Sistemas Informáticos
Curso 2004-05

RECONOCIMIENTO DE TEXTURAS NATURALES MEDIANTE TÉCNICAS DE VISIÓN POR COMPUTADOR

Alberto Gamero Lancha
Mª José Guerrero San Miguel
José Manuel Sánchez Granero

Dirigido por:
Prof. Gonzalo Pajares Martinsanz
Dpto. Sistemas Informáticos y Programación

Facultad de Informática
Universidad Complutense de Madrid

Por medio del presente documento, D. Alberto Gamero Lancha, con DNI 11846758, Dña. MªJosé Guerrero San Miguel, con DNI 50875023 y D. José Manuel Sánchez Granero, con DNI 50983523,

autorizan a la Universidad Complutense 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.

FDO: Alberto Gamero FDO: Mª José Guerrero

FDO: José Manuel Sánchez

LISTA DE CONTENIDOS

Pág.

1. Resumen

La memoria presenta una aplicación basada en el lenguaje JAVA para el procesamiento específico de imágenes con una serie de funcionalidades asociadas. La aplicación consta de dos partes esenciales: procesamiento e interfaz gráfico.

La parte de interfaz gráfico tiene como finalidad el intercambio de información entre usuario y aplicación. Dado que el tipo de datos a procesar son imágenes, este interfaz se caracteriza por poseer una serie de ventanas donde se muestran las imágenes y otros datos relevantes asociados.

La parte de procesamiento comprende una serie de rutinas específicas encaminadas a tratar las imágenes entrantes. Las rutinas son específicas del proceso en el que se enmarca el proyecto dentro de una aplicación más amplia en agricultura de precisión, formando parte de un proyecto del CSIC, destacando: filtrado, binarización y tratamientos morfológicos. El diseño software de esta parte está pensado para su posterior integración en la aplicación propia del CSIC permitiendo la incorporación de nuevas funcionalidades futuras.

Para la realización del proyecto se han tenido en cuenta las fases del ciclo de vida de desarrollo que comprende la Ingeniería del Software, con un diseño modular y escalable con el fin de permitir su continuación en el futuro.

2. Abstract

This report presents a software application based on JAVA language. It is intended for a specific image processing and consists of several associated functionalities. The application has two main parts: processing and graphical interface.

The goal of the graphical interface is the interchange of information between the user and the own application. As the data types to be processed are images, this interface is based on a set of viewers where the images and other relevant associated data are displayed.

The processing part consists of several specific routines directed toward the processing of the incoming images. These routines are specific of the main process where this work is embedded, i.e. under a broad application based on agricultural precision as a part of a Spanish CSIC (Consejo Superior de Investigaciones Científicas) project. Some of these routines are: filtering, binarization, morphological processing. The software design of this part has been carried out thinking in its future integration under the amin CSIC application. This allows the incorporation of new future functionalities.

The software development has been carried out taking into account the live cycle in the Software Engineering. As this application must be extended in the future it has a modular and scalable design.

Palabras clave: Ingeniería del Software, diseño orientado a objetos, procesamiento de imágenes, JAVA, binarización, morfología.

3. Introducción

El proyecto se inicia dentro de la asignatura de Sistemas Informáticos, en el marco de una colaboración entre el Consejo Superior de Investigaciones Científicas (CSIC) y el Departamento de Sistemas Informáticos y Programación de la Facultad de Informática, siendo el profesor director de este proyecto también investigador principal en el ámbito de la UCM. El proyecto se ha presentado a la última convocatoria del Plan Nacional de I+D+I para el trienio 2003-2007.

Desde el CSIC se plantea la necesidad de desarrollar un sistema robótico automatizado para agricultura selectiva, entendiendo por tal los tratamientos encaminados a la fumigación de campos de cultivo de cereales para eliminar las malas hierbas, en una primera fase el objetivo es la eliminación de la gramínea denominada “avena loca”.

Este tipo de tratamientos conlleva una doble problemática, a saber: a) un elevado riesgo de contaminación medioambiental por el hecho de utilizar herbicidas altamente tóxicos; b) un elevado coste del tratamiento por el elevado precio de los productos fitosanitarios utilizados. Por este motivo se plantea tanto la necesidad de localizar exactamente la ubicación de las malas hierbas como la densidad de las mismas. El propósito es fumigar sólo en los puntos donde se localizan y con la cantidad justa.

Es el propio CSIC quien plantea el sistema robótico a utilizar en el ámbito de este proyecto. Se trata de un tractor dotado con un sistema de visión en su parte delantera, que proporcionará información visual del campo de cultivo, un equipo Hardware de procesamiento avanzado, y un depósito de herbicida en la parte posterior del tractor. El sistema visual proporciona las imágenes que han de ser tratadas mientras que el equipo de procesamiento ha de tener la información disponible para cuando el depósito llegue a la zona de concentración de malas hierbas identificada por el sistema visual y de procesamiento. Este será el tiempo real considerado en la aplicación, es decir, el tiempo suficiente como para que el tractor no avance más deprisa que el procesamiento de la información.

Se trata de un proyecto realmente ambicioso, que no puede abordarse íntegramente desde la perspectiva de la asignatura de Sistemas Informáticos.

Aunque el CSIC está interesado solamente en el procesamiento de las imágenes, es decir, en los métodos de tratamiento de las mismas, el planteamiento del proyecto tiene una doble vertiente: por un lado se desarrollará una serie de funciones que son las que luego se integrarán en la estructura del sistema del CSIC y por otro se desarrolla una estructura propia del proyecto o envoltura que permite integrar dichas funciones con el propósito de pruebas de validación e integración.

En resumen, el proyecto que se presenta en esta memoria integra bajo una aplicación las funcionalidades requeridas por la aplicación del CSIC. Se trata pues de un proyecto de desarrollo Software, que ha de ser lo suficientemente versátil y escalable como para que en el futuro pueda darse continuidad al mismo.

La presente memoria está organizada de forma que en la sección 2 se exponen las fases del ciclo de vida que comprende el proyecto, en la sección 3 se pone el énfasis en el análisis y diseño del sistema; finalmente, en la sección 4 se exponen las conclusiones y el trabajo a desarrollar en el futuro como continuación necesaria del proyecto.

Imagen del Robot con el sistema instalado

4. Ciclo de vida del proyecto

4.1 Descripción del sistema

El sistema consiste en una serie módulos que, interactuando entre si, permite la identificación de los diferentes surcos de cultivo y que es utilizado para la detección de la presencia de “malas hierbas”. En caso de obtener una detección positiva el sistema indicaría al dispositivo correspondiente la necesidad de emplear herbicida.
Debido a la implementa en JAVA del sistema permite ser compatible con distintas plataformas como puede ser LINUX o WINDOWS.
El sistema consiste en un paso inicial para la investigación de un problema concreto como es el estudio de la visón por computador de campos de cultivo que permita ahorrar el empleo de herbicidas.

4.2 Requisitos del proyecto total

Los requisitos del proyecto global en el que se engloba el nuestro son los siguientes:

- REQ1: Identificar los surcos en un cultivo de cereal
- REQ2: Detección de “malas hierbas” entre los surcos
- REQ3: Detectar la densidad de las malas hierbas con el fin de determinar la cantidad de herbicida para fumigar
- REQ4: Implementar el desarrollo en una plataforma universal, con la posibilidad de tratamiento remoto y con posibilidad de tratamiento distribuido.
- REQ5: Los procesamientos tenderán a realizarse en tiempo real como proyección de futuro

Debido a la gran envergadura de este proyecto y a su densidad nos hemos ocupado de una parte concreta que ha sido la identificación de “malas hierbas” entre surcos identificados anteriormente. Los requisitos más concretos sobre la parte en que hemos estado concentrados se describen a continuación.

