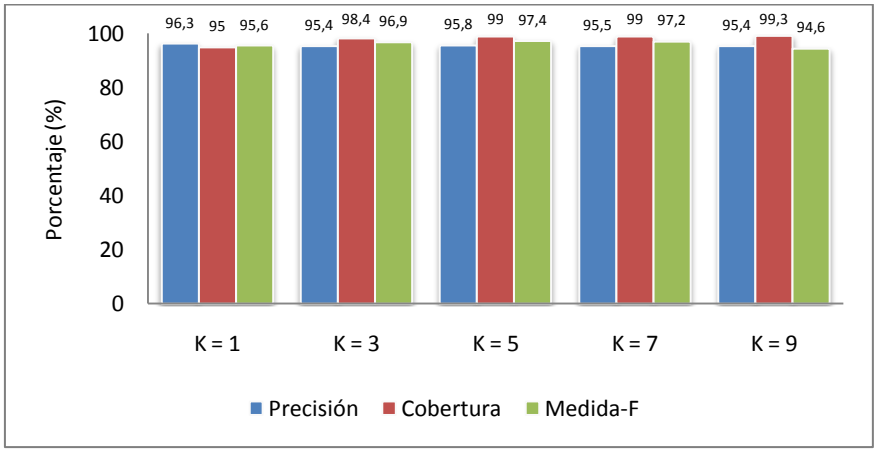

Figura 11.36 –Medidas de calidad en función de K del clasificador específico para "Día/Noche"

Tabla 11.20 – Medidas de calidad del clasificador específico con K=5 del concepto "Día/Noche"

Matriz de Confusión	
TP = 930	***FP = 41***
FN = 9	***TN = 14***

Precisión = 95,8%
Cobertura = 99%
Medida-F = 97,4%

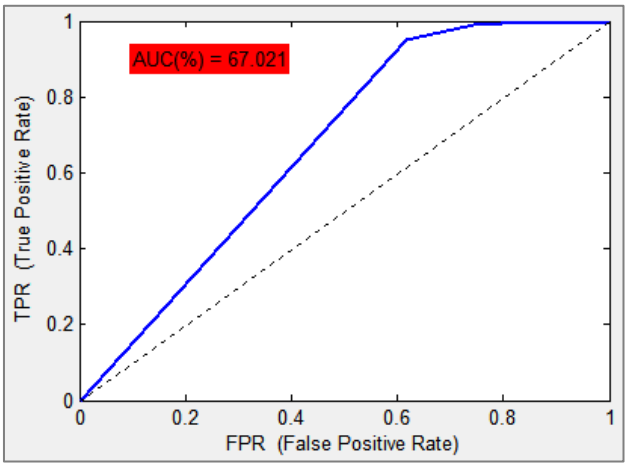

Figura 11.37 – Curva ROC y AUC del clasificador específico para el concepto "Día/Noche"

A la hora de detectar si la imagen se ha tomado de día o de noche, tanto para el clasificador general como para el específico se obtienen unos resultados excelentes, puesto que las medidas de calidad se encuentran siempre por encima del 95%.

Sin embargo, observando las curvas ROC en ambos métodos (figuras 11.7 y 11.37), se contempla que el clasificador específico es mucho mejor puesto el AUC es mayor. Esto se debe a que en el caso general el número de imágenes tomadas de noche que se clasifican correctamente es muy pequeño, mientras que en el caso específico dicho número es considerablemente mayor.

Agua:

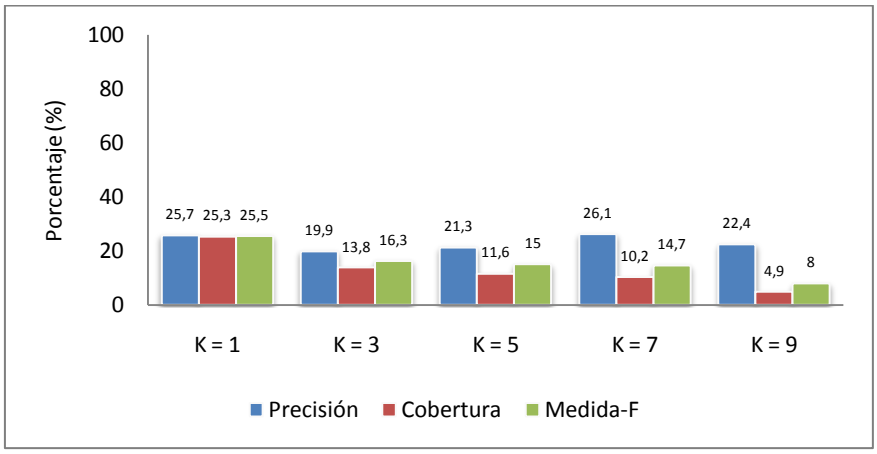

Figura 11.38 – Medidas de calidad en función de K del clasificador específico para "Agua"

Tabla 11.21 – Medidas de calidad del clasificador específico con K=1 del concepto "Agua"

Matriz de Confusión	
TP = 57	***FP = 165***
FN = 168	***TN = 604***

Precisión = 25,7%
Cobertura = 25,3%
Medida-F = 25,5%

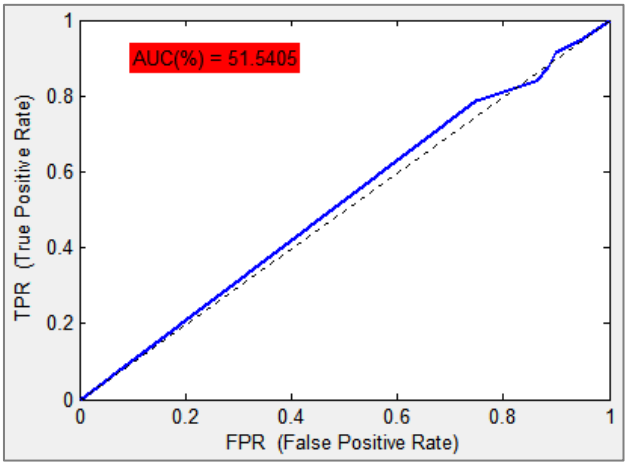

Figura 11.39 – Curva ROC y AUC del clasificador específico para el concepto "Agua"

Para el método en el que se utilizan las características generales, se ha concluido que la clasificación realizada es admisible. En este método, el específico, también se puede considerar una clasificación aceptable para el concepto agua aunque los resultados son incluso menores. La curva ROC supera escasamente el 50% (nivel que supondría la aleatoriedad en la clasificación) y tanto la precisión, cobertura y consecuentemente medida-F tienen valores bastante bajos y que disminuyen a medida que *K* aumenta.

Así se deduce que las características específicas utilizadas para detectar agua en las imágenes no son lo suficientemente buenas para que el clasificador diferencie la presencia o no de agua en ellas, al igual que ocurre con el método general.

Vegetación:

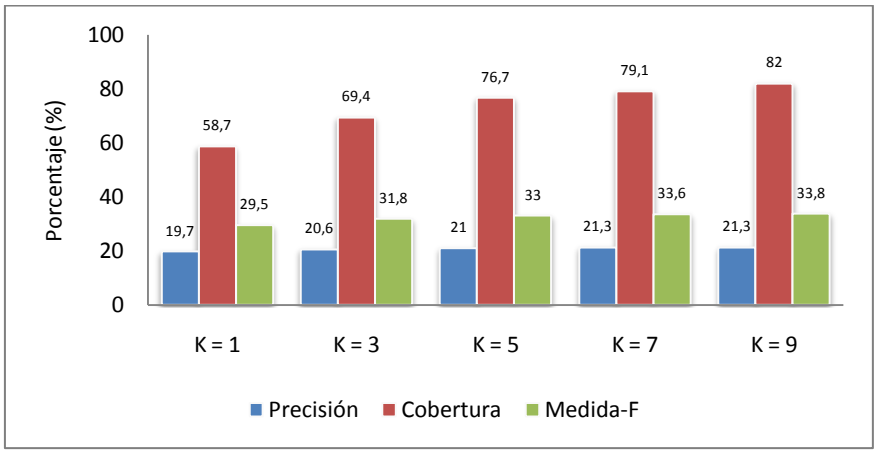

Figura 11.40 –Medidas de calidad en función de K del clasificador específico para "Vegetación"

Tabla 11.22 – Medidas de calidad del clasificador específico con K=9 del concepto "Vegetación"

Matriz de Confusión	
TP = 169	***FP = 624***
FN = 37	***TN = 164***

Precisión = 21,3%
Cobertura = 82%
Medida-F = 33,8%

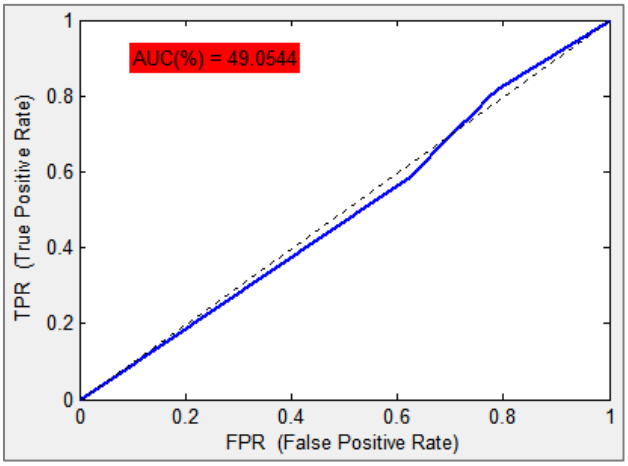

Figura 11.41 – Curva ROC y AUC del clasificador específico para el concepto "Vegetación"

La figura 11.40 muestra como se obtienen valores considerables para la cobertura y valores muy malos de precisión, lo que provoca que la medida-F no sea lo suficientemente buena como debería ser. Comparándolo con el método general se observa que la precisión es mucho menor, y esto se debe a que el número de falsos positivos aumenta considerablemente en este caso. Así el clasificador específico falla con mayor probabilidad al detectar vegetación cuando en realidad no la hay.

Dicho aumento de los falsos positivos provoca que la curva ROC obtenida se encuentre por debajo de la línea de aleatoriedad, lo que denota que el clasificador implementado no sea bueno. Por tanto las características específicas utilizadas para vegetación no aportan información relativa a la aparición de la misma en las imágenes.

Cielo:

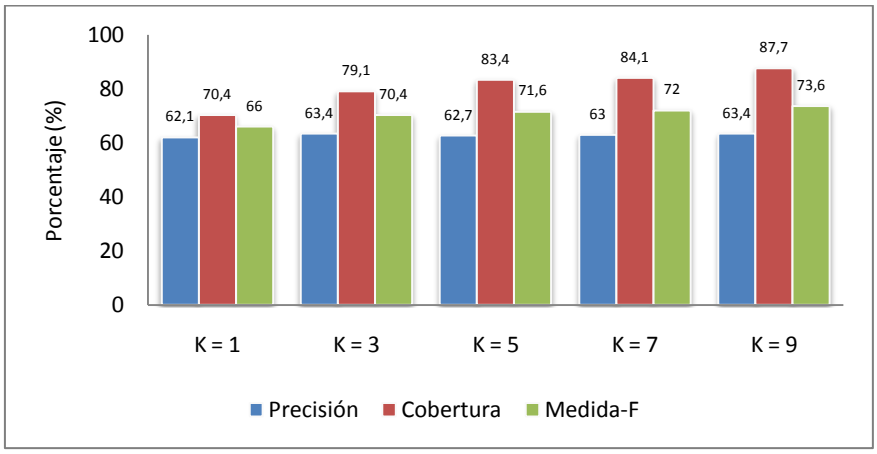

Figura 11.42 –Medidas de calidad en función de K del clasificador específico para "Cielo"

Tabla 11.23 – Medidas de calidad del clasificador específico con K=9 del concepto "Cielo"

Matriz de Confusión	
TP = 545	***FP = 314***
FN = 76	***TN = 59***

Precisión = 63,4%
Cobertura = 87,7%
Medida-F = 73,6%

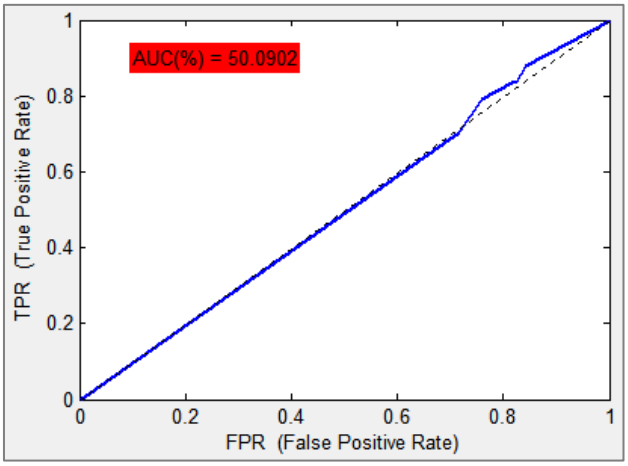

Figura 11.43 – Curva ROC y AUC del clasificador específico para el concepto "Cielo"

El concepto cielo es uno de los conceptos que con más eficacia es detectado en el clasificador general. Sin embargo en el clasificador específico los resultados obtenidos no son lo suficientemente buenos. Las medidas de calidad pueden considerarse buenas, puesto que superan en todos sus casos el 60%, pero la curva ROC mostrada en la figura 11.43 indica que el clasificador específico es prácticamente aleatorio. Es decir, se detecta la presencia de cielo en las imágenes de test de forma aleatoria.

Por tanto a pesar de que las medidas de calidad son buenas, la curva ROC y su correspondiente AUC llevan a concluir que las características de cielo extraídas a las imágenes no identifican correctamente la aparición/ausencia de cielo en ellas.

Resumen:

A continuación se muestra un resumen con los valores de AUC para cada concepto del clasificador específico y entre paréntesis se indica la variación con el clasificador general.

Tabla 11.24 – Resumen de los valores de AUC por concepto del clasificador específico

Exterior/Interior: 55,8809% (+4%)	***Personas: 55,257% (-2,6%)***	***Día/Noche: 67,021% (+12,6%)***
Agua: 51,5405% (-1,4%)	***Vegetación: 49,0544% (-5,9%)***	***Cielo: 50,0902% (-8,7%)***

Máximo – Día/Noche: 67,021% ***Mínimo – Vegetación: 49,0544%***

Media = 56,4443% (+2,3%)

La eficacia del clasificador específico y por tanto los resultados obtenidos para cada concepto dependen totalmente de la fidelidad con las que las características que se extraen para determinar cada uno de ellos representen y diferencien su aparición y su ausencia.

El concepto que peor se clasifica es "Vegetación". El AUC obtenido es inferior al 50%, lo que supone que la determinación de la aparición/ausencia de vegetación en las imágenes no es correcta, realizándose una mala clasificación. De esto se deduce que las características extraídas con el objetivo de determinar vegetación en las imágenes no representan fielmente la aparición de dicho concepto en ellas.

De forma inversa para el concepto "Día/Noche" los resultados obtenidos son muy buenos, superando el AUC el 67%. En este caso las características extraídas permiten al clasificador diferenciar con un alto grado de precisión las imágenes que han sido tomadas de día y de noche.

En el resto de los conceptos evaluados mediante este método se observa que tanto para "Cielo" como para "Agua" los resultados obtenidos no son especialmente buenos. Pueden considerarse aceptables pues su AUC supera el 50% pero el clasificador prácticamente determina su aparición o ausencia de forma aleatoria en ambos. Para los dos conceptos restantes, "Exterior/Interior" y "Personas", la clasificación se considera aceptable, aunque se podrían obtener mejores resultados extrayendo algún tipo de características que definan con más fidelidad estos conceptos en las imágenes.

De forma global se concluye que las características extraídas para cada concepto podrían representarlos más adecuadamente o son insuficientes y se necesita un mayor número de ellas. Para ello cabe notar que aún sistemas de este tipo no están lo suficientemente avanzados para obtener resultados que puedan considerarse excelentes y que es muy complejo determinar mediante un puñado de parámetros un objeto o concepto que aparece con diversas representaciones en las imágenes. Además el introducir un elevado número de características para representar un concepto supondría un alto coste computacional, lo que chocaría con uno de los objetivos de implementar este método: obtener buenos resultados utilizando un número no demasiado elevado de características para disminuir el coste computacional del sistema.

Aún así los resultados obtenidos mediante la implementación del método específico son admisibles, ya que en 5 de los 6 conceptos evaluados la curva ROC supera la línea de aleatoriedad y en tres de ellos el AUC supera el 55%.

11.3.3 Resultados del Clasificador K-NN Combinado

Exterior/Interior:

Al igual que en los dos métodos anteriores se han representado dichos conceptos como uno solo, representando como positivo exterior y como negativo interior.

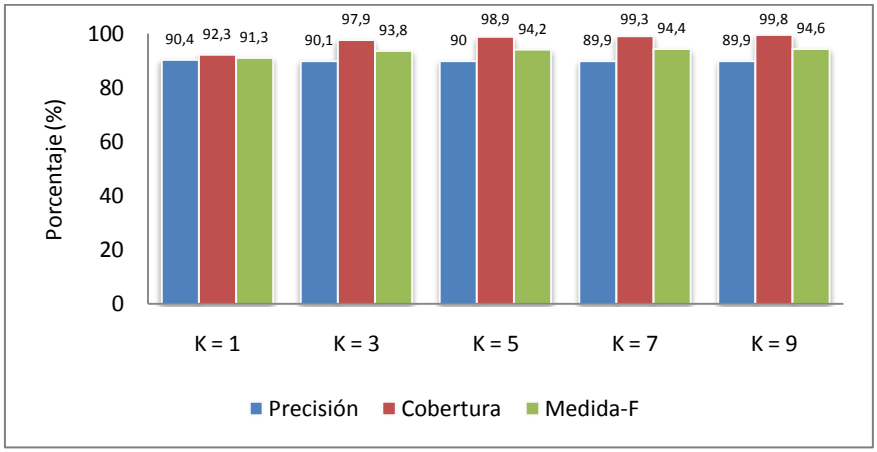

Figura 11.44 – Medidas de calidad en función de K del clasificador combinado para "Exterior/Interior"

Tabla 11.25 – Medidas de calidad del clasificador combinado con K=9 del concepto "Exterior/Interior"

Matriz de Confusión	
TP = 892	***FP = 100***
FN = 2	***TN = 0***

Precisión = 89,9%
Cobertura = 99,8%
Medida-F = 94,6%

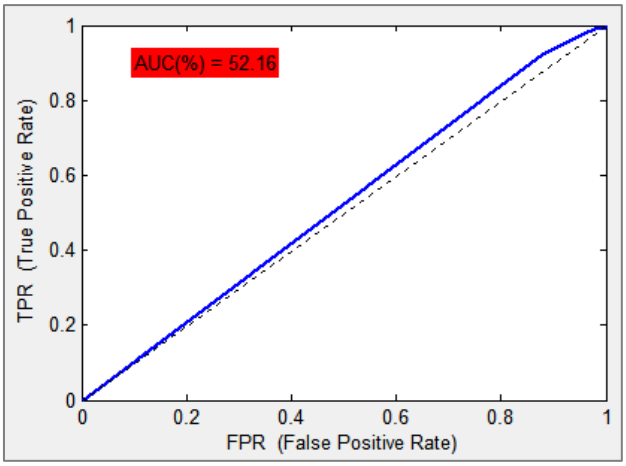

Figura 11.45 – Curva ROC y AUC del clasificador combinado del concepto "Exterior/Interior"

Dado que los resultados obtenidos en las medidas de calidad para este concepto tanto en el clasificador general como en el específico eran excelentes, como cabía esperar los obtenidos en la combinación de ambos continúan siéndolo.

El área bajo la curva ROC continúa también en valores no demasiado buenos debido a que al igual que ocurría en los métodos general, el clasificador detecta con gran precisión las imágenes tomadas en exteriores pero no las tomadas en interiores.

Personas:

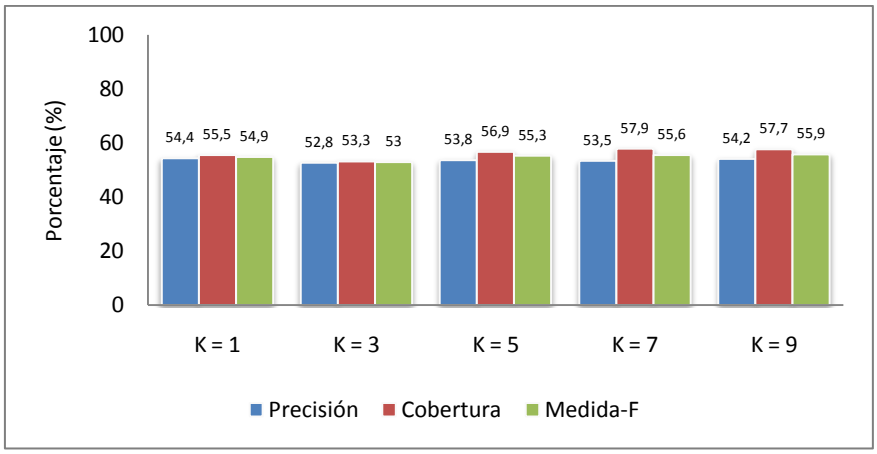

Figura 11.46 – Medidas de calidad en función de K del clasificador combinado para "Personas"

Tabla 11.26 – Medidas de calidad del clasificador combinado con K=9 del concepto "Personas"

Matriz de Confusión	
TP = 289	***FP = 244***
FN = 212	***TN = 249***

Precisión = 54,2%
Cobertura = 57,7%
Medida-F = 55,9%

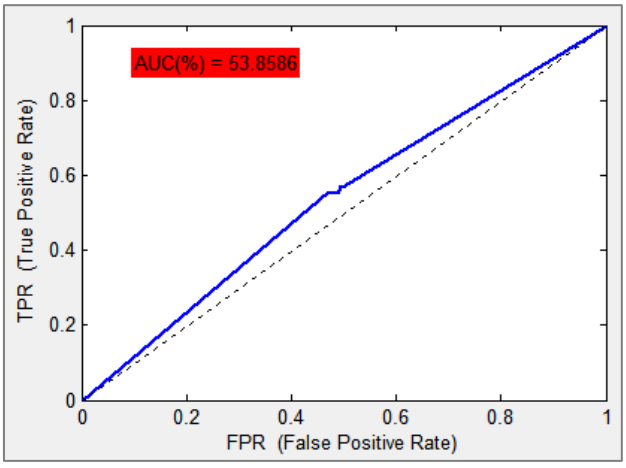

Figura 11.47 – Curva ROC y AUC del clasificador combinado del concepto "Personas"

Al igual que para el concepto "Exterior/Interior" los resultados obtenidos se encuentran en valores intermedios a los del método general y específico. Se debe principalmente a que las características específicas extraídas para determinar este concepto son lo suficientemente significativas para mejorar los resultados que se obtenían con las características generales.

