Proyecto de Sistemas Informáticos

Curso académico
2008 / 2009

CLASIFICACIÓN DE TEXTURAS NATURALES MEDIANTE TÉCNICAS DE VISIÓN POR COMPUTADOR PARA APLICACIONES EN AGRICULTURA DE PRECISIÓN

Autores

Diego López Pozueta

Luis Miguel Ruiz Gil

Profesor Director

Gonzalo Pajares Martinsanz

Dpto. de Ingeniería del Software e Inteligencia Artificial

Facultad de Informática. Universidad Complutense de Madrid

RESUMEN DEL PROYECTO

Resumen

Esta memoria está concebida para explicar la creación, el desarrollo y la utilización de la aplicación de nuestro proyecto final de carrera para 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. Destacamos las dos partes claramente diferenciadas del proyecto:

1. *Interfaz Hombre – Máquina*: como la idea es tratar imágenes, el primer objetivo que se plantea consiste en la implementación de una herramienta de fácil utilización para el usuario donde éste pueda introducir los valores de los parámetros de los algoritmos implementados, abrir las imágenes a clasificar, elegir el método de clasificación y mostrar los resultados obtenidos. Además de lo anterior, hemos implementado algunos extras tales como guardar información básica del procesamiento de los algoritmos (clases aprendidas, número de píxeles de la imagen procesada, tiempo total en procesarla y otras funciones de utilidad) y su exportación a una hoja de cálculo para un posible estudio estadístico.

2. *Algoritmos de Clasificación*: esta parte constituye el núcleo central del proyecto y del procesamiento de las imágenes. Había que implementar varios algoritmos de clasificación y de aprendizaje. Finalmente elegimos los siguientes: *cuantización vectorial no supervisado*, *clasificador paramétrico Bayesiano* y *clasificador no paramétrico de la ventana de Parzen*.

La aplicación ha sido programada en lenguaje JAVA con la ayuda de la librería JAI (Java Advanced Imaging) para el tratamiento de las imágenes, la librería JAMA (Java Matrix) para la manipulación de fórmulas con matrices y la librería JXL para la exportación de datos a hoja de cálculo y su respectiva creación de ficheros de extensión xls.

Palabras Claves

Inteligencia artificial, algoritmos de clasificación, aprendizaje, cuantización vectorial, clasificador paramétrico Bayesiano, clasificador no paramétrico ventana de Parzen, interfaz.

Abstract

This report is conceived to explain the creation, development and use of the application of our final project for the subject Computer Systems.

The project consists in the implementation of an application for the classification of natural textures by computer vision techniques. We emphasized the two clearly differentiated parts from our project:

1. *Man/Machine Interface*: as the idea is to process images, the main goal is to implement an easy-to-use tool where we can introduce the values of the parameters of the implemented algorithms, open the images to classify, choose the classification algorithm and show the obtained results. In addition we have implemented some extra functionalities. For example, we keep basic information from the processing algorithms (learned classes, number of pixels of the processed image, total time in processing it, and other functions) and we can export it to a spreadsheet for a possible statistical analysis.

2. *Classification Algorithms*: this part is the central kernel of the project and the processing of the images. It was necessary to implement several learning and classification algorithms. Finally we chose the following algorithms: unsupervised vector quantization, Bayes parametric classifier and Parzen non-parametric classifier.

The application has been programmed in JAVA language with the aid of JAI (Java Advanced Imaging) library for the processing of the images, JAMA library for the manipulation of formulas with matrices and JXL library for the exportation of data to spreadsheet and its respective creation of xls extension files.

Keywords

Artificial intelligence, classification algorithms, learning, vector quantization, Bayesian parametric classifier, unsupervised non-parametric classifier: Parzen's window, interface.

AUTORIZACIÓN

Autorizamos 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: …….. ……

Diego López Pozueta Luis Miguel Ruiz Gil

ÍNDICE

1 INTRODUCCIÓN

1.1 Descripción del Proyecto

El presente proyecto consiste en el desarrollo e implementación de una aplicación para la clasificación de texturas naturales mediante técnicas de visión por computador.

En esencia consiste en desarrollar y programar una herramienta de fácil manejo para el usuario de forma que éste pudiese clasificar imágenes, ver los resultados obtenidos, seleccionar el algoritmo de clasificación, elegir los valores de los parámetros de estos algoritmos y otras funcionalidades que detallaremos ampliamente más adelante.

La idea puede resultar poco original, de hecho, ya existen varios programas comerciales en el mercado, si bien su implantación para aplicaciones en el ámbito rural exige todavía un gran esfuerzo. De todas formas, el objetivo del proyecto no era desarrollar una aplicación revolucionaria en este campo, sino más bien poner en práctica todos los conocimientos obtenidos a lo largo de la carrera.

Así pues, pusimos en práctica nuestros conocimientos en Inteligencia Artificial, Ingeniería del Software y Programación Orientada a Objetos, entre otros, para desarrollar ésta aplicación.

Como todo proyecto de cierta magnitud, su desarrollo se puede dividir en varias fases o etapas. Nosotros detallamos a continuación las consideradas en este proyecto:

- 1ª Fase: planteamiento del proyecto al cliente (en éste caso al profesor Gonzalo Pajares, nuestro director del proyecto) y su posterior acuerdo entre ambas partes sobre qué y cómo sería la aplicación que deberíamos implementar.
- 2ª Fase: selección y programación de los tres algoritmos de clasificación que finalmente se eligieron para la implementación. Éstos algoritmos son:
  - *Cuantización vectorial no supervisado*.
  - *Clasificador paramétrico Bayesiano*.
  - *Clasificador no paramétrico de la ventana de Parzen*.
- 3ª Fase: desarrollo de la interfaz de la aplicación integrando en ella los algoritmos de clasificación. Además, se introducen diferentes funcionalidades que creímos convenientes para nuestra herramienta.
- 4ª Fase: dedicada a las pruebas y mejoras. Se realizó un ciclo de pruebas en el que se probaba cada funcionalidad por separado. En esta fase se mejoró el aspecto estético de la aplicación.
- 5ª Fase: una vez finalizada la programación de la herramienta, nos pusimos a la redacción de ésta memoria a partir de las anotaciones y apuntes tomados durante todos los meses de desarrollo.
- 6ª Fase: presentación final del proyecto al cliente (en nuestro caso al tribunal).

1.2 Justificación del Proyecto

Actualmente existe un creciente interés en el desarrollo de aplicaciones para el tratamiento de imágenes y/o vídeos, siendo la clasificación un tema de especial relevancia. A continuación mostramos algunas de las aplicaciones en el mundo real:

1) En agricultura:
   - Control de cultivos (irrigación, aplicación de herbicidas, control de riesgos avícolas, etc.)
   - Evaluación de catástrofes naturales: fuegos, daños por inundaciones, heladas en cultivos agrícolas, nevadas, etc.
   - Detección de cambios en determinadas zonas, principalmente urbanas para el control de edificaciones o impactos medioambientales.
   - Vigilancia en prevención de catástrofes (fuegos, inundaciones, etc.).
   - Detección de infraestructuras (carreteras, cañadas reales, etc.).
2) Fines policiales:
   - Casos de personas que utilizan prótesis de goma para ocultar su verdadero rostro (el látex tiene distinta textura que la piel).
   - Reconocimiento de matrículas de vehículos en radares de velocidad.
3) Guiado automático de vehículos: al diferenciar las texturas del asfalto y la pintura un vehículo podría circular solo (proyecto ALVINN http://www.ri.cmu.edu/research_project_detail.html?type=description&project_id=160&menu_id=261).
4) Fines médicos: refuerzo a las técnicas de rayos, etc.

Existen diversas empresas u organismos que desarrollan o utilizan aplicaciones para abordar estos ámbitos. Por citar algunos ejemplos podemos mencionar:

1) Digital Image Processing (Dimap) (http://www.dimap.es/).
2) Proespacio (http://www.proespacio.org/) agrupación de empresas del sector aeroespacial donde usan imágenes por satélite. En este consorcio destacan algunas empresas líderes del sector tanto en España como a nivel Internacional: EADS-Espacio, Indra Espacio, Hispasat, IberEspacio, etc.
3) Organismos oficiales y Centros de Investigación
   - Consejo Superior de Investigaciones Científicas (CSIC). Las imágenes utilizadas en nuestro proyecto son cortesía de este organismo.
   - Instituto Nacional de Técnica Aeroespacial (INTA)

- Centro de Estudios y Experimentación de Obras Públicas (CEDEX), con el que existen trabajos de colaboración previos por parte de uno de nuestro tutor (Pajares y col. 2001, Pajares y col. 2002).

La mayoría de las empresas e instituciones anteriormente mencionadas utilizan software comercial tales como:

- ERDAS IMAGINE (http://www.imagenesgeograficas.com/ERDAS.html).
- Intergraph (http://www.intergraph.com/).
- ENVI-IDL (http://www.ittvis.com/).
- ILOG (http://www.ilog.com/).
- E-Cognition (http://www.e-cognition.net/).

En general, las mencionadas herramientas utilizan algunos de los algoritmos clásicos de clasificación, siendo necesaria la intervención del usuario mediante programación para abordar algunas de las tareas que involucran aspectos de clasificación de texturas, lo cual no siempre es factible o al menos en la medida que cabría esperar debido a múltiples limitaciones.

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

Como se ha expuesto anteriormente, el proyecto consistía en implementar un clasificador de texturas con los algoritmos de clasificación elegidos, y, ése era, en esencia, nuestro objetivo principal.

A continuación resumimos los objetivos propuestos:

- *primer objetivo*: implementación de los algoritmos de clasificación elegidos, garantizando un tiempo de ejecución eficiente y un uso de memoria reducido.
- *segundo objetivo*: implementación de una interfaz de fácil utilización, funcional y accesible a todo tipo de usuarios.
- *tercer objetivo*: garantizar el correcto funcionamiento de la aplicación.
- *cuarto objetivo:* realización de la memoria.

2 PLANIFICACIÓN

En esta sección se expondrá la forma en la que hemos planificado nuestro proyecto. Primero se mostrarán las diferentes etapas en las que hemos dividido el desarrollo. Luego se expondrán los diferentes plazos y objetivos de cada una de las fases del desarrollo que nos propusimos.

2.1 Organización

El proyecto está claramente dividido en dos partes, una que trata la parte de los algoritmos de clasificación de texturas y otra parte que se encarga de la interfaz de la aplicación. Ya que éramos sólo dos personas realizando el proyecto pensamos que era más eficiente que los dos trabajáramos al mismo tiempo en la misma parte que dividirnos las tareas y que cada uno hiciera una de las partes por separado.

Primero decidimos comenzar por los algoritmos de forma y manera que al terminar el núcleo de la aplicación, sólo habría que crear una interfaz con los servicios disponibles de los algoritmos. Por lo tanto, decidimos plantear el proyecto en dos etapas, una para el desarrollo de los algoritmos y otra para el desarrollo de la interfaz. Cada etapa tendrá su propio desarrollo.

2.2 Primera Etapa: Algoritmos

Hemos seguido el modelo establecido en Ingeniería del Software, el cual está dividido en cuatro fases: fase de inicio, fase de elaboración, fase de construcción y fase de pruebas. Cada una de estas fases estará compuesta por una o varias iteraciones establecidas.

2.2.1 Fase de Análisis

Esta fase está compuesta por una iteración, y tiene las siguientes características:

- Fecha de inicio: 20 - 11 - 2008
- Fecha de conclusión: 6 - 12 - 2008
- Objetivos:
  - Aclarar los objetivos del desarrollo de los algoritmos con el tutor.
  - Análisis de riesgos.
  - Captura de requisitos de acuerdo con los objetivos acordados con el tutor.

- Establecimiento del plan de trabajo. La planificación fijada se llevará a cabo lo más estrictamente posible.

- Elección del lenguaje de programación a utilizar para realizar el proyecto (JAVA).

- Aprendizaje y familiarización con el lenguaje establecido, así como con las bibliotecas específicas que se van a usar en la aplicación (Librería Java Advanced Imaging, Java Matrix Package).

- Investigación y documentación de los algoritmos de clasificación a implementar.

- Selección de objetivos que se deben cumplir en la siguiente fase, en función a los resultados obtenidos en esta.

2.2.2 Fase de Diseño

Esta fase está compuesta por una iteración con las siguientes características:

- Fecha de inicio: 7 - 12 - 2008
- Fecha de conclusión: 20 - 12 - 2008
- Objetivos:

  - Diseño global de los algoritmos. Este diseño se realizará de forma modular y genérica, facilitando así su integración en cualquier tipo de interfaz.

  - Especificación de casos de uso y de los servicios prestados por los algoritmos.

  - Elaboración de la documentación correspondiente a esta fase.

  - Selección de objetivos que se deben cumplir en la siguiente fase, en función de los resultados obtenidos en ésta.

2.2.3 Fase de Construcción

Esta fase posee una duración un poco más larga en el tiempo que las anteriores, por ello está formada por dos iteraciones.

Iteración 1

- Fecha de inicio: 21 - 12 - 2008
- Fechas de conclusión: 7 - 1 - 2009
- Objetivos:

- Revisión de la especificación de requisitos para una posible ampliación.

- Revisión y seguimiento de los riesgos detectados en el proyecto hasta el momento.

- Investigación y búsqueda de soluciones para los riesgos más importantes para la correcta realización del proyecto. En nuestro caso se centra en la optimización de memoria y tiempo de ejecución en los algoritmos de clasificación.

- Elaboración del primer prototipo.

Iteración 2

- Fecha de inicio: 8 - 1 - 2009
- Fechas de conclusión: 15 - 1 - 2009
- Objetivos:
  - Elaboración del prototipo final de la aplicación. En este prototipo se incorporarán las soluciones encontradas para los puntos críticos de nuestra aplicación: tiempo de ejecución.
  - Elaboración de la documentación correspondiente a esta fase.
  - Selección de objetivos que se deben cumplir en la siguiente fase.

2.2.4 Fase de Pruebas

Esta fase está compuesta por una iteración, y tiene las siguientes características:

- Fecha de inicio: 16 – 1 - 2009
- Fechas de conclusión: 26 – 1 - 2009
- Objetivos:
  - Diseño de un plan de pruebas para comprobar el correcto funcionamiento de los algoritmos. Se crearán y ejecutarán pruebas unitarias y de integración. Estas pruebas se efectuarán con diferentes cargas, y verificarán si la solución obtenida es correcta y si se cumple con los requisitos.
  - Búsqueda de mejoras para el comportamiento de nuestra aplicación.
  - Revisión de documentos realizados hasta la fecha.

2.3 Segunda Etapa: Interfaz

Hemos seguido el mismo modelo que en la etapa anterior.

2.3.1 Fase de Análisis

Esta fase está compuesta por una iteración, constando de las siguientes características:

- Fecha de inicio: 27 - 1 - 2009
- Fechas de conclusión: 7 - 2 - 2009
- Objetivos:
  - Aclarar los objetivos del desarrollo de la interfaz.
  - Captura de requisitos.
  - Establecimiento del plan de trabajo. Se plantea llevar a cabo la planificación fijada lo más estrictamente posible.
  - Elección del lenguaje de programación a utilizar para realizar el proyecto (JAVA).
  - Investigación y documentación de las funcionalidades que el interfaz implementará.
  - Selección de objetivos que se deben cumplir en la siguiente fase, en función a los resultados obtenidos en ésta.

2.3.2 Fase de Diseño

Esta fase está compuesta por una iteración, y consta de las siguientes características:

- Fecha de inicio: 8 - 2 - 2009
- Fechas de conclusión: 16 - 2 - 2009
- Objetivos:
  - Diseño global de la interfaz. Este diseño se realizará pensando en los algoritmos anteriormente creados.
  - Elaboración de la documentación correspondiente a esta fase.
  - Selección de objetivos que se deben cumplir en la siguiente fase, en función de los resultados obtenidos en ésta.