4.3 GESTIÓN DE CONFIGURACIÓN

4.3.1 Documentación

4.3.1.1 Formato de desarrollo

Todos los archivos han de ser desarrollados en Microsoft Word o un editor de textos similar al mismo.

4.3.1.2 Convenio de nombre de archivos

Cada uno de los archivos, presenta el siguiente esquema de nombramiento:

<nombre_archivo> - <SI>-<versión>.<ext>

donde *<nombe_archivo>* indica el tipo de documento en cuestión,<SI> indica que el documento pertenece a la asignatura de Sistema Informáticos y <versión>el numero de la versión de ese mismo archivo. La extensión <ext>, dependerá si el archivo se encuentra en desarrollo o su versión final. Si se encuentra en desarrollo será .doc; en la versión de entrega y presentación, la extensión será .pdf

4.3.1.3 Entrega

Cada uno de los documentos realizados se han integrado constituyendo esta documentación y memoria final.

4.3.2 PROTOCOLOS DE CREACIÓN DE LOS ARCHIVOS

Los documentos que se realicen en el seno del grupo deberán seguir el siguiente proceso:

- Creación del documento siguiendo la plantilla mencionada en el punto anterior
- Validación por parte del resto de miembros del grupo
- Una vez aprobado, el documento será almacenado por cada uno de los miembros del grupo, teniendo así cada uno una copia de seguridad del citado documento

4.3.3 Almacenamiento de los archivos

Todos los archivos son almacenados por todos los miembros del grupo, de tal forma, que en caso de producirse una pérdida de datos por parte de algún miembro del grupo, no supondrá un problema para el desarrollo de la aplicación. Así mismo, los documentos realizados podrán ser enviados a la cuenta de correo siguiente:

Proyecto_csic@yahoo.es

4.3.4 CREACIÓN Y ESTRUCTURA DE LOS ARCHIVOS DE CÓDIGO

4.3.4.1 Creación

Toda la aplicación es desarrollada en Java, utilizando para ello el software JBuilder.

Cada uno de los archivos que se realicen, formarán parte de un proyecto común dentro de Jbuilder.

4.3.4.2 Estándar de código

La estructura que se debe seguir, es la siguiente:

- Cabecerá de archivo

```
/**
 * Titulo:
 * Descripción:
 * Copyright: Plustarts (c) 2005
* @author Alberto Gamero Lancha, MªJosé Guerrero San Miguel, José Manuel Sánchez Granero
 * @version 1.0
 */
```

- Implementación de Método

```
/**
* Javadoc asociado al metodo
**/
public void setTiempoEjecucion(){
    Double endTime= new Double(System.currentTimeMillis());
    double total=endTime.doubleValue()-logica.getInittime();
```

```
    statusBar.setText("Tiempo ejecucion : "+new
Double(total).toString()+" milisgs");
    statusBar.repaint();
  }
```

4.3.5 COMUNICACIÓN ENTRE LOS MIEMBROS DE PLUSTARTS Y EL PROFESOR

4.3.5.1 Comunicación

Todo contacto con el profesor Gonzalo Pajares se realizará a través de email siendo el correo utilizado el siguiente:

Proyecto_csic@yahoo.es

Así mismo, los miembros del grupo utilizaremos el teléfono para cualquier comunicación externa, realizando reuniones periódicas en la Facultad para comprobar el desarrollo de acuerdo a la planificación del estado en el que se encuentra la aplicación a implementar.

4.4 Riesgos y Calidad

4.4.1 . IDENTIFICACIÓN DE RIESGOS

Riesgo	Tipo de Riesgo	Descripción
PÉRDIDA DE DATOS	**producto**	Posibilidad de que los datos sean irrecuperables a consecuencia de fallos informáticos como caídas de web, intrusiones de virus en el sistema...
DEFECTOS EN LOS MÓDULOS DE SOFTWARE	**producto**	Que los módulos software que se deban reutilizar tengan defectos que limiten su utilidad, teniendo que

		volver a desarrollar otros nuevos
TECNOLOGÍAS DESCONOCIDAS	**producto**	El proyecto desarrollado tal vez requiera un determinado software o hardware que el equipo desconoce
COMPLEJIDAD DE CÓDIGO	**producto**	El código realizado por algún miembro del grupo, puede ser poco legible, poco comentado o incluso muy complicado
PROYECTO OBSOLETO	**producto**	El proyecto realizado puede quedar desfasado debido a la aparición de nuevas herramientas mas potentes (HW/SW)
CRECIMIENTO DESMESURADO DE LA APLICACIÓN EN REQUERIMIENTOS DE MEMORIA	**proyecto y producto**	Posibilidad de que las memoria necesaria para terminar la aplicación supere la impuesta

4.4.2. ANÁLISIS DE RIESGOS

Riesgo	Probabilidad	Efectos y seguimiento
PÉRDIDA DE DATOS	**moderada**	catastróficas
DEFECTOS EN LOS MÓDULOS DE SOFTWARE	**moderada**	tolerables
TECNOLOGÍAS DESCONOCIDAS	**Muy alta**	tolerables
COMPLEJIDAD DE CÓDIGO	**moderada**	tolerable
DIFERENCIAS DE VERSIONES	**moderada**	serias

(COMPATIBILIDADES)		
PROYECTO OBSOLETO	**muy baja**	serias
CRECIMIENTO DESMESURADO DE LA APLICACIÓN EN REQUERIMIENTOS DE MEMORIA	**moderada**	serias

4.4.3. PLANIFICACIÓN DE RIESGOS

Riesgo	Estrategia a seguir
PÉRDIDA DE DATOS	Llevar copias de seguridad de todos aquellos trabajos realizados, tanto físicamente, como en diferentes dispositivos de almacenamiento (discos diferentes, disquetes, etc.)
PROBLEMAS CON EL EDIFICIO (LABORATORIOS)	Buscar diferentes estaciones de trabajo, que reúnan condiciones similares a las de los laboratorios (encontrar ordenadores con el mismo sistema, requerimientos, etc.)
DEFECTOS EN LOS MÓDULOS DE SOFTWARE	Someter a consenso el modulo defectuoso, decidiendo si debe ser modificado por otro miembro del grupo o desechado.
TECNOLOGÍAS DESCONOCIDAS	Uno o varios miembros del grupo elaborarán información y manuales sobre la nueva tecnología, poniéndose al día previamente sobre ella mediante la compra de libros, explicaciones de alguien documentado, etc.

COMPLEJIDAD DE CÓDIGO	Cada uno de los miembros del grupo deberá atenerse al estándar de código y las normas previamente establecidas en la Gestión de configuración. En caso de que la dificultad se encuentre en el propio código (algoritmos complejos), el autor del mismo, deberá detallarlo lo máximo posible al resto de componentes del grupo.
DIFERENCIAS DE VERSIONES (COMPATIBILIDADES)	Hacerse con las últimas versiones sobre las que se va a trabajar en los laboratorios.
PROYECTO OBSOLETO	Intentar aprovechar al máximo la tecnología de que se dispone para evitar que posteriores avances hagan que el proyecto quede demasiado desactualizado
CRECIMIENTO DESMESURADO DE LA APLICACIÓN EN REQUERIMIENTOS DE MEMORIA	Asumir las especificaciones y usar estructuras de datos que permitan un uso eficiente de memoria