Así se considera que se realiza una buena clasificación gracias a las mejoras que introducen las características específicas sobre las generales.

Día/Noche:

Al igual que ocurría con los conceptos "Exterior/Interior" los ahora evaluados se han considerado como un único concepto considerándose como *'positivo'* a "Día" y como *'negativo'* a "Noche".

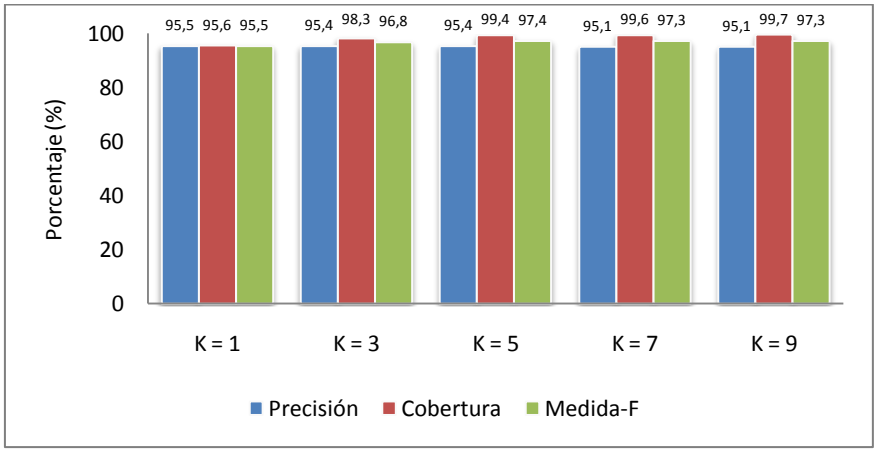

Figura 11.48 – Medidas de calidad en función de K del clasificador combinado para "Día/Noche"

Tabla 11.27 – Medidas de calidad del clasificador combinado con K=5 del concepto "Día/Noche"

Matriz de Confusión	
TP = 933	***FP = 45***
FN = 6	***TN = 10***

Precisión = 95,4%
Cobertura = 99,4%
Medida-F = 97,4%

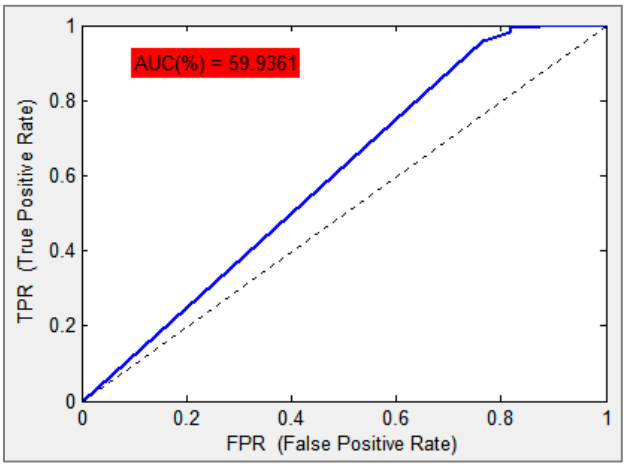

Figura 11.49 – Curva ROC y AUC del clasificador combinado para el concepto "Día/Noche"

De nuevo para el concepto "Día/Noche" los resultados obtenidos son prácticamente un promedio de los obtenidos con los métodos general y específico.

Cabe notar que en el método general el número de imágenes detectadas correctamente como de noche era muy pequeño, mientras que en este caso es mayor. Esto se debe gracias a la combinación con las características específicas, las cuales permiten diferenciar con un mayor grado las imágenes tomadas de día y de noche.

Agua:

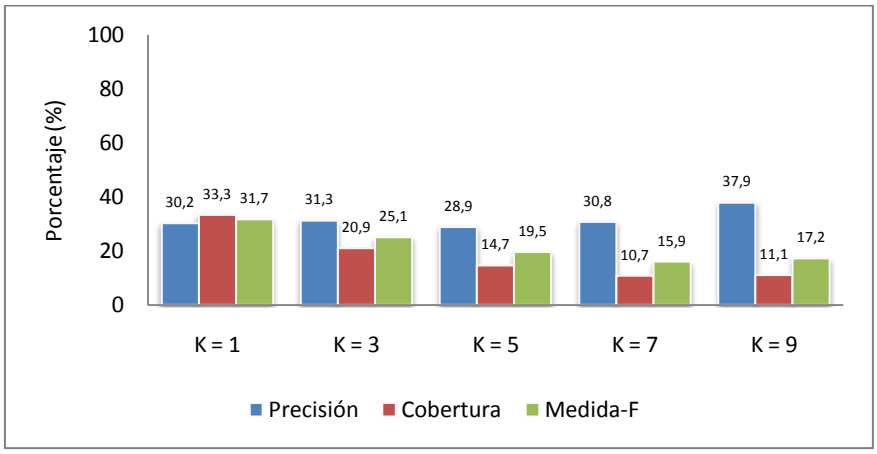

Figura 11.50 – Medidas de calidad en función de K del clasificador combinado para "Agua"

Tabla 11.28 – Medidas de calidad del clasificador combinado con K=1 del concepto "Agua"

Matriz de Confusión	
TP = 75	***FP = 173***
FN = 150	***TN = 596***

Precisión = 30,2%
Cobertura = 33,3%
Medida-F = 31,7%

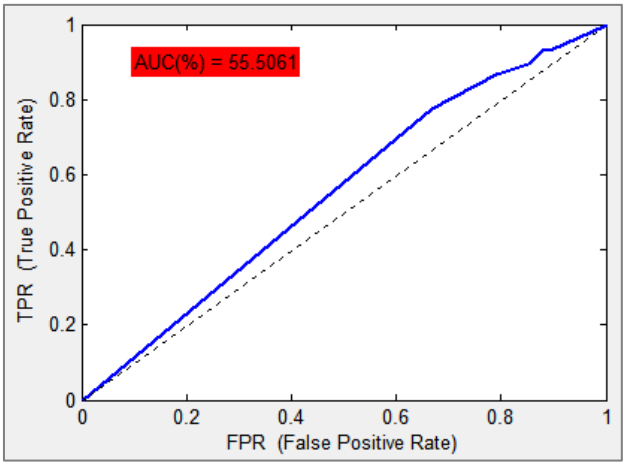

Figura 11.51 – Curva ROC y AUC del clasificador combinado para el concepto "Agua"

Tanto los resultados de precisión, cobertura y medida-F como la curva ROC y el área bajo la misma obtenidos para el método combinado superan a los obtenidos en los métodos anteriores. Por tanto se concluye que la combinación de ambos métodos mejora la clasificación de las imágenes para el concepto "Agua".

Aunque las medidas de calidad obtenidas superan la de los métodos que se combinan, no son lo suficientemente buenas. A pesar de ello se obtienen un AUC que supera el 55% mientras que en el método general y específico se aproximaba a la aleatoriedad. Debido a esto la determinación de la aparición/ausencia de agua en las imágenes es bastante buena.

Caminos:

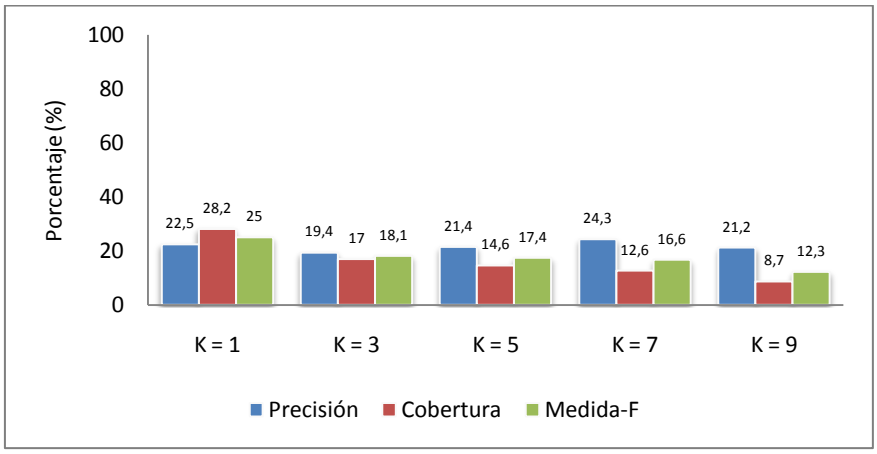

Figura 11.52 – Medidas de calidad en función de K del clasificador combinado para "Caminos"

Tabla 11.29 – Medidas de calidad del clasificador combinado con K=1 del concepto "Caminos"

Matriz de Confusión	
TP = 58	***FP = 200***
FN = 148	***TN = 588***

Precisión = 22,5%
Cobertura = 28,2%
Medida-F = 25%

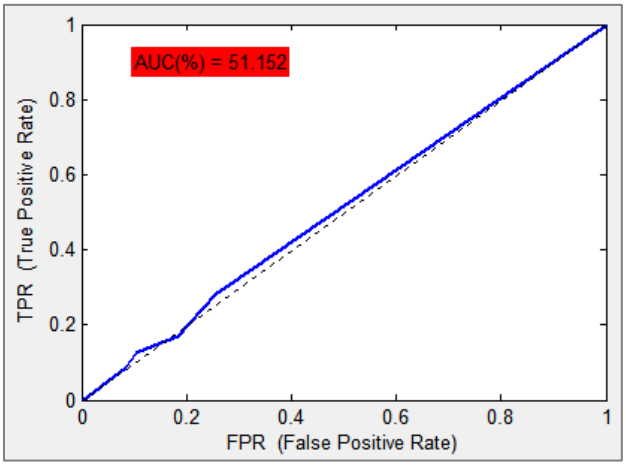

Figura 11.53 – Curva ROC y AUC del clasificador combinado para el concepto "Caminos"

Tal y como se observa en la figura 11.52 y la figura 11.10 correspondiente al clasificador general, para valores de *K* mayores a 5 los resultados de las medidas de calidad son similares. Esto se debe a que para este concepto no se combinan las características generales con las específicas (pues no se han implementado estas últimas para el concepto "Caminos"). Por ello se obtienen los mismos resultados cuando *K* toma valores altos, mientras que los resultados son algo diferentes para valores bajos de *K* debido a las restricciones con la aparición o ausencia de otros conceptos que establece la tarea y se comentaron en apartados anteriores. Dichas restricciones incluso empeoran los resultados que se obtenían con el método general.

La conclusión es similar a la del método general: la clasificación realizada para "Caminos" es prácticamente aleatoria.

Vegetación:

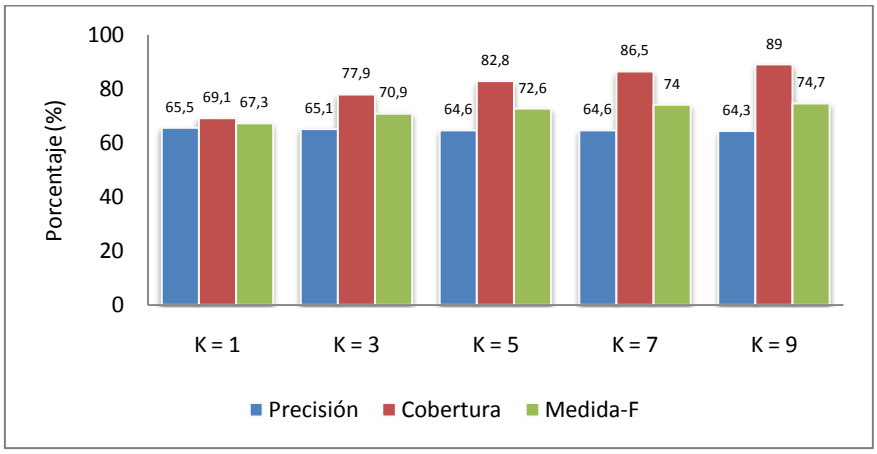

Figura 11.54 –Medidas de calidad en función de K del clasificador combinado para "Vegetación"

Tabla 11.30 – Medidas de calidad del clasificador combinado con K=9 del concepto "Vegetación"

Matriz de Confusión	
TP = 553	***FP = 307***
FN = 68	***TN = 66***

Precisión = 64,3%
Cobertura = 89%
Medida-F = 74,7%

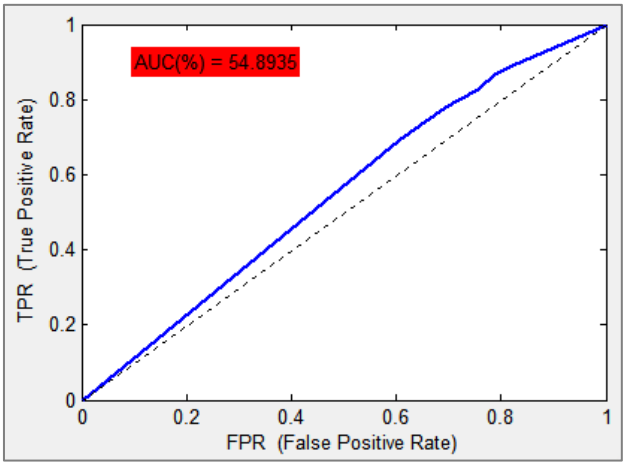

Figura 11.55 – Curva ROC y AUC del clasificador combinado para el concepto "Vegetación"

Notando que los resultados obtenidos en el clasificador específico son considerablemente malos, la combinación de dicho método y el general no reduce los resultados del clasificador general en gran medida como cabía esperar.

Así las medidas de calidad obtenidas en este método mejoran incluso las del método general, y el AUC no se ve afectado en gran medida por las malas características específicas.

Por tanto se realiza una buena clasificación para este concepto a pesar de que para el mismo concepto con las características específicas la clasificación realizada no llegaba a ser aceptable.

Árboles:

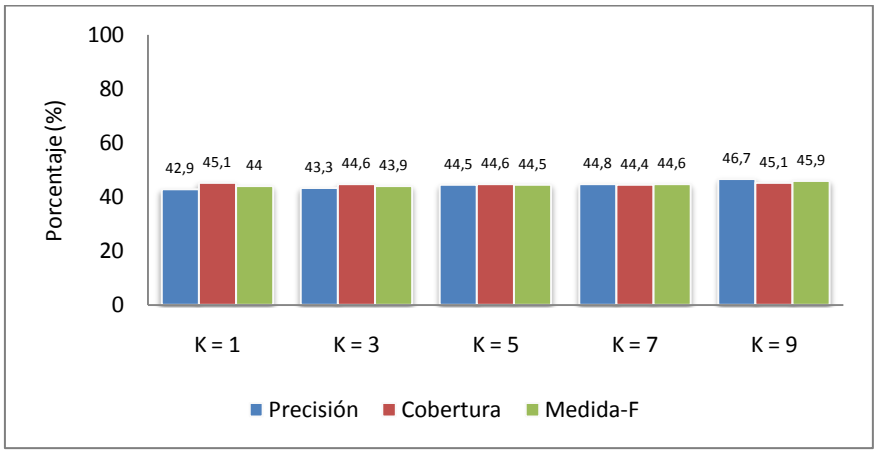

Figura 11.56 –Medidas de calidad en función de K del clasificador combinado para "Árboles"

Tabla 11.31 – Medidas de calidad del clasificador combinado con K=9 del concepto "Árboles"

Matriz de Confusión	
TP = 184	***FP = 210***
FN = 224	***TN = 376***

Precisión = 46,7%
Cobertura = 45,1%
Medida-F = 45,9%

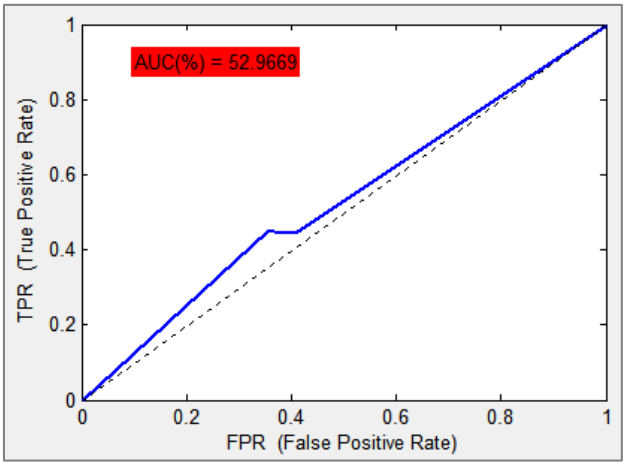

Figura 11.57 – Curva ROC y AUC del clasificador combinado para el concepto "Árboles"

Para el concepto "Árboles" no se obtienen características específicas, pero sus resultados se ven afectados al utilizar estas en el concepto "Vegetación", ya que la ausencia de este restringe la posible aparición de árboles en las imágenes.

Por esto la variación de los resultados con respecto al método general es mínima, mejorándose el AUC pero en valores mínimos. Esto se puede observar en la figura 11.57 y 11.15, donde ambas curvas ROC son prácticamente similares.

Así la conclusión es similar a la del método general, se detecta con mayor precisión la ausencia de árboles que la presencia de ellos en las imágenes.

Montañas:

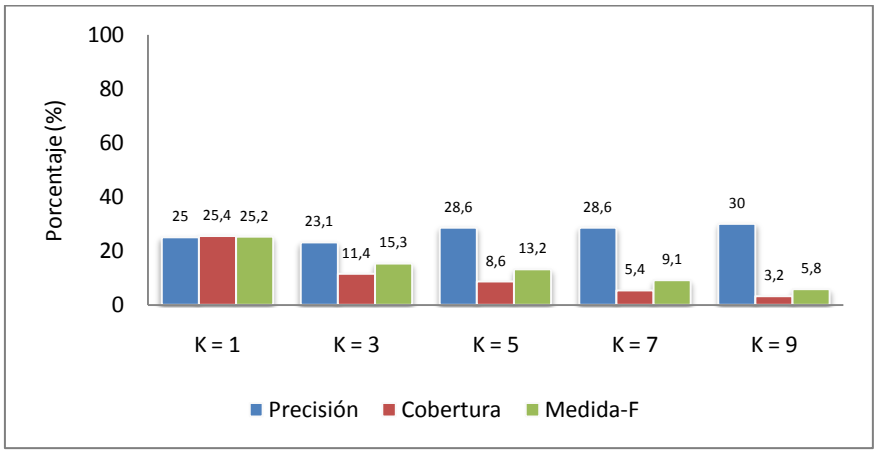

Figura 11.58 –Medidas de calidad en función de K del clasificador combinado para "Montañas"

Tabla 11.32 – Medidas de calidad del clasificador combinado con K=1 del concepto "Montañas"

Matriz de Confusión	
TP = 47	***FP = 141***
FN = 138	***TN = 668***

Precisión = 24,9%
Cobertura = 25,9%
Medida-F = 25,4%

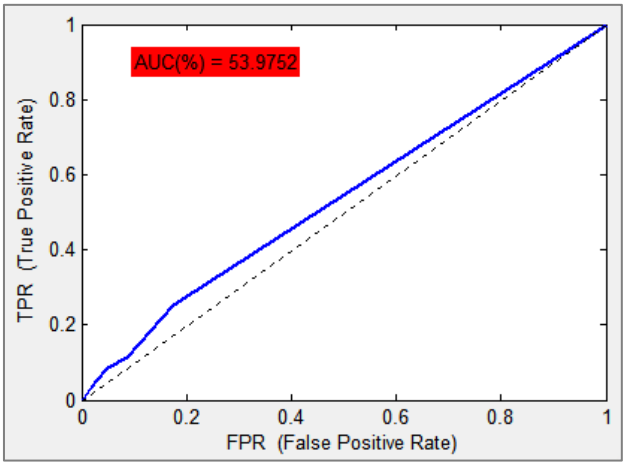

Figura 11.59 – Curva ROC y AUC del clasificador combinado para el concepto "Montañas"

De forma similar al concepto "Árboles" la imposibilidad de aparición de montañas en imágenes tomadas en interiores provoca que a pesar de no haber implementado características específicas para este concepto los resultados obtenidos tengan pequeñas variaciones con respecto a los del método general.

Así las medidas de calidad continúan siendo no demasiado buenas y disminuyendo según aumenta el valor de *K* y el área bajo la curva ROC continua siendo aceptable a pesar de ser menor que en el método general.

