Universidad Zaragoza
1542

Trabajo Fin de Grado

Desarrollo de un sistema de procesado de imagen para la obtención de imágenes similares

Development of an image processing system to retrieve similar images

Autor

Isabel Querol Cisneros

Director/es

Emiliano Bernués

Escuela de Ingeniería y Arquitectura

Junio 2018

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. _____,

con nº de DNI _____ en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) _____, (Título del Trabajo)

_____

_____

_____

_____

_____

_____,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, _____

Fdo: _____

Agradecimientos

A mi director y coordinador del trabajo, Emiliano Bernués, por guiarme en la realización del trabajo y ayudarme en todo momento.

A mi familia, en especial a mi padre, por interesarse en mi trabajo y darme ánimos para ser inconformista y no dar por bueno cualquier resultado.

A David Recuenco, mi mentor en Adidas, por darme la posibilidad y el conocimiento necesario para hacer accesible el sistema a cualquier usuario.

Índice

Desarrollo de un sistema de procesado de imagen para la obtención de imágenes similares

Resumen del proyecto

Este proyecto forma parte de una línea de investigación del área de la teoría de la Señal y Comunicaciones del Departamento de Ingeniería Electrónica y Comunicaciones de la Universidad de Zaragoza. Tiene como objeto el desarrollo de un sistema adaptativo que, mediante la extracción de las características adecuadas, sea capaz de encontrar en una base de datos limitada las imágenes más parecidas a una dada.

Estas características describen el contenido de cada imagen, permitiendo calcular el parecido entre dos imágenes matemáticamente.

Las imágenes quedan definidas por su color, forma y textura, cuyos valores se almacenan en un vector de características propio de cada imagen.

Se pretende que este vector sea lo más corto posible para que la búsqueda sea rápida, pero sin perder la información esencial de la que dependen los resultados.

El desarrollo muestra un cambio gradual de la estrategia escogida. Se parte una extracción de la información básica de la imagen y se progresa hacia una descripción estadística de la misma.

1. Introducción

1.1 Motivación y contexto del proyecto

Los avances en almacenamiento de datos y las tecnologías de recuperación de imágenes han permitido la creación de las grandes bases de datos de las que disponemos en la actualidad.

En la era de la información, es de vital importancia gestionar todas estas colecciones mediante el desarrollo de sistemas de información.

Tradicionalmente los sistemas de clasificación de imágenes se basan en el análisis de características visuales de bajo nivel presentes en dichas imágenes para lograr significados semánticos de alto nivel [1] [2].

Algunos de los sistemas más famosos que abordan este problema son los sistemas CBIR (Content-Based Image Retrieval). Estos sistemas tratan de encontrar imágenes similares basándose en las propiedades del contenido de estas [3] [4] [5]. Una de las principales ventajas que ofrecen, es la posibilidad de encontrar imágenes sin necesidad de que estas estén etiquetadas.

La tecnología CBIR tiene múltiples aplicaciones, desde medicina hasta sistemas biométricos y librerías digitales [6].

1.2 Objetivos

Este proyecto pretende el desarrollo de un sistema CBIR basándose en otros sistemas similares con el objetivo de:

1) Demostrar de forma práctica la eficacia de los sistemas CBIR en la búsqueda de imágenes visualmente similares.
2) Encontrar las estrategias más eficientes, tanto en procesado como en resultados, para describir una imagen en base a su contenido. Se prueban distintos planos de color, filtrados, propiedades estadísticas…
3) Estudiar algunos de los sistemas CBIR desarrollados y elegir de estos aquellas técnicas más interesantes y que proporcionan mejores resultados.
4) Desarrollar un sistema en Visual Studio que muestre las posibilidades de una aplicación CBIR con recursos limitados (Una base de datos de 100.000 imágenes y un sólo PC).

1.3. Metodología del trabajo

Fases del trabajo:

Fase 1. Estudio del estado del arte

Se estudian los distintos tipos de sistemas y tecnologías existentes, desarrollados para la búsqueda de imágenes similares, como son SIMPLE [7], CBIR, redes neuronales [8]…

Fase 2. Análisis de sistemas CBIR concretos

Se analizan en profundidad distintos proyectos de desarrollo de estos sistemas (las características escogidas, la forma de extracción, los parámetros, los resultados…).

Fase 3. Elección de la base de datos

Se escoge la base de datos Places365_test proporcionada por *MIT Computer Science and Artificial Intelligence Laboratory* para poner a prueba el sistema y poder analizar los resultados.

Fase 4. Programación del sistema de extracción de características

Desarrollo del algoritmo de extracción de características en C++ utilizando Visual Studio como editor de código. Para disponer de las funciones necesarias para el procesado de imagen se incluyen en el programa las librerías de OpenCv. Estas permiten el visualizado de la imagen, los cambios de planos de color, algunos filtrados… Además, se ha utilizado Matlab como herramienta matemática para probar partes del algoritmo y comprobar la eficiencia de determinadas funciones en la extracción de información.

El sistema tiene como objetivo la búsqueda de imágenes similares mediante un algoritmo que no necesita entrenamiento previo. Esto se consigue a través de la extracción de las características fundamentales de las imágenes.

Tanto la imagen que se introduce en el sistema, como aquellas de la base de datos con las que va a ser comparada, han de ser procesadas de la misma forma. Esto permite describir las imágenes mediante vectores numéricos que son fácilmente comparables.

El parecido entre dos imágenes será determinado por la distancia euclídea entre sus vectores de características.

En un fichero quedan almacenados todos los vectores de características asociados a las imágenes de la base de datos. En todo momento ha de tenerse en cuenta el tamaño de los vectores de características; cuanto mayor sean estos, mayor será el fichero que los almacena y el tiempo de procesado del sistema.

Para resolver este problema, se aplican modelos estadísticos que permiten describir la imagen reduciendo el número de componentes añadidas al vector.

Estos modelados, además, posibilitan caracterizar la imagen de forma simplificada, pero teniendo en cuenta toda su información.

Fase 5. Selección de los pesos

Se escogen los pesos utilizados en el clasificador del sistema. Los pesos determinan la importancia de unas características sobre otras a la hora de calcular el parecido entre imágenes.

Fase 6. Paralelización del algoritmo

Se paraleliza el algoritmo como estrategia para reducir los tiempos de procesado. Como sólo se cuenta con un procesador, las mejoras son del 20%, pero potencialmente podría desplegarse el sistema para conseguir mejoras superiores.

Fase 7. Desarrollo de la interfaz

Se desarrolla una aplicación cliente-servidor que expone la información obtenida por el programa de forma visual y fácil de entender para un usuario cualquiera.

El código de back-end se programa en Java Script utilizando la librería "Express" que proporciona las funciones necesarias para establecer y controlar un servidor http.

El código de front-end se programa en Java-script, html y css, utilizando diversas herramientas:

- Vue: framework que permite utilizar funcionalidades de Javascript en códigos html.
- Webpack: es un empaquetador de código. Mediante la adición de "loaders" como Babel se *transpila* el código, de forma que pueda ser entendido por cualquier navegador.
- Bootstrap: *framework* que permite facilitar el diseño web.

El cliente envía una imagen al servidor, que la procesa utilizando el sistema desarrollado. El servidor devuelve al cliente el nombre de las imágenes más similares obtenidas por el sistema. Finalmente, el cliente pide estas imágenes al servidor y las representa en la interfaz.

1.4. Estudio del estado del arte

La mayoría de los sistemas de búsqueda de imagen tradicionales se basan en búsqueda por texto. Sin embargo, esto puede llevar a resultados no deseados.

Sistemas más recientes se basan en el contenido de las imágenes para realizar esta búsqueda. Aunque son computacionalmente más caros, consiguen resultados más acertados que los sistemas de clasificación tradicionales [9].

Los sistemas CBIR que se han estudiado y en los que se basa el proyecto definen las imágenes extrayendo características de:

A. COLOR

La comparación de color entre imágenes es el método más importante y más básico en los sistemas CBIR. Las características de color son las más intuitivas y las más importantes en para la percepción de la imagen. Las características de color son muy estables y robustas en comparación con otras características como la textura o la forma.

Los histogramas de color constituyen una herramienta muy potente para encontrar imágenes [10]. El espacio de color HSV es utilizado en múltiples sistemas [6] [9] ya que los niveles de matiz y saturación están más próximos al sistema de visión humano [11] [12] [13].

No sólo los histogramas pueden describir la información de color. Las componentes de color que serán incluidas en el vector pueden calcularse como la media de intensidad de color de los pixeles en una región pequeña de la imagen [14].

B. FORMA

La extracción de la forma permite identificar la localización y el tamaño de los objetos presentes en la imagen.

La forma representa la configuración de la superficie de una imagen o el contorno de un objeto presente en esta [15].

S. D. Ruikar y R. S. Kabade, [9] utilizan el filtro "Prewitt" para encontrar la forma de la imagen. Este filtro paso alto elimina las bajas frecuencias de la imagen y consigue localizar los contornos de los objetos existentes.

Los descriptores EDH (Edge Histogram Descriptor) también están basados en el filtrado de la imagen. Estos descriptores distinguen cinco tipos de borde diferentes y, por tanto, se utilizan cinco filtros distintos para distinguirlos [6].

C. TEXTURA

La textura determina la superficie o configuración de la imagen. No está asociada a una definición concreta debido a la amplia variabilidad encontrada en distintas aplicaciones [9].

Existen métodos muy diversos para describir esta característica.

La transformada Wavelet Discreta [16] descompone la imagen en sus componentes de baja y alta frecuencia. Con ello logra distinguirse la configuración de la imagen, a través de sus contornos y su información más básica [6] [9].

La matriz de co-ocurrencia (GLCM, Gray-Level Coocurrence Matrix) almacena relaciones espaciales entre valores de pixel. Así se describe la variabilidad como el número de veces que se repite una diferencia de valores de intensidad para una distancia de pixeles determinada.

Los valores asociados a las características extraídas quedan almacenados en vectores de características. Estos vectores describen las imágenes y permiten compararlas de forma matemática. Existen métodos muy variados para el cálculo de la distancia [17].

La distancia euclídea calculada como el cuadrado de la diferencia es la más utilizada [9] [14] aunque existen sistemas que utilizan otros cálculos como la distancia Manhattan [6].

El algoritmo puede mejorarse asignando pesos a las diferentes características que permitan dar una importancia mayor a unas sobre otras [9].

Existen además diferentes estrategias que permiten mejorar los sistemas CBIR:

1. D. Xu y Z. Qu [18] proponen un sistema que, tras extraer las características de color, clasifica las regiones de una imagen en similares y no similares y extrae la transformada TF-IDF [10]. Esta transformada distingue las regiones más relevantes para describir la imagen en cuestión. Con esta información, la imagen se incluye en una clase (utilizando el algoritmo KNN [19], *k-nearest neighbor*) y con ello se logra, por ejemplo, encontrar imágenes que presentan el mismo objeto principal.
2. Los sistemas SIMPLE (Searching Images with MPEG-7 (&MPEG-7-like) Powered Localized Descriptors) [20] [7] utilizan descriptores globales para la extracción del contenido de las imágenes (como el descriptor EHD utilizado en el proyecto). Estos sistemas, mediante detectores de puntos de interés (como SURF [21]), encuentran "parches" salientes de la imagen que a continuación se clasifican mediante el uso de descriptores globales.
3. Las redes neuronales convolucionales consiguen clasificar las imágenes en un amplio número de categorías [8]. La desventaja de estos sistemas es la necesidad de un entrenamiento previo del sistema.

2. Desarrollo del proyecto

2.1 Descripción general

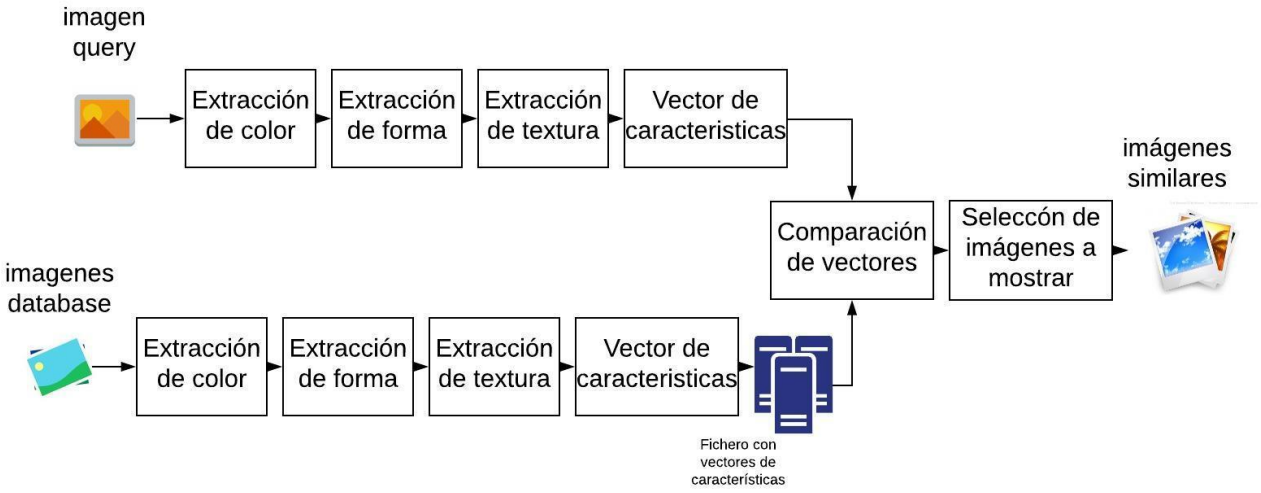


Ilustración 1. Esquema general del sistema

El algoritmo desarrollado para lograr el sistema de búsqueda de imágenes, comienza con la extracción de sus características básicas.