2.3.3 Fase de Construcción

Esta fase es un poco más larga que las anteriores por ello está formada por dos iteraciones.

Iteración 1

- Fecha de inicio: 17 - 2 - 2009
- Fechas de conclusión: 1 - 3 - 2009
- Objetivos:
  - Revisión de la especificación de requisitos para una posible ampliación.
  - Revisión y seguimiento de los riesgos detectados en la interfaz.
  - Elaboración del primer prototipo.

Iteración 2

- Fecha de inicio: 2 - 3 - 2009
- Fechas de conclusión: 10 - 3 - 2009
- Objetivos:
  - Elaboración del prototipo final de la interfaz.
  - Elaboración de la documentación correspondiente a esta fase.
  - Selección de objetivos que se deben cumplir en la siguiente fase.

2.3.4 Fase de Pruebas

Esta fase está compuesta por una iteración, y consta de las siguientes características:

- Fecha de inicio: 11 - 3 - 2009
- Fechas de conclusión: 31 - 3 - 2009
- Objetivos:
  - Diseño de un plan de pruebas para demostrar el correcto funcionamiento de la interfaz. Se crearán y ejecutarán pruebas unitarias y de integración. Se verificará si el comportamiento es el adecuado y si se cumple con los requisitos.
  - Diseño de un plan de pruebas para comprobar el correcto funcionamiento de la aplicación en su totalidad. Estas pruebas se efectuarán con

diferentes cargas y verificarán si la aplicación sigue los requisitos generales del proyecto.

- Búsqueda de mejoras para el comportamiento de nuestra aplicación.
- Revisión de documentos realizados hasta la fecha.

2.4 Tercera Etapa: Memoria y Manual del Usuario

- Fecha de inicio: 16 - 4 - 2009
- Fechas de conclusión: 12 - 5 - 2009
- Objetivo:
  - Realización de la memoria final del proyecto y del manual de usuario acorde con las recomendaciones sugeridas por el tutor y utilizando los documentos anteriormente elaborados.

3 ALGORITMOS DE CLASIFICACIÓN

3.1 Introducción

En éste apartado explicaremos amplia y detalladamente la teoría de los tres algoritmos de clasificación utilizados en nuestro proyecto, un algoritmo *competitivo* (Cuantización Vectorial no supervisado) y otros dos algoritmos de naturaleza *estadística* (estimador paramétrico de Bayes y estimador no paramétrico de la ventana de Parzen). Cómo nuestro tutor es Gonzalo Pajares Martinsanz, profesor con numerosas publicaciones de libros y revistas, la teoría utilizada, la cuál presentamos aquí, está extraída íntegramente de su libro *Ejercicios Resueltos de Visión por Computador* [4].

3.1.1 Definiciones Útiles

A continuación definimos algunos conceptos importantes:

- **Muestra:** en nuestro caso un píxel. Vector de 3 componentes con valores *enteros*, que representan el espacio de color RGB (R, G, B).
- **Clase**: conjunto de muestras.
- **Centro** (de una clase): vector de 3 componentes con valores *reales* (R, G, B) calculado a partir de todas las muestras de una clase, en nuestro caso serán los representantes de las clases, obtenidos por los distintos algoritmos.
- **Clasificar** (una imagen): dada una imagen de entrada y unas clases, aplicamos un algoritmo de clasificación, esto es, para cada píxel de la imagen de entrada se decide a qué clase pertenece.
- **Aprender** (de una imagen): dada una imagen de entrada añadimos las muestras (píxeles) a las distintas clases existentes o, si no pertenece a ninguna clase, creamos una nueva.

3.2 Cuantización Vectorial No Supervisado

3.2.1 Teoría del Algoritmo

Este algoritmo, asume que el número de clases no se conoce inicialmente. Por tanto, se comienza suponiendo una única clase (la primera muestra). El algoritmo progresa como sigue:

1. Para cada muestra se calcula su distancia 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 (ej. euclídea).

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 media 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.

3.2.2 Parámetros del Algoritmo

Este algoritmo necesita que introduzcamos manualmente el valor de un parámetro, el ***umbral*,** que será un valor entero (preferiblemente entre 40 y 80) que determina si se crea o no una nueva clase según lo aprendido hasta el momento.

3.2.3 Aplicación en el Proyecto

Este algoritmo es utilizado para la clasificación de imágenes o para clasificar y aprender simultáneamente.

Además también se utiliza para el aprendizaje inicial de las clases, es decir para el entrenamiento, ya que al principio si no cargamos las clases desde el fichero de tipo *xml* necesitamos establecerlas mediante este procedimiento.

3.2.4 Ejemplo de Utilización

Con los datos de la tabla siguiente aplicar el algoritmo de cuantización vectorial para determinar el número de clases y los centros de cada clase. Considerar el umbral que determina la pertenencia a las clases como T = 20.

Una vez obtenida la distribución de clases, determinar a cuál de ellas pertenece el vector $A = (93, 120, 70)^t$.

patrones

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$
R	200	90	210	35	215	92	87	41
G	160	130	170	23	172	138	128	22
B	120	60	130	44	133	54	66	37

Solución:

1.- Inicialmente se crea una única clase c_1 formada por la primera muestra a procesar $x_1 = \{200,160,120\}$ cuyo centro es $v_1 = \{200,160,120\}$, $c_1 = \{x1\}$.

2.- Segunda muestra $x_2 = \{90,130,60\}$ calculamos la distancia al único centro disponible $d(x_2,v_1) = ||x_2 - v_1|| = 128.84 > T$, por tanto, creamos una nueva clase con el patrón dado $c_2 = \{x_2\}$, cuyo centro resulta ser este mismo patrón $v_2 = \{90,130,60\}$.

3.- Tercera muestra $x_3 = \{210,170,130\}$ calculando las distancias a los dos centros de clase existentes $d(x_3,v_1) = ||x_3 - v_1|| = 17.32$ y $d(x_3,v_2) = ||x_3 - v_2|| = 144.17$, siendo el mínimo $d(x_3,v_1) < T$, por tanto esta muestra se añade a la clase c_1, quedando ésta como sigue $c_1 = \{x_1, x_3\}$. Se actualiza el centro de la clase c_1, calculando el valor medio entre x_1 y x_3, resultando $v_1 = \{205,165,125\}$.

4.- Cuarta muestra $x_4 = \{35,23,44\}$ calculando las distancias a los dos centros de clase existentes $d(x_4,v_1) = ||x_4 - v_1|| = 235.85$ y $d(x_4,v_2) = ||x_4 - v_2|| = 121.37$, siendo el mínimo $d(x_4,v_2) > T$, por tanto, es necesario crear una nueva clase $c_3 = \{x_4\}$ cuyo centro es este mismo patrón $v_3 = \{35,23,44\}$.

5.- Quinta muestra $x_5 = \{215, 172, 133\}$ calculando las distancias a los tres centros de clase existentes $d(x_5,v_1) = ||x_5 - v_1|| = 14.59$, $d(x_5,v_2) = ||x_5 - v_2|| = 150.72$ y $d(x_5,v_3) = ||x_5 - v_3|| = 250.04$, siendo el mínimo $d(x_5,v_1) < T$, por tanto, el nuevo patrón se añade a la clase $c_1 = \{x_1, x_3, x_5\}$ y se actualiza su centro $v_1 = \{208.33, 167.33, 127.67\}$.

6.- sexta muestra $x_6 = \{92, 138, 54\}$ calculando las distancias a los tres centros de clase existentes $d(x_6,v_1) = ||x_6 - v_1|| = 140.79$, $d(x_6,v_2) = ||x_6 - v_2|| = 10.20$ y $d(x_6,v_3) = ||x_6 - v_3|| = 128.74$, siendo el mínimo $d(x_6,v_2) < T$, por tanto, el nuevo patrón se añade a la clase $c_2 = \{x_2, x_6\}$ y se actualiza su centro $v_2 = \{91,134,57\}$.

7.- séptima muestra $x_7 = \{87, 128, 66\}$ calculando las distancias a los tres centros de clase existentes $d(x_7,v_1) = ||x_7 - v_1|| = 141.67$, $d(x_7,v_2) = ||x_7 - v_2|| = 11.53$ y $d(x_7,v_3) = ||x_7 - v_3|| = 119.22$, siendo el mínimo $d(x_7,v_2) < T$, por tanto, el nuevo patrón se añade a la clase $c_2 = \{x_2, x_6, x_7\}$ y se actualiza su centro $v_2 = \{89.67, 132.00, 60.00\}$.

8.- octava muestra $x_8 = \{41, 22, 37\}$ calculando las distancias a los tres centros de clase existentes $d(x_8,v_1) = ||x_8 - v_1|| = 239.46$, $d(x_8,v_2) = ||x_8 - v_2|| = 122.46$ y $d(x_8,v_3) = ||x_8 - v_3|| = 9.27$, siendo el mínimo $d(x_8,v_3) < T$, por tanto, el nuevo patrón se añade a la clase $c_3 = \{x_4, x_8\}$ y se actualiza su centro $v_3 = \{38.0, 22.5, 40.5\}$.

Finalmente, calculamos las distancias de la muestra A a cada uno de los centros dados $v_1 = \{208.33, 167.33, 127.67\}$, $v_2 = \{89.67, 132.00, 60.00\}$, $v_3 = \{38.0, 22.5, 40.5\}$, resultando: d(A, v1) = 137.36, $d(A,v_2) = 15.97$ y $d(A,v_3) = 115.76$.

Resultando por tanto, que A pertenece a la clase c_2.

3.3 Clasificador Paramétrico Bayesiano

3.3.1 Teoría del Algoritmo

3.3.1.1 Caso Normal Multivariable: media desconocida

Supongamos que las muestras siguen una distribución Normal con media **m** y

matriz de covarianza *C*. Por simplicidad en este caso concreto el único parámetro desconocido es la media,

$$p(\mathbf{x}_i/\mathbf{m}) = \frac{1}{(2\pi)^{d/2}|C|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}_i - \mathbf{m})^t C^{-1}(\mathbf{x}_i - \mathbf{m})\right\} \qquad (3.1)$$

$$\ln p(\mathbf{x}_i / \mathbf{m}) = -\frac{1}{2}\ln\left\{(2\pi)^d |C|\right\} - \frac{1}{2}(\mathbf{x}_i - \mathbf{m})^t C^{-1}(\mathbf{x}_i - \mathbf{m}) \qquad (3.2)$$

la expresión anterior se obtiene identificando $\boldsymbol{w}$ con $\mathbf{m}$

$$\nabla_{\mathbf{m}} \ln p(\mathbf{x}_i/\mathbf{m}) = C^{-1}(\mathbf{x}_i - \mathbf{m}) \qquad (3.3)$$

La minimización del riesgo empírico supone que

$$\frac{1}{n}\sum_{i=1}^{n} C^{-1}(\mathbf{x}_i - \mathbf{m}) = \mathbf{0} \qquad (3.4)$$

multiplicando por *C* y despejando **m** se obtiene una estima para la misma, que resulta

$$\mathbf{m}^* = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \qquad (3.5)$$

Este es un resultado bastante satisfactorio, dice que la estima de máxima verosimilitud para la media desconocida de una distribución es exactamente la media aritmética de las muestras, es decir, la *media simple*.

Geométricamente, si consideramos las *n* muestras como una nube de puntos, la media simple es el centroide de la nube, que puede ser considerado el representante de dicha clase.

La media anteriormente obtenida tiene las propiedades estadísticas conocidas y siempre se tiene tendencia a utilizarla aunque no se tenga el conocimiento de que la misma es la solución de máxima verosimilitud.

3.3.1.2 *Caso Normal Multivariable: media y matriz de covarianza desconocida*

En el caso general y más típico de una 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 $\boldsymbol{w} = \{w_1, w_2\}$. Consideremos el caso univariable con $w_1 = \mathbf{m}$ y $w_2 = \sigma^2$ en cuyo caso

$$\ln p(x_i/\boldsymbol{w}) = -\frac{1}{2}\ln 2\pi w_2 - \frac{1}{2w_2}(x_i - w_1)^2 \qquad (3.6)$$

$$\nabla_{\boldsymbol{w}} \ln p(x_i \,/\, \boldsymbol{w}) = \begin{bmatrix} \dfrac{1}{w_2}(x_i - w_1) \\ -\dfrac{1}{2w_2} + \dfrac{(x_i - w_1)^2}{2w_2^2} \end{bmatrix} \qquad (3.7)$$

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

$$\frac{1}{n}\sum_{i=1}^{n} \frac{1}{\hat{w}_2}(x_i - \hat{w}_1) = 0 \qquad -\frac{1}{n}\sum_{i=1}^{n} \frac{1}{2\hat{w}_2} + \frac{1}{n}\sum_{i=1}^{n} \frac{(x_i - \hat{w}_1)^2}{2\hat{w}_2^2} = 0 \qquad (3.8)$$

donde $\hat{w}_1$ y $\hat{w}_2$ son las estimas de máxima verosimilitud para w_1 y w_2, respectivamente. Sustituyendo $\hat{m} = \hat{w}_1$ y $\hat{\sigma}^2 = \hat{w}_2$ obtenemos las estimas de máxima verosimilitud para **m** y σ^2

$$\hat{m} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - \hat{w}_1)^2 \qquad (3.9)$$

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,

$$\mathbf{m} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \qquad C = \frac{1}{n-1}\sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^{\mathrm{t}} \qquad (3.10)$$

La expresión (3.10) nos dice 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 $(\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^{\mathrm{t}}$. Puesto que la verdadera matriz de covarianza es el valor esperado de la matriz $(\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^{\mathrm{t}}$, se obtiene un resultado muy satisfactorio.

3.3.1.3 Teoría de la decisión de Bayes: el clasificador Bayesiano

La teoría de la decisión de Bayes es un método estadístico clásico en clasificación de patrones. Se basa en el supuesto de que el problema de la decisión se enfoca en términos probabilísticos y que todas las probabilidades relevantes resultan conocidas.

Supongamos que no tenemos conocimiento acerca de los valores R, G, y B de la muestra (de aquí en adelante píxel) que estamos analizando. Utilizando la terminología de la teoría de la decisión, podemos decir que la probabilidad a priori de que un píxel pertenezca a cualquiera de las clases es la misma y cuya suma es la unidad.

Sea ***y*** el estado que designa la pertenencia de un píxel a una de las clases, ***c*** el número de clases en que hay que dividir la imagen y considerando a ***y*** una variable aleatoria, ya que la pertenencia a uno de los dos estados es impredecible, podemos decir de forma más precisa que suponemos que existe alguna *probabilidad a priori* $P(y=c_i)$

de que el píxel pertenezca a la clase c_i y alguna *probabilidad a priori* de que pertenezca a la clase c_j $P(y=c_j)$ y así con todas las clases que queramos hasta ***c***.

A la hora de decidir acerca de la pertenencia de un píxel a una clase concreta, la única información son esas probabilidades a priori y parece razonable utilizar la siguiente *regla de decisión*: elegir la que tenga mayor probabilidad.

En nuestro caso conocemos los valores R, G y B del píxel a clasificar. A estos tres valores los llamaremos $\mathbf{x} = \{x_R, x_G, x_B\}$**.** Así, diferentes píxeles darán diferentes valores de **x**, resulta natural por tanto expresar esta variabilidad en términos probabilísticos. En este sentido, consideremos a **x** una variable aleatoria continua cuya distribución depende de la clase.

