Sistemas Informáticos

Curso 2008 - 2009

CLASIFICACIÓN DE TEXTURAS NATURALES MEDIANTE TÉCNICAS DE VISIÓN POR COMPUTADOR

Realizado por:

Daniel de Santos Sierra

Dirigido por:

Prof. Gonzalo Pajares Martinsanz

Dpto. Ingeniería del Software e Inteligencia Artificial

AUTORIZACIÓN

Autorizo a la facultad de Informática de la Universidad Complutense de Madrid, así como al resto de sus centros adscritos a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Firmado: .....

Daniel de Santos Sierra

AGRADECIMIENTOS

Una de las grandes ventajas de hacer el proyecto solo es que puedo escribir lo que quiera en los agradecimientos y para empezar, voy a decir lo que se suele decir en todos los agradecimientos, "espero no dejarme a nadie", pero si lo hago, espero que el rencor sea pasajero.

Le agradezco a Gonzalo Pajares, mi director de proyecto, que me lo haya aceptado, a pesar de todos los inconvenientes que surgieron. Gracias a su bondad he podido finalizar la carrera este año que me posibilita a continuar con mis estudios.

Quiero acordarme de aquellos que no están, mis abuelos, Tomás y Modesta, porque aunque no hayan podido ver a lo que he llegado, sé que estarían orgullosos.

Les doy las gracias a mis abuelos, Isidoro y Conce, ellos me han enseñado esas cosas que no están en los libros y que son tan importantes para la vida. Les doy las gracias por haber luchado para que pueda llegar hasta aquí, porque más que abuelos, para mi han sido mis segundos padres.

A mi padre, Juan, por todas las alegrías que le ha supuesto cada uno de mis progresos y que aportaban más sentido a mi esfuerzo. Por todos los consejos que he recibido y no siempre he aprovechado.

A mi madre, Conchi, porque sin ella la carrera, más que difícil habría sido imposible. Por todo su apoyo y toda la fuerza que me ha dado, por todos los cuidados que he recibido y porque nunca ha pedido nada a cambio y se ha conformado únicamente con verme feliz.

No me podría olvidar de mi hermano, Alberto, recordaré las conversaciones sobre semáforos, procesos zombies, etc, a la hora de la cena y que desquician a mi madre. Siempre ha sido mi modelo a seguir y aunque no consiga alcanzarle, ya veo el miedo en sus ojos. Me ha demostrado que sólo con esfuerzo se puede llegar a conseguir las cosas.

A Carmen, porque ha logrado que todo sea más fácil, porque ha sido un apoyo constante y ha conseguido que no me desespere en cada paso que he dado. Porque me ha enseñado que en la vida estudiar es importante, pero encontrar personas con las que compartirla lo es más. Por las primeras palabras que me dijo que siempre recordaré y siempre sonreiré al recordarlas.

No me puedo olvidar de todos los que me han acompañado durante la carrera Laura, Julián, José Ramón, Naxo, Héctor, Mario, Fran, Ramón, Cristinita y muchos más, y a mis chicas de señas, Mariona y Alba, porque ellos han hecho que cada día sea distinto, por todas las risas, los buenos momentos, el apoyo, en definitiva, por cinco años inolvidables, que espero que no terminen aquí. A mi psicóloga particular, Vane, por todos los post it que he escrito con sus consejos y por haberse convertido en una amiga en tan poco tiempo. A mi amigo, Javi Lázaro, por todas las bromas y los grandes momentos que en muchas ocasiones recuerdo con una grata sonrisa. A Dani, porque aunque diga que somos compañeros, en muchos momentos se ha comportado como un gran amigo.

Agradecerles también a mis tíos, Isidoro y Montse y claro está, Mª Tere y Fernando, por todo el interés que han mostrado.

Y a todos los que me han apoyado con sus palabras y se han alegrado con mis progresos, muchas gracias.

"Podemos hacer una estimación de nuestra propia imagen
observando a quienes nos rodean"

Andrew Matthews

A Carmen, mi familia y amigos,
gracias por el apoyo y todos los momentos.

1. INTRODUCCIÓN

1.1.RESUMEN DEL PROYECTO

1.1.1.RESUMEN

Esta memoria explica la creación, desarrollo y utilización de la aplicación de nuestro proyecto de fin de carrera, correspondiente a la asignatura de Sistemas Informáticos.

El proyecto consiste en la implementación de una aplicación para la clasificación de texturas naturales mediante técnicas de visión por computador. Esta aplicación esta dividida en dos partes: interfaz gráfica y algoritmos de clasificación.

Interfaz gráfica: proporciona al usuario una forma fácil y sencilla para el completo uso de la aplicación. Con esta herramienta, el usuario podrá seleccionar los distintos algoritmos implementados e introducir los parámetros convenientes para la ejecución del algoritmo. Así mismo, podrá abrir cualquier imagen y guardar los resultados del entrenamiento en documentos XML. Una vez ejecutado un método, se creará toda la información pertinente y podrá ser visualizada por el usuario si así lo desea.

Algoritmos de clasificación: constituye el núcleo del proyecto. Se han implementado cinco algoritmos: Cuantización Vectorial, Balasko o Fuzzy clustering no supervisado, K-Medias, Lloyd y Bayes.

La aplicación ha sido construida en JAVA haciendo uso de librerías: JAI, JAMA y JDOM, para el manejo de imágenes, matrices y documentos XML, respectivamente.

1.1.2.PALABRAS CLAVE

Clasificación de texturas naturales, segmentación, cuantización vectorial, Fuzzy clustering no supervisado, k-medias, Lloyd, Bayes, clasificador paramétrico bayesiano, crisp, fuzzy, clustering.

1.1.3.ABSTRACT

This report explains the creation, development and utilization of this final thesis project, corresponding to Sistemas Informáticos course.

This current project involves the development of an application for natural texture classification by means of computer vision techniques. This application is divided into two parts: graphical interface and classification algorithms.

Graphical Interface: Provides the user with an easy and simple procedure to the whole use of the application. With this tool, the user may select the implemented algorithms and set up the parameters corresponding to each former algorithm. Likewise, any image can be opened and training results may be saved on an XML file. Once executed, a method will come up with that all relevant information and may be viewed by the user if required.

Classification algorithms: the core of the project. Five algorithms have been implemented: Vector Quantization, Balasko, K-means, Lloyd and Bayes.

1.1.4.KEY WORDS

Natural Texture Classification, segmentation, vector quantization, unsupervised Fuzzy clustering, K-means, Lloyd, Bayes, parametric Bayesian classifier, crisp, fuzzy logic, clustering.

1.2.JUSTIFICACIÓN DEL PROYECTO

La segmentación de imágenes constituye uno de los primeros y más importantes pasos en el análisis de imágenes, campo que hoy en día cuenta con innumerables aplicaciones: bases de datos multimedia, transmisión de vídeo a través de Internet, fotografía digital, procesado de imágenes médicas y de ingeniería civil, agricultura, etc. Esto hace de la segmentación un problema de notable interés, sobre el que se llevan a cabo múltiples estudios.

Ámbitos sobre los que se desarrollan aplicaciones de esta índole:

1. En agricultura:
   - Control de cultivos, por ejemplo, para aplicación de herbicidas, control de riegos, etc.
   - Evaluación de catástrofes naturales: erupciones volcánicas, inundaciones, fuego, etc.
   - Control de edificaciones e impacto medioambiental.
   - Detección de infraestructuras.
2. Fines policiales:
   - Reconocimiento de matrículas de vehículos.
3. Guiado automático de vehículos:
   - Proyecto ALVINN: [1]
4. Medicina:
   - Refuerzo de las técnicas de rayos, etc.

Encontramos diversas empresas que desarrollan o utilizan aplicaciones de esta índole. Por citar algunos ejemplos:

- SGM (Servicio Geológico Mexicano) http://www.coremisgm.gob.mx/
- Digital Image Processing (DIMAP) http://www.dimap.es/
- Proespacio http://www.proespacio.org/
- Organismos oficiales y centros de investigación.
  - CSIC : Consejo Superior de Investigaciones Científicas.
  - INTA: Instituto Nacional de Técnica Aeroespacial.

Algunas aplicaciones:

i. Medicina:
   1. MIMvista: www.MIMvista.com
   2. Syntermed: www.syntermed.com
   3. MIStar: www.apollomit.com

1 http://www.ri.cmu.edu/research_project_detail.html?type=description&project_id=160&menu_id=261

4. iBrain: www.brain.org.au

ii. Ciencia:
   1. Corner Cube Environment: Software para visualización de datos de neuroimagen www.neurovia.umn.edu
   2. RAT: Herramientas para el procesamiento de datos SAR (radar de apertura sintética) www.cv.tu-berlin.de/rat/
   3. Meteor: Software de procesamiento de imágenes por satélite para la meteorología. www.esands.com

iii. Recursos naturales:
   1. TRMM Orbit Viewer: Mostrar datos para medir precipitaciones. http://daac.gsfc.nasa.gov
   2. SpaDIS: Detección de yacimientos naturales. http://www.vearncombe.com/ver1/spadis3D.html

iv. Otros:
   1. ERDAS IMAGINE: http://www.imagenesgeograficas.com/ERDAS.html
   2. Intergraph http://www.intergraph.com/
   3. ENVI-IDL http://www.ittvis.com/
   4. ILOG http://www.ilog.com/
   5. E-Cognition http://www.e-cognition.net/

En general, las herramientas anteriores usan algunos de los algoritmos clásicos de clasificación, siendo necesaria la intervención del usuario mediante programación para abordar algunas de las tareas relacionadas con la clasificación de texturas, lo cual no está al alcance de todo tipo de usuarios.

Aunque bien es cierto que cada día es mayor la potencialidad de las herramientas o de las nuevas que se van creando a lo largo de los años, no es menos cierto que los retos tecnológicos derivados hacen que en algunos casos la utilización de tales herramientas sea insuficiente para abordar las propuestas de proyectos demandados por los clientes.

Además, y lo que es más importante, en muchos casos no existe la posibilidad de llevar a cabo la investigación necesaria para abordar dicha problemática.

Por todo lo expuesto anteriormente surge una necesidad importante en el ámbito de las aplicaciones reales para abordar el tema de la clasificación de texturas naturales en imágenes y un reto para la comunidad científica para tratar de mejorar los procedimientos existentes.

1.3.OBJETIVOS

Los objetivos a conseguir están basados en el desarrollo de la aplicación propuesta, así como de su memoria.

Luego, los objetivos son los siguientes:

- Estudio de diferentes procedimientos de segmentación y elección de un conjunto de los mismos para el desarrollo de una aplicación.
- Análisis de los procedimientos seleccionados.
- Implementación de los procedimientos.
- Comunicación entre el usuario y el programa mediante una interfaz gráfica.
- Garantizar el correcto funcionamiento de la aplicación.
- Realización de la memoria.

1.4.ORGANIZACIÓN DE LA MEMORIA

La presente memoria está organizada de forma que en el capítulo 2 se explica la planificación del algoritmo. Este apartado incluye la organización del mismo y el plan de fase seguido.

En el capítulo 3 se analizan los distintos métodos de segmentación que van a ser implementados. El estudio de estos métodos conllevará la elección de estructuras de datos que serán usadas en la implementación.

El diseño de la aplicación implementada, será desarrollado en el capítulo 4. Este capítulo contendrá la argumentación de las distintas divisiones en paquetes de las clases implementadas, así como, múltiples diagramas que explican los distintos usos del sistema, comportamiento del mismo, etc. Para finalizar este capítulo, se llevará a cabo un estudio de los diferentes riesgos que encontraremos a la hora de desarrollar una aplicación.

En el capítulo 5 se argumentará las decisiones obtenidas respecto a la construcción del sistema. Entre estas decisiones, podemos encontrar, el sistema operativo, entorno de programación, documentos XML, etc.

Los capítulos 6 y 7 están destinados al estudio de los resultados obtenidos de la aplicación implementada en el entrenamiento y clasificación, respectivamente.

Se realiza un estudio de tiempo en el capítulo 8. Este estudio está destinado para llevar a cabo una comparación, ya sea entre los distintos métodos o entre los mismos métodos variando sus parámetros de entrada.

En el capítulo 9 se mostrará las distintas pruebas realizadas a la aplicación. Dichas pruebas servirán para comprobar su correcto funcionamiento.

El capítulo 10 desarrolla las distintas líneas futuras para la ampliación o mejora de la aplicación.

Para finalizar, en el capítulo 11 se presenta un tutorial para aprender el manejo de la aplicación implementada.

2. PLANIFICACIÓN

Se explica a continuación la planificación del proyecto. Para ello, podremos ver cómo se ha dividido el mismo y como se ha ido afrontando cada una de sus partes y el tiempo necesitado para cada una de ellas.
Este proyecto ha sido realizado en tres meses por una única persona, Daniel de Santos Sierra, lo que supone que los plazos marcados para conseguir los objetivos sean relativamente cortos en relación con proyectos de mayor duración.

2.1.ORGANIZACIÓN

El proyecto consiste en una aplicación que se puede diferenciar en dos partes: interfaz gráfica y procedimientos de segmentación. Para la realización de la aplicación, se ha partido de un estudio de los procedimientos de segmentación y posteriormente se ha creado una interfaz gráfica que satisficieran las necesidades de cada procedimiento.

Dado que el proyecto está realizado únicamente por una persona, no ha sido necesario establecer pautas de grupo, evitando así la dependencia con otras personas para la finalización del trabajo.

Puesto que el tiempo para llevar a cabo el proyecto ha sido escaso, las dos partes del proyecto han sido desarrolladas en paralelo, para poder probar que cada uno de los objetivos conseguidos eran correctos. Los objetivos están descritos en el apartado dedicado al plan de fase.

2.1.1.ORGANIZACIÓN DEL PROYECTO

Para la realización del proyecto, se ha establecido un plan de fase, estableciendo desde un principio las fechas para conseguir objetivos. De esta forma, se consigue un trabajo constante.

La comunicación con terceras personas no ha sido necesaria, de haberlo sido, se habría hecho uso de repositorios, denominados control de versiones [1], por ejemplo, SVN [2], Mercurial [3], Git [4], etc. Aunque estos repositorios también se podrían haber utilizado para llevar un control de trabajo, se ha optado por algo más sencillo, DropBox [5]. Su funcionamiento es sencillo, es una carpeta de nuestro sistema que se sincroniza gracias a internet, de esta forma, siempre mantendremos una copia del proyecto, así como de la memoria tanto en nuestro ordenador como en internet.

Se ha realizado un estudio de los procedimientos a implementar, calculando la complejidad de cada uno y estudiando los datos de entrada necesarios. Los procedimientos a implementar son: Cuantización Vectorial, Fuzzy clustering no supervisado, K-Medias, Lloyd y Bayes.

Posteriormente, se ha realizado un estudio para la mejora de la implementación de los algoritmos, para una mayor eficiencia en tiempo y en espacio.

Una vez realizada una mejora teórica, se ha pasado a implementar el método y la interfaz para el mismo, con los datos de entrada necesarios.

2.2.PLAN DE FASE

2.2.1.FASE DE INICIO

Fecha de inicio: 27 de Mayo de 2009
Fecha de conclusión: 3 de Junio de 2009

Los objetivos de la fase de inicio son los siguientes:

- Concreción con el tutor del alcance y objetivos del proyecto.
- Captura de requisitos de acuerdo con los objetivos acordados con el profesor.
- Estudio de los diferentes procedimientos: Cuantización Vectorial, Fuzzy clustering no supervisado, K-Medias, Lloyd y Bayes.
- Establecimiento del plan de fase.
- Elección del lenguaje para la realización del proyecto. En este caso, se ha optado por JAVA, aunque se han considerado otros como MATLAB y C++. En este caso, por cuestiones de tiempo para la realización del proyecto, se ha optado por JAVA, pero, dado que trabajamos con imágenes, el mejor lenguaje de programación sería C++ por su rapidez.
- Familiarización con la librería JAI [6] para el tratamiento de imágenes. Realización de diferentes ejemplos.
- Familiarización con la librería JDOM [7] para el tratamiento de los ficheros XML. Realización de diferentes ejemplos.
- Familiarización con la librería JAMA [8] para facilitar las operaciones con matrices.

2.2.2.FASE DE DISEÑO

Fecha de inicio: 4 de Junio de 2009

Fecha de conclusión: 11 de Junio de 2009

Los objetivos de la fase de diseño son los siguientes:

- Estudio de patrones de diseño que se utilizarán en dicha aplicación.
- Desarrollo de diagramas UML (Unified Modeling Language [9]): Diagramas de casos de uso, diagramas de clase, diagramas de secuencias, ...
- Concreción de fechas para la realización de cada parte de la aplicación.
- Diseño de los procedimientos de segmentación.
- Diseño de la interfaz gráfica necesaria para los procedimientos.
- Diseño de los documentos XML que utiliza la aplicación.

2.2.3.FASE DE CONSTRUCCIÓN

La fase de construcción está dividida en dos iteraciones. La aplicación tiene una parte de entrenamiento y otra de clasificación, cada una de las iteraciones se ocupa de una de las partes.

ITERACIÓN 1:

Fecha de inicio: 12 de Junio de 2009
Fecha de conclusión: 30 de Junio de 2009

- Revisión de la especificación de requisitos.
- Implementación de los distintos procedimientos para realizar el entrenamiento.
- Revisión y mejora de los procedimientos.

ITERACIÓN 2:

Fecha de inicio: 1 de Julio de 2009
Fecha de conclusión: 15 de Julio de 2009

- Revisión de la especificación de requisitos.
- Implementación de los distintos procedimientos para realizar la clasificación.
- Elaboración del prototipo final de la aplicación.
- Elaboración de la documentación correspondiente a esta fase.

2.2.4.FASE DE PRUEBAS

Fecha de inicio: 16 de Julio de 2009

Fecha de conclusión: 31 de Julio de 2009

La fase de pruebas está compuesta por una iteración. Los objetivos a alcanzar son los siguientes:

- Diseño de un plan de pruebas para comprobar el correcto funcionamiento de los algoritmos.
- Realización de mejoras en la aplicación.
- Elaboración de la documentación correspondiente a esta fase.
- Revisión de la documentación realizada hasta el momento.

2.2.5.FASE DE DOCUMENTACIÓN

Fecha de inicio: 1 de Agosto de 2009
Fecha de conclusión: 31 de Agosto de 2009

Los objetivos de la fase de documentación son los siguientes:

- Realización de la memoria final del proyecto basándose en los documentos anteriores.
- Revisión de la memoria final.
- Revisión y generación de la documentación del código con Javadoc [10].

3. ANÁLISIS

3.1.INTRODUCCIÓN

Se va a realizar un análisis de las distintas decisiones que se tomarán para realizar este proyecto y posteriormente explicaremos cada uno de los métodos que serán implementados y cómo han afectado en ellos las decisiones tomadas.
Los libros de referencia para la especificación de los algoritmos pueden encontrarse en [11] y [12].

3.2.ESTRUCTURA DE DATOS

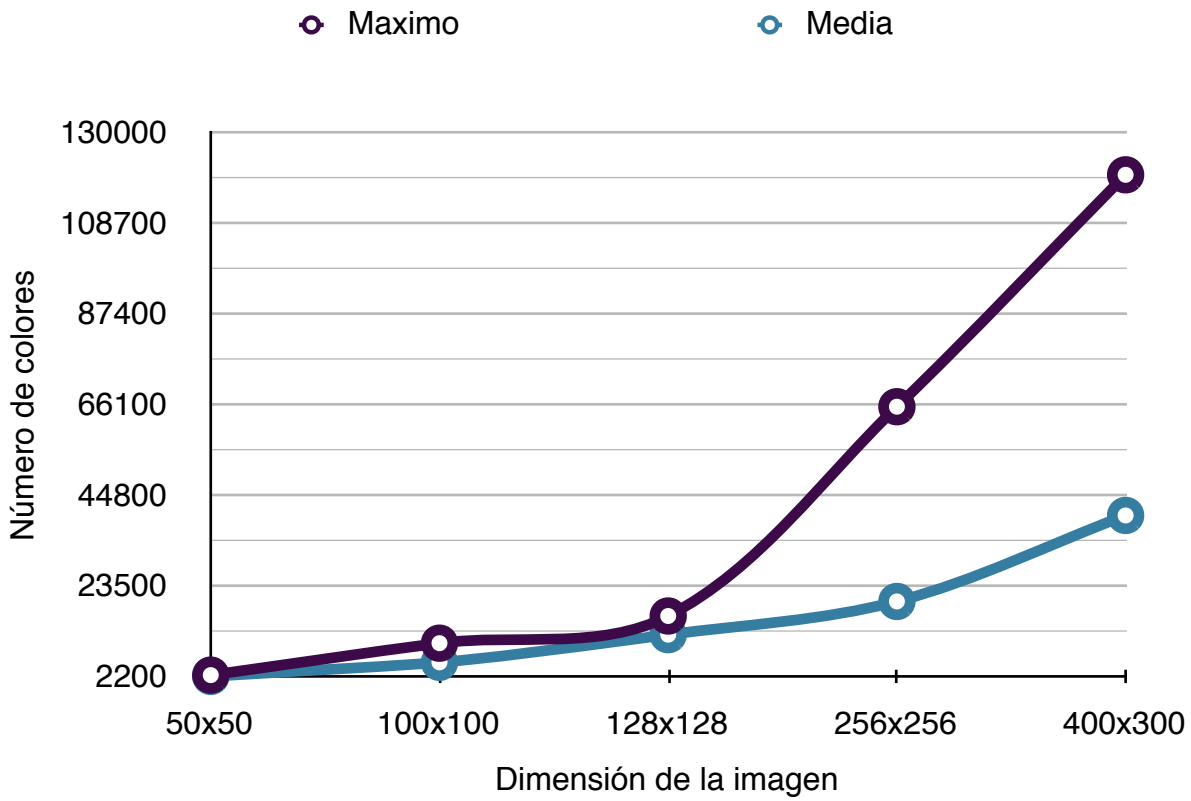
Antes de realizar un análisis de cada método a implementar, se ha realizado un estudio general de cada uno de ellos, llegando a la conclusión de que ningún método utiliza la propiedad espacial de los píxeles para realizar la segmentación. Es decir, todos los métodos agrupan en clases un conjunto de colores, sea cual sea su posición en la imagen. Se verá, posteriormente, en los apartados destinados al análisis de los métodos.
Como consecuencia, no necesitamos una estructura de datos que pueda representar la posición de cada píxel, por lo que se optó por representar la imagen de entrada como una tabla de frecuencias, en el que se asocia el color junto con el número de píxeles que tienen esa misma propiedad.
Se demuestra a continuación, que la estructura de datos elegida consigue mejores tiempos de proceso en todos los procedimientos y un menor espacio utilizado.
Para los estudios, vamos a utilizar imágenes de tamaño $n \times m$ píxeles, por lo que si se quisiera representar en el espacio dichos píxeles, se precisaría de una estructura de datos del orden de $n \times m$.

Se define c como el número de colores que tiene dicha imagen. Podemos afirmar que $c \leq n \times m$, pero lo que nos interesaría demostrar es que $c << n \times m$.

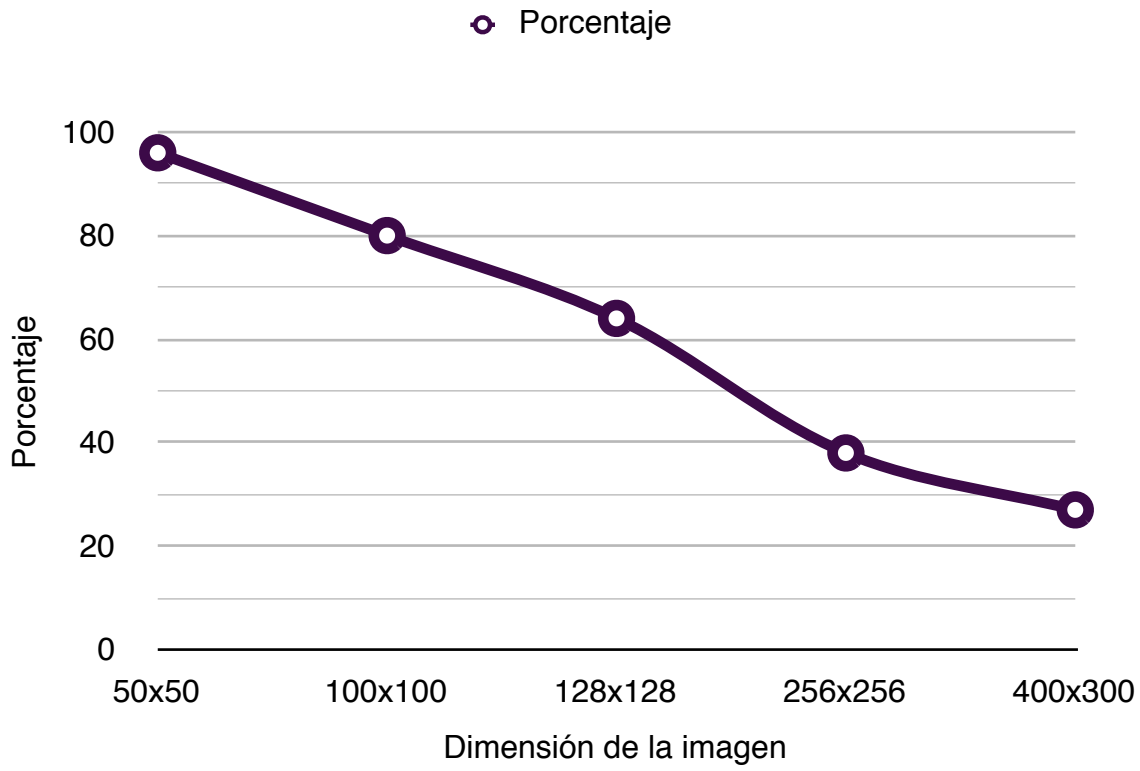
Se realiza un estudio con diferentes imágenes de distintos tamaños. Las imágenes elegidas corresponden a paisajes, y han sido elegidas aleatoriamente a partir del motor de búsqueda de Google. Los tamaños de dichas imágenes son: 50x50, 100x100, 128x128, 256x256 y 400x300. En el eje horizontal representamos el tamaño de dicha imagen ($n \times m$) y en el vertical el número de colores (c).