Se extrae la información de color, forma y textura de todas y cada una de las imágenes presentes en la base de datos descargada.

Este procesado de imagen se aplica tambien a la que introduce el usuario en el sistema, y es la parte fundamental del algoritmo.

La información de cada imagen se almacena en un vector de características guardando siempre el orden de almacenamiento de las componentes de color, forma y textura.

Es necesario que tanto la extracción como el almacenamiento de las componentes se apliquen de la misma forma a la imagen introducida y a todas las imágenes de la base de datos. Esto permite la correcta comparación y el cálculo de la distancia que representa el parecido entre imágenes.

Una vez se han construido todos los vectores de características, se compara el vector de la imagen *query* con todos los vectores de las imágenes de la base de datos. La distancia entre vectores es la distancia euclídea componente a componente.

La elección de los pesos y el número de imágenes a mostrar forman parte del algoritmo de decisión. Los pesos permiten dar una importancia superior a unas características sobre otras y son elegidos en base a la percepción visual humana.

A continuación se detallan las diferentes partes del proyecto, desde la extracción de carácterísticas hasta la elección de los pesos.

En todo momento se tiene en cuenta el tamaño del vector de características del que depende el tiempo de procesado del programa.

2.2. Extracción de las características

Ilustración 2. Maiz.jpg

Los resultados para las diferentes partes del algoritmo se mostrarán utilizando esta imagen como ejemplo.

2.2.1. El color

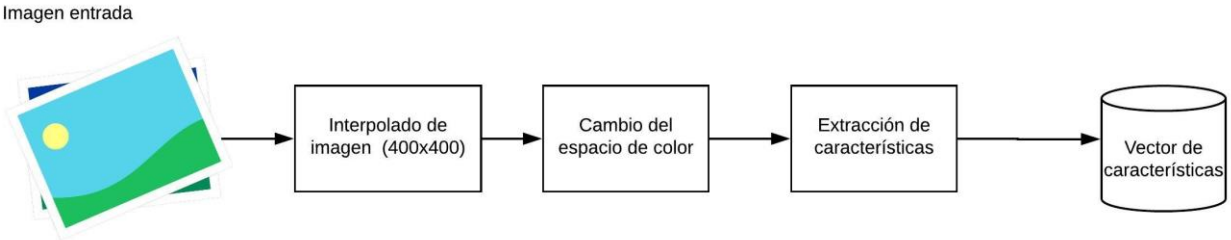


Ilustración 3. Esquema general de extracción del color

El color es la característica más importante a la hora de describir la imagen, influye enormemente en la forma en la que la percibimos.

El color está muy ligado al contenido semántico de la imagen. El mar o el cielo en la gran mayoría de las imágenes aparecerán de color azulado mientras que la hierba o las copas de los arboles serán de color verde.

Pese a que es estable y robusto en comparación con otras características, se debe prestar mucha atención a la hora de tratar el color ya que la comparación de los valores matemáticos que lo representan no siempre está correlada con la diferencia visual que se percibe.

En este proyecto se han utilizado dos sistemas distintos para extraer la información de color: la media por regiones y el histograma.

Estos sistemas se han probado en diferentes espacios de color de la imagen: HSV, RGB y CIELab.

En el sistema de extracción de la información se busca que las componentes calculadas sean representativas de la información global de la imagen. Cumpliendo esta primera premisa, se pretende minimizar al máximo el número de componentes.

2.2.1.1. Media por regiones

Los valores de intensidad de los pixeles de una imagen para los tres canales de color, tienden a variar de forma gradual, por regiones.

Siguiendo con el ejemplo anterior, en una imagen en la que aparece el cielo los niveles de intensidad RGB presentaran pequeñas variaciones en la región de la imagen que represente ese cielo. Tendría sentido representar todo ese cielo con un solo tono de azul.

La media por regiones tiene como objetivo reducir la información de color extraída de la imagen y conseguir que esta sea comparable.

Esta herramienta permite eliminar el posible ruido presente en la imagen, que introduciría componentes no representativas de su color.

La imagen se divide en regiones más pequeñas y sobre cada una de ellas se calcula la media de la intensidad de cada canal de color. Los valores de estas medias se incluyen directamente en el vector.

El color es la forma en la que el sistema de percepción humano mide una parte del espectro electromagnético. Debido a algunas propiedades de este sistema de percepción, no somos capaces de ver todas las posibles combinaciones del espectro visible y agrupamos algunas zonas espectrales en colores [22].

La información de color de la imagen puede ser representada en distintos espacios de color. Los espacios de color son sistemas de interpretación que representan los colores mediante números o coordenadas.

Algunos espacios de color como RGB o HSV están basados en el sistema de percepción humano. Estos espacios resultan de especial interés para el desarrollo de la aplicación en cuestión, se buscan imágenes visualmente similares.

La teoría tricromática [22] establece que existen tres tipos de foto receptores aproximadamente, sensibles al rojo, azul y verde. Existen de hecho, tres tipos de conos sensibles a longitudes de onda largas, medias y cortas.

Por ello, se utiliza el espacio de color RGB, en primer lugar, en el que el color se describe mediante tres componentes: R (rojo), G (verde) y B (azul).

En el vector se incluye la media de cada una de las componentes de cada región, es decir, las componentes de color incluidas en el vector son:

$$3 \times Número\ de\ regiones \tag{1}$$

La imagen completa presenta 400x400 pixeles y se divide en regiones de 20x20, por tanto:

$$Longitud\ del\ vector = 3 \times \frac{400}{20} \times \frac{400}{20} = 1200 \tag{2}$$

Ilustración 4. "Maiz.jpg" transformada a 400x400 pixeles

Ilustración 5. Media de color de la imagen "Maiz.jpg"

El espacio de color RGB presenta dos grandes desventajas. En primer lugar, no es intuitivo psicologicamente, los humanos tienen problemas a la hora de visualizar un color definido en RGB.

En segundo lugar, existe una correlación muy baja entre la diferencia de colores percibida y la distancia euclídea en el espacio RGB [22].

Por estas razones, no se obtuvieron buenos resultados para este espacio de color.

Los colores como el verde y el marron o el amarillo se confundían.

Uno de los pioneros de la ciencia del color, Isaac Newton, dispuso los colores en un circulo llamado el circulo de Newton. Este círculo utiliza las características de matiz y saturación para describir los colores, y resultó ser la forma más natural para los humanos de describirlos. El sistema de visión humano tiende a organizar los colores por matiz, saturación y brillo [22].

Esta representación de colores es la base de los llamados "phenomenal color spaces" que representan el color mediante tres componentes H, S y V.

H (Hue) representa el matiz, y es la componente más importante de las tres a la hora de distinguir los colores. Las otras dos componentes son S (Saturation) que identifica la saturación y V (value) que esta relacionada con la intensidad del color [23].

Este espacio de color no presenta las dos desventajas que presenta el espacio RGB y que influyen negativamente en los resultados.

Con una sencilla fórmula, puede cambiarse el espacio de color desde RGB a HSV:

$$H = \cos^{-1} \frac{\frac{1}{2}|R-G|+|R-B|}{\sqrt{(R-G)^2-(G-B)(R-B)}} \quad (3)$$

$$S = 1 - \left(\frac{3[\min(R,G,B)]}{R+G+B}\right) \quad (4)$$

$$V = \left(\frac{R+G+B}{3}\right) \quad (5)$$

El algoritmo se modifcó, se cambió el espacio de color de la imagen de RGB a HSV. El resto del algoritmo permaneció como se ha explicado.

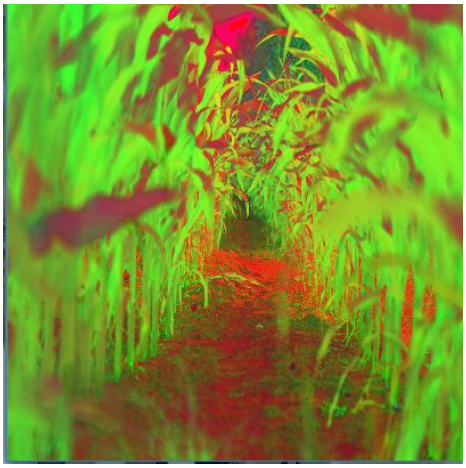


Ilustración 6. Espacio de color HSV de "Maiz.jpg"

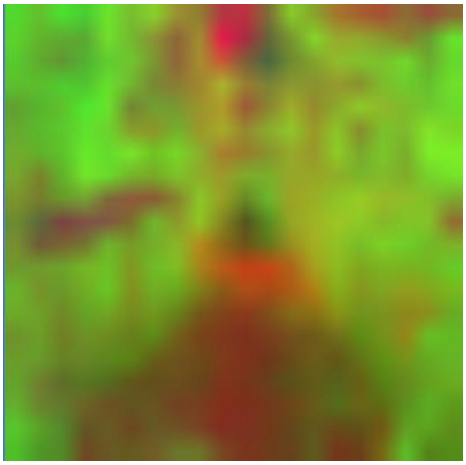


Ilustración 7. Media de color HSV de "Maiz.jpg"

Se presentan mejoras en los resultados pero el porcentaje de imágenes similares sigue siendo demasiado bajo.

Algunas de las imágenes erróneas que se mostraban como resultado en la anterior versión del algoritmo, siguen escogiéndose en esta versión. Se confunden algunos colores como el azul y el blanco.

Existe una discontinuidad de matiz alrededor de los 360º. Esto dificulta las operaciones aritméticas en este espacio de color. Además, existe una mala correlación entre la luz computada y la percibida.

El espacio de color CIElab tiene como objetivo principal lograr un espacio de color perceptualmente igual, es decir, que la distancia Euclídea entre dos colores en el espacio de color CIElab está altamente correlada con la percepción visual humana.

Se intenta en consecuencia un último cambio de espacio de color a CIElab.

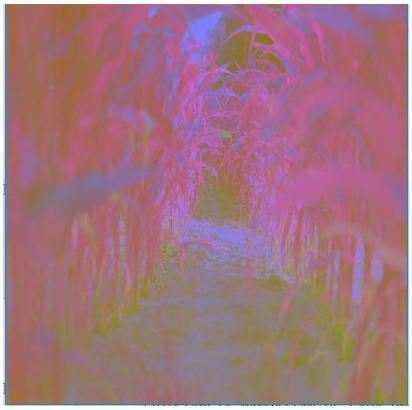


Ilustración 8. Espacio de color CIELab de "Maiz.jpg"

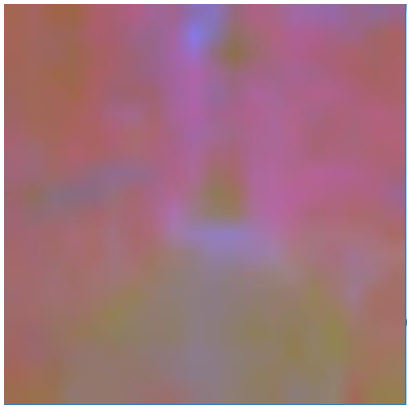


Ilustración 9. Media de color CIELab de "Maiz.jpg"

Los resultados en este último caso no varían, no se producen mejoras.

Se concluye que es la forma de analizar la información la que está fallando y no el espacio de color.

	RGB	HSV	LAB
Maíz	60%	70%	70%
Camión	20%	40%	40%
Playa	10%	20%	20%

Tabla 1. Resultados de precisión para la característica de color en los espacios RGB, HSV y LAB

Este método presenta dos opciones distintas, cada una con un problema asociado.

En primer lugar, puede dividirse la imagen en regiones muy pequeñas. Sin embargo, cuanto más pequeña sea la región, más ruido se introducirá en la información. En consecuencia, dos imágenes de color similar pueden dar como resultado una distancia muy alta.

En segundo lugar, pueden escogerse regiones grandes sobre las que obtener la media. El color que presentarán estas regiones puede no representar el auténtico color de la imagen, se pierde una gran cantidad de información. Como resultado, dos imágenes de color muy diferente pueden presentar una distancia pequeña.

Además, esta forma de comparación de color está muy ligada a la localización de los colores en la imagen.

Por ello, dos imágenes en las que aparezca, por ejemplo, una misma playa, serán consideradas "no similares" si la línea del horizonte cambia notablemente de posición de una a otra.

