
UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

SISTEMAS INFORMÁTICOS 2011/2012

# Implementación de un sistema de detección de verdes para agricultura de precisión

**Autores:**
Julio Ayllón Hidalgo
David Pérez Valenciano

**Profesores directores:**
Gonzalo Pajares Martinsanz
María Guijarro Mata-García

# Declaración de conformidad

Los alumnos:

Julio Ayllón Hidalgo, David Pérez Valenciano aquí firmantes autorizan a la Universidad Complutense de Madrid 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:

Julio Ayllón David Pérez

# RESUMEN

El proyecto consiste en la realización de una aplicación que permita el tratamiento y la manipulación de imágenes digitales, principalmente imágenes agrícolas, para su uso en entornos de agricultura de precisión.

El objetivo de esta aplicación es obtener las partes vegetales de dichas imágenes, utilizando para ello varias técnicas: extracción de las distintas componentes de la imagen, uso de índices cromáticos, binarización de imágenes.

Tras aplicar a una imagen de cultivo las distintas técnicas mencionadas anteriormente, segmentando la imagen en dos partes, se obtiene una imagen binaria formada por dos regiones claramente diferenciadas, una de ellas compuesta por aquellos píxeles que contienen características verdes o vegetales, y la otra parte por aquellos píxeles pertenecientes a las partes consideradas no verdes de la imagen.

Palabras clave: **histograma, índice cromático, binarización, método de Otsu, componentes de una imagen**.

# ABSTRACT

This project consists in developing an application that allows the treatment and the manipulation of digital images, principally agricultural images, to be used in environments of precision agriculture.

The objective is to obtain the vegetable parts of the images, using for it several technologies: extraction of the different image components, use of chromatic indexes and image binarization.

After applying these technologies, the image is segmented into two parts, a binary image is obtained formed by two clearly differentiated regions, one of them composed by pixels that contain green or vegetable characteristics, and the other one composed by pixels that belong to non-green parts of the image.

Keywords: **histogram, chromaticity index, binarization, Otsu's method, components of an image**

# Índice

**RESUMEN** ..... **4**

**ABSTRACT** ..... **5**

**ÍNDICE** ..... **6**

**CAPITULO 1: INTRODUCCIÓN** ..... **9**

1.1 MOTIVACIÓN DEL PROYECTO ..... 9
1.2 OBJETIVOS ..... 10
1.3 ORGANIZACIÓN DE LA MEMORIA ..... 10

**CAPITULO 2: CONCEPTOS TEÓRICOS** ..... **13**

2.1 IMÁGENES DIGITALES ..... 13
2.2 EL HISTOGRAMA ..... 15
2.3 ALGORITMOS DE DETECCIÓN DE VERDES ..... 17
*2.3.1 Normalización del espacio del color* ..... *17*
*2.3.2 Índice exceso de verdes* ..... *18*
*2.3.3 Índice exceso de rojos* ..... *18*
*2.3.4 Diferencia del exceso de verde con el exceso de rojo* ..... *18*
*2.3.5 Extracción del índice de vegetación* ..... *19*
*2.3.6 Vegetative* ..... *19*
*2.3.7 Resumen* ..... *19*
2.4 BINARIZACIÓN DE IMÁGENES ..... 20
*2.4.1 Binarización mediante el método de Otsu* ..... *21*
*2.4.2 Binarización mediante la media* ..... *22*

**CAPITULO 3: FASE DE ANÁLISIS** ..... **23**

3.1 ANÁLISIS DE REQUISITOS ..... 26
*3.1.1 Requisitos funcionales* ..... *26*
*3.1.2 Otros requisitos* ..... *27*
3.2 DIAGRAMA DE CASOS DE USO ..... 27
*3.2.1 Especificación casos de uso* ..... *28*
3.3 RIESGOS ..... 35
*3.3.1 Riesgos tecnológicos* ..... *35*
*3.3.2 Riesgos organizativos y de personal* ..... *36*

**CAPITULO 4: FASE DE DISEÑO** ..... **36**

4.1 ARQUITECTURA MODELO-VISTA-CONTROLADOR ..... 37
4.2 DIAGRAMA DE PAQUETES ..... 39
4.3 DIAGRAMAS DE CLASES ..... 40
4.4 DIAGRAMAS DE SECUENCIA ..... 43

**CAPITULO 5: FASE DE IMPLEMENTACIÓN .....47**

5.1 LENGUAJE DE PROGRAMACIÓN: JAVA ..... 47
*5.1.1 Características de Java* ..... *47*
5.2 LIBRERÍAS ..... 48
*5.2.1 JMathPlot* ..... *48*
*5.2.2 JAI* ..... *49*
5.3 HERRAMIENTAS DE TRABAJO ..... 49
5.4 PROBLEMAS Y RESOLUCIÓN ..... 50

**CAPITULO 6: FASE DE PRUEBAS .....53**

6.1 PRUEBA 1: CARGA Y RECORTE DE IMÁGENES ..... 53
6.2 PRUEBA 2: ALGORITMOS DE EXTRACCIÓN DE VERDES ..... 55
6.3 PRUEBA 3: BINARIZACIÓN DE IMÁGENES ..... 58
6.4 PRUEBA 4: EXTRACCIÓN DE COMPONENTES ESPECTRALES ..... 59

**CAPÍTULO 7: CONCLUSIONES .....61**

**CAPÍTULO 8: TRABAJO FUTURO .....63**

8.1 FUTURO DE LA APLICACIÓN ..... 63

**CAPÍTULO 9: BIBLIOGRAFÍA .....65**

**ANEXO I: CONTENIDO DEL CD-ROM E INSTALACIÓN DE LA APLICACIÓN. .....67**

CONTENIDO DEL CD-ROM ..... 67
INSTALACIÓN Y EJECUCIÓN: ..... 68

**ANEXO II: MANUAL DE USUARIO .....69**

VENTANA INICIO ..... 69
CARGAR IMAGEN ..... 70
VENTANA PRINCIPAL ..... 72
*RECORTE IMAGEN* ..... *72*
*BARRA HERRRAMIENTAS* ..... *72*
OTRAS CONSIDERACIONES ..... 78

# CAPITULO 1: INTRODUCCIÓN

## 1.1 Motivación del proyecto

Hoy en día el trabajo de precisión en la agricultura se hace necesario para la optimización de recursos y la reducción de costes en este sector. La incorporación de las nuevas tecnologías orientadas a mejorar la producción agrícola se ve necesaria y casi imprescindible para un desarrollo acorde con los tiempos que vivimos. El uso de todas estas técnicas y otras muchas se encuentran englobadas en un concepto denominado Agricultura de Precisión.

Es habitual el uso de pesticidas y herbicidas a la hora de eliminar malas hierbas en los campos de cultivo, lo que supone un aumento de la contaminación medioambiental y un grave perjuicio a la atmósfera. Por ello tiene lógica la idea de detectar estas malas hierbas para poder tratarlas de una manera mucho más adecuada y eficiente, reduciendo así los costes de producción. Se podría conseguir un uso de pesticidas en cantidades mucho menores si se aplicaran sólo en las zonas donde se encuentren las malas hierbas mencionadas anteriormente.

Para conseguirlo nos serviremos del tratamiento de imágenes, el cuál ha conocido un desarrollo espectacular en los últimos años y constituye uno de los sectores de investigación más activos en lo que se ha dado en llamar Agricultura de Precisión.

Una primera aproximación para poder llevarlo a cabo es conseguir detectar las zonas verdes de una imagen tomada en un campo de cultivo, y aquí es donde se encuentran los objetivos del presente proyecto: conseguir identificar esas regiones del terreno es el primer paso para poder reconocer las malas hierbas y así aplicar los herbicidas donde realmente se necesitan.

Este proyecto se centra en el aspecto mencionado con anterioridad: separar las zonas verdes tanto de cultivo como de malas hierbas, del resto de las zonas del campo de cultivo, informando del porcentaje de esta cantidad de verde, lo que nos puede dar una idea inicial de si será necesario o no aplicar pesticidas. Aunque pueda no ser determinante para una decisión definitiva y necesite ser estudiado con mucho más detalle, sí que puede ayudar para realizar un estudio más detallado en el futuro.

## 1.2 Objetivos

Como ya se ha explicado anteriormente, el proyecto consiste en realizar una aplicación que permita detectar las zonas verdes en una imagen agrícola.

Dentro de ese objetivo general, describimos los siguientes objetivos específicos:

1) Extraer componentes roja, verde y azul de la imagen original, mostrando el histograma de cada una de ellas.

2) Implementación de los siguientes algoritmos de detección de verdes o índices cromáticos ExG, ExGR, CIVE y VeG (Guijarro y col. 2011), obteniendo una imagen en escala de grises.

3) Métodos de umbralización de esas imágenes en escala de grises obteniendo una imagen binarizada (blanco/negro). Exactamente, dichos métodos son la media y Otsu (1979). Adicionalmente se propone el desarrollo de un método de binarización en el que se podrá elegir el umbral manualmente, en contraposición con los otros dos métodos automáticos.

4) Implementación de una interfaz de usuario, que permita cargar una imagen de campo, y aplicar las técnicas mencionadas anteriormente, tanto de identificación de verdes como de umbralización.

## 1.3 Organización de la memoria

Tras el resumen previo, junto con la motivación y los objetivos del proyecto, a continuación se describe brevemente la organización de la memoria.

En el capítulo 2 se explican una serie de conceptos teóricos que han sido necesarios para poder desarrollar la aplicación, y sobre los que se basa ésta.

En el capítulo 3 se detalla todo el trabajo realizado durante la fase de análisis: captura de requisitos, realización y explicación del diagrama de casos de uso, comentarios sobre el modelo de proceso de desarrollo de software seleccionado, y análisis de riesgos.

En el apartado 4 se justifica la labor realizada en la etapa de diseño,

explicando el uso de patrón Modelo–Vista–Controlador, ilustrando el diagrama de clases de la aplicación propuesta, además de mostrar diferentes diagramas de secuencia.

En el capítulo 5 se detalla la fase de implementación, comentando las herramientas y tecnologías utilizadas, así como una mención expresa sobre las decisiones tomadas durante el desarrollo de la aplicación.

En el apartado 6, se explican y se detallan las pruebas realizadas para la validación de la aplicación desarrollada.

En los capítulos 7, 8 y 9 se detallan las conclusiones, el trabajo futuro propuesto, y la bibliografía respectivamente.

La memoria concluye con dos Anexos, en el primero se explica el contenido del CD-ROM adjunto, mostrando un manual de ejecución de la aplicación, y en el segundo se incluye el manual de usuario, que servirá de ayuda en el momento de utilizar la aplicación.

# CAPITULO 2: Conceptos Teóricos

En este capítulo se introducen algunos de los conceptos teóricos y términos frecuentemente utilizados en el ámbito de la aplicación. Obviamente, al ser un proyecto en el que se están tratando imágenes digitales, se explica dicho concepto, cómo se representan las imágenes y cuál es su contenido, así como la descripción de un histograma, que constituye un elemento esencial de los métodos de procesamiento que se incluyen en la aplicación. Además se ilustran los distintos tipos de algoritmos utilizados en la extracción de verdes en Agricultura de Precisión, finalizando con el concepto de binarización de imágenes.

## 2.1 Imágenes digitales

Las imágenes digitales son capturadas mediante los dispositivos apropiados para su almacenamiento en una computadora. La figura 2.1 representa un proceso de captura de una escena tridimensional, Pajares y Cruz (2007).

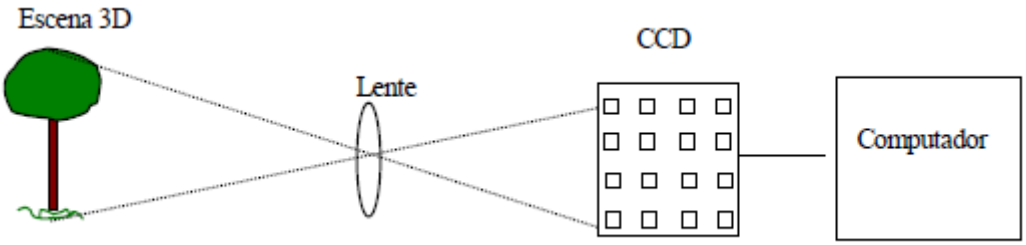



**Figura 2.1** Captura de una imagen 3D por un dispositivo CCD.

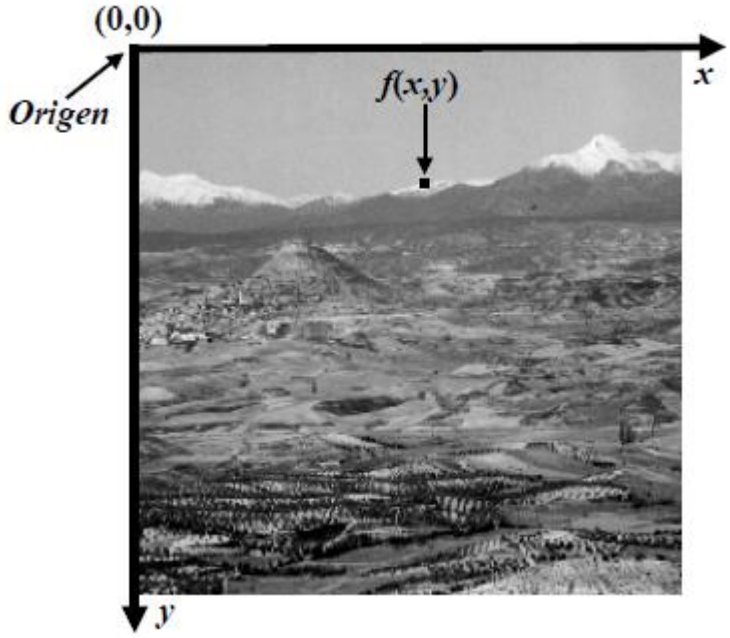
El proceso finaliza con el almacenamiento de la imagen en el computador. Este almacenamiento se lleva a cabo a través de algún formato característico de imágenes (TIFF, BMP, JPEG).

La imagen se representa desde el punto de vista de su tratamiento computacional como una matriz numérica de dimensión MxN, es decir, con M filas y N columnas. El contenido de esa matriz se refiere a valores enteros situados en localizaciones espaciales (x,y) denominados píxeles, dicho valor es el resultado de la cuantización de intensidad o nivel de gris.

Si la imagen es en blanco y negro, se almacena un valor por cada píxel. Este valor es el nivel de intensidad o nivel de gris comentado anteriormente. Se suele utilizar un rango de valores para su representación, que generalmente es de 0 a $2^{n-1}$. Uno de los valores más utilizados con n igual a

8; esto significa que el rango de valores para este caso varía entre 0 y 255. En este caso, el 0 representa el negro absoluto y el 255, el blanco absoluto. Esto indica que podemos tener una resolución o precisión en los grises de 256. El hecho de utilizar 256 niveles es porque con 8 bits del computador se pueden codificar 256 valores distintos desde la combinación 00000000, que representa el nivel 0, hasta la combinación 11111111, que representa el nivel 255.

Sobre cada matriz se establece un sistema de coordenadas con origen normalmente en la esquina superior izquierda y con los ejes *x* e *y* tales que su orientación es positiva hacia la derecha en el caso del eje *x* y hacia abajo en el caso del eje *y,* figura 2.2.




$$I(x,y) = \begin{bmatrix} 2 & 5 & 8 & 0 \\ 2 & 2 & 9 & 1 \\ 2 & 4 & 6 & 3 \\ 2 & 4 & 4 & 4 \end{bmatrix}$$

**Figura 2.2** Representación de una imagen digital en grises.

Con esta disposición de imágenes se tiene acceso a los valores de los píxeles en sus diferentes localizaciones, siendo el convenio de acceso el que se expresa seguidamente con relación a la figura 2.2:

$$I(x,y)=I(2,1)=9$$

Cuando la imagen es en color, para cada localización espacial existen tres valores de intensidad asociados, es decir, los elementos de la matriz vienen dados por tres valores que representan cada uno de los componentes básicos del color en cuestión. Estos componentes son el rojo (R), verde (G) y azul (B) y es lo que conocemos como código RGB. En este caso el conjunto de valores (0, 0, 0) representa al negro, mientras que los valores (255, 255, 255) es el blanco absoluto. La combinación de distintos valores representa otros colores. Debido a lo anterior, una imagen en color posee tres bandas espectrales: rojo, verde, azul; cada una de ellas viene representada por una matriz de números con valores en el rango 0 a 255 para imágenes de 8 bits. Una imagen de color vendría dada por las tres subimágenes de la figura 2.3,

$$R(x,y)=\begin{bmatrix}2&5&8&0\\2&2&9&1\\2&4&6&3\\2&4&4&4\end{bmatrix};\ G(x,y)=\begin{bmatrix}8&4&4&3\\8&3&4&0\\6&9&1&6\\6&9&3&5\end{bmatrix};\ B(x,y)=\begin{bmatrix}7&6&7&3\\7&6&8&4\\1&9&7&2\\3&9&9&6\end{bmatrix}$$

**Figura 2.3** Representación de una imagen digital en color: Rojo(R), Verde(G) y Azul(B).

El píxel que se encuentra en la localización espacial (x,y) posee componentes (r, g, b). Por ejemplo, dada la imagen de la figura 2.3 los valores del píxel en (x,y)=(2,3) serían los siguientes: (4, 3, 9)

Los tres valores de cada píxel para las imágenes en color corresponden al modelo de color RGB.

## 2.2 El histograma

El histograma de una imagen es una función discreta que representa el número de píxeles en la imagen en función de los niveles de intensidad, g. La probabilidad de ocurrencia de un determinado nivel se define como,

$$P(g)=\frac{N(g)}{M}$$

donde M es el número de píxeles en la imagen y N(g) es el número de píxeles en el nivel de intensidad g. Como con cualquier distribución de probabilidad todos los valores de P(g) son menores o iguales que 1 y la suma de todos los valores de P(g) es 1.

El histograma de una imagen es una herramienta visual de gran utilidad para el estudio de imágenes digitales, que puede proporcionar una idea muy aproximada de la distribución de niveles de gris.

Las intensidades o niveles de gris están representadas a lo largo del eje X y el número de ocurrencias para cada intensidad se representan en el eje Y.

Por ejemplo, dada la imagen de la figura 2.4 (a) con 10 niveles de gris, del 0 al 9; su histograma se muestra en (b). En la figura 2.5 (a) se muestra una imagen de grises y en (b) su histograma de los niveles de gris, del 0 a 255.