Sean $p(\mathbf{x} / y = c_j)$ las funciones de *densidad de probabilidad condicionales* para **x** dado que el píxel pertenezca a la clase c_i. Supongamos que conocemos tanto las probabilidades a priori como estas últimas funciones de densidad de probabilidad. Suponer además que medimos las componentes R, G y B de un píxel y descubrimos que vale **x**, la pregunta será ¿cómo influye esta medida sobre nuestra actitud con relación a la clase de que se trata? La respuesta nos la proporciona la *regla de Bayes*, considerando que tanto las probabilidades a priori $P(y = c_j)$ como las densidades condicionales para cada clase $p(\mathbf{x} / y = c_j)$ 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:

$$p(y = c_j / \mathbf{x}) = \frac{p(\mathbf{x} / y = c_j) P(y = c_j)}{p(\mathbf{x})} \qquad (3.11)$$

donde

$$p(\mathbf{x}) = \sum_{j=1}^{c} p(\mathbf{x} / y = c_j) P(y = c_j) \qquad (3.12)$$

La regla de Bayes muestra cómo la observación del valor **x** cambia las probabilidades a priori a las *probabilidades a posteriori* $p(y = c_j / \mathbf{x})$.

Una vez que se determinan esas probabilidades a posteriori, la siguiente regla de decisión se utiliza para clasificar **x**.

$$\mathbf{x} \in c_i \;\; sii \;\; p(y = c_i / \mathbf{x}) > p(y = c_j / \mathbf{x}) \;\; \forall\, i \neq j, \;\; i, j = 1,2,..c \qquad (3.13)$$

Ahora bien, si nos fijamos en el segundo término de la expresión que obtiene las *probabilidades a posteriori* del teorema de Bayes y eliminado el término no discriminante $p(\mathbf{x})$(no aporta nada en la decisión), se tiene una forma alternativa de clasificar el vector de atributos **x**:

$$\mathbf{x} \in c_i \;\; sii \;\; P(\mathbf{x} / y = c_i) P(y = c_i) > P(\mathbf{x} / y = c_j) P(y = c_j) \;\; \forall i \neq j, \;\; i, j = 1,2,..,c \qquad (3.14)$$

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 la regla anterior y teniendo en cuenta el signo negativo en el término exponencial de la función de densidad de probabilidad Normal, así

$$\mathbf{x} \in c_i \;\; sii \;\; d_M^2(\mathbf{x},\mathbf{m}_i) < d_M^2(\mathbf{x},\mathbf{m}_j) \;\; \forall i \neq j, \;\; i,j = 1,2,..,c \qquad (3.15)$$

donde $\mathbf{m}_i$, $\mathbf{m}_j$ son los vectores media de las clases c_i y c_j respectivamente.Sin pérdida de generalidad, la distancia de un vector $\mathbf{x}_k$ a la clase c_i resulta ser:

$$d_M^2(\mathbf{x}_k,\mathbf{m}_i) = (\mathbf{x}_k - \mathbf{m}_i)^{\mathrm{t}} C_i^{-1} (\mathbf{x}_k - \mathbf{m}_i) \qquad (3.16)$$

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,

$$d_E^2(\mathbf{x},\mathbf{m}_i) = (\mathbf{x} - \mathbf{m}_i)^t (\mathbf{x} - \mathbf{m}_i) = \mathbf{x}^t\mathbf{x} - 2\mathbf{x}^t\mathbf{m}_i + \mathbf{m}_i^t\mathbf{m}_i \qquad (3.17)$$

En la expresión anterior el término $\mathbf{x}^t\mathbf{x}$ 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 anterior se obtiene la siguiente función discriminante, donde se deduce que la distancia Euclídea al cuadrado mínima hace la expresión siguiente máxima:

$$fd(\mathbf{x}) = \mathbf{x}^{\mathrm{t}}\mathbf{m}_{\mathrm{i}} - \frac{1}{2}\mathbf{m}_{\mathrm{i}}^{\mathrm{t}}\mathbf{m}_{\mathrm{i}} \qquad (3.18)$$

3.3.2 Parámetros del Algoritmo

En nuestro caso no utilizamos ningún parámetro en éste algoritmo que tengamos que introducir manualmente, sino que $\mathbf{m}$ y C se calculan durante el procesamiento del algoritmo de la siguiente manera:

$$\mathbf{m} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \qquad C = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^{\mathrm{t}}$$

3.3.3 *Ejemplo de Utilización*

Supongamos la siguiente distribución de las muestras en el espacio $\Re^2$ asignadas a las clases c_1 y c_2.

c_1	x_{11}	1	3	1	2	3	c_2	x_{21}	6	6	7	8	8
	x_{12}	2	3	5	2	3		x_{22}	4	3	4	4	5

Dado $\mathbf{x}_k = (4,5)$ clasificarlo como perteneciente a una de las dos clases.

Solución:

Estimamos los vectores media y las correspondientes matrices de covarianza,

clase c_1:

media: $\boldsymbol{m}_1 = \frac{1}{n}\sum_{j=1}^{5} \boldsymbol{x}_{1j} = \frac{1}{5}\begin{bmatrix}10\\15\end{bmatrix} = \begin{bmatrix}2\\3\end{bmatrix}$

matriz de covarianza:

$$\boldsymbol{C}_1 = \frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{x}_i - \boldsymbol{m}_1)(\boldsymbol{x}_i - \boldsymbol{m}_1)' =$$

$$\frac{1}{4}\left[\begin{pmatrix}-1\\-1\end{pmatrix}\begin{pmatrix}-1 & -1\end{pmatrix} + \begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}1 & 0\end{pmatrix} + \begin{pmatrix}-1\\2\end{pmatrix}\begin{pmatrix}-1 & 2\end{pmatrix} + \begin{pmatrix}0\\-1\end{pmatrix}\begin{pmatrix}0 & -1\end{pmatrix} + \begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}1 & 0\end{pmatrix}\right] =$$

$$\frac{1}{4}\left[\begin{pmatrix}1 & 1\\1 & 1\end{pmatrix} + \begin{pmatrix}1 & 0\\0 & 0\end{pmatrix} + \begin{pmatrix}1 & -2\\-2 & 4\end{pmatrix} + \begin{pmatrix}0 & 0\\0 & 1\end{pmatrix} + \begin{pmatrix}1 & 0\\0 & 0\end{pmatrix}\right] = \frac{1}{4}\begin{pmatrix}4 & -1\\-1 & 6\end{pmatrix} = \begin{pmatrix}1.0 & -0.25\\-0.25 & 1.5\end{pmatrix}$$

clase c_2:

media: $\boldsymbol{m}_2 = \frac{1}{5}\sum_{j=1}^{5} \boldsymbol{x}_{1j} = \frac{1}{5}\begin{bmatrix}35\\20\end{bmatrix} = \begin{bmatrix}7\\4\end{bmatrix}$

matriz de covarianza:

$$\boldsymbol{C}_2 = \frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{x}_i - \boldsymbol{m}_2)(\boldsymbol{x}_i - \boldsymbol{m}_2)' =$$

$$\frac{1}{4}\left[\begin{pmatrix}-1\\0\end{pmatrix}\begin{pmatrix}-1 & 0\end{pmatrix} + \begin{pmatrix}-1\\-1\end{pmatrix}\begin{pmatrix}-1 & -1\end{pmatrix} + \begin{pmatrix}0\\0\end{pmatrix}\begin{pmatrix}0 & 0\end{pmatrix} + \begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}1 & 0\end{pmatrix} + \begin{pmatrix}1\\1\end{pmatrix}\begin{pmatrix}1 & 1\end{pmatrix}\right] =$$

$$\frac{1}{4}\left[\begin{pmatrix}1 & 0\\0 & 0\end{pmatrix} + \begin{pmatrix}1 & 1\\1 & 1\end{pmatrix} + \begin{pmatrix}0 & 0\\0 & 0\end{pmatrix} + \begin{pmatrix}1 & 0\\0 & 0\end{pmatrix} + \begin{pmatrix}1 & 1\\1 & 1\end{pmatrix}\right] = \frac{1}{4}\begin{pmatrix}4 & 2\\2 & 2\end{pmatrix} = \begin{pmatrix}1.0 & 0.5\\0.5 & 0.5\end{pmatrix}$$

Dado $\mathbf{x}_k = (4,5)$, obtenemos las distancias a las dos clases como sigue,

clase c_1: $d_M^2(\mathbf{x}_k, \mathbf{m}_1) = (\mathbf{x}_k - \mathbf{m}_1)C_1^{-1}(\mathbf{x}_k - \mathbf{m}_1)^t = (2,2)\begin{pmatrix} 1.0435 & 0.1739 \\ 0.1739 & 0.6957 \end{pmatrix}\begin{pmatrix} 2 \\ 2 \end{pmatrix} = 8.35$

clase c_2: $d_M^2(\mathbf{x}_k, \mathbf{m}_2) = (\mathbf{x}_k - \mathbf{m}_2)C_2^{-1}(\mathbf{x}_k - \mathbf{m}_2)^t = (-3,1)\begin{pmatrix} 2.0 & -2.0 \\ -2.0 & 4.0 \end{pmatrix}\begin{pmatrix} -3 \\ 1 \end{pmatrix} = 34.00$

Como $d^2_M(\mathbf{x}_k,\mathbf{m}_1) < d^2_M(\mathbf{x}_k,\mathbf{m}_2)$ deducimos que $\boldsymbol{x}_k = (4,5) \in c_1$.

3.3.4 Aplicación en el Proyecto

Este algoritmo es utilizado para la clasificación de imágenes o para clasificar y aprender simultáneamente.

Una vez obtenidas las clases con sus respectivas muestras, ya sea mediante el aprendizaje de una imagen o cargándolas desde un fichero del tipo *xml*, calculamos para cada clase la media y la covarianza como se muestra a continuación:

$$\mathbf{m} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \qquad C = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^t$$

El valor de las medias y covarianzas obtenidas serán posteriormente utilizadas para realizar la clasificación.

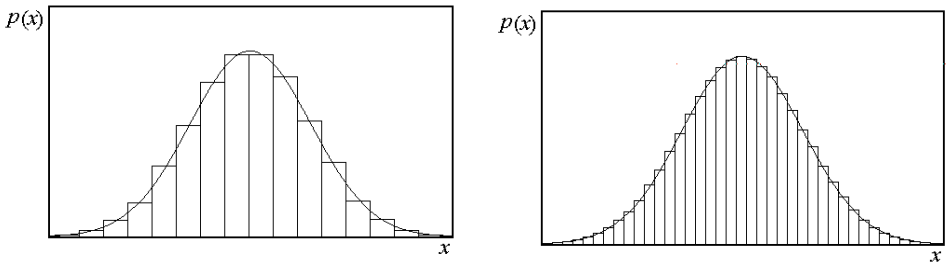
3.4 Clasificador No Paramétrico: ventana de Parzen

3.4.1 Teoría del Algoritmo

A diferencia de la estimación paramétrica, donde la función de densidad de probabilidad *fdp* se obtiene estimando los parámetros desconocidos de un modelo conocido, en la estimación no paramétrica no se conoce el modelo. Las técnicas no paramétricas son básicamente variaciones de la aproximación del histograma de una *fdp* desconocida. Consideremos el caso unidimensional. La figura que mostramos a continuación muestra dos ejemplos de una *fdp* y su aproximación por el método del histograma, esto es, el eje *x* (espacio unidimensional) se divide primero en intervalos de longitud *h*.

La probabilidad de una muestra *x* localizada en un intervalo se puede deducir fácilmente. Si *N* es el número total de muestras y una fracción de ellas k_N se sitúan dentro de uno de los intervalos, la correspondiente probabilidad se aproxima por la *razón de frecuencia*,

$$P \approx k_N / N \quad (3.19)$$



(*a*) (*b*)

Figura 3.1 Aproximación de la función de densidad de probabilidad por el método del histograma con (a) intervalos de longitud grande y (b) pequeña

Esta aproximación tiende a la verdadera probabilidad *P* a medida que $N \to \infty$. El correspondiente valor de la fdp se considera constante a lo largo del intervalo y se aproxima por,

$$\hat{p}(x) \equiv \hat{p}(\hat{x}) \approx \frac{1}{h}\frac{k_N}{N}, \qquad |x - \hat{x}| \le \frac{h}{2} \qquad (3.20)$$

donde $\hat{x}$ es el punto medio del intervalo. Esto determina la amplitud de la curva del histograma sobre el intervalo. Ésta es una aproximación razonable para *p(x)* continua y *h* suficientemente pequeña. Se puede demostrar que $\hat{p}(x)$ converge hacia el verdadero valor *p(x)* a medida que $N \to \infty$ dado que,

$$h_N \to 0 \qquad k_N \to \infty \qquad \frac{k_N}{N} \to 0 \qquad (3.21)$$

Se introduce h_N para mostrar la dependencia de *N*. Estas condiciones pueden comprenderse sin grandes detalles matemáticos. La primera ya ha sido discutida. Las otras dos muestran la forma en la que debe crecer k_N para garantizar la convergencia. En efecto, en todos los puntos donde $p(x) \neq 0$, una vez fijada la dimensión h_N suficientemente pequeña, la probabilidad *P* de los puntos que caen en este intervalo es finita. Además, $k_N \approx PN$ y k_N tiende a infinito a medida que *N* crece a infinito. Por otra parte, a medida que la dimensión h_N del intervalo tiende a cero, la correspondiente probabilidad también tiende a cero, justificando la última condición. En la práctica el número *N* de datos es finito. Las condiciones precedentes dan idea de cómo deben elegirse los diferentes parámetros. *N* debe ser suficientemente grande, h_N suficientemente pequeño y el número de puntos dentro de cada intervalo suficientemente grande también. Cómo de grandes o pequeños depende del tipo de *fdp* y del grado de aproximación que se desea. A continuación describimos el método de la

Ventana de Parzen.

En el caso multidimensional, en lugar de intervalos de dimensión h, el espacio p-dimensional se divide en hipercubos con la longitud de los lados h y volumen h^p. Sean $\mathbf{x}_i$, $i = 1,2,\ldots,N$ los vectores de atributos disponibles. Definimos la función $\phi(\mathbf{x})$ de modo que,

$$\phi(\mathbf{x}_i) = \begin{cases} 1 & para \quad |x_{ij}| \le \frac{1}{2} \\ 0 & en\ otros\ casos \end{cases} \qquad (3.22)$$

donde x_{ij}, $j = 1,..,d$ son las componentes de $\mathbf{x}_i$. En otras palabras, la función es igual a 1 para todos los puntos dentro del hipercubo de lado la unidad, centrado en el origen y 0 fuera de él. Con esto la ecuación (3.20) se puede reescribir como,

$$\hat{p}(\mathbf{x}) = \frac{1}{h^p}\left(\frac{1}{N}\sum_{i=1}^{N}\phi\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right)\right) \qquad (3.23)$$

La interpretación de la ecuación (3.23) es sencilla, consideramos un hipercubo con longitud de lado h centrado en $\mathbf{x}$, que es el punto donde queremos estimar la *fdp*. La suma es k_N, esto es, el número de puntos que caen dentro del hipercubo. Entonces la estima *fdp* se obtiene dividiendo k_N por N y el respectivo volumen del hipercubo h^p. No obstante, volviendo a la ecuación (3.23) y observándola desde una perspectiva diferente, vemos que estamos intentando aproximar una función continua *p(x)* mediante una expansión de términos de funciones discontinuas $\phi(\cdot)$. Esto condujo a la generalización de (3.23) propuesta por Parzen (1962) utilizando funciones continuas en lugar de $\phi(\cdot)$, dichas funciones son conocidas como núcleos o funciones potenciales o ventanas de Parzen. Ejemplos típicos de funciones de este tipo son los núcleos Gaussianos,

$$p(\mathbf{x}) = \frac{1}{n}\sum_{j=1}^{n}\left\{\frac{1}{(2\pi)^{p/2}|C|^{1/2}h^p}\exp\left[-\frac{1}{2h^2}(\mathbf{x}-\mathbf{x}_j)C^{-1}(\mathbf{x}-\mathbf{x}_j)\right]\right\} \qquad (3.24)$$

Una vez estimadas las funciones de densidad de probabilidad para cada una de las c_1, c_2, …, c_c clases a través de (3.24), la clasificación de una muestra $\mathbf{x}$ se lleva a cabo mediante la siguiente decisión,

$$\mathbf{x} \in c_i \quad \text{sii} \quad p_i(\mathbf{x}) > p_j(\mathbf{x}) \quad \forall j \neq i \qquad (3.25)$$

3.4.2 *Parámetros del Algoritmo*

En este algoritmo de clasificación el único parámetro que el usuario deberá introducir previamente es *h,* también llamado ancho de la ventana de Parzen, siendo su valor un número real.