Ilustración 10. Maiz.jpg

Ilustración 11. Camion.jpg

Ilustración 12. Barca.jpg

RESULTADOS RGB

Ilustración 13. Resultados del espacio RGB para la imagen Maiz.jpg

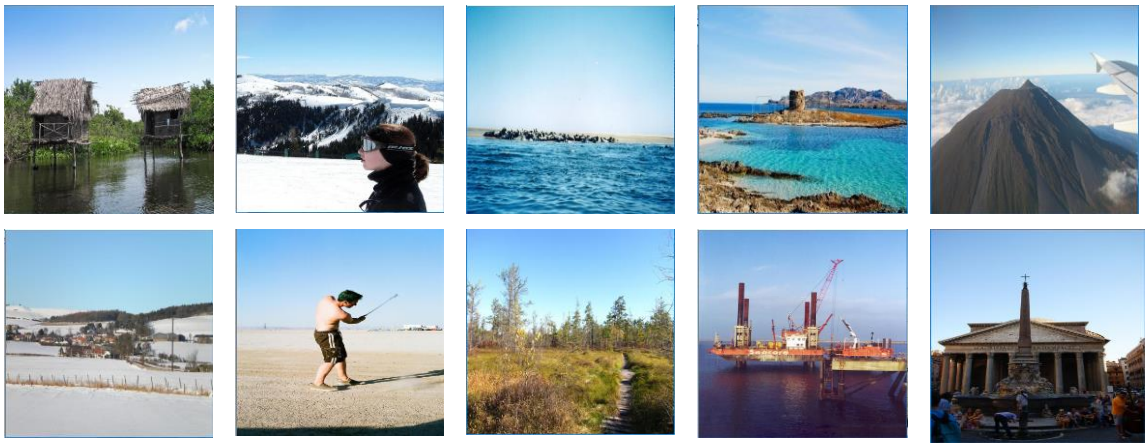


Ilustración 14 . Resultados del espacio RGB para la imagen Camion.jpg

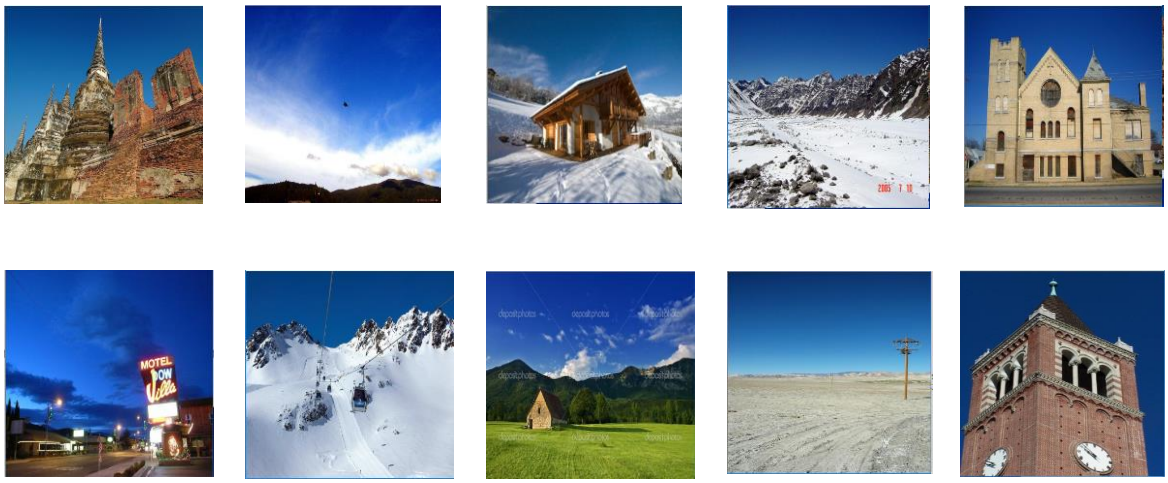


Ilustración 15. Resultados del espacio RGB para la imagen Barca.jpg

RESULTADOS HSV

Ilustración 16. Resultados del espacio HSV para la imagen Maiz.jpg

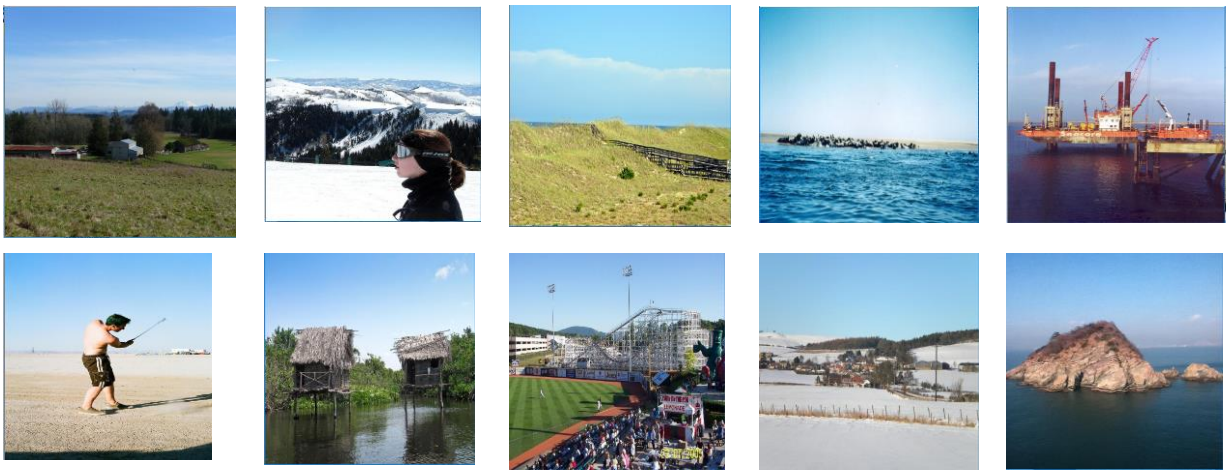


Ilustración 17. Resultados del espacio HSV para la imagen Camion.jpg

Ilustración 18. Resultados del espacio HSV para la imagen Barca.jpg

RESULTADOS CIELab

Ilustración 19. Resultados del espacio CIELab para la imagen Maiz.jpg

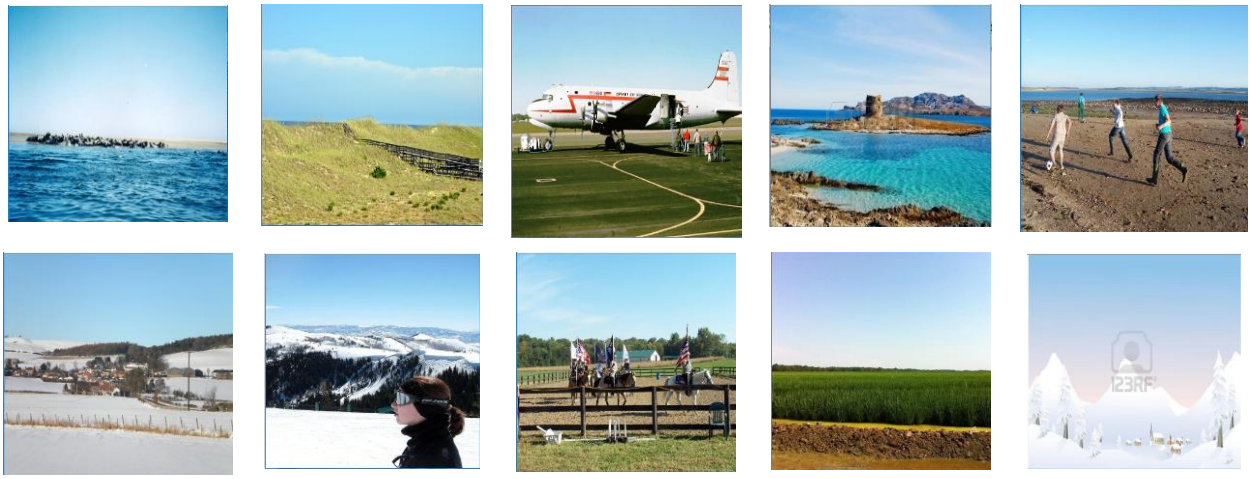


Ilustración 20. Resultados del espacio CIELab para la imagen Camion.jpg

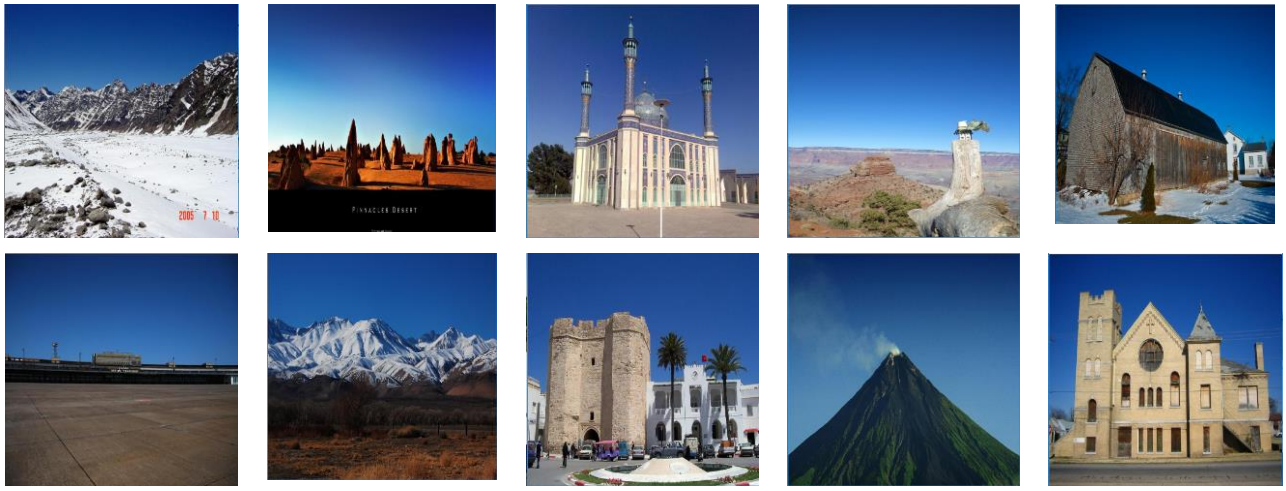


Ilustración 21. Resultados del espacio CIELab para la imagen Barca.jpg

2.2.1.2. Histograma HSV

Un histograma es una representación gráfica mediante barras que informa acerca del grado de presencia de un color en una imagen [6].

Cada valor del histograma representa el número de veces que un color aparece en la imagen. La cuantificación reduce el número total de colores representados por el histograma [6].

El espacio de color CIElab no puede cuantificarse uniformemente porque es un espacio de color no lineal [22].

Teniendo en cuenta la no-linealidad del espacio de color CIElab y las desventajas del espacio de color RGB, se elige para el algoritmo el espacio de color HSV.

Puede calcularse un histograma para cada canal de color o uno conjunto de los tres.

La primera de las técnicas se conoce como *Split histogram* [24]. Esta técnica se basa en calcular un histograma independiente para cada canal de color. Esto tiene como resultado tres histogramas, uno por canal de color. La información proporcionada por este tipo de histograma refleja la distribución de valores para cada canal, pero no informa acerca de los colores presentes en la imagen.

En una aplicación de búsqueda de imágenes similares, se pretende encontrar y comparar los colores presentes en la imagen y no la distribución independiente de los canales de color.

Esto se debe a que dos imágenes con la misma distribución de canales de color independientes pueden ser muy diferentes visualmente.

El valor de histograma se calcula para cada color determinado en la imagen, el color es la combinación de las tres componentes.

La cuantificación 8:4:4 es comúnmente utilizada. Se codifica el matiz con 3 bits (valores posibles entre 0-7) y las otras dos componentes con 2 bits (valores posibles entre 0-3) [24].

El valor del histograma se calcula como:

$$\boldsymbol{valor\ de\ histograma} = 16 \times H + 4 \times S + V \qquad (6)$$

Como resultado se obtienen 128 posibles valores del histograma o colores distintos, y por tanto 128 componentes que van a ser añadidas al vector de características. Estas componentes se encuentran en el rango de valores [0-1].

En el sistema, los valores del histograma se incluyen directamente en el vector de características sin ningún procesado adicional.

	Porcentaje de aciertos (5.000 imágenes)
Maíz	90%
Camión	90%
Playa	70%

Tabla 2. Resultados de precisión para la característica de color como histograma HSV

Tras incluir el histograma HSV para la extracción de la información de color, dejan de confundirse colores lejanos en el espacio de color.

El histograma es robusto ante cambios de localización, rotación e incluso escalado de la imagen, superando en eficiencia el método de la media por regiones.

Además, se reduce la longitud del vector considerablemente; el número de componentes de color disminuye en más de un tercio con respecto al número de componentes obtenidas mediante el cálculo de la media por regiones.