Playa:

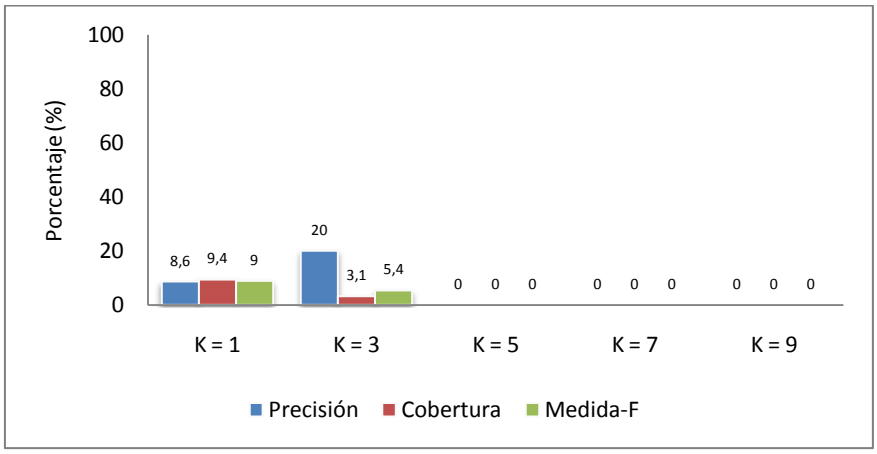

Figura 11.60 –Medidas de calidad en función de K del clasificador combinado para "Playa"

Tabla 11.33 – Medidas de calidad del clasificador combinado con K=1 del concepto "Playa"

Matriz de Confusión	
TP = 3	***FP = 32***
FN = 29	***TN = 930***

Precisión = 8,6%
Cobertura = 9,4%
Medida-F = 9%

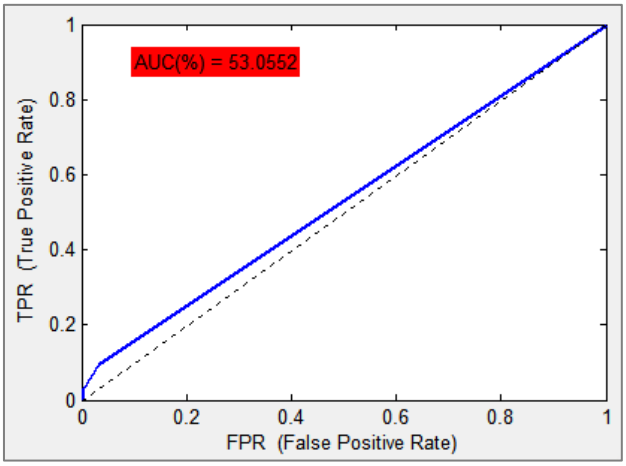

Figura 11.61 – Curva ROC y AUC del clasificador combinado para el concepto "Playa"

Al igual que los dos conceptos evaluados anteriormente, a pesar de no utilizar características específicas para la detección de playa en las imágenes, los resultados se ven modificados ínfimamente debido a las restricciones que establece la tarea propuesta.

Dicha variación es mínima y mejora los resultados del clasificador general, pero es imposible de determinar si a priori mejorarían o empeorarían ya que no se debe intrínsecamente a este concepto si no a la precisión del clasificador en otros conceptos.

Edificios:

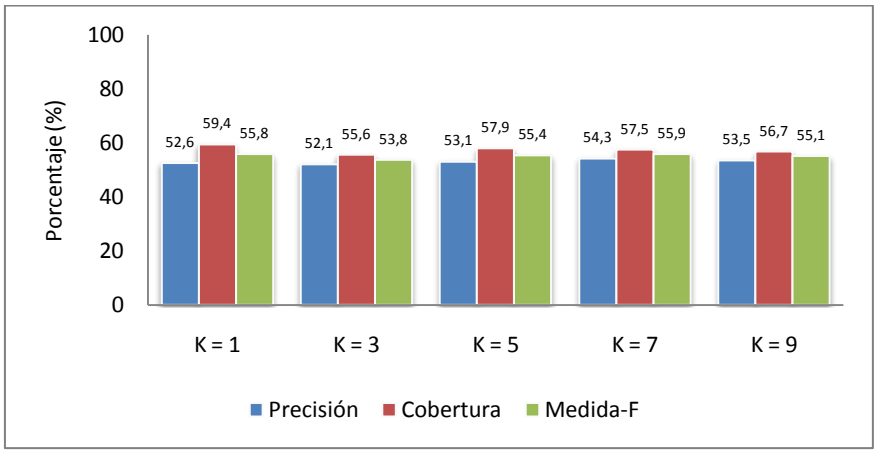

Figura 11.62 – Medidas de calidad en función de K del clasificador combinado para "Edificios"

Tabla 11.34 – Medidas de calidad del clasificador combinado con K=7 del concepto "Edificios"

Matriz de Confusión	
TP = 275	***FP = 231***
FN = 203	***TN = 285***

Precisión = 54,3%
Cobertura = 57,5%
Medida-F = 55,9%

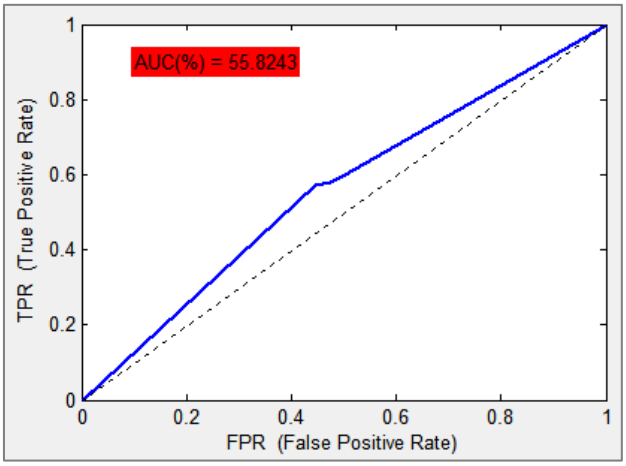

Figura 11.63 – Curva ROC y AUC del clasificador combinado para el concepto "Edificios"

Para la determinación de edificios en las imágenes tampoco se obtienen características específicas, pero en este caso las restricciones de la tarea no afectan ni positiva ni negativamente a la clasificación de este concepto.

Por tanto los resultados son exactamente los mismos que se obtenían en el clasificador general, donde las medidas de calidad son admisibles pues superan el 50% en todos sus casos al igual que ocurre con el área bajo la curva ROC. Así la clasificación de este concepto en el método combinado es similar a la del método general y por tanto es también aceptable.

Animales:

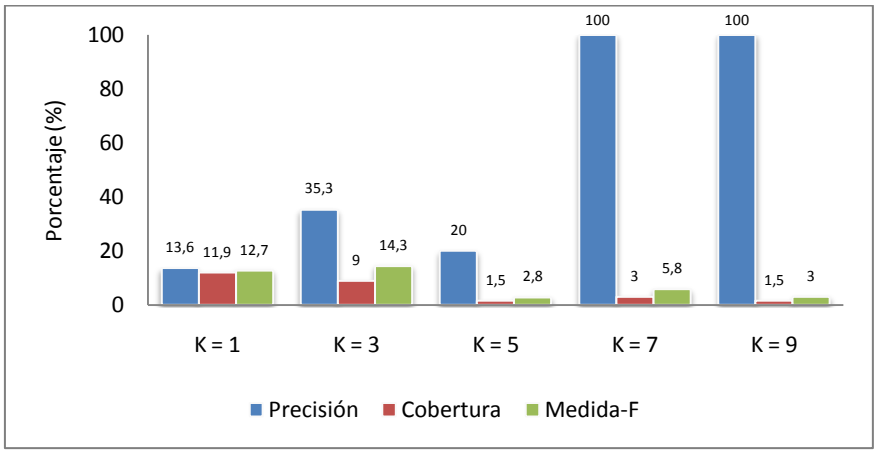

Figura 11.64 –Medidas de calidad en función de K del clasificador combinado para "Animales"

Tabla 11.35 – Medidas de calidad del clasificador combinado con K=3 del concepto "Animales"

Matriz de Confusión	
TP = 6	***FP = 11***
FN = 61	***TN = 916***

Precisión = 35,3%
Cobertura = 9%
Medida-F = 14,3%

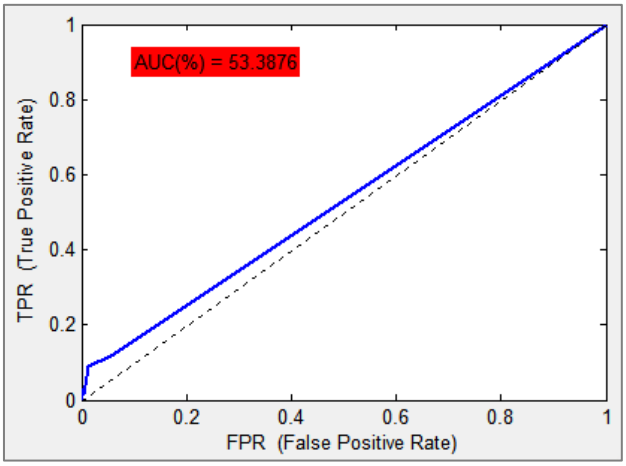

Figura 11.65 – Curva ROC y AUC del clasificador combinado para el concepto "Animales"

De forma similar a lo que ocurre en el concepto "Edificios", las restricciones de la tarea no afectan de forma ninguna a la hora de detectar la presencia o ausencia de animales, pues ninguno de todos los conceptos evaluados restringe la aparición de este en las imágenes.

Así los resultados obtenidos por el método combinado son exactamente similares a los del método general, donde el clasificador a pesar de clasificar con muy baja precisión la presencia de animales, detecta con muy buena precisión la ausencia de estos, provocando que la clasificación tal y como muestra el AUC sea admisible.

Cielo:

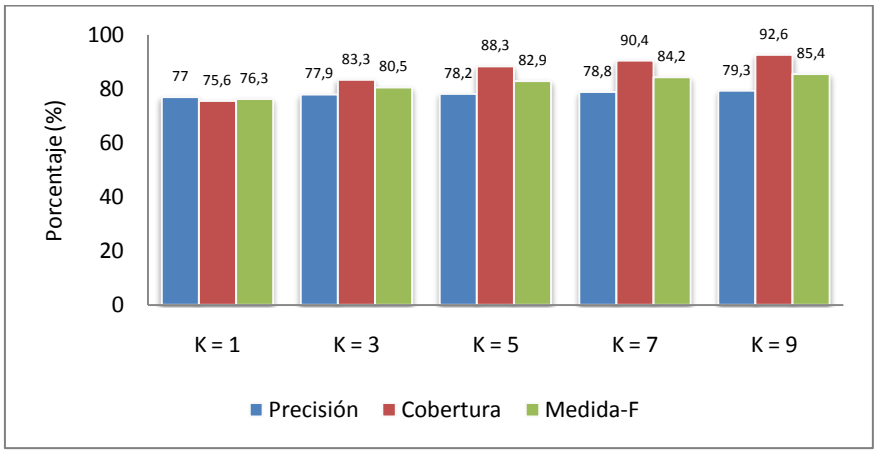

Figura 11.66 –Medidas de calidad en función de K del clasificador combinado para "Cielo"

Tabla 11.36 – Medidas de calidad del clasificador combinado con K=9 del concepto "Cielo"

Matriz de Confusión	
TP = 659	***FP = 172***
FN = 53	***TN = 110***

Precisión = 79,3%
Cobertura = 92,6%
Medida-F = 85,4%

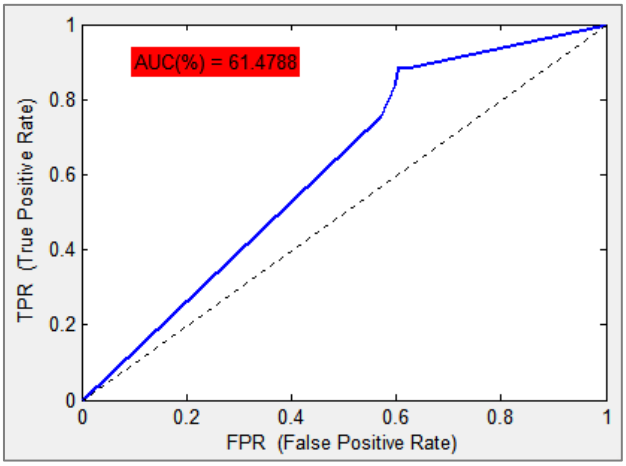

Figura 11.67 – Curva ROC y AUC del clasificador combinado para el concepto "Cielo"

Las medidas de calidad obtenidas para el método combinado son aún mejores que las del método general y el específico. De tal forma que se obtienen una precisión y cobertura bastante altas, por lo que se detecta con un alto grado de precisión la presencia de cielo en las imágenes.

Además la ausencia del concepto evaluado se predice correctamente en un porcentaje bastante alto, lo que combinado con lo anteriormente descrito genera una curva ROC con un AUC superior al 60% y mayor que el de los métodos que se combinan.

La combinación de las características generales y las específicas de cielo conlleva a la obtención de unos resultados excelentes. Por tanto la clasificación realizada con el método combinado es muy buena, mejor que para los métodos general y específico.

Soleado:

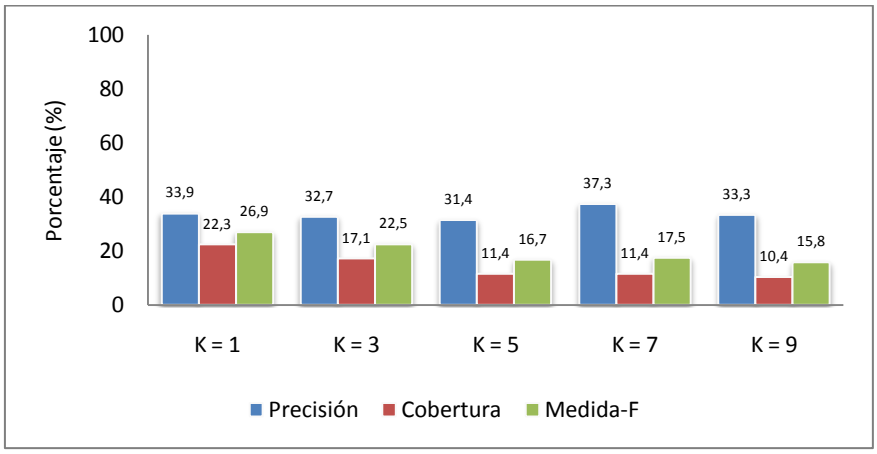

Figura 11.68 –Medidas de calidad en función de K del clasificador combinado para "Soleado"

Tabla 11.37 – Medidas de calidad del clasificador combinado con K=1 del concepto "Soleado"

Matriz de Confusión	
TP = 43	***FP = 84***
FN = 150	***TN = 717***

Precisión = 33,9%
Cobertura = 22,3%
Medida-F = 26,9%

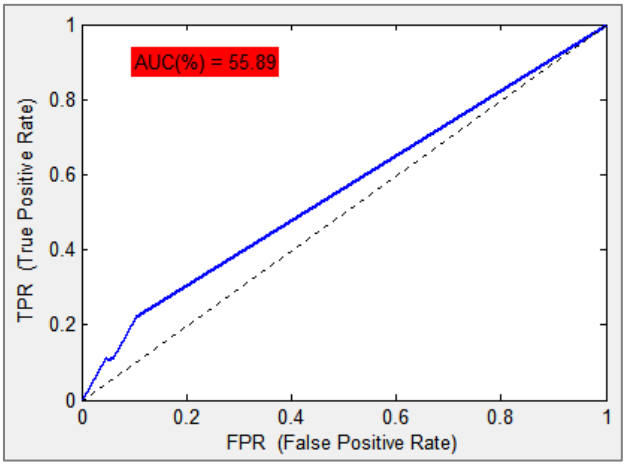

Figura 11.69 – Curva ROC y AUC del clasificador combinado para el concepto "Soleado"

La restricción que supone la aparición/ausencia de cielo en las imágenes supone que los resultados al evaluar el concepto "Soleado" varíen con respecto al método general a pesar de que la implementación es la misma puesto que no se extraen características específicas para este concepto.

Dicha restricción conlleva a que disminuyan en pequeña medida tanto las medidas de calidad como la curva ROC y el área bajo la misma. Aunque los resultados de este clasificador es posible que sean más correctos que los del general puesto que la clasificación del concepto "Cielo" que es el que restringe a este es mucho mejor en el caso combinado que en el general.

Parcialmente Nublado:

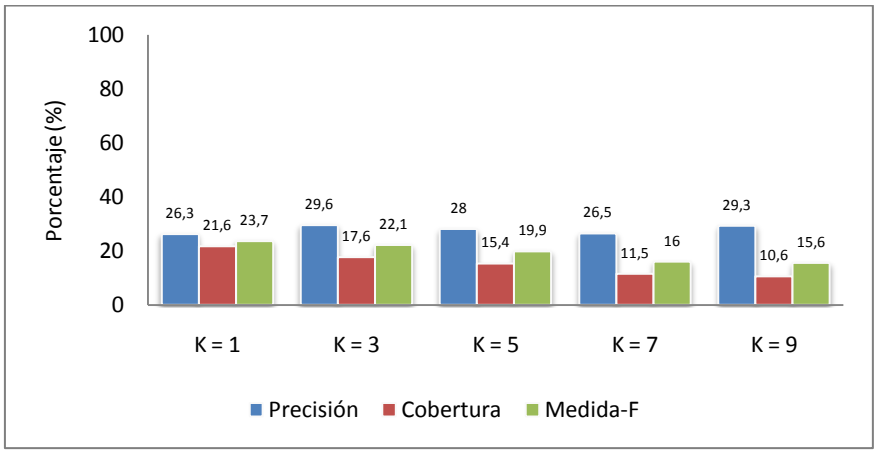

Figura 11.70 – Evolución de las medidas de calidad en función de K del concepto "P.Nublado"

Tabla 11.38 – Medidas de calidad del clasificador combinado con K=1 del concepto "P.Nublado"

Matriz de Confusión	
TP = 49	***FP = 137***
FN = 178	***TN = 630***

Precisión = 26,3%
Cobertura = 21,6%
Medida-F = 23,7%

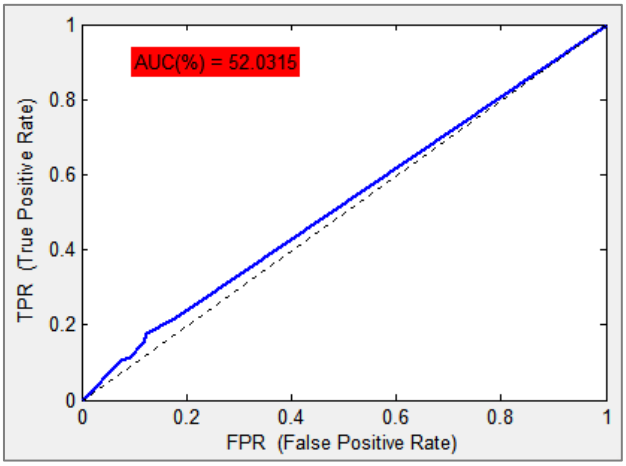

Figura 11.71 – Curva ROC y AUC del clasificador combinado para el concepto "P.Nublado"

Al igual que ocurre con el tipo de cielo soleado, para el tipo de cielo parcialmente nublado los resultados se ven modificados en cierta forma debido a que la clasificación de cielo es diferente para el método combinado que para el método general.

Así los resultados son menores en el clasificador combinado que en el general, pero la diferencia es tan pequeña que prácticamente se puede decir que no se introducen cambios y se clasifica de forma similar en ambos métodos.

Por tanto, tal y como muestra la curva ROC, se concluye que la detección de cielos parcialmente nublados en las imágenes es aceptable pero podría ser mejor incluyendo algún tipo de parámetros específicos.

Nublado:

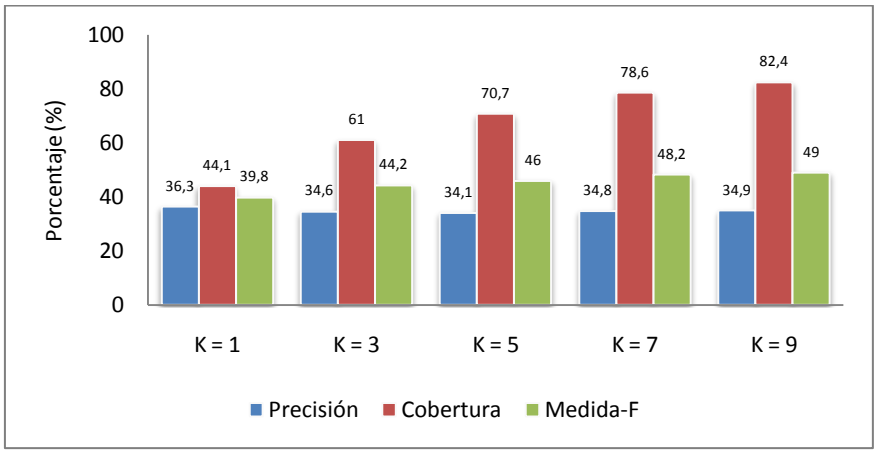

Figura 11.72 –Medidas de calidad en función de K del clasificador combinado para "Nublado"

Tabla 11.39 – Medidas de calidad del clasificador combinado con K=9 del concepto "Nublado"

Matriz de Confusión	
TP = 239	***FP = 445***
FN = 51	***TN = 259***

Precisión = 34,9%
Cobertura = 82,4%
Medida-F = 49%

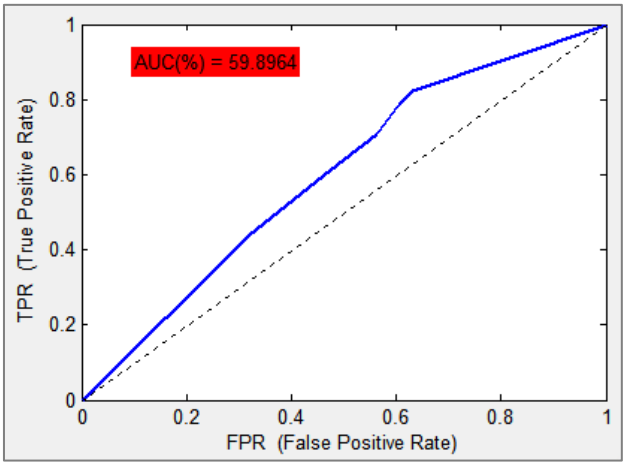

Figura 11.73 – Curva ROC y AUC del clasificador combinado para el concepto "Nublado"

De forma inversa a los dos tipos de cielo evaluados anteriormente, para el caso de cielos nublados las medidas de calidad obtenidas mediante el clasificador combinado son algo mejores que las del clasificador general.