3.4.3 *Ejemplo de Utilización*

Supongamos la distribución de datos en el espacio $\Re^3$ y con las tres clases c_1, c_2 y c_3 que mostramos a continuación. Dado el vector **A** = (135, 170,208) determinar su pertenencia a una de las tres clases mediante el procedimiento de la ventana de Parzen. Considerar $h = 4$.

	c_1					c_2					c_3				
R	200	210	215	210	198	90	92	87	91	85	30	20	24	28	22
G	160	170	172	165	177	130	138	128	134	123	44	40	42	50	46
B	120	130	133	134	138	60	54	66	60	55	178	180	184	176	181

Solución:

Estimamos las funciones de densidad de probabilidad mediante la ecuación (3.24) para cada una de las tres clases. Puesto que las muestras son 3-dimensionales, $p = 3$.

Puesto que todas las clases tienen 5 muestras, $n = 5$ y por tanto el sumatorio de la ecuación (3.24) se extiende desde $j = 1$ hasta 5.

Para cada una de las clases se determinan los vectores $\mathbf{x}_j$ que intervienen en el cómputo de la ecuación (3.24) siendo exactamente los que se indican en la siguiente tabla:

	c_1					c_2					c_3				
	$^1\mathbf{x}_1$	$^1\mathbf{x}_2$	$^1\mathbf{x}_3$	$^1\mathbf{x}_4$	$^1\mathbf{x}_5$	$^2\mathbf{x}_1$	$^2\mathbf{x}_2$	$^2\mathbf{x}_3$	$^2\mathbf{x}_4$	$^2\mathbf{x}_5$	$^3\mathbf{x}_1$	$^3\mathbf{x}_2$	$^3\mathbf{x}_3$	$^3\mathbf{x}_4$	$^3\mathbf{x}_5$
R	200	210	215	210	198	90	92	87	91	85	30	20	24	28	22
G	160	170	172	165	177	130	138	128	134	123	44	40	42	50	46
B	120	130	133	134	138	60	54	66	60	55	178	180	184	176	181

Las matrices de covarianza C_1, C_2 y C_3 son

$$C_1 = \begin{bmatrix} 52.8 & 1.4 & 9.0 \\ 1.4 & 42.7 & 37.0 \\ 9.0 & 37.0 & 46.0 \end{bmatrix} \quad C_2 = \begin{bmatrix} 8.5 & 16.0 & -2.5 \\ 16.0 & 32.8 & -5.5 \\ -2.5 & -5.5 & 23.0 \end{bmatrix} \quad C_3 = \begin{bmatrix} 17.2 & 8.6 & -7.3 \\ 8.6 & 14.8 & -7.4 \\ -7.3 & -7.4 & 9.2 \end{bmatrix}$$

Clase c_1:

$$p_1(\mathbf{x}) = \frac{1}{5}\frac{1}{(2\pi)^{3/2}|C_1|^{1/2}4^3}\left\{\exp\left[-\frac{1}{2x4^2}\left(\mathbf{x}-{}^1\mathbf{x}_1\right)C_1^{-1}\left(\mathbf{x}-{}^1\mathbf{x}_1\right)\right]+\right.$$
$$\exp\left[-\frac{1}{2x4^2}\left(\mathbf{x}-{}^1\mathbf{x}_2\right)C_1^{-1}\left(\mathbf{x}-{}^1\mathbf{x}_2\right)\right]+\exp\left[-\frac{1}{2x4^2}\left(\mathbf{x}-{}^1\mathbf{x}_3\right)C_1^{-1}\left(\mathbf{x}-{}^1\mathbf{x}_3\right)\right]+$$
$$\left.\exp\left[-\frac{1}{2x4^2}\left(\mathbf{x}-{}^1\mathbf{x}_4\right)C_1^{-1}\left(\mathbf{x}-{}^1\mathbf{x}_4\right)\right]+\exp\left[-\frac{1}{2x4^2}\left(\mathbf{x}-{}^1\mathbf{x}_4\right)C_1^{-1}\left(\mathbf{x}-{}^1\mathbf{x}_4\right)\right]\right.$$

Procediendo de forma análoga para las clases c_2 y c_3 obtendríamos$p_2(\mathbf{x})$ y $p_3(\mathbf{x})$.

Ahora estamos en condiciones de obtener los valores de estas funciones para **x**= **A** = (135, 170, 208) obteniendo finalmente tras la realización de los cálculos correspondientes,

$$p_1(\mathbf{A}) = 9.4x10^{-3}; \quad p_2(\mathbf{A}) = 1.2x10^{-122} \text{ y } p_3(\mathbf{A}) = 1.1x10^{-141}$$

Como $\max\{p_1(\mathbf{A}), p_2(\mathbf{A}), p_3(\mathbf{A})\} = p_1(\mathbf{A})$, el vector A se clasifica como perteneciente a la clase c_1 que se corresponde con el valor de la máxima probabilidad.

3.4.4 Aplicación en el Proyecto

Éste algoritmo es utilizado para la clasificación de imágenes o para clasificar y aprender simultáneamente. El problema de éste algoritmo es que su tiempo de ejecución es elevado, incluso para imágenes pequeñas, como explicaremos más adelante en los resultados obtenidos.

4 DISEÑO

En este capítulo se exponen los requisitos analizados para la aplicación, los riesgos encontrados, el análisis de los casos de usos, el diagrama de componentes que se seguirá, el diagrama de clases del paquete de algoritmos y un ejemplo de un diagrama de secuencia.

4.1 Requisitos

Los requisitos que debe cumplir la aplicación son básicamente que un usuario cualquiera pueda aprender y clasificar una imagen por los tres algoritmos de que consta la aplicación, y su posterior visualización del resultado de la imagen clasificada.

4.1.1 Funcionales

1. Los algoritmos deberán realizar las funciones de entrenamiento y clasificación.
2. Parámetros a elegir por el usuario (como mínimo) serán:
    a. *Umbral* y *h*.
    b. Imágenes con las que se realizará el aprendizaje de las clases.
    c. Imágenes a clasificar.
3. El entrenamiento se llevará acabo con tantas imágenes como el usuario desee.
4. Los algoritmos serán capaces de aprender cualquier número de clases diferentes, todo dependerá del valor elegido para el *umbral*.

4.2 Riesgos

El desarrollo del proyecto tuvo algunos riesgos que explicamos a continuación junto con su solución.

4.2.1 Técnologicos

RIESGO	DESCRIPCIÓN
PÉRDIDA DE DATOS	Posibilidad de pérdida de información, código o documentación debido a un fallo de la tecnología usada para este fin.
INSUFICIENCIA DE RECURSOS	Puede que la aplicación requiera de alguna tecnología de la que no dispongamos.

INCOMPATIBILIDAD DE LAS PARTES	El proyecto desarrollado tal vez requiera un determinado software o hardware que el equipo desconoce
COMPLEJIDAD DE CÓDIGO	El código realizado por algún miembro del grupo, puede ser poco legible, poco comentado o incluso muy complicado.
TIEMPO DE EJECUCIÓN	El tiempo de respuesta del algoritmo debido a una mala algoritmia puede ser excesivo y desagradable a vista del usuario

4.2.2 Organización

RIESGO	DESCRIPCIÓN
PERDIDA DE DATOS	Posibilidad de pérdida de información, código o documentación debido a una mala gestión de los documentos.
FALTA DE COMUNICACIÓN	Trabajo defectuoso por que los componentes no se comunican entre ellos.

4.2.3 Análisis y Solución

RIESGO	PROBABILIDAD	EFECTO	SOLUCIÓN
PÉRDIDA DE DATOS	Moderada	Catastrófico	Llevar copias de seguridad de todos aquellos trabajos realizados, tanto físicamente, como en diferentes dispositivos de almacenamiento (discos diferentes, disquetes, etc.) añadiendo la fecha de última modificación.
INSUFICIENCIA DE RECURSOS	Moderada	Catastrófico	Hacer un reconocimiento de en que recursos puede fallar e investigar en como solventar el posible fallo.
INCOMPATIBILIDAD DE LAS PARTES	Moderada	Catastrófico	Antes de empezar a codificar establecer un entorno que satisfaga las necesidades de todas las partes.
COMPLEJIDAD DECÓDIGO	Moderada	Tolerable	Añadir siempre comentarios de explicación del código para mayor claridad.
TIEMPO DE EJECUCIÓN	Alta	Grave	Investigación de cómo optimizar al máximo el tiempo de ejecución de cada algoritmo
FALTA DE COMUNICACIÓN	Baja	Catastrófico	Establecer un canal de comunicación común para el intercambio de información y avances.

4.3 Diagramas de Casos de Uso

En la definición del proyecto sólo se nos exigía las funciones de aprender, cargar/salvar un aprendizaje, y la clasificación con diferentes algoritmos. Nosotros hemos ampliado los requisitos con más funcionalidades, tales como la gestión de un historial y la funcionalidad de cancelar un procesamiento en cualquier momento de su ejecución mediante la técnica de programación basada en "hilos".

Por tanto, los diferentes casos de uso de la aplicación son los siguientes:

- un método de aprendizaje que cree las diferentes clases a partir de una imagen de entrada
- un método que permita guardar en un archivo el aprendizaje (las diferentes clases con sus píxeles correspondientes)
- un método para cargar los aprendizajes guardados
- un método de clasificación por cada algoritmo, el cual recibirá una imagen y clasificará cada píxel según el algoritmo y mostrara la imagen resultante en una pestaña
- un método de clasificación y aprendizaje por cada algoritmo, el cual clasificará los píxeles de una imagen, mostrará el resultado en una pestaña y agregará cada píxel a la clase a la cual pertenece, aumentando de esta forma el número de muestras de las clases
- un método para detener el procesamiento de una imagen y volver al estado siguiente al inicio del procesamiento
- un método que guarde el historial de clasificación. Dicho historial tendrá la siguiente información para cada clasificación: tiempo de ejecución, nombre de la imagen que se uso, su número de píxeles, el algoritmo utilizado y la fecha
- un método para abrir el historial
- un método para exportar el historial a una archivo de hoja de cálculo del tipo *xls*
- un método para borrar el historial actual

El diagrama de caso de uso de la aplicación es el que se muestra en la figura 4.1.

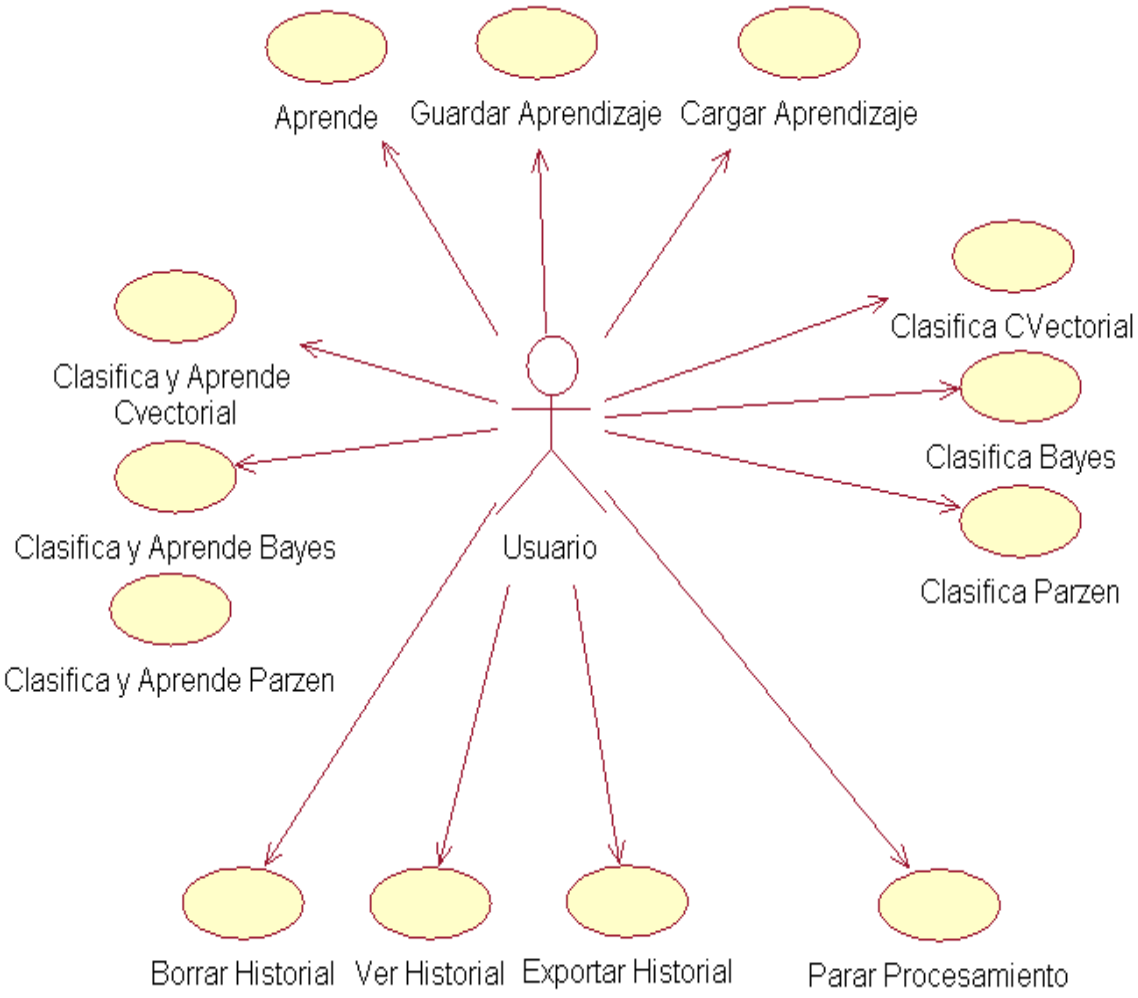


Figura 4.1 Diagrama de casos de uso

4.4 Diagrama de Componentes

La aplicación desarrollada se ha basado en un diseño modular en paquetes, siguiendo la estructura definida en la figura 4.2.