RESULTADOS HISTOGRAMA HSV

Ilustración 22. Resultados del histograma HSV para la imagen Maiz.jpg

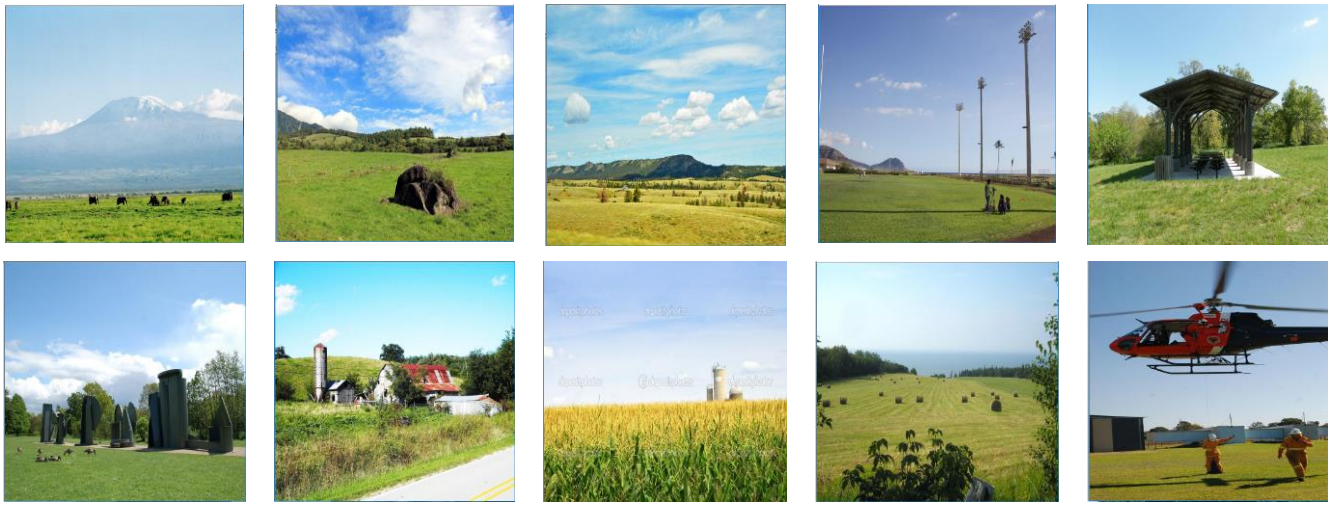


Ilustración 23. Resultados del histograma HSV para la imagen Camion.jpg

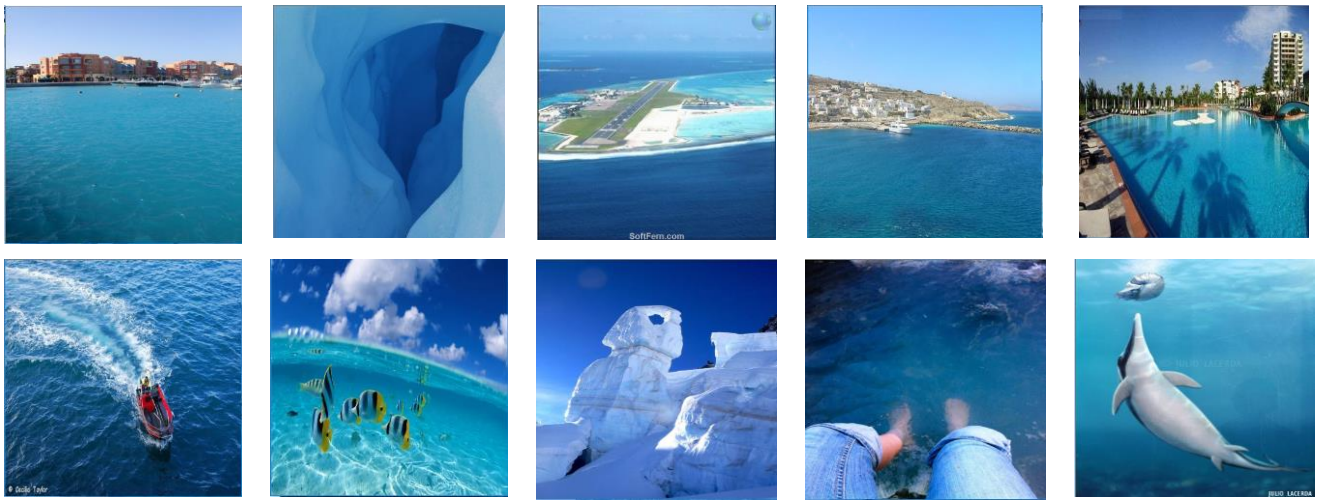


Ilustración 24. Resultados del histograma HSV para la imagen Barca.jpg

2.2.2. La forma

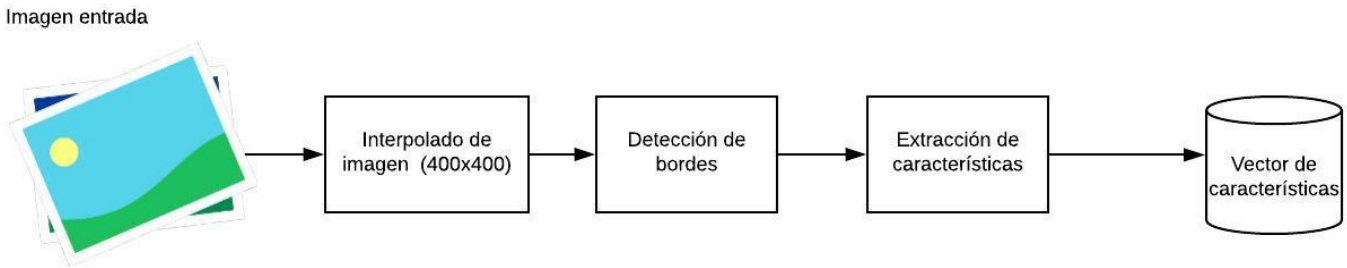


Ilustración 25. Esquema de extracción de la forma

La forma de la imagen queda representada por los contornos de los elementos existentes en la misma. La extracción de la forma es de gran utilidad para identificar objetos presentes en la imagen y su ubicación dentro de esta.

Para extraer la forma de la imagen, se deben localizar los contornos o bordes de los objetos que la caracterizan.

Estos bordes representan cambios bruscos en los valores de intensidad de los pixeles, es decir altas frecuencias de la imagen. Una forma de localizar estas altas frecuencias de la imagen es eliminar las bajas frecuencias por medio de un filtrado paso bajo.

2.2.2.1. Filtro de Canny

El detector de bordes de Canny fue creado por John F. Canny en 1986, también se conoce como el detector óptimo y presenta los siguientes criterios esenciales:

-Tasa de error baja: lo único detectado son bordes y no otros elementos de la imagen.

-Distancia mínima entre la ubicación de los pixeles que forman parte del borde detectado y la de los pixeles reales de la imagen que forman el borde buscado.

-Respuesta mínima, sólo se recibe una respuesta del detector por borde detectado [25].

El filtrado se realiza en distintos pasos:

1. Se elimina el posible ruido presente en la imagen mediante un filtrado gaussiano. Aquí se muestra un ejemplo de un filtro Gaussiano con tamaño de núcleo igual a 5:

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 3 & 4 & 2 \end{bmatrix} \quad (7)$$

2. Se aplican dos máscaras convolucionales en las direcciones *x* (horizontal) e *y* (vertical) de la imagen:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad (8)$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \quad (9)$$

3. A continuación, se calculan la fuerza y el gradiente mediante:

$$G = \sqrt{{G_x}^2 + {G_y}^3} \quad (10)$$

$$\theta = \tan^{-1}\left(\frac{G_x}{G_y}\right) \quad (11)$$

El gradiente se redondea a uno de estos cuatro posibles angulos: 0º, 45º, 90º y 135º.

4. Se eliminan los pixeles que no se consideran parte del borde (Non-maximum supression), solo quedarán finas lineas, los bordes candidatos.

5. Histeresis. Este es el últmo paso, se eligen dos umbrales, uno superior y otro inferior. Se aceptan los pixeles cuyo gradiente sea mayor al umbral superior y se rechazan aquellos cuyo gradiente sea menor al umbral inferior. Los pixeles cuyo gradiente se situe entre ambos umbrales, solo serán aceptados si están conectados a un pixel cuyo gradiente es mayor al umbral superior.

   Para el desarrollo de este trabajo se escogieron los siguientes valores:

   $$Kernel\ size = 3$$

   $$radio = 3$$

   $$umbral\ inferior = 30$$

   $$umbral\ superior = umbral\ inferior \times radio$$

Tras localizar mediante el algoritmo de Canny los bordes en la imagen, se convierte la imagen de los bordes a escala de grises pues la información de color no va a ser relevante para la identificación de la forma. La imagen tiene 400x400 pixeles.

Para poder normalizar los valores de pixel de los bordes de manera evidente entre [0-255], todos aquellos pixeles de la imagen de bordes cuyo valor sea distinto de 0 pasan a tener un valor de 255. En el algoritmo propuesto no interesa el valor de intensidad del pixel que forma el borde, sino su posición en la imagen.

Para conseguir que las componentes almacenadas en el vector sean representativas de la información de la imagen y eliminar parte del ruido introducido, se utiliza la técnica de la media por regiones.

Se escogen regiones de 20x20 por lo que la forma queda representada por 400 componentes en el vector de características.

Las ilustraciones 26-29 representan el proceso completo realizado sobre la imagen ejemplo "Maiz.jpg":

Ilustración 26. Filtrado de bordes en color

Ilustración 27. Filtrado de bordes en blanco y negro

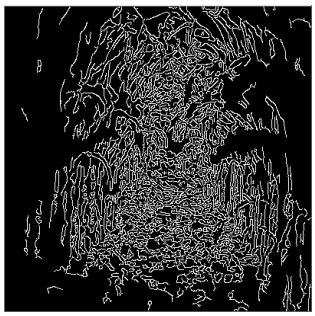


Ilustración 28. Histéresis valores 0 o 255

Ilustración 29. Media calculada en regiones de 20x20

La media por regiones en la extracción de la forma, presenta problemas similares a los detallados en el apartado 2.2.1.1. Este método se ve afectado por posibles cambios en la localización de los objetos, la rotación o el escalado de la imagen.

El algoritmo, asimismo, no distingue entre tipos de borde por lo que se ignora la estructura exacta de los elemenots de la imagen.

Esta es una de las imágenes que se obtenía como resultado utilizando la imagen "Maiz.jpg" como query:

Ilustración 30. Places365_test_00002504.jpg

Puede verse que la máxima variabilidad se encuentra en la parte central pero el sistema no ha tenido en cuenta cómo eran los bordes presentes en la imagen "Maiz.jpg".

2.2.2.2. Edge Histogram Descriptor

El histograma es una de las estructuras más utilizadas para extraer características globales de la imagen. Es invariante a posibles cambios como rotación y traslación y la cuantificación lo hace además robusto a los cambios de tamaño de la imagen.

Los bordes pueden ser representados por medio de histogramas. Un histograma de bordes representa la frecuencia y la direccionalidad de los cambios de brillo de la imagen. La información acerca de los bordes no puede ser duplicada por un histograma de color o por las características homogéneas de la textura [26].

Los descriptores EHD (Edge Histogram Descriptors) representan la frecuencia de aparición de 5 tipos de borde diferentes en una región más pequeña de la imagen a la que se denomina sub-imagen.

Para encontrar estos 5 tipos de borde es necesario filtrar la imagen de 5 formas diferentes:

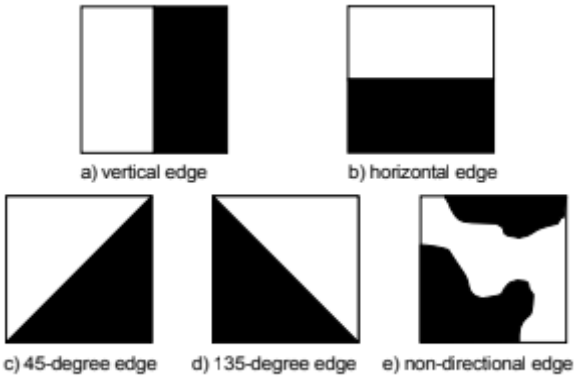


Ilustración 31.Tipos de bordes en EHD

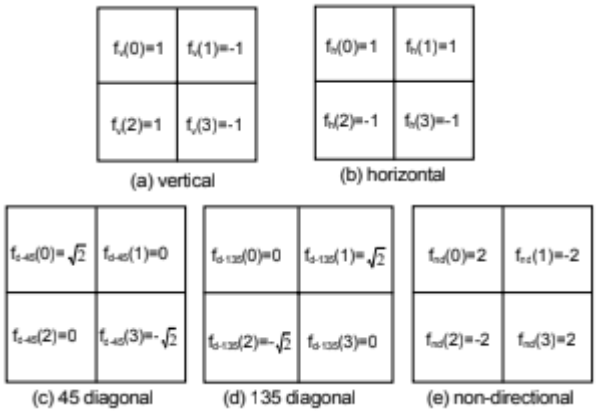


Ilustración 32. Filtros utilizados en EHD para cada tipo de borde

El proceso de construcción del histograma sigue estos pasos:

1. En primer lugar, se divide la imagen completa en regiones del mismo tamaño a las que llamaremos sub-imágenes. En el caso del sistema propuesto la imagen de 400x400 pixeles se divide en 16 sub-imágenes de 100x100 pixeles.

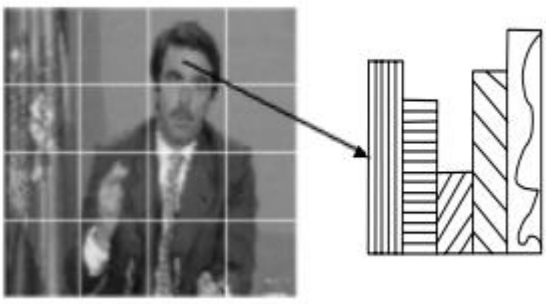


Ilustración 33. Imagen dividida en sub-imágenes

2. Cada una de las sub-imágenes se divide en bloques más pequeños a los que llamaremos bloques. Las imágenes de 100x100 pixeles se dividen en 25x25 bloques de tamaño 4x4 pixeles.

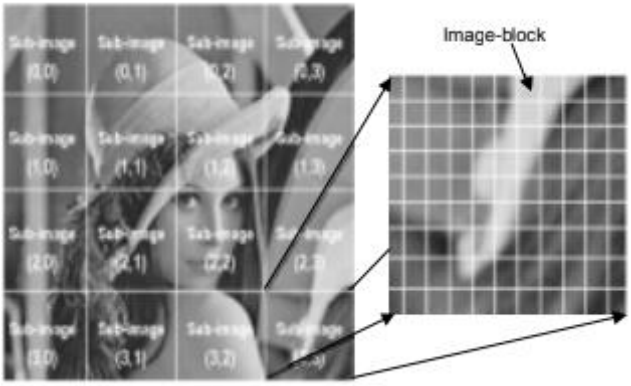


Ilustración 34. Bloques en los que se divide cada sub-imagen

3. Para poder realizar el filtrado de bloques 2x2, se convierten los bloques 4x4 en bloques 2x2 calculando la media en cada una de las 4 regiones de 2x2 sub-bloques presentes en un bloque 4x4.

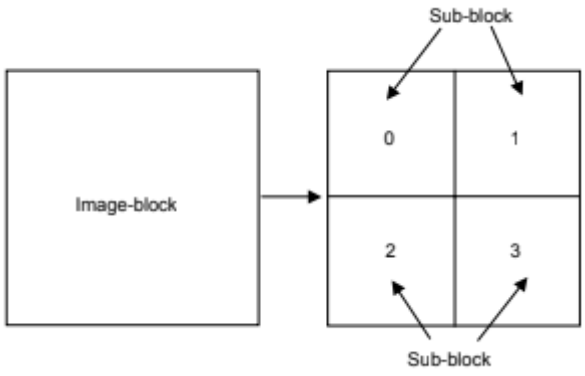


Ilustración 35. Sub-bloques en los que se dividen los bloques

4. Tras el filtrado se clasifica el tipo de borde que representa el bloque (si representa algún tipo de borde). El valor del histograma que represente ese tipo de borde en esa sub-imagen aumenta en una unidad.
5. Finalmente, el histograma se normaliza dividiendo por el número total de bloques.