4.4.4. SEGUIMIENTO DE RIESGOS

Riesgo	Seguimiento
PÉRDIDA DE DATOS	Pérdida de documentos pendientes de entrega implica su retraso, rehacer documentos, perdida de tiempo que podría haber sido empleado en avanzar

	trabajo.
PROBLEMAS CON EL EDIFICIO (LABORATORIOS)	Intentar no depender de la tecnología que se encuentra en los laboratorios para, en caso de no poder ser utilizados, no tener que detener el desarrollo del proyecto
DEFECTOS EN LOS MÓDULOS DE SOFTWARE	Intentar ir integrando cada poco tiempo todos los módulos y hacer una comprobación de su funcionamiento
TECNOLOGÍAS DESCONOCIDAS	Búsqueda de documentación necesaria y estudio de dicha documentación
COMPLEJIDAD DE CÓDIGO	Establecer estándares de código en la gestión de configuración.
DIFERENCIAS DE VERSIONES (COMPATIBILIDADES)	Tratar de probar siempre el proyecto en los lugares donde éste pueda llegar a ser utilizado
PROYECTO OBSOLETO	Tratar de hacer que el proyecto no dependa demasiado de la versión de los programas utilizados, para que una mejora de éstos provoque también una mejora en el proyecto
CRECIMIENTO DESMESURADO DE LA APLICACIÓN EN REQUERIMIENTOS DE MEMORIA	Seguir la especificación y estructuras de datos eficientes

4.4.5 REFLEJO EXTERNO DE LA CALIDAD

4.4.5.1 ASPECTO GRÁFICO

Una buena apariencia de la interfaz gráfica de usuario hará más agradable la interacción con la aplicación, al tiempo que permitirá una utilización intuitiva y eficaz de la aplicación y los recursos que ésta ofrece al usuario.
El acceso a las acciones más comunes debe ser lo más rápido posible, permitiendo que la utilización de la aplicación sea lo más dinámica posible.

4.4.5.2 CONSUMO DE RECURSOS

El consumo de recursos del sistema debe ser acorde con la envergadura del proyecto desarrollado, y constituirá un objetivo prioritario durante todo el desarrollo el lograr minimizar este consumo.
Deberemos tener en cuenta, puesto que la aplicación en sí deberá adaptarse a un funcionamiento en tiempo real.

4.4.5.3 AYUDA DE LA APLICACIÓN

La aplicación cuenta con una ayuda en la que se especifica cómo llevar a cabo todas y cada una de las acciones posibles, de una forma sencilla e intuitiva.

4.4.6 REFLEJO INTERNO DE LA CALIDAD

Para lograr este objetivo prioritario externo (que el cliente y el usuario percibirán externamente), la calidad y su control se verán reflejados en todos los aspectos internos del proceso de desarrollo del proyecto.

*4.4.6.*1 CÓDIGO

Así pues, identificamos otro punto importante - en cuanto a calidad se refiere- en el código: una buena estructuración que permita comprender, reestructurar y reutilizar los diferentes módulos desarrollados. En este sentido, para controlar el desarrollo de la aplicación contemplaremos en todo momento los casos de uso ya implementados. Usaremos estándares de codificación, a fin de obtener un formato uniforme en todo el código desarrollado por diferentes miembros del grupo. Éste estándar definirá desde la nomenclatura de variables y métodos, hasta el uso de tabulaciones y formatos de estructuras sintácticas (uso de llaves, comentarios, etc.).
Este aspecto cobra especial relevancia teniendo en cuenta que la envergadura del proyecto podría permitir a grupos de cursos posteriores continuar su desarrollo, por lo que una buena estructuración del código y el seguir unos determinados estándares ayudará a dicha ampliación del proyecto.

4.4.6.2 DOCUMENTACIÓN DE CÓDIGO

La documentación del proyecto debe ser lo más clara y concisa posible. Así mismo debe realizar una completa descripción del documento tratado. Para ello usaremos plantillas de documentación adaptadas a métodos, clases, interfaces y paquetes.

Generaremos documentos html en los que se explique claramente tanto la funcionalidad como el modo de uso de cada método.
También este punto es de especial importancia si el proyecto es posteriormente continuado por grupos de cursos posteriores.

4.4.6.3 REUNIONES DE REVISIÓN DE CALIDAD

Desde el comienzo de la realización del proyecto elaboramos un plan de calidad que consistía en lo siguiente:
Antes del cumplimiento de un plazo para la entrega de un documento o prototipo, se celebra una reunión anterior donde se realiza la integración del trabajo correspondiente a cada miembro. En dicha reunión se decide cuándo se producirá la revisión de calidad del material entregado. En una reunión de revisión de calidad se procede a exponer el orden del día, debatiendo para cada punto las distintas propuestas de los miembros del equipo y adoptando una solución definitiva para cada punto. Después de dicha reunión se decide quién solucionará cada uno de los posibles problemas encontrados.
A lo largo del periodo de desarrollo todos los miembros del grupo están atentos a posibles mejoras en la aplicación.
Todas las pautas de dicho plan de calidad han sido seguidas durante la realización del proyecto.

4.5 VISIÓN DEL SISTEMA Y DESARROLLO

El proyecto consistió en el desarrollo de un sistema de tratamiento de imágenes, utilizando para ello el lenguaje de programación Java. En este punto realizamos un enfoque desde el punto de vista del usuario, entendiendo como tal al sistema del cual formará parte nuestro desarrollo, así podemos observar las siguientes funciones:

Carga de imágenes: El sistema permite cargar diversos tipos de imágenes que son reducidas para su visualización

Selección rango de verde: El sistema permite al usuario realizar la selección de lo que podemos determinar “verdadero verde”, es decir, el color verde que considera el usuario necesario emplear para que el sistema funcione con fiabilidad.
Selección tipo de calculo de distancia: el sistema permite elegir la distancia que empleará en sus cálculos para obtener distintas aproximaciones, así se podrá elegir dos tipos de distancias la euclidea y la de Mahalahobis.

Selección rango-distancia: Permite al usuario definir cual será la distancia en la medida euclidea de los colores para determinar cuales de ellos pueden ser considerados verdes.

Selección de matriz de erosión y dilatación: Permitirá obtener una binarización más fiable eliminando posibles ruidos introducidos por los cálculos explicados anteriormente.

4.6 REQUISITOS SOBRE TRATAMIENTOS DE ERRORES DEL SISTEMA

En este punto exponemos las capacidades que presenta nuestro sistema para actuar en caso de error o de mal funcionamiento. Todas los posibles fallos podrán ser observados mediante la interfaz, permitiendo al sistema recuperar se de ellos.

4.7 MEDIDAS DE TIEMPOS

Debido a la gran cantidad de datos tratados por el sistema se espera un tiempo de respuesta bastante alto debido a la gran cantidad de arrays de bytes a tratar. El sistema medirá el tiempo tratado en las diversas operaciones a modo de información para el usuario.
El tiempo es un punto fundamental en el tratamiento de imágenes de cara a la evolución futura del sistema. En nuestro sistema pues se cuenta con un rango concreto de respuesta hasta la llegada de una nueva imagen.
Se probará con distintos tipos de imágenes para ver el comportamiento para cada una de ellas, así se puede esperar un mayor tiempo de respuesta en imágenes grandes que en pequeñas.

4.8 REQUISITOS DE SEGURIDAD

En general no se dispondrá de ninguna seguridad especial en el empleo del sistema debido a que esta destinado a su empleo en investigaciones sucesivas.
En cuanto a la propia de seguridad del sistema en cuanto fiabilidad se probará concienzudamente para evitar bloqueos innecesarios de tanto el propio sistema como del sistema operativo sobre el que esta funcionando.

4.9 REQUISITOS SOFTWARE y HARDWARE

Será necesario que la máquina donde se instale nuestra aplicación disponga del espacio necesario en disco (5 Mb aprox.) y de un ratón. Creemos que son requisitos tan mínimos que no el sistema podrá ser empleado en casi cualquier ordenador actual, pero la velocidad de respuesta si dependerá de la potencia de la maquina sobre la que trabajemos.