En la leyenda encontramos Máximo y Media. Se define Máximo como el número máximo de colores distintos que puede haber en la imagen, o lo que es lo mismo, el número de píxeles que la componen, es decir, $n \times m$. Se define Media como el número de colores distintos, de media, que hemos encontrado al realizar el estudio, con imágenes del tamaño correspondiente.

Podemos observar que para imágenes pequeñas la diferencia entre las dos gráficas es menor y que estas gráficas se separan a medida que la imagen se hace más grande, esto es debido a que a medida que las imágenes crecen los colores se repiten.
Esta gráfica, muestra también el espacio que necesitaríamos con las dos soluciones que estamos tratando. La solución que representa la propiedad espacial, correspondería con el Máximo, mientras que la solución que proponemos corresponde con la Media.
A simple vista, podemos ver que el espacio usado por nuestra solución es considerablemente menor a medida que las imágenes son mayores.



Otra forma de demostrar que $c << n \times m$ es mediante el porcentaje de color que hay en una imagen, respecto al tamaño de la misma.



Como se esperaba, a medida que la imagen aumenta de tamaño, el porcentaje de color decrece, lo que viene a demostrar que para imágenes mayores que 256x256 la cantidad de colores que la componen es menor que el 40%, por lo que, efectivamente, para imágenes lo suficientemente grandes podemos asegurar que $c << n \times m$.

Hemos demostrado entonces, que el espacio es considerablemente menor utilizando la tabla de frecuencias.

De igual forma podemos demostrar que con dicha tabla los algoritmos a implementar tendrán menor coste computacional, ya que estos deberán recorrer todos los colores que forman la imagen por lo menos una vez, como es el caso del algoritmo de cuantización vectorial. Por lo que, podemos asegurar que la tabla de frecuencias tiene un coste $O(c)$, mientras que la solución que representa la propiedad espacial tiene un coste $O(n \times m)$.

3.3.ESPECIFICACIÓN DEL PROYECTO

Como se ha explicado en el apartado anterior, se utilizará una tabla de frecuencias para representar la imagen de entrada. Por analogía, se usará el término histograma para referirnos a la tabla de frecuencias, pero con ciertas distinciones.

Un histograma representaría la cantidad de píxeles que tienen el mismo color de entre todos los colores posibles. En nuestro caso, nuestro histograma sólo tendrá aquellos colores que tengan

como mínimo un píxel con dicho color. Es decir, sería la representación gráfica de la tabla de frecuencias, o lo que es lo mismo, los valores no nulos del histograma.
En este apartado se explicarán todos los procedimientos implementados para realizar la segmentación.
Se ha realizado un estudio de cinco procedimientos: Cuantización vectorial, Fuzzy clustering no supervisado, K-Medias, Lloyd y Bayes.
En esta memoria, se hace uso de los términos entrenar y clasificar. A continuación se definirán los mismos para evitar ambigüedades.
Nos referimos a entrenamiento, cuando lo que queremos es determinar los centros de las clases que tiene una imagen, según un algoritmo y sus parámetros de entrada. Dichos centros se consideran aprendidos para un posterior uso.
Nos referimos a clasificación, cuando, a partir de un entrenamiento previo, usamos los datos aprendidos para segmentar una imagen.

3.3.1.ALGORITMO CUANTIZACIÓN VECTORIAL

IDEA GENERAL

La idea general de este procedimiento consiste en agrupar píxeles que tienen características parecidas. En este caso, dicha característica es la propiedad RGB de cada uno de ellos, es decir, su color.
Para realizar esa agrupación, el procedimiento explora píxel a píxel, de tal forma que si el píxel puede formar parte de algún grupo existente lo agrupa con él, pero si no, crea un grupo donde él es el único componente y donde, posteriormente, pueden ser agregados más píxeles.

ESPECIFICACIÓN FORMAL

Este algoritmo asume que el número de clases no se conoce inicialmente.

1. Para cada patrón se calcula su distancia[2] con todos los centros existentes. Para el primer elemento él mismo constituye el primer centro.
2. Tomar el centro más cercano utilizando una medida de distancia.
3. Si dicha distancia es menor que un umbral determinado previamente, se asocia el elemento a la clase y se calcula la media de todos los elementos que pertenecen a dicha clase. Esta medida nos proporciona el nuevo centro.
4. Si la distancia es mayor que el umbral prefijado se crea una nueva clase, asignando el valor del centroide al del elemento.

[2] Distancia euclídea entre los colores de los píxeles.

IMPLEMENTACIÓN

Disponemos de un umbral, dado por el usuario y un histograma, obtenido de la imagen de entrada.
En la especificación del algoritmo, podemos ver que se comparan las propiedades RGB[3] de cada píxel con las propiedades RGB de cada centro. Al usar un histograma como datos de entrada, cada paso del algoritmo se ejecuta sobre un conjunto de píxeles que tienen las mismas propiedades RGB, es decir, sobre píxeles que tienen el mismo color, ya que por tener el mismo color pertenecerán al mismo centro.
Debemos tener esto en cuenta a la hora de realizar el paso 3 antes descrito, puesto que al actualizar el centro no añadimos un nuevo píxel, sino que, es muy posible que añadamos más de uno.
Este método será utilizado, prácticamente, para obtener centros, relativamente óptimos, que serán usados posteriormente con otros métodos, como por ejemplo K-Medias, LLoyd y Bayes.

CONCLUSIÓN

Observamos que este método de segmentación no tiene en consideración la propiedad espacial de los píxeles, por lo que utilizar un histograma como datos de entrada supondría un menor coste en tiempo, como se ha razonado en el apartado Estructura de Datos, sección 3.2.

El resultado que ofrezca este método dependerá de la secuencia de procesamiento de los colores.
Para demostrar esta dependencia, vamos a mostrar dos ejemplos.
Para no complicarlo, vamos a agrupar un conjunto de números $x = \{2,3,6\}$ y un umbral $d = 3$. Esto quiere decir que, la distancia entre dos puntos que pertenezcan a una clase debe cumplir: $|x_i - x_j| \leq 3$

EJEMPLO 1:
Se ejecuta el algoritmo tomando los números de entrada en el orden de arriba.
En un primer paso, crearíamos un nuevo centro para 2, $c_1 = \{2\}$ cuya media sería $m_1 = 2$.

[3] RGB: color de un píxel. Un color está compuesto por tres componentes R (rojo), G (verde) y B (azul).

En un segundo paso, dado que $|m_1 - x_2| \leq 3$ añadimos el número a la clase 1 y actualizamos la media: $c_1 = \{2,3\}$ y $m_1 = 2.5$.

En el tercer paso, dado que $|m_1 - x_3| > 3$ creamos una nueva clase: $c_2 = \{6\}$

Tenemos como resultado dos clases: $c_1 = \{2,3\}$ $c_2 = \{6\}$

EJEMPLO 2:
Se ejecuta el algoritmo tomando los datos de entrada en el siguiente orden: $x_1 = 3; x_2 = 6; x_3 = 2$

Resultado: $c_1 = \{3,6,2\}$

Por tanto, se puede observar que los mismos datos de entrada en ordenes diferentes producen salidas distintas. Esto supone que imágenes parecidas den resultados distintos, por lo que este método, será utilizado únicamente para obtener centros iniciales que usarán los métodos K-Medias, Lloyd y Bayes para realizar el entrenamiento.

3.3.2.MÉTODO PSEUDOALEATORIO. AGRUPAMIENTO BORROSO NO SUPERVISADO

IDEA GENERAL

Este método debido a Balasko no realiza la segmentación de la imagen, simplemente, ofrece a otros métodos los centros necesarios para que se puedan ejecutar dichos métodos.
Los centros devueltos por este método son una estimación de los realmente existentes.

ESPECIFICACIÓN FORMAL

El procedimiento pseudoaleatorio descrito en Balasko y col. viene dado por la siguiente expresión:

$$\mathbf{v} = 2D\overline{\mathbf{M}} \circ R + D\overline{\mathbf{m}} \tag{3.1}$$

donde $\overline{\mathbf{m}}$ es la media de los valores de Y con dimensión $1xp$, $\overline{\mathbf{M}} = \max(abs(Y - \overline{\mathbf{m}}))$ es una matriz de dimensión $1xp$, $D = [1\ldots1]^t$ de dimensión $cx1$; R es una matriz de números aleatorios de dimensión cxp donde cada elemento de la matriz se obtiene utilizando la función $rand(.) - 0.5$, los valores aleatorios están restringidos al rango $[0,1]$. El operador ∘ expresa multiplicación de matrices elemento a elemento.

Debemos normalizar los valores de entrada en el rango $[0,1]$. La normalización resulta conveniente para controlar el hecho de que todos los datos contribuyan por igual en el cómputo.

$$Y = \frac{X - MIN}{MAX - MIN} \quad MIN = \min\{X\}; MAX = \max\{X\} \tag{3.2}$$

$$X, Y, MIN, MAX \in \Re^p$$

IMPLEMENTACIÓN

Los datos de entrada de los que disponemos, son el número de clústeres, c , y el histograma H de la imagen.

Dado que nuestros datos de entrada es H y no X, modificamos la fórmula de normalización.

$$Hy = \frac{Hx - MIN}{MAX - MIN} \quad MIN = \min\{Hx\}; MAX = \max\{Hx\} \tag{3.3}$$

$$Hx, Hy, MIN, MAX \in \Re^p$$

El método antes descrito, hace referencia a la propiedad RGB de cada píxel. En nuestro caso, los datos de entrada son píxeles agrupados por tener el mismo color, por lo que, deberemos tener esto en cuenta a la hora de realizar la media.

Si analizamos la fórmula $\overline{\mathbf{M}} = \max(abs(Hy - \overline{\mathbf{m}}))$, podemos observar que: $0 \le Hy \le 1$, puesto que es el resultado de la normalización. Por lo tanto, los únicos valores, de todos los valores que disponemos de Hy , que nos interesan son los extremos, 0 y 1. Dado que Hy está normalizada, siempre existirán estos valores. Luego, $\overline{\mathbf{M}} = \max(1 - \overline{\mathbf{m}}, \overline{\mathbf{m}})$.

Analizamos ahora R, que es una matriz de dimensión cxp , donde todos sus valores están comprendidos en el rango $[-0.5, 0.5]$, según su definición. Lo que podría suponer que en el resultado $\mathbf{v}$, tuviéramos valores negativos y esto no debería ser posible, puesto que nuestros valores esperados deberían estar en el rango $[0,1]$. Para evitar esto, volvemos a normalizar, obteniendo así $\mathbf{v'}$. Posteriormente, recuperaremos los valores reales de los centros con la siguiente fórmula:

$$C = (v' + MIN)(MAX - MIN) \tag{3.4}$$

Donde C son los colores de los centros.

CONCLUSIÓN

Es un método rápido para obtener centros de partida para poder realizar el entrenamiento con otros métodos.

Por la aleatoriedad que introduce este algoritmo, a veces los centros obtenidos pueden no servir para alguno de los otros métodos de segmentación. Por ejemplo, K-Medias

Crisp o Lloyd pueden despreciar alguno de los centros ofrecidos, que quedarán sin modificar al ejecutar dichos métodos.

3.3.3.AGRUPAMIENTO BORROSO

IDEA GENERAL

El objetivo de la técnica de *Agrupamiento borroso* o *Fuzzy Clustering* [11] consiste en dividir *n* objetos $x \in X$ caracterizados por *p* propiedades en *c* "clústeres" o grupos.

ESPECIFICACIÓN GENERAL

Los elementos de un clúster deben ser tan similares entre sí como sea posible y a la vez deben ser tan diferentes a los elementos de otros clústeres como también sea posible, el proceso se controla por el uso de medidas de similitud basadas en distancias. Así la similitud o la diferencia entre dos puntos x_k y x_l pueden interpretarse como la distancia entre esos puntos.

$$d : X \times X \to \mathbb{R}^+ \qquad (3.5)$$
$$d(x_k, x_l) = d_{kl} \geq 0;\ d_{kl} = d_{lk}\ y\ d_{kl} \leq d_{kj} + d_{jl}$$

Cada partición del conjunto $X = \{x_1, x_2, ..., x_n\}$ puede enfocarse desde dos perspectivas: fuzzy y no fuzzy. Una partición no fuzzy se conoce en terminología inglesa como "crisp". Si se desea realizar una partición del conjunto X en c clústeres tendremos $S_i\{i = 1, ..., c\}$ subconjuntos. A partir de esta consideración se define lo que se conoce como grado de pertenencia μ_{ik} de cada objeto x_k al subconjunto S_i.

Dado $X = \{x_1, x_2, ..., x_n\}$ y el conjunto V_{cn} de todas las matrices reales de dimensión cxp, con $2 \leq c < n$. Se puede obtener una matriz representando la partición de la siguiente manera $U = \{\mu_{ik}\} \in V_{cn}$. Tanto en el supuesto "crisp" como en el fuzzy se deben cumplir las siguientes condiciones:

1) $\mu_{ik} \in \{0,1\}$ *crisp o* $\mu_{ik} \in [0,1]$ *fuzzy* $\quad 1 \leq i \leq c; 1 \leq k \leq n$ (3.6)

2) $\sum_{i=1}^{c} \mu_{ik} = 1 \quad 1 \leq k \leq n$

3) $0 < \sum_{k=1}^{n} \mu_{ik} < n \quad 1 \leq i \leq c$

La localización de un clúster S_i se representa por su centro $v_i = \{v_{i_1}, v_{i_2}, ..., v_{i_n}\} \in \Re^p$ con $i = 1,..,c$, alrededor del cual se concentran los objetos.

ESPECIFICACIÓN "CRISP"

Un objeto x_k se dice que pertenece a un conjunto S_i dado y no pertenece al resto. Esto se expresa con los valores discretos $\{0,1\}$ de la siguiente forma: $\mu_{ik} = 1$ para indicar que pertenece y $\mu_{ik} = 0$ para expresar que no pertenece.

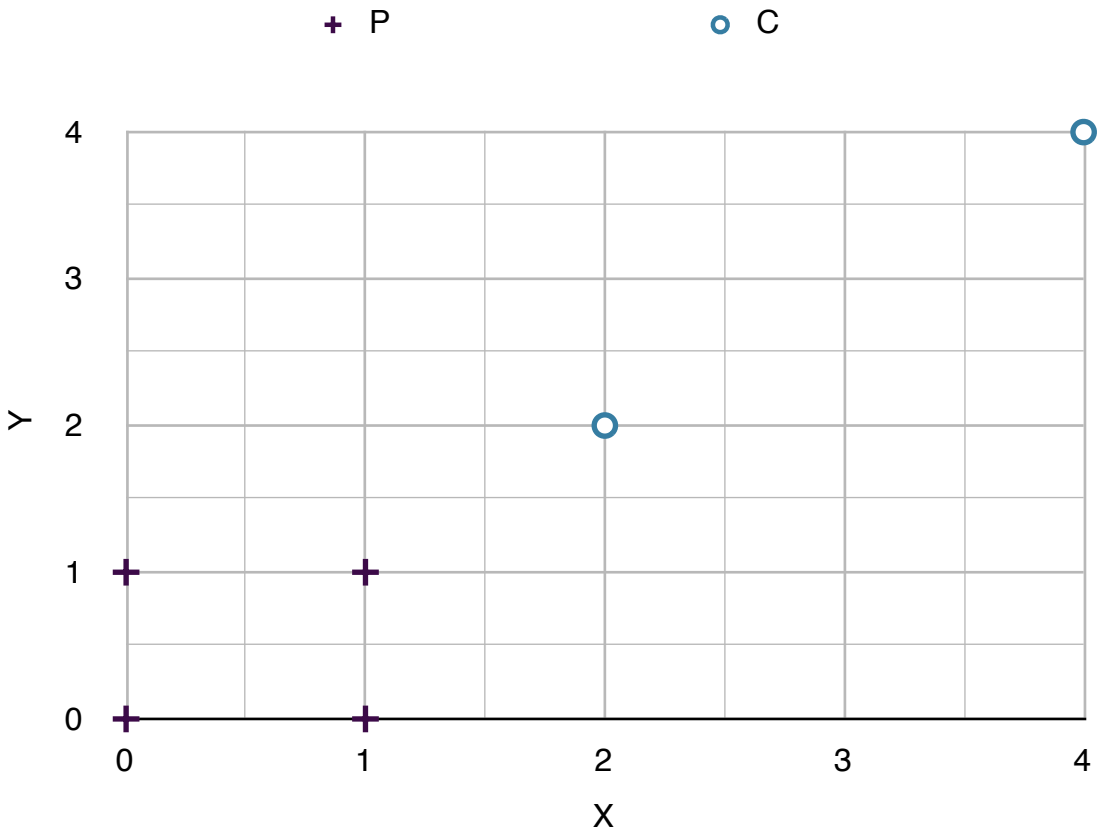
IMPLEMENTACIÓN "CRISP"

En cada paso, se calcula los grados de pertenencia y posteriormente se vuelve a calcular los centros, para este fin, se utiliza la media geométrica, que deberá tener en cuenta que se está tratando con un histograma, es decir, que para cada color perteneciente a un centro, puede haber más de un píxel.
El procedimiento terminará cuando los clústeres no varíen entre dos iteraciones consecutivas o cuando se haya alcanzado un máximo de iteraciones.

CONCLUSIÓN "CRISP"

Al realizar un agrupamiento borroso crisp, podemos llegar a perder alguno de los centros de partida. Esto sucede si alguno de los centros está lo suficientemente alejado como para que todos los grados de pertenencia con los colores sea 0.

Supongamos el siguiente caso:
Disponemos del siguiente conjunto de puntos, $P = \{(0,0),(0,1),(1,0),(1,1)\}$, $P \in \Re^2$.
Y tenemos los siguientes centros de partida $C = \{(2,2),(4,4)\}$.



En la figura previa, se puede ver con facilidad que todos los puntos de P tienen menor distancia (euclídea) al punto C_1 que al punto C_2, por lo que, como resultado, tendremos que el punto C_1 evoluciona hasta convertirse en la media de los cuatro puntos de P, mientras que el punto C_2, no evolucionará, formando así una clase vacía.

ESPECIFICACIÓN "FUZZY"

Disponemos de las siguientes fórmulas:

$$v_i = \frac{1}{\sum_{k=1}^{n} (\mu_{ik})^m} \sum_{k=1}^{n} (\mu_{ik})^m x_k \quad i = 1,...,c \tag{3.7}$$

$$\mu_{ik} = \frac{\left(\frac{1}{\|x_k - v_i\|_G^2}\right)^{2/m-1}}{\sum_{j=1}^{c}\left(\frac{1}{\|x_k - v_i\|_G^2}\right)^{2/m-1}} \quad i = 1,...,c;\ k = 1,...,n \tag{3.8}$$

donde $\|x_k - v_i\|_G^2 = (x_k - v_i)^t G(x_k - v_i)$

El exponente m se conoce como peso exponencial y disminuye la influencia del ruido al obtener los centros de los clústeres, reduciendo la influencia de los valores pequeños de μ_{ik} frente a los valores altos μ_{ik}. Cuanto mayor sea $m > 1$ mayor es dicha influencia.

El procedimiento se resume en los siguientes pasos:

1. Elegir c $(2 \le c \le n)$, m $(1 < m < \infty)$ y la matriz G de dimensión pxp siendo simétrica y definida positiva. Inicializar $U^{(0)}$ y poner $t = 0$.
2. Calcular los c centros fuzzy de los clústeres a partir de (3.7) $\{v_i^{(t)}\}$ utilizando $U^{(t)}$.
3. Calcular los nuevos grados de pertenencia de la matriz $U^{(t+1)}$, utilizando $\{v_i^{(t)}\}$ a partir de:

$$\begin{cases} \mu_{ik} = \dfrac{\left(\frac{1}{\|x_k - v_i\|_G^2}\right)^{2/m-1}}{\sum_{j=1}^{c}\left(\frac{1}{\|x_k - v_i\|_G^2}\right)^{2/m-1}} & si\ x_k \neq v_i^{(t)} \\ \mu_{jk} = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases} & si\ x_k = v_i^{(t)} \end{cases} \tag{3.9}$$

4. Elegir una norma matricial y calcular $\Delta = \|U^{(t+1)} - U^{(t)}\|_G$. Si $\Delta > \varepsilon$ poner $t = t + 1$ y regresar al paso 2, de lo contrario detener el proceso.

IMPLEMENTACION "FUZZY"

En el caso que nos ocupa, no disponemos de $X = \{x_1, x_2, ..., x_n\}$, puesto que la imagen de entrada viene dada por un histograma $H = \{H_1, H_2, ..., H_h\} \in \Re^p$ con h elementos.

Cada elemento de H , tiene asociado el número de píxeles $F = \{f_1, f_2, ..., f_h\}$ donde $1 \le f_k \le n$ y $\sum_{k=1}^{h} f_k = n$.

Se tiene entonces que, los grados de pertenencia ya no son de un píxel con un centro, sino, de un color o conjunto de píxeles con un centro. Como consecuencia de este cambio, se modificará la formula de $\mathbf{v}_i$ antes dada.

$$\mathbf{v}_i = \frac{1}{\sum_{k=1}^{h} (\mu_{ik})^m \cdot f_k} \sum_{k=1}^{h} (\mu_{ik})^m \cdot H_k \cdot f_k \quad i = 1, ..., c \tag{3.10}$$

Para el calculo de μ_{ik} utilizaremos la misma fórmula, pero teniendo en cuenta que el grado de pertenencia será del color de un conjunto de píxeles a los distintos centros.

$$\begin{cases} \mu_{ik} = \dfrac{\left(\dfrac{1}{\|H_k - v_i\|_G^2} \right)^{2/m-1}}{\sum_{j=1}^{c} \left(\dfrac{1}{\|H_k - v_i\|_G^2} \right)^{2/m-1}} & si\ H_k \neq v_i^{(t)} \\ \mu_{jk} = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases} & si\ H_k = v_i^{(t)} \end{cases} \tag{3.11}$$

Se estable el límite de iteraciones máximas a 400.

CONCLUSIÓN "FUZZY"

Con el agrupamiento borroso fuzzy, todos los centros evolucionan constantemente, consiguiendo así que no se produzca, desde un principio, el mismo problema que con el método crisp.

3.3.4.ALGORITMO GENERALIZADO DE LLOYD

IDEA GENERAL

Dados unos centros iniciales, este algoritmo, ajusta dichos centros para realizar una segmentación de la imagen.
El usuario ofrecerá al algoritmo los parámetros necesarios, número de iteraciones y la razón de aprendizaje. Con dichos parámetros conseguimos que el algoritmo haga variar los centros con mayor facilidad, o por el contrario, que varíen muy poco.

ESPECIFICACIÓN FORMAL

El objetivo consiste en determinar los centros $\mathbf{c}_j$ de los $j=1,...,m$ clústeres.

1. Inicio: dados los puntos de datos $x(k)$, $k=1,2,...$, y centros de salida iniciales $c_j(0)$,
2. Determinar el centro $c_j(k)$ más próximo al punto $x(k)$

$$j = \arg\min_j \left\| x(k) - c_j(k) \right\|^2 \tag{3.12}$$

3. Actualizar el centro de salida utilizando las ecuaciones siguientes

$$\begin{aligned} c_j(k_j+1) &= c_j(k_j) + \gamma(k_j)\left\lfloor x(k) - c_j(k_j) \right\rfloor \\ k_j &= k_j + 1 \end{aligned} \tag{3.13}$$

Obsérvese que cada centro puede tener su propia razón de aprendizaje, lo que se indica con k_j en $\gamma(k_j)$

IMPLEMENTACIÓN

En nuestro caso, no disponemos de $X = \{x_1, x_2, ..., x_n\}$, puesto que nuestra imagen de entrada viene dada por un histograma $H = \{H_1, H_2, ..., H_h\} \in \Re^p$ con h elementos, donde $1 \le h \le n$. Cada elemento de H tiene asociado el número de píxeles $F = \{f_1, f_2, ..., f_h\}$ donde $1 \le f_k \le n$ y $\sum_{k=1}^{h} f_k = n$.

Por lo tanto, no podríamos usar la fórmula del paso 3 de la especificación en un sólo paso, tendríamos que ejecutarla por cada H_k, f_k veces. Esto no resulta interesante, puesto que, claramente, podríamos modificar la fórmula para que cuente todos los píxeles del mismo color en un sólo paso, evitando así cálculos innecesarios.

Desarrollando la fórmula anterior, llegamos a la siguiente:

$$c_j(k_j+1) = c_j(k_j)\left((1-\gamma(k_j))^{f(k)}-1\right)\lfloor H(k) - c_j(k_j)\rfloor \qquad (3.14)$$
$$k_j = k_j + 1$$

Esta fórmula sí es interesante, puesto que nos permite tratar un conjunto de píxeles, con el mismo color, en un único paso. Aumentando así la eficiencia del procedimiento.