En el vector de características se incluyen tres histogramas distintos, todos ellos obtenidos mediante el método explicado previamente.

En primer lugar, encontramos el histograma local que presenta 80 componentes:

$$n_{componentes}\ histograma_{local} = 16_{subimagenes} \times 5_{bordes} \quad (9)$$

En segundo lugar, un histograma global que representa la frecuencia de aparición de cada uno de los bordes en la imagen completa. Tiene tantos componentes como tipos de borde, es decir, cinco.

Por último, tenemos un histograma semiglobal que también se incluye en el vector. Este histograma identifica la frecuencia de aparición de los distintos tipos de bordes en diferentes agrupaciones de sub-imágenes. Las sub-imágenes se agrupan de 13 formas distintas, por lo que este histograma presenta 65 componentes.

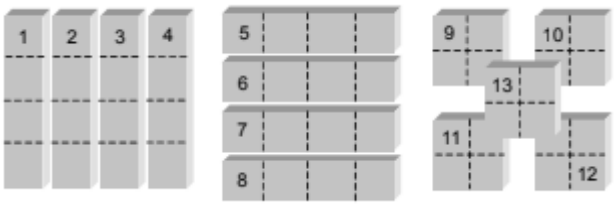


Ilustración 36. Grupos de sub-imágenes para calcular el histograma semiglobal

Se suman al vector un total de 150 componentes que representan la forma y cuyos valores se encuentran en un rango [0-1].

Incluir el histograma global y semiglobal permite identificar elementos con formas similares en la imagen sin que estos se localicen en la misma región exacta de la imagen.

Este procesado mejora en gran medida los resultados con respecto al anterior. El sistema es independiente de la localización de los elementos. Los distintos tipos de borde que distingue el algoritmo permiten determinar la configuración de los objetos de la imagen.

Es capaz de distinguir elementos como, por ejemplo, las hojas de la imagen "Maiz.jpg".

Ilustración 37. Imagen más similar a la imagen "Maiz.jpg" (Teniendo solo en cuenta los descriptores EHD)

2.2.3. La textura

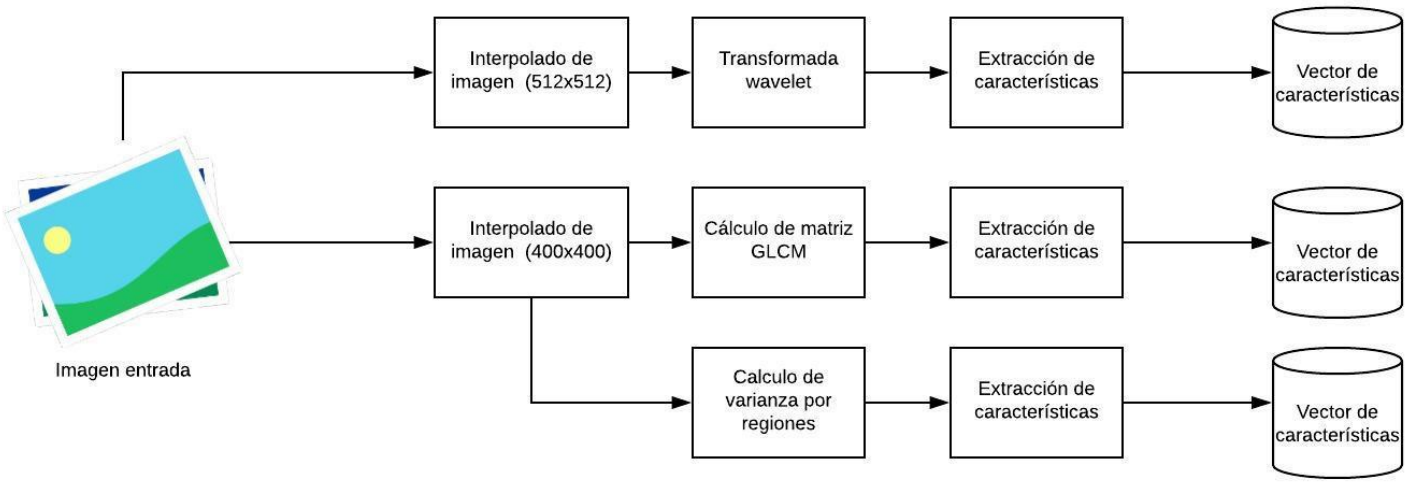


Ilustración 38. Esquema de extracción de textura

La textura proporciona información acerca de la disposición espacial de la intensidad de los pixeles de una imagen o de una región de la imagen. El estudio de la textura permite distinguir superficies, terrenos o tejidos. Una textura se considera rugosa si la diferencia entre los valores máximos y mínimos es grande y si su separación es importante respecto del filtro empleado en el análisis. Si las distancias y la separación son pequeñas, la textura se considera suave.

La clasificación y recuperación de imágenes a través del análisis de la textura, presenta tres enfoques diferentes: estadístico, espectral y estructural. Las técnicas estadísticas de caracterización se consideran óptimas para lograr la clasificación de imágenes.

Las técnicas de descomposición de wavelet y la matriz de co-ocurrencia (GLCM) son dos herramientas muy útiles para lograr la representación estadística de la textura.

2.2.3.1. La transformada wavelet

La transformada wavelet es la base para cortar datos o funciones en diferentes componentes de frecuencia y estudiar cada uno de esos componentes con una resolución ajustada a su escala.

Lo que diferencia a la transformada wavelet de otras transformadas es su dependencia con el tiempo y la frecuencia, es por ello adecuada en el análisis de señales no estacionarias. Se utiliza para representar la aproximación y el detalle de la imagen [9]. Permite capturar las altas frecuencias de las zonas de los bordes y las bajas de las zonas homogéneas, agrupando la mayor parte de la energía de la imagen en una pequeña porción de los coeficientes de la transformación.

El diagrama de flujo de la transformada wavelet se expone en la ilustración 18. En este diagrama L hace referencia a las bajas frecuencias y H a las altas frecuencias. También se muestran en esta ilustración distintos niveles de la transformada, el nivel 1 y el nivel 2 concretamente.

La sub-imagen *LL* es la imagen compuesta por las bajas frecuencias de la imagen original. Representa la información básica de la imagen. La sub-imagen *HL* es el componente de baja frecuencia en la dirección horizontal y alta frecuencia en la dirección vertical, en ella se manifiestan los bordes horizontales de la imagen original.

La sub-imagen *LH* es el componente de alta frecuencia en dirección horizontal y baja frecuencia en la dirección vertical, representa los bordes verticales de la imagen.

Por último, la sub-imagen *HH* identifica el componente de alta frecuencia, manifiesta los bordes oblicuos de la imagen original.

La mayor parte de la energía de la imagen original se concentra en la región *LL* de la transformada [3].

Ilustración 39, Diagrama de flujo de la transformada Wavelet

La transformada wavelet discreta de Haar se considera la precursora de las transformadas de Wavelet y el primer paso en la construcción de sus algoritmos de aplicaciones. Las funciones de

Haar componen una primera base de funciones muy sencillas y con importantes propiedades como la ortonormalidad, permitiendo la descomposición y reconstrucción perfecta de las señales.

Para una entrada representada por una lista de 2^n componentes, la transformada wavelet de Haar empareja los valores de entrada guardando la diferencia y utilizando la suma para calcular el siguiente nivel de la transformada.

Es necesario por tanto que la entrada a esta función sea potencia de 2^n por lo que la imagen se escala a un tamaño de 512x512 pixeles. Se elige este tamaño porque es suficientemente pequeño para no distorsionar las imágenes de menor tamaño sin que implique perder demasiada información por la reducción de la imagen.

Como se ha explicado, es la imagen de baja frecuencia la que almacena la mayor parte de la energía y sobre la que se calcula el siguiente nivel de la transformada.

Esta imagen porta la información básica de la imagen, es una forma muy eficiente de extracción de información de la imagen.

La transformada wavelet puede calcularse sobre cada uno de los canales de color o sobre la imagen a escala de grises.

La ventaja de la primera de las técnicas es que se proporciona más información sobre la característica más importante del análisis, el color.

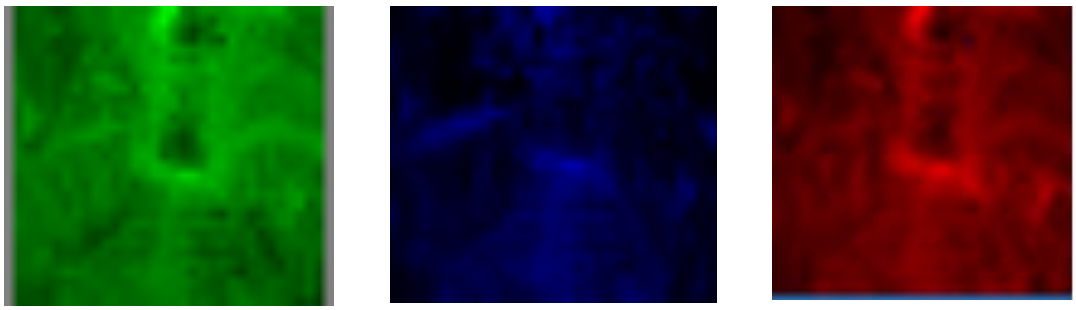


Ilustración 40. Transformada wavelet para los canales G, B, R

Sin embargo, aunque se trate de la información más básica de la imagen, el ruido no se elimina del todo, pudiendo aparecer valores que distorsionen el resultado.

Como se ha visto en la extracción de color y forma, la comparación de las imágenes dependiente de la localización de los valores presenta malos resultados para escalado y rotación de la imagen así como para la traslación de elementos.

Por tanto, la imagen de baja frecuencia, pese a ser la imagen que almacena la mayor parte de la información, no es suficiente para determinar la textura de una imagen.

Son necesarios tanto la aproximación como el detalle para determinar la configuración espacial de la intensidad de los pixeles.

Los momentos estadísticos permiten representar el contenido de manera que los valores calculados sean representativos de la información global de la imagen. El cálculo de los momentos estadísticos (media y desviación típica) desliga las características representadas de su posición, constituyen una herramienta válida para casos de transformación de la imagen [6].

Para cada nivel de la transformada, se calculan los dos primeros momentos, media y desviación típica de cada una de las cuatro imágenes (LL, HL, LH, HH) presentes en la transformada.

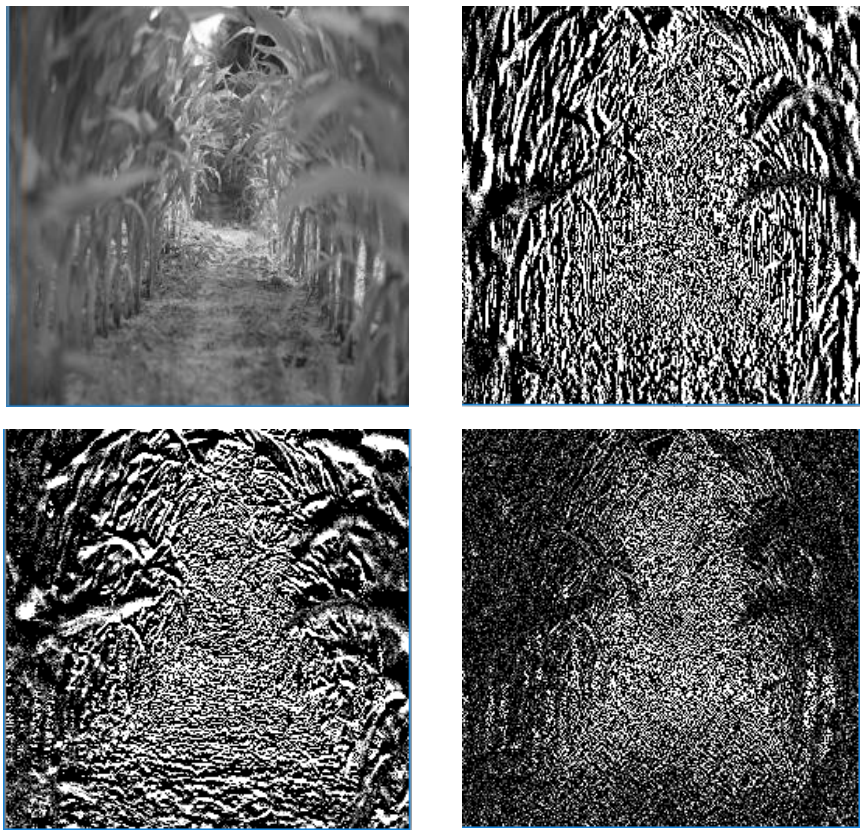


Ilustración 41. Imágenes de la transformada wavelet: LL, HL, LH, HH

De nuevo se consiguen no sólo mejores resultados sino además disminuir el número de componentes que representarán la textura en el vector.

$$n_{componentes} = 2 \times n_{niveles} \times 4 \quad (12)$$

Se añaden 32 componentes al vector que representan la textura de la imagen.

Ilustración 42. Imagen más similar a la imagen maíz (Teniendo sólo en cuenta la transformada Wavelet)

2.2.3.2. Matriz de co-ocurrencia (GLCM)

La textura es un atributo que presenta la distribución espacial de los niveles de gris de una determinada región.