Nuestra aplicación deberá ser totalmente compatible con cualquier sistema operativo, por lo que estará realizado en Java que permite, con su máquina virtual, ser independiente de la plataforma.
Ése es precisamente el principal motivo de nuestra elección, además de que es un lenguaje familiar para todos los miembros del equipo.
Como requisitos software para la posible ejecución del procesador de texto, basta con tener un sistema operativo que soporte la instalación de la máquina virtual de Java y, por supuesto, que esté instalada.

4.10 REQUISITOS DE USUARIO

El empleo de la interfaz será muy sencillo pero el usuario tendrá que poseer conocimiento del tratamiento de imagen que desea realizar. Por ejemplo, un usuario que no conozca en que consiste el empleo de la distancia euclidea o de Mahalahobis no sabrá a ciencia cierta como emplear el sistema. Para ello se adjuntará una ayuda a parte de la documentación del sistema para los usuario no familiarizados con el sistema.

4.11 REQUISITOS DE DATOS

4.11.1. DATOS DE ENTRADA

Serán imágenes del formato de tipo JPEG y BMP que permitirá al sistema tratarlas y sacarlas por pantalla binarizadas.

4.11.2. DATOS DE SALIDA

La salida del sistema se realiza por pantalla mostrando la binarización de la imagen introducida por pantalla, mas adelante se planteará el guardar dicha imagen o bien tratarla para poder sacar las formulas de las rectas que representan los surcos de los cultivos.

4.12. ESPECIFICACIÓN DE COMPONENTES FUNCIONALES

El sistema realizará una serie de funciones que se describen a continuación:

- **Carga de Imágenes**

Nuestra aplicación permitirá cargar desde el menú archivo, cargar una imagen del formato JPEG, GIF o PNG que será visualizada en el interfaz antes de su tratamiento, esta será la imagen fuente donde se aplicaran las distintas funcionalidades disponibles.

- **Selección de diferentes valores de verde**

El sistema permitirá definir al usuario el rango de colores que desea capturar para la binarización de la imagen. El usuario tendrá la capacidad de dar los valores de los componentes RGB de un color que se utilizará como base desde donde se tomarán todos los cálculos.

- **Selección con previsualización del tono de color**

El sistema dispondrá de una ventana que permitira al usuario previsualizar el color que tomará como origen en el tratamiento de la imagen, de esta forma el trabajo de selección no se realizará de forma tan intuitiva sino que con esta herramienta se conocerá antes del tratamiento el color concreto base.

- **Selección de la distancia empleada para la binarización**

El usuario del sistema podrá seleccionar que tipo de distancia quiere utilizar, bien la distancia euclídea o la distancia de Mahalohobis. La primera realiza un tratamiento más rápido pero de menor calidad ya que el propio usuario debe introducir el rango a partir del cual se despreciará los colores. El otro método es más lento pero de menor calidad.

- **Distancia euclidea**

En la distancia euclidea se dispondrá de un cuadro de texto donde se podrá definir el rango a partir del cual despreciar los colores que no cumplen unas determinadas condiciones.

- **Distancia Mahalohobis**

En la distancia de Mahalohobis se dispondrá de la capacidad de seleccionar la forma de calcular la distancia entre el color seleccionado y el rango que este método calcula directamente entre la distancia de Mahalohobis o la distancia euclidea.

- **Selección de mascara de erosión y dilatación**

El sistema permitirá seleccionar la matriz de erosión y dilatación empleada para la eliminación de ruido de esta forma se podrán realizar diferentes pruebas del sistema para comprobar su comportamiento el la aplicación de distintas mascaras.

- **Visualización de la imagen tratada**

El sistema permitirá previsualizar la imagen tratada después del proceso de filtrado aplicando las técnicas elegidas por el usuario.

- **Visualización del tiempo de tratamiento**

Debido a que el sistema ha sido diseñado para realizar el estudio del tratamiento de imágenes se dispondrá de la capacidad de observar el tiempo transcurrido durante el tratamiento de las imágenes, que normalmente será elevado.

- **Visualización de componentes RGB**

El sistema dispondrá de la capacidad de mostrar el contenido de los componentes RGB de la imagen original colocando el cursor en el píxel indicado.

- **Ayuda**

El sistema dispondrá de un botón de ayuda que permitirá ofrecer una grata orientación al usuario debido a los tecnicismos empleados en el sistema.

4.13 PLAN DE FASE

Cabe destacar 3 iteraciones:

- **Primera iteración**

- Comunicación con el profesor: Reunión con le profesor y con la encargada del proyecto de desarrollo del CSIC para indicar los puntos del proyecto.
- Análisis de riesgos
- Captura de requisitos de acuerdo con lo extraído en las reuniones con el profesor
- Especificación de casos de uso y servicios prestados por la aplicación
- Documentación y gestión de configuración
- Documento de calidad
- Repartición del trabajo a desarrollar en la documentación
- Diseño global de la aplicación
- Plan de desarrollo
- Aprendizaje de nuevas tecnologías
- Estudiar código a reutilizar
- Implementar versiones pequeñas de prueba para comprobar el control de nuevas tecnologías (JAI, análisis de arrays de pixeles, obtención de pequeñas binarizaciones, etc.)
- Entrega parcial del proyecto obteniendo una binarización con ruidos.
- Selección de objetivos que se deben cumplir en la segunda iteración.

- **Segunda iteración**

- Investigación para la eliminación de ruidos en la binarización de la primera iteración
- Reunión con el profesor para posibles orientaciones sobre como mejorar la eficiencia
- Investigación de métodos aplicables a la mejora del rendimiento
- Pruebas en la utilización de diferentes máscaras para la erosión y la dilatación de la imagen
- Revisión de la especificación de requerimientos para una posible ampliación
- Revisión de los diferentes diagramas de la aplicación
- Pruebas de comportamiento del sistema

- Aprobación por parte del profesor
- Entrega del segundo prototipo

➢ **Tercera iteración**

- Pruebas de carga y funcionamiento
- Mejoras en el comportamiento del sistema
- Preparación de presentaciones al tribunal
- Revisión de documentos con el fin de aclarar diversos puntos
- Retoques de implementación de mejoras de casos de uso
- Ampliación del diagrama de clases
- Presentación de prueba al profesor para posible orientación
- Repaso de los objetivos planteados de la asignatura
- Preparación del cd de entrega y de los manuales de usuario y de instalación al profesor
- Entrega y presentación final al tribunal.

5. ARQUITECTURA DE LA APLICACIÓN

5.1 OBLIGACIONES DEL DISEÑO DE SOFTWARE

5.1.1. SISTEMA OPERATIVO

En un principio usaremos Windows pero, dada la característica multiplataforma de Java (lenguaje de programación en el que desarrollaremos la aplicación), podremos utilizar nuestro procesador de texto en cualquier plataforma que admita la máquina virtual de Java.

5.2. OBLIGACIONES DEL DISEÑO DE HARDWARE

El sistema funcionará en las máquinas de los laboratorios con las consecuentes restricciones. Para obtener más información acerca de los equipos utilizados preguntar a los técnicos encargados del mantenimiento de los laboratorios.

5.3. OBLIGACIONES DEL INTERFAZ DE USUARIO

5.3.1. CARACTERÍSTICAS DE USUARIO

La aplicación esta destina para usuarios que posean conocimiento de ciertas teorías del tratamiento de imágenes y que les sirva como introducción en sus investigaciones futuras. El sistema esta destinado a servir como punto de inicio a posteriores investigaciones en el campo de la visión por computador y por lo tanto ha sido diseñado con este objetivo.

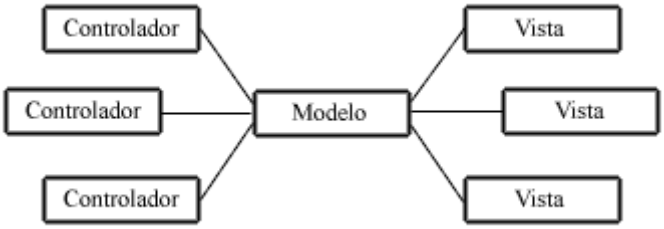
A parte de esto esta también diseñado para la integración con otros sistemas siendo totalmente independiente del interfaz, y de los distintos modulos permitiendo una integración fácil y sencilla.