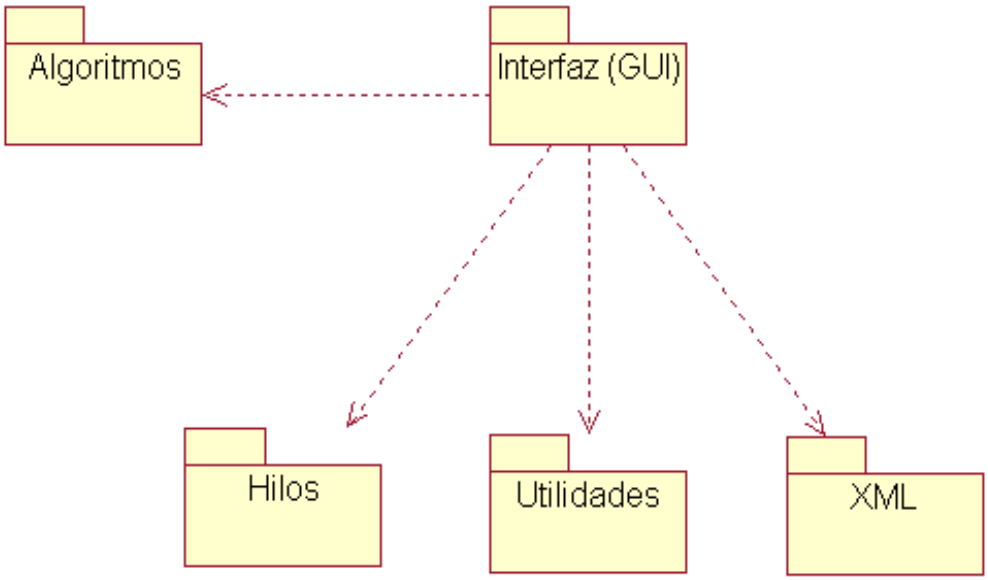


Figura 4.2 Diagrama de componentes

Los algoritmos y la interfaz se han implementado dentro de sus respectivos paquetes. Por lo tanto, la interfaz depende directamente del paquete de algoritmos ya que accede a todos sus servicios.

El paquete de hilos es el encargado de crear los diferentes hilos de ejecución, y es controlado desde la interfaz. Estos hilos nos ayudan a que en cualquier momento de un procesamiento se permita al usuario cancelar la operación y volver al estado anterior al procesamiento.

Dentro del paquete de utilidades se encuentran los módulos encargados de la creación de las tablas especiales que muestran los colores de los píxeles y también del paso de los archivos de XML a XLS

Dentro del paquete XML nos encargamos del manejo de los historiales de clasificación de la aplicación, de guardar y cargar las clases aprendidas.

El paquete de algoritmos está organizado de la siguiente manera:

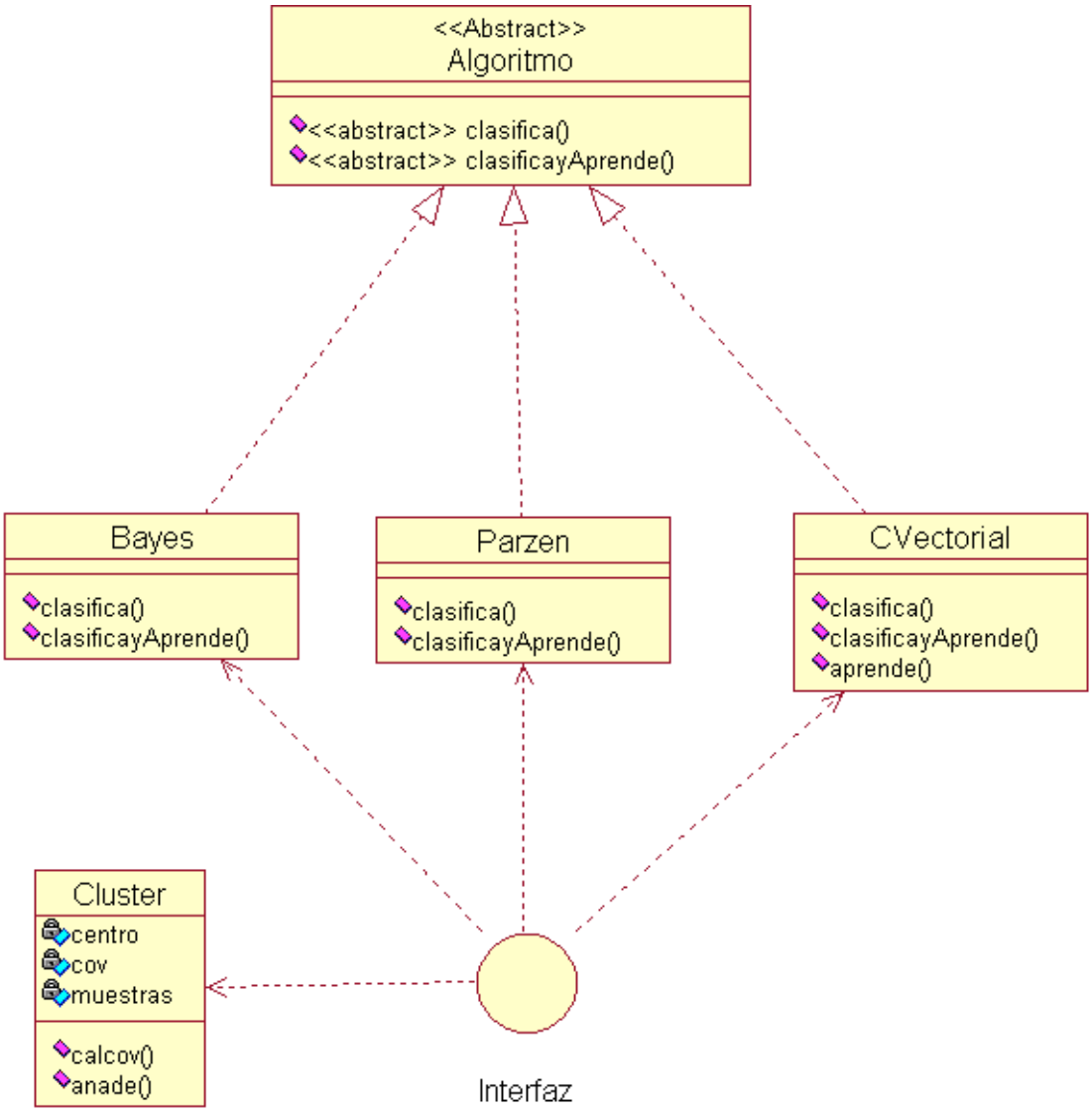


Figura 4.3 Diagrama de clases del paquete de algoritmos

Hemos creado una clase abstracta Algoritmos, para que la creación de un nuevo algoritmo de clasificación resulte lo más fácil posible. Como hemos implementado tres algoritmos hemos creado sus tres clases correspondientes. Por otra parte, tenemos la clase Cluster, que representa las diferentes clases de píxeles. La interfaz es la encargada de pasar las clases aprendidas a los algoritmos y además los píxeles a clasificar de las imágenes de entrada.

4.5 Diagrama de Secuencia

Este es un ejemplo del diagrama de secuencia para la función de clasificar una imagen con el algoritmo de Bayes. Anteriormente al proceso de clasificación, el sistema habrá aprendido y tendrá disponibles las clases creadas en la variable *clases*. En la figura 4.4, se muestra un esquema del diagrama de secuencia basado en el algoritmo de Bayes, que se explica posteriormente.

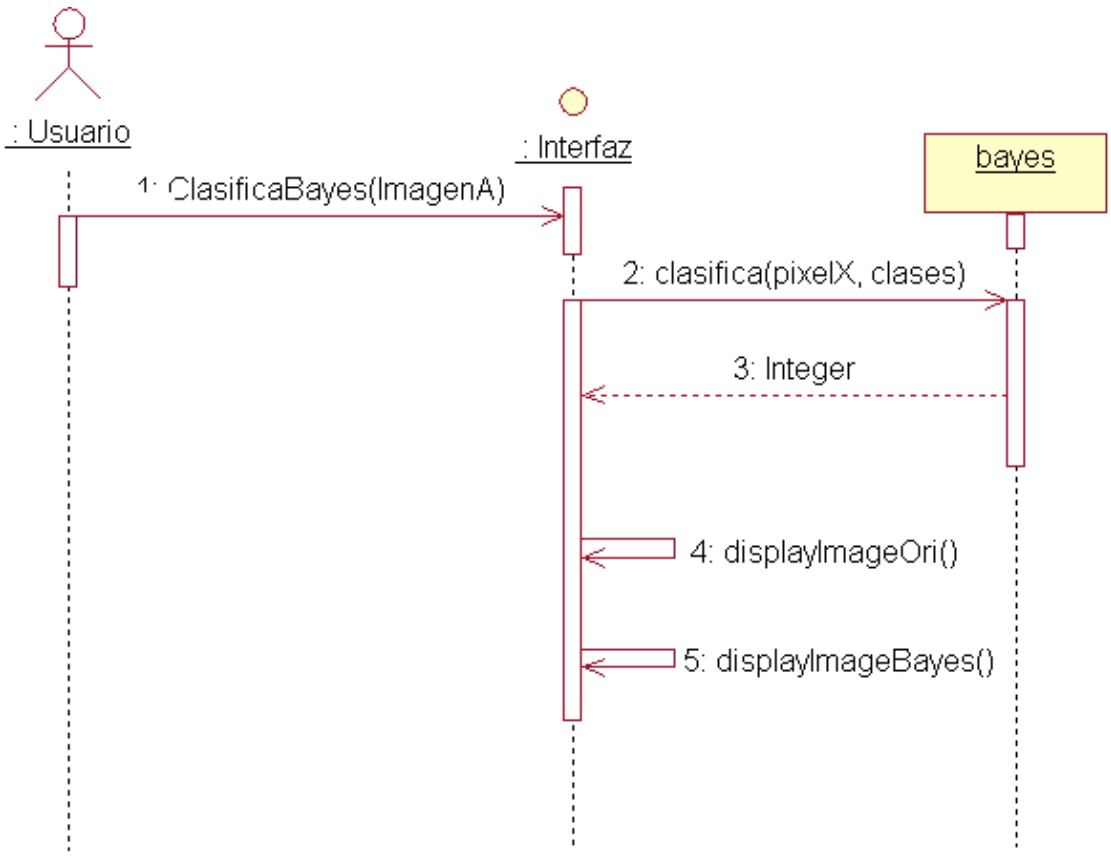


Figura 4.4 Diagrama de secuencia de la función de clasificación con el algoritmo de Bayes

Los puntos 2 y 3 relativos a la figura 4.4, conforman un bucle, el cual va transfiriendo uno a uno los píxeles de la imagen a clasificar al objeto *bayes*. Este último clasifica el píxel y devuelve un *integer* que identifica la clase a la cual pertenece.

Una vez clasificados todos los píxeles, se muestran tanto la imagen original como la imagen clasificada en su pestaña correspondiente.

De esta manera vemos cómo se realiza el proceso de comunicación entre los diferentes módulos de la aplicación.

5 CONSTRUCCIÓN

5.1 Introducción

El objetivo de la construcción como actividad del proyecto es la codificación de los componentes de la aplicación a partir de las especificaciones obtenidas en el proceso de diseño.

5.2 Construcción de los Algoritmos

Siguiendo la parte de organización, dividimos la construcción de los algoritmos en dos iteraciones.

En la primera iteración elaboramos la primera versión de los algoritmos. Para su realización nos basamos en códigos ya existentes de los algoritmos aunque escritos en otro lenguaje (Matlab). El principal problema fue determinar el tipo de estructuras de datos que íbamos a usar en JAVA.

En la segunda iteración mejoramos los algoritmos haciéndolos lo mas óptimos posibles para JAVA, poniendo especial atención en el tratamiento de errores y tratando de usar las herramientas de programación orientada a objetos.

Para la codificación del algoritmo de Cuantización vectorial no supervisado, no tuvimos grandes problemas, ya que es el más sencillo de todos y entendíamos a la perfección su mecanismo de funcionamiento.

Para la codificación de los otros dos algoritmos (Bayesiano y ventana de Parzen) la dificultad fue mayor, ya que decidimos hacer uso de la librería JAMA para los cálculos que involucraban matrices, requiriendo además mucha más información que en el caso del algoritmo anterior.

5.3 Construcción de la Interfaz

En la primera iteración de la construcción de la interfaz nos limitamos a poner las funcionalidades ofrecidas por los algoritmos y a mostrar las imágenes clasificadas en una pestaña. Para ello tuvimos que familiarizarnos un poco más con la librería JAI y con algunas funcionalidades de la librería AWT para poder mostrar las imágenes lo más eficientemente posible

En la segunda iteración nos concentramos en mejorar la interfaz. Principalmente nos propusimos como objetivo hacer la aplicación lo mas intuitiva posible, así como mostrar toda la información pertinente de una manera clara. En este sentido por ejemplo, pasamos de mostrar los colores de clasificación en formato numérico a mostrarlos en celdas con el color correspondiente, además las celdas muestran la

información numérica (valores RGB) al superponer el cursor del ratón. Por otra parte, creamos pestañas diferentes para los algoritmos y añadimos la información del estado del procesamiento de la imagen en la pestaña principal (tiempo transcurrido, porcentaje efectuado,…).

6 PRUEBAS

6.1 Introducción

Las pruebas las hemos programado en tres partes. Por un lado las pruebas individuales de los algoritmos. Éstas han sido realizadas dentro del desarrollo de los algoritmos (primera etapa de nuestro proyecto). Por otro lado se realizaron las pruebas pertinentes a la interfaz, las cuales fueron efectuadas en el momento del desarrollo de la mencionada interfaz (segunda etapa de nuestro proyecto). Y por último las pruebas finales del sistema y de instalación fueron realizadas al finalizar el desarrollo de la interfaz en su totalidad.

6.2 Pruebas de los Algoritmos

6.2.1 Pruebas Unitarias

En esta actividad se realizaron las pruebas unitarias de cada uno de los componentes del módulo Algoritmos. El objetivo es comprobar su correcta estructura y su ajuste a la funcionalidad establecida por los requisitos.

Para ello se crea un plan de pruebas donde se define el entorno necesario para la realización de cada nivel de prueba, así como las verificaciones asociadas a las pruebas unitarias, la coordinación y secuencia a seguir en la ejecución de las mismas y los criterios de registro y aceptación de los resultados.

Al probar los algoritmos nos centramos básicamente en su funcionalidad de clasificar píxeles correctamente, y en el aprendizaje. A cada algoritmo se le aplicaron varios problemas resueltos en libros, por lo tanto su verificación consistía en comparar los resultados del algoritmo con las soluciones que ya teníamos. Al principio encontramos algunos errores, pero fueron rápidamente identificados, de suerte que tras su corrección los módulos correspondientes cumplían con las funcionalidades requeridas.

6.2.2 Pruebas de Integración

El objetivo de las pruebas de integración consiste en verificar si los componentes interactúan correctamente a través de sus interfaces, tanto internas como externas, cubren la funcionalidad establecida, y se ajustan a los requisitos especificados para las verificaciones correspondientes.

Por lo tanto, tuvimos que poner a prueba la creación de todos los objetos de diseño y comprobar su comportamiento. Tuvimos que crear varios objetos de tipo algoritmos y varios objetos de tipo clases y comprobar que la interacción no producía errores.

6.3 Pruebas de la Interfaz

6.3.1 Pruebas Unitarias

En esta actividad probamos los diferentes módulos de la interfaz. Para ello creamos un plan de pruebas en el cual pusimos especial atención en el hecho de que las imágenes se mostraran de manera correcta en las diferentes pestañas.

Prácticamente toda la interfaz la creamos con la ayuda del editor visual de ventanas del entorno de desarrollo Netbeans, el cual ha sido suficientemente probado en numerosas aplicaciones, por lo tanto no hubo ningún problema en lo relativo a las pruebas unitarias.

La única parte fuera de lo común de la interfaz es la tabla de la pestaña principal que muestra los datos de las diferentes clases, puesto que tuvimos que implantar una tabla especial que permitiera añadir los colores de clasificación. Para ello cambiamos la creación de tablas estándares, no obstante no encontramos ningún problema en la nueva funcionalidad añadida, por consiguiente, las pruebas unitarias se realizaron con el éxito esperado.

6.3.2 Pruebas de Integración

Al estar prácticamente todos los módulos creados de manera automática mediante el editor de ventanas del entorno de desarrollo, no hubo ningún problema en las pruebas de integración puesto que todos los módulos se crean y se comunican perfectamente con arreglo a las especificaciones previstas.

6.4 Pruebas Finales

En las pruebas finales hemos incluido las pruebas al sistema y las pruebas de implantación.