El aumento de dichos valores no es demasiado significativo, aumentando el número de verdaderos positivos pero también el número de falsos positivos. Por tanto en el caso combinado el clasificador tiende con un mayor porcentaje a determinar que el tipo de cielo de las imágenes es nublado.

Esto provoca que aumente el área bajo la curva ROC, de lo que se deduce que se realiza una buena clasificación para este concepto.

Resumen:

A continuación se muestra un resumen con los valores de AUC de cada concepto del algoritmo combinado. Entre paréntesis se indica la variación con el método general en primer lugar y con el método específico en los conceptos en los cuales se implementa en segundo lugar.

Tabla 11.40 – Resumen de los valores de AUC por concepto del clasificador combinado

Ext/Int: 52,16% (+0,3%)(-3,7%)	***Personas: 53,8586% (-4%)(-1,4%)***	***Día/Noche: 59,9361% (+5,4%)(-7,1%)***
Agua: 55,5061% (+2,7%)(+4%)	***Caminos: 51,152% (-0,6%)***	***Vegetación: 54,8935% (-0,1%)(+5,8%)***
Árboles: 52,9669% (+0,1%)	***Montañas: 53,9752% (-0,1%)***	***Playa: 53,0552% (+0,1%)***
Edificios: 55,8243% (0%)	***Animales: 53,3876% (0%)***	***Cielo: 61,4788% (+2,7%)(+11,4%)***
Soleado: 55,89% (-0,3%)	***P.Nublado: 52,0315% (-0,6%)***	***Nublado: 59,8964% (+1,3%)***

Máximo – Cielo: 61,4788% ***Mínimo – Caminos: 51,152%***

Media = 55,1828% (+1%)(-1,3%)

Antes de evaluar resumidamente la calidad del clasificador combinado cabe destacar que las características específicas solo se han extraído para 6 de los 15 conceptos, pero se han evaluado el resto debido a las restricciones entre conceptos que impone la tarea propuesta por ImageCLEF 2008. Por ejemplo para los conceptos árboles, montañas o playa, a pesar de que no se implementa la parte específica se observan cambios en los resultados con respecto al método general.

Al igual que ocurre en el método general el peor AUC se obtiene en la detección de caminos en las imágenes. Donde no por mucho se supera el 50% y por tanto se puede decir que se clasifica de forma aleatoria la aparición/ausencia de caminos.

El máximo corresponde con el concepto "Cielo", superando un valor del 60%. Por lo que en este caso la combinación de tanto las características generales como las específicas supone un aumento considerable y por tanto una mejor clasificación para este concepto.

De forma global para la mayoría de conceptos se obtienen valores mayores de AUC que en los métodos general y específico, aumentando por tanto también la media obtenida. Así la precisión del clasificador combinado es bastante buena para todos los conceptos evaluados.

11.4 Comparación de los clasificadores General, Específico y Combinado

La tabla 11.41 muestra un resumen de los resultados que se obtienen para cada uno de los conceptos con los diferentes clasificadores implementados. En ella se observan, para cada concepto y tipo de clasificador, el valor de *K* que proporciona las mejores medidas de calidad empleadas (precisión, cobertura y medida-F) y dichas medidas. Se encuentra sombreado en azul el clasificador que mejor resultados proporciona en cada concepto.

Detallado el contenido de la tabla, en ella se observa que el clasificador combinado es el mejor para la mayoría de los conceptos, excepto para: "exterior/interior", "personas", "caminos", "soleado" y "parcialmente nublado", con respecto a los cuales la diferencia no es demasiado considerable. Así el clasificador específico es el mejor para el concepto "personas", mientras que el clasificador general lo es para el resto.

Tabla 11.41 – Resultados de los clasificadores general, específico y combinado por concepto

	GENERAL				ESPECÍFICO				COMBINADO			
	K	Prec.	Cob.	Medida-F	K	Prec.	Cob.	Medida-F	K	Prec.	Cob.	Medida-F
Ext/Int	7	90%	99,9%	94,7%	9	89,9%	99,8%	94,6%	9	89,9%	99,8%	94,6%
Personas	9	53,8%	54,9%	54,3%	7	55,7%	66,7%	60,7%	9	54,2%	57,7%	55,9%
Día/Noche	9	94,5%	100%	97,2%	5	95,8%	99%	97,4%	5	95,4%	99,4%	97,4%
Agua	1	26%	29,8%	27,8%	1	25,7%	25,3%	25,5%	1	30,2%	33,3%	31,7%
Caminos	1	23,2%	30,6%	26,4%	-	-	-	-	1	22,5%	28,2%	25%
Vegetación	9	64,3%	88,9%	74,6%	9	21,3%	82%	33,8%	9	64,3%	89%	74,7%
Árboles	9	46,7%	45,1%	45,9%	-	-	-	-	9	46,7%	45,1%	45,9%
Montañas	1	24,9%	25,9%	25,4%	-	-	-	-	1	24,9%	25,9%	25,4%
Playa	1	8,1%	9,4%	8,7%	-	-	-	-	1	8,6%	9,4%	9%
Edificios	7	54,3%	57,5%	55,9%	-	-	-	-	7	54,3%	57,5%	55,9%
Animales	3	35,3%	9%	14,3%	-	-	-	-	3	35,3%	9%	14,3%
Cielo	9	75,9%	90,2%	82,4%	9	63,4%	87,7%	73,6%	9	79,3%	92,6%	85,4%
Soleado	1	29,9%	27,5%	28,6%	-	-	-	-	1	33,9%	22,3%	26,9%
P.Nublado	1	26,6%	26%	26,3%	-	-	-	-	1	26,3%	21,6%	23,7%
Nublado	9	32,9%	79%	46,5%	-	-	-	-	9	34,9%	82,4%	49%

11.5 Comparación de resultados con los participantes de ImageCLEF 2008

Para referenciar y comparar los resultados obtenidos en este proyecto con los obtenidos por los participantes en la tarea Visual Concepto Detection Task (VCDT) de ImageCLEF 2008 se muestra la tabla 11.42, donde se aprecia la media del AUC de todos los conceptos para cada uno de los métodos implementados por los participantes y los tres métodos desarrollados en este proyecto (***UC3M-JRP***).

Tal y como muestra la tabla anteriormente mencionada, los tres métodos implementados en este proyecto se encuentran posicionados algo por debajo de la mitad de los participantes, en concreto con un 10% aproximadamente menos de AUC medio. Los resultados no pueden por tanto considerarse excelentes pero sí como buenos, pues se supera el umbral del 50% y también se supera con creces a los peores métodos de los participantes.

Con un total de 56 métodos implementados por los participantes y contando los empleados en este proyecto, el mejor de los aquí desarrollados es el método específico, situado en la posición 38. El método combinado se sitúa en la posición 41 y finalmente el que peor media ofrece de los tres, el método general, se sitúa en la posición 43. Todos ellos se encuentran muy cercanos puesto que sus AUC medios son muy similares, separados entre el mejor y el peor por tan solo un 2% aproximadamente.

Por tanto se concluye que con una implementación no demasiado compleja en cada uno de los tres métodos, tanto en los bloques de extracción de características como en los clasificadores, se obtienen unos resultados aceptables y competitivos con respecto a los de los participantes de la tarea VCDT de ImageCLEF 2008.

Tabla 11.42 – Posición de los métodos empleados respecto a los participantes de la tarea VCDT

Pos.	Grupo	Método	AUC(%)
1	XRCE	TVPA_XRCE_KNN	90.66
2	XRCE	TVPA_XRCE_LIN	88.73
3	RWTH	PHME	86.19
4	UPMC	B50trees100pc	82.74
5	LSIS	MLP1_LSIS_GLOT	80.67
6	LSIS	MLP1_vcdt_LSISI	80.67
7	LSIS	New_kda_results	80.51
8	LSIS	GLOT-metode23_LSIS	79.92
9	LSIS	Method2_LSIS	79.75
10	MMIS	Ainhoa	77.94
11	MMIS	Alexei	77.65
12	IPAL_I2R	I2R_IPAL_FuseNMCE	76.44
13	Budapest	Acad-budapest-glob2	74.90
14	Budapest	Acad-acad-mednofi	74.18
15	IPAL_I2R	I2R_IPAL_FuseMCE	74.05
16	IPAL_I2R	I2R_IPAL_Hist	73.80
17	MMIS	CombinedReplacement	73.69
18	Budapest	Acad-acad-medppnnpnn	73.61
19	Budapest	Acad-acad-medfi	73.57
20	CEA_LIST	CEA_LIST_4	73.40
21	Budapest	Acad-budapest-lownfi	73.32
22	Budapest	Acad-acad-lowppnnpnn	73.05
23	Budapest	Acad-budapest-lowfi	73.03
24	IPAL_I2R	I2R_IPAL_model	72.01
25	UPMC	B50trees100C5N5	71.98
26	UPMC	B50trees100pc_COOC5	71.58
27	CEA_LIST	CEA_LIST_2	71.44
28	IPAL_I2R	I2R_IPAL_Linear	68.65
29	Budapest	Acad-acad-lowppnn	67.15
30	Budapest	Acad-acad-logreg2	66.53
31	Budapest	Acad-acad-logreg1	66.39
32	Budapest	Acad-acad-mixed	63.80
33	IPAL_I2R	I2R_IPAL_Texture	62.93
34	IPAL_I2R	I2R_IPAL_Cor	62.62
35	MMIS	MMIS_Ruihu	62.50
36	Budapest	Acad-acad-medppnn	59.30
37	UPMC	B50trees100pc_T25	57.09
38	**UC3M-JRP**	**Específico**	**56.44**
39	IPAL_I2R	I2R_IPAL_Edge	55.79
40	TIA	INAOE-psms-00_HJ_TIA	55.64
41	**UC3M-JRP**	**Combinado**	**55.18**
42	UPMC	B50trees100COOC5T25	54.19
43	**UC3M-JRP**	**General**	**54.12**
44	UPMC	B50trees100C5N5T25	53.78
45	Budapest	Acad-budapest-glob1	52.78
46	LSIS	FusionA_LSIS	50.84
47	LSIS	FusionH_LSIS	50.20
48	TIA	INAOE-psms-02_HJ_TIA	47.07
49	TIA	INAOE-1b-01_HJ_TIA	42.15
50	TIA	INAOE-rf-00_HJ_TIA	36.11
51	CEA_LIST	CEA_LIST_3	34.25
52	Makere	MAK	30.83
53	TIA	INAOE-kr-00_HJ_TIA	28.90
54	TIA	INAOE-rf-03_HJ_TIA	26.37
55	HJ_FA	HJ_Result	19.96
56	TIA	INAOE-kr-04_HJ_TIA	17.58

PARTE V:

PRESUPUESTO

12. Presupuesto de proyecto

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:

Juan Rodríguez Povedano

2.- Departamento:

Ingeniería Telemática

3.- Descripción del Proyecto:

- Titulo: **Anotación Automática de Imágenes**
- Duración (meses): **14**
- Tasa de costes Indirectos: 20%

4.- Presupuesto total del Proyecto (valores en Euros):

25.444,00 Euros

5.- Desglose presupuestario (costes directos):

PERSONAL					
Apellidos y nombre	N.I.F.	Categoría	Dedicación (personas/mes)[a)]	Coste (personas/mes)	Coste (Euro)
Villena Román, Julio	8042900-F	Ingeniero Senior	0,5	4.289,54	2.144,77
Rodríguez Povedano, Juan	70814713-J	Ingeniero	7	2.694,39	18.860,73
		Personas/ mes 7,5		**Total**	**21.005,50**

[a)] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)

Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS					
Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d)]
Ordenador Portatil	660,00	100	18	60	198,00
				Total	**198,00**

[d)] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} xCxD$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS		
Descripción	Empresa	Coste imputable
	Total	**0,00**

OTROS COSTES DIRECTOS DEL PROYECTO[e)]		
Descripción	Empresa	Coste imputable
	Total	**0,00**

[e)] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,…

6.- Resumen de costes:

COSTES TOTALES	
Descripción	**Presupuesto**
Personal	21.006
Amortización	198
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	4.241
Total	**25.444**

PARTE VI:

CONCLUSIONES Y LÍNEAS FUTURAS

13. Conclusiones y Líneas Futuras

13.1 Conclusiones

En primer lugar cabe destacar que los sistemas de clasificación de imágenes continúan siendo hoy día un reto de investigación. Cada vez se requieren formas más eficientes de tratar diferentes contenidos multimedia tales como imágenes y vídeos. La forma más extendida actualmente es la utilización de etiquetas introducidas por el usuario que describen los conceptos que aparecen en el contenido multimedia, pero sería sin duda un gran avance la utilización de sistemas que automáticamente generen esas etiquetas, tal y como se propone en este proyecto. Esto es ahora posible pero no con la eficacia que requiere, por lo que continuamente se están desarrollando nuevos métodos que buscan mayores precisiones a la hora de determinar el contenido de una imagen o un conjunto de ellas (un vídeo).

La tarea *Visual Concept Detection Task* de ImageCLEF 2008 proponía el desarrollo de un sistema capaz de detectar la aparición y/o ausencia de 17 determinados conceptos en imágenes. Cada uno de los participantes en dicha tarea gozaba de plena libertad para su implementación, y con la misma se ha procedido en este proyecto. Así el desarrollo del sistema propuesto se ha llevado a cabo en las siguientes cuatro etapas:

- Preprocesado
- Extracción de características
- Clasificación
- Evaluación

La primera etapa consiste en la mejora de las características de la imagen, eliminando el posible ruido que pueda aparecer en ella. Por ello se ha implementado una serie de filtros que reducen tanto el ruido impulsivo como el ruido gaussiano. Se considera suficiente la eliminación de estos ruidos porque las imágenes que proporciona la base de datos no contienen demasiado ruido y se presentan con buena calidad.

La etapa de extracción de características consiste en la obtención de uno o varios conjuntos de parámetros que en cierta forma describirán el contenido de la imagen y servirán al sistema para realizar el aprendizaje y la clasificación. Por esto es considerada la etapa más importante, ya que conlleva en un alto porcentaje la obtención de mejores o peores resultados por parte del sistema. Como norma general la relevancia va ligada con la dificultad, y no podía ser diferente en este caso, donde con los métodos que hoy en día existen es muy complejo el describir el contenido de una imagen mediante un conjunto de parámetros. Debido a la complejidad que implica esta etapa en el sistema desarrollado se han utilizado conjuntos de parámetros o descriptores que son considerados básicos, puesto que cubren únicamente características visuales básicas y elementales, tales como descriptores de color y textura. Esto conlleva a que la descripción de la imagen en algunos casos sea algo ambigua, generando que las tasas de acierto por parte del clasificador no sean lo suficientemente buenas. Para mejorar las prestaciones del sistema este sería el punto que en primer lugar habría que abordar.

La tercera etapa se encarga de implementar el clasificador del sistema. La elección del algoritmo de clasificación K-NN como el utilizado se fundamenta en que es una de las técnicas más utilizadas en sistemas de este tipo debido que proporciona buenos resultados y conlleva poca complejidad. En esta etapa se entrenan los clasificadores con las imágenes de entrenamiento y posteriormente se evalúan con las imágenes de test. Notar que para cada concepto a evaluar se ha implementado un clasificador individual pero no independiente, ya que la tarea impone ciertas restricciones en la aparición y/o ausencia de ciertos conceptos según la aparición y/o ausencia de otros. Posiblemente la utilización de otros métodos de clasificación conlleve la mejora de los resultados, pero no en un grado demasiado alto si no se modifica la etapa más relevante, la de extracción de características.

La cuarta y última etapa consiste en la obtención y evaluación de los resultados del sistema. En ella se muestran diversas medidas de calidad con el fin de evaluar la calidad de cada clasificador y del sistema en general. Así los resultados obtenidos en este proyecto sugieren que el sistema desarrollado proporciona unos resultados aceptables y competitivos comparados con los obtenidos por los participantes de la tarea *Visual Concept Detecion Task* de ImageCLEF 2008.

Aparte del objetivo principal del proyecto se ha marcado un objetivo secundario, el cual consiste en determinar si es más eficaz utilizar un gran número de características extraídas de forma general a las imágenes, es decir, independientemente de los conceptos a identificar, utilizar un número considerablemente menor de características específicas según el concepto a determinar o una combinación de ambos métodos.

La conclusión que se puede extraer de los resultados obtenidos para los tres métodos de extracción de características (general, específico y combinado) es que el método más eficaz es el clasificador específico siempre y cuando los parámetros obtenidos sean correctamente elegidos y permitan diferenciar la aparición y ausencia del concepto a determinar. Este método reduce además el coste computacional del sistema cuando el número de conceptos a evaluar por el sistema es pequeño, ya que utiliza un número no muy elevado de características. A medida que aumentan los conceptos que se quiere evaluar aumenta la carga computacional.

El clasificador general genera los peores resultados debido a que las características extraídas proporcionan una descripción subjetiva de la imagen, ya que las características no se obtienen en función de los conceptos que se ha de determinar. Además el coste computacional es alto puesto que se necesitan un gran número de parámetros para poder describir en cierta forma el contenido de la imagen.

Por último, el clasificador combinado utiliza una combinación de ambos métodos, lo que conlleva que se obtengan resultados intermedios. De forma general puede ser el que mejor resultados obtenga, ya que al combinar ambos métodos suaviza los resultados de los conceptos en los que con uno u otro método no se obtenían buenos resultados. A costa de esta mejora general para todos los conceptos, la carga computacional del sistema se eleva en gran medida.

Numéricamente estas conclusiones se observan en el valor del AUC medio obtenido para todos los conceptos de cada clasificador. El AUC medio del método general es de 54,1172%, el del clasificador combinado es de 55,1828% (mejora en aproximadamente un 1% al general) y el del clasificador específico es de 56,4443% (mejorando un 2,3% al método general y un 1,3% al método específico aproximadamente).

Cabe notar que los resultados de los tres métodos implementados son muy parejos y no se encuentran grandes diferencias. Además si la elección de las características a extraer se realiza de manera óptima la eficacia del sistema será mayor.

13.2 Líneas Futuras

Las posibles líneas futuras de trabajo a seguir para mejorar la idea de este proyecto y obtener así un sistema que proporcione una descripción más fiel y real de las imágenes son enumeradas unas para las etapas desarrolladas en este proyecto y otras con carácter general.

Mejoras del preprocesado:

A pesar de que las imágenes de la base de datos con la que se trabaja se encuentran con gran calidad, podrían haberse realizado otras operaciones para mejorar algunas de sus cualidades. La mayoría de ellas a pesar de no haberse implementado se encuentran explicadas en el apartado 2.3 del Estado del Arte.

- Para realzar el contraste: una ecualización del histograma, un filtro paso alto y/o un realzado basado en el dominio frecuencial.

- Para un mayor suavizado: mejorar los sistemas de adquisición de las imágenes o incluir nuevos filtros paso bajo.

- Para mejorar los bordes: operadores de gradiente, laplacianos o el filtro de Canny.

Mejoras en la extracción de características:

Los descriptores empleados se basan únicamente en dos características de bajo nivel, color y textura. Por ello podrían incluirse nuevos descriptores de bajo nivel que describan otras características de los objetos o conceptos a determinar en las imágenes. Se propone por tanto incluir descriptores de forma tales como el RSD (*Region-based Shape Descriptor*) o el CSD (*Contour-based Shape Descriptor*) que describan la forma de los objetos tal y como lo hace el sistema visual humano.

Además el sistema mejoraría al incluir descriptores de alto nivel, como los descriptores SIFT (*Scale Invariant Feature Transform*) o GLOH (*Gradient Location Orientation Histogram*) [10], los cuales localizan los puntos de interés de las imágenes.

También se debería mejorar la extracción de características del método específico en los conceptos en los que no ha funcionado como se esperaba, como en agua, vegetación o cielo. Para estos tres conceptos podrían extraerse una serie de características que definieran la textura de las zonas de la imagen en la que suelen aparecer sobre una determinada componente de color (verde para vegetación y azul para agua y cielo).

Mejoras en la etapa de clasificación:

Para la clasificación de los tres métodos implementados se ha empleado el algoritmo K-NN, el cual no produce los mejores resultados a costa de una implementación no demasiado compleja. Así se podría mejorar la clasificación utilizando métodos de clasificación más complejos tales como las SVM (máquinas de vectores de soporte) o las redes neuronales, métodos con los cuales algunos de los participantes obtienen resultados excelentes.

Mejoras generales:

Aparte de las posibles mejoras en las etapas implementadas pueden implementarse nuevas etapas o mejorarse otros aspectos que afectan al sistema.

- Incluir una etapa de segmentación. El sistema quizás mejoraría si se realizara una segmentación que dividiera o especificará la situación de los conceptos en las imágenes. El problema es que al no aparecer estos conceptos claramente diferenciados la segmentación a realizar es altamente compleja.