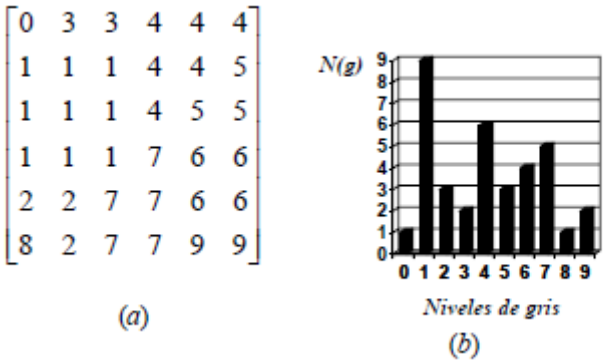



**Figura 2.4** (a) Imagen con 10 niveles de gris del 0 al 9; (b) Su histograma.

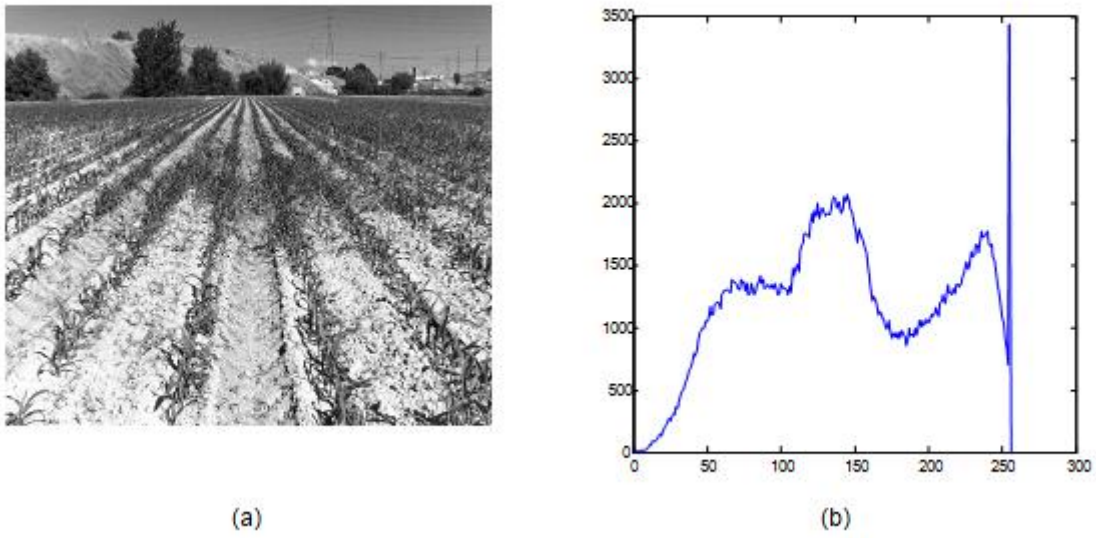



**Figura 2.5** (a) Imagen original de grises; (b) Su histograma.

## 2.3 Algoritmos de detección de verdes

Estos algoritmos constituyen una estrategia para identificar las diferentes partes de verdes de una imagen original, transformando la imagen en el espacio RGB a una en escala de grises, para proceder posteriormente a su binarización, como se describe posteriormente.

Los métodos evaluados y que se han seleccionado para su implementación han sido los siguientes:

- Exceso de verde (ExG).
- Diferencia del exceso de verde con el exceso de rojo (ExGR), en el que se utiliza tanto el algoritmo mencionado en el punto anterior, como el algoritmo llamado exceso de rojo (ExR).
- Extracción del índice de vegetación (color index of vegetation extraction).
- Vegetative (VEG).

Más adelante, se explican cada uno de ellos con más detalle.

### 2.3.1 Normalización del espacio del color

Antes de calcular los índices cromáticos de vegetación, es necesario realizar una normalización del espacio del color. Inicialmente partimos de una imagen que se encuentra en el espacio de color RGB, donde cada píxel está formado por tres valores, los correspondientes a los canales rojo(R), verde(G) y azul(B), dichos valores se sitúan en el rango de 0 a 255, donde 0 se corresponde con el negro, y 255 con el blanco. A esta imagen la aplicamos el siguiente proceso de normalización:

- canal rojo, $r = \frac{R}{R+G+B}$
- canal verde, $g = \frac{R}{R+G+B}$
- canal azul, $b = \frac{R}{R+G+B}$

donde R, G y B son las componentes espectrales en el modelo de color RGB en el rango [0,255] obtenidas de la siguiente forma:

$$R = \frac{R}{R_{max}} \qquad G = \frac{G}{G_{max}} \qquad B = \frac{B}{B_{max}}$$

donde $R_{max}$ , $G_{max}$ , $B_{max}$ tienen un valor de 255.

Tras haber aplicado la normalización del espacio del color a las imágenes agrícolas, como se explica anteriormente, es necesario utilizar los mencionados índices cromáticos para poder segmentar las partes verdes de la imagen del resto. Estos algoritmos se sirven de las componentes de la imagen procedentes del espectro visible.

En los siguientes apartados se explican cada uno de los procedimientos por los que se obtienen las ecuaciones necesarias para la detección de verdes.

### 2.3.2 Índice exceso de verdes

Dicho índice, también conocido como Excess Green Index (ExG), se define como sigue:

$$ExG= 2g – r – b$$

El contenido de cada uno de los nuevos píxeles de la imagen que se forma será el resultado de multiplicar por 2 el valor de la componente verde del mismo píxel, y restar a éste el contenido de las componentes roja y azul del píxel situado en la posición que estamos tratando en la imagen RGB. El resultado es por tanto una imagen en escala de grises que se identifica como ExG.

### 2.3.3 Índice exceso de rojos

El índice Excess Red Index (ExR) o exceso de rojos se define como:

$$ExR= 1.4r – g$$

El contenido de cada uno de los nuevos píxeles de la imagen que se forma será el resultado de multiplicar por 1.4 el valor de la componente roja del mismo píxel, y restar a éste el contenido de las componentes verde del píxel situado en la posición que estamos tratando en la imagen RGB.

### 2.3.4 Diferencia del exceso de verde con el exceso de rojo

Este índice conocido como Excess Green minus Excess Red index (ExGR) se define como sigue,

$$ExGR= ExG – ExR$$

El contenido de cada uno de los nuevos píxeles de la imagen que se forma será el resultado de restar el valor obtenido al calcular el índice ExG y el conseguido al aplicar ExR.

### 2.3.5 Extracción del índice de vegetación

Este índice conocido como Color Index of Vegetation Extraction (CIVE), se define como sigue:

CIVE= 0.441r – 0.811g + 0.585b + 18.78745

El contenido de cada uno de los nuevos píxeles de la imagen que se forma será el resultado de multiplicar por 0.441 el valor de la componente roja del mismo píxel, restar a éste el contenido de la componente verde multiplicado por 0.811, sumar el valor obtenido de la componente azul del píxel multiplicado por 0.585 situado en la posición que estamos tratando en la imagen RGB, y sumar el valor 18.78745.

### 2.3.6 Vegetative

El índice Vegetative (VEG) se define como:

$VEG = \frac{g}{r^a b^{1-a}}$, siendo a =0.667

El contenido de cada uno de los nuevos píxeles de la imagen que se forma será el resultado de dividir el valor de la componente verde del mismo píxel, entre el resultado obtenido al multiplicar el contenido de la componente roja elevado al número a por el valor azul elevado a 1 - a del píxel situado en la posición que estamos tratando en la imagen RGB.

Tras aplicar las ecuaciones correspondientes a la imagen original, se obtiene una imagen en escala de grises.

### 2.3.7 Resumen

Las imágenes mostradas en la figura 2.6, son el resultado de la aplicación de los índices que se indican.

| Índice | Ecuación |
|---|---|
| **ExG** | 2g – r – b |
| **ExR** | 1.4 – r |
| **ExGR** | ExG – ExR |
| **CIVE** | 0.441r – 0.811g + 0.585b + 18.78745 |
| **VEGETATIVE** | $g/r^a b^{1-a}$ |

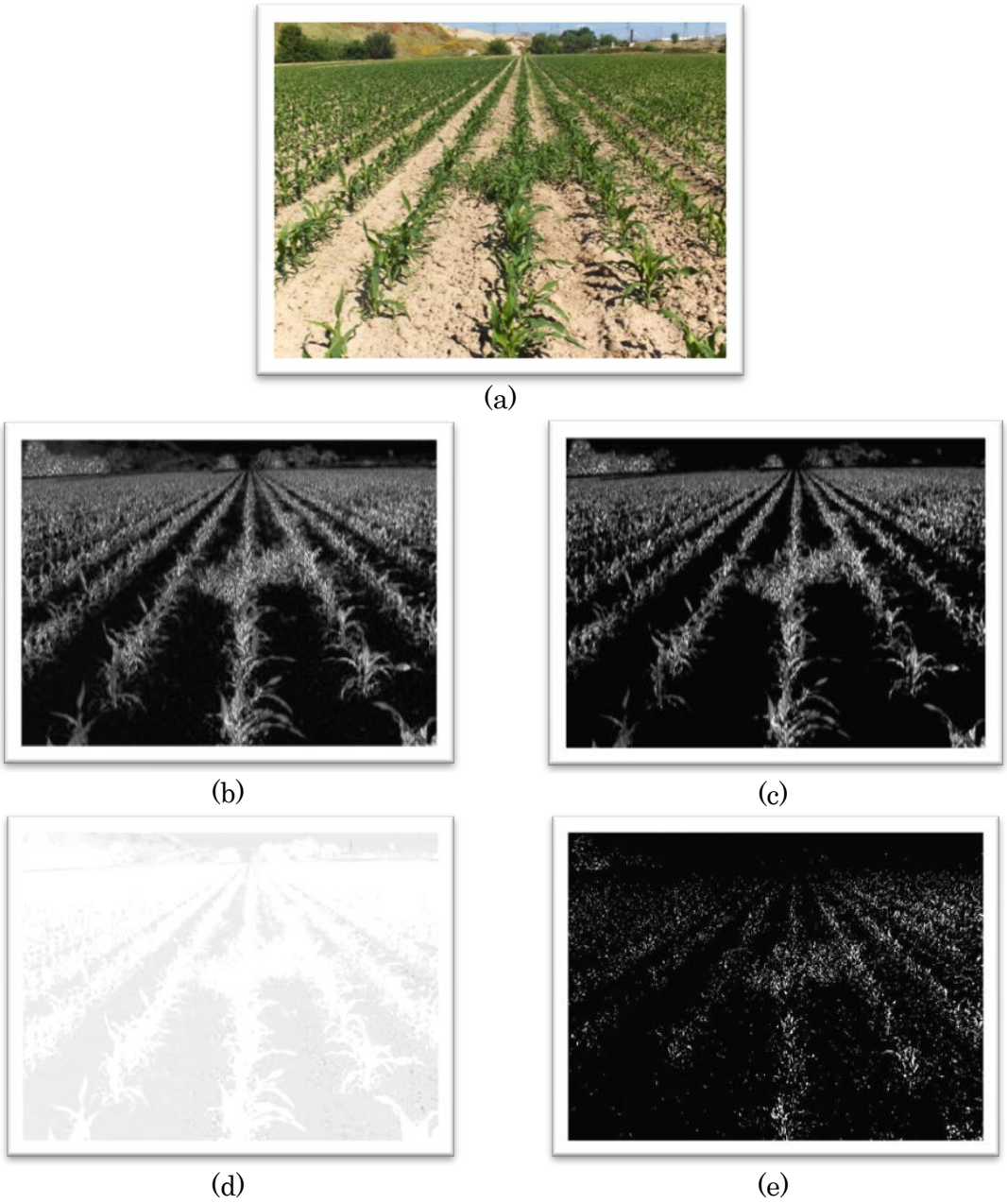



**Figura 2.6** (a) Imagen original; (b) Imagen ExG; (c) Imagen ExGR; (d) Imagen CIVE;(e) Imagen VEG

## 2.4 Binarización de imágenes

La binarización de una imagen digital consiste en transformar la imagen en escala de grises en una imagen en blanco y negro. Para realizar la operación de binarización, se deberá elegir un valor adecuado dentro de los niveles de grises (umbral). Todos los niveles de grises menores al umbral calculado se convertirán en negro y todos los que resulten ser mayores en

blanco.

$$g(x,y) = \begin{cases} 0 \; si \; f(x,y) < T \\ 1 \; si \; f(x,y) \geq T \end{cases}$$

En nuestro caso, el proceso de binarización transformará las partes verdes de la imagen a blanco y el resto a negro. El punto clave es seleccionar un umbral óptimo, es decir, un valor de gris desde el cual se decide cuáles de dichas tonalidades son o no verdes.

Para realizar la binarización nos basaremos en el histograma de la imagen, que constituye el punto de partida del proceso.

Existen diferentes métodos de binarización de imágenes. De entre ellos, el método de Otsu (1979) es uno de los más clásicos y populares. Otro método posible es la binarización mediante la media estadística del histograma como valor de umbral.

### 2.4.1 Binarización mediante el método de Otsu

El método de Otsu resulta ser un método de binarización de imágenes digitales ampliamente utilizado en la literatura. Se basa en conceptos estadísticos, en concreto la dispersión de los valores (en este caso, niveles de gris) a través de la varianza.

Si partimos de una imagen con L niveles de intensidad, en nuestro caso 256, y asumiendo que el umbral buscado es T, las probabilidades hasta T y desde T hasta L resulta ser,

$$w1(t) = \sum_{z=1}^{T} P(z) \; , \;\; w2(t) = \sum_{z=T+1}^{L} P(z)$$

donde P(z) es el total de apariciones del nivel de intensidad z.

A continuación se obtienen las medias y las varianzas asociadas,

$$\mu1(t) = \sum_{z=1}^{T} z * \frac{P(z)}{w1(t)}, \quad \mu2(t) = \sum_{z=T+1}^{L} z * \frac{P(z)}{w2(t)}$$

$$\sigma1(t)^2 = \sum_{z=1}^{T} (z - \mu1(t))^2 \frac{P(z)}{w1(t)}, \quad \sigma2(t)^2 = \sum_{z=T+1}^{L} (z - \mu2(t))^2 \frac{P(z)}{w2(t)}$$

Finalmente se obtiene la varianza ponderada:

$$\sigma_w^2(t) = w1(t)\sigma1(t)^2 + w2(t)\sigma2(t)^2$$

Se elige el umbral T correspondiente al nivel de intensidad que proporcione la mínima varianza ponderada.

### 2.4.2 Binarización mediante la media

Por media se entiende el valor medio de los niveles de intensidad en la imagen. Se obtiene directamente a partir del histograma de la imagen, mediante la siguiente ecuación:

$$m = \sum_{i=0}^{L-1} z_i * p(z_i)$$

donde L es el número total de niveles de intensidad, en nuestro caso 256.

Las imágenes mostradas en la figura 2.7, son el resultado de aplicar los mencionados métodos de binarización a la imagen de la figura 2.6(a).

(a) (b)

**Figura 2.7** (a) Binarización con la media; (b) Binarización método Otsu.

# CAPITULO 3: Fase de análisis

Este capítulo tiene por objeto la explicación y documentación de las fases típicas de todo proyecto de desarrollo de software, que son exactamente las aplicadas en nuestro caso:

- **Fase de análisis**, centrada en el qué, se identifican los requisitos del sistema.
- **Fase de diseño**, se detallan los aspectos más importantes de la arquitectura del sistema, en nuestro caso a través de diagramas UML.
- **Fase de implementación**, centrada en cómo se hacen las cosas, las tareas principales en esta fase son: traducir el diseño a un lenguaje de programación, generación de código, implementación de funciones.
- **Fase de pruebas**, se realizan distintos tipos de pruebas software.

Todas las acciones y tareas que se incluyen y se realizan sobre cada una de estas fases, es lo que se denomina como el ciclo de vida de un proyecto software.

Uno de estos modelos del ciclo de vida, es el del desarrollo en cascada, figura 3.1. Es una visión del proceso de desarrollo de software como una sucesión de fases que producen productos intermedios. Al final de cada una de las fases se revisan las tareas y los productos. Es decir, cada etapa deja el camino preparado para la siguiente, de forma que esta última no puede comenzar hasta que no terminado la anterior.

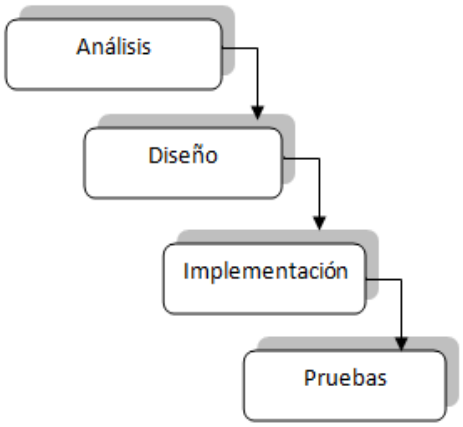



**Figura 3.1** Modelo en cascada del desarrollo software

La principal limitación es que no se permiten iteraciones entre las fases y por tanto resulta ser poco realista desde el punto de vista práctico. Por eso existe una variante, que es el modelo en cascada con retroalimentación entre fases, mostrado en la figura 3.2. Si después de la terminación de alguna etapa los resultados no son los esperados, podemos volver a una fase anterior. Este modelo es mucho más flexible y realista que el anterior.

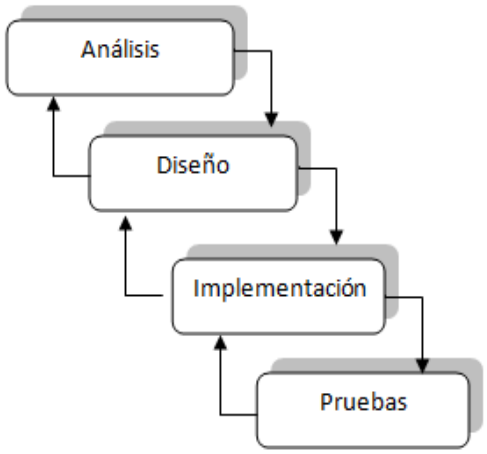



**Figura 3.2** Modelo en cascada del desarrollo software con retroalimentación

Este modelo de proceso es el que finalmente hemos elegido a la hora desarrollar nuestro proyecto software.

En las etapas de análisis y diseño se emplean diagramas UML (Arlow, 2006), utilizados para definir, documentar y construir un sistema.

¿Qué es UML?

UML (Unified Modeling Language) es una notación estándar para desarrollo de sistemas usando el enfoque orientado a objetos. Es un lenguaje de modelado gráfico utilizado para especificar. UML es sólo una notación, no dicta estándares para el desarrollo. Además soporta todo el ciclo de vida de desarrollo software. Los diagramas incluidos en UML, agrupados según su ámbito de aplicación, son:

**Análisis en el contexto organizacional:**

- Diagrama de casos de uso.

**Análisis y diseño desde la perspectiva estática:**

- Diagrama de clase

- Diagrama de objetos

**Análisis y diseño desde la perspectiva dinámica:**

- Diagrama de secuencia
- Diagrama de colaboración
- Diagrama de estados

**Implementación:**

- Diagrama de componentes
- Diagrama de despliegue

UML ofrece una amplia variedad de diagramas si bien, para las especificaciones de nuestro sistema utilizaremos solamente tres: diagrama de casos de uso, diagrama de clases y diagrama de secuencia.