El objetivo de las pruebas del sistema consiste en comprobar la integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen. Por lo tanto repasamos todos los requisitos paso a paso.

Al final de las pruebas del sistema hemos concluido que la aplicación reacciona con el comportamiento deseado, cumpliendo en su totalidad los requisitos que se propusieron al inicio del proyecto y posteriormente durante la revisión de los mismos.

Con respecto a las pruebas de instalación, hemos realizado varias pruebas en diferentes equipos de ordenadores para saber si la instalación se hacía correctamente y también para verificar si cambiaba algo el programa al ser ejecutado en otra máquina. Estas pruebas también fueron superadas con éxito por la aplicación desarrollada.

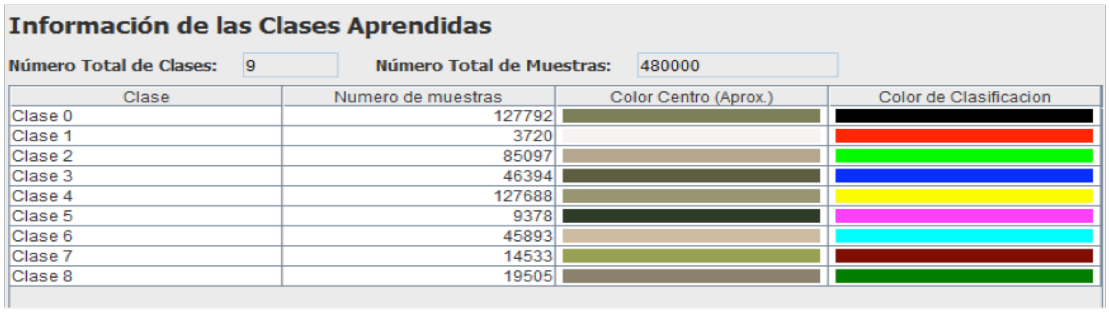
7 RESULTADOS DEL PROYECTO

7.1 Introducción

En este apartado mostraremos una serie de ejemplos relativos a los resultados obtenidos tras la ejecución de los diferentes algoritmos de clasificación y otras funcionalidades de nuestra aplicación.

7.2 Aplicación: Aprendizaje

A continuación mostramos un ejemplo de las clases creadas al entrenar con una imagen. En la figura 7.1 se muestra la información presentada en la pestaña principal, la cual muestra el número total de clases creadas, el número total de píxeles aprendidos, además para cada clase se muestra su número de muestras, el color de su centro y el color de clasificación.



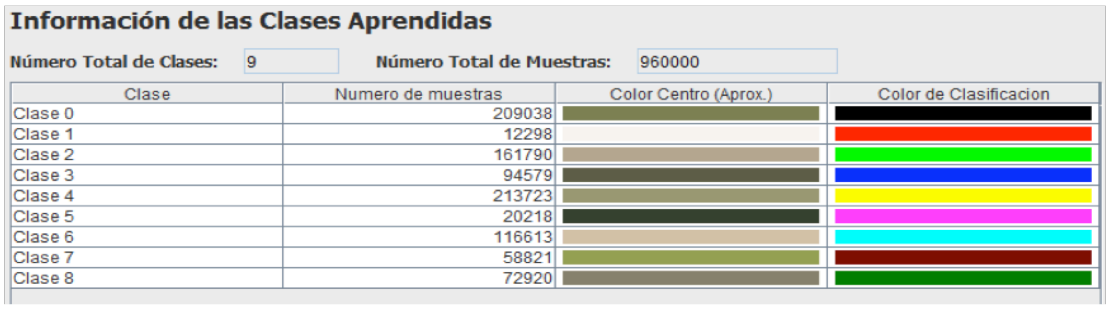
Información de las Clases Aprendidas

Número Total de Clases: 9 **Número Total de Muestras:** 480000

Clase	Numero de muestras	Color Centro (Aprox.)	Color de Clasificacion
Clase 0	127792		
Clase 1	3720		
Clase 2	85097		
Clase 3	46394		
Clase 4	127688		
Clase 5	9378		
Clase 6	45893		
Clase 7	14533		
Clase 8	19505		

Figura 7.1 Ejemplo de la información de las clases de la pestaña principal (primer aprendizaje)

A continuación, en la figura 7.2, mostramos cómo cambian las clases después de aprender otra imagen. En este ejemplo vemos que no se han creado nuevas clases, pero que el número de muestras se ha multiplicado por dos, ya que hemos aprendido una nueva imagen del mismo tamaño que la primera, por lo tanto, tenemos ahora el doble de píxeles.



Información de las Clases Aprendidas

Número Total de Clases: 9 **Número Total de Muestras:** 960000

Clase	Numero de muestras	Color Centro (Aprox.)	Color de Clasificacion
Clase 0	209038		
Clase 1	12298		
Clase 2	161790		
Clase 3	94579		
Clase 4	213723		
Clase 5	20218		
Clase 6	116613		
Clase 7	58821		
Clase 8	72920		

Figura 7.2 Ejemplo de la información de las clases de la pestaña principal (segundo aprendizaje)

7.3 Clasificación de Imágenes

7.3.1 Por Cuantización Vectorial No Supervisado

Usamos como *umbral* el valor 70 y mostramos la imagen original en la figura 7.3 mientras que la imagen clasificada con dicho parámetro aparece en la figura 7.4. Para ello se utiliza el aprendizaje realizado en el punto 7.2.

Figura 7.3 Imagen original

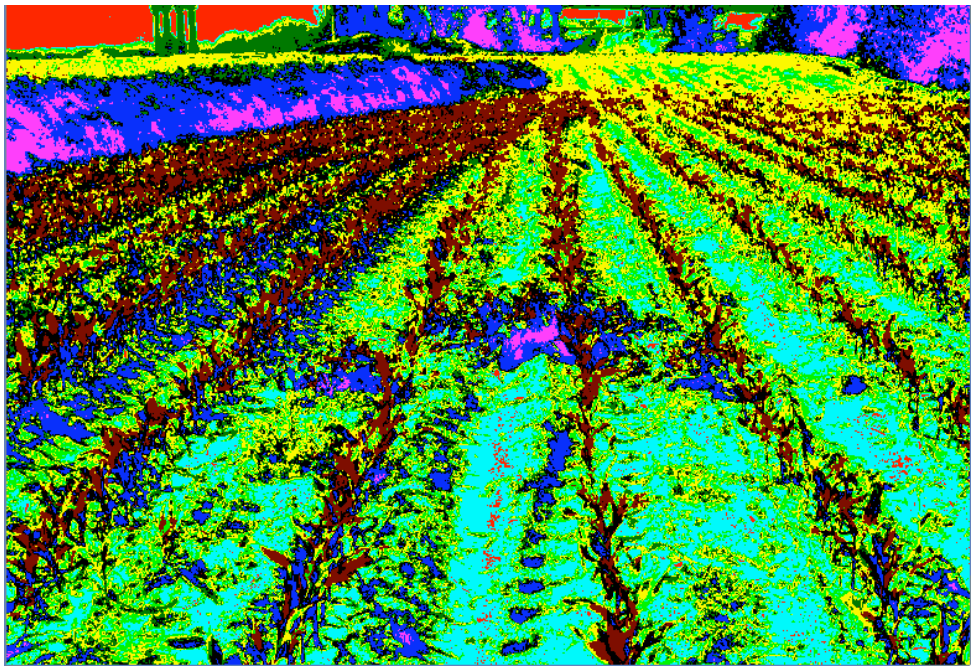


Figura 7.4 Imagen clasificada por cuantización vectorial no supervisado

7.3.2 *Por el Algoritmo de Bayes*

Mostramos la imagen original en la figura 7.5 y la clasificación en la figura 7.6. Para ello se utiliza el aprendizaje realizado en el punto 7.2.

Figura 7.5 Imagen original

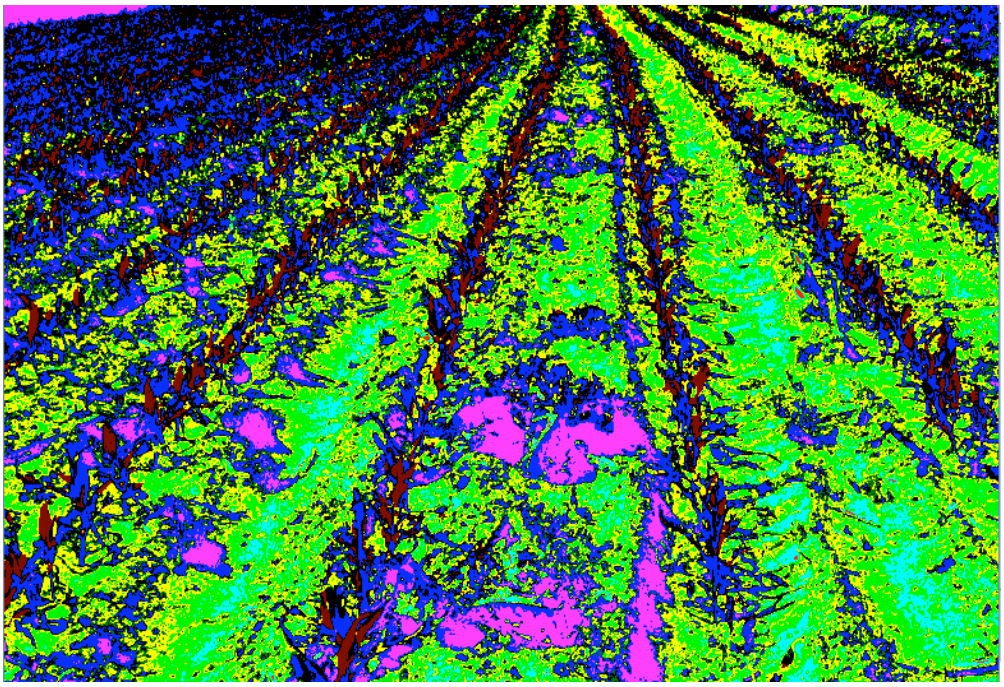


Figura 7.6 Imagen clasificada por el algoritmo de Bayes

7.3.3 *Por el Algoritmo ventana de Parzen*

Mostramos la imagen original en la figura 7.7 y la clasificación en la figura 7.8. Para ello se utiliza el aprendizaje realizado en el punto 7.2 con $h = 0.5$.