- Reducir el coste computacional. De cara al usuario es muy importante que el tiempo que consume el sistema para desarrollar la tarea sea el mínimo posible. Así este puede reducirse implementando tras la extracción de características una etapa intermedia que disminuya la dimensionalidad de los vectores de características. Para ello puede implementarse un método de Análisis por Componentes Principales (PCA), el cual selecciona cuáles son los parámetros significativos de los vectores para la descripción de la imagen descartando el resto.

- Mejorar la base de datos. Con el fin de obtener una base de datos más homogénea en cuanto al número de imágenes por concepto tanto para el conjunto de entrenamiento como de test.

- Integrar el sistema en una aplicación de etiquetado automático de imágenes.

PARTE VII:

ANEXO

A. Librería de funciones

En esta sección se presenta la librería completa de funciones implementada para el desarrollo de la tarea. Se ha utilizado como herramienta de programación el software matemático ***Matlab*** debido a la gran cantidad de librerías que dispone acerca del tratamiento digital de imágenes.

A continuación se muestra el código de las diferentes funciones implementadas. Se encuentran organizadas en cierta forma según los módulos en los que se ha dividido el desarrollo de la tarea. Denotar que las líneas precedidas por el símbolo '*%*' y con la fuente en verde son comentarios de las líneas o bloques a los que preceden.

A.1 Preprocesado

```
%   - PREPROCESADO -
%
%   ImgPreprocesada = Preprocesado(Imagen)
%
%   "Recibe una Imagen y le aplica un preprocesado realizando un filtro gaussiano de 3x3 con desviación típica = 0,5 para eliminar el ruido gaussiano y un filtro de medianas de
%   3x3 para eliminar el ruido impulsivo, devolviendo ImgPreprocesada". El atributo Imagen debe ser una imagen a color para poder extraer sus 3 componentes R,G,B y de tipo
%   uint8.

function [ImgPreprocesada] = Preprocesado(ImgOriginal)

% Obtiene el tamaño de la imagen
M = size(ImgOriginal,1);
N = size(ImgOriginal,2);

%% 1º ELIMINACIÓN DEL RUIDO GAUSSIANO (FILTRO GAUSSIANO)

% Crea un filtro gaussiano de 3x3 con desviación típica = 0,5 , por defecto
FiltroGauss = fspecial('gaussian');

% Se aplica el filtro gaussiano a la imagen
ImgFilt1 = imfilter(ImgOriginal,FiltroGauss);


%% 2º ELIMINACIÓN DEL RUIDO IMPULSIVO (FILTRO DE MEDIANAS)

% Separa la imagen a filtrar en sus 3 componentes RGB, pues el filtro de medianas se aplica en 2 dimensiones
    ImgFilt2_R = ImgFilt1(:,:,1);
    ImgFilt2_G = ImgFilt1(:,:,2);
    ImgFilt2_B = ImgFilt1(:,:,3);

% Aplica un filtro de medianas (3x3) a cada componente de color de la imagen
    ImgPreprocesada_R = medfilt2(ImgFilt2_R, [3 3],'symmetric');
    ImgPreprocesada_G = medfilt2(ImgFilt2_G, [3 3],'symmetric');
    ImgPreprocesada_B = medfilt2(ImgFilt2_B, [3 3],'symmetric');

% Reconstruye la imagen combinando las 3 componentes ya filtradas
    ImgPreprocesada = cat(3,ImgPreprocesada_R,ImgPreprocesada_G,ImgPreprocesada_B);

%% ELIMINACIÓN DEL MARCO QUE SE GENERA AL REALIZAR LOS FILTRADOS ANTERIORES
Corte = [2 2 (N-3) (M-3)]; % Se establece el área a cortar
ImgPreprocesada = imcrop(ImgPreprocesada,Corte);% Corta la imagen con el área establecido
ImgPreprocesada = imresize(ImgPreprocesada,[M N]);% Reajusta el tamaño actual al tamaño original de la imagen

% Pone el valor de los pixeles de las esquinas como la media de sus vecinos, puesto que el filtro de medianas hace que las esquinas aparezcan% como pixeles negros. Lo hace para
cada componente RGB
for i=1:3
    ImgPreprocesada(1,1,i) = round(median([ImgPreprocesada(1,2,i),ImgPreprocesada(2,1,i),ImgPreprocesada(2,2,i)]));
    ImgPreprocesada(M,1,i) = round(median([ImgPreprocesada(M,2,i),ImgPreprocesada(M-1,1,i),ImgPreprocesada(M-1,2,i)]));
    ImgPreprocesada(1,N,i) = round(median([ImgPreprocesada(1,N-1,i),ImgPreprocesada(2,N,i),ImgPreprocesada(2,N-1,i)]));
    ImgPreprocesada(M,N,i) = round(median([ImgPreprocesada(M,N-1,i),ImgPreprocesada(M-1,N,i),ImgPreprocesada(M-1,N-1,i)]));
end
```

A.2 Extracción de Características

A.2.1 Extracción de características generales

```
%   - EXTRAER PARAMETROS GENERALES -
%
%   ParametrosGenerales = ExtParamGenerales(Imagen,ID)
%
%   "Devuelve los parámetros generales de una 'Imagen' de entrada. Estos son: El descriptor de Colores Dominantes (DCD), la Cuantificación de Color en 5 niveles y los parámetros
%   de Textura de las componentes R,G y B del modelo de color RGB y de la componente V del modelo de color HSV."

function [ParametrosGenerales] = ExtParamGenerales(Imagen,ID)

% Convierte la Imagen a double
if ~isa(Imagen, 'double')
    ImagenDouble = im2double(Imagen);
else
    ImagenDouble = Imagen;
end

% Obtiene el tamaño de la Imagen
M = size(Imagen,1);
N = size(Imagen,2);

% Obtiene las componentes R,G,B del modelo RGB
R = ImagenDouble(:,:,1)*255;
G = ImagenDouble(:,:,2)*255;
B = ImagenDouble(:,:,3)*255;

%% DESCRIPTOR DE COLORES DOMINANTES (DCD), con K=5 colores dominantes
K = 5;
columnaR = im2col(R,[M N]);
columnaG = im2col(G,[M N]);
columnaB = im2col(B,[M N]);
columnaRGB = [columnaR columnaG columnaB];

% Desactiva los warnings que se producen al crear grupos vacios con Kmeans
warning('off','stats:kmeans:EmptyCluster');

StartMatrix = zeros(5,3);
[id centroides] = kmeans(columnaRGB,K,'Start',StartMatrix,'EmptyAction','Singleton');

CDominantes1 = round(centroides(1,:));
CDominantes2 = round(centroides(2,:));
CDominantes3 = round(centroides(3,:));
CDominantes4 = round(centroides(4,:));
CDominantes5 = round(centroides(5,:));

CDominantes = [CDominantes1 CDominantes2 CDominantes3 CDominantes4 CDominantes5];

% Probabilidad de los colores dominantes
ProbCD = hist(id,K)/numel(id);
DCD = [CDominantes ProbCD];

%% CUANTIFICACIÓN DE COLOR en 5 niveles
ProbNiveles = zeros(3,5);
for k=1:3
```

```
for i=1:M
    for j=1:N
        if (Imagen(i,j,k) <= 51)
            ICuantificada(i,j,k) = 26;
            ProbNiveles(k,1) = ProbNiveles(k,1) + 1;
        elseif (Imagen(i,j,k) <= 102)
            ICuantificada(i,j,k) = 77;
            ProbNiveles(k,2) = ProbNiveles(k,2) + 1;
        elseif (Imagen(i,j,k) <= 153)
            ICuantificada(i,j,k) = 128;
            ProbNiveles(k,3) = ProbNiveles(k,3) + 1;
        elseif (Imagen(i,j,k) <= 204)
            ICuantificada(i,j,k) = 179;
            ProbNiveles(k,4) = ProbNiveles(k,4) + 1;
        elseif (Imagen(i,j,k) <= 255)
            ICuantificada(i,j,k) = 230;
            ProbNiveles(k,5) = ProbNiveles(k,5) + 1;
        end
    end
end

ProbNiveles(k,:) = ProbNiveles(k,:)/(M*N);

end

Cuantificacion = [ProbNiveles(1,:) ProbNiveles(2,:) ProbNiveles(3,:)];


%% CARACTERÍSTICAS DE TEXTURA
% Obtiene las ImagenDoubles R,G,B y la ImagenDouble V del modelo HSV
R = Imagen(:,:,1);
G = Imagen(:,:,2);
B = Imagen(:,:,3);
HSV = rgb2hsv(Imagen);
V = HSV(:,:,3);

% Obtiene los parametros de Textura para cada ImagenDouble RGB
TexturaR = Textura(R);
TexturaG = Textura(G);
TexturaB = Textura(B);
TexturaV = Textura(V);


%% DEVUELVE LOS PARÁMETROS CALCULADOS ANTERIORMENTE COMO EL VECTOR DE CARACTERÍSTICAS GENERALES
ParametrosGenerales = [ID DCD Cuantificacion TexturaR TexturaG TexturaB TexturaV];

end
```

A.2.2 Extracción de características específicas

```
%   - EXTRAER PARAMETROS ESPECÍFICOS -
%
%   ParametrosEspecificos = ExtParamEspecificos(Imagen,ID)
%
%   "Devuelve los parámetros especificos (Interior, Personas, Dia/Noche, Agua, Vegetación y Cielo) de una 'Imagen' de entrada."

function [ParametrosEspecificos] = ExtParamEspecificos(Imagen,ID)

% Parametros Interior
ParametrosInterior = ExtParametrosInterior(Imagen);

% Parametros Personas
ParametrosPersonas = ExtParametrosPersonas(Imagen);

% Parametros DIA/NOCHE
ParametrosDia = ExtParametrosDia(Imagen);

% Parametros Agua
ParametrosAgua = ExtParametrosAgua(Imagen);

% Parametros Vegetacion
ParametrosVegetacion = ExtParametrosVegetacion(Imagen);

% Parametros Cielo
ParametrosCielo = ExtParametrosCielo(Imagen);


% Vector con todos los parametros
ParametrosEspecificos = [ID ParametrosInterior ParametrosPersonas ParametrosDia ParametrosAgua ParametrosVegetacion ParametrosCielo];

end
```

A.2.2.1 Extracción de características de Exterior/Interior

```
%   - EXTRAER PARAMETROS "INTERIOR" -
%
%   ParametrosInterior = ExtParametrosInterior(Imagen)
%
%   "Devuelve en el vector 'ParametrosInterior' diversos parámetros que describen el histograma de la Luminancia Y del modelo de color YUV de la 'Imagen' que recibe como
%   entrada." Para más información acerca de los parámetros del Histograma, escriba: help Textura

function ParametrosInterior = ExtParametrosInterior(imagen)

% Convierte la imagen a double
if(~isa(imagen,'double'))
    imagen = double(imagen);
end


%% PARÁMETROS IDENTIFICATIVOS DEL HISTOGRAMA DE LA COMPONENTE 'Y' DEL MODELO YUV
% Obtiene la componente de Luminancia Y del modelo YUV
R = imagen(:,:,1);
G = imagen(:,:,2);
B = imagen(:,:,3);

Y = uint8(0.299*R + 0.587*G + 0.114*B);

% Obtiene los parametros del Histograma de la Luminancia
ParametrosL = Textura(uint8(Y));


%% DEVUELVE LOS PARÁMETROS DE LA LUMINANCIA COMO EL VECTOR DE CARACTERÍSTICAS DE "INTERIOR/EXTERIOR"
ParametrosInterior = ParametrosL;

end
```

A.2.2.2 Extracción de características de Personas

```
%   - EXTRAER PARAMETROS "PERSONAS" -
%
%   ParametrosPersonas = ExtParametrosPersonas(Imagen)
%
%   "Devuelve en el vector 'ParametrosPersonas' los parámetros de Tamura que extrae a la componente V del modelo de color HSV a la 'Imagen' de entrada. Para más información
%   acerca de los parámetros de Tamura, escriba: help Tamura"

function ParametrosPersonas = ExtParametrosPersonas(Imagen)
%% PARÁMETROS DE TAMURA DE LA COMPONENTE V
% Obtiene el tamaño de la Imagen
M = size(Imagen,1);
N = size(Imagen,2);

% Obtiene la componente V del modelo de color HSV
HSV = rgb2hsv(Imagen);
V = HSV(:,:,3);

% Se queda con un cuadro central (10% horizontal y vertical), pues las
% personas suelen estar encuadradas
restoV = 15*N/100;
restoH = 15*M/100;

CuadroCentral = V(restoH:M-restoH,restoV:N-restoV);

% Calcula los parametros de Tamura del cuadro anterior
```

```
ParametrosTamura = Tamura(V);


%% DEVUELVE LOS PARÁMETROS DE TAMURA COMO EL VECTOR DE CARACTERÍSTICAS DE "PERSONAS"
ParametrosPersonas = ParametrosTamura;

end
```

A.2.2.3 Extracción de características de Día/Noche

```
%   - EXTRAER PARAMETROS "DIA" -
%
%   ParametrosDia = ExtParametrosDia(Imagen)
%
%   "Devuelve en el vector 'ParametrosDia' las probabilidades de aparición de los pixels en determinados rangos de las componente Y del modelo de color CMY y la componente Y del
%   modelo de color YUV de la 'Imagen' que recibe como entrada."

function ParametrosDia = ExtParametrosDia(Imagen)

% Obtiene la altura de la Imagen
M = size (Imagen,1);

% Convierte la imagen a 'uint8'
if (~isa(Imagen,'uint8'))
    Imagen = uint8(Imagen);
end


%% PARÁMETROS DE LA COMPONENTE DE AMARILLO 'Y' DEL MODELO CMY
% Se obtiene la componente Y = 1-B del modelo de color CMY
B = Imagen(:,:,3);
Y = 255 - B;

% Obtiene el Histograma normalizado de Y
HistY = imhist(Y)/numel(Y);

% Se calculan las probabilidades de aparicion de determinados rangos en la componente Y
ProbClaridadY = round(sum(HistY(1:55))*100);
ProbMediaY = round(sum(HistY(56:200))*100);
ProbOscuridadY = round(sum(HistY(201:256))*100);

ProbRangosY = [ProbOscuridadY ProbMediaY ProbClaridadY];


%% PARÁMETROS DE LA COMPONTENTE DE AMARILLO 'Y' DEL MODELO CMY EN LA PARTE SUPERIOR DE LA IMAGEN
% Obtiene la parte superior de la Imagen, un 10% de la altura de la misma
ImagenSupY = Y(1:(0.1*M),:);

% Obtiene su Histograma normalizado
HistSupY = imhist(ImagenSupY)/numel(ImagenSupY);

% Se calculan las probabilidades de aparicion de determinados rangos en la de la parte superior en la componente 'Y'
ProbClaridadSupY = round(sum(HistSupY(1:55))*100);
ProbMediaSupY = round(sum(HistSupY(56:200))*100);
ProbOscuridadSupY = round(sum(HistSupY(201:256))*100);

ProbRangosSupY = [ProbOscuridadSupY ProbMediaSupY ProbClaridadSupY];


%% PARÁMETROS DE LA COMPONENTE DE LUMINANCIA 'Y' DEL MODELO YUV
% Convierte la imagen a double
if(~isa(Imagen,'double'))
    Imagen = double(Imagen);
end

% Se obtiene la componente de Luminancia Y, denominada L para diferenciarla de la anterior
R = Imagen(:,:,1);
G = Imagen(:,:,2);
B = Imagen(:,:,3);

L = uint8(0.299*R + 0.587*G + 0.114*B);

% Se calcula el histograma normalizado
HistL = imhist(L)/numel(L);

% Se calculan las probabilidades de aparicion de determinados rangos en la componente L
ProbOscuridadL = round(sum(HistL(1:55))*100);
ProbMediaL = round(sum(HistL(56:200))*100);
ProbClaridadL = round(sum(HistL(201:256))*100);

ProbRangosL = [ProbOscuridadL ProbMediaL ProbClaridadL];

%% PARÁMETROS DE LA COMPONENTE DE LUMINANCIA 'Y' DEL MODELO YUV EN LA PARTE SUPERIOR DE LA IMAGEN
% Obtiene la parte superior de la Imagen, un 10% de la altura de lamisma
ImagenSupL = L(1:(0.1*M),:);

% Obtiene su Histograma normalizado
HistSupL = imhist(ImagenSupL)/numel(ImagenSupL);

% Se calculan las probabilidades de aparicion de determinados rangos en la de la parte superior en la componente 'Y'
ProbOscuridadSupL = round(sum(HistSupL(1:55))*100);
ProbMediaSupL = round(sum(HistSupL(56:200))*100);
ProbClaridadSupL = round(sum(HistSupL(201:256))*100);

ProbRangosSupL = [ProbOscuridadSupL ProbMediaSupL ProbClaridadSupL];


%% DEVUELVE LOS PARÁMETROS CALCULADOS ANTERIORMENTE COMO EL VECTOR DE CARACTERÍSTICAS DE "DIA/NOCHE"
ParametrosDia = [ProbRangosY ProbRangosSupY ProbRangosL ProbRangosSupL];

end
```

A.2.2.4 Extracción de características de Agua

```
%   - EXTRAER PARAMETROS "AGUA" -
%
%   ParametrosAgua = ExtParametrosAgua(Imagen)
%
%   "Devuelve en el vector 'ParametrosAgua' los parámetros de Textura que extrae a la parte inferior de la 'Imagen' de la componente B del modelo de color RGB."

function ParametrosAgua = ExtParametrosAgua(Imagen)
%% PARÁMETROS DE TEXTURA DE LA COMPONENTE B EN LA PARTE INFERIOR DE LA IMAGEN
% Obtiene la altura de la imagen
M = size (Imagen,1);

% Obtiene la componente B del modelo de color RGB
B = Imagen(:,:,3);

% Elimina el 40% superior de la imagen y se queda con el cuadro restante
restoH = 40*M/100;

CuadroB = B(restoH:M,:);

%% DEVUELVE LOS PARÁMETROS ANTERIORES COMO EL VECTOR DE CARACTERÍSTICAS DE "AGUA"
ParametrosAgua = Textura(CuadroB);

end
```

A.2.2.5 Extracción de características de Vegetación

```
%   - EXTRAER PARAMETROS "VEGETACIÓN" -
%
%   ParametrosVegetacion = ExtParametrosVegetacion(Imagen)
%
%   "Devuelve en el vector 'ParametrosVegetacion' los parámetros de Textura de la componente C2 del modelo C1C2C3 explicado en la memoria del proyecto. Para más información
%   acerca de los parámetros de Textura, escriba: help Textura

function ParametrosVegetacion = ExtParametrosVegetacion(Imagen)

% Convierte la Imagen a 'uint8'
if (~isa(Imagen,'uint8'))
    Imagen = uint8(Imagen);
end
```

```
%% PARÁMETROS DE TEXTURA DE LA COMPONENTE C2
% Obtiene las componente C2=(G-B)/2 del modelo C1,C2 y C3
G = Imagen(:,:,2);
B = Imagen(:,:,3);

C2 = (G-B)/2;
% Obtiene el Histograma normalizado
HistC2 = imhist(C2)/numel(C2);

% Se calculan los parametros de Textura pero sin tener en cuenta el valor del nivel 0, puesto que representa la probabilidad de aparicion de los pixels que no contienen
información de verde

% Se calcula la Media
Media = 0;

for i=1:127
    Media = Media + (HistC2(i+1)*i);
end

% Se calculan los demás parametros de Textura
if (Media ~= 0)
    Varianza = 0;% Varianza de la Intensidad
    Asimetria = 0;% Asimetria o Skewness
    Kurtosis = 0;% Kurtosis o Alargamiento
    Uniformidad = 0;% Uniformidad

    for i=1:127
        Varianza = Varianza + HistC2(i+1)*((i-Media)^2);
        Asimetria = Asimetria + HistC2(i+1)*((i-Media)^3);
        Kurtosis = Kurtosis + HistC2(i+1)*((i-Media)^4);
        Uniformidad = Uniformidad + (HistC2(i+1))^2;
    end

    Desviacion = sqrt(Varianza);% Desviacion tipica o contraste medio

    CoefAsimetria = Asimetria/(Desviacion^3);% Coeficiente de Asimetria
    CoefKurtosis = Kurtosis/(Desviacion^4) - 3;% Coeficiente de Kurtosis

    Suavidad = 1-(1/(1+(Varianza/(255-1)^2)^2));% Suavidad normalizada para el rango [0,1], para ello se divide por (255-1)^2

else
    Desviacion = 0;
    CoefAsimetria = 0;
    CoefKurtosis = 0;
    Uniformidad = 0;
    Suavidad = 0;
end

%% DEVUELVE LOS PARAMETROS DE TEXTURA DE LA COMPONENTE C2 COMO EL VECTOR DE CARACTERÍSTICAS DE "VEGETACIÓN"
 ParametrosVegetacion = [Media Desviacion CoefAsimetria CoefKurtosis Uniformidad Suavidad];

end
```