Tras explicar cuáles son las etapas de desarrollo de software, haber comentado el modelo de proceso utilizado y hacer una pequeña introducción al lenguaje de modelado (UML) procedemos a comentar los objetivos y el trabajo realizado durante la fase de análisis dentro del ciclo de vida de desarrollo software.

Objetivos de la fase de análisis:

- Captura de requisitos del sistema a desarrollar en el proyecto.
- Estudio de los casos de uso y realización del diagrama de casos de uso.
- Análisis de riesgos.
- Elección del lenguaje de programación.
- Estudio y familiarización de conceptos usados en el ámbito de la detección de verdes para imágenes en entornos de agricultura, explicados y documentados en el capítulo dos, que resumimos aquí por comodidad:
  - Índices cromáticos → ExG, ExGR, CIVE, Vegetative.
  - Binarización → Método de Otsu, Método de la media.

## 3.1 Análisis de requisitos

Este apartado tiene como finalidad documentar y explicar los requisitos software de nuestro sistema o aplicación.

### 3.1.1 Requisitos funcionales

- La aplicación permitirá cargar una imagen RGB para su tratamiento
- A partir de una imagen RGB, se extraen cada una de las componentes que forman la imagen con dicho modelo:
    - Extraer componente roja (R) y mostrar su histograma.
    - Extraer componente verde (G) y mostrar su histograma.
    - Extraer componente azul (B) y mostrar su histograma.
- El usuario podrá aplicar a la imagen una serie de métodos o procedimientos (índices cromáticos) que permiten la detección de verdes. En la aplicación se podrán utilizar los siguientes:
    - Exceso de verde (ExG)
    - Diferencia del exceso de verde con el exceso de rojo (ExGR)
    - Color index of vegetation extraction (CIVE)
    - Vegetative (VEG)

Tras aplicar estos procedimientos se obtendrá una imagen en escala de grises. Además se mostrará su histograma.

- A esta imagen en escala de grises se le podrá aplicar un umbral para su posterior binarización. La aplicación desarrollada contempla tres formas de binarizar: mediante el método de Otsu, mediante la media y se dará la opción de poder aplicar un umbral seleccionado por el usuario.

### 3.1.2 Otros requisitos

- La aplicación guardará en la ruta donde se encuentra la imagen original que se va a cargar en la aplicación las imágenes generadas tras aplicar algoritmos, binarizaciones o las creadas al extraer alguna de sus componentes.
- El sistema permitirá el tratamiento de cualquier imagen con formato JPG, JPEG, PNG, BMP, TIFF.
- Se podrá recortar la parte superior de la imagen original para poder eliminar información poco relevante para detectar verdes.
- Multiplataforma: la aplicación se podrá ejecutar en cualquier máquina independientemente del sistema operativo y arquitectura.

## 3.2 Diagrama de casos de uso

Los diagramas de casos de uso documentan el comportamiento de un sistema desde el punto de vista del usuario. Por tanto, los casos de uso determinan los requisitos funcionales del sistema, esto quiere decir que representan las funciones que el sistema puede ejecutar. El esquema mostrado en la figura 3.3 representa el diagrama correspondiente para la aplicación desarrollada.

La sección 3.2.1 tiene por objetivo la especificación de cada uno de los casos de uso que aparecen en el diagrama de la figura 3.3. La especificación del caso de uso está formada por una breve descripción del mismo, por precondiciones (descripciones del estado del sistema antes de la ejecución del caso de uso), el flujo principal (acciones principales en la ejecución del escenario) y por las postcondiciones (descripciones del estado del sistema tras ejecutar el caso de uso).


**Figura 3.3** Diagrama de casos de uso

### 3.2.1 Especificación casos de uso

Para aclarar y esclarecer los casos de uso del diagrama mostrado en la figura 3.3 explicamos con detalle cada uno de ellos como se ha explicado con anterioridad.

| |
|---|
| **Caso de uso**: Extraer componentes de la imagen |
| **Identificador:** 1 |
| **Breve descripción**:<br>Generación de imágenes correspondientes a las componentes espectrales de una imagen en formato RGB. |
| **Actores principales:**<br>Usuario |
| **Precondiciones:**<br>Imagen con modelo RGB cargada. |
| **Flujo principal:**<br>1. El caso de uso puede comenzar tras haber abierto una imagen.<br>2. El sistema genera tres imágenes diferentes, correspondientes a las subimágenes que forman la imagen original (roja, verde y azul). |
| **Postcondiciones:**<br>Las imágenes formadas son en escala de grises |

| |
|---|
| **Caso de uso**: Extraer componente azul |
| **Identificador:**2 |
| **Identificador padre:**1 |
| **Breve descripción**:<br>Generación de la imagen azul correspondientes a las componentes espectrales de una imagen en formato RGB. |
| **Actores principales:**<br>Usuario |
| **Precondiciones:**<br>Imagen con modelo RGB cargada. |
| **Flujo principal:**<br>1. El caso de uso puede comenzar tras haber abierto una imagen.<br>2. El sistema genera una imagen en escala de grises, formada únicamente por los píxeles correspondiente a la componente espectral azul que forma parte de la imagen RGB. |
| **Postcondiciones:**<br>La imagen formada es en escala de grises |

| |
|---|
| **Caso de uso**: Extraer componente verde |
| **Identificador:**3 |
| **Identificador padre:**1 |
| **Breve descripción**:<br>Generación de la imagen verde correspondientes a las componentes espectrales de una imagen en formato RGB. |
| **Actores principales:**<br>Usuario |

| |
|---|
| **Precondiciones:**<br>Imagen con modelo RGB cargada. |
| **Flujo principal:**<br>1. El caso de uso puede comenzar tras haber abierto una imagen.<br>2. El sistema genera una imagen en escala de grises, formada únicamente por los píxeles correspondiente a la componente espectral verde que forma parte de la imagen RGB. |
| **Postcondiciones:**<br>La imagen formada es en escala de grises |

| |
|---|
| **Caso de uso**: Extraer componente roja |
| **Identificador:**4 |
| **Identificador padre:**1 |
| **Breve descripción**: Generación de la imagen roja correspondientes a las componentes espectrales de una imagen en formato RGB. |
| **Actores principales:**<br>Usuario |
| **Precondiciones:**<br>Imagen con modelo RGB cargada. |
| **Flujo principal:**<br>1. El caso de uso puede comenzar tras haber abierto una imagen.<br>2. El sistema genera una imagen en escala de grises, formada únicamente por los píxeles correspondiente a la componente espectral roja que forma parte de la imagen RGB. |
| **Postcondiciones:**<br>La imagen formada es en escala de grises |

| |
|---|
| **Caso de uso**: Aplicar algoritmo detección de verdes |
| **Identificador:** 5 |
| **Breve descripción**:<br>Generar una imagen tras haber aplicado un índice cromático para la detección de verdes en una imagen RGB. |
| **Actores principales:**<br>Usuario |
| **Precondiciones:**<br>Imagen con modelo RGB cargada. |
| **Flujo principal:**<br>1. El caso de uso puede comenzar tras haber abierto una imagen.<br>2. El sistema genera una nueva imagen en escala de grises tras haber aplicado un algoritmo de extracción de verdes.<br>3. Dicha imagen se guarda en una carpeta situada en la ruta donde se ejecuta la aplicación. |
| **Postcondiciones:**<br>La imagen formada es en escala de grises |

| |
|---|
| **Caso de uso**: Aplicar algoritmo CIVE |
| **Identificador:** 6 |
| **Identificador padre:**5 |
| **Breve descripción**:<br>Generar una imagen tras haber aplicado el algoritmo CIVE para la detección de verdes en una imagen RGB. |
| **Actores principales:**<br>Usuario |
| **Precondiciones:**<br>Imagen con modelo RGB cargada. |
| **Flujo principal:**<br>1. El caso de uso puede comenzar tras haber abierto una imagen.<br>2. El sistema genera una nueva imagen en escala de grises tras haber aplicado el algoritmo CIVE.<br>3. Dicha imagen se guarda en una carpeta situada en la ruta donde se ejecuta la aplicación |
| **Postcondiciones:**<br>La imagen formada es en escala de grises |

| |
|---|
| **Caso de uso**: Aplicar algoritmo ExG |
| **Identificador:** 7 |
| **Identificador padre:**5 |
| **Breve descripción**:<br>Generar una imagen tras haber aplicado el algoritmo ExG para la detección de verdes en una imagen RGB. |
| **Actores principales:**<br>Usuario |
| **Precondiciones:**<br>Imagen con modelo RGB cargada. |
| **Flujo principal:**<br>1. El caso de uso puede comenzar tras haber abierto una imagen.<br>2. El sistema genera una nueva imagen en escala de grises tras haber aplicado el algoritmo ExG.<br>3. Dicha imagen se guarda en una carpeta situada en la ruta donde se ejecuta la aplicación |
| **Postcondiciones:**<br>La imagen formada es en escala de grises |

| |
|---|
| **Caso de uso**: Aplicar algoritmo ExGR |
| **Identificador:** 8 |
| **Identificador padre:**5 |
| **Breve descripción**:<br>Generar una imagen tras haber aplicado el algoritmo ExGR para la detección de verdes en una imagen RGB. |
| **Actores principales:**<br>Usuario |
| **Precondiciones:**<br>Imagen con modelo RGB cargada. |
| **Flujo principal:**<br>1. El caso de uso puede comenzar tras haber abierto una imagen.<br>2. El sistema genera una nueva imagen en escala de grises tras haber aplicado el algoritmo ExGR.<br>3. Dicha imagen se guarda en una carpeta situada en la ruta donde se ejecuta la aplicación |
| **Postcondiciones:**<br>La imagen formada es en escala de grises |

| |
|---|
| **Caso de uso**: Aplicar algoritmo Vegetative |
| **Identificador:** 9 |
| **Identificador padre:**5 |
| **Breve descripción**:<br>Generar una imagen tras haber aplicado el algoritmo Vegetative para la detección de verdes en una imagen RGB. |
| **Actores principales:**<br>Usuario |
| **Precondiciones:**<br>Imagen con modelo RGB cargada. |
| **Flujo principal:**<br>1. El caso de uso puede comenzar tras haber abierto una imagen.<br>2. El sistema genera una nueva imagen en escala de grises tras haber aplicado el algoritmo Vegetative.<br>3. Dicha imagen se guarda en una carpeta situada en la ruta donde se ejecuta la aplicación |
| **Postcondiciones:**<br>La imagen formada es en escala de grises |

| |
|---|
| **Caso de uso**: Binarizar imagen |
| **Identificador:** 10 |
| **Breve descripción**:<br>Tratar una imagen para conseguir otra en blanco y negro. |
| **Actores principales:**<br>Usuario |
| **Precondiciones:**<br>Imagen en escala de grises tras haber aplicado algún índice cromático. |
| **Flujo principal:**<br>1. El caso de uso puede comenzar tras haber creado una imagen en escala de grises producto de haber aplicado algún algoritmo de detección de verdes.<br>2. El sistema genera una nueva imagen con dos colores (blanco y negro) tras haber usado algún método de binarización.<br>3. Dicha imagen se guarda en una carpeta situada en la ruta donde se ejecuta la aplicación. |
| **Postcondiciones:**<br>La imagen formada es en blanco y negro |

| |
|---|
| **Caso de uso**: Binarización Otsu |
| **Identificador:** 11 |
| **Identificador padre:**10 |
| **Breve descripción**:<br>Tratar una imagen para conseguir otra en blanco y negro mediante binarización por el método de Otsu. |
| **Actores principales:**<br>Usuario |
| **Precondiciones:**<br>Imagen en escala de grises tras haber aplicado algún índice cromático. |
| **Flujo principal:**<br>1. El caso de uso puede comenzar tras haber creado una imagen en escala de grises producto de haber aplicado algún algoritmo de detección de verdes.<br>2. El sistema genera una nueva imagen con dos colores (blanco y negro) tras haber usado el método de Otsu.<br>3. Dicha imagen se guarda en una carpeta situada en la ruta donde se ejecuta la aplicación. |
| **Postcondiciones:**<br>La imagen formada es en blanco y negro |

| |
|---|
| **Caso de uso**: Binarización Selección Umbral |
| **Identificador:** 12 |
| **Identificador padre:**10 |
| **Breve descripción**:<br>Tratar una imagen para conseguir otra en blanco y negro mediante binarización. |
| **Actores principales:**<br>Usuario |
| **Precondiciones:**<br>Imagen en escala de grises tras haber aplicado algún índice cromático. |
| **Flujo principal:**<br>1. El caso de uso puede comenzar tras haber creado una imagen en escala de grises producto de haber aplicado algún algoritmo de detección de verdes.<br>2. El sistema genera una nueva imagen con dos colores (blanco y negro) tras haber binarizado la imagen al seleccionar un umbral manualmente.<br>3. Dicha imagen se guarda en una carpeta situada en la ruta donde se ejecuta la aplicación. |
| **Postcondiciones:**<br>La imagen formada es en blanco y negro |

| |
|---|
| **Caso de uso**: Binarización Media |
| **Identificador:** 13 |
| **Identificador padre:**10 |
| **Breve descripción**:<br>Tratar una imagen para conseguir otra en blanco y negro mediante binarización por el método de la media. |
| **Actores principales:**<br>Usuario |
| **Precondiciones:**<br>Imagen en escala de grises tras haber aplicado algún índice cromático. |
| **Flujo principal:**<br>1. El caso de uso puede comenzar tras haber creado una imagen en escala de grises producto de haber aplicado algún algoritmo de detección de verdes.<br>2. El sistema genera una nueva imagen con dos colores (blanco y negro) tras haber usado el método de la media.<br>3. Dicha imagen se guarda en una carpeta situada en la ruta donde se ejecuta la aplicación. |
| **Postcondiciones:**<br>La imagen formada es en blanco y negro |

| |
|---|
| **Caso de uso**: Mostrar histograma |
| **Identificador:** 14 |
| **Breve descripción**:<br>Mostrar histograma de una imagen. |
| **Actores principales:**<br>Usuario |
| **Precondiciones:**<br>Haber cargado una imagen en formato RGB, o haber aplicado algún algoritmo de detección de verdes. |
| **Flujo principal:**<br>1. El caso de uso puede tras cargar una imagen comenzar o tras haber creado una imagen en escala de grises producto de haber aplicado algún algoritmo de detección de verdes.<br>2. El sistema genera y muestra el histograma correspondiente de la imagen seleccionada. |

## 3.3 Riesgos

La gestión de riesgos se ocupa de identificar riesgos y elaborar planes para minimizar el impacto sobre el proyecto.

Esta gestión incluye varias acciones a realizar:

- Identificar el riesgo.
- Evaluar su probabilidad de aparición.
- Estimar su impacto.
- Establecer un plan de contingencia por si ocurre el problema.

Nos ocuparemos de dos tipos de riesgos: 1) los tecnológicos, riesgos incluidos en la complejidad que tendrá el sistema a construir; y 2) los riesgos de personal y organizativos, que son los asociados con la experiencia de los integrantes del grupo que van a realizar el trabajo.

### 3.3.1 Riesgos tecnológicos

| Riesgo | Probabilidad | Impacto | Plan de contingencia |
|---|---|---|---|
| Pérdida de datos | Baja | Catastrófico | Tener varias copias de seguridad de código, documentación y todo lo relativo a la realización del proyecto. |

| | | | |
|---|---|---|---|
| Desconocimiento de tecnología a utilizar o de las librerías usadas | Media | Moderada | Adquirir el conocimiento de las librerías usadas mediante manuales y documentación. |
| Tratamiento de imágenes de grandes dimensiones | Alta | Grave | Utilización de algoritmo de reducción de imágenes. |
| Tutor o cliente cambia requisitos | Media | Moderada | Realización de software reutilizable, para que los cambios sean más fáciles de realizar. |
| Tiempo ejecución algoritmos | Media | Grave | Realización de algoritmos óptimos |

### 3.3.2 Riesgos organizativos y de personal

| Riesgo | Probabilidad | Impacto | Plan de contingencia |
|---|---|---|---|
| Enfermedad de alguno de los miembros del grupo | Baja | Moderado | Conocimiento de todos los miembros del estado y la realización de todas las partes del proyecto, para poder redistribuir trabajo. |
| Falta de comunicación entre los miembros del grupo | Baja | Grave | Establecer reuniones periódicas entre dichos miembros, además de una comunicación fluida vía email o vía telefónica. |
| Desconocimiento de algunas partes del proyecto por parte de alguno de los miembros | Media | Grave | Intentar solventarlo con reuniones o mediante buenos comentarios en el código. |

# CAPITULO 4: Fase de diseño

El presente capítulo trata sobre diferentes aspectos del diseño de la aplicación. Comentaremos en qué consiste la arquitectura Modelo–Vista–Controlador, y cómo se ha aplicado en nuestro sistema, apoyándonos para ello en el uso de diagramas UML, en concreto, diagrama de clases y el diagrama de paquetes.

Los diagramas de clases describen los tipos de objetos presentes en el sistema y las relaciones existentes entre ellos de una manera estática. Los diagramas de clase también muestran los atributos y operaciones de una clase y las restricciones a las que se ven sujetos. Además, para especificar el modelo de comportamiento y las interacciones entre elementos del sistema utilizaremos diagramas de secuencia.

## 4.1 Arquitectura Modelo-Vista-Controlador

La arquitectura Modelo–Vista–Controlador (MVC) es un patrón de diseño que separa la lógica de negocio, la interfaz de usuario y los datos con los que se trabaja. Divide la aplicación en tres partes:

- Un modelo que contiene la información con la que se trabaja, es decir, los datos.
- Vistas que gestionan cómo se muestra la información al usuario de una manera accesible.
- Uno o varios controladores que realizan las funciones correspondientes en el modelo como respuesta a eventos procedentes generalmente de la vista.

Una ventaja de usar MVC es que se incrementa la reutilización. Para ello el modelo no puede tener acceso a ninguna clase del Controlador ni de la Vista. Además el cambio de modelo no debe afectar a las vistas, y el controlador debe ver las clases del modelo, pero no de la vista. En la figura 4.1 puede observarse las principales características de este patrón, así como la manera de interactuar entre los distintos componentes que forman parte de la arquitectura MVC.