CONCLUSIÓN

Este método modifica aquellos centros que están más cerca de los colores, por lo que, al igual que en el agrupamiento borroso crisp, puede haber centros lo suficientemente alejados como para que no evolucionen durante la ejecución completa del algoritmo.

3.3.5. TEORÍA DE LA DECISIÓN DE BAYES: EL CLASIFICADOR BAYESIANO

IDEA GENERAL

La teoría de la decisión de Bayes es un método estadístico clásico de clasificación de patrones.
Dicha teoría plantea que la tarea de predecir la clase a la que pertenece un elemento se puede tratar en términos probabilísticos, [13].

ESPECIFICACIÓN FORMAL

En el caso general y más típico de una distribución de probabilidad Gaussiana o Normal multivariable, ni la media m ni la matriz de covarianza C son conocidas. Por tanto, esos parámetros desconocidos constituyen las componentes del vector de parámetros $w = \{m, C\}$. Consideremos el supuesto univariable con $m = m$ y $C = \sigma^2$, en cuyo caso

$$\ln p(x_i / w) = -\frac{1}{2}\ln 2\pi C - \frac{1}{2C}(x_i - m)^2 \tag{3.15}$$

$$\nabla_w \ln p(x_i / w) = \begin{bmatrix} \frac{1}{C}(x_i - m) \\ -\frac{1}{2C} + \frac{(x_i - m)^2}{2C^2} \end{bmatrix} \tag{3.16}$$

La minimización sobre los datos de entrenamiento conduce ahora a las condiciones,

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\hat{C}}(x_i - \hat{m}) = 0 \qquad -\frac{1}{n}\sum_{i=1}^{n}\frac{1}{2\hat{C}} + \frac{1}{n}\sum_{i=1}^{n}\frac{(x_i - \hat{m})^2}{2\hat{C}^2} = 0 \tag{3.17}$$

donde $\hat{m}$ y $\hat{C}$ son las estimas de máxima verosimilitud para m y C, respectivamente. Sustituyendo $\hat{m} = \hat{m}$ y $\hat{\sigma}^2 = \hat{C}$ obtenemos las estimas de máxima verosimilitud para m y σ^2

$$\hat{m} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - m)^2 \tag{3.18}$$

Aunque el análisis del caso multivariable es básicamente muy similar, se requiere mucha más manipulación. El resultado muy bien conocido en estadística es que las estimas de máxima verosimilitud para m y C están dadas por,

$$m = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad C = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - m)(x_i - m)^t \tag{3.19}$$

Estas expresiones nos dicen que la estima de máxima verosimilitud para el vector media es la media simple. La estima de máxima verosimilitud para la matriz de covarianza es la media aritmética de las *n* matrices $(x_i - m)(x_i - m)^t$. Puesto que la verdadera matriz de covarianza es el valor esperado de la matriz $(x_i - m)(x_i - m)^t$, se obtiene un resultado muy satisfactorio.

Una vez estimados los parámetros **m** y C, la función de densidad de probabilidad queda perfectamente especificada por la siguiente ecuación suponiendo que dicha función sigue una distribución Gaussiana.

$$p(x|m,C) = \frac{1}{(2\pi)^{d/2}|C|^{1/2}} \exp\left\{-\frac{1}{2}(x-m)^t C^{-1}(x-m)\right\} \tag{3.20}$$

Utilizando el teorema de Bayes y considerando que tanto las probabilidades a priori $P(y = c_j)$ como las densidades condicionales para cada clase $p(x / y = c_j)$ son conocidas o se pueden estimar, es posible determinar para una observación dada x la probabilidad de que esa observación pertenezca a una determinada clase. Estas probabilidades, llamadas probabilidades a posteriori, pueden usarse para construir una regla discriminante

$$p(y = c_j / x) = \frac{p(x / y = c_j)P(y = c_j)}{p(x)} \tag{3.21}$$

donde

$$p(x) = \sum_{j=1}^{c} p(x / y = c_j)P(y = c_j) \tag{3.22}$$

A partir de las ecuaciones anteriores dado x , la regla de decisión viene establecida por,

$$x \in c_i \; sii \; p(y = c_i / x) > p(y = c_j / x) \; \forall i \neq j, \; i, j = 1,2,...,c \tag{3.23}$$

Fijándose en el segundo término de la expresión (3.16) del teorema de Bayes y eliminado el término no discriminante $p(x)$ (no aporta nada en la decisión), se tiene una forma alternativa de clasificar el vector de atributos x :

$$x \in c_i \; sii \; p(y=c_i)P(y=c_i) > p(y=c_j)P(y=c_j) \; \forall i \neq j, \; i,j=1,2,...,c \qquad (3.24)$$

Generalmente las distribuciones de densidad de probabilidad se eligen Normales o Gaussianas.

Un caso especial surge cuando las probabilidades a priori son iguales para todas las clases, ya que en esta situación la distancia de Mahalanobis se puede utilizar como función discriminante mediante la siguiente regla de decisión a partir de (3.20) y teniendo en cuenta el signo negativo en el término exponencial de la función de densidad de probabilidad Normal, así

$$x \in c_i \; sii \; d_M^2(x,m_i) < d_M^2(x,m_j) \quad \forall i \neq j, i,j=1,2,...,c \qquad (3.25)$$

donde m_i , m_j son los vectores media de las clases c_i y c_j respectivamente. La distancia de Mahalanobis es una distancia implícita en la ecuación (3.20). Sin pérdida de generalidad, la distancia de un vector x_k a la clase c_i resulta ser:

$$d_M^2(x_k,m_i) = (x_k - m_i)^t C_i^{-1}(x_k - m_i) \qquad (3.26)$$

En el supuesto de que las matrices de covarianza sean la identidad, la distancia de Mahalanobis al cuadrado resulta ser la distancia Euclídea al cuadrado, en cuyo caso tendríamos,

$$d_E^2(x,m_i) = (x - m_i)^t (x - m_i) = x^t x - 2x^t m_i + m_i^t m_i \qquad (3.27)$$

En la expresión anterior el término $x^t x$ no discrimina, ya que se repite en todas las clases, de forma que puede despreciarse. Ahora, si se cambia de signo y se divide por 2 en la ecuación (3.27) se obtiene la siguiente función discriminante,

$$fd_i(x) = x^t m_i - \frac{1}{2} m_i^t m_i \qquad (3.28)$$

Como el resultado de (3.27) es una cantidad positiva, al haber eliminado el término $x^t x$ y cambiado de signo los restantes, se deduce que la distancia Euclídea al cuadrado mínima hace la expresión (3.28) máxima.

IMPLEMENTACIÓN

La especificación anterior toma los colores de cada píxel de uno en uno, dado que nuestro histograma los ha agrupado según sus colores, *x* representará un color que tiene asociada una frecuencia de aparición en la imagen. Por lo que deberemos prestar atención a la hora de realizar la media y la covarianza, para actualizarlas de forma correcta.

La regla de decisión ya no tomará la decisión de si un píxel forma parte de una determinada clase, sino que, decidirá si un conjunto de píxeles forman parte de dicha clase, es decir, si un color pertenece a esa clase o no.

4. DISEÑO

4.1.ESPECIFICACIÓN DEL DISEÑO

La aplicación está dividida en tres paquetes: MétodosAC, GUI y Librerías.

MétodosAC: contiene la implementación de los métodos comentados en secciones anteriores.
Este paquete está compuesto por el patrón Factoría, que decidirá en todo momento qué método se ejecutará.
Debe ser capaz de, dado un histograma, devolver la información necesaria resultante de realizar la segmentación de la imagen según el método y los parámetros elegidos por el usuario.

GUI: paquete para la interfaz gráfica de la aplicación.
Debe ofrecer al usuario la posibilidad de manejar la aplicación de forma clara y sencilla. Para ello, el usuario podrá abrir nuevas imágenes, realizar el entrenamiento y clasificación de las imágenes según el método y los parámetros deseados y guardar y abrir el resultado de un entrenamiento previo.

Librerías: conjunto de librerías utilizadas, JAI, JDOM y JAMA.

JAI: La API de Java Advanced Imaging [6], proporciona un conjunto de interfaces orientadas a objetos que permite la manipulación de imágenes fácilmente, en aplicaciones y applets. JAI proporciona un alto rendimiento independientemente de la plataforma.
Esta librería será utilizada para leer la imagen y crear los resultados. Además será utilizada para crear el histograma necesario en nuestra aplicación.

JDOM: Java Document Object Model, [7], proporciona una solución Java para acceso, manipulación y escritura de archivos XML.
Esta librería será utilizada para manejar documentos XML, muy necesarios en nuestra aplicación, dado que el resultado del entrenamiento ha sido escrito en este lenguaje de marcado.

JAMA: Java Matrix, [8], para la manipulación de matrices.

Se ha utilizado el patrón Singleton para crear la mayor parte de las clases. El patrón Singleton, es un patrón creacional que trata de asegurar que sólo existe una única instancia de una clase, y que hay un punto global de acceso a la misma.

4.1.1.DIAGRAMA DE CASOS DE USO

Para crear un diagrama de casos de uso, debemos saber en primer lugar, a qué usuarios va destinada la aplicación, de tal forma que, dado estos usuarios podemos decir, de forma unívoca, que actividades pueden realizar sobre nuestro sistema cada uno de ellos.
En nuestro caso, el usuario que puede utilizar nuestra aplicación es cualquiera, dado que no queremos ofrecer a usuarios distintos actividades distintas. Es decir, cualquier persona podría utilizar la aplicación con toda su funcionalidad, sin ningún tipo de restricción.

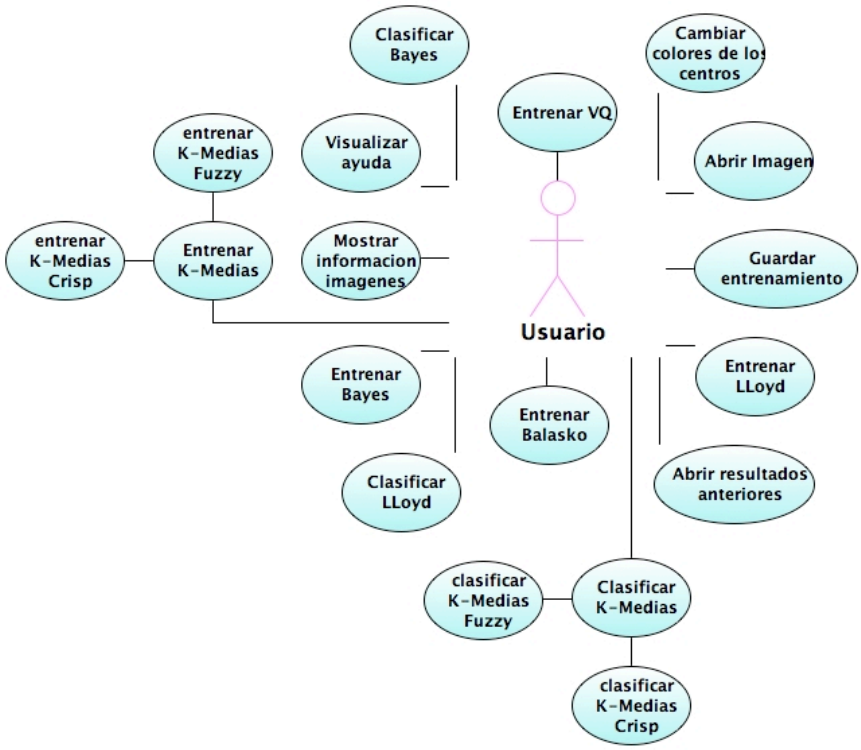


Figura 4.1.: Casos de uso: representa las actividades que puede realizar el usuario sobre la aplicación.

4.1.2. DIAGRAMA DE CLASES GENERAL

Basándonos en el diagrama de casos de uso, se distinguen las clases que necesitará el sistema. Por ejemplo, dado que se quiere entrenar y clasificar con el método de Lloyd, necesitamos una clase que, dados los parámetros necesarios, realice el entrenamiento y la clasificación. Para este mismo método, necesitamos que tome los parámetros que proporciona el usuario, por lo que, vemos necesaria otra clase, que se ocupe de recoger dichos parámetros.

Vemos entonces que podemos dividir la aplicación en dos partes, la parte gráfica (GUI) que será la encargada de realizar la comunicación con el usuario y otra parte encargada de realizar el entrenamiento y clasificación.

No obstante, se hace evidente dividir en dos paquetes la aplicación, añadiendo un tercer paquete de librerías que facilitará el manejo de imágenes, XML y matrices.

A continuación tenemos el diagrama de clases de la aplicación. Para simplificarlo, no se describen ni los métodos ni los atributos de cada clase. Se comentarán posteriormente aquellos métodos de especial relevancia.

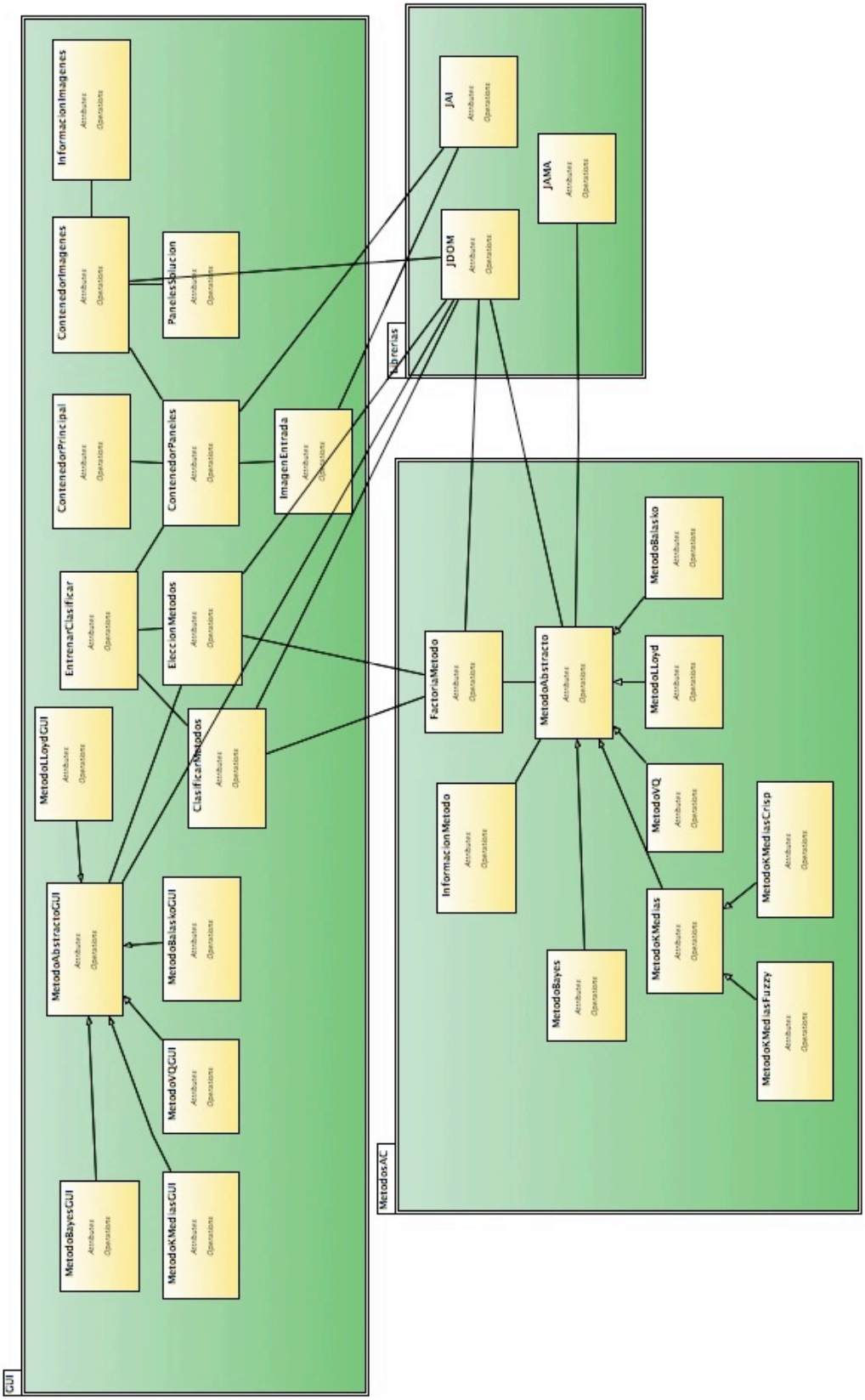


Figura 4.2.: Diagrama de clases: representa las clases necesarias para implementar la aplicación.

A continuación se muestran los diagramas de clases divididos en paquetes para poder verlos con mayor claridad.

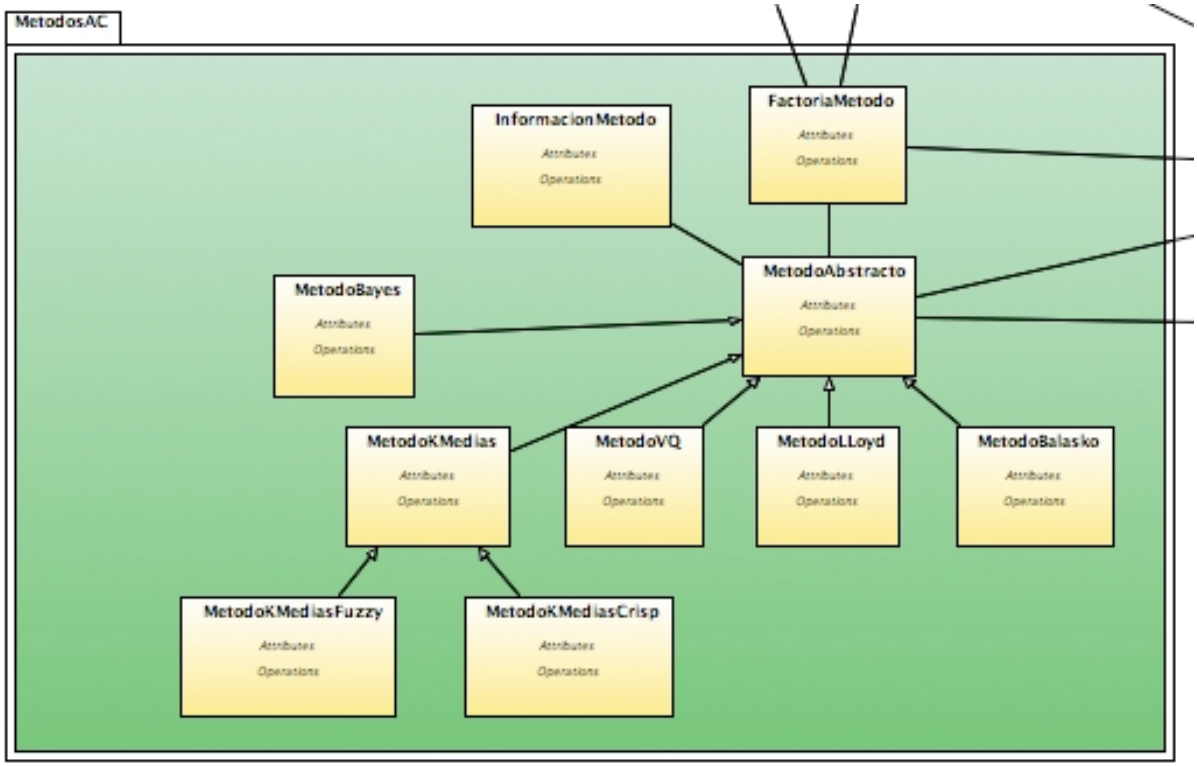


Figura 4.3.: Diagrama de clases correspondiente con el paquete MetodosAC.

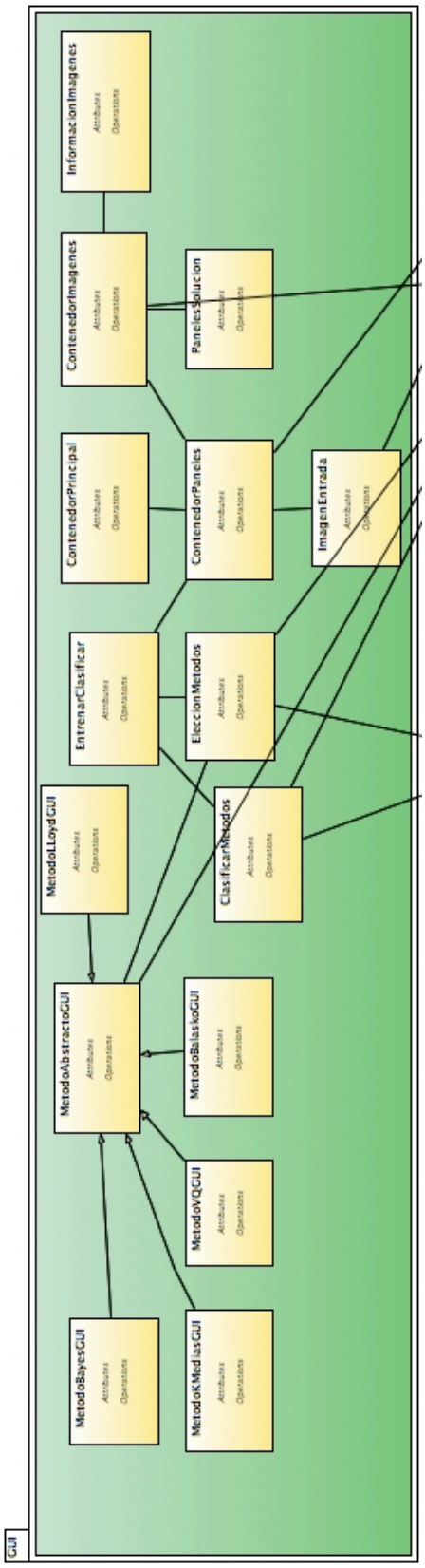


Figura 4.4.:Diagrama de clases del paquete GUI

4.1.3.DIAGRAMAS DE SECUENCIA

A continuación, mostramos la interacción de las distintas clases de nuestro sistema a la hora de realizar alguna de las operaciones definidas en los casos de uso.
El sistema permite entrenar y clasificar con un conjunto de métodos ya comentados en secciones anteriores, por lo que, después de elegir la imagen a estudiar, el usuario tendrá que decidir si quiere entrenar o clasificar.

El siguiente diagrama de secuencia muestra la interacción de las distintas clases si el usuario decide entrenar.

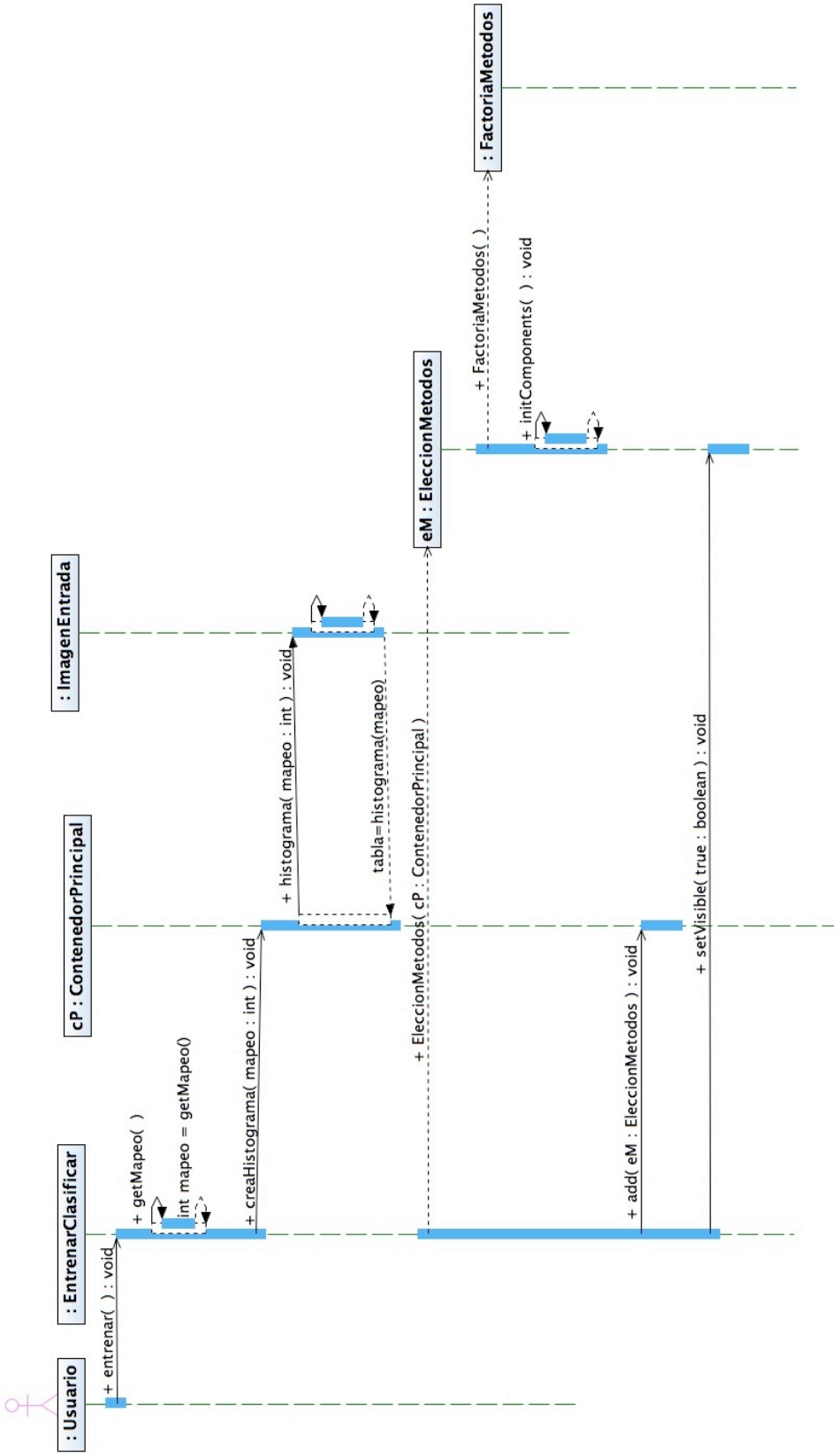


Figura 4.5: Diagrama de secuencias: muestra los métodos ejecutados cuando el usuario decide entrenar.