5.4 ARQUITECTURA Modelo/Vista/Controlador

La arquitectura MVC (*Model/View/Controller*) fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales son que el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas.

Este modelo de arquitectura se puede emplear en sistemas de representación gráfica de datos, o en sistemas CAD, en donde se presentan partes del diseño con diferente escala de aumento, en ventanas separadas.

En la figura siguiente, vemos la arquitectura MVC en su forma más general. Hay un Modelo, múltiples Controladores que manipulan ese Modelo, y hay varias Vistas de los datos del Modelo, que cambian cuando cambia el estado de ese Modelo.



Este modelo de arquitectura presenta varias ventajas:

- Hay una clara separación entre los componentes de un programa; lo cual nos permite implementarlos por separado
- Hay un API muy bien definido; cualquiera que use el API, podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.
- La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

Al incorporar el modelo de arquitectura MVC a un diseño, las piezas de un programa se pueden construir por separado y luego unirlas en tiempo de ejecución. Si uno de los Componentes, posteriormente, se observa que funciona mal, puede reemplazarse sin que las otras piezas se vean afectadas. Este escenario contrasta con la aproximación monolítica típica de muchos programas Java. Todos tienen un *Frame* que contiene todos los elementos, un controlador de eventos, un montón de cálculos y la presentación del resultado. Ante esta perspectiva, hacer un cambio aquí no es nada trivial.

Definición de las partes

El *Modelo* es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

La *Vista* es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo.

El *Controlador* es el objeto que proporciona significado a las ordenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

6. DISEÑO DETALLADO

Para llevar a cabo esta aplicación, se ha aplicado una estructura modelo-vista-controlador que permitiera desarrollar modularmente el sistema de forma que fuera muy fácil dosificar el trabajo y permitir su ampliación en sucesivos desarrollos. Dado que el sistema que soporta la aplicación debe ser escalable.

El sistema ha estado sometido a cambios durante su diseño, ya que, hemos podido distinguir dos fases en cuanto a diseño. Una primera parte consistió en la ampliación del sistema según íbamos investigando las diferentes formas en las que se podría aplicar los diferentes compromisos del sistema a modo de prueba y otra segunda fase en la que se empezó a utilizar la estructura modelo-vista-controlador para permitir el desarrollo con las nuevas tecnologías aprendidas.

6.1 MOTIVOS DE LA DECISIÓN

La decisión de llevar a cabo el sistema con este tipo de estructura (modelo-vista-controlador) se debe a la propia experiencia en anteriores proyectos que nos permitió conocer su espectacular eficiencia en el hecho de repartir el trabajo y de la forma tan fácil que permite aceptar ampliaciones y cambios.

El sistema está pensado especialmente en eso, en los cambios que pueda sufrir en un futuro y en su ampliación; así este sistema consta básicamente de tres partes: un controlador que lleva precisamente las funciones de control del sistema solicitando las operaciones a los restantes módulos y que permite la interconexión entre la interfaz y el sistema, lo podemos considerar como el puente de mando de la aplicación; el interfaz diseñada para ser de aspecto amigable y fácilmente utilizable de forma que al usuario no le sea complicada; y por último los diversos módulos en los que se lleva a cabo las funcionalidades del sistema.

Como se ha mencionado anteriormente esta estructura era la que más se nos adecuaba a nuestro proyecto y por estos motivos decidimos este tipo de diseño que se explicará con detalle en los puntos siguientes.

6.2 CAMBIO DE ESTRUCTURA

El cambio de estructura fue un punto importante en el desarrollo de la aplicación. Pero, ¿por qué surgió la necesidad de un cambio de estructura? Pues fue debido a la necesidad de investigación de diferentes tecnologías que necesitábamos emplear en el desarrollo del proyecto como fueron las librerías JAI, las librerías JAMA y el estudio del tratamiento de las imágenes que llevaba JAVA. Así la estructura anterior consistía en una única clase principal la cual se auxiliaba de otras clases que implementaban las diferentes funciones según la tecnología estudiada. Debido a esto la estructura empezaba a crecer de manera caótica debido a que utilizábamos los módulos ya implementados en los que habíamos desarrollado diferentes estructuras de la aplicación para realizar pruebas con las tecnologías que investigábamos.

Por este motivo, fue necesaria una reestructuración del diseño del sistema. Se optó por seguir la opción de un diseño modular en el que cada parte funcional quedara separada definitivamente de forma que se pudiera repartir el trabajo de forma fácil y que la integración se realizara de manera poco costosa.

Así fue como decidimos realizar un cambio en la estructura de forma que el caos reinante durante la investigación de las tecnologías acabó en un diseño claro de fácil ampliación y más fácil integración.

6.3 DEFINICIÓN DE LA ESTRUCTURA DE LA APLICACIÓN

Como se ha adelantado anteriormente la estructura en la que se basa este sistema es en el modelo-vista-controlador que permite mantener totalmente separada la estructura de la funcionalidad del aspecto gráfico de la aplicación manteniendo un controlador como parte común que es quien decide como repartir las diferentes peticiones del sistema a los módulos correspondientes.

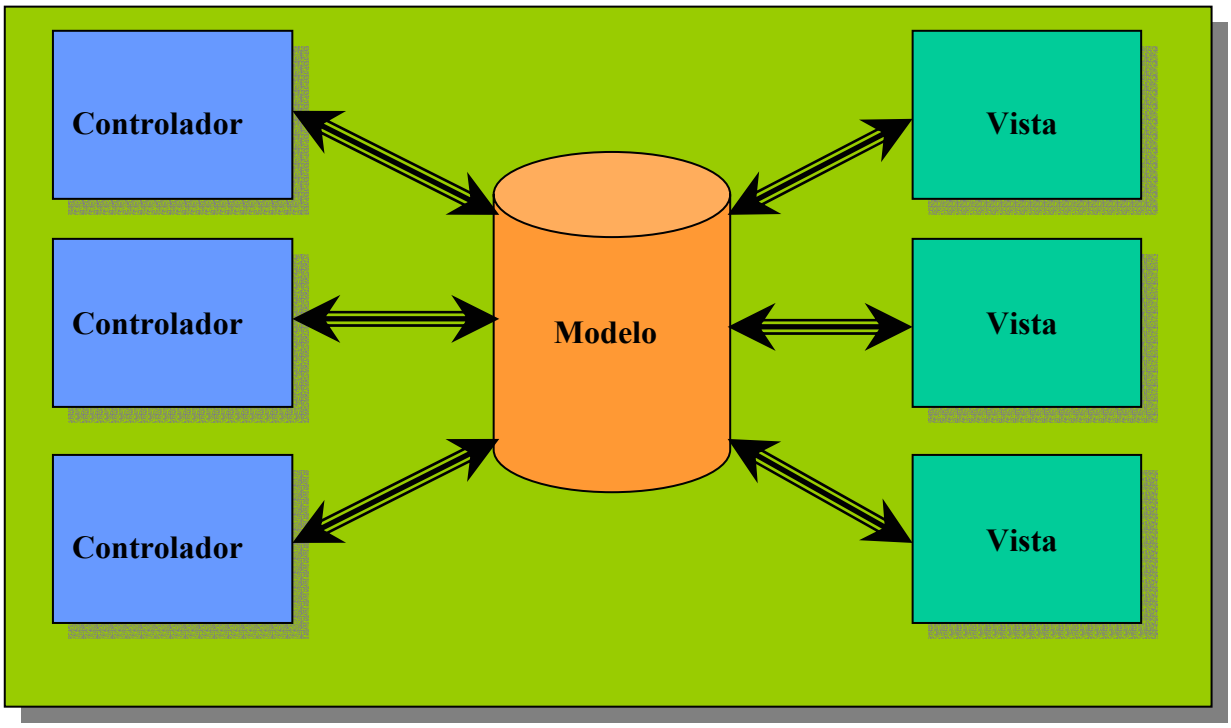


Figura 1 Modelo Vista -Controlador

Así de esta forma el módulo esta separado en distintos módulos:

- ***GUI*** (Graphical User Interface): Este módulo posee todos los aspectos relacionados con la interacción sistema-usuario y permite ciertas funciones básicas como es la de cargar imágenes y visualizar la ayuda. Presenta los siguientes ficheros:
    - Aplicación.java que contiene el main de la aplicación y de donde se arranca el sistema.
    - Interfaz.java que se trata del frame principal del sistema que visializa el usuario para interactuar con el sistema.
    - FiltroImagen.java que realiza un filtro para los tipos de archivos que puede cargar la aplicación.