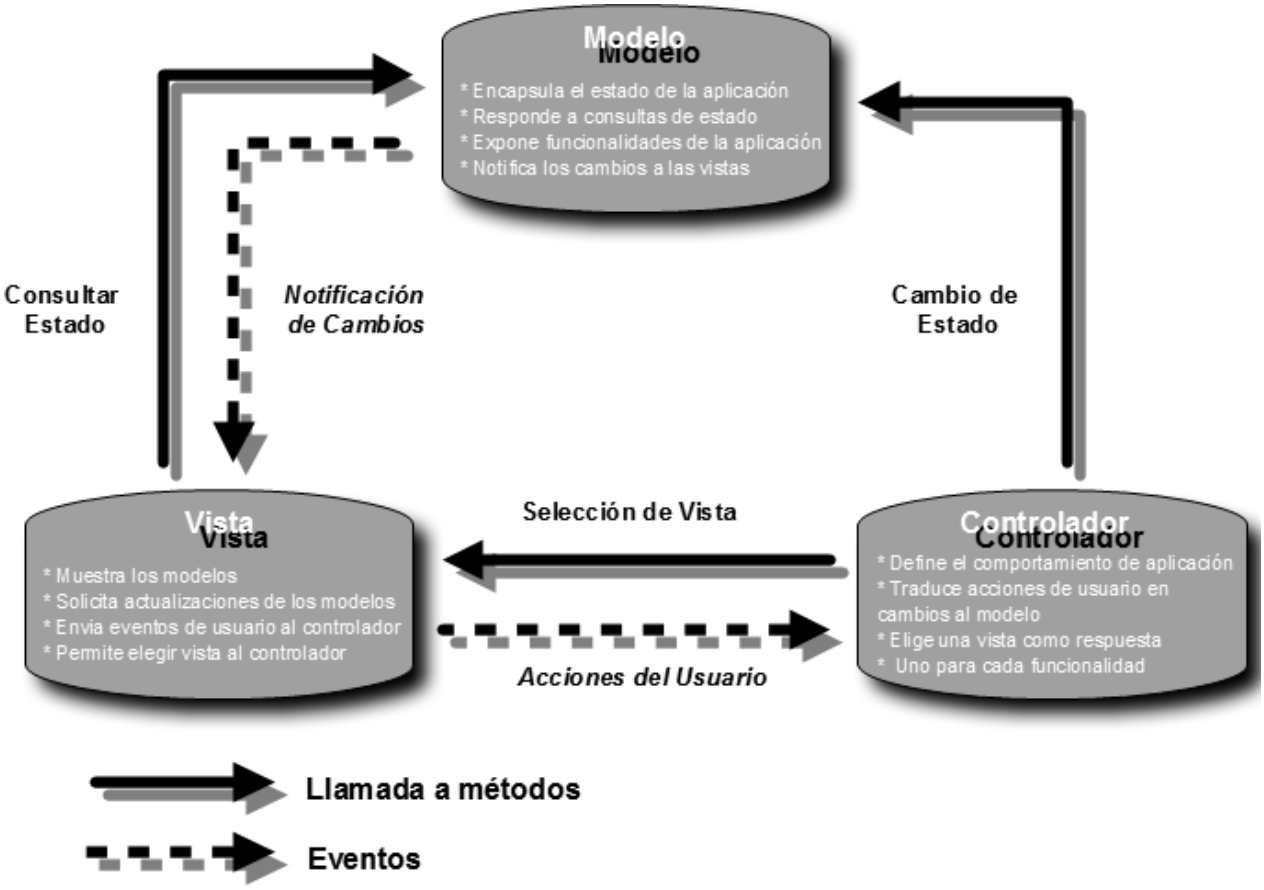



**Figura 4.1** Arquitectura modelo-vista-controlador

El uso de este patrón permite la evolución de forma separada de cada una de las partes, lo que proporciona una mayor flexibilidad a la hora de desarrollar los componentes del sistema.

Aunque existen diferentes implementaciones de MVC, normalmente el flujo que sigue es el siguiente:

- Usuario interactúa con la interfaz de usuario.
- El controlador gestiona el evento de entrada.
- El controlador informa al modelo sobre la acción elegida por el usuario, pudiendo acceder al modelo y modificarlo si fuera necesario de acuerdo a dicha acción.
- Se modifica la vista, tomando los datos del modelo.
- La interfaz de usuario espera nuevas interacciones del usuario.

Las ventajas sobre el uso de MVC son las que se expresan a continuación:

- Teniendo un mismo modelo, es posible conservar diferentes vistas.
- Es posible añadir nuevas vistas sin la necesidad de modificar el modelo.
- Menor acoplamiento entre vistas y modelo.
- Diseño más claro.

Obviamente, este modelo no es único, existen muchas variaciones e interpretaciones, si bien, lo importante es separar los tres elementos de que consta, algo que cumple el utilizado en este trabajo.

## 4.2 Diagrama de paquetes

Para la implementación de nuestro diseño nos hemos ajustado a esta arquitectura MVC, sin ser demasiado estrictos, si bien con el propósito de adaptarnos a sus características en la medida de lo posible.

La separación de cada una de las partes de nuestra aplicación se fundamenta en lo expuesto sobre estas líneas, para ello, en la figura 4.2 mostramos el diagrama de paquetes del sistema:

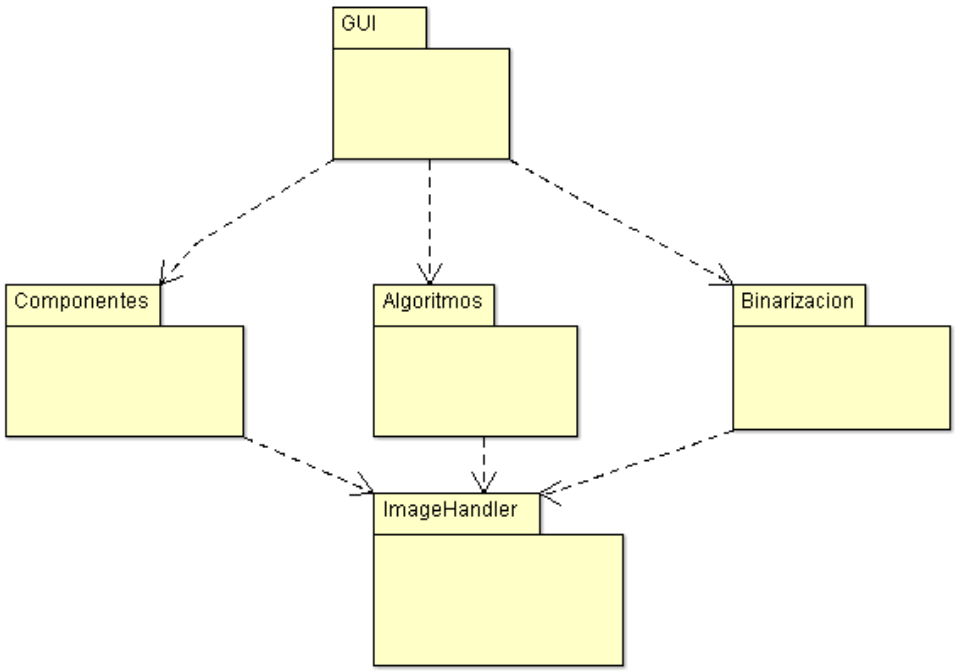



**Figura 4.2** Esquema de paquetes del sistema

El paquete GUI representa la vista en la aplicación. Dicho paquete contiene las diferentes clases para crear la interfaz gráfica de usuario.

Los paquetes componentes, algoritmos y binarización forman el controlador del patrón MVC que se ocuparán de modificar las imágenes, extrayendo las correspondientes componentes roja, verde y azul, aplicando sobre dichas componentes los índices de extracción de verdes o generando una imagen binaria.

El modelo está representado por las imágenes con las que trabajaremos, las cuales, tras haber sido cargadas y/o modificadas, se encontrarán accesibles a través de las clases del paquete imageHandler.

## 4.3 Diagramas de clases

¿Qué es y para qué sirve?

El diagrama de clases es el diagrama principal para el diseño estático, presentando y describiendo las clases y objetos del sistema con sus relaciones estructurales y de herencia. En la definición de clases se muestran los atributos y métodos incluidos en la clase.

En el esquema la figura 4.3 aparece representada la organización y las clases del paquete *algoritmos*, usando para ello el diagrama de clases del mencionado paquete. En la mencionada figura se presenta el modelado de las clases que nos permitirá generar los algoritmos de extracción de verdes. Contiene la clase abstracta *AlgoritmoDeteccionVerdes* que define el escalón más elevado en la jerarquía de estas clases, siendo la clase que abarca todas las funcionalidades necesarias para implementar los métodos de detección de verdes, conteniendo el método abstracto *aplicaAlgoritmo(),* que será necesario definir por cada una de las clases que hereden de la abstracta, donde se implementará la ecuación correspondiente a cada algoritmo. Cada una de estas clases heredadas representan los diferentes algoritmos de detección de verdes, documentados en el capítulo dos.

- Método CIVE: representado por la clase *AlgoritmoCIVE.*
- Método ExG: representado por la clase *AlgoritmoExG.*
- Método ExGR: representado por la clase *AlgoritmoExGR.*
- Método Vegetative: representado por la clase *AlgoritmoVegetative.*

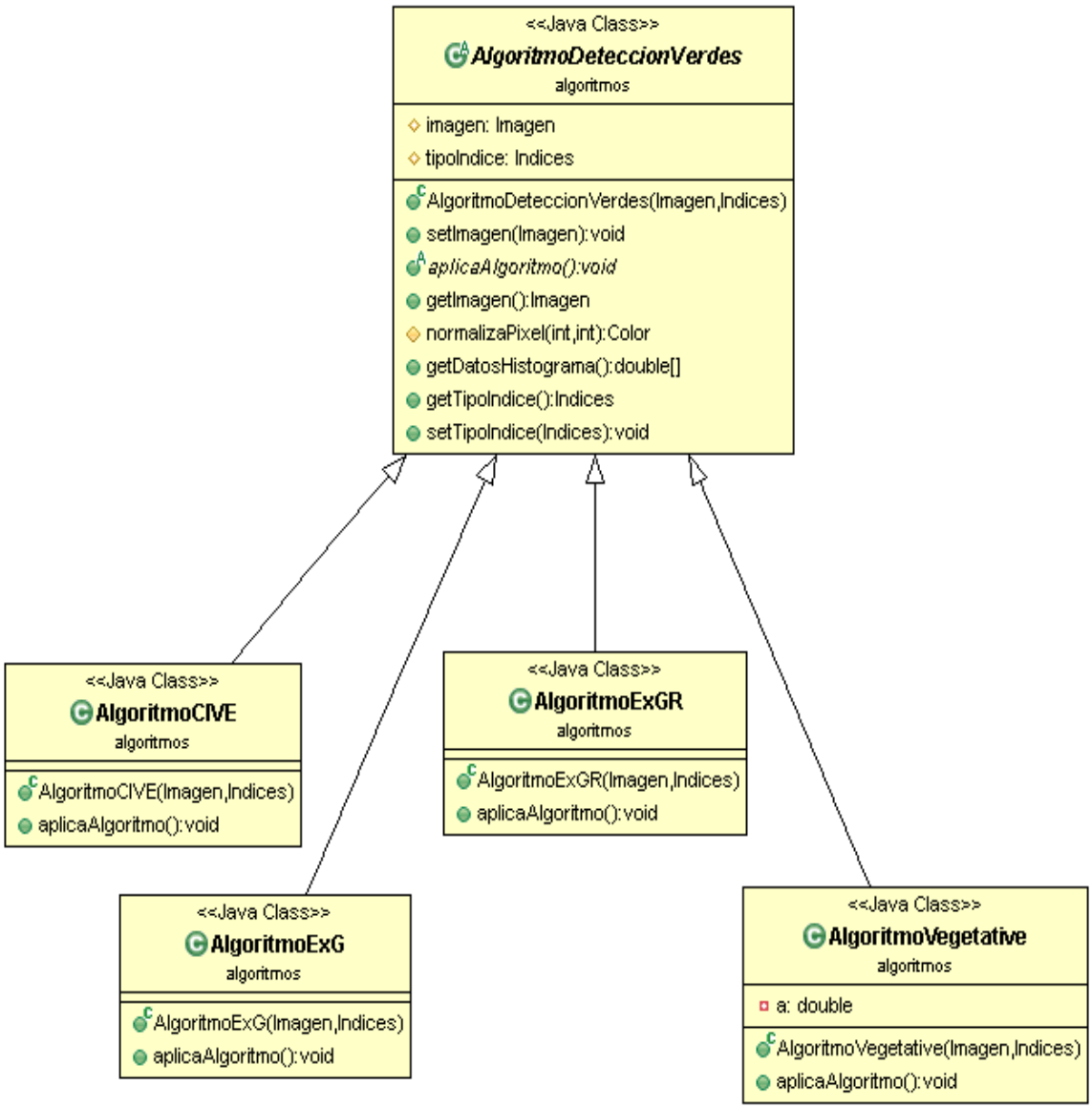



**Figura 4.3** Diagrama de clases del paquete algoritmos

Este diseño resulta ser muy flexible ya que permite añadir más tipos de algoritmos, simplemente habría que añadir una nueva clase que herede de *AlgoritmoDeteccionVerdes* e implemente sus métodos abstractos.

Por tanto, seguimos una estrategia similar para modelar el paquete *binarización,* como se muestra en el siguiente diagrama de clases de la figura 4.4:

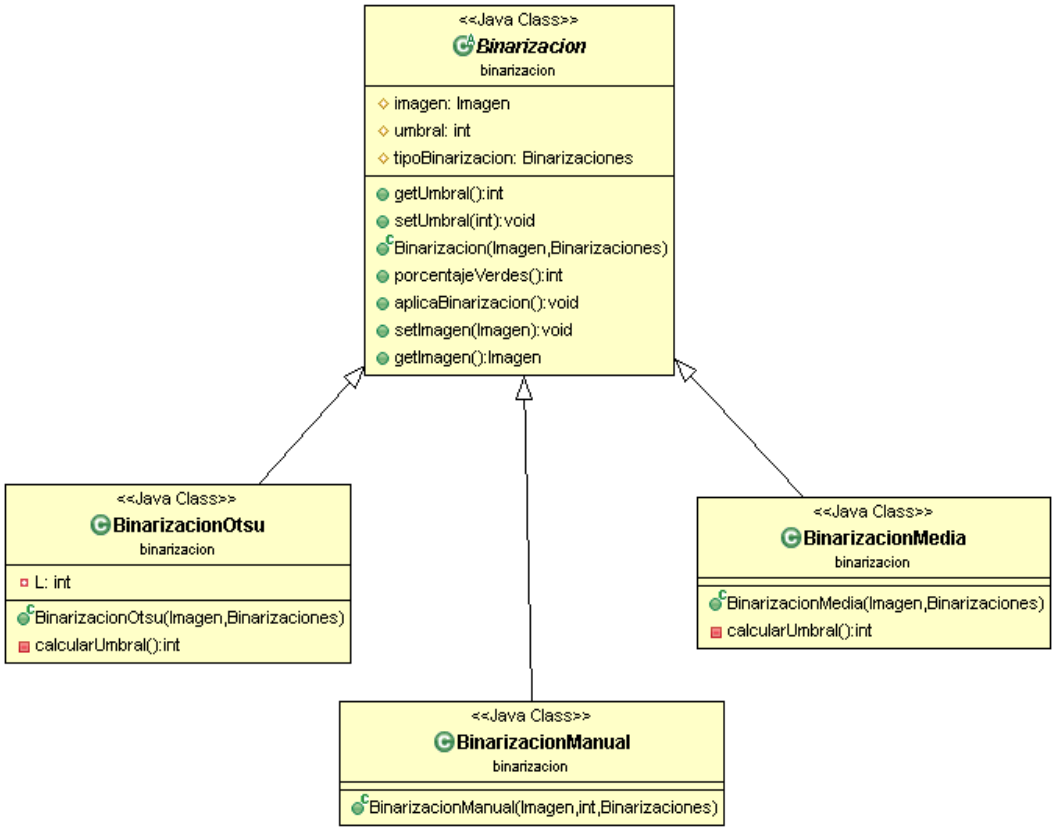



**Figura 4.4** Diagrama de clases del paquete binarización

La clase abstracta *Binarizacion* recoge las características y el comportamiento necesario para implementar los métodos que nos permitirán generar imágenes en blanco y negro. Esta clase implementa métodos y atributos, como por ejemplo el atributo umbral a partir del cual binarizamos, comunes a las subclases: *BinarizacionOtsu*, *BinarizacionMedia*, *BinarizacionManual*. Los objetos que se crearán serán ejemplares de estas últimas. Las clases heredadas de la clase *Binarizacion* representan los distintos patrones que usamos a la hora de crear imágenes binarias.

- Método Otsu: modelado por la clase *BinarizacionOtsu.*
- Método de la media: representado por la clase *BinarizacionMedia*.
- Método en el que se selecciona un umbral de manera manual: representado por la clase *BinarizacionManual*

## 4.4 Diagramas de Secuencia

Es un tipo de diagramas de interacción que nos ayuda a establecer con mayor detalle el escenario de un sistema, describiendo la manera en que colaboran los objetos a través de mensajes. Los objetos se muestran en el escenario mediante líneas verticales, lo que se conoce como línea de vida del objeto y los mensajes se representan como flechas horizontales entre las líneas de vida de dos objetos.

La figura 4.5 muestra cómo interaccionan los objetos cuando se abre una imagen.

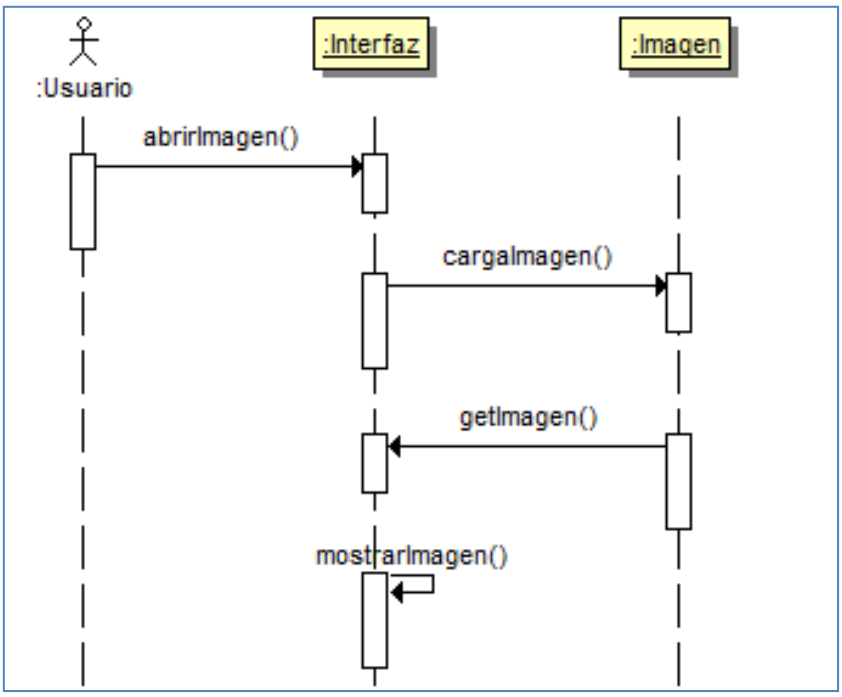



**Figura 4.5** Diagrama de secuencia escenario "Cargar Imagen"

El escenario comienza con la acción del usuario que elige abrir una imagen desde la interfaz. Tras haber seleccionado la imagen, se cargan los píxeles de ésta en un objeto imagen de nuestro modelo, y ésta le pasa la información necesaria a la interfaz para que pueda mostrar la imagen cargada.