Si observamos el diagrama de la figura 4.5., vemos que, una vez el usuario decide entrenar, se recuperan datos de la interfaz EntrenarClasificar, y gracias a estos datos se crea el histograma de la imagen de entrada. La variable mapeo, si por ejemplo fuera 2, significará que la aplicación sólo cogerá de cada fila de la imagen un píxel de cada dos, y sólo de una de cada dos filas. Como se muestra a continuación, de la imagen de entrada (figura 4.6.a) sólo tomaríamos los cuadrados negros que muestra la imagen de submuestreo (figura 4.6.b), siendo el mapeo igual a dos.

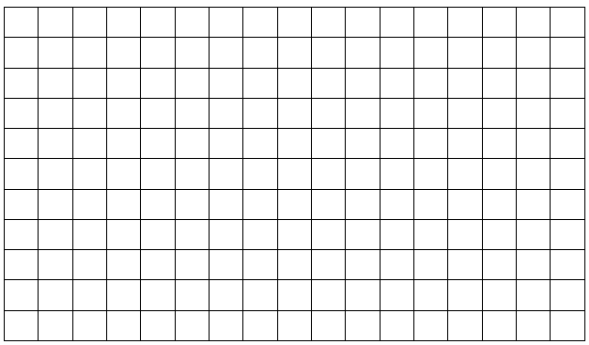


Figura 4.6.a.: Imagen de entrada

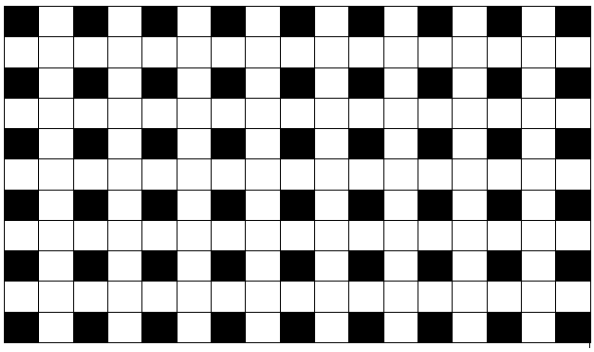


Figura 4.6.b.: Submuestreo

Este submuestreo se realiza para reducir el tiempo de procesamiento y pintado de la imagen segmentada.

El histograma creado sólo será generado una única vez para la misma imagen y con el mismo mapeo que se haya utilizado la primera vez.

Una vez en la fase de entrenamiento, el usuario podrá seleccionar cualquier método para comenzar el estudio. El sistema proporcionará valores por defecto en cada método.
El diagrama de secuencia de la figura 4.7. muestra como se ejecutaría un método seleccionado anteriormente, al que hemos introducido los datos necesarios.
Para realizar el diagrama lo más general posible, se ha utilizado X como nombre del método en ejecución. En todos los métodos se realizan los mismos pasos, las únicas variaciones destacables son: Agrupamiento borroso no supervisado no realizaría el último paso de pintar el resultado y Cuantización Vectorial sólo mostraría el resultado si el usuario lo pide explícitamente.

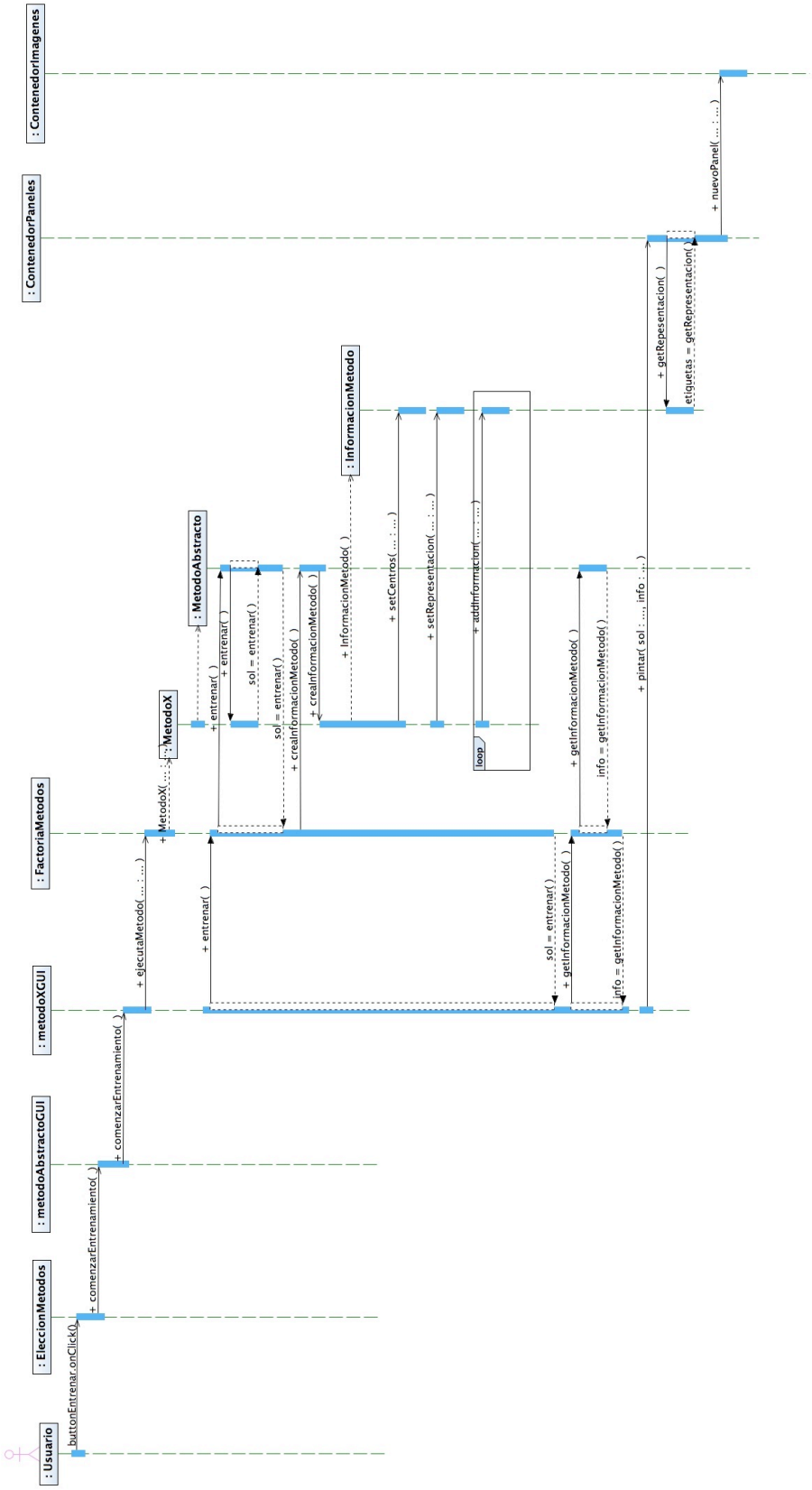


Figura 4.7.: Diagrama de secuencias: métodos ejecutados si el usuario quiere entrenar con la imagen de entrada y el método X.

Para realizar el diagrama lo más general posible, se han omitido algunos atributos en el método ejecutarMetodo. Éste contendrá además de los atributos arriba mencionados, los atributos necesarios para ejecutar dicho método. Por ejemplo, Cuantización Vectorial, necesita un umbral, agrupamiento borroso no supervisado, el número de centros, etc.

Esta aplicación tiene dos funciones: entrenar y clasificar. Los diagramas de secuencia anteriores corresponde al entrenamiento de los distintos métodos. Ahora vamos a ver la secuencia generada cuando el usuario decide clasificar, en vez de entrenar.

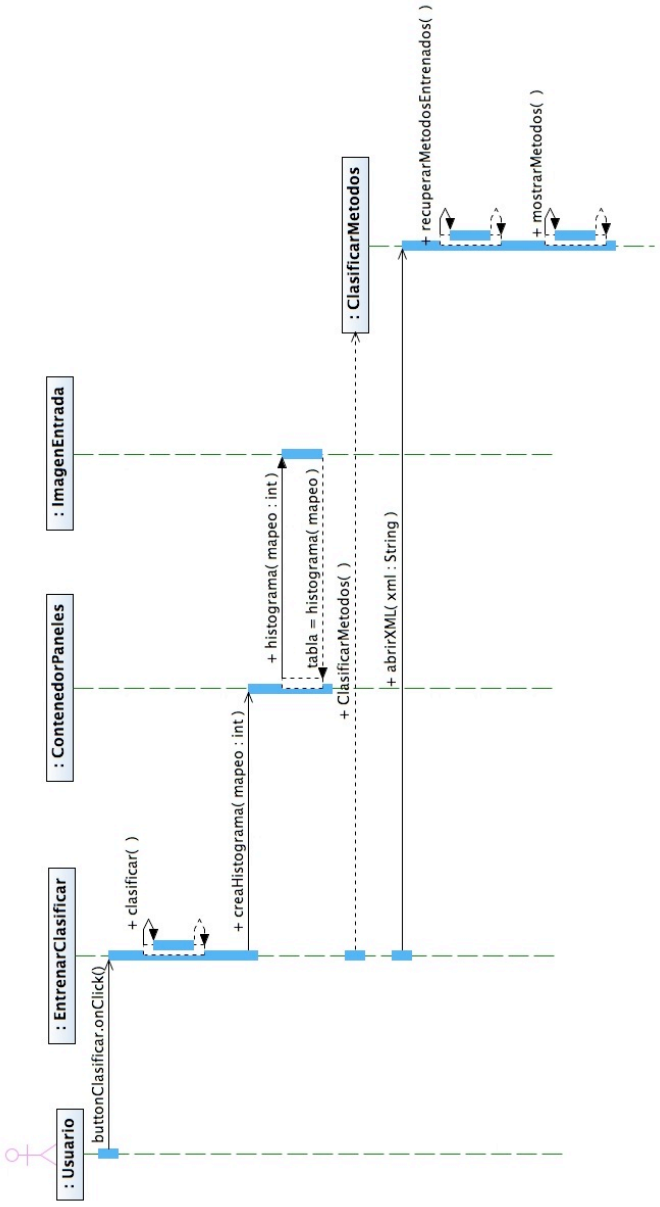


Figura 4.8.: Diagrama de secuencia: métodos ejecutados si el usuario decide clasificar

Para poder realizar una clasificación, debemos haber realizado anteriormente un entrenamiento y haber guardado el resultado del mismo en un documento XML. Cuando clasifiquemos, la aplicación leerá un documento XML, resultado de un entrenamiento y ofrecerá al usuario ejecutar aquellos métodos que hayan sido entrenados anteriormente. El usuario no podrá cambiar los parámetros del método, ya que se ejecutarán según como se hayan entrenado.

4.1.4.DIAGRAMAS DE ACTIVIDADES

En los diagramas de secuencia, hemos mostrado los métodos que se desencadenan al realizar acciones sobre la aplicación, pero no podemos mostrar, por ejemplo, en qué momento se puede realizar dicha acción. Para ello disponemos de los diagramas de actividades.

Con los diagramas de actividades mostramos el flujo secuencial de actividades llevadas a cabo por un usuario en un sistema.

El diagrama de la figura 4.9. muestra las actividades que se realizan cuando el usuario quiere que la aplicación clasifique imágenes.
Podemos observar, que el primer requisito para poder realizar la clasificación es seleccionar una imagen y un archivo XML resultado de un entrenamiento previo. Hasta que no se cumplan estos dos requisitos, el usuario no podrá pulsar el botón para clasificar.

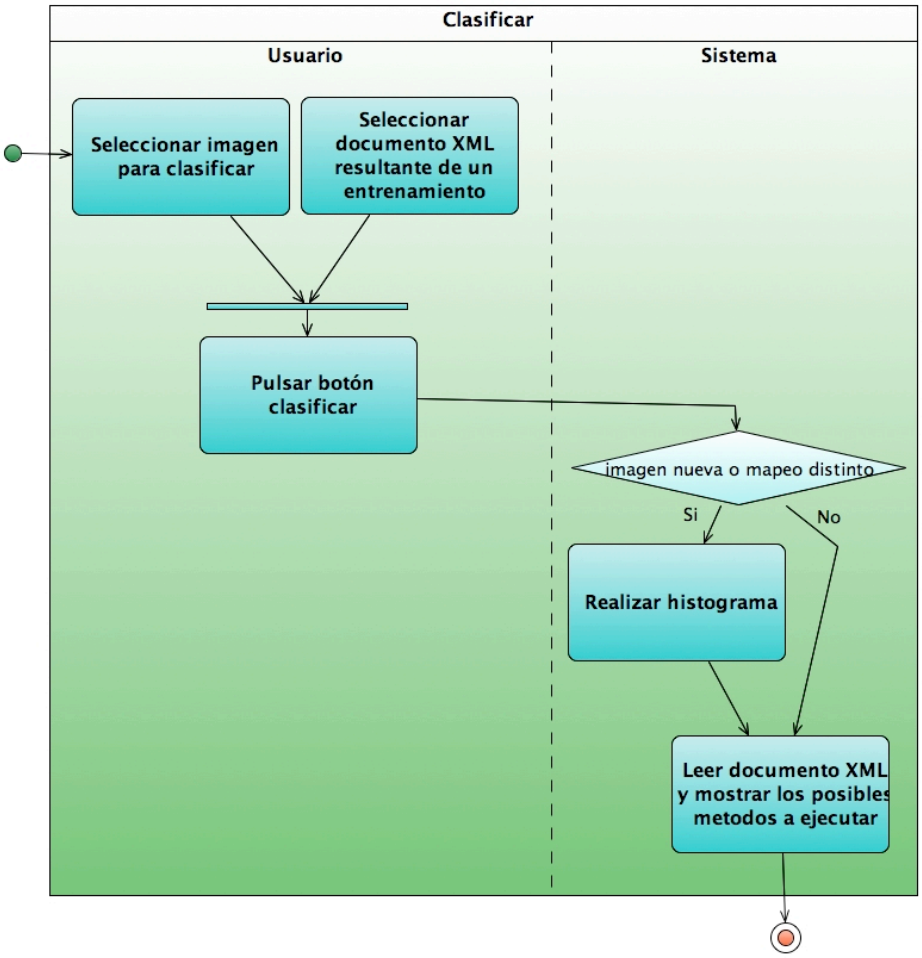


Figura 4.9.: Diagrama de actividad para empezar a clasificar.

4.2.RIESGOS

A continuación se exponen los diferentes riesgos asumidos durante el desarrollo de la aplicación, desglosándose en tecnológicos y de organización.

4.2.1.TECNOLÓGICOS

Riesgo	Descripción	Solución
Pérdida de datos	Pérdida de información, código o documentación.	Múltiples copias del trabajo realizado en distintos sistemas, USB, internet, etc.
Insuficiencia de recursos	No disponer de la tecnología suficiente para el desarrollo de la aplicación.	Realizar una investigación más exhaustiva de las tecnologías existentes.
Complejidad del código	El código realizado puede ser poco legible, poco comentado o muy complicado	Comentar todas las partes del código, explicando en todo momento qué se realiza en cada función.
Tiempo de ejecución	El tiempo de respuesta de un algoritmo puede ser excesivo y desagradable a vista del usuario	Realizar un estudio de cada algoritmo para reducir el tiempo de ejecución.

4.2.2.ORGANIZACIÓN

Riesgo	Descripción	Solución
Pérdida de datos	Pérdida de información, código o documentación debida a una mala gestión de los documentos	Múltiples copias del trabajo realizado en distintos sistemas, USB, internet, etc.

5. CONSTRUCCIÓN

5.1.INTRODUCCIÓN

El objetivo de esta fase consiste en la codificación de los componentes de la aplicación a partir de las especificaciones obtenidas en el proceso de diseño.

5.2.SISTEMA OPERATIVO

La aplicación ha sido desarrollada sobre el sistema operativo Windows XP. Dicho sistema ha sido instalado en una máquina virtual [4]. La máquina virtual utilizada es VirtualBox [5], de Sun. El sistema operativo que contiene dicha máquina virtual es Mac OS X.

5.3.ENTORNO DE DESARROLLO

Para la construcción de la aplicación, se ha elegido como entorno de desarrollo Eclipse [6], con SDK[7] (Software Development Kit). La versión de Java utilizada es JDK 6.13 con JavaFx 1.1[8]. Actualmente la versión del JDK ha sido actualizada a la versión 6.14 y JavaFx ha sido actualizada a la versión 1.2.

[4] http://es.wikipedia.org/wiki/Maquina_virtual

[5] http://www.virtualbox.org/

[6] http://www.eclipse.org/

[7] http://es.wikipedia.org/wiki/Software_development_kit

[8] http://java.sun.com/javase/downloads/index.jsp

5.4.CONSTRUCCIÓN DE LA APLICACIÓN

Como se ha detallado en el apartado de organización del proyecto, sección 2.2.3., la fase de construcción esta dividida en dos iteraciones.

En la primera iteración se ha desarrollado todo lo correspondiente al entrenamiento de los algoritmos, tanto la parte de métodos como la parte de interfaz gráfica, puesto que, el tiempo para la construcción ha sido muy ajustado, a medida que se creaban los algoritmos se iban probando. Uno de los principales problemas en esta iteración ha sido el manejo de las librerías JAI y JAMA y la elección de las estructuras de datos necesarias para realizar la aplicación.

La segunda iteración ha estado dedicada a la clasificación de los algoritmos, así como todo lo relativo a la entrada y salida de los archivos XML necesarios para poder realizar la clasificación partiendo de los resultados de un entrenamiento previo. La dificultad de esta iteración ha sido considerablemente menor que la anterior, puesto que los algoritmos para clasificar sólo son ligeramente distintos a los algoritmos de entrenamiento. Por lo que, la mayor parte del tiempo, se ha dedicado a la estructura de los documentos XML.

Para la implementación de los algoritmos se ha seguido la especificación obtenida en el apartado de análisis. La dificultad en la implementación no ha sido excesivamente elevada, dado que se realizó un estudio previo. La mayor dificultad se encuentra a la hora de verificar que los algoritmos implementados realizan exactamente lo que se deseaba desde un principio.

Para devolver los diferentes resultados, se opta por representar cada uno de ellos en pestañas distintas y se da al usuario la posibilidad de mostrar la información relevante, resultado del entrenamiento o clasificación de la imagen. Si la aplicación está entrenando, devolverá como resultado una imagen segmentada, cuyas clases están etiquetadas con los colores de los centros de dichas clases. Puede ser que los colores de los centros sean similares entre sí, por lo que se ofrece al usuario una interfaz para cambiar las etiquetas de las clases siendo representadas como éste decida. A la hora de realizar la clasificación el color de las etiquetas que el usuario a seleccionado serán conservados.

Los documentos XML generados siguen la siguiente estructura:

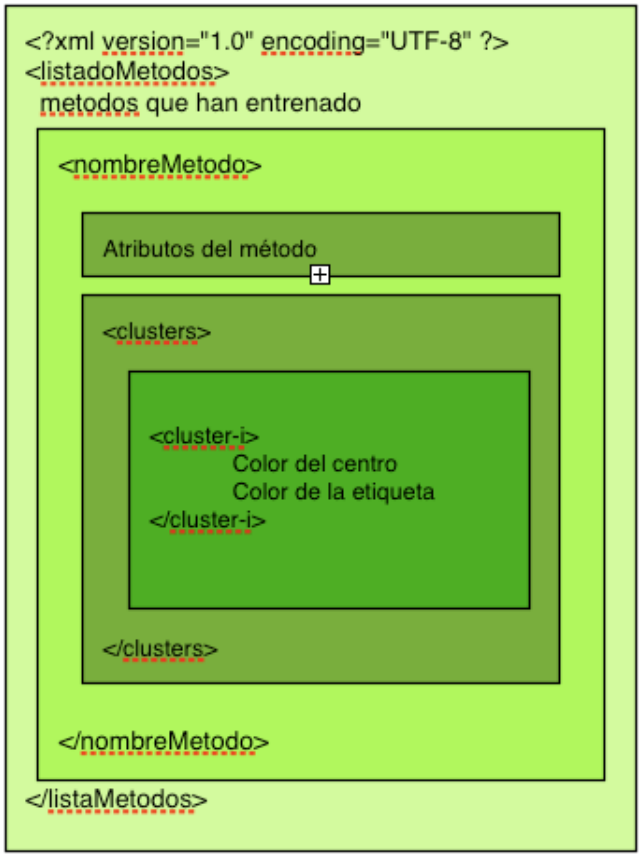


Figura 5.1.: Estructura de los documentos XML

Como se puede observar, entre distintos métodos, el XML generado es distinto. Es decir, por ejemplo, si comparamos los métodos Cuantización Vectorial y Lloyd, podemos observar que los atributos del método son distintos. Esto es posible, porque cada método está implementado en clases JAVA distintas, tal y como se puede apreciar en el diagrama de clases. Con esto conseguimos que cada clase genere y lea su parte del XML.

6. RESULTADOS DEL ENTRENAMIENTO

Pasamos ahora a comentar los resultados obtenidos durante el entrenamiento de los distintos métodos implementados.
Para ello, vamos a fijar una imagen de entrada y vamos a ir comentando cada uno de los métodos utilizados y el resultado generado.

Partimos de la siguiente imagen, pero para hacer el estudio, vamos a seleccionar un píxel de cada dos:

Figura 6.1.: Imagen de partida.

6.1.CUANTIZACIÓN VECTORIAL

Realizamos el estudio de la imagen de entrada con un umbral de 150.
El resultado obtenido son cinco clústeres y la siguiente imagen.

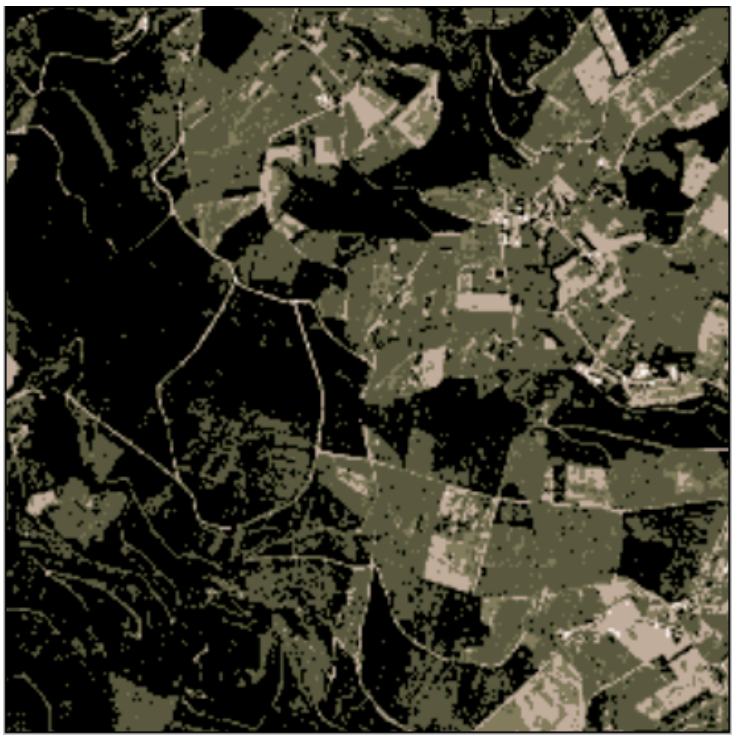


Figura 6.2. Resultado obtenido con cuantización vectorial
Umbral: 150

La interfaz de usuario es capaz de mostrar información relativa a la ejecución del algoritmo. La siguiente imagen proporciona la información resultante de esta ejecución:

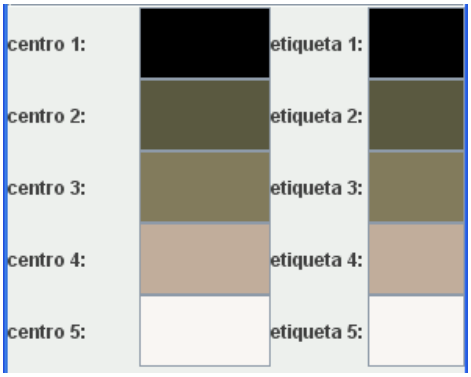


Figura 6.3.: Información del método cuantización vectorial

En la interfaz mostrada en la figura 6.3. podemos distinguir varias partes:

- Izquierda: Color de los centros. Representa el color real de los centros.
- Derecha: Color de las etiquetas. Colores mostrados en la imagen resultado.