La matriz de co-ocurrencia a escala de grises (Gray Level Co-ocurrence Matrix), también conocida como la matriz de dependencia espacial a escala de grises, caracteriza la textura de una imagen calculando la frecuencia de aparición de un par de pixeles de valores específicos y con una determinada relación espacial.

Esta matriz analiza la dependencia espacial de la textura de la imagen a escala de grises [27].

Consideramos una matriz *Imagen* cuyos elementos representan un valor de intensidad de gris y una matriz C que será la matriz de co-ocurrencia resultante de la matriz *Imagen*.

Se determina la relación espacial que quiere buscarse y representarse en la matriz para esta imagen, se expresa mediante dos valores [a, b].

Para una distancia D entre pixeles, se determina uno de los 4 posibles ángulos como: 135º [-D,-D], 90º [-D, 0], 45º [-D, D] y 0º [0, D].

Cada uno de los componentes de la matriz C representa el número de veces que se ha repetido esa relación espacial entre los pixeles de la matriz I con valores de intensidad igual al número de columnas y al número de filas respectivamente.

Es por ello que la matriz de co-ocurrencia debe tener tantas filas y columnas como posibles valores o intensidades de gris vayan a analizarse.

En el sistema propuesto se eligen 256 posibles intensidades de gris por lo que la matriz de co-ocurrencia presenta 256x256 valores.

La relación espacial escogida es [-1, 0], es decir, una distancia 1 en un ángulo de 90º ya que el análisis en esta dirección ha demostrado tener mejores resultados en la búsqueda de imágenes [6]. Una relación espacial con distancia D demasiado grande tiene como consecuencia una matriz GLCM poco representativa de la información de textura de la imagen.

Cada componente de la matriz C representa el número de veces que se repite la relación de distancia en un determinado ángulo, para un par de pixeles concretos en la matriz Imagen.

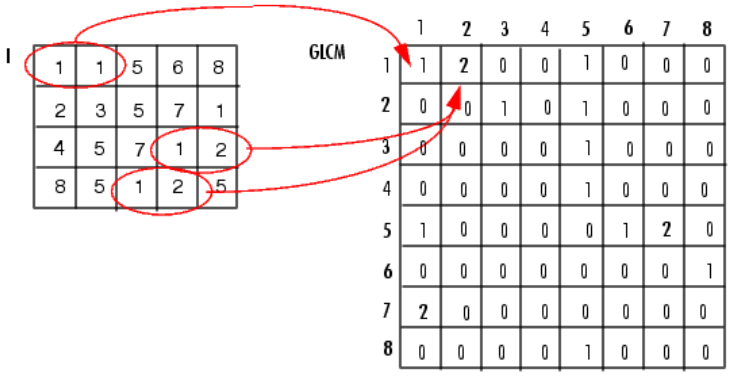


Ilustración 43. Construcción de la matriz GLCM

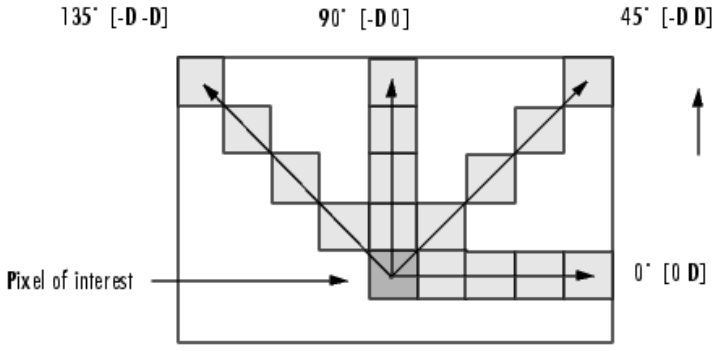


Ilustración 44. Relaciones espaciales entre pixeles para el cálculo de la matriz GLCM

La matriz GLCM puede ser calculada para la imagen completa o para distintas regiones de la imagen. Calcularlo para distintas regiones de la imagen liga la información a una localización en la imagen. Esta característica puede ser o no buscada, aunque la herramienta se vuelva poco robusta frente a transformaciones, dos imágenes son más parecidas si presentan la misma textura en la misma región.

Por otro lado, el cálculo por regiones supone multiplicar la información tantas veces como regiones creadas, la matriz GLCM tiene 256x256 componentes independientemente de la región. Los cálculos de todas las matrices y la longitud del vector (en el que se debe almacenar la información de todas las regiones) aumentan el tiempo de procesado del programa.

Para reducir el tamaño de las matrices GLCM, puede aplicarse una transformada wavelet mediante la cual se obtendrá su información más básica (bajas frecuencias).

Comparar la información de GLCM componente a componente, es decir, introducir las componentes de la matriz directamente en el vector, supone que sólo imágenes cuyas relaciones entre pixeles determinadas tengan una frecuencia de aparición comparable, sean consideradas similares.

Esta suposición implica que dos imágenes, aunque presenten los mismos elementos, si su brillo es distinto, no serán consideradas similares.

Por ello, se recurre a modelos estadísticos que consiguen representar la información de manera que cada componente sea significativo y tenga en cuenta la información global.

El cálculo de la desviación típica de las columnas de la matriz GLCM, proporciona la información de variabilidad respecto a esa relación espacial dada para cada nivel de gris que representa la matriz.

Estos datos son comparables y evitan el problema del brillo. Además, el número de componentes que se añaden al vector es 256 veces menor que en el caso del almacenamiento de la matriz completa.

Ilustración 45. Imagen más similar a la imagen "Maiz.jpg" (Teniendo sólo en cuenta la matriz GLCM)

2.2.3.3. La varianza

La varianza resulta de particular importancia para la descripción de las texturas. Este segundo momento estadístico representa las desviaciones de intensidad con respecto a la media para una región dada. Por consiguiente, mide el contraste de la imagen en esa región.

Una región homogénea de la imagen presentará varianza nula, sin embargo, una región con alta variabilidad y grandes contrastes será asociada a un valor de varianza alto.

La varianza se calcula como:

$$var(X) = \sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n} \quad (13)$$

La información de la varianza se calcula para imágenes completas o para regiones de la imagen.

El cálculo de la varianza por regiones tiene sentido en la comparación de imágenes. Esta información permite distinguir como se distribuyen las texturas en la imagen. Sin embargo, la información se encuentra ligada a la posición en la imagen y si se utiliza sin procesado previo para comparar imágenes, cualquier imagen cuyos elementos similares estén desplazados respecto a la imagen query no será identificada como similar.

Ilustración 46. Representación de los valores de varianza calculada en regiones de 40x40

Para independizar la varianza calculada de su posición en la imagen, se utiliza un algoritmo muy similar al del cálculo de los descriptores EDH.

La imagen se divide primero en 16 sub-imágenes de 100x100 pixeles. Estas sub-imágenes se dividen a su vez en bloques de 4x4 pixeles. Se calcula la varianza de cada uno de estos bloques y el valor obtenido se suma al histograma local en el componente que representa la sub-imagen en cuestion.

Así el histograma local está compuesto por 16 componentes, cada una de las cuales lleva información de la media de los valores de varianza de la sub-imagen correspondiente a la componente en cuestión del histograma.

El histograma global presenta la media de los valores de varianza obtenidos para la imagen completa.

Por último, el histograma semiglobal contiene la información de la media de los valores de varianza obtenidos para diferentes grupos de sub-imágenes. Se cuentan 13 componentes, correspondientes a 13 grupos de sub-imágenes para este vector.

Los valores calculados de varianza representan regiones mucho más pequeñas sin que esto incluya ruido, como sucedería si se realizase simplemente un cálculo de varianza por regiones.

Estos valores no se utilizan directamente, sino que pasan a formar parte de un histograma que elimina el efecto el posible ruido ya que cada una de sus componentes tiene en cuenta el valor de muchas regiones que pertenecientes a otra de tamaño mayor.

Empleando este algoritmo, la varianza queda representada en el vector por 30 componentes.

Ilustración 47. Representación de los valores de varianza calculada en regiones de 4x4

2.3. Comparación de vectores

Para poder realizar el sistema, es necesario llevar a cabo el mismo procesado sobre las imágenes de la base de datos y sobre la imagen "*query*" en cuestión.

En primer lugar, se desarrolla un programa cuya función consiste en calcular un vector de características para cada una de las imágenes de la base de datos de la cual serán seleccionadas las imágenes consideradas parecidas a la dada. (Anexo 1)

Este programa recibe como entrada el nombre del fichero en el que se van a guardar los vectores y un fichero que contiene los nombres y la ubicación de todas las imágenes que van a formar parte de la base de datos.

El programa lee línea a línea los nombres de las imágenes que debe buscar en el disco. De cada una de estas imágenes calcula su vector de características asociado y guarda la información en el fichero de salida especificado. Cada línea de este fichero representa un vector de características correspondiente a una de las imágenes de la base de datos.

El segundo programa, recibe como entrada el nombre completo de la imagen que va a buscarse (*path* completo) y el nombre y ruta del fichero donde se almacenan los vectores de características de las imágenes de la base de datos.

El programa procesa la imagen introducida y extrae el vector de características asociado de la misma forma que lo hacía el primer programa sobre todas las imágenes de la base de datos.

A continuación, línea a línea lee el fichero de características correspondiente a las imágenes de la base de datos.

Las imágenes de la base de datos están nombradas con un número ascendente desde 1 hasta 100.000, por ello, el programa puede asociar el número de línea del fichero con el vector de características a la imagen que está procesando.

El vector de características calculado para la imagen "*query*" se compara con cada uno de los vectores presentes en el fichero.

La comparación es una distancia euclídea componente a componente:

$$distancia = \sum_{i=0}^{n} \left| vector_{query}[i] - vector_{database}[i] \right| \tag{14}$$

Para poder asignar pesos a cada una de las características, y de esta forma elegir aquellas que tienen más relevancia para determinar el parecido entre imágenes, se calcula la distancia para cada una de las características y se suman estas distancias multiplicadas por sus pesos correspondientes:

$$distancia_{color} = \sum_{i=0}^{127} \left| vector_{query}[i] - vector_{database}[i] \right| \tag{15}$$

$$distancia_{forma} = \sum_{i=128}^{277} \left| vector_{query}[i] - vector_{database}[i] \right| \tag{16}$$

$$distancia_{wavelet} = \sum_{i=278}^{309} \left| vector_{query}[i] - vector_{database}[i] \right| \tag{17}$$

$$distancia_{varianza} = \sum_{i=310}^{339} \left| vector_{query}[i] - vector_{database}[i] \right| \tag{18}$$

$$distancia_{glcm} = \sum_{i=340}^{595} \left| vector_{query}[i] - vector_{database}[i] \right| \quad (19)$$

$$distancia_{total} = w_{color} \times distancia_{color} + w_{forma} \times distancia_{forma} + w_{wavelet} \times distancia_{wavelet} + w_{varianza} \times distancia_{varianza} + w\_glcm \times distancia_{glcm} \quad (20)$$

En un primer lugar estas distancias eran calculadas para todas las imágenes por el hilo principal del programa. Sin embargo, tras la paralelización del algoritmo, se introducen tantos ficheros de vectores de características como hilos paralelos vayan a crearse. Cada uno de estos ficheros de características lleva información para un conjunto de imágenes de la base de datos. Los hilos calculan las distancias de cada una de las imágenes de sus ficheros y las incluyen en un vector de distancias global. Es necesario tener en cuenta que ha de incluirse un "mutex" (mecanismo de exclusión mutua) para el acceso de estos hilos al vector global de distancias ya que si no, podrían darse problemas típicos de la paralelización.

En la comparación de vectores, es necesario que los valores de cada característica estén normalizados. De lo contrario, unas características tendrán más importancia que otras sin que esto haya sido planeado.

Se calcula la distancia como:

$$distancia_{característica} = \sum_{i=m}^{N-m} \frac{\left| vector_{query}[i] - vector_{database}[i] \right|}{1 + vector_{query}[i] + vector_{database}[i]} \times \frac{1}{N} \quad (21)$$

Donde m y n dependen de la característica que se esté analizando.

Esto asegura que la distancia calculada para cada característica se encuentra dentro del rango [0-1].

Sin embargo, esto no asegura que los valores de estas distancias vayan a ser comparables. Los valores máximos y mínimos de las distancias de las características calculadas para todas las imágenes de la base de datos con respecto a la imagen query, siguen siendo muy dispares.

Dividir cada distancia de cada característica calculada para cada imagen de la base de datos, por la media de la distribución de estas distancias, permite normalizar los valores y que estos sean comparables entre sí.

$$distancia_{total} = w_{color} \times \frac{distancia_{color}}{distancia_{color}} + w_{forma} \times \frac{distancia_{forma}}{distancia_{forma}} + w_{wavelet} \times \frac{distancia_{wavelet}}{distancia_{wavelet}} + w_{varianza} \times \frac{distancia_{varianza}}{distancia_{varianza}} + w\_glcm \times \frac{distancia_{glcm}}{distancia_{glcm}} \quad (22)$$

En la ecuación (22), aparecen los pesos que permiten dar más importancia a unas características sobre otras.

Para elegir el número de imágenes que van a mostrarse, pueden seguirse diversas estrategias. La elección de un umbral permite establecer a partir de que distancia las imágenes dejan de considerarse parecidas.

El problema de esta estrategia es la heterogeneidad de la base de datos. Si tanto las imágenes de la base de datos como las imágenes que van a ser introducidas son desconocidas, el umbral depende de la imagen de entrada. Es decir, un mismo umbral será demasiado bajo para una imagen de entrada y demasiado alto para otra.

Una estrategia, más sencilla y evidente, es representar un número de imágenes dado. Para ello se ordenan las imágenes de menor a mayor distancia y se seleccionan las imágenes desde cero hasta el número marcado.