Tras abrir la imagen, una opción interesante será eliminar un trozo de la parte superior de la imagen, descartando información que será de poca utilidad cuando posteriormente queramos aplicar los algoritmos de detección de verdes y de binarización. Este comportamiento queda plasmado en el diagrama de secuencia de la figura 4.6:

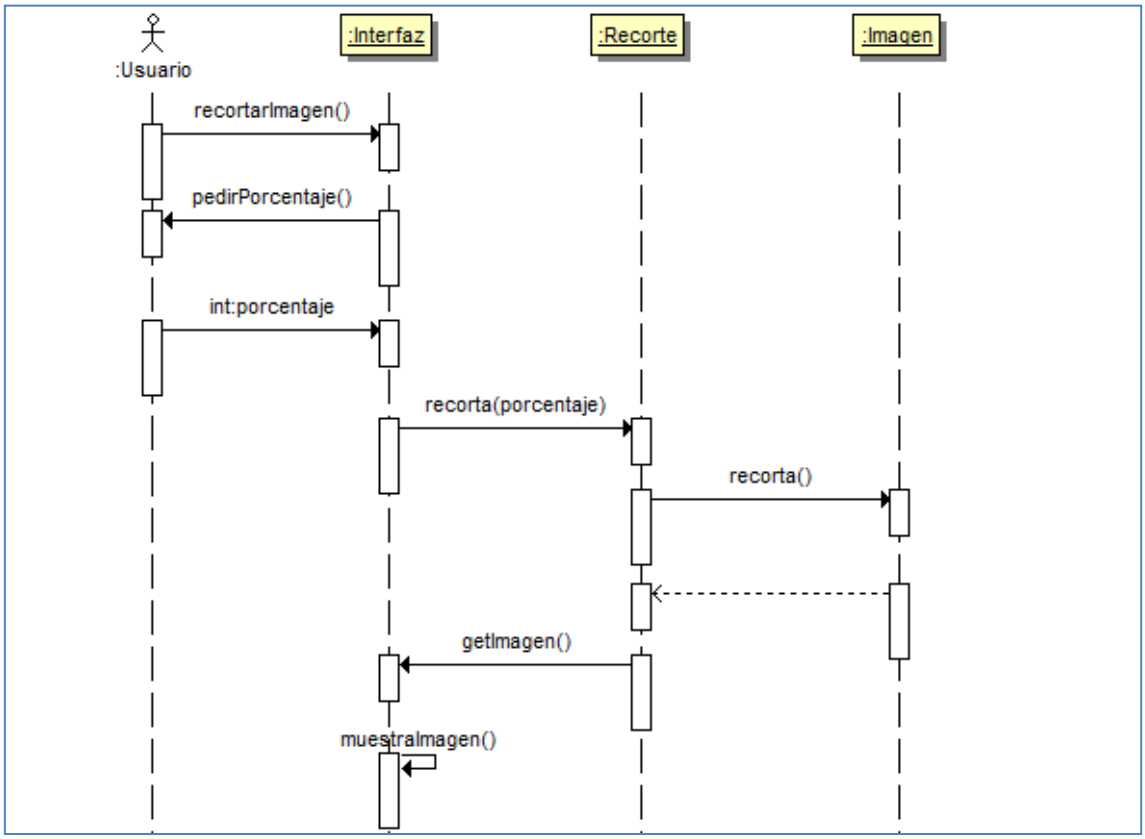



**Figura 4.6** Diagrama de secuencia escenario "Recortar Imagen"

Tras solicitar al usuario el porcentaje que queremos recortar de la parte superior de la imagen, la interfaz realiza una solicitud a un objeto de la clase *Recorte* que aplique el porcentaje pasado como parámetro. Éste se encargará de modificar la imagen, y pasarle la información necesaria a la interfaz para que pueda mostrar los cambios, es decir, la imagen original sin el recorte realizado sobre ella, que se corresponde con la porción que se ha eliminado.

Como ya es sabido, el principal objetivo que se persigue es detectar los píxeles verdes, y diferenciarlos del resto. Tras haber recortado la imagen, tendremos que ejecutar distintos esquemas que nos ayudarán precisamente a eso. Dicho comportamiento queda reflejado en la figura 4.7.

El usuario desea aplicar un algoritmo de detección de verdes, en este caso concreto exceso de verde (ExG), para ello, la interfaz utiliza un objeto que herede de la clase *AlgoritmoDeteccionVerdes*, que aplicará el esquema adecuado modificando la imagen. Para ilustrar este escenario hemos utilizado un objeto de la clase ExG, pero si en lugar de utilizar el índice ExG, se decide utilizar cualquiera de los otros tres (ExGR, CIVE, Vegetative), el esquema resulta ser el mismo, solamente habría que cambiar el objeto *algoritmoExg* por el adecuado. Tras haber aplicado el algoritmo, la interfaz necesita los datos apropiados para actualizarse, para

poder mostrar una imagen a la que se la ha aplicado el algoritmo correspondiente, en este caso ExG.

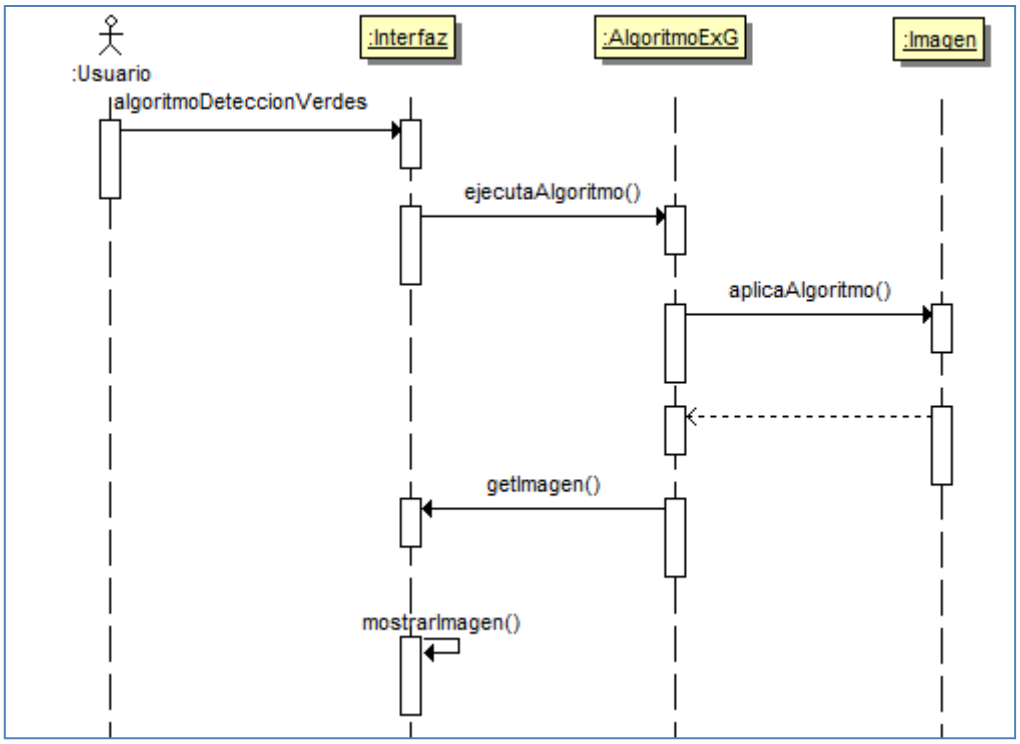



**Figura 4.7** Diagrama de secuencia escenario "aplicar índice cromático ExG"

Una vez que se haya aplicado un algoritmo de detección de verdes, el usuario podrá binarizar la imagen, quedando en blanco las zonas verdes y en negro el resto. Este escenario se representa en la figura 4.8, donde el objeto binarización será del tipo *BinarizacionOtsu*, *BinarizacionMedia, BinarizacionManual,* detallados en el diagrama de clases mostrado en la figura 4.4. Este objeto se encargará de modificar la imagen para que segmente en dos colores, como ya se ha explicado en el capítulo dos. La interfaz necesitará la imagen para poder mostrarla.

Otra opción, consiste en extraer las componentes espectrales roja, verde y azul a la imagen original. En la figura 4.9 mostramos cómo interaccionan los distintos tipos de objetos que son necesarios para poder realizar esta funcionalidad.

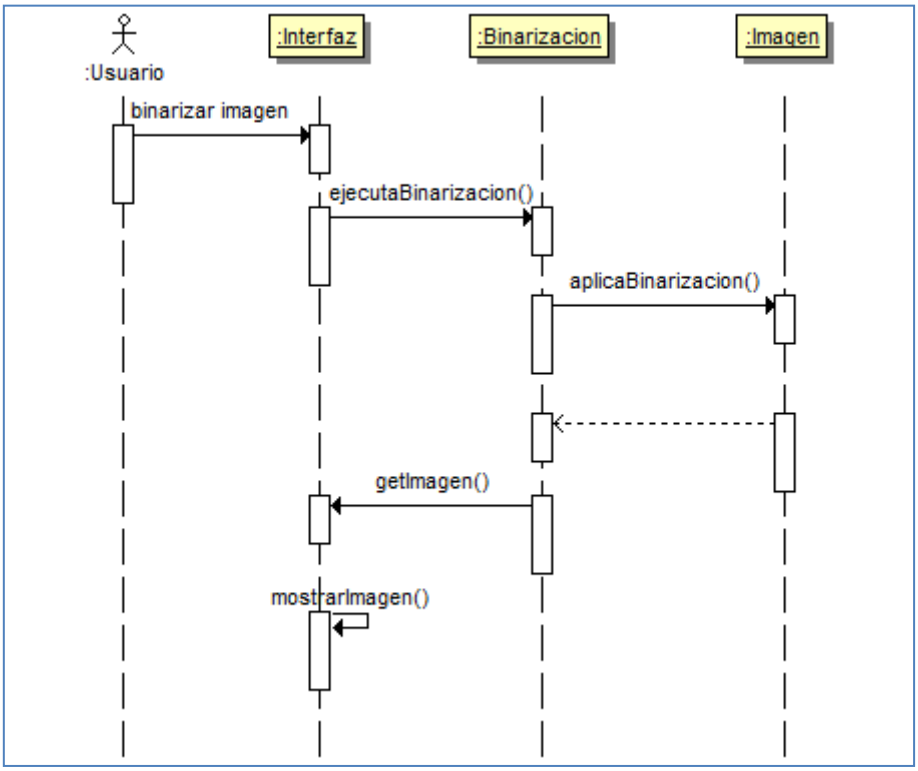



**Figura 4.8** Diagrama de secuencia escenario "aplicar binarización"

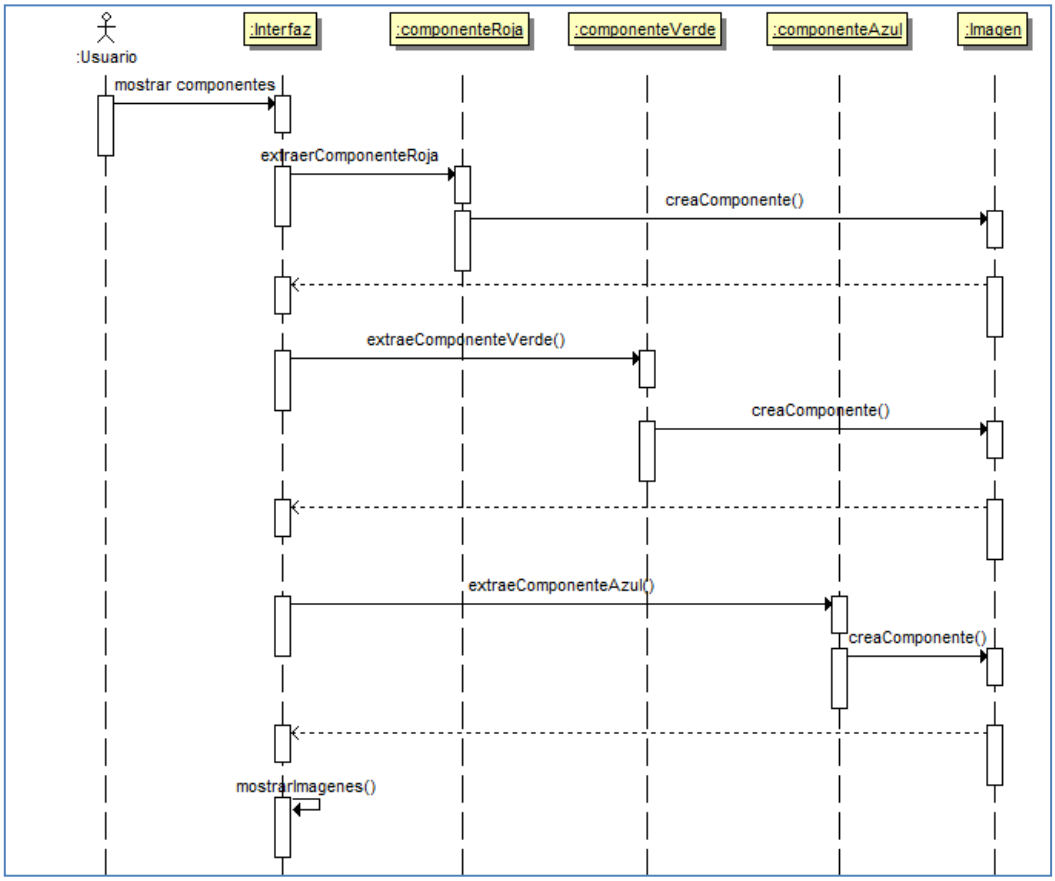



**Figura 4.9** Diagrama de secuencia escenario "extraer componentes"

# CAPITULO 5: Fase de implementación

Este capítulo se centra en los aspectos relacionados con la implementación de la aplicación desarrollada así como en la descripción de las diferentes decisiones que se han tomado para su desarrollo y de las incidencias que en dicha fase nos hemos encontrado. También se incluyen aspectos relacionados con las librerías usadas para el tratamiento de imágenes, así como del resto de herramientas utilizadas.

## 5.1 Lenguaje de programación: Java

El requisito de que la aplicación fuera multiplataforma fue decisivo para la elección de Java como lenguaje de programación para el desarrollo. A pesar de lo cual cabe mencionar el hecho de que este lenguaje ofrece muchas otras ventajas las cuales se enumeran a continuación:

### 5.1.1 Características de Java

- Open Source
  - El hecho de que Java sea Open Source hace que sea flexible y que evolucione según las necesidades del mercado y de los usuarios.
- Orientado a objetos
  - Esta característica proporciona, entre otras, técnicas como la modularidad, herencia, abstracción, polimorfismo y encapsulamiento, las cuales se hacen muy útiles para los objetivos de esta aplicación sobre todo de cara a una futura evolución y ampliación.
- Independiente de la plataforma
  - Uno de los requisitos era que la aplicación se pudiera ejecutar independientemente del sistema operativo o la arquitectura de la máquina, por lo que esta circunstancia llega a ser importante.
- Recolector de basura
  - Esta característica, ligada a la orientación a objetos, proporciona la ventaja frente a otros lenguajes de que es el propio Java el que se encarga de destruir automáticamente los objetos que ya no son referenciados, liberando así la memoria asociada a ellos. En nuestra aplicación, sobre todo al tratar imágenes grandes o al generar muchas en una misma sesión, esto se hace esencial para el rendimiento óptimo de la aplicación.

- Facilidad de aprendizaje
  - Java es un lenguaje relativamente sencillo de aprender comparado con otros muchos. Además nos permite ampliar rápidamente conocimientos en el caso de ser necesario debido a que la gran comunidad de programadores a nivel mundial que lo utiliza hace que ante nuevos retos de programación sea sencillo encontrar ayuda y aprender con ello.
- Librerías estándar
  - Una de las características que le otorga más potencia es su uso de librerías comunes que facilitan infinidad de operaciones a la hora de programar.

- Entornos de desarrollo
  - La facilidad a la hora de programar se hace mayor si viene acompañada de entornos de desarrollo accesibles y fáciles de usar. Java, al ser uno de los lenguajes más utilizados en todo el mundo, posee muchas alternativas a la hora de elegir con qué herramienta queremos trabajar. En nuestro caso ha sido Eclipse, por su facilidad y su amplia difusión.

## 5.2 Librerías

Para trabajar con imágenes las librerías estándar de Java no ofrecían determinadas funcionalidades que para nuestro desarrollo se antojaban necesarias.

Asimismo, el hecho de que la información mostrada fuera a ser determinante para una futura toma de decisión aplicada a la Agricultura de Precisión hacía que la información que se fuera a mostrar debiera ser precisa y clara, de tal forma que se mostrara de una manera gráfica al usuario. Para ello creímos necesario complementar las funciones de Java.

### 5.2.1 JMathPlot

JMathPlot (2012) es una biblioteca de gráficos en 2D y 3D de libre distribución que facilita enormemente las representaciones gráficas.

En nuestra aplicación era necesario mostrar de una manera clara la suficiente información para que cualquier decisión que se tomara al respecto fuera acertada, por lo que mostrar las imágenes junto con un histograma que se pudiera manipular así como obtener sus valores de una manera más

intuitiva desde el punto de vista gráfico nos pareció imprescindible.

Para ello elegimos utilizar esta librería, la cual aporta dichas funcionalidades haciendo que la información que obtiene el usuario sea precisa y clara.

### 5.2.2 JAI

Java Advanced Imaging API (JAI, 2012) es una extensión que provee a Java de una interfaz de programación que permite al desarrollador crear sus propias imágenes y manipularlas sin las restricciones ni el coste de licencias restrictivas asociadas al uso comercial de software. Además está específicamente diseñada por Sun para Java, lo que facilita enormemente el hecho de que la aplicación deba ser multiplataforma, cosa que no ocurría con otras librerías de tratamiento de imágenes específicas para determinados sistemas operativos.

Si bien, esta librería no fue necesaria hasta que ampliamos los requisitos para poder trabajar con imágenes tiff, las cuales no son soportadas por las librerías básicas de Java.

El tener que manipular imágenes de este tipo hizo que el uso de esta librería fuera imprescindible ya que las imágenes cargadas en este formato se convierten al estándar "jpeg" antes de mostrarse en la interfaz, pudiendo así trabajar con ellas de una manera rápida y sencilla.

## 5.3 Herramientas de trabajo

El uso de un lenguaje como Java hace que todo lo que rodea al desarrollo de una aplicación basada en él no sea nada complicado. El uso extendido a nivel mundial de este lenguaje hace que exista un gran abanico de herramientas que le proporcionan al desarrollador una gran cantidad de posibilidades, haciendo que el trabajo sea flexible y sencillo.

En nuestro caso, se decidió elegir como entorno de desarrollo Eclipse como ya se ha mencionado.