A.2.2.6 Extracción de características de Cielo

```
%   - EXTRAER PARAMETROS "CIELO" -
%
%   ParametrosCielo = ExtParametrosCielo(Imagen)
%
%   "Devuelve en el vector 'ParametrosCielo' diversos parámetros de varianza de los pixels y un parámetro que mide la dispersión del histograma, extraidos en la parte superior
%   de la 'Imagen' de la componente V del modelo HSV."

function ParametrosCielo = ExtParametrosCielo(Imagen)
%% VARIANZA DE LA PARTE SUPERIOR DE LA COMPONENTE 'V' (EL 10%)
M = size (Imagen,1);
N = size (Imagen,2);

% Extrae la componente de color V del modelo de color HSV
HSV = rgb2hsv(Imagen);
V = HSV(:,:,3);

% Se queda con la decima parte superior de la imagen
ImagenSup = V(1:M*0.1,1:N);

% Se calcula su varianza
Varianza = var(var(ImagenSup));

%% CÁLCULO DE LAS 2 MÍNIMAS VARIANZAS DE 5 CUADROS IGUALES DE LA PARTE SUPERIOR DE LA IMAGEN
% Se divide en 5 cuadros la imagen superior, si no es divisible por 5, se  agranda el tamaño de la imagen para que lo sea
resto = rem(N,5);

if (resto==0)
    ancho = N/5;
else
    ImagenSup = imresize(ImagenSup, [size(ImagenSup,1) ceil(N/5)*5]);
    ancho = size(ImagenSup,2)/5;
end

% Calcula la varianza para los 5 cuadros
Varianzas = zeros(1,5);
for i=1:5
    Cuadro = ImagenSup(:,((ancho*i)-ancho+1):ancho*i);
    Varianzas(i) = var(var(Cuadro));
end

% Se obtienen las 2 varianzas menores
[VarianzaMin1,indice] = min(Varianzas);
Varianzas(indice) = 1;
VarianzaMin2 = min(Varianzas);

%% COEFICIENTE DE DISPERSÓN
% Se obtiene el histograma normalizado de la ImagenSup
HistSup = imhist(ImagenSup)/numel(ImagenSup);

% Se calcula el nº de niveles que tienen un valor superior a un determinado umbral, en concreto, el 10% del máximo del histograma
CoefDispersion = 0;
Umbral = 0.1*max(HistSup);

for i=1:256
    if(HistSup(i)>Umbral)
        CoefDispersion = CoefDispersion + 1;
    end
end

%% DEVUELVE LOS PARÁMETROS ANTERIORES COMO EL VECTOR DE CARACTERÍSTICAS DE "CIELO"
ParametrosCielo = [Varianza VarianzaMin1 VarianzaMin2 CoefDispersion];

end
```

A.2.3 Extracción de los parámetros de Tamura

```
%   - EXTRAER PARAMETROS DE TAMURA -
%
%   Modificado por Juan Rodríguez Povedano
%   Archivo Original en: http://en.pudn.com/downloads123/sourcecode/graph/texture_mapping/detail522203_en.html
%
%   - Descripción:
%
%   ParametrosTamura = Tamura(Imagen)
%
%   "Devuelve en el vector 'ParametrosTamura' los parámetros de Tamura (Coarseness, Contraste y Dirección) de la 'Imagen' que recibe como
%   entrada.

function ParametrosTamura = Tamura(Imagen)

% Obtiene el tamaño de la imagen
[M N] = size(Imagen);


%% COARSENESS (VECINDAD 8x8)
EH = zeros(M,N);
EV = zeros(M,N);
Coarseness = zeros(M,N);
```

```
% Obtiene el valor promedio de la vecindad y le resta el valor de las vecindades opuestas, tanto en horizontal como vertical

for i=4:M-4
    for j=9:N-7
        EH(i,j) = sum(sum(Imagen(i-3:i+4,j:j+7)))-sum(sum(Imagen(i-3:i+4,j-8:j-1)));
    end
end
for i=8:M-8
    for j=5:N-3
        EV(i,j)=sum(sum(Imagen(i-7:i,j-4:j+3)))-sum(sum(Imagen(i+1:i+8,j-4:j+3)));
    end
end

EH=EH/64;
EV=EV/64;

% Obtiene el valor del coarseness entre el máximo de las diferencias horizontal y vertical
for i=1:M
    for j=1:N
        Coarseness(i,j) = max(EH(i,j),EV(i,j));
    end
end

% Obtiene dos parámetros del Coarseness de Tamura: El valor medio y la desviación
CoarsenessMedio = mean(mean(Coarseness));
CoarsenessDesv = std(std(Coarseness));


%% CONTRASTE
% Obtiene el histograma normalizado de la imagen
[Cuenta,Niveles] = imhist(Imagen);
PI = Cuenta/(M*N);

% Calcula la media
media = sum(Niveles.*PI);

% Calcula el 4º momento 'u4'
u4 = sum((Niveles - repmat(media,[256,1])).^4.*PI);

% Calcula la desviación
Desviacion = sum((Niveles - repmat(media,[256,1])).^2.*PI);

% Calcula Alpha4
Alpha4= u4/Desviacion^2;

% Obtiene el valor del Contraste de Tamura
Contraste = sqrt(Desviacion)/Alpha4.^(1/4);


%% DIRECCIONALIDAD
% Genera dos filtros de Prewitt, uno horizontal y otro vertical
PrewittH=[-1 0 1;-1 0 1;-1 0 1];
PrewittV=[1 1 1;0 0 0;-1 -1 -1];

% Filtra la imagen con los filtros anteriores
IncrementoH = filter2(PrewittH,Imagen);
IncrementoV = filter2(PrewittV,Imagen);

% Calcula la dirección de Tamura
Direccion = zeros(M,N);

for i=1:M
    for j=1:N
        Direccion(i,j) = (pi/2) + atan (IncrementoV(i,j)/IncrementoH(i,j));
        if (isnan(Direccion(i,j)))
            Direccion(i,j) = 0;
        end
    end
end

% Obtiene dos parámetros de la Dirección de Tamura: El valor medio y la desviación
DireccionMedia = mean(mean(Direccion));
DireccionDesv = sqrt(var(var(Direccion)));


%% DEVUELVE UN VECTOR CON LOS PARÁMETROS DE TAMURA CALCULADOS ANTERIORMENTE
ParametrosTamura = [CoarsenessMedio CoarsenessDesv Contraste DireccionMedia DireccionDesv];
```

A.2.4 Extracción de parámetros de Textura

```
%   - EXTRAER PARAMETROS DE TEXTURA -
%
%   ParametrosTextura = Textura(Imagen)
%
%   "Devuelve en el vector 'ParametrosTextura' los parámetros de Textura (Media, Desviación Típica, Coeficiente de Asimetría, Coeficiente de Kurtosis, Uniformidad y Suavidad)
%   extraídos a una 'Componente' de una imagen". 'Componente' debe ser una matriz de MxNx1, es decir, una imagen en niveles de gris.

function ParametrosTextura = Textura(Componente)

% Obtenemos el histograma normalizado de la componente
hist_norm = imhist(Componente)/numel(Componente);

% Se obtienen los momentos centrales

media = mean(mean(Componente));% Intensidad Media
varianza = 0;% Varianza de la Intensidad
asimetria = 0;% Asimetría o Skewness
kurtosis = 0;% Kurtosis o Alargamiento
uniformidad = 0;% Uniformidad

for i=0:255
        varianza = varianza + hist_norm(i+1)*((i-media)^2);
        asimetria = asimetria + hist_norm(i+1)*((i-media)^3);
        kurtosis = kurtosis + hist_norm(i+1)*((i-media)^4);
        uniformidad = uniformidad + (hist_norm(i+1))^2;
end

desviacion = sqrt(varianza);% Desviación típica o contraste medio

coefAsimetria = asimetria/(desviacion^3);% Coeficiente de asimetría
coefKurtosis = kurtosis/(desviacion^4) - 3;% Coeficiente de Kurtosis

suavidad = 1-(1/(1+(varianza/(255-1)^2)^2));% Suavidad normalizada para el rango [0,1], para ello se divide por (255-1)^2

ParametrosTextura = [media desviacion coefAsimetria coefKurtosis uniformidad suavidad];
```

A.3 Clasificación

A.3.1 Clasificador general

```
%   - CLASIFICADOR GENERAL -
%
%   Clasificacion = ClasificadorGen(ParamImagenG,ParamTrainG,AnotTrain,K)
%
%   "Recibe los Parámetros Generales ('ParamImagenG') de una imagen, los Parámetros Generales ('ParamTrainG') y las Anotaciones ('AnotTrain')de todas las imágenes de
%   Entrenamiento. Devuelve la 'Clasificación' de la imagen mediante el algoritmo KNN, buscando los 'K' vecinos más cercanos mediante la distancia euclídea."
%
%   Reliza la Clasificación utilizando unicamente los parámetros generales extraidos a las imágenes.

function [clasificacion] = ClasificadorGen(ParamImagenG,ParamTrainG,AnotTrain,K)
%% OBTIENE LOS VECINOS MÁS CERCANOS MEDIANTE LA DISTANCIA EUCLÍDEA
% Obtiene el tamaño de las matrices de parametros
[M N] = size(ParamTrainG);

%%% Calcula las distancias euclídeas entre los parametros de la imagen y los
%%% parametros de las clases
DistanciasGenerales = sqrt(sum(abs(ParamTrainG(:,2:N) - repmat(ParamImagenG(:,2:N),M,1)).^2,2));

% Crea unas variables para guardar las minimas distancias y sus indices
MinDistG = zeros(K,1);
IndicesG = zeros(K,1);

% Obtiene las K distancias mas cercanas para los parametros generales
for i=1:K
```

```
       % Obtiene el indice y el valor
        [MinDistG(i),IndicesG(i)] = min(DistanciasGenerales);

       % Cambia el valor de esa distancia al maximo de las distancias para
       % poder obtener otro minimo
        DistanciasGenerales(IndicesG(i))= max(DistanciasGenerales);
end

%% CLASIFICACIÓN%%
NumClases = 13;
clasificacion = zeros(1,NumClases+1);
clasificacion(1) = ParamImagenG(1);

%% Exterior/Interior%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
i = 1;

for j=1:K
        if(AnotTrain(IndicesG(i),2) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
        i=i+1;
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
if (H0 > H1)
    clasificacion(2) = 0;
else
    clasificacion(2) = 1;
end

%% Personas%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
i = 1;

for j=1:K
        if(AnotTrain(IndicesG(i),3) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
        i=i+1;
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
if (H0 > H1)
    clasificacion(3) = 0;
else
    clasificacion(3) = 1;
end

%% Dia/Noche%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
i = 1;

for j=1:K
        if(AnotTrain(IndicesG(i),4) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
        i=i+1;
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
if (H0 > H1)
    clasificacion(4) = 0;
else
    clasificacion(4) = 1;
end

%% Agua%%
% Restricción de Interior
if (clasificacion(2) == 0)
    clasificacion(5) = 0;
else
   % Inicializa la cuenta de pertenecientes a cada clase
    H0 = 0;
    H1 = 0;

   % Cuenta de los 'k' vecinos cuantos hay de cada clase
    i = 1;

    for j=1:K
            if(AnotTrain(IndicesG(i),5) == 0)
                H0 = H0 + 1;
            else
                H1 = H1 + 1;
            end
            i=i+1;
    end

   % Clasifica a la clase mayoritaria de los 'k' vecinos
    if (H0 > H1)
        clasificacion(5) = 0;
    else
        clasificacion(5) = 1;
    end
end

%% Caminos%%
% Restricción de Interior
if (clasificacion(2) == 0)
    clasificacion(5) = 0;
else
   % Inicializa la cuenta de pertenecientes a cada clase
    H0 = 0;
    H1 = 0;

   % Cuenta de los 'k' vecinos cuantos hay de cada clase
    i = 1;

    for j=1:K
            if(AnotTrain(IndicesG(i),6) == 0)
                H0 = H0 + 1;
            else
                H1 = H1 + 1;
            end
            i=i+1;
    end

   % Clasifica a la clase mayoritaria de los 'k' vecinos
    if (H0 > H1)
        clasificacion(6) = 0;
    else
        clasificacion(6) = 1;
    end
end

%% Vegetación%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;
```

```
% Cuenta de los 'k' vecinos cuantos hay de cada clase
i = 1;

for j=1:K
        if(AnotTrain(IndicesG(i),7) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
        i=i+1;
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
if (H0 > H1)
    clasificacion(7) = 0;
else
    clasificacion(7) = 1;
end

%% Árboles%%
% Restricción de Vegetación
if (clasificacion(7) == 0)
    clasificacion(8) = 0;
else
   % Inicializa la cuenta de pertenecientes a cada clase
    H0 = 0;
    H1 = 0;

   % Cuenta de los 'k' vecinos cuantos hay de cada clase
    i = 1;

    for j=1:K
            if(AnotTrain(IndicesG(i),8) == 0)
                H0 = H0 + 1;
            else
                H1 = H1 + 1;
            end
            i=i+1;
    end

   % Clasifica a la clase mayoritaria de los 'k' vecinos
    if (H0 > H1)
        clasificacion(8) = 0;
    else
        clasificacion(8) = 1;
    end
end
%% Montañas%%
% Restricción de Interior
if (clasificacion(2) == 0)
    clasificacion(5) = 0;
else
    % Inicializa la cuenta de pertenecientes a cada clase
    H0 = 0;
    H1 = 0;

   % Cuenta de los 'k' vecinos cuantos hay de cada clase
    i = 1;

    for j=1:K
            if(AnotTrain(IndicesG(i),9) == 0)
                H0 = H0 + 1;
            else
                H1 = H1 + 1;
            end
            i=i+1;
    end

   % Clasifica a la clase mayoritaria de los 'k' vecinos
    if (H0 > H1)
        clasificacion(9) = 0;
    else
        clasificacion(9) = 1;
    end
end
%% Playa%%
% Restricción de Interior
if (clasificacion(2) == 0)
    clasificacion(5) = 0;
else
   % Inicializa la cuenta de pertenecientes a cada clase
    H0 = 0;
    H1 = 0;

   % Cuenta de los 'k' vecinos cuantos hay de cada clase
    i = 1;

    for j=1:K
            if(AnotTrain(IndicesG(i),10) == 0)
                H0 = H0 + 1;
            else
                H1 = H1 + 1;
            end
            i=i+1;
    end

   % Clasifica a la clase mayoritaria de los 'k' vecinos
    if (H0 > H1)
        clasificacion(10) = 0;
    else
        clasificacion(10) = 1;
    end
end

%% Edificios%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
i = 1;

for j=1:K
        if(AnotTrain(IndicesG(i),11) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
        i=i+1;
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
if (H0 > H1)
    clasificacion(11) = 0;
else
    clasificacion(11) = 1;
end

%% Animales%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
i = 1;

for j=1:K
        if(AnotTrain(IndicesG(i),12) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
        i=i+1;
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
if (H0 > H1)
    clasificacion(12) = 0;
else
    clasificacion(12) = 1;
end
```

```
%% Cielo%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
i = 1;

for j=1:K
        if(AnotTrain(IndicesG(i),13) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
        i=i+1;
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
if (H0 > H1)
    clasificacion(13) = 0;
else
    clasificacion(13) = 1;
end

%% Tipo de Cielo%%

% Se comprueba que hay cielo y no es de noche
if (clasificacion(13)==1 && clasificacion(4)==1)

OK = 0;
Knew = K;

while (OK == 0)

   % Inicializa la cuenta de pertenecientes a cada clase
    H0 = 0;% Clase Soleado
    H1 = 0;% Clase Parcialmente Nublado
    H2 = 0;% Clase Nublado

    i = 1;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
    for j=1:Knew
            if(AnotTrain(IndicesG(i),14) == 1)
                H0 = H0 + 1;
            elseif(AnotTrain(IndicesG(i),15) == 1)
                H1 = H1 + 1;
            else
                H2 = H2 + 1;
            end
            i = i+1;
    end

   % Clasifica a la clase mayoritaria de los 'k' vecinos
    if (H0>H1 && H0>H2)% Clasifica como Soleado
        clasificacion(14) = 1;
        clasificacion(15) = 0;
        clasificacion(16) = 0;
        OK = 1;
    elseif (H1>H0 && H1>H2)% Clasifica como Parcialmente Nublado
        clasificacion(14) = 0;
        clasificacion(15) = 1;
        clasificacion(16) = 0;
        OK = 1;
    elseif (H2>H0 && H2>H1)% Clasifica como Nublado
        clasificacion(14) = 0;
        clasificacion(15) = 0;
        clasificacion(16) = 1;
        OK = 1;
    else

       % Si no puede clasificar porque hay el mismo numero de 2 o más
       % clases, se introduce sucesivamente un vecino más hasta que
       % pueda clasificar
        Knew = Knew + 1;

       % Obtiene una nueva distancia mas cercana, (General o
       % Especifica)
            [MinDistG(size(MinDistG)+1),IndicesG(size(IndicesG)+1)] = min(DistanciasGenerales);% Obtiene el indice y el valor

           % Cambia el valor de esa distancia al maximo de las distancias
           % para poder obtener otro minimo si fuera necesario
            DistanciasGenerales(IndicesG(size(IndicesG)))= max(DistanciasGenerales);
    end

end

% Si no hay cielo o hay cielo pero es de noche pone los tipos de cielo
% todos a cero
elseif(clasificacion(13)==0 || clasificacion(4)==0)
    clasificacion(14) = 0;
    clasificacion(15) = 0;
    clasificacion(16) = 0;
end

end
```

A.3.2 Clasificador específico

```
%   - CLASIFICADOR ESPECÍFICO -
%
%   Clasificacion = ClasificadorEsp(ParamImagenEsp,ParamTrainEsp,AnotTrain,K)
%
%   "Recibe los Parámetros Especificos ('ParamImagenEsp') de la imagen a clasificar, los Parámetros Especificos y las Anotaciones ('AnotTrain') de las imágenes de Entrenamiento.
%   Devuelve la 'Clasificación' de la imagen mediante el algoritmo KNN, buscando los 'K' vecinos más cercanos mediante la distancia euclidea."

function [Clasificacion] = ClasificadorEsp(ParamImagenEsp,ParamTrainEsp,AnotTrain,K)
%% OBTIENE LOS VECINOS MÁS CERCANOS MEDIANTE LA DISTANCIA EUCLÍDEA
% Obtiene el tamaño de los vectores de parametros especificos
[R] = size(ParamTrainEsp,1);

%%% Calcula las distancias euclideas entre los parámetros especificos de la imagen y los parámetros especificos de las imagenes de entrenamiento
DistanciasExterior = sqrt(sum(abs(ParamTrainEsp(:,2:7) - repmat(ParamImagenEsp(:,2:7),R,1)).^2,2));
DistanciasPersonas = sqrt(sum(abs(ParamTrainEsp(:,8:12) - repmat(ParamImagenEsp(:,8:12),R,1)).^2,2));
DistanciasDia = sqrt(sum(abs(ParamTrainEsp(:,13:24) - repmat(ParamImagenEsp(:,13:24),R,1)).^2,2));
DistanciasAgua = sqrt(sum(abs(ParamTrainEsp(:,25:30) - repmat(ParamImagenEsp(:,25:30),R,1)).^2,2));
DistanciasVegetacion = sqrt(sum(abs(ParamTrainEsp(:,31:36) - repmat(ParamImagenEsp(:,31:36),R,1)).^2,2));
DistanciasCielo = sqrt(sum(abs(ParamTrainEsp(:,37:40) - repmat(ParamImagenEsp(:,37:40),R,1)).^2,2));

% Crea variables para guardar las minimas distancias y sus indices
MinDistExterior = zeros(ceil(K/2),1);
IndicesExterior = zeros(ceil(K/2),1);

MinDistPersonas = zeros(ceil(K/2),1);
IndicesPersonas = zeros(ceil(K/2),1);

MinDistDia = zeros(ceil(K/2),1);
IndicesDia = zeros(ceil(K/2),1);

MinDistAgua = zeros(ceil(K/2),1);
IndicesAgua = zeros(ceil(K/2),1);

MinDistVegetacion = zeros(ceil(K/2),1);
IndicesVegetacion = zeros(ceil(K/2),1);

MinDistCielo = zeros(ceil(K/2),1);
IndicesCielo = zeros(ceil(K/2),1);

% Obtiene las K distancias mas cercanas para los parametros especificos
for i=1:K

       % Obtiene el indice y el valor
        [MinDistExterior(i),IndicesExterior(i)] = min(DistanciasExterior);
        [MinDistPersonas(i),IndicesPersonas(i)] = min(DistanciasPersonas);
        [MinDistDia(i),IndicesDia(i)] = min(DistanciasDia);
        [MinDistAgua(i),IndicesAgua(i)] = min(DistanciasAgua);
        [MinDistVegetacion(i),IndicesVegetacion(i)] = min(DistanciasVegetacion);
        [MinDistCielo(i),IndicesCielo(i)] = min(DistanciasCielo);
```

```
        % Cambia el valor de esa distancia al maximo de las distancias para poder obtener otro minimo
        DistanciasExterior(IndicesExterior(i))= max(DistanciasExterior);
        DistanciasPersonas(IndicesPersonas(i))= max(DistanciasPersonas);
        DistanciasDia(IndicesDia(i))= max(DistanciasDia);
        DistanciasAgua(IndicesAgua(i))= max(DistanciasAgua);
        DistanciasVegetacion(IndicesVegetacion(i))= max(DistanciasVegetacion);
        DistanciasCielo(IndicesCielo(i))= max(DistanciasCielo);

end

%% CLASIFICACIÓN%%
Categorias = 6;
Clasificacion = zeros(1,Categorias+1);
Clasificacion(1) = ParamImagenEsp(1);

%% Exterior/Exterior%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
i = 1;

for j=1:K
        if(AnotTrain(IndicesExterior(i),2) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
        i=i+1;
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
if (H0 > H1)
    Clasificacion(2) = 0;
else
    Clasificacion(2) = 1;
end

%% Personas%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
i = 1;

for j=1:K
        if(AnotTrain(IndicesPersonas(i),3) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
        i=i+1;
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
if (H0 > H1)
    Clasificacion(3) = 0;
else
    Clasificacion(3) = 1;
end

%% Dia/Noche%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
i = 1;

for j=1:K
        if(AnotTrain(IndicesDia(i),4) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
        i=i+1;
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
if (H0 > H1)
    Clasificacion(4) = 0;
else
    Clasificacion(4) = 1;
end

%% Agua%%
% Restricción de Exterior
if (Clasificacion(2) == 0)
    Clasificacion(5) = 0;
else
   % Inicializa la cuenta de pertenecientes a cada clase
    H0 = 0;
    H1 = 0;

   % Cuenta de los 'k' vecinos cuantos hay de cada clase
    i = 1;

    for j=1:K
            if(AnotTrain(IndicesAgua(i),5) == 0)
                H0 = H0 + 1;
            else
                H1 = H1 + 1;
            end
            i=i+1;
    end

   % Clasifica a la clase mayoritaria de los 'k' vecinos
    if (H0 > H1)
        Clasificacion(5) = 0;
    else
        Clasificacion(5) = 1;
    end
end


%% Vegetación%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
i = 1;

for j=1:K
        if(AnotTrain(IndicesVegetacion(i),7) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
        i=i+1;
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
if (H0 > H1)
    Clasificacion(6) = 0;
else
    Clasificacion(6) = 1;
end


%% Cielo%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
```

```
i = 1;

for j=1:K
        if(AnotTrain(IndicesCielo(i),13) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
        i=i+1;
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
if (H0 > H1)
    Clasificacion(7) = 0;
else
    Clasificacion(7) = 1;
end

end
```