Los centros antes mostrados, corresponden con la media de los colores de los píxeles que están en los centros, puesto que la imagen de entrada tiene colores parecidos los centros son similares, por lo que para apreciar mejor el resultado vamos a variar los colores de los centros. Este es el resultado:

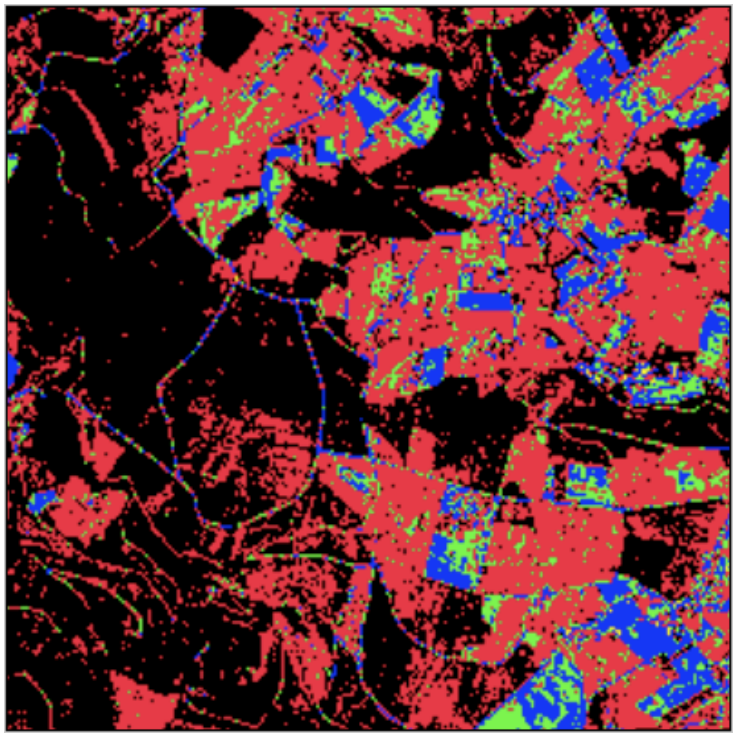


Figura 6.4.: Resultado obtenido con cuantización vectorial. Umbral: 150

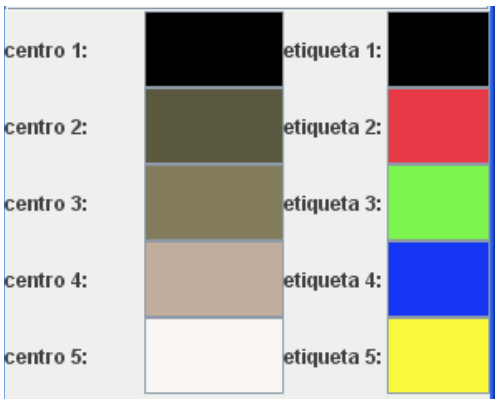


Figura 6.5.: Información del método

6.2.AGRUPAMIENTO BORROSO

6.2.1.CRISP

Realizamos dos estudios, en el primer estudio, utilizamos los clústeres obtenidos en el apartado anterior. En el segundo, utilizaremos clústeres sacados mediante el método agrupamiento borroso no supervisado.

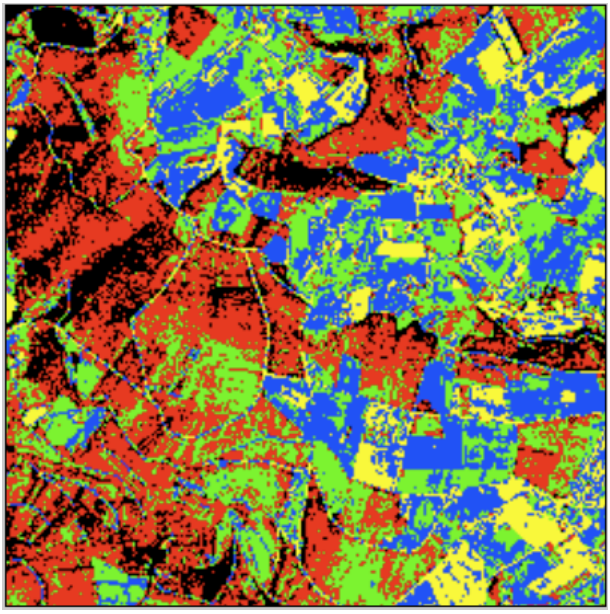


Figura 6.6.: Resultado obtenido con los centros calculados anteriormente con cuantización vectorial.

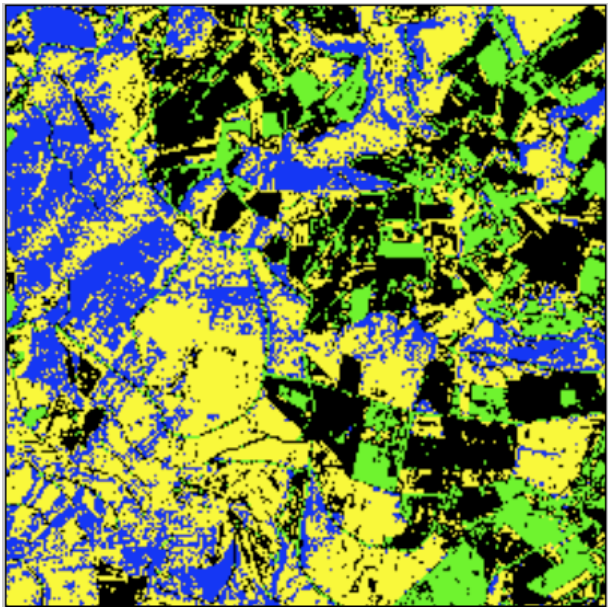


Figura 6.7.: Resultado obtenido con los centros calculados anteriormente con el método agrupamiento borroso no supervisado.

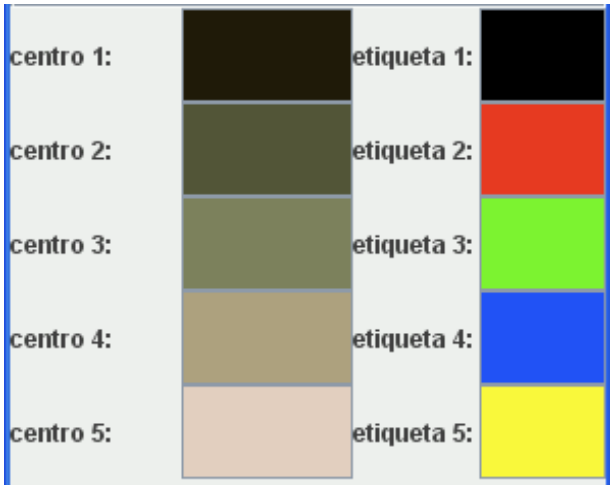


Figura 6.8.: Centros y etiquetas de los distintos centros correspondiente al resultado de arriba.

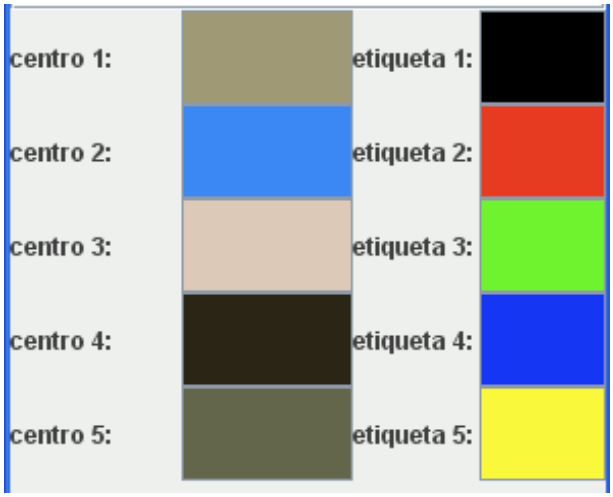


Figura 6.9.: Centros y etiquetas de los distintos centros correspondiente al resultado de arriba

Tenemos que realizar varias observaciones. Los dos métodos de partida, cuantización vectorial y agrupamiento borroso no supervisado, devolvieron cinco centros. Vemos entonces que en los resultados, la figura de la izquierda sigue conservando los cinco centros, pero la figura de la derecha sólo tiene cuatro. Esto es debido a que los centros de agrupamiento borroso no supervisado, por ser pseudoaleatorios, están muy alejados de los colores de la imagen, por lo que sólo aquellos centros más cercanos a estos colores progresan hasta encontrar centros realmente buenos.

6.2.2.FUZZY

Al igual que en el caso previo, se utilizará los mismos centros de entrada generados con los métodos de cuantización vectorial y agrupamiento borroso no supervisado.

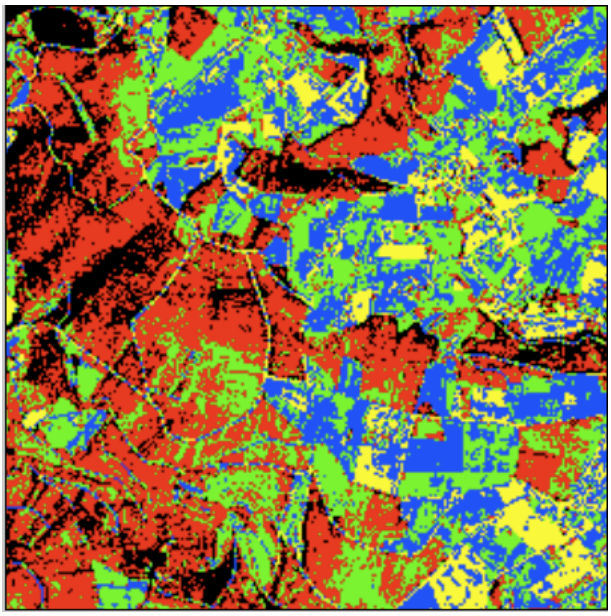


Figura 6.10.: Resultado obtenido con los centros calculados anteriormente con cuantización vectorial.

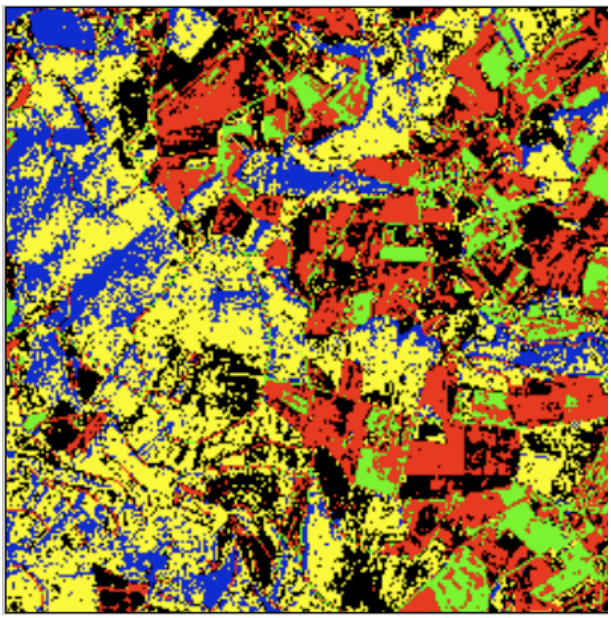


Figura 6.11.: Resultado obtenido con los centros calculados anteriormente con el método agrupamiento borroso no supervisado.

Para este estudio se ha usado la matriz Euclídea

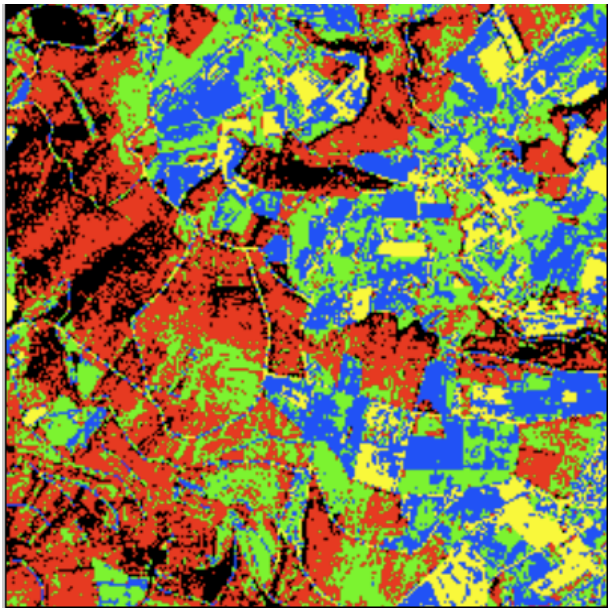


Figura 6.12.: Resultado obtenido con los centros calculados anteriormente con cuantización vectorial.

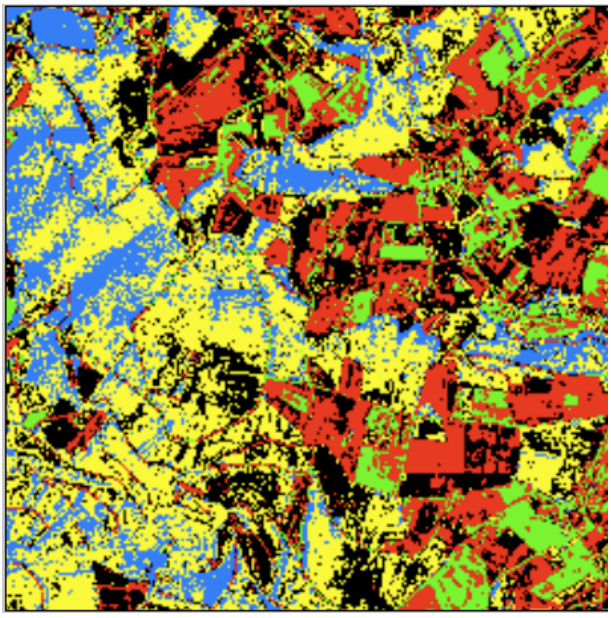


Figura 6.13.: Resultado obtenido con los centros calculados anteriormente con el método agrupamiento borroso no supervisado.

Para este estudio se ha usado la matriz diagonal

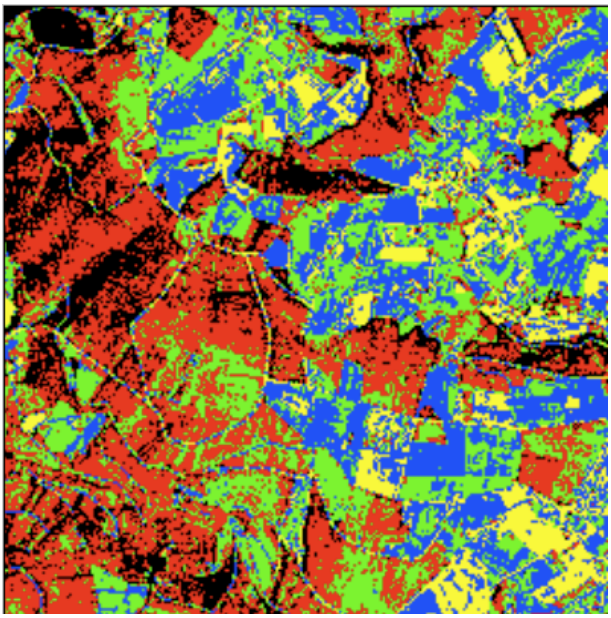


Figura 6.14.: Resultado obtenido con los centros calculados anteriormente con cuantización vectorial.

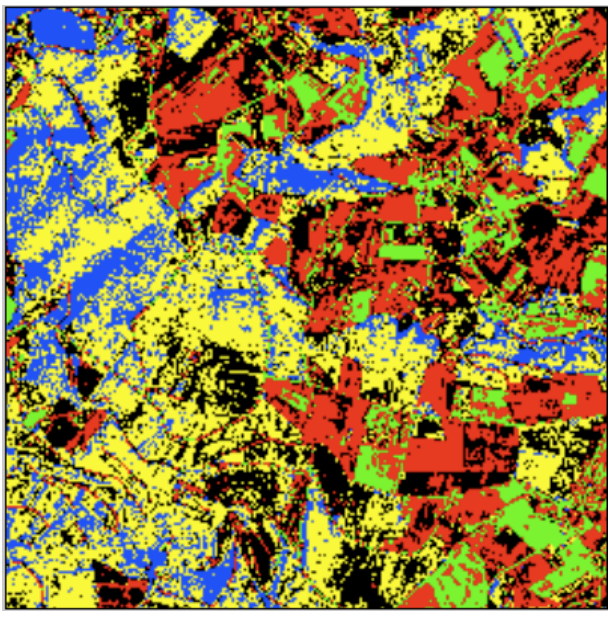


Figura 6.15.: Resultado obtenido con los centros calculados anteriormente con el método agrupamiento borroso no supervisado.

Para este estudio se ha usado la matriz Mahalanobis

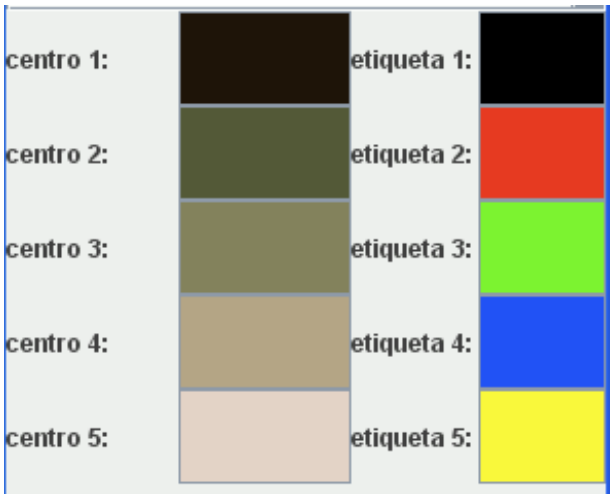


Figura 6.16.: Centros y etiquetas correspondientes a los resultados obtenidos a partir de los centros de cuantización vectorial. (Imágenes de la izquierda)

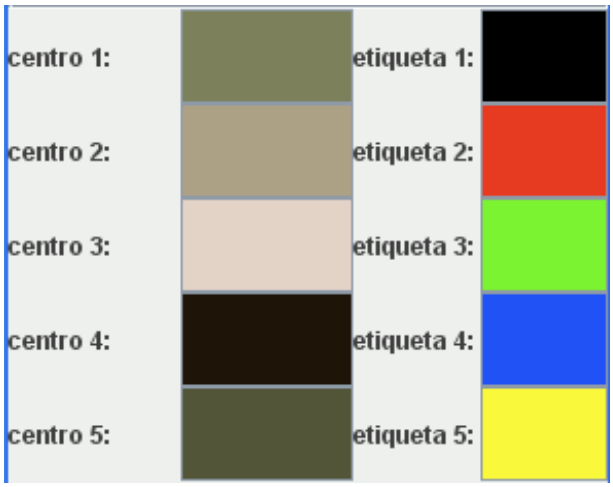


Figura 6.17.: Centros y etiquetas correspondientes a los resultados obtenidos a partid de los centros del agrupamiento borroso no supervisado. (Imágenes de la derecha)

En los seis estudios anteriores se ha partido de cinco centros calculados con los métodos mencionados al pie de imagen, peso exponencial 2 y error 0,05.

Observamos que en todos los casos, los resultados son muy parecidos, tendríamos que esforzarnos mucho para encontrar diferencias entre ellos.

Cabe destacar, si comparamos el método crisp con el método fuzzy, que en el método fuzzy sí se ha conseguido modificar los cinco centros de entrada para segmentar la imagen con cinco clústeres. En cambio, como hemos visto anteriormente, en el método crisp perdimos uno de los centros y como resultado teníamos una imagen segmentada en cuatro clústeres. Esto es debido a que en el método fuzzy, al contrario que en el crisp, todos los centros evolucionan constantemente, en cada iteración del método, dado que el grado de pertenencia de un color al clúster está comprendido en el intervalo [0,1]. En cambio, en el método crisp, si los centros de partida están muy alejados de los colores de los píxeles, sólo evolucionarán los más cercanos, dejando los más alejados como al principio.

6.3.ALGORITMO GENERALIZADO DE LLOYD

Se han realizado dos estudios con los resultados que se muestran en las figuras 6.18 a 6.21.

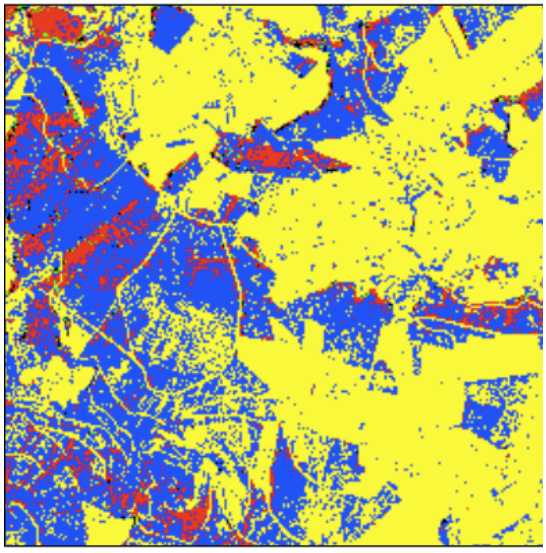


Figura 6.18.: Resultado obtenido con los centros calculados anteriormente con cuantización vectorial. Número de iteraciones: 5 Aprendizaje: 0,02

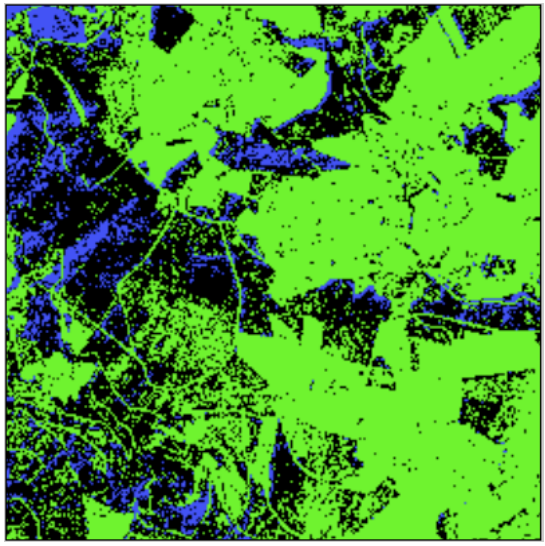


Figura 6.19.: Resultado obtenido con los centros calculados anteriormente con el método agrupamiento borroso no supervisado. Número de iteraciones: 5 Aprendizaje: 0,02

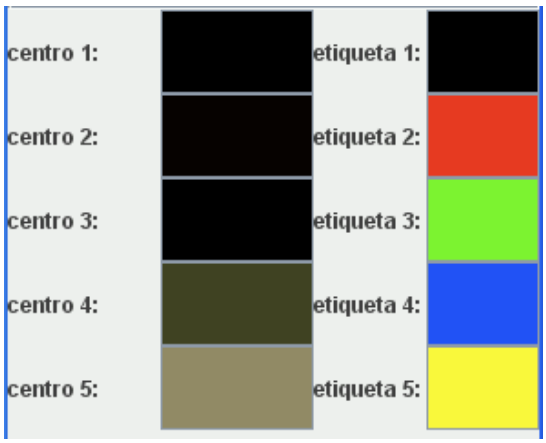


Figura 6.20.: Centros y etiquetas correspondientes a la imagen de arriba.

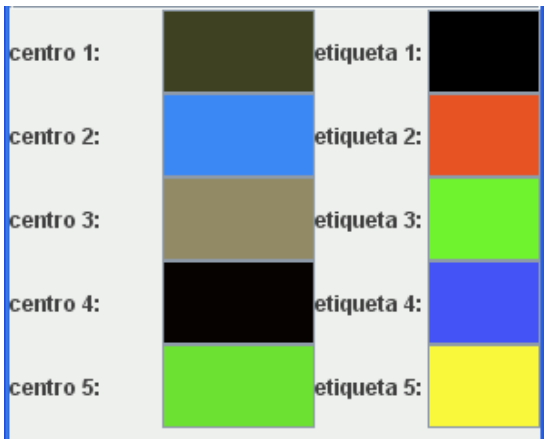


Figura 6.21.: Centros y etiquetas correspondientes a la imagen de arriba

Si observamos atentamente, vemos que en ambas figuras, no se han representado los cinco clústeres de los que partíamos. En la imagen de la izquierda tenemos cuatro y en imagen de la derecha tres. En la imagen de la izquierda, parece que a simple vista sólo hay tres centros, pero si nos fijamos bien, el color verde también está representado, si bien sólo con muy pocos

píxeles. La diferencia ha sido que en la imagen de la izquierda, dos centros han convergido a un único centro, mientras que en la derecha dos centros se han perdido. Este último caso es debido a que, al igual que en el agrupamiento borroso crisp, los centros perdidos están muy alejados de los colores de la imagen.