Junto a este entorno, se ha utilizado un sistema de control de versiones de software, comúnmente conocido como Software Version Number (SVN). La utilización de un repositorio es necesaria o al menos conveniente para cualquier desarrollo que requiera trabajo en equipo. El uso de versiones, así como la facilidad de trabajo de una manera independiente de los miembros del grupo ha hecho que la sincronización y el avance hayan

sido dinámicos e ininterrumpidos.

El trabajo de la parte documental se ha realizado bajo el control de Google Docs (2012). La diferente información, sobre todo en la fase de diseño e investigación, así como la redacción de las diferentes partes de la memoria se ha realizado "en la nube" (la información se almacena en servidores de Internet), ya que la necesidad de trabajar de una manera independiente hacían que la accesibilidad de todos los documentos en cualquier momento fuera necesaria.

Además del uso del repositorio, se decidió que el intercambio de otros elementos que no necesitaban entrar en el control de versiones se hiciera mediante Dropbox (2012). Las diferentes imágenes de prueba utilizadas así como otros archivos se han intercambiado mediante esta herramienta.

## 5.4 Problemas y resolución

Al tratar un campo totalmente nuevo para los miembros del grupo como es la aplicación de técnicas de tratamiento de imágenes en Agricultura de Precisión, el desarrollo de una aplicación para tal fin ha supuesto un importante trabajo de investigación así como diversas líneas y posibilidades dentro del mismo, lo cual en ocasiones ha hecho que no se finalizara de forma satisfactoria, haciendo que o bien se desecharan o se buscaran soluciones alternativas al respecto.

Uno de los principales problemas ha sido el uso de imágenes en formato tiff. Al no ser soportadas por las librerías básicas de Java, se tuvo que buscar una alternativa, como se ha explicado anteriormente en la sección 5.2.

Otro de los grandes problemas fue que el comportamiento de Swing (librería utilizada para desarrollar la interfaz gráfica, 2012) dependiendo de los diferentes sistemas operativos no fuera exactamente el mismo, lo cual llevó a trabajar la interfaz haciendo que esta fuera más ligera.

El último gran quebradero de cabeza en la fase de desarrollo fue el hecho de que la aplicación, tras trabajar con imágenes de gran tamaño y generar un número considerable de éstas, como resultados intermedios de los distintos procesos, se bloqueaba debido al uso excesivo de memoria. Se decidió redimensionar las imágenes grandes a un tamaño adecuado para ser tratadas y modificadas sin que la aplicación se colapsara por el exceso de uso de memoria.

En caso de querer trabajar con imágenes muy grandes se debería ejecutar la aplicación por consola pasándole al archivo .jar que se encuentra

en el CD-ROM incluido en la entrega del proyecto denominado VERDES.jar, los parámetros correspondientes:

- *Xms*: Indica el tamaño mínimo del *heap* que ha de reservar la máquina virtual.

- *Xmx*: Indica el tamaño máximo del *heap*.

Un ejemplo es el que se muestra a continuación:

```
$ java –jar VERDES.jar -Xmx1024m  -Xms512m
```

# CAPITULO 6: Fase de pruebas

El objetivo de la realización de pruebas respecto del desarrollo de la aplicación es cerciorarnos y comprobar el correcto funcionamiento del sistema así como garantizar que cumple con todos los requisitos propuestos en la fase de análisis.

Se han realizado una serie de pruebas que se enumeran a continuación:

1) Carga y recorte de imágenes.

2) Aplicación de índices cromáticos.

3) Binarización.

4) Extracción de las componentes espectrales.

## 6.1 Prueba 1: Carga y recorte de imágenes

Los objetivos de esta primera prueba son exactamente:

- Verificar el correcto funcionamiento de la aplicación al abrir una determinada imagen, independientemente del tamaño de la imagen o su formato. Se probará con imágenes de gran tamaño, y se mostrarán imágenes de tipo “jpg”, “tiff” y “bmp” abiertas por la aplicación.
- Comprobar que la imagen se recorta de manera correcta tras aplicar esta funcionalidad.

La figura 6.1 muestra cómo la aplicación carga una imagen en formato JPEG, con una dimensión de 4000x3000 píxeles (ancho x alto). Al ser de un gran tamaño necesitamos redimensionar.


**Figura 6.1** Imagen original en formato JPEG de gran tamaño

Tras cargar la imagen, podemos recortar la parte superior, en este caso se ha aplicado un recorte del 20%, tal y como se muestra en la figura 6.2.

**Figura 6.2** Imagen original recortada 20%.

Se puede observar que la parte superior de la imagen que se muestra en la figura 6.1 ya no aparece, había zona de árboles, que no nos resultaban de utilidad, pudiendo dar lugar a resultados erróneos al calcular el porcentaje de píxeles verdes en relación a las partes agrícolas de interés.

Además comprobamos que se pueden abrir imágenes en formato tiff y bmp. Los resultados mostrados en las figuras 6.3 y 6.4 verifican que la aplicación puede cargar imágenes con estos formatos respectivamente.


**Figura 6.3** Imagen original en formato TIFF

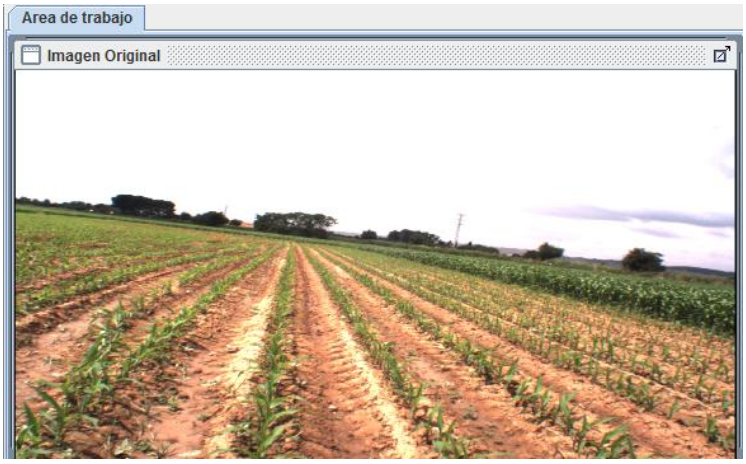


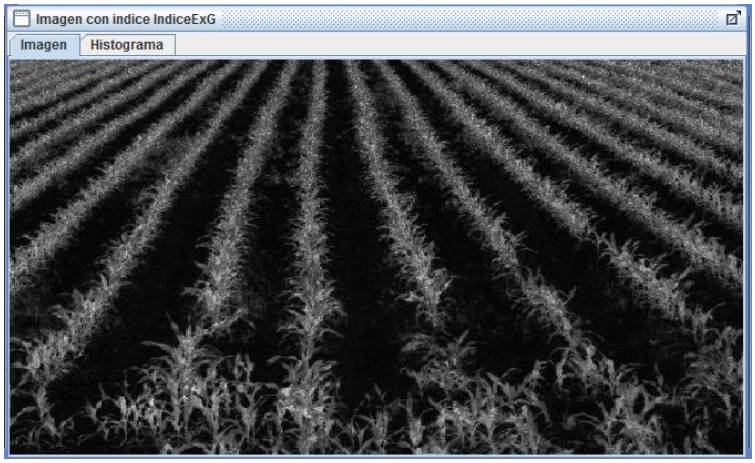

**Figura 6.4** Imagen original en formato BMP

## 6.2 Prueba 2: Algoritmos de extracción de verdes

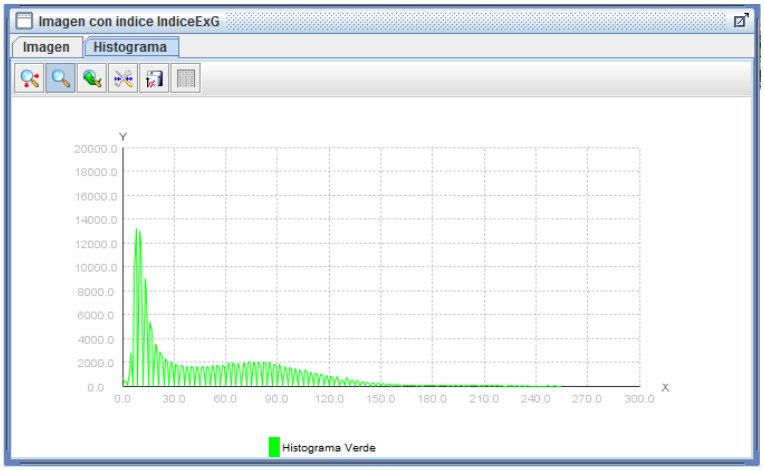
Lo que se pretende con esta segunda prueba es:

- Verificar que se generan y se aplican correctamente los índices cromáticos, y se obtienen imágenes en escala de grises.
- Comprobar que se genera el histograma de la imagen de manera satisfactoria, y sus datos son correctos.

La imagen de la figura 6.5 (a) muestra el resultado obtenido al aplicar uno de los algoritmos de detección de verdes a la imagen de la figura 6.2, en concreto el índice de exceso de verde (ExG):




(a)




(b)

**Figura 6.5** (a) Imagen ExG; (b) Histograma

La figura 6.5 (b) identifica el histograma de la imagen representada en (a), el eje Y muestra el número de píxeles que contienen los colores representados por el eje X. Se puede observar que aparece una gran cantidad de los valores comprendidos en el rango entre 0 y 120, oscuros, desde ese valor en adelante el número de píxeles con valores más altos, claros, disminuye.

Para validar el correcto funcionamiento del resto de índices cromáticos, en las figuras 6.6 y 6.7 se muestran respectivamente los resultados obtenidos tras aplicar los métodos de extracción de verdes ExGR y CIVE.

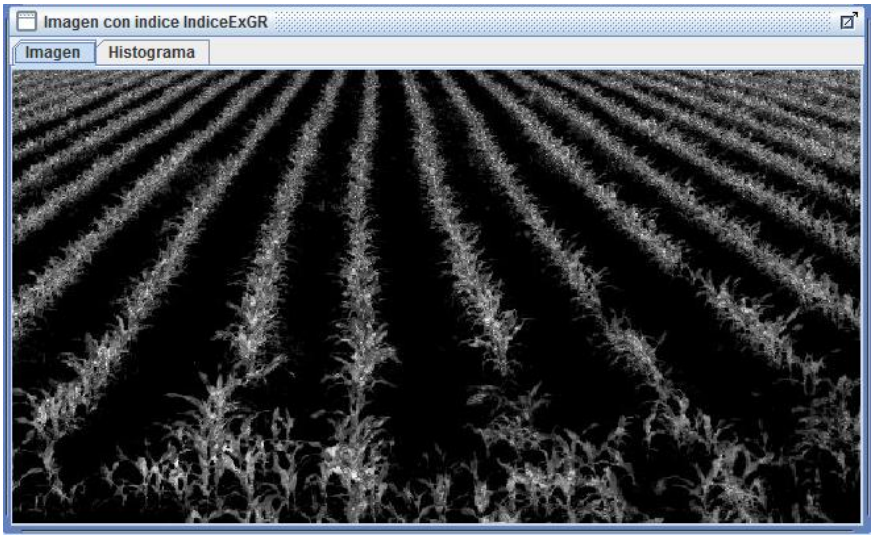



**Figura 6.6** Imagen ExGR


**Figura 6.7** Imagen CIVE

Se puede apreciar cómo en estas imágenes las zonas vegetales poseen un color muy cercano al blanco puro, mientras el resto de elementos presentan un color más oscuro, quedando perfectamente diferenciadas ambas zonas.

## 6.3 Prueba 3: Binarización de imágenes

Tras la identificación de las partes verdes, queda un último paso relativo a la segmentación de la imagen en dos partes, los píxeles blancos que representarán la parte verde de la imagen, y en negro el resto de ella. Con lo que vamos a comprobar que las binarizaciones son correctas. Partiendo de la imagen 6.5 tras haber aplicado el método ExG, generamos tres imágenes que se corresponden con la binarización por el método de la media, mostrado en la figura 6.8, la binarización por el método de Otsu, cuyo resultado se muestra en la figura 6.9, y seleccionando un valor umbral comprendido entre los valores obtenido por los anteriores métodos, que se presenta en la imagen de la figura 6.10.

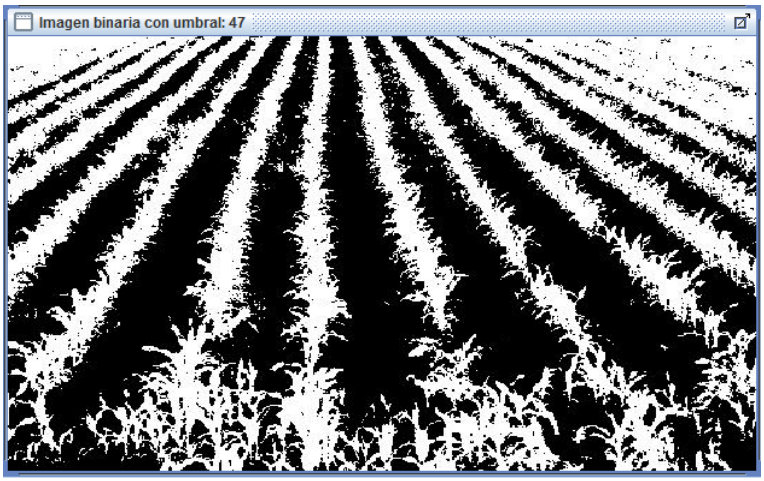



**Figura 6.8** Binarización método media con umbral 47.

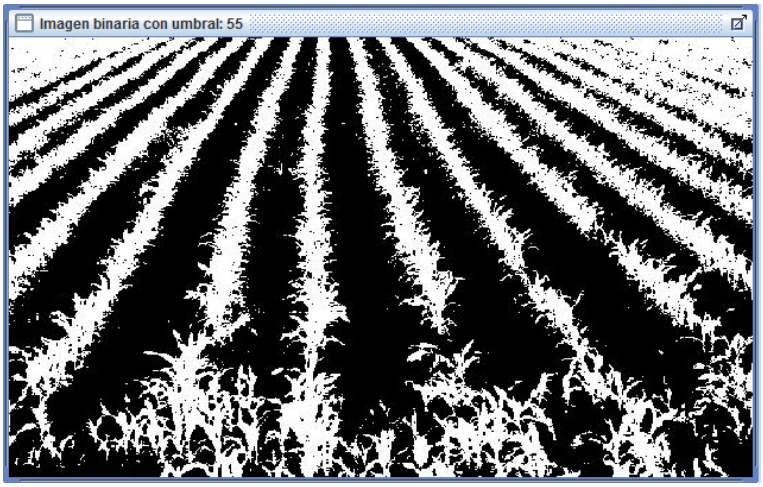



**Figura 6.9** Binarización método Otsu con umbral 55.

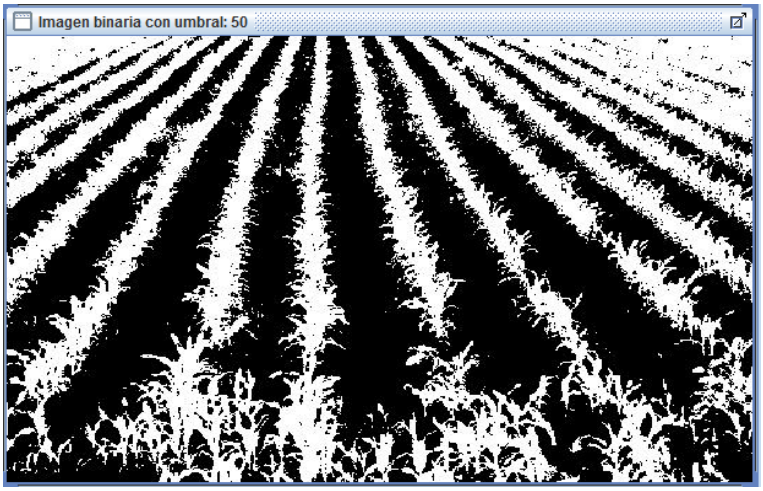
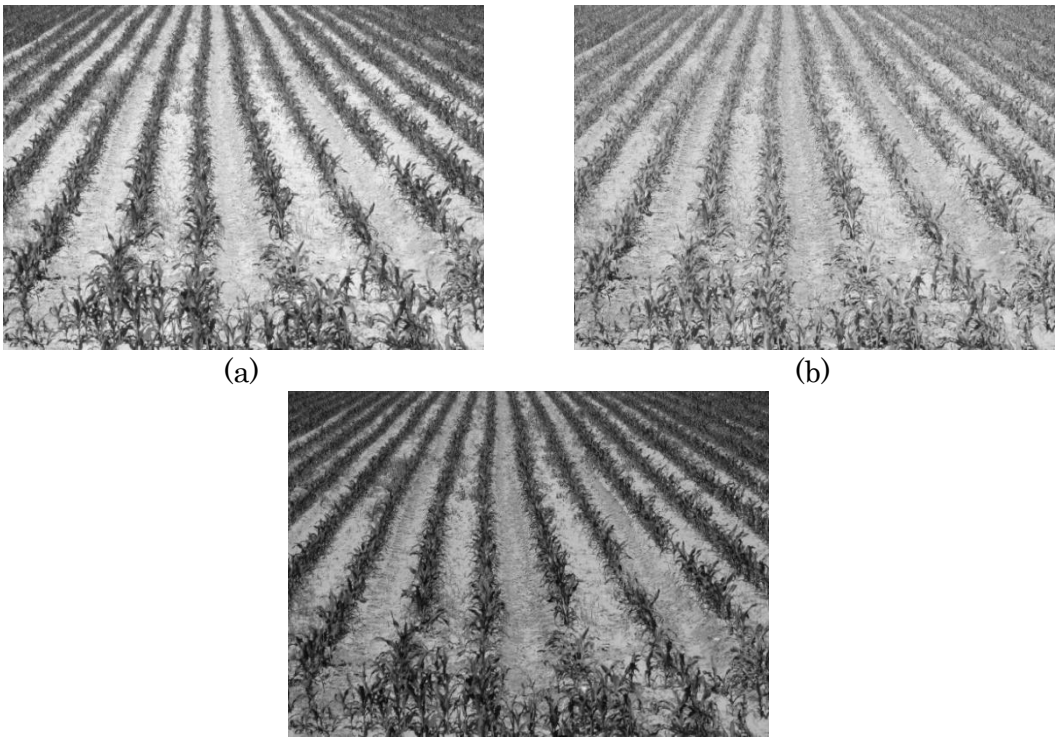



**Figura 6.10** Binarización método manual con umbral 50.

En las tres imágenes obtenemos una imagen en blanco y negro, recordando que los píxeles blancos se corresponden con los verdes de la imagen original.

## 6.4 Prueba 4: Extracción de componentes espectrales

Otro requisito que se verifica de una manera correcta es el de extracción de las tres componentes espectrales de la imagen: rojo, verde, azul. Mostramos las subimágenes obtenidas a partir de la imagen de la figura 6.2.



(a) (b)

(c)

**Figura 6.11** (a) Imagen Roja; (b) Imagen Verde; (c) Imagen Azul

Como apunte final, queda comprobado que las imágenes se generan y se guardan perfectamente en una carpeta, uno de los requisitos especificados en el capítulo dos.