A.3.3 Clasificador combinado

```
%   - CLASIFICADOR COMBINADO -
%
%   Clasificacion = ClasificadorComb
%   (ParamImagenG,ParamImagenEsp,ParamTrainG,ParamTrainEsp,AnotTrain,K)
%
%   "Recibe los Parámetros Generales ('ParamImagenG') y Específicos ('ParamImagenEsp') de una imagen, junto a los Parámetros Generales ('ParamTrainG') y Específicos
%   ('ParamTrainEsp') y las Anotaciones ('AnotTrain')de todas las imágenes de Entrenamiento. Devuelve la 'Clasificación' de la imagen mediante el algoritmo KNN, buscando los 'K'
%   vecinos más cercanos mediante la distancia euclídea."
%
%   Realiza la clasificación combinada, es decir, utiliza tanto los parámetros generales como los específicos extraídos a las imágenes. Obtiene los vecinos más cercanos para los
%   conceptos a los que se extraen los parámetros específicos mediante la combinación lineal de las distancias euclídeas de los parámetros generales y específicos de las
%   imágenes. Para el resto de conceptos se obtienen los vecinos más cercanos igual que con el Clasificador General, sólo con los parámetros generales de las imágenes.

function [Clasificacion] = ClasificadorComb(ParamImagenG,ParamImagenEsp,ParamTrainG,ParamTrainEsp,AnotTrain,K)
%% OBTIENE LOS VECINOS MÁS CERCANOS MEDIANTE LA DISTANCIA EUCLÍDEA
% Obtiene el tamaño de las matrices de parametros
[M N] = size(ParamTrainG);
[R] = size(ParamTrainEsp,1);

% Calcula las distancias euclídeas entre los parametros de la imagen y los parametros de las clases
DistanciasGenerales = sqrt(sum(abs(ParamTrainG(:,2:N) - repmat(ParamImagenG(:,2:N),M,1)).^2,2));
DistanciasGenerales = DistanciasGenerales/max(DistanciasGenerales);
DistanciasInterior = sqrt(sum(abs(ParamTrainEsp(:,2:7) - repmat(ParamImagenEsp(:,2:7),R,1)).^2,2));
DistanciasInterior = DistanciasGenerales + (DistanciasInterior/max(DistanciasInterior));
DistanciasPersonas = sqrt(sum(abs(ParamTrainEsp(:,8:12) - repmat(ParamImagenEsp(:,8:12),R,1)).^2,2));
DistanciasPersonas = DistanciasGenerales + (DistanciasPersonas/max(DistanciasPersonas));
DistanciasDia = sqrt(sum(abs(ParamTrainEsp(:,13:24) - repmat(ParamImagenEsp(:,13:24),R,1)).^2,2));
DistanciasDia = DistanciasGenerales + (DistanciasDia/max(DistanciasDia));
DistanciasAgua = sqrt(sum(abs(ParamTrainEsp(:,25:30) - repmat(ParamImagenEsp(:,25:30),R,1)).^2,2));
DistanciasAgua = DistanciasGenerales + (DistanciasAgua/max(DistanciasAgua));
DistanciasVegetacion = sqrt(sum(abs(ParamTrainEsp(:,31:36) - repmat(ParamImagenEsp(:,31:36),R,1)).^2,2));
DistanciasVegetacion = DistanciasGenerales + (DistanciasVegetacion/max(DistanciasVegetacion));
DistanciasCielo = DistanciasGenerales + sqrt(sum(abs(ParamTrainEsp(:,37:40) - repmat(ParamImagenEsp(:,37:40),R,1)).^2,2));
DistanciasCielo = DistanciasGenerales + (DistanciasCielo/max(DistanciasCielo));

% Crea variables para guardar las minimas distancias y sus indices
MinDistG = zeros(K,1);
IndicesG = zeros(K,1);
MinDistInterior = zeros(K,1);
IndicesInterior = zeros(K,1);
MinDistPersonas = zeros(K,1);
IndicesPersonas = zeros(K,1);
MinDistDia = zeros(K,1);
IndicesDia = zeros(K,1);
MinDistAgua = zeros(K,1);
IndicesAgua = zeros(K,1);
MinDistVegetacion = zeros(K,1);
IndicesVegetacion = zeros(K,1);
MinDistCielo = zeros(K,1);
IndicesCielo = zeros(K,1);


% Obtiene las K distancias mas cercanas para los parametros generales
for i=1:K

       % Obtiene el indice y el valor
        [MinDistG(i),IndicesG(i)] = min(DistanciasGenerales);

       % Cambia el valor de esa distancia al maximo de las distancias para
       % poder obtener otro minimo
        DistanciasGenerales(IndicesG(i))= max(DistanciasGenerales);

end

% Obtiene las K distancias mas cercanas para los parametros especificos
for i=1:K

       % Obtiene el indice y el valor
        [MinDistInterior(i),IndicesInterior(i)] = min(DistanciasInterior);
        [MinDistPersonas(i),IndicesPersonas(i)] = min(DistanciasPersonas);
        [MinDistDia(i),IndicesDia(i)] = min(DistanciasDia);
        [MinDistAgua(i),IndicesAgua(i)] = min(DistanciasAgua);
        [MinDistVegetacion(i),IndicesVegetacion(i)] = min(DistanciasVegetacion);
        [MinDistCielo(i),IndicesCielo(i)] = min(DistanciasCielo);

       % Cambia el valor de esa distancia al maximo de las distancias para poder obtener otro minimo
        DistanciasInterior(IndicesInterior(i))= max(DistanciasInterior);
        DistanciasPersonas(IndicesPersonas(i))= max(DistanciasPersonas);
        DistanciasDia(IndicesDia(i))= max(DistanciasDia);
        DistanciasAgua(IndicesAgua(i))= max(DistanciasAgua);
        DistanciasVegetacion(IndicesVegetacion(i))= max(DistanciasVegetacion);
        DistanciasCielo(IndicesCielo(i))= max(DistanciasCielo);

end

%% CLASIFICACIÓN%%
NumClases = 13;
Clasificacion = zeros(1,NumClases+1);
Clasificacion(1) = ParamImagenG(1);

%% Exterior/Interior%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
for j=1:K
        if(AnotTrain(IndicesInterior(j),2) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
if (H0 > H1)
    Clasificacion(2) = 0;
else
    Clasificacion(2) = 1;
end

%% Personas%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
for j=1:K
        if(AnotTrain(IndicesPersonas(j),3) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
```

```
if (H0 > H1)
    Clasificacion(3) = 0;
else
    Clasificacion(3) = 1;
end

%% Dia/Noche%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
for j=1:K
        if(AnotTrain(IndicesDia(j),4) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
if (H0 > H1)
    Clasificacion(4) = 0;
else
    Clasificacion(4) = 1;
end

%% Agua%%
% Restricción de Interior
if (Clasificacion(2) == 0)
    Clasificacion(5) = 0;
else
   % Inicializa la cuenta de pertenecientes a cada clase
    H0 = 0;
    H1 = 0;

   % Cuenta de los 'k' vecinos cuantos hay de cada clase
    for j=1:K
            if(AnotTrain(IndicesAgua(j),5) == 0)
                H0 = H0 + 1;
            else
                H1 = H1 + 1;
            end
    end

   % Clasifica a la clase mayoritaria de los 'k' vecinos
    if (H0 > H1)
        Clasificacion(5) = 0;
    else
        Clasificacion(5) = 1;
    end
end

%% Caminos%%
% Restricción de Interior
if (Clasificacion(2) == 0)
        Clasificacion(5) = 0;
else
    % Inicializa la cuenta de pertenecientes a cada clase
    H0 = 0;
    H1 = 0;

   % Cuenta de los 'K' vecinos cuantos hay de cada clase
    for j=1:K
            if(AnotTrain(IndicesG(j),6) == 0)
                H0 = H0 + 1;
            else
                H1 = H1 + 1;
            end
    end

   % Clasifica a la clase mayoritaria de los 'k' vecinos
    if (H0 > H1)
        Clasificacion(6) = 0;
    else
        Clasificacion(6) = 1;
    end
end

%% Vegetación%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
for j=1:K
        if(AnotTrain(IndicesVegetacion(j),7) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
if (H0 > H1)
    Clasificacion(7) = 0;
else
    Clasificacion(7) = 1;
end

%% Árboles%%
% Restricción de Vegetación
if (Clasificacion(7) == 0)
    Clasificacion(8) = 0;
else
   % Inicializa la cuenta de pertenecientes a cada clase
    H0 = 0;
    H1 = 0;

   % Cuenta de los 'K' vecinos cuantos hay de cada clase
    for j=1:K
            if(AnotTrain(IndicesG(j),8) == 0)
                H0 = H0 + 1;
            else
                H1 = H1 + 1;
            end
    end

   % Clasifica a la clase mayoritaria de los 'k' vecinos
    if (H0 > H1)
        Clasificacion(8) = 0;
    else
        Clasificacion(8) = 1;
    end
end
%% Montañas%%
% Restricción de Interior
if (Clasificacion(2) == 0)
        Clasificacion(5) = 0;
else
   % Inicializa la cuenta de pertenecientes a cada clase
    H0 = 0;
    H1 = 0;

   % Cuenta de los 'K' vecinos cuantos hay de cada clase
    for j=1:K
            if(AnotTrain(IndicesG(j),9) == 0)
                H0 = H0 + 1;
            else
                H1 = H1 + 1;
            end
    end

   % Clasifica a la clase mayoritaria de los 'k' vecinos
    if (H0 > H1)
        Clasificacion(9) = 0;
    else
        Clasificacion(9) = 1;
    end
end
```

```
%% Playa%%
% Restricción de Interior
if (Clasificacion(2) == 0)
    Clasificacion(5) = 0;
else
   % Inicializa la cuenta de pertenecientes a cada clase
    H0 = 0;
    H1 = 0;
   % Cuenta de los 'K' vecinos cuantos hay de cada clase
    for j=1:K
            if(AnotTrain(IndicesG(j),10) == 0)
                H0 = H0 + 1;
            else
                H1 = H1 + 1;
            end
    end

   % Clasifica a la clase mayoritaria de los 'k' vecinos
    if (H0 > H1)
        Clasificacion(10) = 0;
    else
        Clasificacion(10) = 1;
    end
end

%% Edificios%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
for j=1:K
        if(AnotTrain(IndicesG(j),11) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
if (H0 > H1)
    Clasificacion(11) = 0;
else
    Clasificacion(11) = 1;
end

%% Animales%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
for j=1:K
        if(AnotTrain(IndicesG(j),12) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
if (H0 > H1)
    Clasificacion(12) = 0;
else
    Clasificacion(12) = 1;
end

%% Cielo%%

% Inicializa la cuenta de pertenecientes a cada clase
H0 = 0;
H1 = 0;

% Cuenta de los 'k' vecinos cuantos hay de cada clase
for j=1:K
        if(AnotTrain(IndicesCielo(j),13) == 0)
            H0 = H0 + 1;
        else
            H1 = H1 + 1;
        end
end

% Clasifica a la clase mayoritaria de los 'k' vecinos
if (H0 > H1)
    Clasificacion(13) = 0;
else
    Clasificacion(13) = 1;
end

%% Tipo de Cielo%%

% Se comprueba que hay cielo y no es de noche
if (Clasificacion(13)==1 && Clasificacion(4)==1)

OK = 0;
Knew = K;
while (OK == 0)

   % Inicializa la cuenta de pertenecientes a cada clase
    H0 = 0;% Clase Soleado
    H1 = 0;% Clase Parcialmente Nublado
    H2 = 0;% Clase Nublado

% Cuenta de los 'k' vecinos cuantos hay de cada clase
    for j=1:Knew
            if(AnotTrain(IndicesG(j),14) == 1)
                H0 = H0 + 1;
            elseif(AnotTrain(IndicesG(j),15) == 1)
                H1 = H1 + 1;
            else
                H2 = H2 + 1;
            end
    end

   % Clasifica a la clase mayoritaria de los 'k' vecinos
    if (H0>H1 && H0>H2)% Clasifica como Soleado
        Clasificacion(14) = 1;
        Clasificacion(15) = 0;
        Clasificacion(16) = 0;
        OK = 1;
    elseif (H1>H0 && H1>H2)% Clasifica como Parcialmente Nublado
        Clasificacion(14) = 0;
        Clasificacion(15) = 1;
        Clasificacion(16) = 0;
        OK = 1;
    elseif (H2>H0 && H2>H1)% Clasifica como Nublado
        Clasificacion(14) = 0;
        Clasificacion(15) = 0;
        Clasificacion(16) = 1;
        OK = 1;
    else

       % Si no puede clasificar porque hay el mismo numero de 2 o más clases, se introduce sucesivamente un vecino más hasta que pueda clasificar
        Knew = Knew + 1;

       % Obtiene una nueva distancia mas cercana
            [MinDistG(size(MinDistG)+1),IndicesG(size(IndicesG)+1)] = min(DistanciasGenerales);% Obtiene el indice y el valor

           % Cambia el valor de esa distancia al maximo de las distancias para poder obtener otro minimo si fuera necesario
            DistanciasGenerales(IndicesG(size(IndicesG)))= max(DistanciasGenerales);
    end
end

% Si no hay cielo o hay cielo pero es de noche pone los tipos de cielo
% todos a cero
elseif(Clasificacion(13)==0 || Clasificacion(4)==0)
    Clasificacion(14) = 0;
    Clasificacion(15) = 0;
    Clasificacion(16) = 0;
end
end
```

A.4 Evaluación

A.4.1 Evaluar clasificador

```
%   - EVALUAR CLASIFICADOR -
%
%   [MatricesConfusion, MedidasCalidad] = EvaluarClasificador(Clasificacion,AnotTest)
%
%   "Recibe la matriz con el resultado de la clasificación de las imagenes de Test ('Clasificacion') y la clasificación real de las mismas ('AnotTest'). Con ellas evalua la
%   calidad del Clasificador implementado, devolviendo las Matrices de Confusión de cada cada clase en la matriz 'MatricesConfusion' y otras medidas de calidad de los
%   clasificadores en la matriz 'MedidasCalidad'."

function [MatricesConfusion, MedidasCalidad] = EvaluarClasificador (Clasificacion,AnotTest)

[M N] = size(Clasificacion);

MatricesConfusion = zeros(N-1,4);
MedidasCalidad = zeros(N-1,3);

for j=2:N
%% OBTIENE LAS MATRICES DE CONFUSIÓN PARA CADA CLASE
TN = 0; TP = 0; FN = 0; FP = 0;

for i=1:M
    if (Clasificacion(i,j)==0 && AnotTest(i,j)==0)
        TN = TN+1;
    elseif (Clasificacion(i,j)==1 && AnotTest(i,j)==1)
        TP = TP+1;
    elseif (Clasificacion(i,j)==0 && AnotTest(i,j)==1)
        FN = FN+1;
    elseif (Clasificacion(i,j)==1 && AnotTest(i,j)==0)
        FP = FP+1;
    end
end
MatricesConfusion(j-1,:) = [TP FP FN TN];

%% OBTIENE OTRAS MEDIDAS DE CALIDAD DE LOS CLASIFICADORES

Precision = (round(TP/(TP + FP)*1000))/10;
Cobertura = (round(TP/(TP + FN)*1000))/10;
MedidaF = (round((20*Precision*Cobertura)/(Precision + Cobertura)))/10;
MedidasCalidad(j-1,:) = [Precision Cobertura MedidaF];

end
end
```

A.4.2 Curva ROC

```
%   - ROC -
%
%   [TPR FPR AUC] = ROC(Tipo,MConf1,MConf3,MConf5,MConf7,Mconf9)
%
%   "Devuelve la tasa de Verdaderos Positivos (TPR) y la tasa de Falsos Positivos (FPR) para poder dibujar la curva ROC, y el Área Bajo la Curva (AUC) como parámetro para
%   evaluar la calidad del clasificador."

function [TPR FPR AUC] = ROC(Tipo,MConf1,MConf3,MConf5,MConf7,MConf9)
%% Ordena las Matrices de Confusión en una matriz. Las que son 'Inverso' son las de Interior y Noche, pues son las inversas de Exterior y Día
if (strcmp(Tipo,'Normal'))
    % Se situan todas las matrices de confusión en una única matriz
    MatricesConfusion = [MConf1;MConf3;MConf5;MConf7;MConf9];

elseif (strcmp(Tipo,'Inverso'))
    NuevoOrden = [4 3 2 1];
    for i=1:4
        MConf1_2(i) = MConf1(NuevoOrden(i));
        MConf3_2(i) = MConf3(NuevoOrden(i));
        MConf5_2(i) = MConf5(NuevoOrden(i));
        MConf7_2(i) = MConf7(NuevoOrden(i));
        MConf9_2(i) = MConf9(NuevoOrden(i));
    end
   % Se situan todas las matrices de confusión en una única matriz
    MatricesConfusion = [MConf1_2;MConf3_2;MConf5_2;MConf7_2;MConf9_2];
end

%% Calcula la Tasa de Verdaderos Positivos (TPR) y la Tasa de Falsos Negativos (FPR). Inicializa los vectores con las TPR y las FPR Negativos
TPR = [0 0 0 0 0 0 1];
FPR = [0 0 0 0 0 0 1];

% Calcula las TPR y FPR de cada Matriz de Confusión
for i=1:5
    TPR(i+1) = MatricesConfusion(i,1)/(MatricesConfusion(i,1) + MatricesConfusion(i,3));
    FPR(i+1) = MatricesConfusion(i,2)/(MatricesConfusion(i,2) + MatricesConfusion(i,4));
end
% Ordena los vectores TPR y FPR de menor a mayor combinadamente
[FPR,I1] = sort(FPR);

for i=1:7
    TPR2(i) = TPR(I1(i));
end

TPR = TPR2;

%% Calcula el AUC (Area Under Curve)

% Calcula el AUC(%)
AUC = trapz(FPR,TPR)*100;

end
```

A.5 Procesos generales

A.5.1 Entrenamiento

```
%   - ENTRENAMIENTO -
%
%   "Realiza el Entrenamiento del sistema de reconocimiento. En esta etapa carga, preprocesa y extrae los parámetros generales y especificos%   de las imágenes de
%   entrenamiento."

%% CARGA LAS IMAGENES Y LOS PARÁMETROS NECESARIOS
Raiz = 'Imagenes';  % Directorio donde se encuentran las imágenes
DirTrain = 'train'; % Directorio donde se encuentran las imágenes de Entrenamiento

% Guarda la ruta del directorio de las imágenes de Entrenamiento
Train = dir([Raiz '/' DirTrain]);

% Obtiene el número de imágenes del directorio de Entrenamiento (No se
% tienen en cuenta las dos primeras porque no son imágenes, sino datos)
NumTrain = (size(Train,1) - 2);

% Lee las anotaciones
[AnotacionesTrain] = LeerAnotaciones(NumTrain,'Datos/AnotacionesTrain.txt');

% Carga las matrices de parametros si existen y si no las crea vacias
load('Datos/ParamEntrenamientoGen.mat');
load('Datos/ParamEntrenamientoEsp.mat');
if (~Flag)
ParametrosTrainG = zeros(NumTrain,60);
end
if (~Flag2)
ParametrosTrainEsp = zeros(NumTrain,40);
end

% Muestra por pantalla que se inicia el Entrenamiento
disp(' Cargando, Preprocesando y Extrayendo parámetros a las imágenes de Entrenamiento...');

%% OBTIENE LOS PARÁMETROS DE LAS IMAGENES DE ENTRENAMIENTO
for i=3:(NumTrain+2);% Comienza en 3 porque las dos primeras no son imágenes, sino datos

   % Obtiene el nombre de la imagen
```

```
    nombre = Train(i).name;

   % Carga la imagen
    Img = imread([Raiz '/' DirTrain '/' nombre]);

   % Preprocesa la imagen
    [ImgPreprocesada] = Preprocesado(Img);

   % Obtiene los parametros Generales de la imagen preprocesada
    if (~Flag)
        [ParametrosTrainG(i-2,:)] = ExtParamGenerales(ImgPreprocesada,AnotacionesTrain(i-2,1));
    end

   % Obtiene los parametros Especificos de la imagen preprocesada
    if (~Flag2)
        ParametrosTrainEsp(i-2,1) = AnotacionesTrain(i-2,1);
        ParametrosTrainEsp(i-2,2:7) = ExtParametrosInterior(ImgPreprocesada);
        ParametrosTrainEsp(i-2,8:12) = ExtParametrosPersonas(ImgPreprocesada);
        ParametrosTrainEsp(i-2,13:24) = ExtParametrosDia(ImgPreprocesada);
        ParametrosTrainEsp(i-2,25:30) = ExtParametrosAgua(ImgPreprocesada);
        ParametrosTrainEsp(i-2,31:36) = ExtParametrosVegetacion(ImgPreprocesada);
        ParametrosTrainEsp(i-2,37:40) = ExtParametrosCielo(ImgPreprocesada);
    end

   % Muestra una barra de progreso con el porcentaje de la Carga, Preprocesado y Extracción de parámetros de las imagenes de entrenamiento
    stopBar = Progressbar(i/(NumTrain+2),0);
    if (stopBar)
        disp(' Entrenamiento Cancelado');
        disp(' ');
        break;
    end
end

%% GUARDA LAS MATRICES CON LOS PARÁMETROS DE ENTRENAMIENTO
% Guarda los parametros si no se ha cancelado el entrenamiento
if (~stopBar)
    disp(' ¡El Entrenamiento se ha realizado con éxito!');
    save Datos/DatosEntrenamiento.mat AnotacionesTrain;
    save Datos/ParamEntrenamientoGen.mat ParametrosTrainG Flag;
    save Datos/ParamEntrenamientoEsp.mat ParametrosTrainEsp Flag2;
else
    clear all;
end

%% SE BORRAN LAS VARIABLES INTERMEDIAS
clear i Img ImgPreprocesada Train Raiz nombre DirTrain cancelado stopBar Flag Flag2
```