- DialogoRGB.java que implementa un frame que permite seleccionar de mamnera visual el color que se utilizará posteriormente para la binarización de las imágenes usando el módulo distancias que se explica a continuación.
- Interfaz_AboutBox.java que implementa el típico frame “Acerca de..” que aparece en caso todas las aplicaciones.

- ***Control:*** Este paquete permite la interconexión del interfaz con los diferentes módulos del aplicación distribuyendo el trabajo, así la aplicación permite estar modulada ya que será el controlador el que es capaz de relacionar la funcionalidad con la petición del usuario. Esta compuesto por el siguiente archivo:

  - Controlador.java que permite la distribución del trabajo de la aplicación

- ***Píxel:*** Este paquete implementa el píxel básico que necesita la aplicación para su funcionamiento y está compuesto por una única clase:

  - Píxel.java que implementa un único píxel formado por cuatro componentes que son el valor alpha (transparencia), red (rojo), green (verde), blue (azul). Es capaz de a través de un valor entero(que representa un píxel en JAVA) descomponerlo en estos cuatro componentes para su fácil manejo además de otras utilidades como es comparar varios pixeles.

- ***Distancia:*** este paquete permite la implementación de la binarización del sistema a partir de varias distancias: la distancia Euclidea y la distancia de Mahalanobis. A través de un valor que se introduce por parámetro el sistema consigue binarizar la imagen a través de una serie de cálculos dependiendo de la función seleccionada por el usuario. Este módulo permite la ampliación de otras distancias heredando de la clase Distancia.java. Los ficheros que contiene son:

  - Distancia.java que es la clase base en la que se basa la distancia y que permite el calculo del píxel medio que necesitará cualquiera que se la elección del usuario.
  - Mahalanobis.java que implementa la distancia de mahalanobis que hereda de Distancia.java ya que necesita alguno de sus métodos para su funcionamiento. Permite binarizar utilizando la distancia de Mahalnobis.
  - Euclidea.java como la anterior hereda de Distancia.java he implementa la binarización a través de la distancia euclidea, a partir de un rango y un color definido por el usuario.

- ***Jama:*** es un paquete que implementa diferentes funciones con matrices que han sido necesarias para la realización de la implementación y hemos tenido que investigar para su utilización.

- ***Morfológicas:*** este paquete permite la implementación de las diferentes funciones morfológicas que se necesitaban para obtener una binarización correcta de la imagen original. Son operaciones complejas que necesitan la utilización de una clase auxiliar para que su realización sea más sencilla. Los ficheros que contienen estos módulos son:

- Morfologicas.java clase principal de la que debes heredar todas las posibles operación que se desean implementar.
- Erosion.java que implementa la operación morfológica de erosión (permite eliminar los puntos aislados en una binarización) y que hereda de Morfologicas.java ya que emplea funciones de ellas al igual que la clase que se explica a continuación.
- Dilatacion.java que también hereda de la clase Morfologicas.java y que implementa la operación morfológica de dilatación (compactación de puntos cercanos).
- Par.java que es la clase auxiliar que hemos mencionado anteriormente y que permite el tratamiento de imágenes como pares para que la implementación de estas operaciones sean más sencillas.

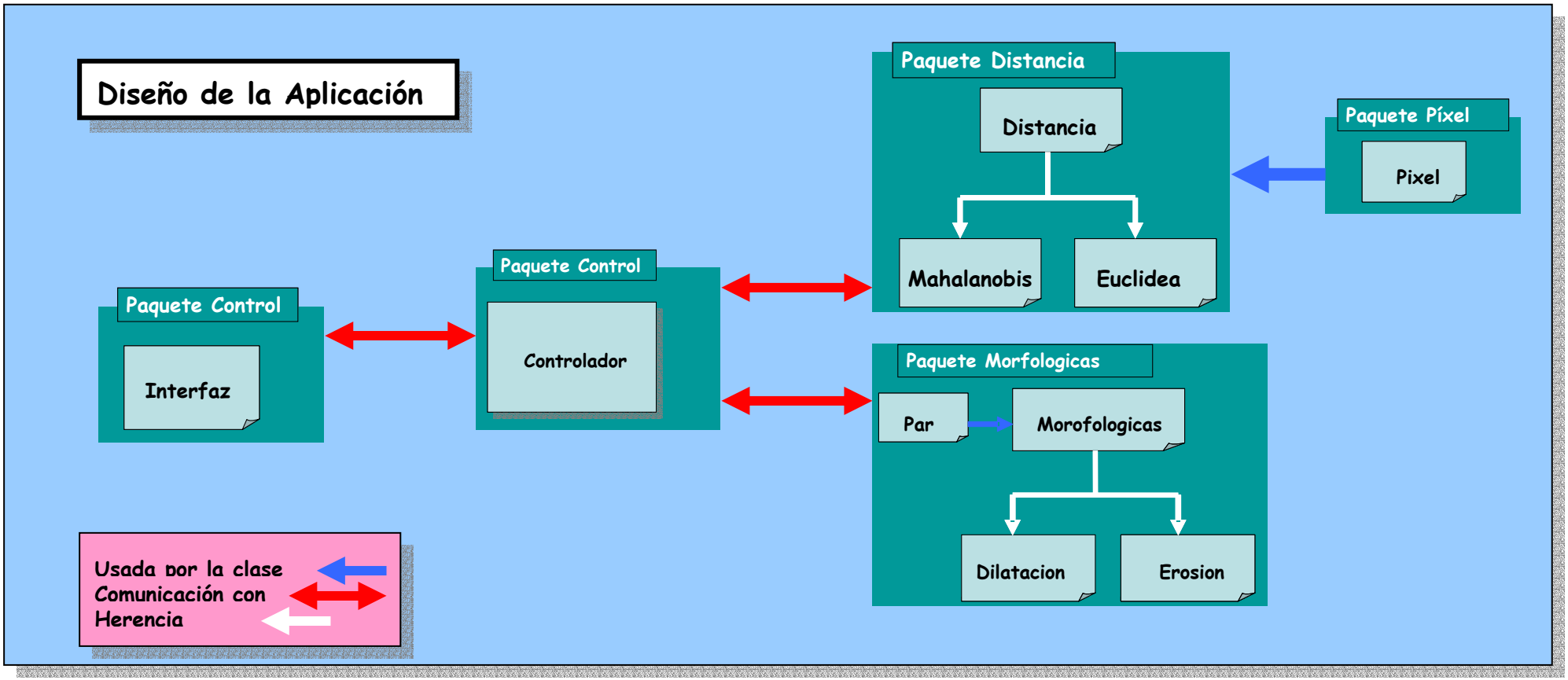
De esta forma queda definida la estructura que hemos definido para nuestro sistema con la idea de que la integración y la ampliación sea de forma sencilla.