Figura 7.7 Imagen original

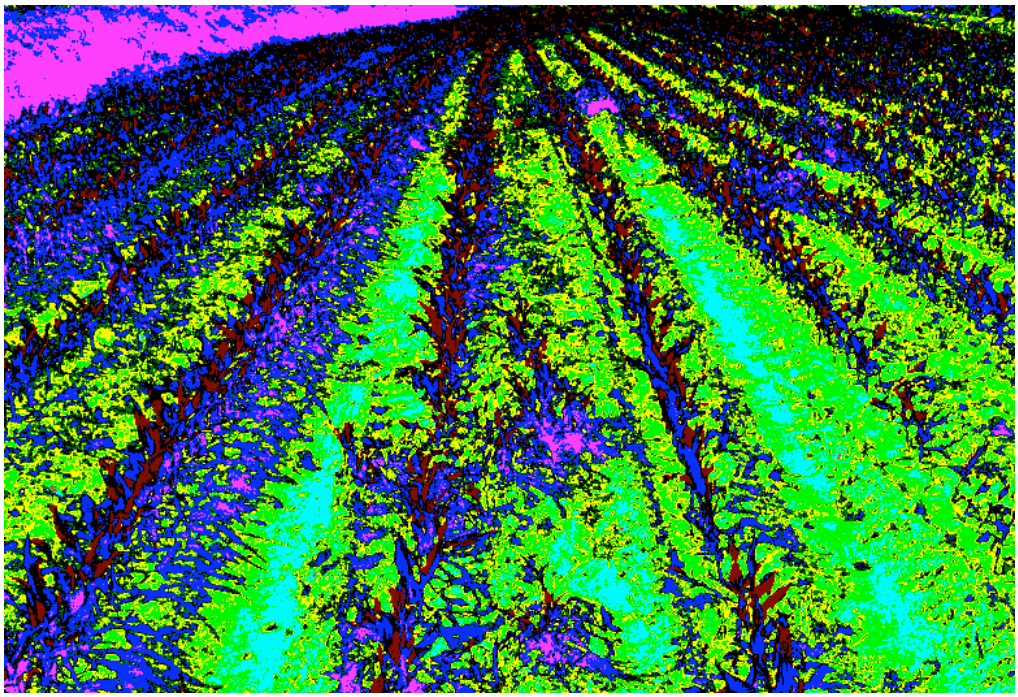


Figura 7.6 Imagen clasificada por el algoritmo de ventana de Parzen

7.4 Historial de Procesamientos

A continuación mostramos un ejemplo de un historial de clasificación en su versión *xml* en la figura 7.7 y otro ejemplo de historial en su versión hoja de cálculo (fichero xls) en la figura 7.8. En ambas versiones podemos ver para cada clasificación la información acerca del nombre de la imagen que se clasificó, su número de píxeles, el algoritmo utilizado, el tiempo de ejecución y la fecha completa del inicio del procesamiento.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <Historial>
  - <InfoProcesamiento>
      <Fecha>Fri May 29 16:58:14 PDT 2009</Fecha>
      <Imagen>05020001.JPG</Imagen>
      <Pixeles>480000</Pixeles>
      <Algoritmo>Cuantizacion Vectorial</Algoritmo>
      <Tiempo>2 seg</Tiempo>
    </InfoProcesamiento>
  - <InfoProcesamiento>
      <Fecha>Fri May 29 16:58:40 PDT 2009</Fecha>
      <Imagen>1.JPG</Imagen>
      <Pixeles>76800</Pixeles>
      <Algoritmo>Cuantizacion Vectorial</Algoritmo>
      <Tiempo>0 seg</Tiempo>
    </InfoProcesamiento>
  - <InfoProcesamiento>
      <Fecha>Sat Jun 06 18:59:41 PDT 2009</Fecha>
      <Imagen>05020007.JPG</Imagen>
      <Pixeles>480000</Pixeles>
      <Algoritmo>Cuantizacion Vectorial</Algoritmo>
      <Tiempo>1 seg</Tiempo>
    </InfoProcesamiento>
```

Figura 7.7 Versión xml del historial

Fecha	Imagen	Pixeles	Algoritmo	Tiempo
Fri May 29 16:58:14 PDT 2009	05020001.JPG	480000	Cuantizacion Vectorial	2 seg
Fri May 29 16:58:40 PDT 2009	1.JPG	76800	Cuantizacion Vectorial	0 seg
Sat Jun 06 18:59:41 PDT 2009	05020007.JPG	480000	Cuantizacion Vectorial	1 seg
Sat Jun 06 19:00:48 PDT 2009	05020021.JPG	480000	Bayes	35 seg
Sat Jun 06 19:07:27 PDT 2009	2.JPG	76800	Cuantizacion Vectorial	1 min y 32 seg
Sat Jun 06 19:22:10 PDT 2009	05020005.JPG	480000	Cuantizacion Vectorial	1 seg
Sat Jun 06 19:23:38 PDT 2009	05020011.JPG	480000	Bayes	13 seg
Sat Jun 06 19:24:45 PDT 2009	pruF.jpg	1862	Parzen	1 min y 42 seg

Figura 7.8 Versión hoja de cálculo del historial

8 FUTURO

La elaboración y el diseño de la aplicación han sido realizados pensando en el futuro, de cara a una posible ampliación o mejora de nuestro proyecto

La realización de una aplicación para un cliente particular como en nuestro caso es el Instituto de Automática Industrial del CSIC (Consejo Superior de Investigaciones Científicas), requería un diseño claro y estructurado para un fácil mantenimiento de la aplicación por personas ajenas a los desarrolladores de la aplicación. Además, en muchos casos es necesaria la modificación de la aplicación para adaptarse a nuevos requisitos y la ampliación para definir nuevas funcionalidades no incluidas en la aplicación inicial.

Las posibles modificaciones o ampliaciones que se le pueden añadir al proyecto elaborado son las siguientes:

- Ampliación del paquete identificado como *Algoritmos* incluyendo un nuevo algoritmo de clasificación. Esta inclusión sería trivial, debiendo añadir únicamente una clase implementando las dos funciones abstractas de la clase *Algoritmo*.

- Eliminación de un algoritmo de clasificación. Esta modificación no tiene ninguna complicación para el paquete principal, dado el diseño modular de éste.

- Integración del paquete elaborado en una interfaz distinta a la utilizada en la aplicación. El único requisito imprescindible para realizar la integración es que la interfaz esté desarrollada en Java o en un lenguaje de programación que permita hacer llamadas a métodos implementados en Java. Cumplido este requisito, la integración es sencilla y rápida.

- Ampliación o modificación de las propiedades de la clasificación. Los algoritmos utilizados clasifican las imágenes teniendo en cuenta las propiedades R,G,B de cada píxel. Si se desea realizar una clasificación mediante estos algoritmos, teniendo en cuenta otras propiedades o para otros objetos, únicamente se deberá introducir una clase que contenga estas propiedades y hacer una modificación mínima en los métodos de clasificación y entrenamiento. Esto es debido a que los métodos implementados de manera genérica, para poder ser utilizados para otros tipos de clasificaciones.

Además de lo expresado previamente, cualquier desarrollador y a instancia y requerimiento de un cliente puede realizar alguna ampliación más especifica dependiendo de su objetivo propuesto.

9 CONCLUSIONES

El proyecto, durante todas sus fases, ha seguido las consignas y paradigmas establecidos en ingeniería del software. Por lo tanto, hemos tratado de realizar todas las actividades siguiendo los patrones y pautas profesionales, imitando en este sentido los procesos que habitualmente se realizan en las empresas de desarrollo de software. En este sentido, se le ha otorgado la categoría de aplicación real en su vertiente comercial.

Dadas las exigencias del proyecto, hemos podido comprobar, con esta experiencia, cómo sería un desarrollo de este tamaño en un entorno profesional. Gracias a esto, hemos aprendido a comunicarnos entre sí, como si de un equipo de desarrollo se tratara (aunque en este caso de tamaño mínimo por sus dos componentes), el valor de una buena planificación y la importancia de llegar a nuestros objetivos en los plazos estipulados al principio.

Con respecto a los resultados de la aplicación, hemos constatado que los tres algoritmos se comportan de manera distinta, aunque todos proporcionan resultados aceptables. El algoritmo de cuantización vectorial es de lejos el más rápido computacionalmente hablando y proporciona unos resultados aceptables, similares a los obtenidos mediante el algoritmo de Bayes, el cual resulta un poco más lento en lo que a tiempos de ejecución se refiere. Por otro lado se sitúa el algoritmo de ventana de Parzen, el cual tiene un tiempo de ejecución mucho mayor que los anteriores, lo que limita bastante su uso cuando se trabaja con imágenes de un tamaño considerable.

En lo que se refiere al futuro de la aplicación, hemos tratado de escribir un código limpio y comentando para su fácil comprensión. De este modo cualquier tarea de mantenimiento o de modificación se puede realizar de una manera sencilla. Además, hemos acompañado la aplicación de un manual de usuario, el cual facilita a un nuevo usuario el manejo de la misma. Por otra parte, hemos tratado de hacer la aplicación lo más modular posible de forma y manera que la modificación de un algoritmo o el hecho de agregar un nuevo algoritmo de clasificación resulte una tarea fácil y no implique la modificación del resto de módulos que integran el proyecto.

10 AGRADECIMIENTOS

Queremos dar gracias a nuestro director del proyecto y profesor de la Facultad de Informática de la Universidad Complutense de Madrid, Don Gonzalo Pajares Martinsanz, por aceptarnos en su proyecto para la asignatura de Sistemas Informáticos, a pesar de que nos encontrábamos cursando nuestro último curso de la carrera en la Université de Montréal en Canadá.

11 BIBLIOGRAFÍA

[1] Álvarez Hernández, Álvaro; Fernández Buján, Ramón y Herraiz Molina, Antonio; Proyecto *"Clasificación de texturas naturales mediante técnicas de visión por computador"*, curso académico 2006/2007.

[2] Norvig, Peter y Russell, Stuart *ARTIFICIAL INTELLIGENCE: A MODERN APPROACH*. Prentice Hall Series in Artificial Intelligence 1995, Saddle River, New Jersey.

[3] Pajares, Gonzalo y J.M. de la Cruz, *Visión por Computador: Imágenes digitales y Aplicaciones*, RA-MA, 2001.

[4] Pajares, Gonzalo y J.M. de la Cruz, *Ejercicios Resueltos de Visión por Computador*, RA-MA, 2001.

[5] Pajares, Gonzalo *et al*. Articulo "*A new vision-based approach to differential sparring in precision agriculture*". Revista ELSEVIER, 2007.

[6] Vizcaíno Lamas, Javier; Toledo, Álvaro y Marco Rubio, Héctor; Proyecto *"Clasificación de texturas naturales mediante técnicas de visión por computador"*, curso académico 2006/2007.

[7] https://jai.dev.java.net/

[8] http://www.dimap.es/

[9] http://www.proespacio.org/

[10] http://www.ri.cmu.edu/research_project_detail.html?project_id=160&menu_id=261

ANEXO

INSTALACIÓN Y EJECUCIÓN

Windows

Para instalar y ejecutar la aplicación se necesitan unos mínimos conocimientos en informática.

En primer lugar se debe descomprimir el fichero *Clasificador_Imagenes.rar* en el destino deseado.

Después es necesario instalar la librería JAI (si ésta no se encuentra aún en su sistema) para el correcto funcionamiento de la herramienta. Para ello simplemente ejecutar el archivo *JAI_jdk.exe* y *JAI_jre.exe* que se encuentra en el directorio *Aplicación/Librerías/Windows* que se ha creado al descomprimir el fichero rar.

Una vez que las librerías de imágenes estén correctamente instaladas, lanzar la ejecución el archivo *Clasificador_Imagenes.jar*.

Unix

Como en el caso anterior, para instalar y ejecutar la aplicación se necesitan unos mínimos conocimientos en informática en este entorno.

En primer lugar se debe descomprimir el fichero *Clasificador_Imagenes.rar* en el destino deseado.

Después es necesario instalar la librería JAI (si ésta no se encuentra aún en su sistema) para el correcto funcionamiento de la herramienta. Para ello simplemente ejecutar el archivo *JAI_jdk.bin* y *JAI_jre.bin* que se encuentra en el directorio *Aplicación/Librerías/Unix* que se ha creado al descomprimir el fichero rar.

Una vez que la librería de imágenes esté correctamente instalada, lanzar a ejecución el archivo *Clasificador_Imagenes.jar*.

Mac OS X

Lamentablemente aún no es posible su ejecución en éste sistema operativo debido a que Mac OS X aún no tiene disponible ni la librería JAI ni la versión JDK1.6 necesarios para el correcto funcionamiento de la aplicación.

Nota: para más información acerca de la librería JAI →https://jai.dev.java.net/

MANUAL DE USUARIO

Inicio

Al iniciar la aplicación, sólo el menú ayuda y aplicación están activos. En el siguiente apartado explicaremos sus funcionalidades y las de los demás menús. Las pestañas están activas continuamente permitiendo navegar entre ellas siempre que se desee. No obstante, las pestañas imagen original, clasificación por cuantización vectorial, clasificación por Bayes y clasificación por Parzen sólo resulta de interés la navegación por ellas si se ha clasificado alguna imagen previamente, en cuyo caso podremos ver las imágenes que se están procesando.

En la pestaña principal encontramos los campos para la introducción de los parámetros de la aplicación, la información acerca del procesamiento actual y la información de las clases aprendidas.

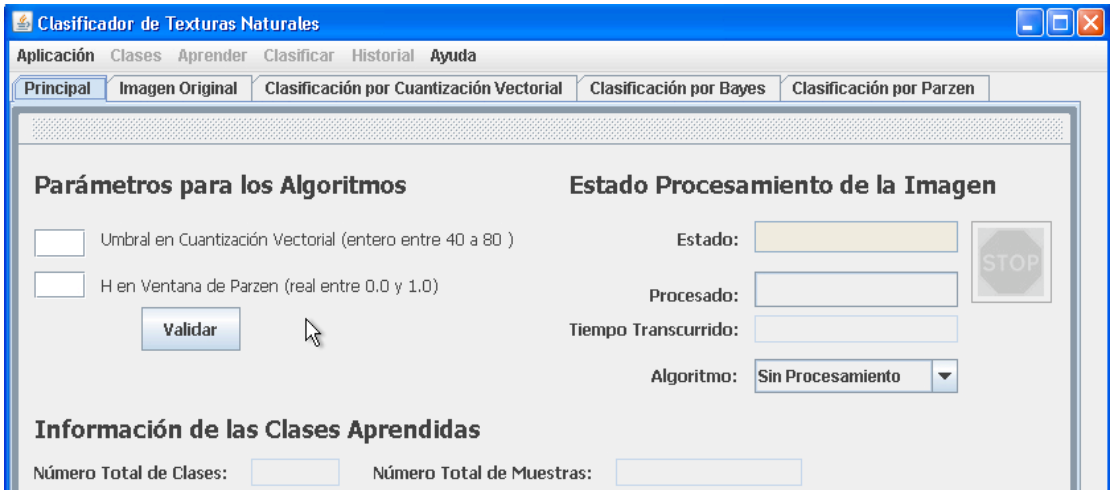


Figura A1 Captura de pantalla del estado inicial de la aplicación

Lo primera acción a realizar para poder usar la aplicación es introducir el valor de los parámetros *umbral* (para el algoritmo de cuantización vectorial) y *h* (para el algoritmo ventana de Parzen). En la figura A2 hemos introducido el valor 60 para el *umbral* y el valor 0.5 para *h*.

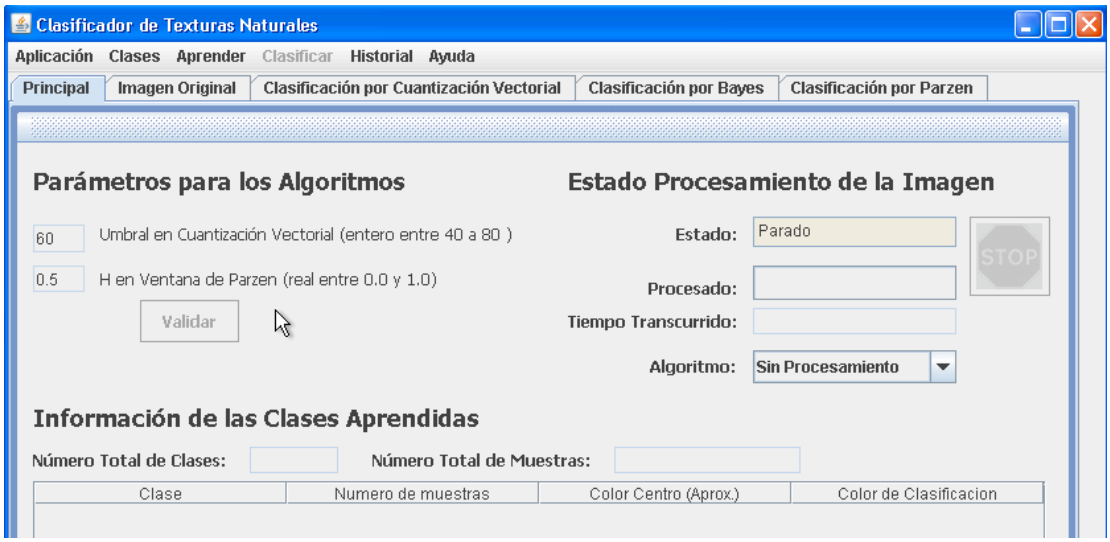


Figura A2 Estado de la aplicación después de validar los parámetros

Una vez validados los parámetros, todos los menús están activos a excepción del menú clasificar, que se activará cuando hayamos aprendido o cargado las clases en detrimento del menú aprender que se desactivará.

La pestaña principal de la figura A2 muestra diversa información importante a la hora de manejar la aplicación. Esta información se puede dividir en tres zonas o partes.

La primera zona muestra información de los valores de los parámetros *umbral* y *h*. Se puede modificar estos parámetros siempre que se desee a través del menú aplicación. Para que los valores queden fijados es necesario pulsar el botón validar, que sólo está activo en el caso en que no hayan sido aún validados los valores.

Figura A3 Parámetros antes de validar

Figura A4 Parámetros después de validar

La segunda zona muestra información del estado del procesamiento de una imagen durante la ejecución de la clasificación. En caso de no ser así, sólo muestra el estado parado.

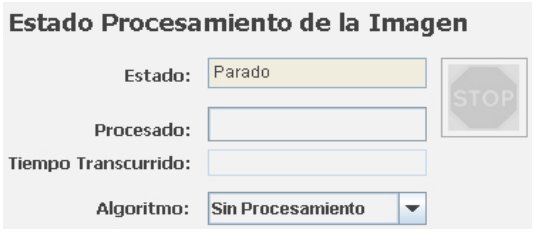


Figura A5 Procesamiento parado

Figura A6 Ejemplo de procesamiento en ejecución

La tercera zona muestra información importante, relativa a las clases aprendidas en tiempo actual, es decir que se va actualizando a medida que aprendemos o cargamos las clases.

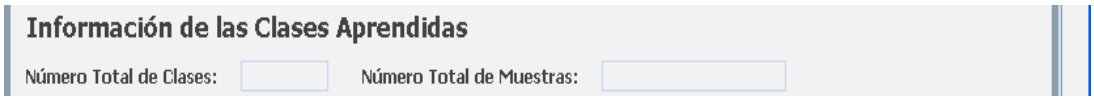


Figura A7 Información de las clases cuando no hemos aprendido ni cargado las clases

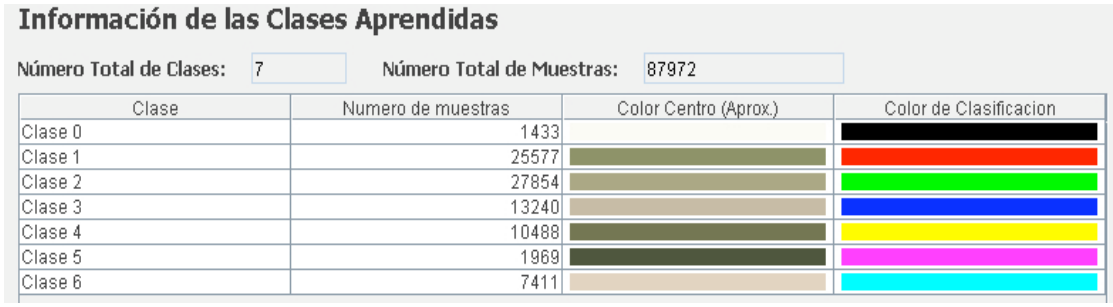


Figura A8 Ejemplo de información de las clases cuando ya hemos aprendido o cargado las clases

Menús

Aplicación

Figura A9 Captura de pantalla del menú aplicación

- Submenú **Acerca de…**: muestra una breve información del proyecto.
- Submenú **Modificar Parámetros**: permite cambiar el valor de los parámetros *umbral* y *h*, ambos necesarios para el funcionamiento de los algoritmos de clasificación.
- Submenú **Reiniciar**: simplemente reinicia la aplicación, es decir, elimina el valor de los parámetros *umbral* y *h* y vacía la información de las clases.
- Submenú **Salir**: como su nombre indica, cierra la aplicación.

Clases

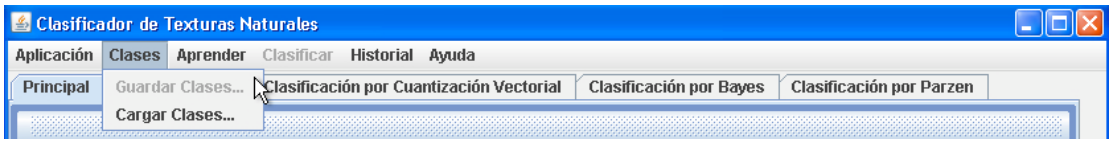


Figura A10 Captura de pantalla del menú clases

- Submenú **Guardar Clases…**: crea y guarda en un fichero del tipo *xml* las clases con sus centros y muestras correspondientes. Visible sólo si hay clases aprendidas o cargadas.
- Submenú **Cargar Clases…**: carga desde un fichero *xml* (con la estructura adecuada) las clases con sus centros y muestras.

Nota: Ejemplo de la estructura del fichero xml con la información de las clases guardadas.