6.4.CLASIFICADOR BAYESIANO

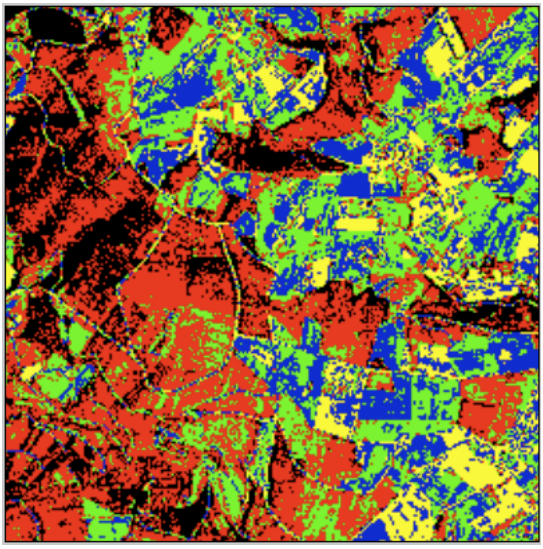


Figura 6.22.: Resultado obtenido a partir de los centros de cuantización vectorial

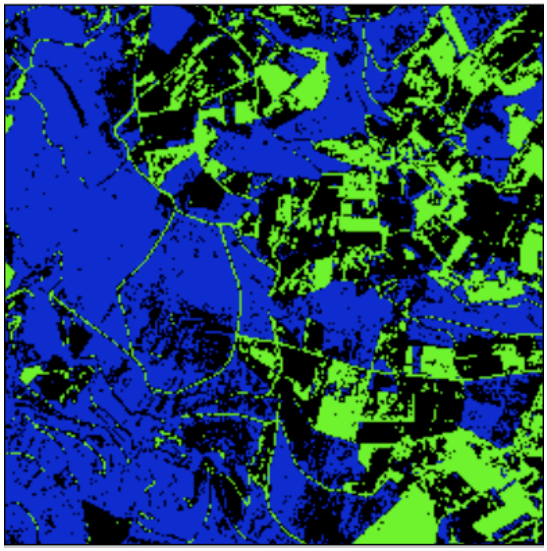


Figura 6.23.: Resultado obtenido a partir de los centros del agrupamiento borroso no supervisado

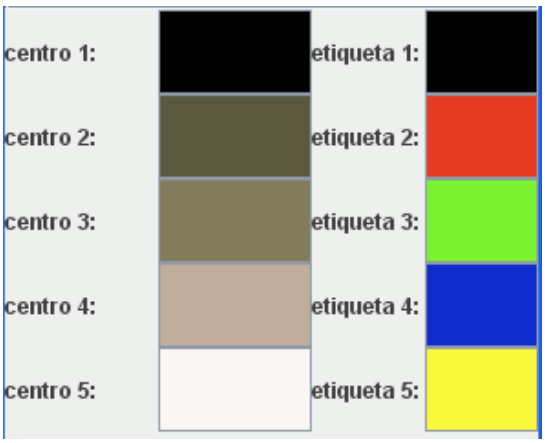


Figura 6.24.: Centros y etiquetas correspondientes con la imagen de arriba

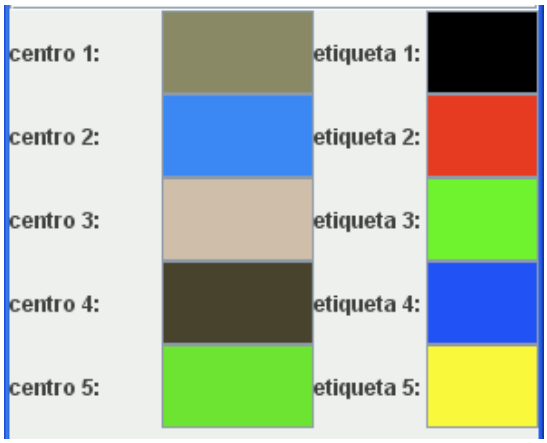


Figura 6.25.: Centros y etiquetas correspondientes con la imagen de arriba

Podemos observar que el resultado obtenido en la figura 6.23, sólo representa tres de los cinco clústeres iniciales, esto es debido a que los dos colores de los clústeres no representados en la imagen, están más alejados de los colores de los píxeles que los tres clústeres si representados.

7. RESULTADOS EN LA CLASIFICACIÓN

Una vez realizado un entrenamiento, podremos proceder a realizar la clasificación de imágenes. Para este apartado, se han seleccionado los métodos entrenados en el capítulo anterior, por lo que, los métodos ejecutados partirán de los centros aprendidos en el entrenamiento.

Figura 7.1.: Imagen a clasificar

Los únicos métodos que pueden realizar la clasificación son K-Medias, Lloyd y Bayes.
Dado que los centros obtenidos y las etiquetas ya están aprendidos y no se modifican, en este apartado no se muestra la interfaz gráfica que los representa, dado que se muestran en el capítulo anterior.

7.1.AGRUPAMIENTO BORROSO

Se ha reducido el tamaño del resultado obtenido, para poder realizar comparaciones entre un resultado y su adyacente.

7.1.1.CRISP

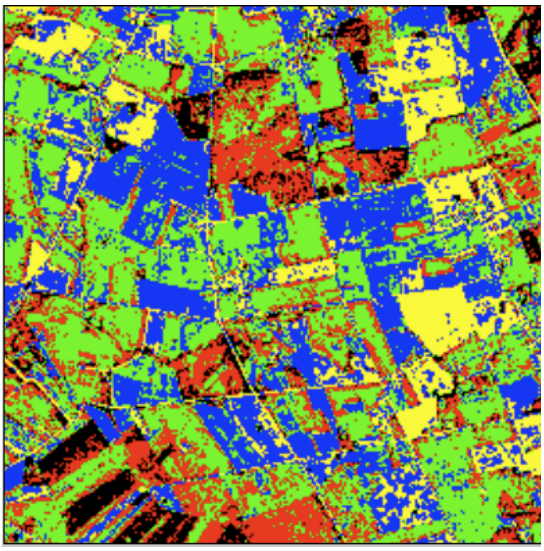


Figura 7.2.: Resultado obtenido a través del entrenamiento de la figura 6.6.

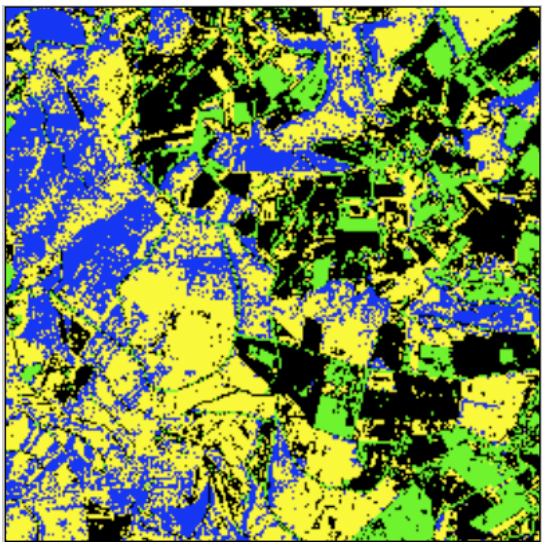


Figura 7.3.: Resultado obtenido a través del entrenamiento de la figura 6.7.

7.1.2.FUZZY

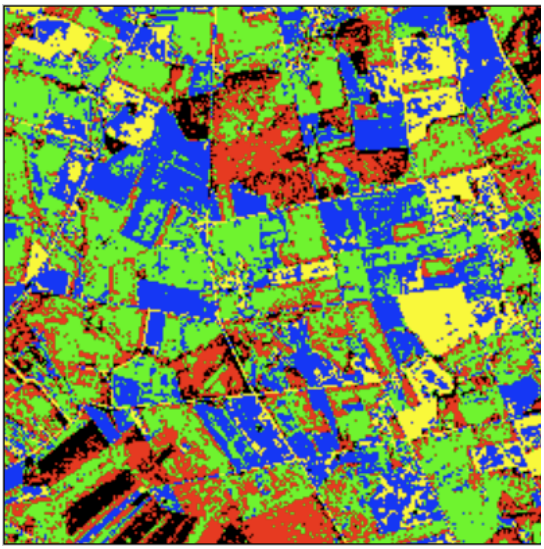


Figura 7.4.: Resultado obtenido a través del entrenamiento de la figura 6.10.

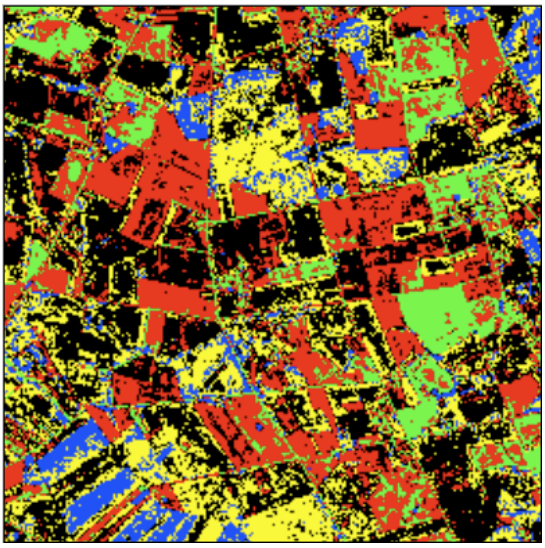


Figura 7.5.: Resultado obtenido a través del entrenamiento de la figura 6.11.

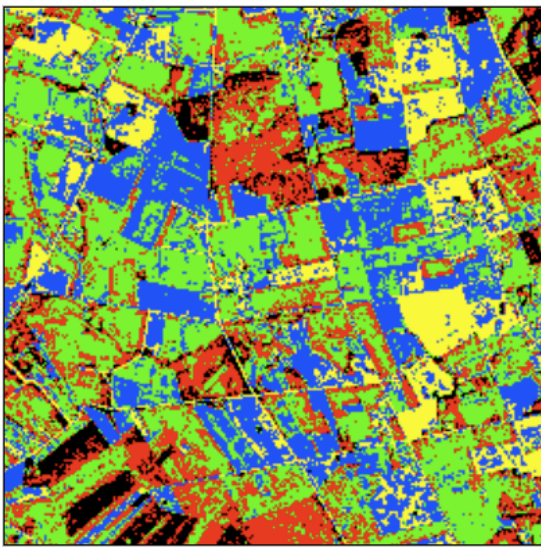


Figura 7.6.: Resultado obtenido a través del entrenamiento de la figura 6.12.

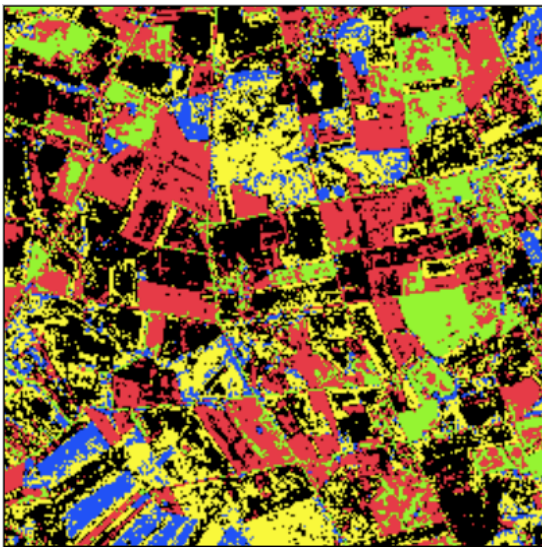


Figura 7.7.: Resultado obtenido a través del entrenamiento de la figura 6.13.

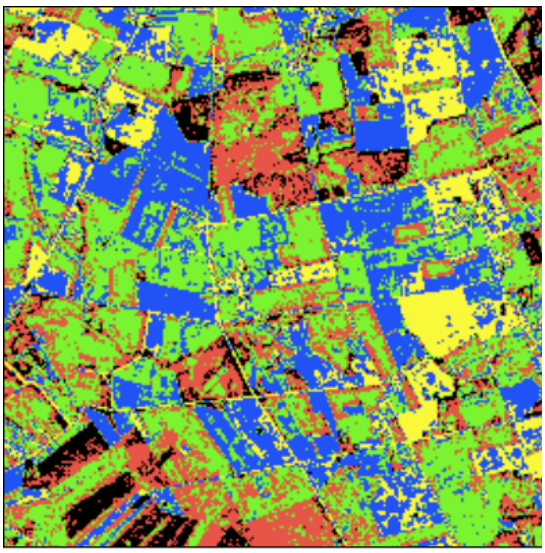


Figura 7.8.: Resultado obtenido a través del entrenamiento de la figura 6.14.

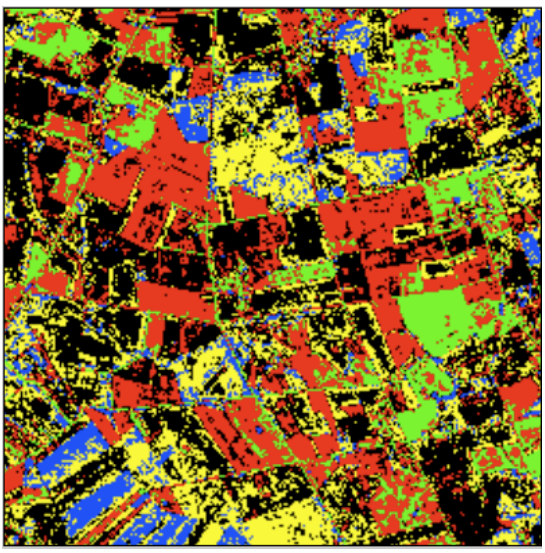


Figura 7.9. Resultado obtenido a través del entrenamiento de la figura 6.15.

7.2.ALGORITMO GENERALIZADO DE LLOYD

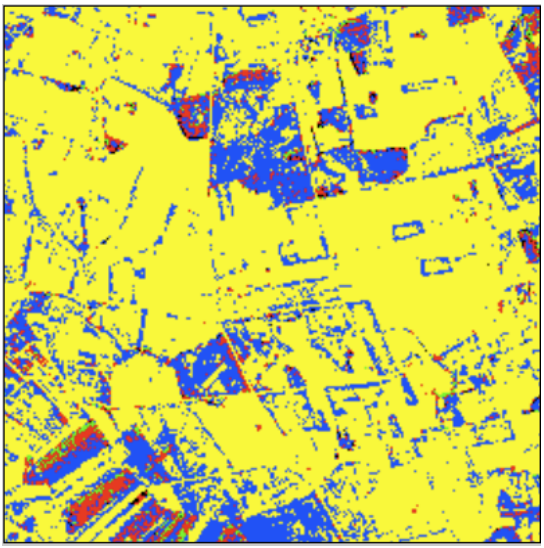


Figura 7.10.: Resultado obtenido a través del entrenamiento de la figura 6.18.

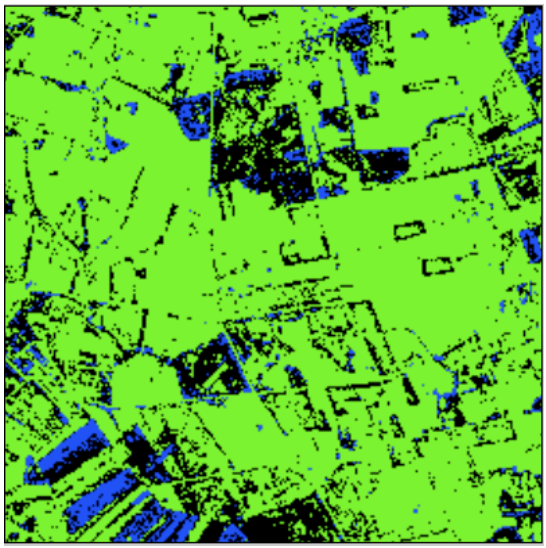


Figura 7.11.: Resultado obtenido a través del entrenamiento de la figura 6.19.

7.3.CLASIFICADOR BAYESIANO

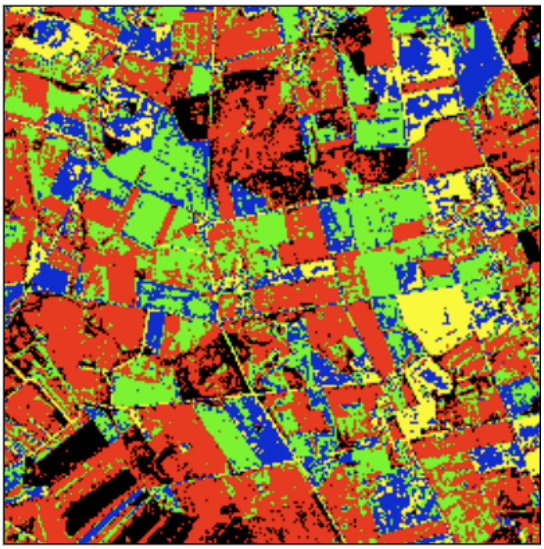


Figura 7.12.: Resultado obtenido a través del entrenamiento de la figura 6.22.

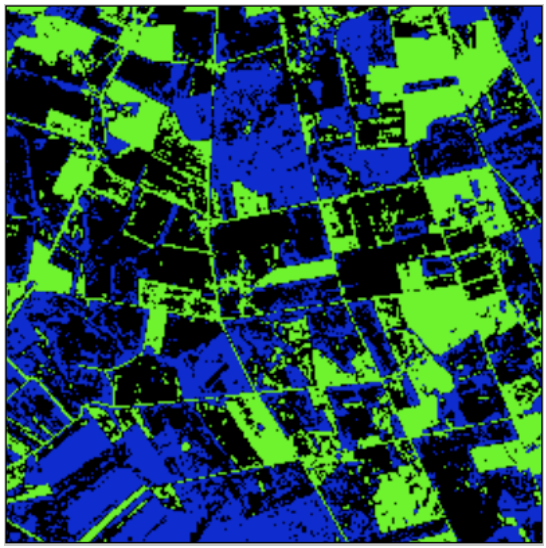


Figura 7.13.: Resultado obtenido a través del entrenamiento de la figura 6.23.

8. ESTUDIO DE TIEMPOS

8.1.COMPARACIÓN ENTRE MÉTODOS

Mostramos, a continuación, en la tabla 8.1. que asocia cada método con el tiempo de ejecución del mismo en milisegundos. Los métodos y tiempos siguientes corresponderían a los tiempos generados en el apartado "Resultados en el entrenamiento", por lo que todos los métodos segmentan la imagen en cinco clases. Los tiempos no incluyen el tiempo de etiquetado.

	Método	Figura	Tiempo (ms)
1	Cuantización Vectorial	6.2.	90
2	Agrupamiento borroso	---	76
3	K-Medias	6.6.	2333
4		6.7.	2243
5		6.10.	21110
6		6.11.	54148
7		6.12.	21359
8		6.13.	43533
9		6.14.	19842
10		6.15.	54891
11	Lloyd	6.18.	410
12		6.19.	370
13	Bayes	6.22.	492472
14		6.23.	751656

Tabla 8.1.: Tiempos de ejecución.

Se han numerado a la izquierda cada una de las filas para poder realizar comentarios.
Cabe destacar en K-Medias, la diferencia de tiempos entre los pares de filas 5-6, 7-8 y 9-10. Entre ellas, el estudio realizado es común, es decir, se basan en los mismos parámetros de entrada, la única diferencia son los centros de partida. Debido a ello, se produce este cambio

tan considerable en tiempo de ejecución. Los centros de las filas 5, 7 y 9 son el resultado del método de cuantización vectorial, mientras que los centros 6, 8 y 10 son el resultado del método agrupamiento borroso no supervisado.

Los centros relativos al método cuantización vectorial, ya están prácticamente ajustados a la imagen, sólo hace falta realizar algunos ajustes para conseguir un resultado que satisfaga que el error cometido es menor que el error establecido por el usuario. En cambio, los centros obtenidos con el método agrupamiento borroso no supervisado, por ser pseudoaleatorios, necesitan un mayor ajuste, por lo que conseguir que el error cometido sea menor que el error establecido por el usuario conlleva más tiempo, incluso, puede ser que al finalizar el algoritmo el error cometido sea mayor y este haya parado por el límite de iteraciones establecido. Por decisión de implementación, el límite de iteraciones es 400.

Podemos observar que en las filas 11 y 12, correspondientes con el algoritmo de Lloyd, el tiempo de ejecución es pequeño, esto es debido a que el número de iteraciones ha sido 5.

En cambio, el algoritmo de Bayes es el que más tarda con diferencia. Esto es debido a todos los cálculos con matrices necesarios, así como todas las comparaciones que se realizan dentro del método.

8.2.COMPARACIÓN ENTRE PARÁMETROS

8.2.1.CUANTIZACIÓN VECTORIAL

Realizamos un estudio variando el parámetro umbral del método cuantización vectorial.

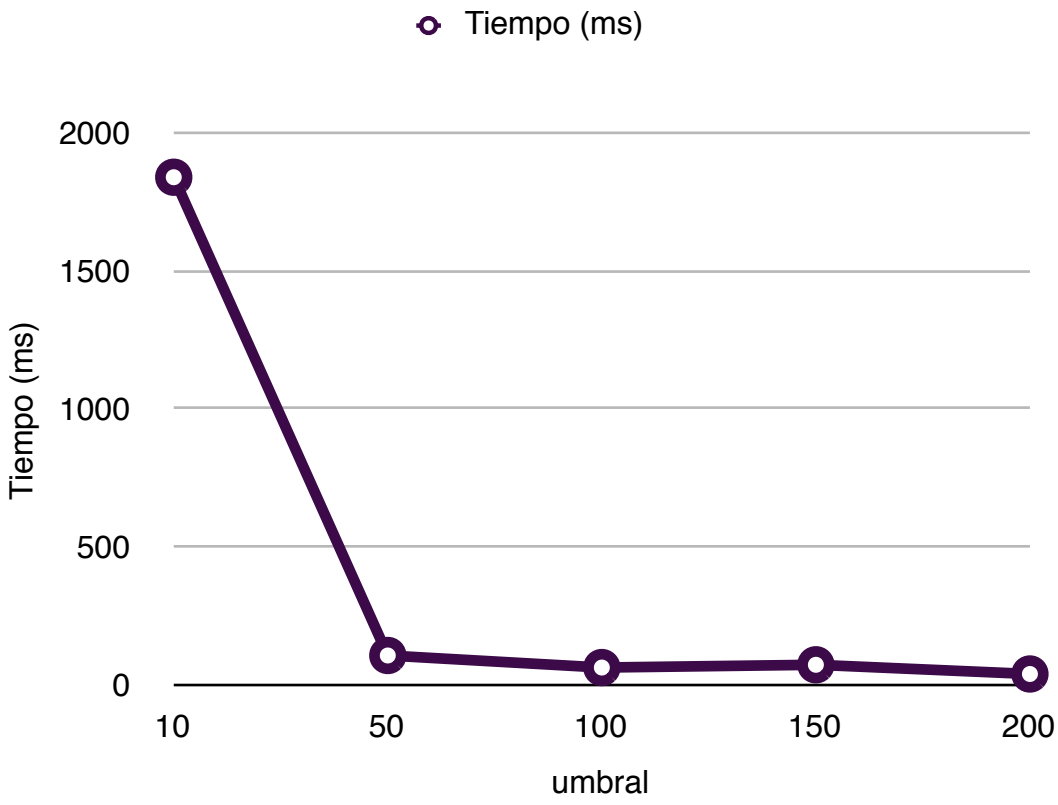


Figura 8.1.

En la figura 8.1., observamos que a medida que el umbral decrece el tiempo aumenta, esto es debido a que a menor umbral, el número de centros también aumenta, por lo que, el número de comparaciones consecuentemente también se incrementa y esto se traduce en más tiempo de computo.

8.2.2.K-MEDIAS FUZZY

Realizamos un estudio con los siguientes parámetros fijos del método K-Medias Fuzzy:

- Cinco centros iniciales, obtenidos con el método de cuantización vectorial y con umbral 150.
- Peso exponencial: 2
- Matriz G: identidad.

Variamos así el parámetro error.

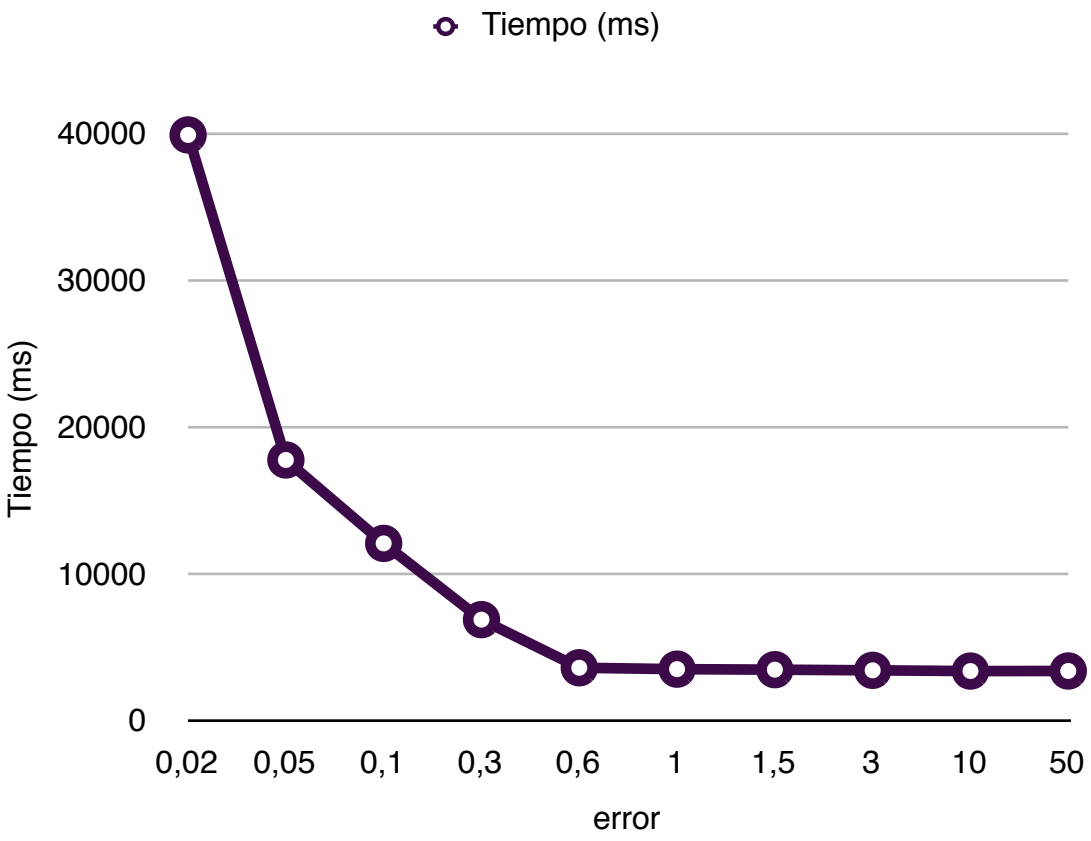


Figura 8.2.