A.5.2 Test

```
%   - TEST -
%
%   Test(TipoClasificador,K)
%
%   "Recibe como parámetros el Tipo de Clasificador Elegido y el valor de K elegidos para realizar el Test. Guarda como datos los parámetros Generales y Especificos, la
%   Clasificación de las imágenes de Test, las Matrices de Confusión y las Medidas de Calidad del clasificador escogido." Realiza el Test del sistema de reconocimiento. En esta
%   etapa carga, preprocesa y extrae los parámetros generales y especificos de las imágenes de test, realiza la clasificación y evalúa el clasificador.

function Test(TipoClasificador,k)
%% CARGA LAS IMAGENES Y LOS PARÁMETROS NECESARIOS
%%% Especifica la direccion del directorio donde están las imágenes
Raiz = 'Imagenes';   % Directorio donde se encuentran las imágenes
DirTest = 'test';   % Directorio donde se encuentran las imágenes de Test

%%% Guarda la ruta del directorio de las imágenes de Test
Directorio = dir([Raiz '/' DirTest]);

%%% Obtiene el número de imágenes del directorio de Test (No se tienen en cuenta dos porque las dos primeras porque no son imágenes, sino datos)
NumTest = (size(Directorio,1) - 2);

%%% Lee las anotaciones de las imagenes de Test
[AnotacionesTest] = LeerAnotaciones(NumTest,'Datos/AnotacionesTest.txt');

% Carga las matrices de parametros si existen y si no las crea vacias (Los vectores de caracteristicas generales y especificas tanto de
% Entrenamiento como de Test se encuentran guardados en memoria para reducir el coste computacional del Test)
load('Datos/DatosEntrenamiento.mat');
load('Datos/ParamEntrenamientoEsp.mat');
load('Datos/ParamEntrenamientoGen.mat');

load('Datos/ParamTestG.mat');
load('Datos/ParamTestEsp.mat');

if (~Flag)
    ParametrosTestG = zeros(NumTest,60);
end
if (~Flag2)
    ParametrosTestEsp = zeros(NumTest,40);
end

%% OBTIENE LOS PARÁMETROS DE TEST Y CLASIFICA LAS IMAGENES
for i=3:(NumTest+2);%Comenzamos en 3 porque las dos primeras no son imágenes, sino datos

   % Obtiene el nombre de la imagen
    nombre = Directorio(i).name;

   % Carga la imagen
    Img = imread([Raiz '/' DirTest '/' nombre]);

   % Preprocesamos la imagen
    [ImgPreprocesada] = Preprocesado(Img);

   % Obtiene los parámetros Generales de la imagen preprocesada
    if (~Flag)
        [ParametrosTestG(i-2,:)] = ExtParamGenerales(ImgPreprocesada,AnotacionesTest(i-2,1));
    end

   % Obtiene los parámetros Especificos de la imagen preprocesada
    if (~Flag2)
        ParametrosTestEsp(i-2,1) = AnotacionesTest(i-2,1);
        ParametrosTestEsp(i-2,2:7) = ExtParametrosInterior(ImgPreprocesada);
        ParametrosTestEsp(i-2,8:12) = ExtParametrosPersonas(ImgPreprocesada);
        ParametrosTestEsp(i-2,13:24) = ExtParametrosDia(ImgPreprocesada);
        ParametrosTestEsp(i-2,25:30) = ExtParametrosAgua(ImgPreprocesada);
        ParametrosTestEsp(i-2,31:36) = ExtParametrosVegetacion(ImgPreprocesada);
        ParametrosTestEsp(i-2,37:40) = ExtParametrosCielo(ImgPreprocesada);
    end

   % Clasifica la imagen según se haya decidido: General / Especifica / Combinada
    if (strcmp(TipoClasificador,'General'))
        [Clasificacion(i-2,:)] = ClasificadorGen(ParametrosTestG(i-2,:),ParametrosTrainG,AnotacionesTrain,k);
    elseif (strcmp(TipoClasificador,'Especifico'))
        [Clasificacion(i-2,:)] = ClasificadorEsp(ParametrosTestEsp(i-2,:),ParametrosTrainEsp,AnotacionesTrain,k);
    elseif (strcmp(TipoClasificador,'Combinado'))
        [Clasificacion(i-2,:)] = ClasificadorComb(ParametrosTestG(i-2,:),ParametrosTestEsp(i-2,:),ParametrosTrainG,ParametrosTrainEsp,AnotacionesTrain,k);
    end

   % Muestra una barra de progreso con el porcentaje de la Carga, Preprocesado, Extracción de parámetros y Clasificación de las imagenes de entrenamiento
    stopBar= Progressbar(i/(NumTest+2),0);
    if (stopBar)
        disp(' ');
        disp(' Test Cancelado');
        disp(' ');
        break;
    end
end

%% GUARDA LAS MATRICES CON LOS PARÁMETROS DE TEST
save Datos/Clasificacion.mat Clasificacion;
save Datos/ParamTestG.mat ParametrosTestG Flag;
save Datos/ParamTestEsp.mat ParametrosTestEsp Flag2;

%% EVALUA LA CALIDAD DEL CLASIFICADOR PARA CADA CLASE Y LO GUARDA
if (~stopBar)
    [MatricesConf,MedidasCalidad] = EvaluarClasificador(Clasificacion,AnotacionesTest);
end

% Guarda las Matrices de Confusion y las Medidas de Calidad
valorK = num2str(k);
nombreMatricesConf = strcat('Calidades/',TipoClasificador,'/MedidasCalidad-K',valorK,'.mat');
save (nombreMatricesConf,'MatricesConf','MedidasCalidad');
end
```

PARTE VIII:

BIBLIOGRAFÍA

Bibliografía

[1] AH-PINE, J. et al (2008). *XRCE´s Participation to ImageCLEF 2008* [en línea]. Xerox Research Centre Europe, Meylan, Francia, 2008. Disponible en Web: http://www.clef-campaign.org/2008/working_notes/ah-pine-paperCLEF2008.pdf [Consulta: 15 de enero de 2010].

[2] ALBA, José Luís (Universidad de Vigo) y CID SUERO, Jesús (Universidad Carlos III de Madrid). *Tema 4: Reconocimiento de patrones* [en línea]. Asignatura de procesado de imágenes, Universidad de Vigo, curso 2005-2006. Disponible en Web: http://www.gts.tsc.uvigo.es/pi/Reconocimiento.pdf [Consulta: 15 de octubre de 2009].

[3] ALBA, José Luis y MARTÍN, Fernando (Universidad de Vigo), CID SUERO, Jesús (Universidad Carlos III de Madrid) y MORA, Inmaculada (Universidad Rey Juan Carlos). *Procesado de Imagen* [en línea]. Asignatura de procesado de imágenes, Universidad de Vigo, curso 2005-2006. Disponible en Web: http://www.gts.tsc.uvigo.es/pi [Consulta: 7 de septiembre de 2009].

[4] Báez Rojas, J.J et al(2004). *Segmentación de imágenes en color* [en línea]. Revista Mexicana de Física, vol. 50, nº6, diciembre 2004, pp. 579-587. Disponible en Web: http://redalyc.uaemex.mx/redalyc/pdf/570/57050605.pdf [Consulta: 2 de diciembre de 2009].

[5] CAICEDO RUEDA, Juan Carlos. *Extracción de características para recuperación de imágenes por contenido* [en línea]. Ingeniería de Sistemas de Comunicación, Universidad de Colombia, Mayo de 2009. Disponible en Web: http://dis.unal.edu.co/profesores/ypinzon/2013326-206/docs/Articulo1Caicedo.pdf [Consulta: 20 de noviembre de 2009].

[6] CID SUERO, Jesús. *Curso de procesado básico de imágenes: IMAGine* [en línea]. Cursos interactivos de tratamiento digital de imagen, Universidad Carlos III de Madrid, 2007. Disponible en Web: http://www.tsc.uc3m.es/imagine/index.html [Consulta: 5 de septiembre de 2009].

[7] DELCOR BALLESTEROS, Jordi y PÉREZ NORIEGA, Verónica. *Descripción, Indexación, Búsqueda y Adquisición de secuencias de video mediante descriptores MPEG-7* [en línea]. Proyecto tutorado por Javier Ruiz Hidalgo, Universidad Politécnica de Catalunya, 2006. Disponible en Web: http://upcommons.upc.edu/pfc/bitstream/2099.1/3855/1/54955-1.pdf [Consulta: 3 de noviembre de 2009].

[8] DESELAERS, Thomas. *Features for Image Retrieval* [en línea]. Cátedra en Ciencias de la Informática, RWTH Aachen Universtity. Disponible en Web: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.126.241&rep=rep1&type=pdf [Consulta: 20 de noviembre de 2009].

[9] DOURADO LEMA, Juan Manuel y CALVO ESTÉVEZ, Rosa María. *Reconocimiento de objetos* [en línea]. Grupo de visión artificial y reconocimiento de patrones (VARPA), Universidad de A Coruña, 2007. Disponible en Web: http://varpa.lfcia.org/Docencia/VAFiles/Curso0607/tema5.pdf [Consulta: 3 de septiembre de 2009].

[10] DR. FRAUEL, Yann. *Características y Descriptores* [en línea]. Capítulo 4, asignatura de visión computacional, Postgrado en Ciencias e Ingeniería de la Computación, Universidad Nacional de México, 2008. Disponible en Web: http://leibniz.iimas.unam.mx/~yann/Vision/Clase04.pdf [Consulta: 15 de diciembre de 2009].

[11] GRUBINGER, Michael, CLOUGH, Paul y LEUNG, Clement. *ICPR TC-12, Multimedia and Visual Information Systems* [en línea]. International Association for Pattern Recognition (IAPR), 2003. Disponible en Web: http://www.iapr.org/members/newsletter/Newsletter06-02/index_files/Page621.htm [Consulta: 17 de noviembre de 2009].

[12] HARALICK, R.M., SHANMUGAN, K. y DINSTEIN, I. (1973). *Texture features for image classification.* IEEE Transactions on Systems, Man and Cybernetics, U.S.A, vol. SMC-3, nº 6, pp. 610-521.

[13] HERVE, Glotin y ZHONGQIU, Zhao. *Profile Entropic visual Features for Visual Concept Detection in CLEF 2008 campaign* [en línea]. Laboratorio de Ciencias de la Información y Sistemas, Universidad Sud Toulon-Var, Francia, 2008. Disponible en Web: http://www.clef-campaign.org/2008/working_notes/section5_zhao_LSIS_CLEF08_VCDT_notes.pdf [Consulta: 15 de enero de 2010].

[14] JINGTIAN, Jiang, XIAOGUANG Rui y NENGHAI Yu. *Feature Annotation for Visual Concept Detection in ImageCLEF 2008* [en línea]. Microsoft Key Laboratory of Multimedia Computing and Communication, dpto. de EEIS, Universidad de Ciencia y Teconlogía de China, 2008. Disponible en Web: http://www.clef-campaign.org/2008/working_notes/jiang-paperCLEF2008.pdf [Consulta: 15 de enero de 2010].

[15] KITTLER, J. *Reconocimiento de patrones* [en línea]. Notas del seminario de reconocimiento de patrones del grupo de tratamiento de imágenes, Instituto de Ingeniería Eléctrica, Universidad de la República de Uruguay, 2002. Disponible en Web: http://iie.fing.edu.uy/ense/asign/recpat/material.php [Consulta: 15 de octubre de 2009].

[16] KRISHNAPURAM, B. et al. *Sparse Multinomial Logistic Regression: Fast Algorithms and Generalization Bounds* [en línea]. Algoritmos rápidos y límites de generalización. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 27, nº 6, junio de 2005. Disponible en Web: http://www.lx.it.pt/~mtf/Krishnapuram_Carin_Figueiredo_Hartemink_2005.pdf [Consulta: 22 de octubre de 2009].

[17] L.BROWN, Charles y N.SMIRNOV, Sergei. *Why is water blue?* [en línea]. Journal of chemical education, vol. 70, nº 8, 1993, pp. 612-614. Disponible en Web: http://www.dartmouth.edu/~etrnsfer/water.htm [Consulta: 2 de diciembre de 2009].

[18] MARTÍNEZ, Elena. *Realce de la imagen en dominio espacial* [en línea]. Curso de procesamiento digital de imágenes, dpto. de Ciencias de la Computación, Universidad Nacional de México, 2005. Disponible en Web: http://turing.iimas.unam.mx/~elena/Teaching/PDI-Lic.html [Consulta: 8 de septiembre de 2009].

[19] MARTÍNEZ, Elena. *Análisis de Texturas* [en línea]. Capítulo 5.4, curso de visión computacional, dpto. de Ciencias de la Computación, Universidad Nacional de México, 2008. Disponible en Web: http://turing.iimas.unam.mx/~elena/CompVis/Chapter5-4.pdf.gz [Consulta: 8 de noviembre de 2009].

[20] MOUJAHID, Abdelmalik, INZA Iñaki y LARRAÑAGA Pedro. *Clasificadores K-NN* [en línea]. Dpto. de Ciencias de la Computación e Inteligencia Artificial, Universidad del País Vasco, 2007. Disponible en Web: http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t9knn.pdf [Consulta: 12 de marzo de 2010].

[21] OHTA, Yuichi et al (1980). *Computer Graphics and Image Processing.* Departamento de Ciencias de la Información de la Universidad de Kyoto, Japón, Volumen 13, capítulo 3, pp. 222-241.

[22] ORTIZ RODRÍGUEZ, Alberto. *Tema 9: Texturas y descriptores de regiones* [en línea]. Apuntes de Visión Industrial, Ingeniería Técnica Industrial especialidad en Electrónica, Universidad de las Islas Baleares, enero de 2008. Disponible en Web: http://dmi.uib.es/~aortiz/4730-tema9.pdf [Consulta: 8 de noviembre de 2009].

[23] ORTIZ ZAMORA, Francisco Gabriel. *Procesamiento morfológico de imágenes en color: Aplicación a la reconstrucción geodésica* [en línea]. Capítulo 2: Fundamentos de color, PFC de la Escuela Politécnica Superior, Universidad de Alicante, 2002. Disponible en Web:

http://descargas.cervantesvirtual.com/servlet/SirveObras/57915842105571617400080/008591_2.pdf [Consulta: 14 de noviembre de 2009].

[24] OVERELL, Simon et al. *Experiments combining different evidence sources* [en línea]. Paper del grupo MMIS para la tarea VCDT de ImageCLEF 2008, 2008. Disponible en Web: http://www.clef-campaign.org/2008/working_notes/overell-ImageCLEF-paperCLEF2008.pdf [Consulta: 15 de enero de 2010].

[25] PLATERO DUEÑAS, Carlos. *Técnicas de preprocesado* [en línea]. Capítulo 4 de la asignatura de visión artificial, dpto. de Electrónica, Automática e Informática Industrial (ELAI), Universidad Politécnica de Madrid, 2007. Disponible en Web: http://www.elai.upm.es/spain/Asignaturas/Robotica/ApuntesVA/cap4Procesadov1.pdf [Consulta: 7 de septiembre de 2009].

[26] REINOSO GARCÍA, Oscar. *Teoría de la decisión de Bayes* [en línea]. Asignatura de inteligencia artificial y reconocimiento de patrones, dpto. de Ingeniería de Sistemas Industriales, Universidad Miguel Hernández , curso 2007-2008. Disponible en Web: http://isa.umh.es/asignaturas/iarp/teoria/bayes.pdf [Consulta: 18 de Febrero de 2010].

[27] S. MANJUNATH, B., SALEMBIER, Philippe y SIKORA, Thomas (2002). *Introduction to MPEG-7: multimedia content description interface,* Wiley, West Sussex, Inglaterra, pp. 194-197.

[28] SÁNCHEZ CALLE, Ángel y MORENO DÍAZ, Ana Belén. *Características de los patrones percibidos por un SVA*. Apuntes de Visión Artificial, Ingeniería Informática, Universidad Rey Juan Carlos, 2008. Disponible en Web: http://www.gavab.es/wiki/download/va/temario/tema4.pdf [Consulta: 10 de noviembre de 2009].

[29] TOLLARI, Sabrina et al (2008). *UPMC/LIP6 at ImageCLEF photo 2008: on the exploitation of visual concepts (VCDT)* [en línea]. Laboratorio de Informática, Universidad Pierre et Marie Curie de Paris, Francia, 2008. Disponible en Web: http://www.clef-campaign.org/2008/working_notes/Tollari_LIP6_paperCLEF2008.pdf [Consulta: 15 de enero de 2010].

[30] YAVLINSKY, A., SCHOFIELD, E. y RÜGER, S. (2005). *Automated Image Annotation Using Global Features and Robust Noparametric Densitiy Estimation*, Springer, Berlin – Heidelberg, pp. 507-517.

[31] *Capítulo 3.1. Filtros de imágenes en el dominio del espacio y filtros de suavizado* [en línea]. Asignatura de procesamiento digital de imágenes, 5º curso de Ingeniería Informática, Universidad de Sevilla, 2009. Disponible en Web: http://grupo.us.es/gtocoma/pid/tema3-1.pdf [Consulta: 9 de octubre de 2009].

[32] *Descriptores Visuales* [en línea]. Artículo de la enciclopedia virtual "Wikipedia", 2006. Disponible en Web: http://es.wikipedia.org/wiki/Descriptores_visuales [Consulta: 1 de noviembre de 2009].

[33] *Extracción de características de textura basada en la transformada wavelet discreta* [en línea]. Capítulo 3, Escuela Superior de Ingenieros de la Universidad de Sevilla. Disponible en Web: http://bibing.us.es/proyectos/abreproy/11494/fichero/PROYECTO%252FCapitulo+3.pdf [Consulta: 8 de noviembre de 2009].

[34] *Filtrado espacial* [en línea]. Ingeniería en Automatización y Control Industrial, Universidad Nacional de Quilmes, 2005. Disponible en Web: http://iaci.unq.edu.ar/materias/vision/archivos/apuntes/Filtrado%20Espacial.pdf [Consulta: 8 de septiembre de 2009].

[35] *Image CLEF – Image Retrieval in CLEF* [en línea]. Descripción de las tareas propuestas por CLEF para la recuperación de información multilingüe y multimedia, 2008. Disponible en Web: http://www.imageclef.org [Consulta: 20 de julio de 2009].

[36] *Lenna* [en línea]. Artículo de la enciclopedia virtual "Wikipedia", 2007.
Disponible en Web: http://es.wikipedia.org/wiki/Lenna [Consulta: 9 de octubre de 2009].

[37] *Maquinas de vectores de soporte* [en línea]. Artículo de la enciclopedia virtual "Wikipedia", 2007.
Disponible en Web: http://es.wikipedia.org/wiki/M%C3%A1quinas_de_vectores_de_soporte
[Consulta: 21 de febrero de 2010].

[38] *Modelo de color CMYK* [en línea]. Artículo de la enciclopedia virtual "Wikipedia", 2008. Disponible en Web: http://es.wikipedia.org/wiki/Modelo_CMYK [Consulta: 25 de noviembre de 2009].

[39] *Modelo de color HSV* [en línea]. Artículo de la enciclopedia virtual "Wikipedia", 2007.
Disponible en Web: http://es.wikipedia.org/wiki/Modelo_de_color_HSV [Consulta: 8 de octubre de 2009].

[40] *Red neuronal artificial* [en línea]. Artículo de la enciclopedia virtual "Wikipedia", 2008.
Disponible en Web: http://es.wikipedia.org/wiki/Red_neuronal_artificial [Consulta: 18 de febrero de 2010].

[41] *Segmentación* [en línea]. Curso: "Tópicos avanzados en la visualización: reconocimiento 3D", servicio de Computación, Instituto Politécnico Nacional de México, 7 de junio de 2002. Disponible en Web: http://delta.cs.cinvestav.mx/~fraga/Cursos/Reconocimiento3D/2005/segmentacion.pdf
[Consulta: 10 de octubre de 2009].

[42] *Técnicas de segmentación de imágenes* [en línea]. Curso: "El ordenador en el laboratorio: medir con imágenes." dpto. de Física Matemática y Fluidos, UNED, 2001.
Disponible en Web: http://dfmf.uned.es/actividades/no_reglada/laboratorio/segmentaci%F3n2.pdf
[Consulta: 10 de octubre de 2009].