# CAPÍTULO 7: Conclusiones

El proyecto realizado ha seguido el ciclo de vida típico de cualquier aplicación que se puede desarrollar a nivel educativo o empresarial, intentado poner en práctica técnicas de ingeniería del software y afianzando distintos conceptos aprendidos durante la carrera tales como el uso de patrones (MVC) o documentación mediante diagramas UML.

Se han aplicado nuevos conceptos sobre imágenes digitales así como su tratamiento y manipulación. Se han utilizado diferentes librerías software, para ello así como mejorar la destreza en el lenguaje elegido para el desarrollo, en este caso Java.

La realización de una interfaz gráfica nos ha permitido aprender a diseñar con eficacia en Swing, biblioteca de Java ampliamente utilizada en aplicaciones de escritorio multiplataforma.

La implementación de los distintos algoritmos de extracción de verdes y de las binarizaciones nos ha permitido conocer algunos conceptos y técnicas utilizadas en el tratamiento de imágenes aplicado a la Agricultura de Precisión, tomando conciencia con ello de los beneficios que puede tener introducir nuevas tecnologías en agricultura.

Los objetivos planteados al inicio del trabajo se han cumplido ampliamente. Se ha diseñado una interfaz gráfica con que permite imágenes y aplicar los distintos algoritmos y binarizaciones, mostrando la información necesaria para una primera toma de decisión en relación a las aplicaciones propias de la Agricultura.

Como conclusión final cabe destacar el compromiso de los miembros del grupo a la hora de conseguir los diferentes objetivos y plazos previstos no sólo por los responsables de dirigir el trabajo, sino también por los propios componentes, haciendo que éste sea un paso más para aprender a trabajar en equipo de una manera organizada y responsable de cara a la vida laboral.

# CAPÍTULO 8: Trabajo Futuro

Como se ha destacado al inicio de la memoria, hoy en día el trabajo de precisión en la agricultura se hace necesario para la optimización de recursos y la reducción de costes en un sector en el que queda mucho por hacer. La incorporación de las nuevas tecnologías orientadas a mejorar la producción agrícola se ve necesaria y casi imprescindible para un desarrollo acorde con los tiempos que vivimos.

## 8.1 Futuro de la aplicación

Tras conseguir obtener información de una manera gráfica de las partes verdes de una imagen, el siguiente paso lógico sería identificar en dichas partes las malas hierbas para así distinguirlas de las partes del cultivo propiamente dicho.

Una posible ampliación del proyecto podría ser el añadir e implementar otros algoritmos de detección de verdes u otras formas de binarizar que complementaran las ya desarrolladas en esta aplicación, lo cual no sería complicado por la manera en la que ha sido diseñada, ya que se ha orientado para su futura ampliación y modificación gracias a la reutilización de sus componentes, todo ello de una manera limpia y transparente haciendo el código accesible e interpretable con facilidad a cualquier ingeniero de desarrollo que pudiera continuar con la mejora de la aplicación. También la modularización y la arquitectura utilizada, así como el lenguaje de programación elegido hacen que dicho trabajo sea relativamente sencillo.

Otro posible trabajo futuro podría ser el mejorar la carga de las imágenes, de tal manera que se estudiaran en tiempo de ejecución y, en función de su tipo, características, más o menos luz, más o menos cantidad de cielo u otros datos significativos, recomendar qué algoritmo habría que utilizar para tratar de conseguir la mayor información posible, para lo que se podrían utilizar algunas técnicas de naturaleza inteligente, tal como aprendizaje automático.

# CAPÍTULO 9: Bibliografía

Arlow, J., Neustadt, I. UML 2. Anaya, 2006.

Dropbox (accedido Junio 2012). www.dropbox.com/

Google Docs (accedido Junio 2012). https://docs.google.com/?tab=wo

Guijarro, M., Pajares, G., Riomoros, I., Herrera, P.J., Burgos-Artizzu, X.P., Ribeiro, A. Automatic segmentation of relevant textures in agricultural images. Computers and Electronics in Agriculture 75, 75–83, 2011.

JAI (accedido Junio 2012).
http://java.sun.com/javase/technologies/desktop/media/jai/

JMathPlot (accedido Junio 2012).
http://jmathtools.sourceforge.net/doc/jmathplot/html/main.html.

Librería Swing (accedido Junio 2012).
http://docs.oracle.com/javase/6/docs/api/

Otsu, N. A threshold selection method from gray-level histogram. IEEE Transactions on System Man and Cybernetics 9, 62–66, 1979.

Pajares, G. y de la Cruz, J.M. Ejercicios resueltos de visión por computador. Ra-Ma, 2007.

Pajares, G. y de la Cruz, J.M. Visión por computador: Imágenes digitales y aplicaciones. 2º edición. Ra-Ma, 2007.

Pajares, G., de la Cruz, J.M., Molina, J. M., Cuadrado, J., López, A. Imágenes digitales: Procesamiento práctico con Java.Ra-Ma, 2003.

Riomoros, M. I. Segmentación automática de texturas en imágenes agrícolas. Proyecto Fin de Máster. Facultad de Informática, Universidad Complutense de Madrid, 2010 (disponible E-prints Complutense).

Zitinski, P. Y. Fusión de imágenes mediante wavelets para extracción de las partes verdes en imágenes agrícolas. Proyecto Fin de Máster. Facultad de Informática, Universidad Complutense de Madrid, 2011 (disponible E-prints Complutense).

# ANEXO I: Contenido del CD-ROM e instalación de la aplicación.

En el primero de los anexos se detalla en el contenido del CD-ROM entregado junto con la memoria y cómo poner en funcionamiento el sistema desarrollado. El segundo de los anexos contiene el manual de usuario de la aplicación.

## Contenido del CD-ROM

El contenido del CD que se adjunta con la memoria, se organiza en varias carpetas, como se muestra en la figura A1.1: Ejecutable, Código fuente, Memoria, Documentación Javadoc, Imágenes:

- En la carpeta Ejecutable se encuentra un archivo denominado Verdes.jar necesario para ejecutar la aplicación.
- El código de la aplicación se encuentra en la carpeta Código Fuente.
- En la carpeta Memoria se encuentra la memoria que documenta todo el desarrollo del proyecto.
- Para posibles ampliaciones del proyecto se adjunta documentación generada por la aplicación Javadoc, comentando los paquetes y clases del código desarrollado.
- Se añade una carpeta con distintas imágenes para probar la aplicación.

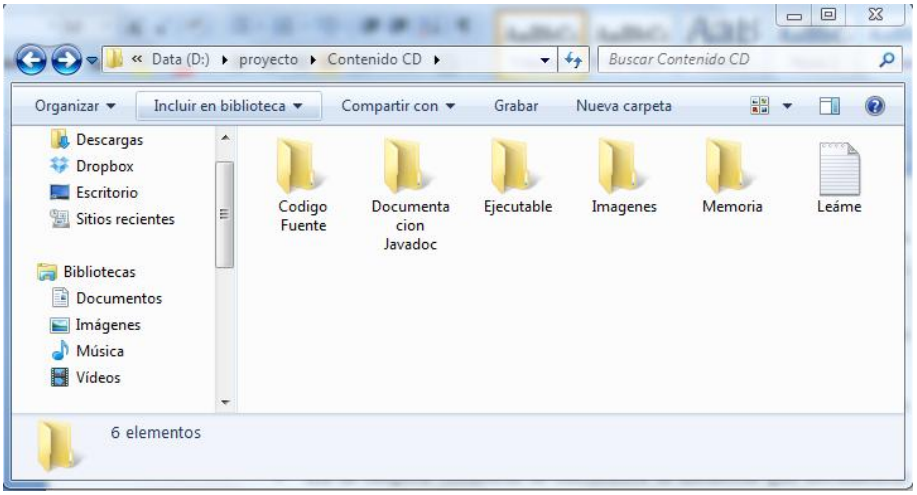



**Figura AI.1** Contenido CD.

## Instalación y ejecución:

Para poder utilizar la aplicación es necesario copiar la carpeta nombrada como Ejecutable, incluida en el CD adjunto, en el destino deseado dentro del disco duro del ordenador donde vaya a ejecutarse la aplicación. Dentro de esa carpeta se realiza doble click en el archivo VERDES.jar, quedando disponible para su inmediata utilización.

Se recomienda extraer todo el contenido del CD para mayor más facilidad a la hora de acceder a la documentación y tener a mano imágenes de muestra para probar la aplicación.

**Importante**: no ejecutar la aplicación desde el CD y no abrir las imágenes directamente desde el CD, utilizar imágenes que se encuentren en el disco duro del ordenador donde se ejecute la aplicación.

Las imágenes que se adjuntan en la entrega son una muestra, que para poder cargarlas en la aplicación, será necesario copiarlas antes en el disco duro del ordenador.

Para poder abrir archivos con extensión .jar es necesario tener instalada la máquina virtual de java (JVM) en el ordenador donde se ejecute la aplicación, la cual puede descargarse desde el siguiente enlace:

http://www.java.com/es/download/

# ANEXO II: MANUAL DE USUARIO

La aplicación está formada por una interfaz gráfica que se compone de dos ventanas, la inicial, que se muestra en la figura AII.1 y es la que aparece al iniciar la aplicación, y la ventana principal, que permite al usuario ejecutar y realizar la mayoría de las funcionalidades de la aplicación.

## VENTANA INICIO

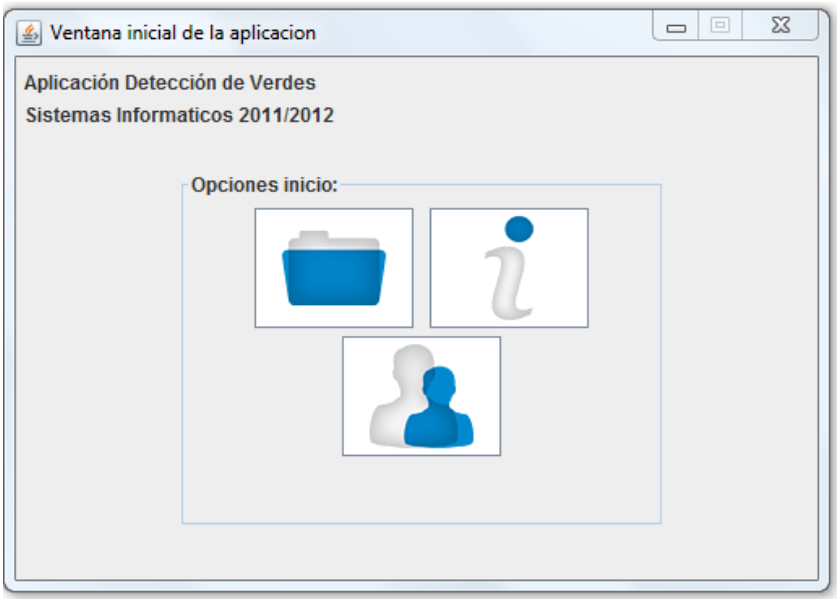



**Figura AII.1** Ventana inicial de la aplicación.

La ventana inicial presenta tres botones:

- El primero de ellos permite cargar una imagen, y navegar por la ventana principal de la aplicación.
- El segundo sirve para mostrar el manual de usuario en caso de que hubiera alguna duda en el momento de utilizar la aplicación.
- El tercero proporciona información sobre los autores del proyecto.


**Figura AII.2** Cuadro de diálogo sobre los autores del proyecto.

## CARGAR IMAGEN

Antes de poder aplicar los algoritmos de extracción de verdes o ejecutar los distintos tipos de binarizaciones a una determinada imagen, necesitamos cargarla, para lo que utilizamos la opción correspondiente dentro de la ventana inicial. Aparece un cuadro de diálogo para seleccionar la ruta de la imagen que queremos abrir para su posterior tratamiento, tal como se muestra en la figura AII.3:

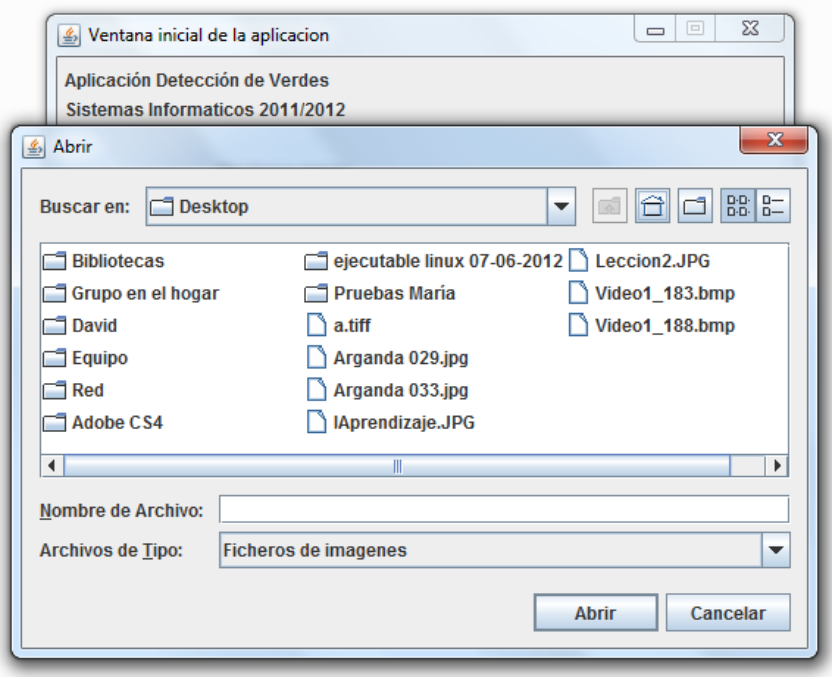



**Figura AII.3** Selección de imagen.

Esta ventana nos permite seleccionar la imagen y abrirla, para ello se deberá pulsar el botón *Abrir*, o si por el contrario deseamos volver a la ventana de inicio, pulsamos *Cancelar*. Tras elegir la ruta donde se encuentra ubicada la imagen y pulsar *Abrir*, si anteriormente se había cargado otra imagen, nos aparecerá una opción que nos pregunta si queremos eliminar las imágenes que se habían generado en la anterior ejecución, y que están almacenadas en una carpeta que se encuentra en la ruta donde se sitúa la imagen original. Si queremos conservar dichas imágenes, no se debe pulsar la casilla "Selecciones para borrar las imágenes generadas anteriormente".

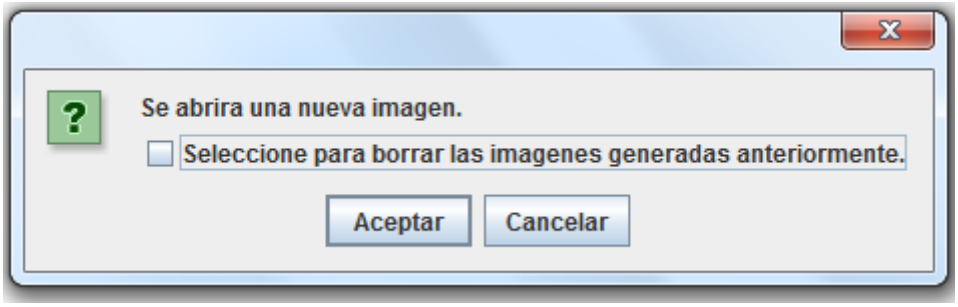



**Figura AII.4** Conservar imágenes.

Si por el contrario se quiere borrar, se selecciona la mencionada casilla:

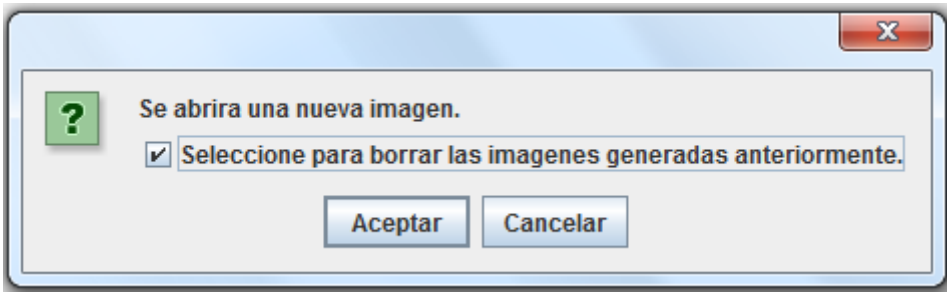



**Figura AII.5** Borrar imágenes.

Tras pulsar el botón *Aceptar*, ahora sí se abre la imagen que habíamos escogido. Esta acción nos permite navegar por la ventana principal, que irá mostrando las imágenes creadas como se muestra en la figura AII.6:

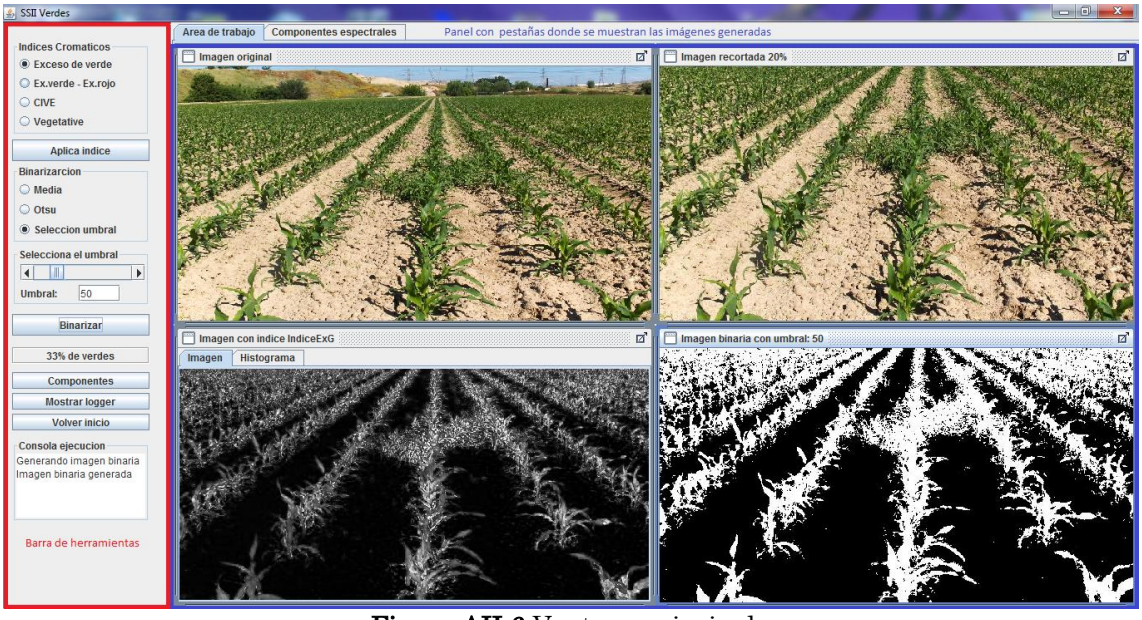



**Figura AII.6** Ventana principal.

# VENTANA PRINCIPAL

## RECORTE IMAGEN

Para eliminar información poco relevante a la hora de extraer los verdes (cielo, árboles), al abrir la imagen nos aparece un cuadro de opción, pidiéndonos qué porcentaje de la parte superior de la imagen queremos recortar, si no se quiere realizar recorte alguno se deberá seleccionar 0%. Tras seleccionar una cifra es necesario pulsar *Aceptar* para aplicar el recorte a la imagen, si se selecciona *Cancelar*, se aplicará un porcentaje de 20% por defecto.