A medida que el error se acerca a 0, el tiempo requerido para ejecutar el método aumenta. Esto ocurre, porque los centros están en constante cambio, por lo que encontrar soluciones con muy poco error es más difícil y lleva más tiempo.

8.2.3.LLOYD

Comprobamos como varía el tiempo en función al número de iteraciones del algoritmo.

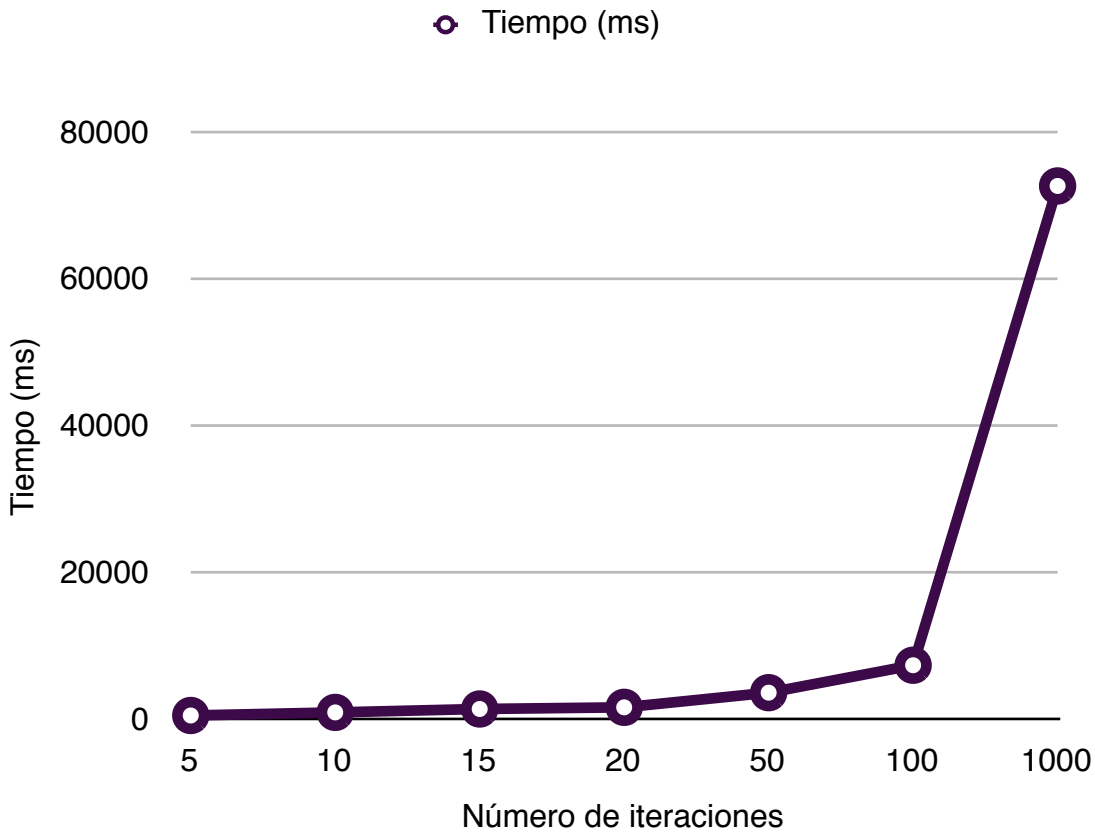


Figura 8.3.

Evidentemente, a medida que el número de iteraciones aumenta, el tiempo también lo hace.

8.2.4.BAYES

Variamos el número de centros iniciales. Para ello, partimos de los centros obtenidos a la hora de realizar el estudio de la sección 8.2.1.

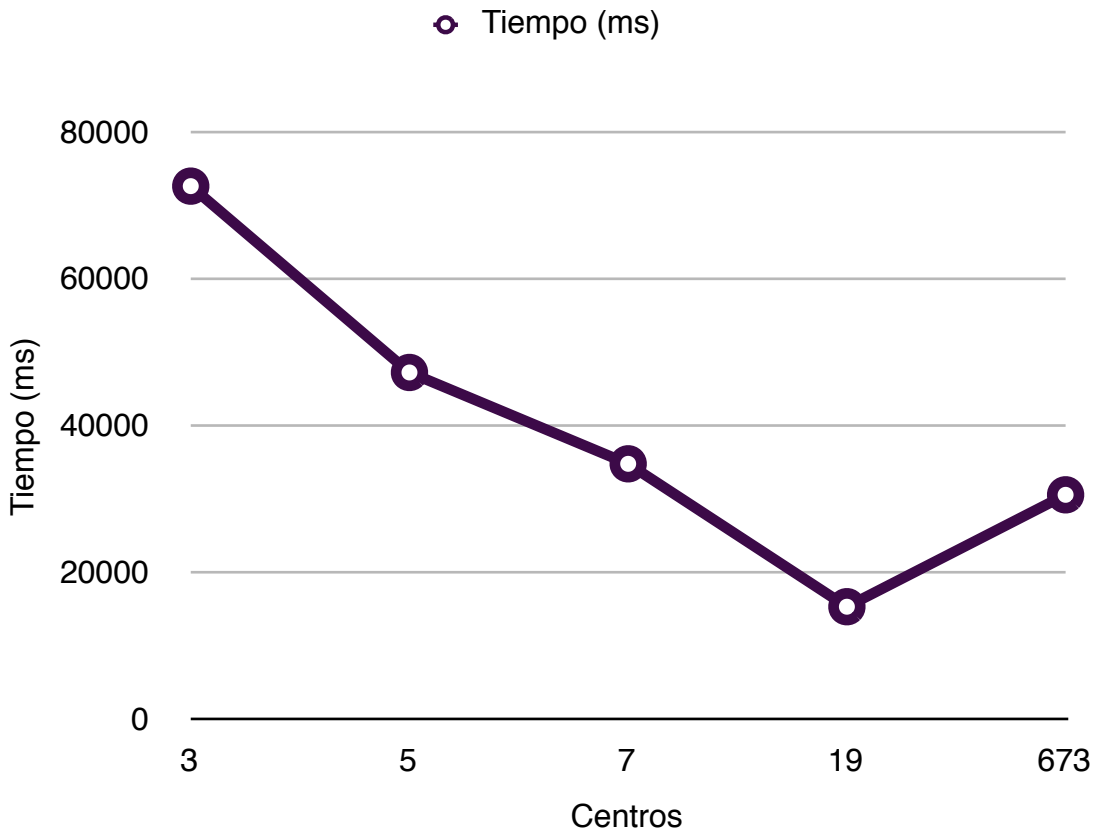

Tiempo (ms)
80000
60000
40000
20000
0
Tiempo (ms)
3
5
7
19
673
Centros

Figura 8.4.

9. PRUEBAS

Dividimos las pruebas realizadas en tres apartados.

9.1.PRUEBAS DE LOS ALGORITMOS

Comprobamos que cada uno de los algoritmos se ajusta a la especificación realizada anteriormente.

Para ello, se aplican varios ejemplos sencillos para poder realizar una comprobación manual, comparando así el resultado de nuestro algoritmo con los resultados obtenidos al realizar la comprobación manual. Por lo tanto, se crearon imágenes, no reales, de varios colores.

Posteriormente, para imágenes reales, se realizaron comparaciones con otros programas ya existentes, implementados en MATLAB.

Se realizó una comprobación de la correcta interacción entre los distintos objetos de nuestro sistema.

Estas comprobaciones tuvieron como consecuencia la modificación de algunos de los algoritmos, pero estas modificaciones no afectaron a otras clases, por lo que la modificación de los algoritmos no conllevó demasiados problemas.

9.2.PRUEBAS DE LA INTERFAZ

Comprobamos que la interfaz gráfica cumple con todas las funcionalidades especificadas.

Aseguramos que todos los componentes funcionan correctamente. Entre estas funcionalidades se debe cumplir que, todos los botones ofrezcan la ayuda necesaria, diciendo qué acción se desencadena al pulsarse. Comprobamos que no nos deje realizar operaciones no permitidas, por ejemplo, pulsar el botón para clasificar, sin haber cargado anteriormente una imagen y un documento XML, etc.

Probamos que las imágenes se cargan correctamente en cada una de las pestañas, así como, tenemos que observar que la información de la imagen es la correcta y que el cambio de etiquetas se realiza correctamente.

Observamos la correcta interacción entre los objetos, para ello, debemos comprobar que cumple con los requisitos ya explicados anteriormente, como por ejemplo, que cada interfaz gráfica sólo se crea una vez.

Estas comprobaciones tuvieron como consecuencia el estado final de la interfaz, añadiendo consejos a los botones y algunas funciones olvidadas a la hora de implementar.

9.3.PRUEBAS FINALES

Una vez realizadas todas las mejoras y modificaciones, pasamos a verificar la aplicación con las pruebas finales. Para ello, utilizamos distintas imágenes y comprobamos que todos los algoritmos ofrecen resultados parecidos.

Realizamos una comparación entre los resultados obtenidos con la aplicación y los requisitos iniciales del sistema y aseguramos que se cumplen todos los requisitos.

10.FUTURO

La elaboración y el diseño de la aplicación ha sido desarrollada pensando en posibles ampliaciones o mejoras del proyecto.

Algunas de las posibles ampliaciones son las siguientes:

- Nuevos algoritmos de segmentación: gracias al diseño de la aplicación, añadir un nuevo algoritmo de segmentación es sencillo, bastando crear dos clases, una para el paquete GUI y otra para el paquete MétodosAC.
- Mejora de los algoritmos de segmentación implementados.
- Añadir funcionalidades a la aplicación, por ejemplo, mostrar la vista previa de la imagen a abrir, etc.

También se podrían realizar otras ampliaciones, pero conllevarían más modificaciones:

- Reconocimiento de bordes.
- Reconocimiento de objetos.
- Añadir ontologías, por ejemplo, para el reconocimiento de escenarios u objetos.

11.TUTORIAL

Este apartado tiene como objetivo el aprender a utilizar la aplicación desarrollada.

Abrimos la aplicación y tendremos la siguiente vista:

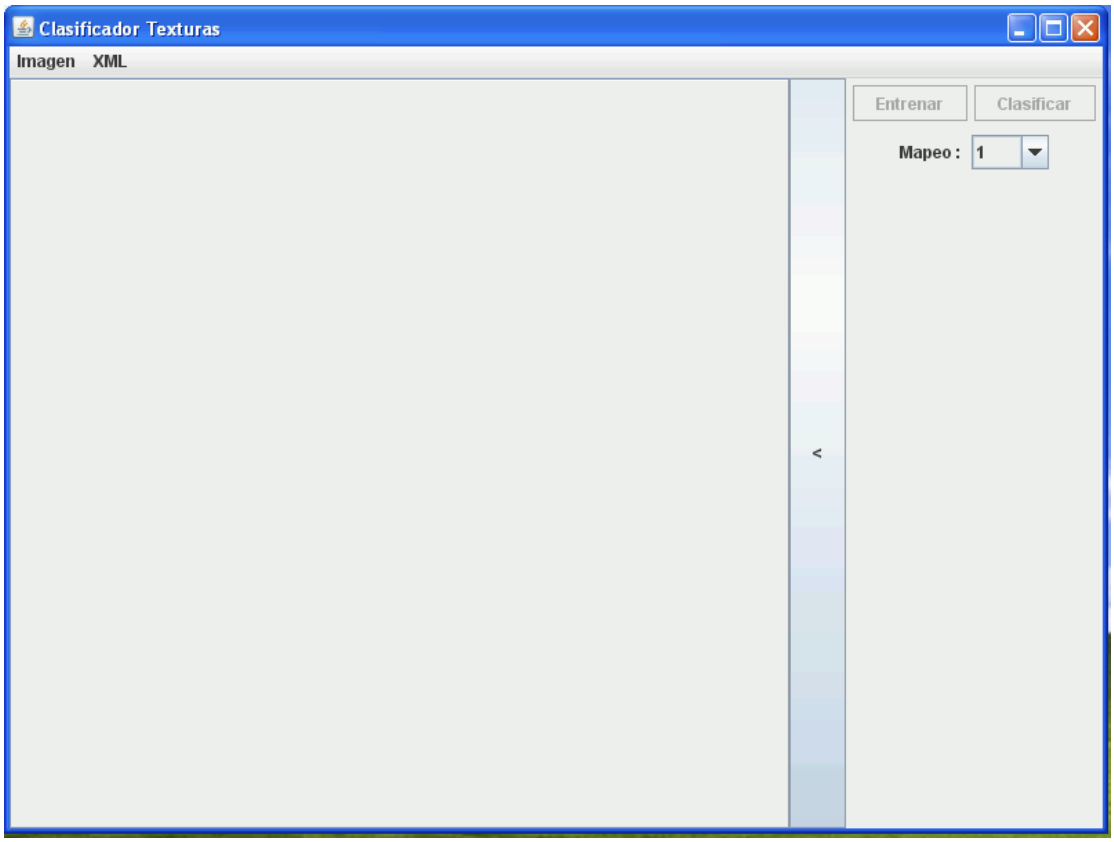


Figura 11.1.: Vista inicial de la aplicación.

Vemos que no podemos ni Entrenar ni Clasificar, esto es debido a que aún no hemos abierto ninguna imagen. Para ello, pulsamos Imagen en la barra superior.

Figura 11.2.: Menú para abrir una imagen

Pulsamos ahora en Abrir imagen y tendremos el siguiente menú emergente, donde seleccionaremos la imagen deseada.

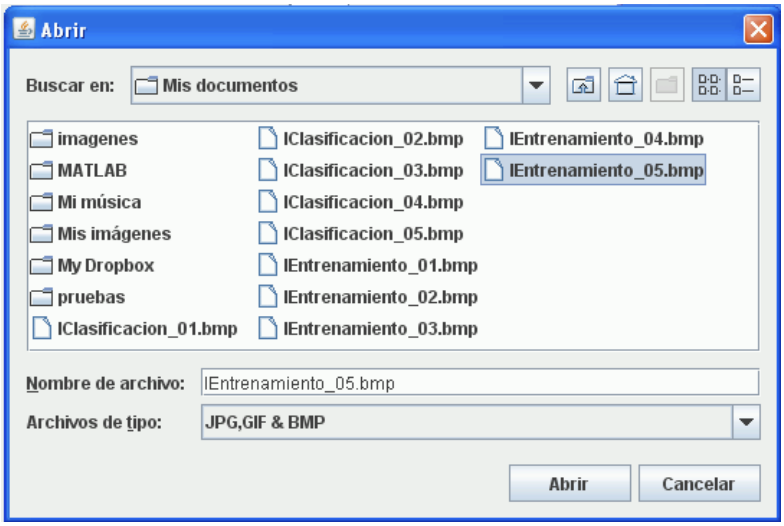


Figura 11.3.: Menú emergente para seleccionar una imagen para realizar un estudio

Seleccionamos una imagen, en este caso IEntrenamiento_05.bmp y pulsamos Abrir.
Vemos que la vista inicial ha cambiado y que ya podemos Entrenar.

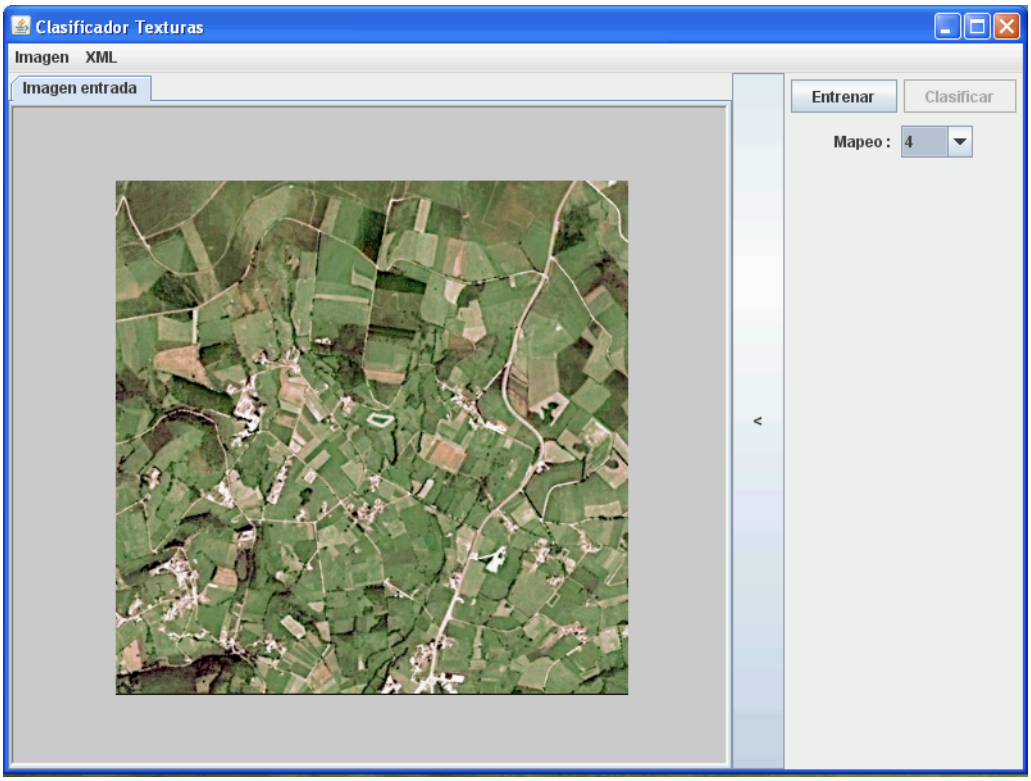


Figura 11.4.: Vista inicial con la imagen seleccionada

Para comenzar el entrenamiento, vamos a modificar la variable de mapeo y pondremos 4. El significado de esta variable ya ha sido explicado anteriormente, en la sección 4.1.3.
Pulsamos Entrenar y pasamos a la siguiente vista.

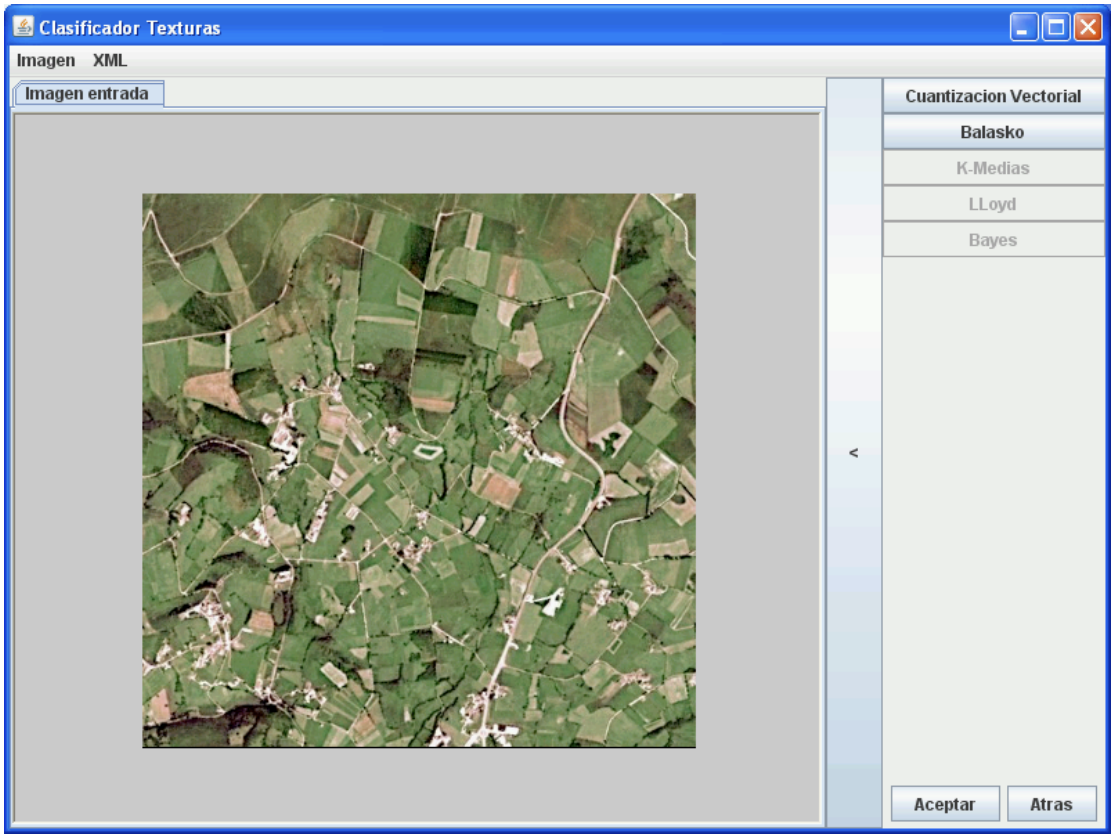


Figura 11.5.: Vista para la selección de métodos

Observemos que los únicos métodos que podemos entrenar son Cuantización Vectorial y Balasko. Esto es debido a que los demás métodos, K-Medias, Lloyd y Bayes, necesitan de unos centros de partida, por lo que no podremos ejecutar estos métodos hasta no disponer de dichos centros de partida.
Para conseguir estos centros, tenemos dos formas, entrenar alguno de los métodos disponibles, o abrir un documento XML con el resultado de un entrenamiento anterior. A continuación mostraremos como entrenar todos los métodos y posteriormente enseñaremos como abrir un XML.

Al pulsar sobre cualquier botón correspondiente a los métodos implementados, la interfaz gráfica se modificará mostrando los parámetros necesarios para ejecutar dicho procedimiento.

Pulsamos en Cuantización Vectorial.

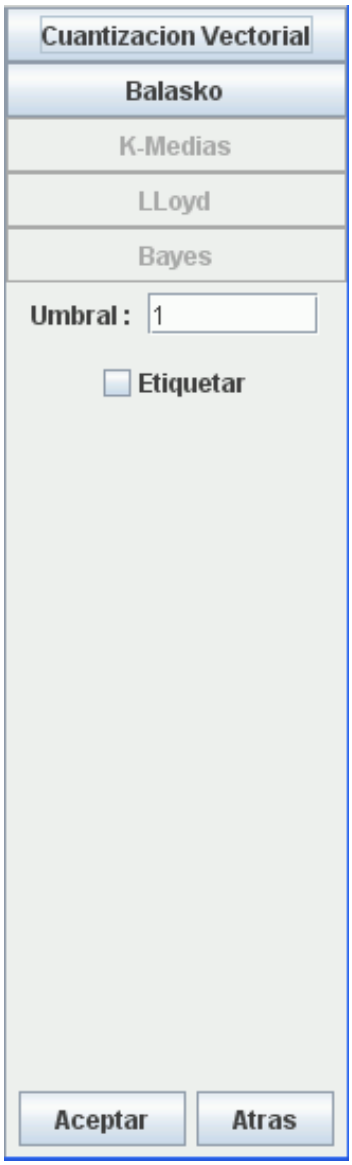


Figura 11.6.: Cuantización Vectorial

Con esta interfaz podremos modificar los parámetros necesarios. Estos parámetros ya han sido explicados anteriormente, en las sección 3.3.1.

El cuadrado Etiquetar sirve para obtener la representación resultado de realizar este entrenamiento.

Una vez hayamos seleccionado los parámetros necesarios, podemos pulsar Aceptar, para que el sistema realice el entrenamiento.

Una vez hayamos entrenado, la vista cambiará y podremos entrenar cualquiera de los métodos. Pulsamos el botón Balasko.

Escribimos el número de centros y pulsamos Aceptar.

Figura 11.7.: Balasko

Pulsamos el botón K-Medias.

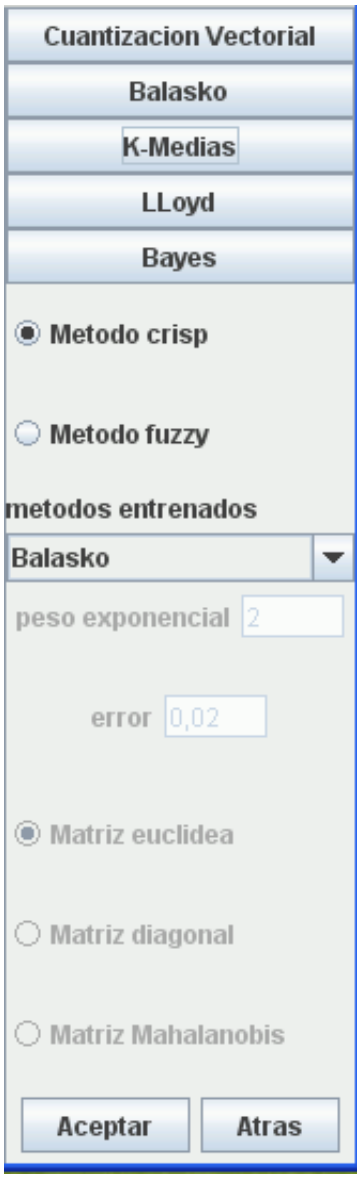


Figura 11.8.: K-Medias Crisp

Tenemos una de las dos siguientes vistas, dependiendo de si elegimos para entrenar el Método crisp o el Método fuzzy.