Para una mayor información acerca de las clases y su estructura, ver **Anexo A**

6.4 CONCLUSIÓN

El diseño de esta aplicación ha estado pensada para facilitar la integración, el reparto del trabajo y la posibilidad de futura ampliaciones ya que la modulación del sistema es tal que por ejemplo con heredar de la clase Morfológicas.java se podría implementar otra operación morfológica que se incluiría en el caso de uso añadiendo una función al controlador y de la misma forma otro tipo de distancia heredando de Distancia.java.

6.5 Diseño DE LA APLICACIÓN



7. Plan de Pruebas

1.-Carga de una imagen

Descripción:

- Se procede a la carga de una imagen desde una ventana de explorador a través del sistema, obteniéndose una visualización de la imagen a tratar.

Pasos seguidos:

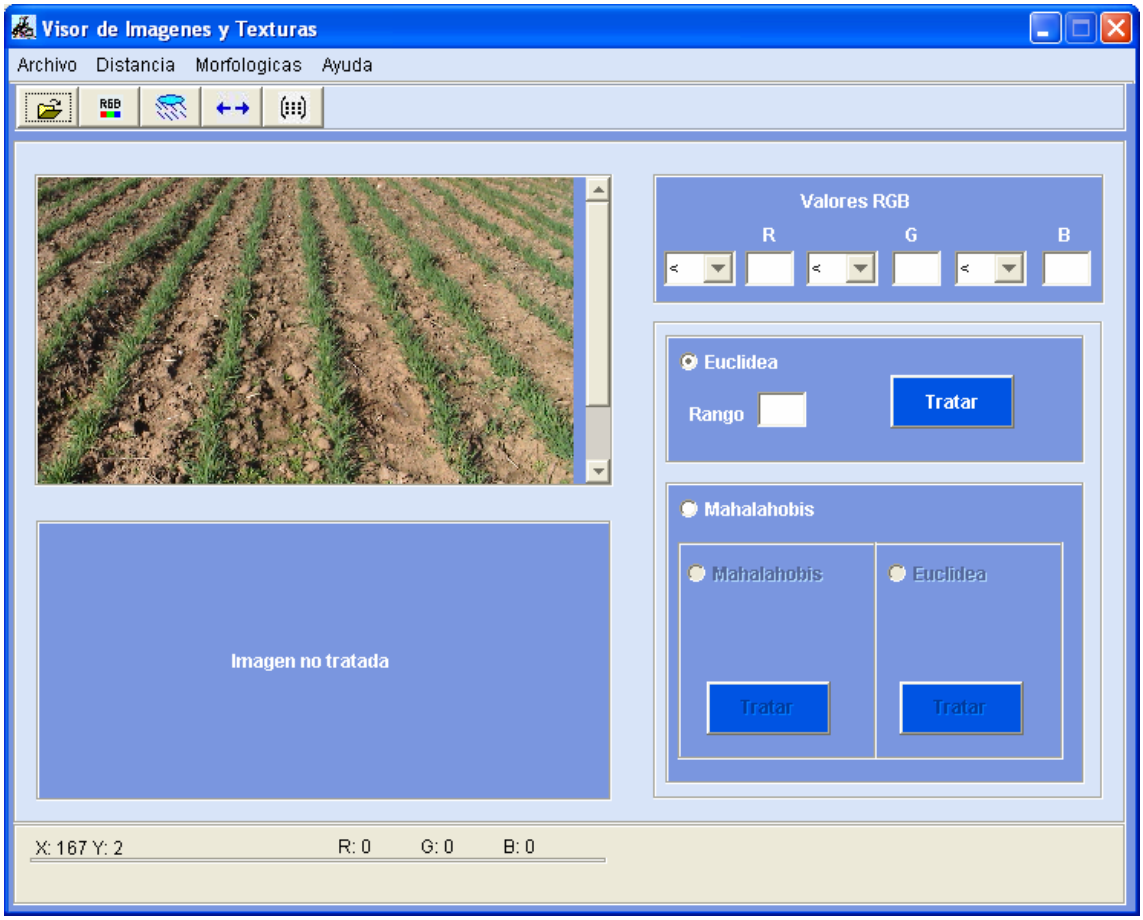
- Arranque del sistema
- Pulsación del botón “abrir imagen”

Resultados esperados:

- En el sistema aparece una pre-visualización de la imagen que se va a tratar
- Esta cargada en el sistema la imagen sobre la que se van a realizar las diversas funciones que presenta el sistema

Resultado obtenidos:

- Aparece un filtro de archivos para seleccionar imágenes por defecto
- Visualización de la imagen en tamaño más pequeño a la original adaptándose al tamaño del cuadro para su visualización
- La imagen se encuentra cargada en el sistema



2.-Selección de un rango de color

- A partir de este momento se supondrá que todos los pasos a seguir partirán desde el punto de la imagen cargada mencionado y comprobado anteriormente

Descripción:

- Se podrá seleccionar un rango de colores a partir de un cuadro de diálogo para seleccionar la gama de verde si el usuario lo desea.

Pasos Seguidos:

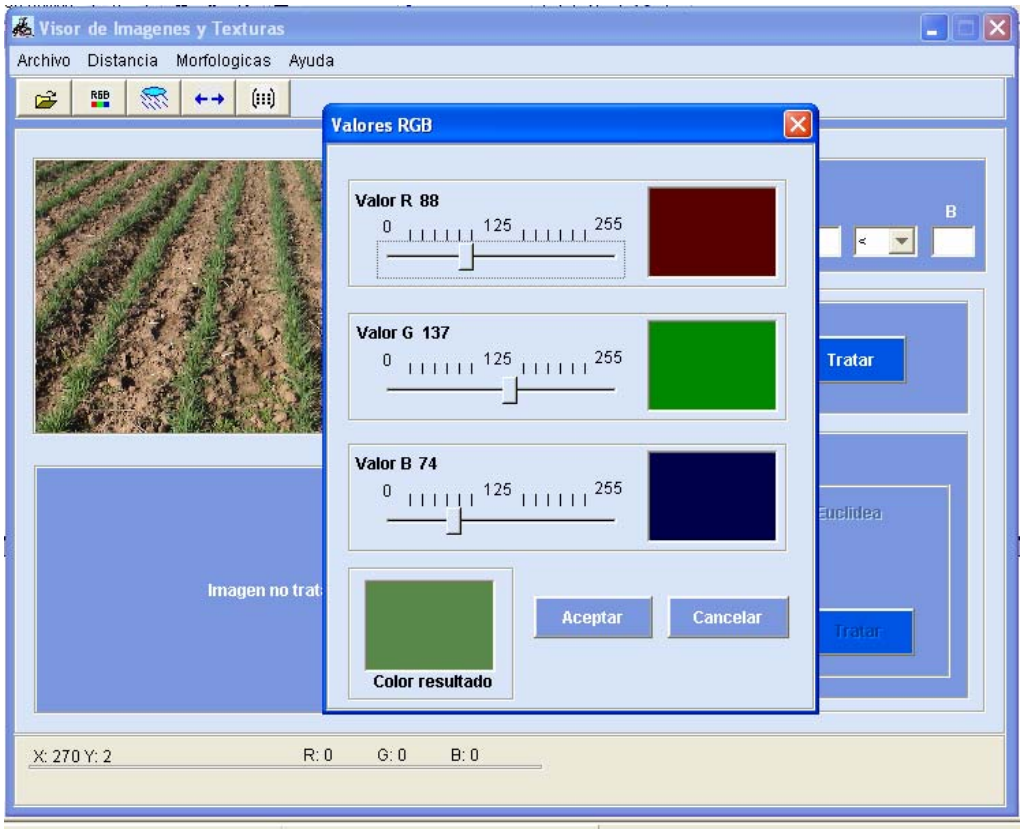
- Pulsar el botón “RGB”
- Seleccionar el rango de colores que se desea
- Pulsar aceptar