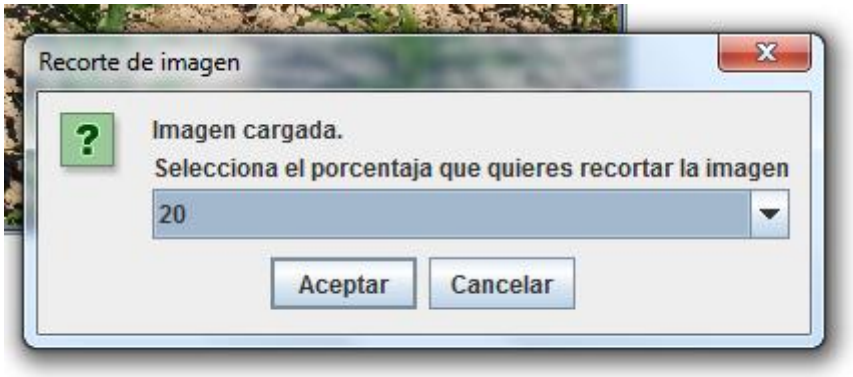



**Figura AII.7** Recorte imagen.

## BARRA HERRRAMIENTAS

Como puede verse en la figura AII.6, la ventana principal, está dividida en dos zonas: un panel con pestañas donde se muestran las imágenes, y en la zona izquierda una barra de herramientas que permitirá al usuario interactuar y ejecutar las funcionalidades de la aplicación.

Tras haber cargado y recortado la imagen, podemos empezar a tratarla. El siguiente paso consiste en aplicar los algoritmos de detección de verdes. En la barra de herramientas podemos seleccionar los distintos tipos de métodos de extracción de verdes implementados: Exceso de verde, Exceso de verde - Exceso de rojo, CIVE, Vegetative:

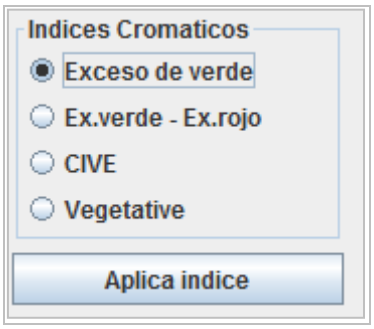



**Figura AII.8** Selección de algoritmo extracción de verdes.

Se puede seleccionar el algoritmo que se quiera utilizar y pulsar el botón *Aplica índice*, pudiéndose aplicar otros índices pulsando en el botón de radio correspondiente y volviendo a pulsar el botón. Se obtiene una nueva imagen que mostramos en un marco dentro del panel donde se encuentran el resto de imágenes, junto con su respectivo histograma.


**Figura AII.9** Marco con imagen tras aplicar índice.

Si se desea visualizar el histograma asociado a la imagen es necesario pinchar la pestaña *Histograma*:

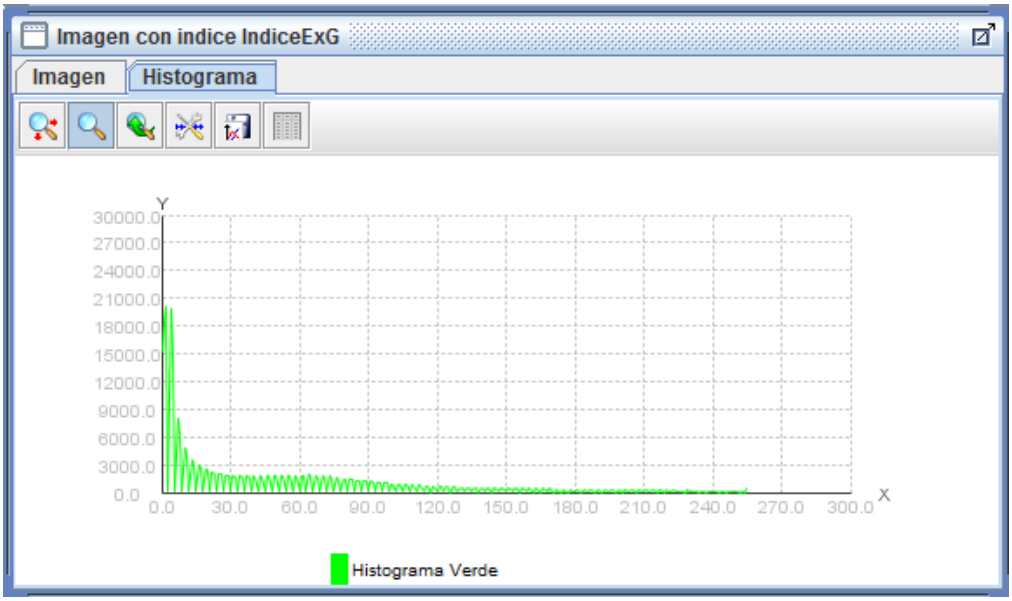



**Figura AII.10** Histograma de la imagen

.

Otra opción que nos permite la barra de herramientas es seleccionar el método de binarizar la imagen tras haberle aplicado el índice correspondiente, esta opción estará deshabilitada hasta el momento en que se aplique o ejecute algún método de detección de verdes descrito anteriormente. Como se muestra en la figura AII.11, la manera de seleccionar la binarización concreta es muy similar a lo explicado previamente en relación a los distintos métodos:


**Figura AII.11** Selección de binarización

Si queremos binarizar usando el método de la media pulsamos en el botón de radio *Media*, de la misma manera si queremos usar el método de Otsu, después pulsaremos el botón *Binarizar*. Existe otra manera de conseguir una imagen en blanco y negro, esa opción se consigue pinchando *Selección umbral*, permitiendo elegir el valor del umbral de manera manual, sin cálculos como los anteriores métodos. Todo píxel menor que el umbral se muestra de color negro, mientras que los que sean mayores o iguales se representan en color blanco, considerándose verde.

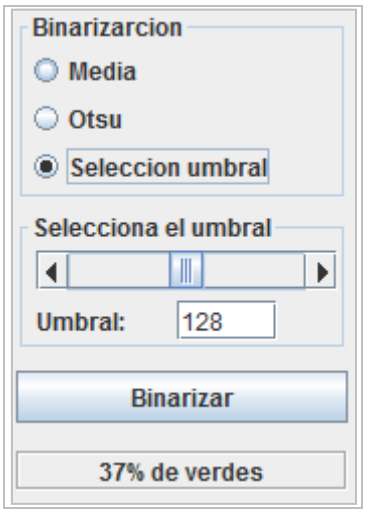



**Figura AII.12** Selección de umbral

Existen dos formas de introducir el umbral, cuyo rango es de 0 a 255, arrastrando el *scrollbar* hasta obtener el valor deseado, o introduciendo el umbral en el cuadro de texto. Tras haber introducido el valor pulsaremos *Binarizar*, con lo que obtendremos información del porcentaje de píxeles verdes obtenidos tras binarizar, en cuyo momento se genera una nueva imagen en blanco y negro:


**Figura AII.13** Marco con imagen binaria

Se genera un nuevo marco con la imagen en blanco y negro, en la parte superior del marco se muestra el umbral seleccionado a mano o el calculado por cualquiera de los otros métodos disponibles.

Existen otras opciones disponibles dentro de la barra de herramientas:

- Extraer las componentes de la imagen.
- Visualizar el *logger* de la aplicación, que recoge información sobre las acciones realizadas durante la ejecución.
- Volver a la ventana inicial.


**Figura AII.14** Botones que permiten realizar otras acciones.

Pulsando el botón *Componentes* se generan las componentes espectrales roja, verde y azul de la imagen original. Se muestran en una nueva pestaña donde se pueden observar las tres imágenes, acompañadas por sus correspondientes histogramas.

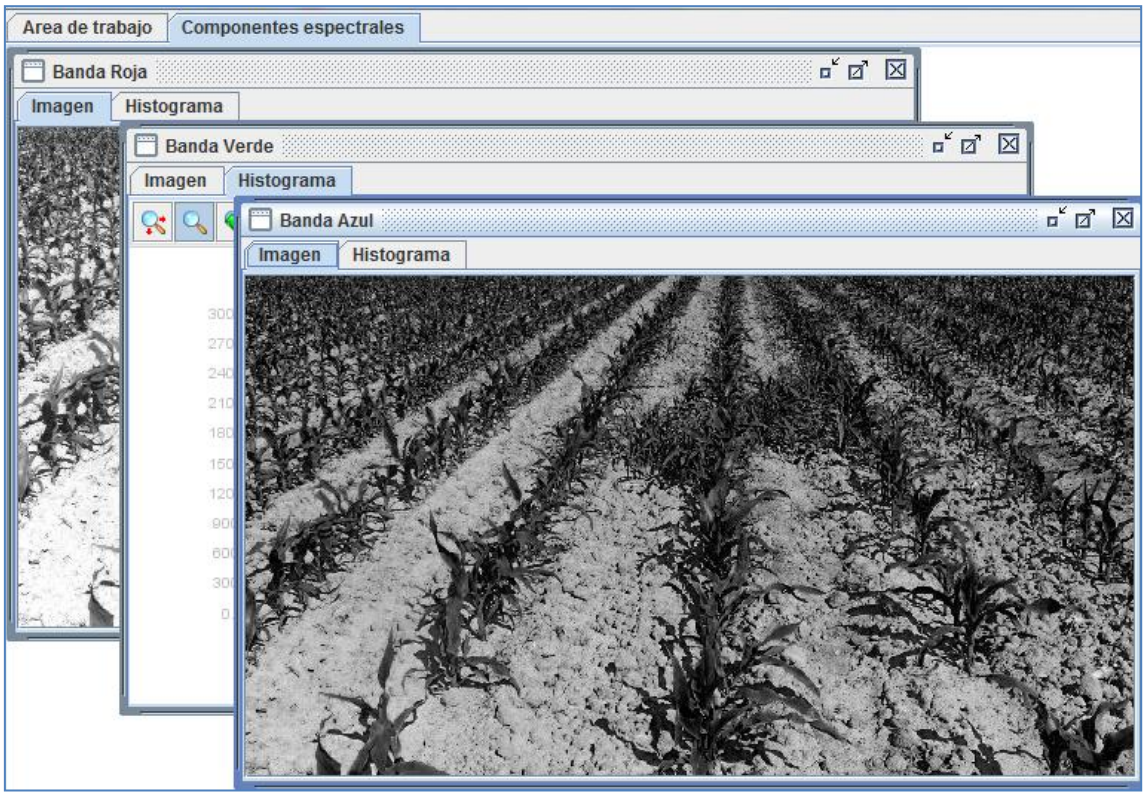



**Figura AII.15** Pestaña componentes espectrales

Como se ha explicado, se pueden ver todas las acciones realizadas desde que se inicia la aplicación mediante el botón *Mostrar Logger*.

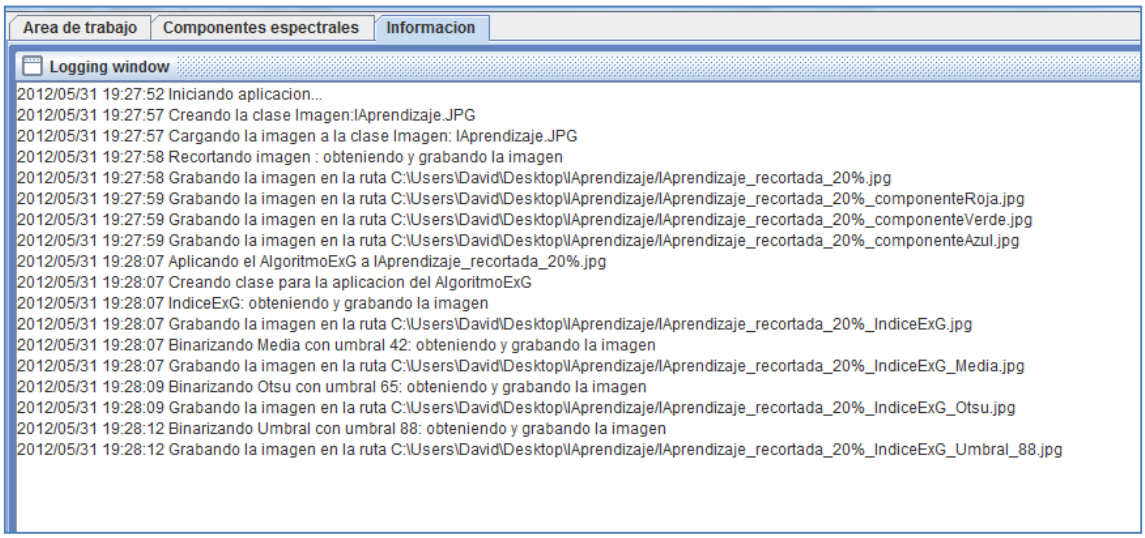



**Figura AII.16** Logger

La opción *Volver inicio*, nos permite retornar a la ventana de inicio y realizar las opciones ya conocidas por si queremos cargar otra imagen o consultar la ayuda. La apariencia de la interfaz de inicio es similar a lo comentado previamente, pero ahora muestra un botón que nos permite volver a la ventana principal:

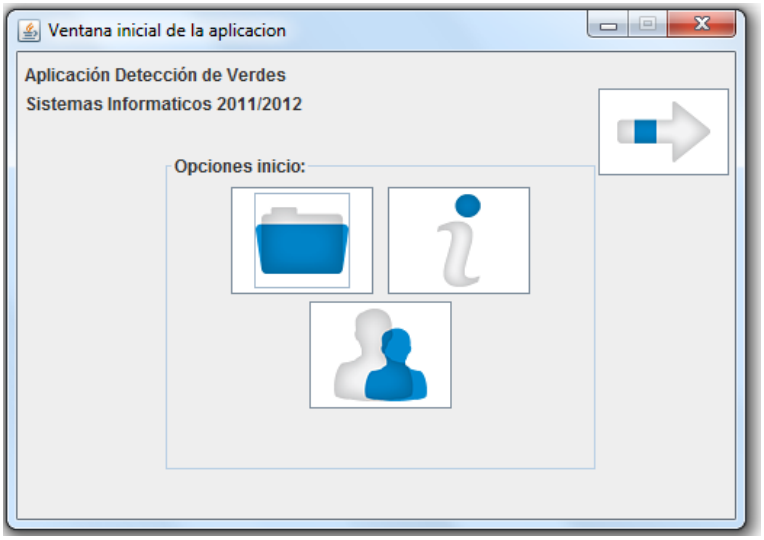


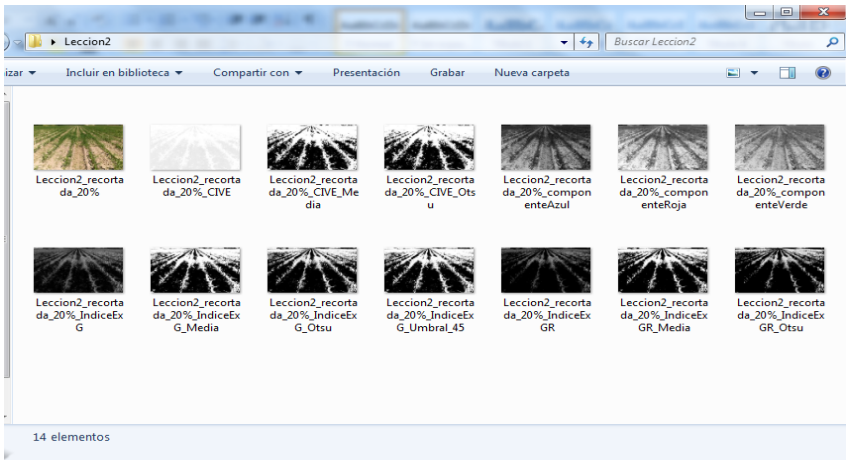

**Figura AII.17** Ventana inicio con opción de volver a la ventana principal.

Pinchando el botón que contiene una flecha podemos volver a la ventana principal de la aplicación.

## Otras consideraciones

Todas las imágenes generadas se guardan en una carpeta, y tras cerrar la aplicación o tras abrir una nueva imagen se da la opción de borrarlas o conservarlas.

La carpeta, que se crea en la misma ruta donde se encuentra la imagen original cargada en la aplicación, se nombra de la misma manera que la imagen. Cada una de las imágenes tiene nombres descriptivos, por ejemplo: Imagen_recortada_20%_IndiceExG_Media, indica que es una imagen recortada al 20% a la que se le ha aplicado el índice ExG, y binarizada mediante el método de la media.




Dentro de la ventana principal, en la parte inferior de la barra de herramientas, se puede encontrar un área de texto, en la que se muestra el estado en que se encuentra la ejecución de alguna acción: extracción de componentes, recorte de imagen, binarización o aplicación de índices.

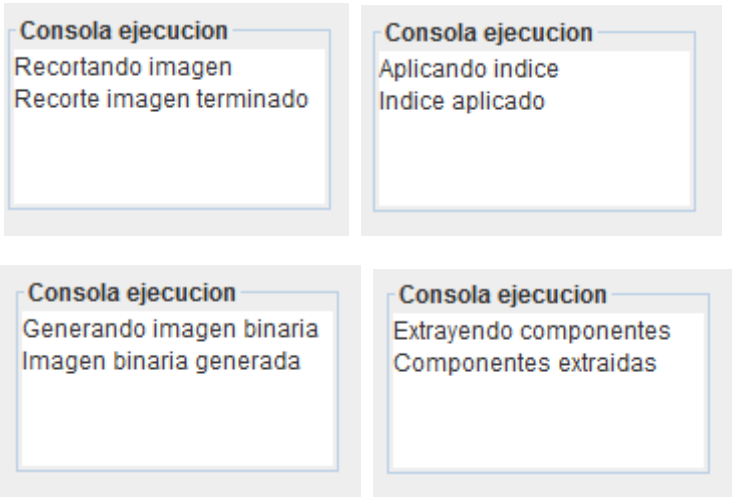



**Figura AII.18** Consola de ejecución con varios estados.

Por ejemplo, si queremos obtener las imágenes roja, verde y azul, mientras se ejecutan las acciones necesarias nos muestra el mensaje "Extrayendo componentes", al mostrar el resultado se añade un nuevo mensaje "Componentes extraídas".

Existe la posibilidad de maximizar las imágenes mostradas en los marcos pulsando el icono que se encuentra en la parte superior derecha del marco, identificado con el símbolo siguiente: 


**Figura AII.19** Maximizar imagen.