3. Resultados

Finalmente, el vector de características está formado por 596 componentes:

- 128 componentes de color correspondientes al histograma de color HSV.
- 150 componentes de forma, procedentes de los 3 histogramas de EHD (80 componentes de histograma local, 5 de histograma global y 13 de histograma semiglobal).
- 32 componentes de la transformada wavelet para representar la textura, como resultado del cálculo de los dos primeros momentos para las cuatro imágenes de cada uno de los cuatro primeros niveles de la transformada.
- 30 valores de varianza obtenidos siguiendo el algoritmo de cálculo de EHD.
- 256 valores de GLCM que representan la desviación típica de cada una de las columnas de esta matriz calculada para la imagen completa.

La base de datos de la que se extraen las imágenes está formada por 100.000 imágenes.

Se utilizan 100 hilos paralelos cada uno de los cuales compara el vector de características asociado a la imagen introducida con los vectores de 1.000 imágenes de la base de datos. Se cuentan en total 100 ficheros de 1.000 imágenes, uno para cada hilo paralelo.

El tiempo de procesado desde la ejecución del programa hasta que se muestran las imágenes seleccionadas es de 40 segundos en un solo ordenador.

Los mejores resultados se obtuvieron para los pesos: 0.4 en color, 0.1 para GLCM, 0.15 para la transformada Wavelet, 0.25 para EDH y 0.1 para varianza.

En la tabla 3 se muestran los resultados obtenidos para diferentes imágenes.

	Precisión
Camión	80%
Maíz	100%
Barca	70%

Tabla 3. Precisión del sistema para 3 imágenes query de entrada

La precisión *P* se calcula como el número de imágenes similares obtenidas entre el número de imágenes totales devueltas por el sistema (multiplicado por 100 para obtenerlo en porcentaje). La precisión se calcula para determinar el grado de exactitud del sistema. [6]

$$P = \frac{numeroDeImagenesSimilaresObtenidas}{numeroDeImagenesObtenidas} \quad (23)$$

Ilustración 48. Maiz.jpg

Ilustración 49. Camion.jpg

Ilustración 50. Barca.jpg

Ilustración 51. Resultados del sistema para la imagen Maiz.jpg

Ilustración 52. Resultados del sistema para la imagen Camion.jpg

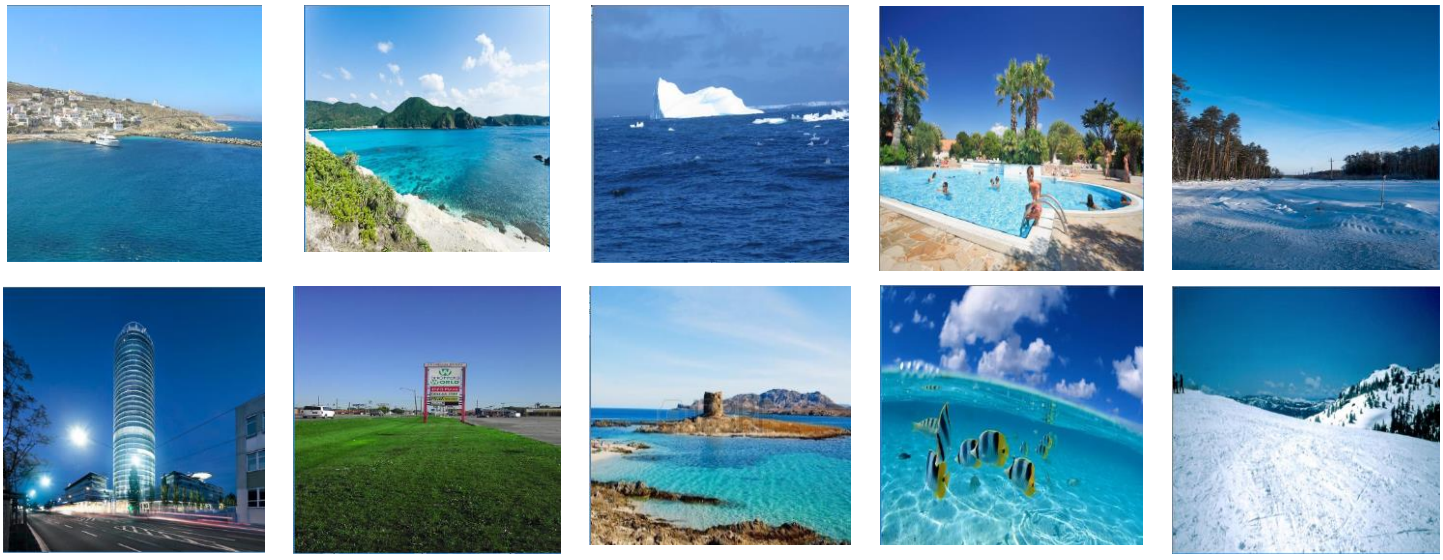


Ilustración 53. Resultados del sistema para la imagen Barca.jpg

4. Conclusiones

En base a los resultados obtenidos para las diferentes estrategias escogidas, podemos concluir aplicar modelos estadísticos a la información extraída de las imágenes, permite sistemas robustos ante posibles transformaciones.

A lo largo del desarrollo del sistema se ha visto como la comparación de la información extraída y comparada sin procesado previo sólo funciona para imágenes muy similares que no se ven afectadas por el ruido y presentan los elementos en la misma ubicación.

Los histogramas son herramientas muy útiles para comprimir la información y además permiten describir la imagen de forma global desligando la información de la característica extraída de la posicion concreta en la imagen.

Los descriptores como EHD utilizan los histogramas para representar la información, calculando estos en diferentes secciones de la imagen para añadir la información de la localización de la característica en la imagen.

Los momentos estadísticos como la desviación típica tambien son útiles técnicas para resumir la información y crear modelos que permiten comparar la información de diferentes fuentes. Se ha comprobado como utilizar el primer y el segundo momento para resumir la información de la transformada wavelet y de la matriz GLCM proporcionaba mejores resultados que la información cruda de ambas.

Por otro lado, el espacio de color HSV al estar basado en el sistema de percepción visual humano, es mas adecuado para sistemas que pretenden la búsqueda de imágenes similares. Además permite resumir su información en histogramas uniformes obteniéndose buenos resultados.

Cuando se esta desarrollando un sistema de procesado de imagen, es necesario tener en cuenta el tamaño de la información extraída así como el de la base de datos que va a analizarse. De ambos dependerá el tiempo de procesado del sistema que permitirá que ese sea o no viable.

La paralelización del algoritmo permite disminuir los tiempos de procesado pero sobre un mismo procesador y sin desplegar el sistema, sólo permite mejoras del 20%.

Por último, para comparar información, es necesario la normalización de las variables que van a compararse o en su defecto los resultados calculados no serán consistentes con la información.

5. Líneas futuras

El sistema presentado puede mejorarse de diversas formas.

Actualmente el algoritmo compara la imagen de entrada con un total de 100.000 imágenes de la base de datos. Cuantas más imágenes sean comparadas con la imagen de entrada, mayor será la probabilidad de encontrar una imagen parecida.

Ampliar la base de datos, supone aumentar el tiempo de procesado. Para reducir este tiempo de procesado, puede desplegarse el programa (ya paralelizado) en un conjunto de ordenadores en lugar de un solo ordenador con un solo procesador.

La base de datos utilizada en el sistema no presenta categorías diferenciadas y etiquetadas. Esto tiene como consecuencia que no se conozca con exactitud la precisión del sistema. Ajustar el sistema a una base de datos con diferentes categorías permitiría una mejor evaluación de los resultados.

Los algoritmos que utilizan puntos de interés como FAST, permiten diferenciar formas y situar objetos en la imagen. Sería interesante añadir uno de estos algoritmos al sistema y ver cómo afecta a los resultados.

La aplicación presentada informa al usuario del parecido de la imagen mediante un número de estrellas. Si los usuarios pudiesen opinar del parecido de color, forma o textura de la imagen puntuando mediante estrellas, se podrían ajustar los pesos de las características en función del feed-back proporcionado por estos usuarios.

3. Bibliografía

[1] Chang S F, Chen W, Sundaram H. Semantic visual templates: linking visual features to semantics [A]. Proceedings of 1998 International Conference on Image Processing [C]. Chicago, Illinois, 1998.4-7.

[2] Zhao R, Grosky W I. Narrowing the semantic gap-improved text-based web document retrieval using visual features [J]. IEEE Transaction on Multimedia, 2002, 4(2):189-200.

[3] Agarwal, S.; Verma, A.K.; Dixit, N., "Content Based Image Retrieval using Color Edge Detection and Discrete Wavelet Transform," in Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference on , vol., no., pp.368-372, 7-8 Feb. 2014.

[4] Katare, A.; Mitra, S.K.; Banerjee, Asim, "Content Based Image Retrieval System for Multi Object Images Using Combined Features," in Computing: Theory and Applications, 2007. ICCTA '07. International Conference on , vol., no., pp.595-599, 5-7 March 2007.

[5] Jianlin Zhang; Wensheng Zou, "Content-Based Image Retrieval using color and edge direction features," in Advanced Computer Control (ICACC), 2010 2nd International Conference on , vol.5, no., pp.459-462, 27-29 March 2010

[6] A. Nazir, R. Ashraf, T. Hamdani and N. Ali, "Content based image retrieval system by using HSV color histogram, discrete wavelet transform and edge histogram descriptor," 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), Sukkur, 2018, pp. 1-6.

[7] N. Anagnostopoulos, C. Iakovidou, S. A. Chatzichristofis, Y. Boutalis and M. Lux, "Localized global descriptors for image retrieval: An extensive evaluation on adaptations to the SIMPLE model," *2016 IEEE International Conference on Imaging Systems and Techniques (IST)*, Chania, 2016, pp. 312-317.

[8] J. Yao, Y. Deng, Y. Yu and C. Sun, "A fast image retrieval method with convolutional neural networks," 2017 36th Chinese Control Conference (CCC), Dalian, 2017, pp. 11110-11115.

[9] S. D. Ruikar and R. S. Kabade, "Content based image retrieval by combining feature vector," 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, 2016, pp. 1517-1523.

[10] Salton G, Wong A, Yang CS. A vector space model for automatic indexing [J]. Communications of the ACM, 1975, 18(11):613-620.

[11] M. Singha and K. Hemachandran, Performance analysis of color spaces in image retrieval, Assam Univ. J., vol. 7, no. 2, 2011.

[12] X. Wan and L. Angeles, Color Distribution Analysis and Quantization for Image Retrieval, vol. 2670, no. 213, pp. 8-16.

[13] S. Sural, Gang Qian, and S. Pramanik, Segmentation and histogram generation using the HSV color space for image retrieval, Proceedings. Int. Conf. Image Process., vol. 2, p. II-589-II-592, 2002.

[14] Guan-Lin Shen and Xiao-Jun Wu, "Content Based Image Retrieval by combining color, texture and CENTRIST," 2013 Constantinides International Workshop on Signal Processing (CIWSP 2013), London, 2013, pp. 1-4.

[15] Benjamin B. Kimia, "Symmetry-Based Shape Representations," Laboratory for Engineering Man/Machine Systems (LEMS), IBM, Watson Research Center, October 1999.

[16] H. Farsi and S. Mohamadzadeh, Colour and texture feature-based image retrieval by using hadamard matrix in discrete wavelet transform, IET Image Process., vol. 7, no. 3, pp. 212-218, 2013.

[17] D. Zhang and G. Lu, Evaluation of similarity measurement for image retrieval, Proc. 2003 Int. Conf. Neural Networks Signal Process. ICNNSP03, vol. 2, pp. 928-931, 2003

[18] D. Xu and Z. Qu, "An Image Classification Method Based on Matching Similarity and TF-IDF Value of Region," 2013 Sixth International Symposium on Computational Intelligence and Design, Hangzhou, 2013, pp. 112-115

[19] Tan S. Neighbor-weighted k-nearest neighbor for unbalanced text corpus [J]. Expert Systems with Applications, 2005, 28(4):667-671.

[20] C. Iakovidou, N. Anagnostopoulos, A. C. Kapoutsis, Y. S. Boutalis, and S. A. Chatzichristofis, "Searching images with MPEG-7 (& mpeg7-like) powered localized descriptors: The SIMPLE ans[wer to effective content based image retrieval," in 12th Intl. Workshop on Content-Based Multimedia Indexing CBMI, Klagenfurt, AT, June 2014, pp. 1–6.

[21] H. Bay, T. Tuytelaars, and L. J. V. Gool, "Surf: Speeded up robust features," in ECCV (1), 2006, pp. 404–417

[22] Colour spaces - perceptual, historical and applicational background Marko TkalEiE, Jurij F. TasiE Faculty of electrical engineering University of Ljubljana TriaSka 25, 1001 Ljubljana, Slovenia email: marko.tkalcic@fe.uni-lj.si

[23] C.-H. Su, H.-S. Chiu, and T-M. Hsieh, An efficient image retrieval based on HSV color space, 2011 Int. Conf. Electr. Control Eng., pp. 5746-5749, 2011.