```
<Clases_Aprendidas>
        <Clase> <Numero>0
                </Numero>
                <Centro>        <R>117.02015444015444</R>
                                <G>121.66096525096525</G>
                                <B>75.14378378378379</B>
                </Centro>
                <Muestras>      <Muestra>       <R>130</R>
                                                <G>141</G>
                                                <B>98</B>
                                </Muestra>
                                //resto de muestras
                </Muestras>
        </Clase>
        //resto de clases
</Clases_Aprendidas>
```

Aprender

Figura A11 Captura de pantalla del menú aprender

Esta funcionalidad sólo está activa cuando aún no hay clases aprendidas ni cargadas. Así pues, cargamos la imagen de la cual queremos aprender y extraer las clases con sus respectivas muestras y centros. Este aprendizaje se realiza mediante el algoritmo de clasificación cuantización vectorial no supervisado.

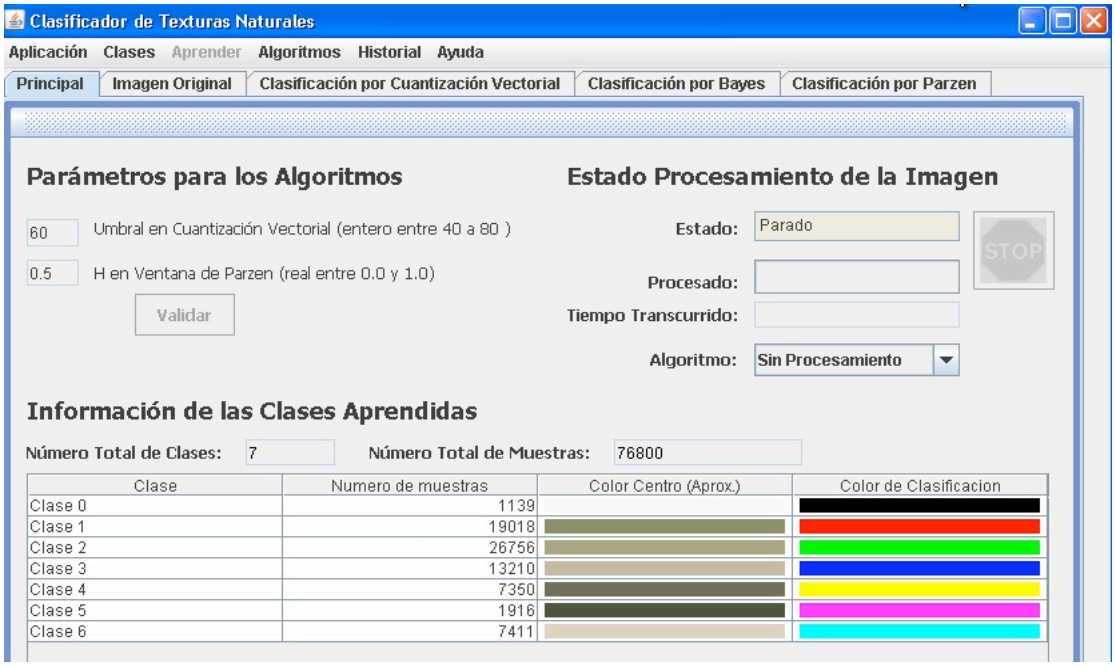


Figura A12 Captura de pantalla de la aplicación después de haber aprendido las clases

Clasificar

Figura A13 Captura de pantalla del menú clasificar

- Submenú **Cuantización Vectorial No Supervisado**: nos permite clasificar o clasificar y aprender una imagen mediante éste algoritmo de clasificación.
- Submenú **Bayes**: nos permite clasificar o clasificar y aprender una imagen mediante el algoritmo de clasificación paramétrico Bayesiano.

- Submenú **Ventana de Parzen**: nos permite clasificar o clasificar y aprender una imagen mediante el algoritmo de clasificación no paramétrico de la ventana de Parzen.

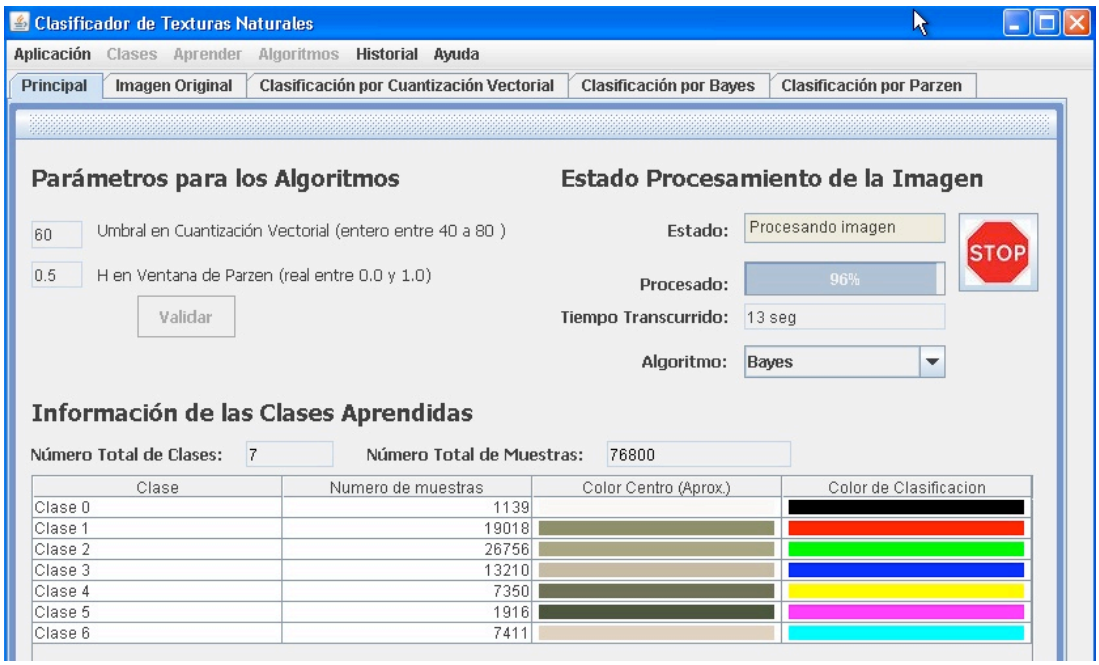


Figura A14 Captura de pantalla del transcurso del procesamiento de clasificación de una imagen mediante el algoritmo de Bayes

Una vez finalizado el procesamiento se muestra la imagen clasificada en la pestaña **Imagen Original** (figura A15) y en la pestaña de clasificación correspondiente se muestra el resultado de la clasificación (figura A16).

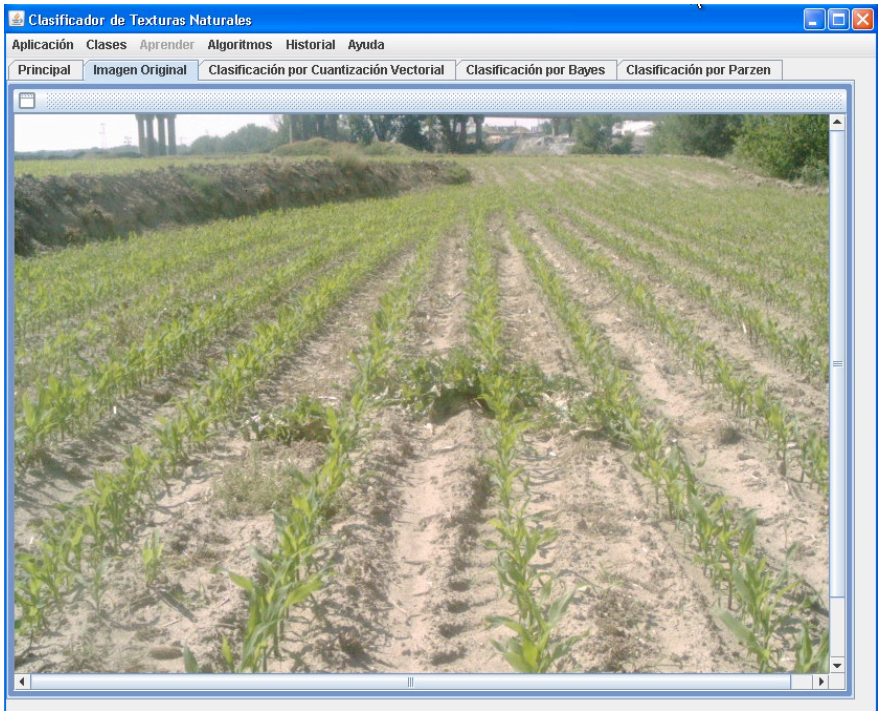


Figura A15 Pestaña Imagen Original

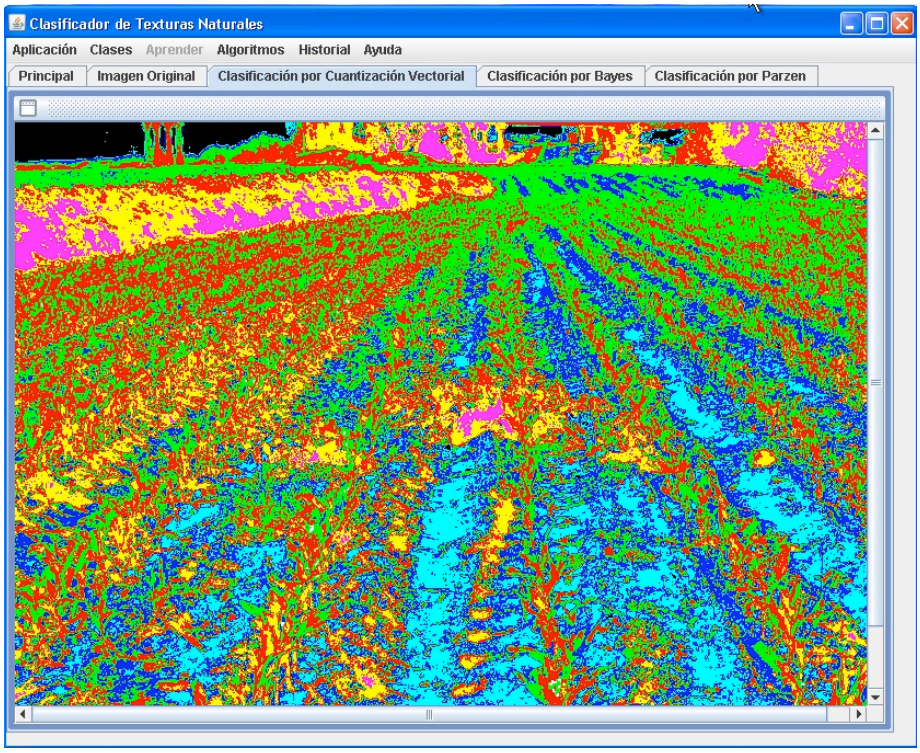


Figura A16 Ejemplo del resultado de clasificar una imagen mediante el algoritmo de cuantización vectorial.

Historial

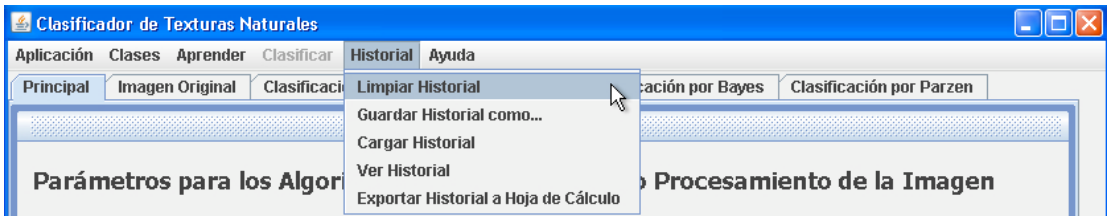


Figura A17 Captura de pantalla del menú historial

Una de las funcionalidades principales de la aplicación es el registro del historial, que guarda en el fichero *historial.xml* la información relativa a *todos* los procesamientos de imagen que han finalizado correctamente.

- Submenú **Limpiar Historial**: simplemente borra toda la información guardada en el historial.
- Submenú **Guardar Historial como…**: permite guardar el historial en otro fichero *xml* en el lugar y con el nombre deseado.
- Submenú **Cargar Historial…**: permite cargar en el historial la información desde un fichero *xml* con la estructura adecuada.
- Submenú **Ver Historial**: abre el archivo historial.xml para visualizar la información de nuestro historial de procesamientos.

- Submenú **Exportar Historial a Hoja de Cálculo**: para exportar un fichero *xml* con el historial a un fichero *xls*, extensión de los archivos de hojas de cálculo, para poder realizar análisis estadísticos de los procesamientos realizados.

Nota: Ejemplo de la estructura del fichero xml con la información de los procesamientos.

```
<Historial>
        <InfoProcesamiento>
                <Fecha>Fri May 29 16:58:14 PDT 2009</Fecha>
                <Imagen>05020001.JPG</Imagen>
                <Píxeles>480000</Píxeles>
                <Algoritmo>Cuantizacion Vectorial</Algoritmo>
                <Tiempo>2 seg</Tiempo>
        </InfoProcesamiento>
        //resto de procesamientos
</Historial>
```

Ayuda

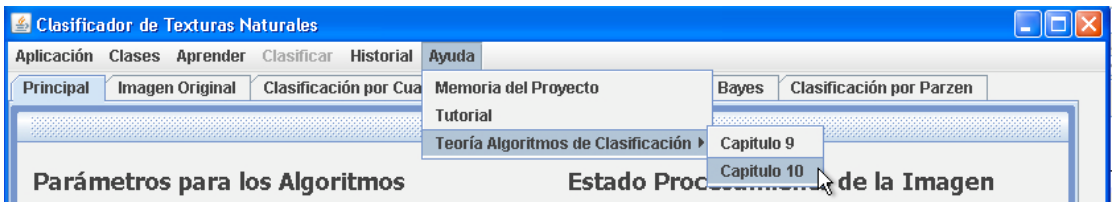


Figura A18 Captura de pantalla del menú ayuda

- Submenú **Memoria del Proyecto**: abre el fichero en el formato *pdf* protegido con el documento de esta memoria.
- Submenú **Tutorial**: abre el fichero en formato *pdf* protegido con el tutorial de la aplicación.
- Submenú **Teoría Algoritmos de Clasificación**: abre el fichero en formato pdf protegido con la teoría utilizada en el proyecto, extraída del libro de Gonzalo Pajares *Ejercicios Resueltos de Visión por Computador* [4].

CONTENIDO DEL CD

- Memoria del Proyecto

- Proyecto y código fuente (para abrir con Netbeans u otra plataforma de desarrollo en JAVA)

- Aplicación(véase Instalación y Ejecución más arriba en éste mismo Anexo)