En ambas imágenes, observamos que tenemos una pestaña de nombre "métodos entrenados", con esta pestaña seleccionaremos los centros entrenados anteriormente, bajo el nombre del método con el que los hemos calculado, en este caso, está seleccionado Balasko.

En la imagen de la derecha podemos modificar todos los parámetros correspondiente a este método.

Podemos modificar los parámetros, peso exponencial, error y la matriz G, mencionados en parámetros anteriores.

Pulsamos aceptar para comenzar el entrenamiento.

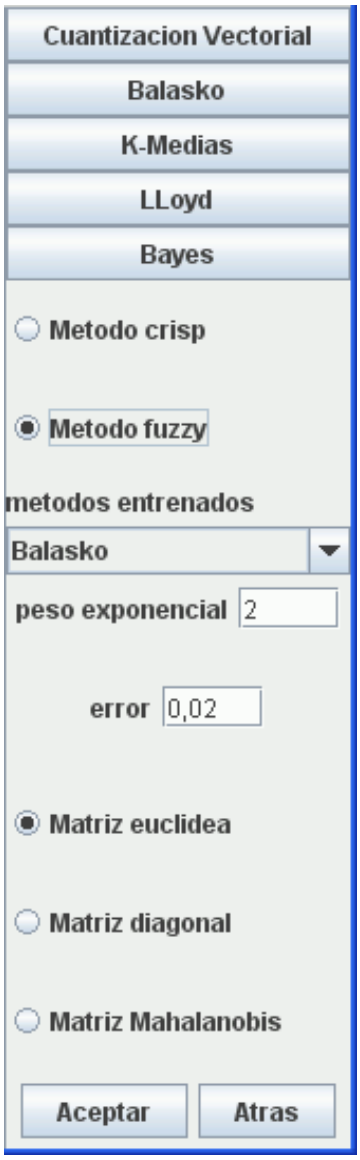


Figura 11.9.: K-Medias Fuzzy

Pulsamos el botón Lloyd.

Podemos modificar los parámetros necesarios para la ejecución del método, número de iteraciones, aprendizaje y los centros de partida, bajo el nombre del método que los ha calculado.
Pulsamos Aceptar para comenzar con el entrenamiento.

Figura 11.10.: Lloyd

Pulsamos el botón Bayes.

En este método únicamente tendremos que seleccionar los centros de partida.
Pulsamos aceptar para comenzar con el entrenamiento.

Figura 11.11.: Bayes

Vamos a visualizar ahora los resultados y la información correspondiente. Para este tutorial, mostramos el resultado de K-Medias Fuzzy partiendo de dos centros obtenidos con el método de agrupamiento borroso.

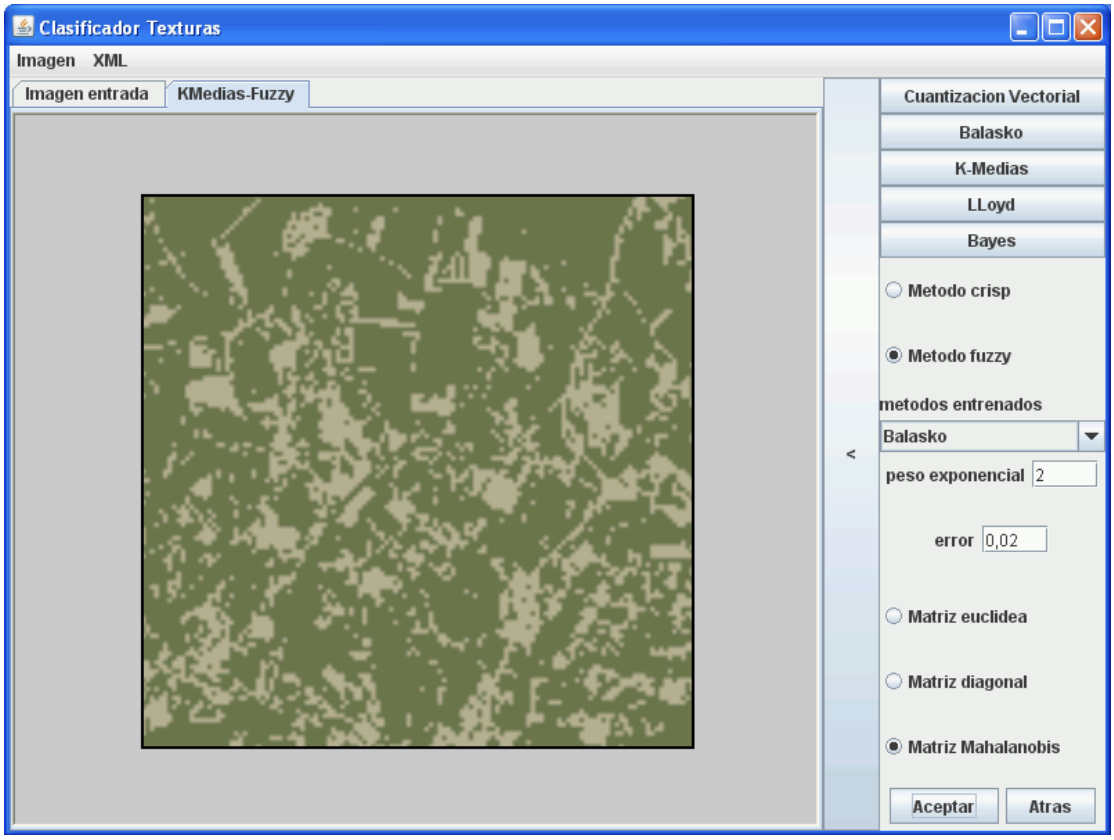


Figura 11.12.: Resultado obtenido con K-Medias Fuzzy

Vemos que se ha creado una nueva pestaña, así podremos navegar entre las distintas imágenes creadas. Para mostrar la información de la imagen pulsamos el botón "<". Este botón cambiará si la información está visible o no. Si la información está visible, el botón será ">" y si la información esta oculta "<".

Pulsamos el botón "<" para mostrar la información.

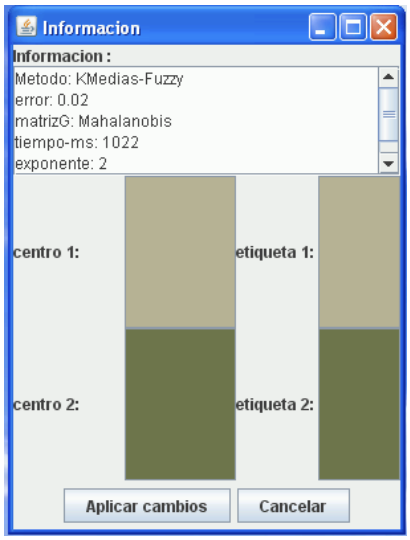


Figura 11.13.: Información de la imagen

La información de la imagen la podemos dividir en tres zonas:

- Arriba: Mostramos la información obtenidas al realizar el entrenamiento. Nombre del método, nombre de los parámetros y valor de éstos y el tiempo en milisegundos.
- Abajo izquierda: Color de los centros en los que se ha segmentado la imagen.
- Abajo derecha: Etiqueta de los centros. Color que representa a los centros en la imagen.

El color de las etiquetas puede ser modificado, para ello sólo tenemos que pulsar sobre el color de la etiqueta elegida, apareciendo el siguiente menú emergente.

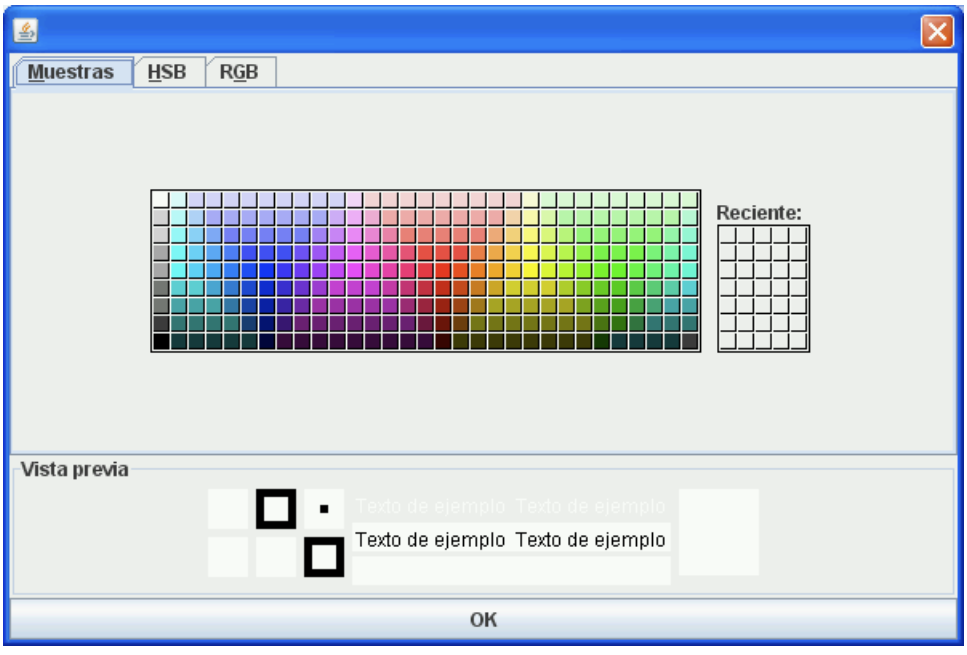


Figura 11.14.: Menú emergente para seleccionar el color de la etiqueta.

Una vez seleccionado un color, el color de la etiqueta correspondiente se modificará.

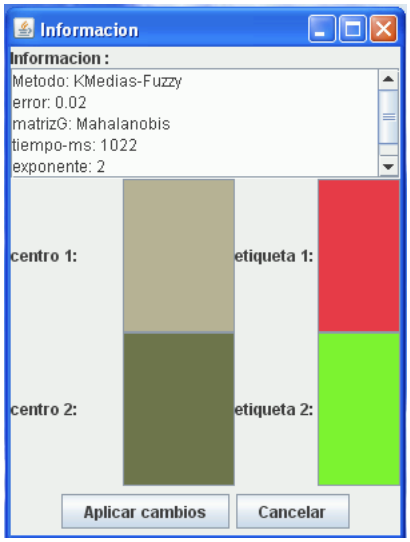


Figura 11.15.: Información de la imagen

Para aceptar los cambios, pulsamos en Aplicar cambios. La imagen volverá a ser pintada con los colores seleccionados.

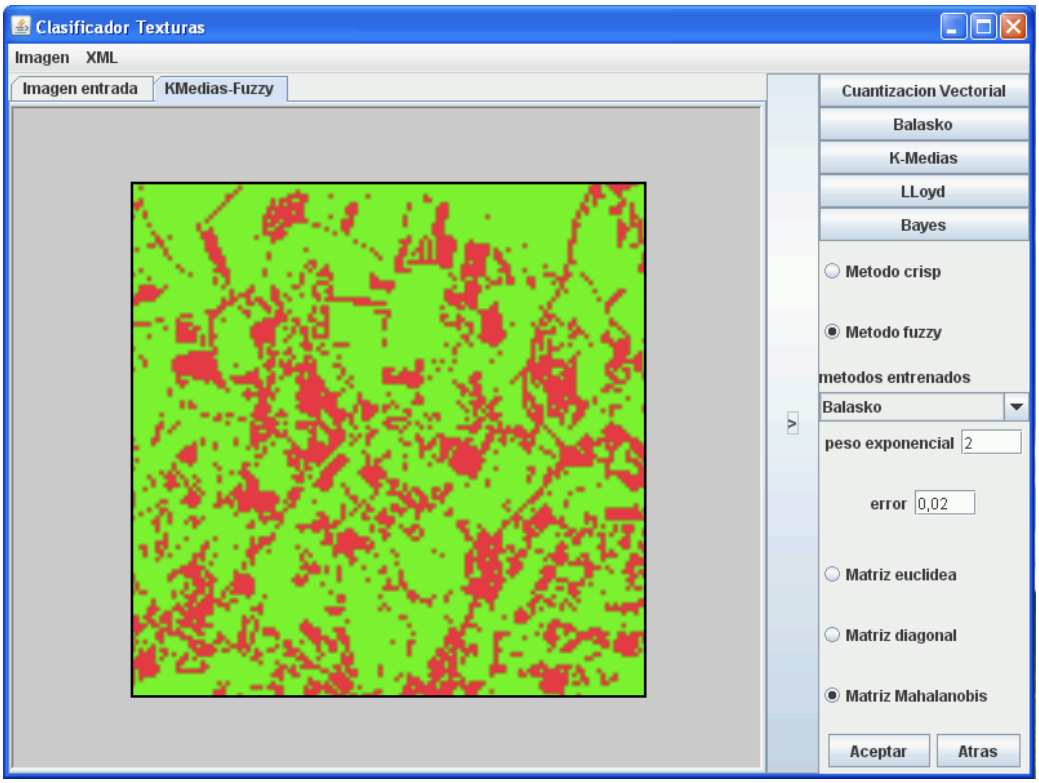


Figura 11.16.: Resultado final

Guardamos ahora el resultado del entrenamiento. Para ello, pulsamos, en el menú superior, sobre XML y posteriormente Guardar

Figura 11.17.: Menú para guardar el resultado

Escribimos el nombre del fichero en el menú emergente.

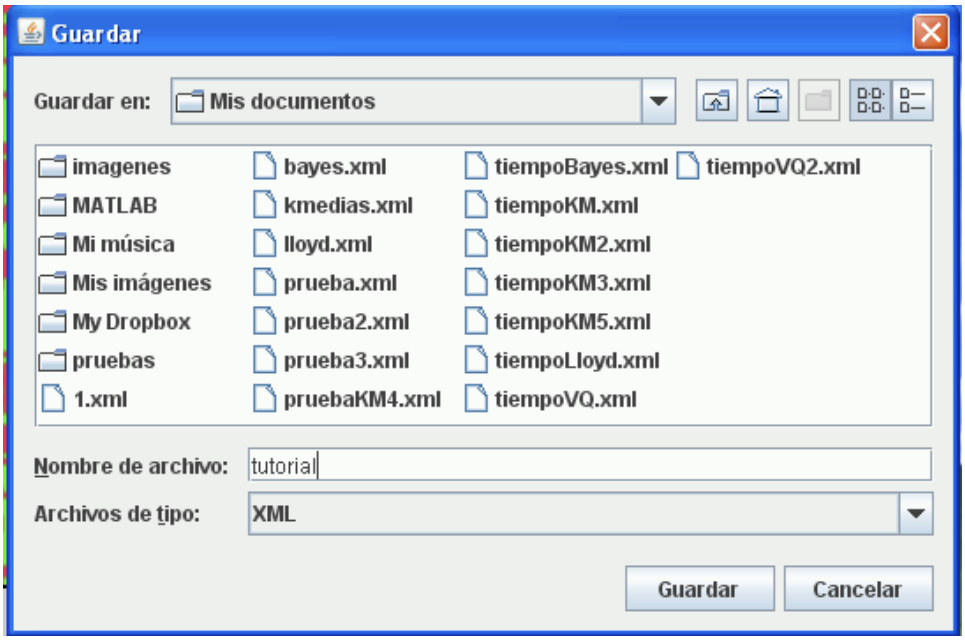


Figura 11.18.: Menú emergente para guardar el documento XML.

Pulsamos Guardar.

Hemos aprendido a utilizar la parte de entrenamiento. Pasamos ahora a la parte de clasificación.
Para ello, volvemos a abrir una imagen, como en el entrenamiento.

Para poder clasificar, debemos abrir un documento XML con el resultado de un entrenamiento previo.
Para abrir dicho documento, pulsamos, en el menú superior, XML y posteriormente Abrir XML

Figura 11.19.: Menú para abrir un documento XML

Seleccionamos el documento anterior, "tutorial.xml" y lo abrimos.
Observamos que el botón Clasificar ya se puede pulsar. Vamos a poner el ratón sobre dicho botón y a esperar tres segundos. Una vez hayan pasado los tres segundos aparecerá información sobre la acción que desarrolla el botón.

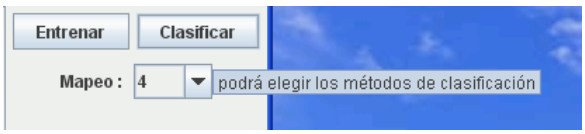


Figura 11.20.: Botones entrenar y clasificar, con información sobre la acción que desarrolla el botón clasificar.

Al igual que antes, dejamos la variable Mapeo en 4 y pulsamos Clasificar.

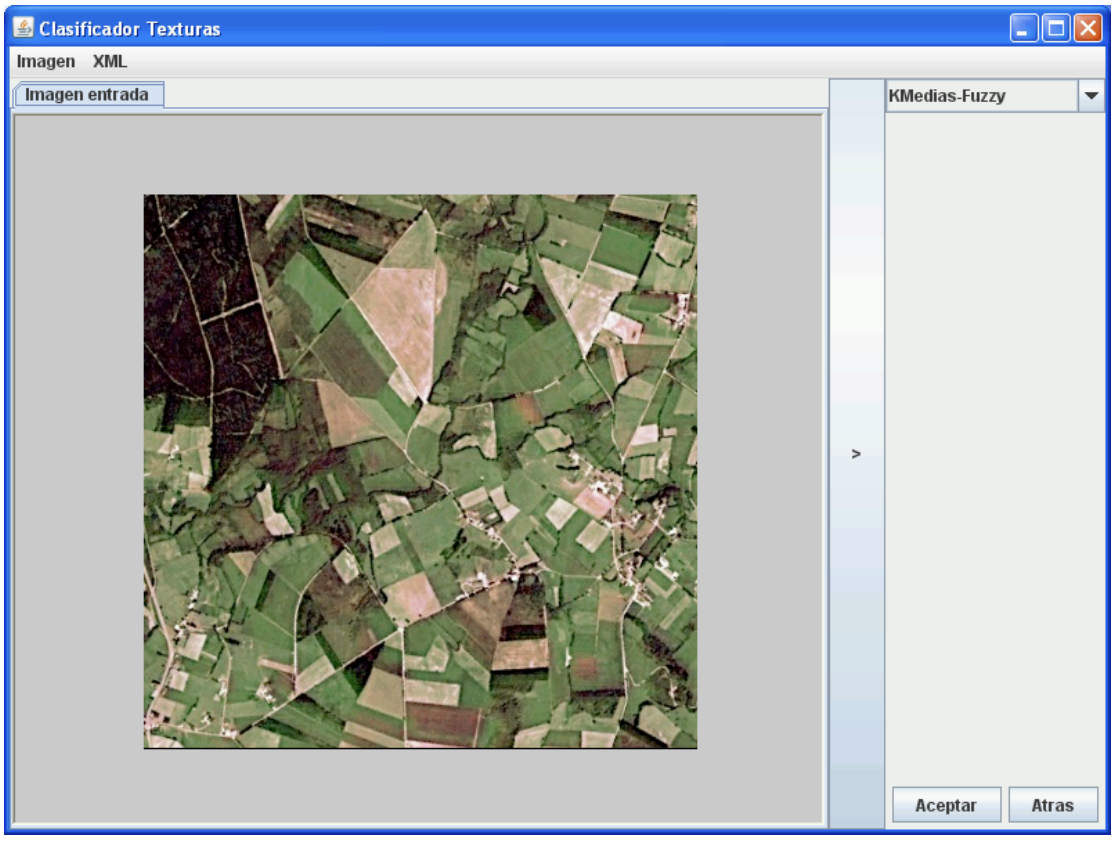


Figura 11.21.: Imagen para clasificar

Para realizar la clasificación, seleccionaremos el método anteriormente y pulsaremos Aceptar para realizar la segmentación.

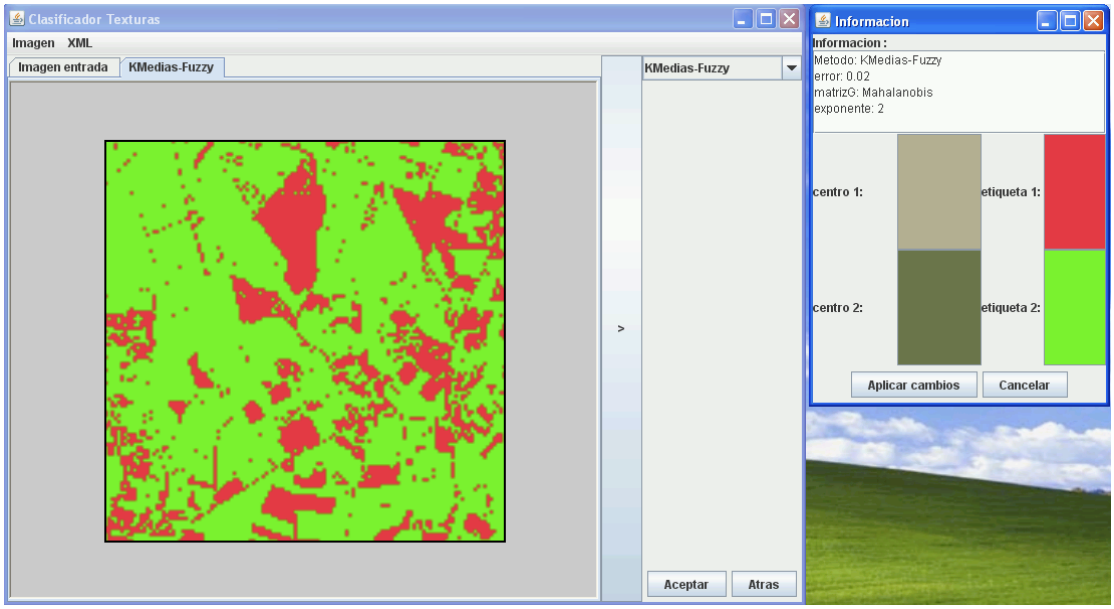
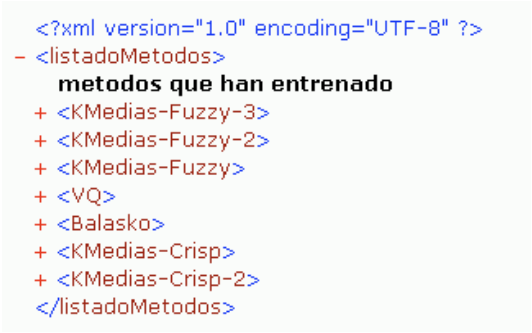


Figura 11.22.: Resultado obtenido al realizar la clasificación.

Si en cualquier momento pulsáramos el botón Atrás, volveríamos a la pantalla que tiene los botones Entrar y Clasificar.

Mostramos a continuación un documento XML generado. Dado que estos documentos XML poseen un tamaño considerable, sólo mostraremos algunas partes relevantes.

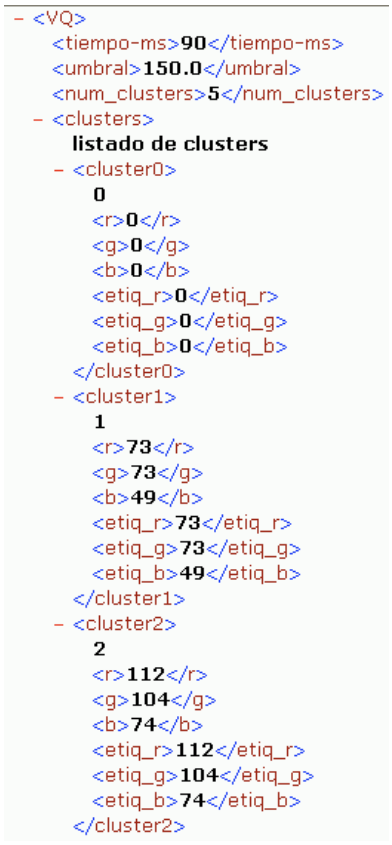


Bajo la etiqueta listadoMetodos, encontramos todos los métodos ejecutados. Si un método es ejecutado varias veces, recibirá como nombre:

nombre_metodo-numero

Figura 11.23.: Esquema XML general

El siguiente ejemplo, muestra la descripción de un método ejecutado.



La información guardada es la siguiente:

tiempo en milisegundos, correspondiente con el tiempo tardado en la ejecución del método.

parámetros necesarios del método.

número de centros.

listado de centros. Dentro de este listado, encontramos los centros que ha devuelto el sistema. Su estructura es la siguiente:

- Número del clúster
- Color del clúster
- Etiqueta del clúster

Figura 11.24.: Estructura XML del método Cuantización Vectorial

12.Bibliografía

1. "Control de versiones," http://es.wikipedia.org/wiki/Control_de_versiones.
2. "Subversion," http://es.wikipedia.org/wiki/Subversion.
3. "Mercurial," http://mercurial.selenic.com/wiki/.
4. "Git," http://git-scm.com/.
5. "DropBox," http://www.getdropbox.com.
6. Sun, "Java Advanced Imaging," http://java.sun.com/javase/technologies/desktop/media/.
7. Sun, "Java Document Object Model," http://www.jdom.org/.
8. Sun, "Java Matrix," http://math.nist.gov/javanumerics/jama/.
9. P. Stevens, *Using UML: Software Engineering with objects and components.*
10. Sun, "Javadoc Tool," http://java.sun.com/j2se/javadoc/.
11. G. Pajares and J.M. de la Cruz, *Visión por Computador: Imágenes digitales y Aplicaciones*, RA-MA, 2007.
12. G. Pajares and J.M. de la Cruz, *Ejercicios Resueltos de Visión por Computador*, RA-MA, 2007.
13. R.O. DUDA, P.E. Hart and D.G. Stork, *Pattern classification*, Willey, 2001.