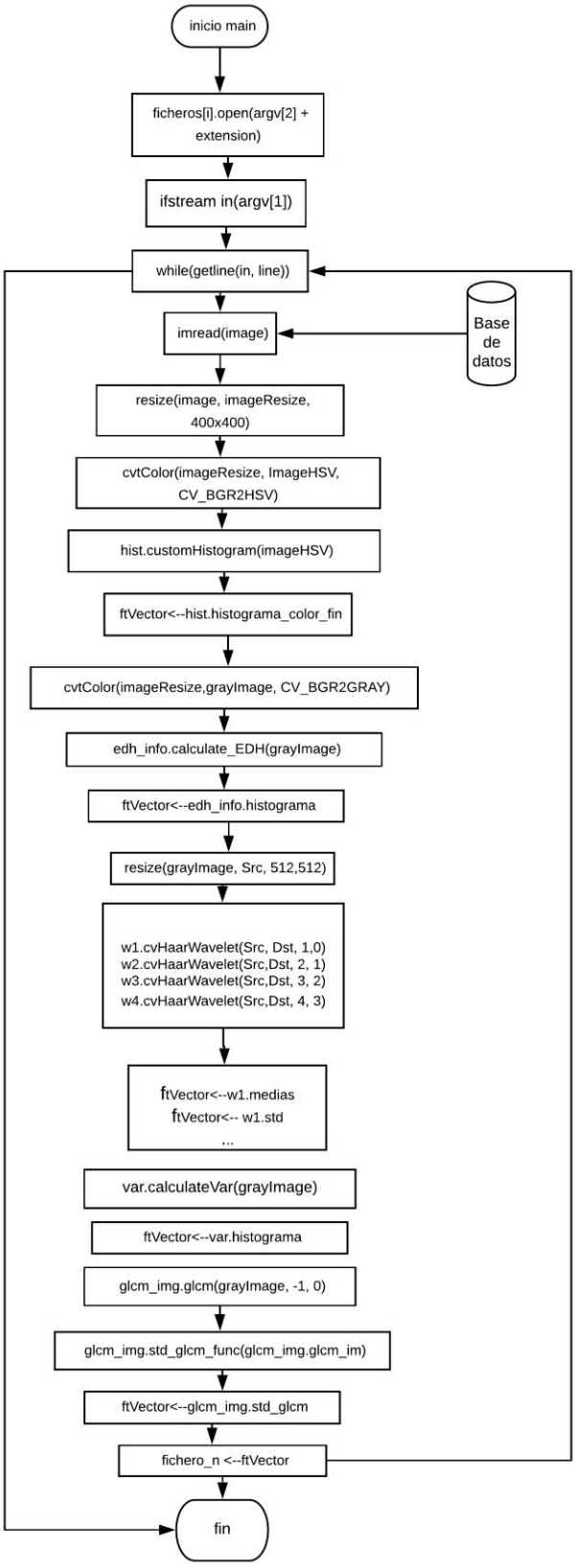
[24] Ljubovic, Vedran & Supic, H. (2013). "Comparative study of color histograms as global feature for Image Retrieval". 1059-1063.

[25] M. Ali and D. Clausi, "Using the Canny edge detector for feature extraction and enhancement of remote sensing images," IGARSS 2001. Scanning the Present and Resolving the Future. Proceedings. IEEE 2001 International Geoscience and Remote Sensing Symposium (Cat. No.01CH37217), Sydney, NSW, 2001, pp. 2298-2300 vol.5.

[26] Sun Won Won, Chee & Kwon Park Park, Dong & Park Park, Soo-Jun. (2002). "Efficient Use of MPEG7 Edge Histogram Descriptor". Etri Journal - ETRI J. 24. 23-30. 10.4218/etrij.02.0102.0103.

[27] R. Jain, R. Kasturi, and B. G. Schunck, Machine Vision, McGraw Hill International Editions, 1995

ANEXO 1. Código de extracción de características y almacenamiento en la base de datos



El código del hilo principal del programa (main), comienza con la creación de un conjunto de ficheros de salida. Se crean tantos ficheros de salida como hilos paralelos vayan a utilizarse en el algoritmo de búsqueda de imagen.

En cada fichero de salida va a almacenarse la información de los vectores de características de un grupo de imágenes de la base de datos.

El nombre del fichero de salida es dl introducido como argumento más una extensión compuesta por el número de hilo y la extensión de ficheros de texto ".txt".

El fichero de entrada cuyo nombre y ruta se introduce como argumento, contiene el nombre de todas las imágenes de la base de datos. Este fichero se lee línea a línea para poder cargar todas las imágenes de la base de datos (utilizando la función "imread" de Opencv) y almacenar sus características en los ficheros.

Las imágenes de la base de datos se cargan en formato RGB y a continuación se escalan a 400x400 pixeles para normalizar su tamaño y poder aplicar sobre todas ellas el mismo procesado.

Para poder extraer la información de color como histograma HSV, se convierte el espacio de color RGB de la imagen cargada al espacio de color HSV utilizando la función "cvtcolor" proporcionada por Opencv.

"Hist" es un objeto de la clase "Histograma" definida en el algoritmo. Esta clase presenta un método "customHistogram" que recibe como parámetro una imagen y calcula su histograma de color en formato 8:2:2.

Los valores calculados por "customHistogram" se almacenan en el atributo "histograma_color_fin" del objeto "hist". Estos valores se añaden al vector de características de la imagen en cuestión.

Se crea a continuación una nueva imagen, "grayImage", como resultado de la conversión de la imagen cargada a escala de grises. Para ello se utiliza de nuevo la función de Opencv "cvtcolor".

Esta conversión es necesaria para poder calcular los descriptores EDH que, como se explica en el apartado 2.2.2, sólo tienen en cuenta la intensidad del pixel.

"edh_info" es un objeto de la clase "EDH" cuyo método "calculate_edh" calcula los histogramas locales, global y semiglobal propios de EDH, de la imagen introducida como parámetro. El resultado se guarda en el atributo "histograma" del objeto y este se almacena en el vector de características de la imagen para representar su información de forma.

Como se dijo en el apartado 2.2.3.1 las dimensiones de la imagen sobre la que se calcula la transformada Wavelet han de ser potencia de dos, es por ello por lo que la imagen se escala de nuevo a 512x512 pixeles utilizando la función "resize" de Opencv.

La imagen escalada es la imagen a escala de grises puesto que la transformada wavelet utilizada en el algoritmo no tiene en cuenta el color de la imagen. El resultado se guarda en la imagen "Src".

Los objetos w1, w2, w3 y w4 pertenecen a la clase Wavelet y en cada uno de ellos se almacena la información de uno de los cuatro niveles de la transformada que se calculan.

El método "cvtHaarWavelet" de la clase "Wavelet" calcula la transformada wavelet de la imagen introducida como primer parámetro y almacena el resultado en la imagen obtenida como segundo parámetro. El tercer parámetro indica el número de niveles de la transformada que van a calcularse y el último, el nivel de transformada del que se parte.

Las medias y desviaciones típicas de las cuatro imágenes de la transformada (LL, LH, HL y HH) de cada nivel calculado, se almacenan en los atributos "medias" y "std" que son vectores de cuatro componentes.

Las componentes de estos vectores se almacenan en el vector de características.

El método "calculateVar" del objeto "var" perteneciente a la clase "Varianza" definida en el algoritmo, calcula el histograma de varianza descrito en 2.2.3.3 que se almacena en el vector de características.

También sobre la imagen a escala de grises, de tamaño 400x400, se calcula la matriz GLCM. El objeto "glcm_img" pertenece a la clase "GLCM" definida en el algoritmo.

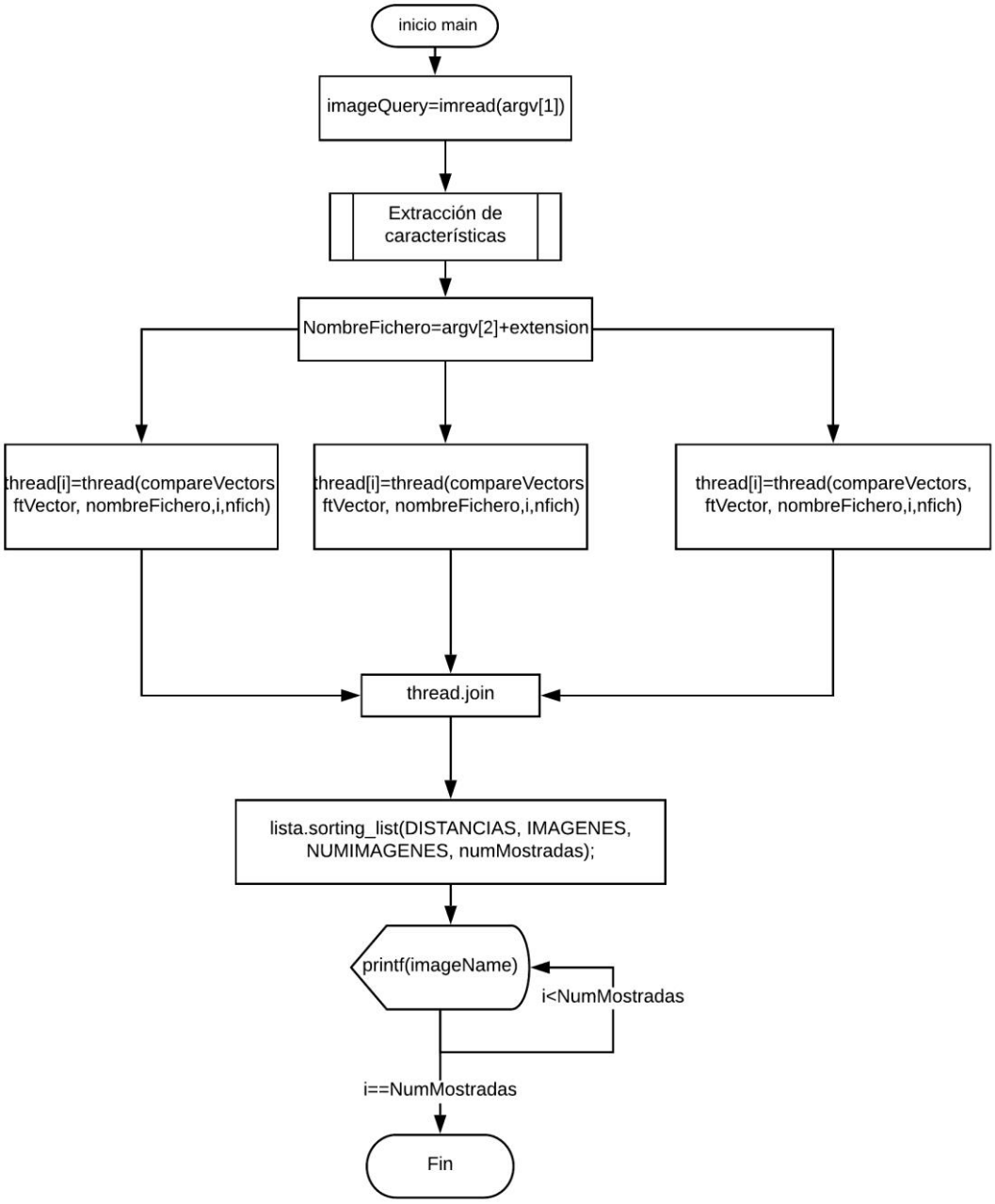
El método "glcm" de esta clase, calcula la matriz GLCM correspondiente a la imagen introducida como parámetro. El segundo y el tercer parámetro de esta función indican el ángulo y distancia de la relación espacial entre pixeles que va a buscarse (tal y como se explica en 2.2.3.2).

El método "std_glcm_func" de la clase GLCM calcula la desviación típica de las columnas de la matriz GLCM obtenida.

El resultado de esta función se almacena en el vector de características.

El vector completo se almacena en una línea del fichero correspondiente.

ANEXO 2. Código búsqueda de imágenes similares



El algoritmo de búsqueda de imágenes similares parte de la carga de la imagen "query" mediante la función "imread" proporcionada por Opencv. El nombre de esta imagen se introduce como argumento.

Para el cálculo del vector de características de la imagen *query*, se recurre al mismo procesado descrito en el Anexo 1.

Una vez calculado el vector de características de la imagen, el algoritmo de búsqueda crea hilos paralelos para agilizar el procesado del programa.

Cada hilo paralelo recibe como argumentos la función que va a ejecutar y los argumentos de esta.

La función "compareVectors" ejecutada por todos y cada uno de los hilos del programa, recibe como parámetros el vector de características de la imagen *query*, el nombre del fichero con los vectores de características asociado al hilo en cuestión, el número de hilo y el número total de hilos creados.

Esta función lee línea a línea el fichero especificado como segundo parámetro.

En cada una de las líneas de este fichero, se almacena uno de los vectores de características asociado a una de las imágenes de la base de datos.

Se calculan las distancias de color, forma, Wavelet, GLCM y varianza del vector asociado a la imagen *query* con todos los vectores almacenados en el fichero de entrada, siguiendo el proceso descrito en 2.3.

Estas distancias se almacenan en vectores diferentes cuyo tamaño se calcula como el número de imágenes de la base de datos (variable global) dividido entre el número total de ficheros introducido como último parámetro de la función. Así se consiguen vectores de tamaño igual al número de imágenes procesadas por cada hilo.

Para calcular la distancia total, cada vector de distancias asociado a una característica concreta se multiplica por su peso correspondiente. La distancia total calculada se almacena en un vector "DISTANCIAS" global protegido con un "mutex" que garantiza el acceso en exclusión mutua de los hilos.

Cuando todos los hilos han terminado, se ordena el vector "DISTANCIAS" de menor a mayor distancia utilizando el método "sorting_list" de la clase "sorted_list".

Las componentes del vector global "IMÁGENES" representan el índice de cada imagen, es por ello que este vector debe ser introducido en el método "sorting_list" para que cada vez que se produzca un cambio en el orden del vector "DISTANCIAS" este se replique en el vector "IMÁGENES" y no se pierda la asociación imagen-distancia.

Como las imágenes de la base de datos están nombradas con un índice ascendente, sólo con el índice almacenado en el vector "IMÁGENES" puede cargarse la imagen deseada.

El nombre completo de la imagen se imprime por pantalla para que pueda ser enviado al cliente desde el servidor de la aplicación representada en el Anexo 3.

ANEXO 3. Esquema de la aplicación cliente-servidor