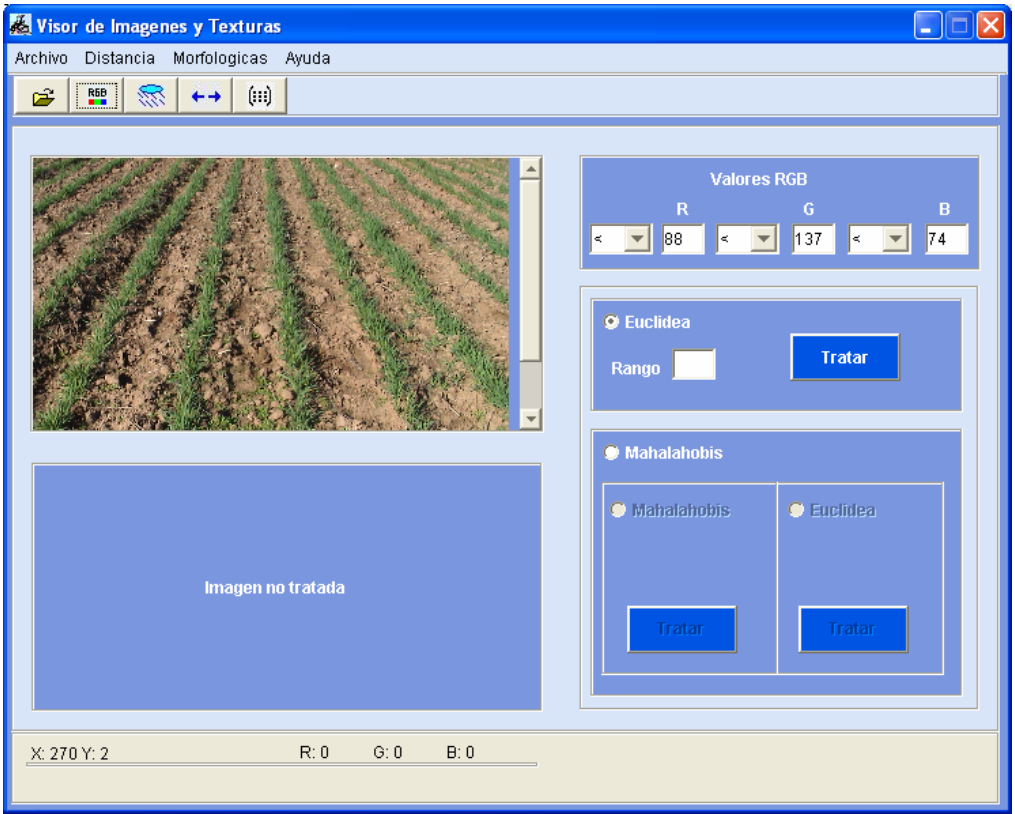
Resultado esperados:

- Se puede ir visualizando en el cuadro de diálogo el color elegido
- Al pulsar aceptar dicho valores son cargados en el sistema y se emplearán para las diferentes funciones a realizar

Resultado observado:

- Se visualiza correctamente el color elegido por el usuario
- Los valores son cargados correctamente en el sistema al pulsar aceptar





3.-Aplicación de la distancia euclidea

Descripción:

- Aplicación para la binarización de la imagen la distancia euclidea.

Pasos seguidos:

- Carga de imagen
- Introducción de valores RGB para la consideración del color:

	Red	Green	Blue
<	75		75
>		101	
=			

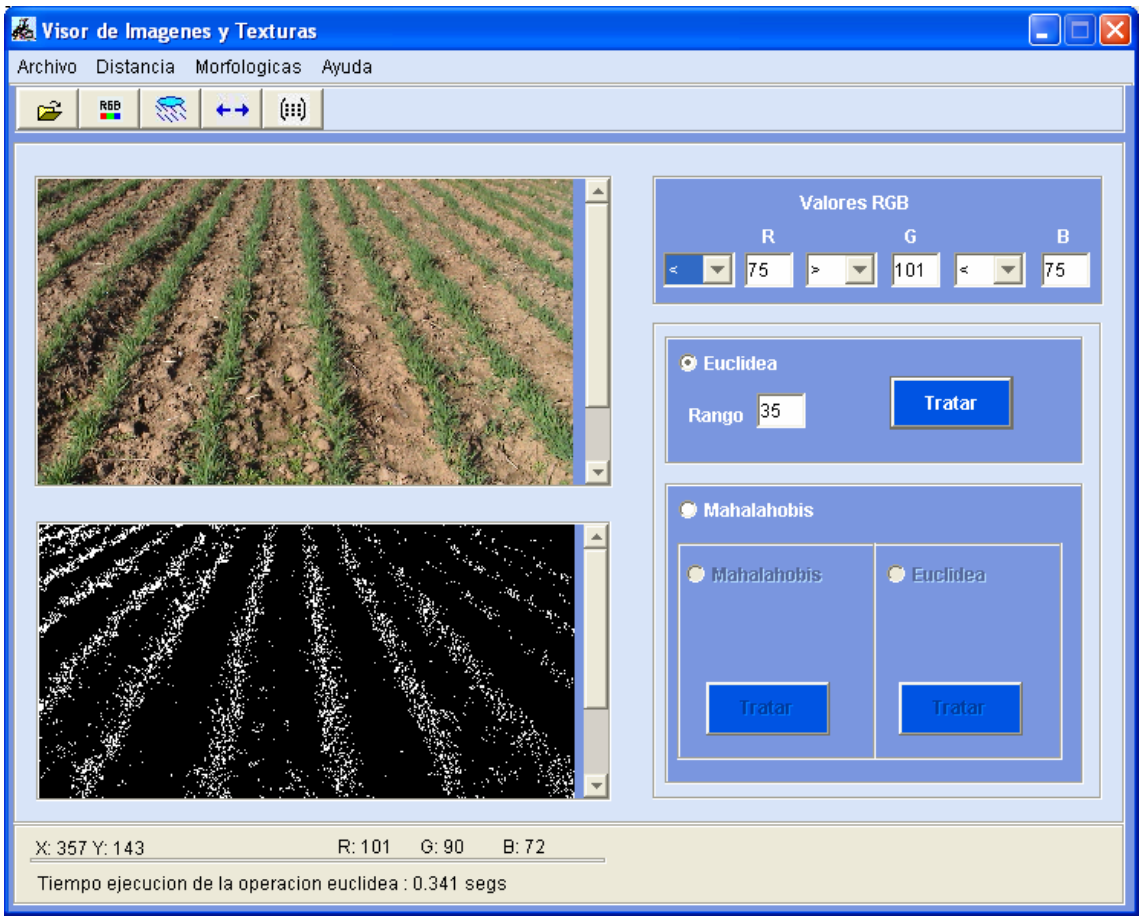
- Seleccionar la distancia euclidea
- Introduccion del rango de margen
- Pulsar botón “Tratar”

Resultados esperados:

- Obtención de una binarización de la imagen correcta
- Previsualización de dicha imagen
- Previsualización del tiempo de tratamiento de la imagen

Resultados Observados:

- Binarización correcta de la imagen debido a los rangos y valores obtenidos.
- Previsualización en orden al tamaño de la imagen original cargada.
- Aparición de ruido en la imagen.
- Observación de un tiempo de tratamiento de 0,341 seg.



4.-Aplicación de la distancia Mahalanobis-euclidea

Descripción:

- Aplicación de la distancia Mahalanobis-euclidea para la obtencion de la binarización de la imagen.

Pasos seguidos:

- Carga de la imagen
- Introduccionde valores RGB para la consideración del color:

	Red	Green	Blue
<	75		75
>		99	
=			

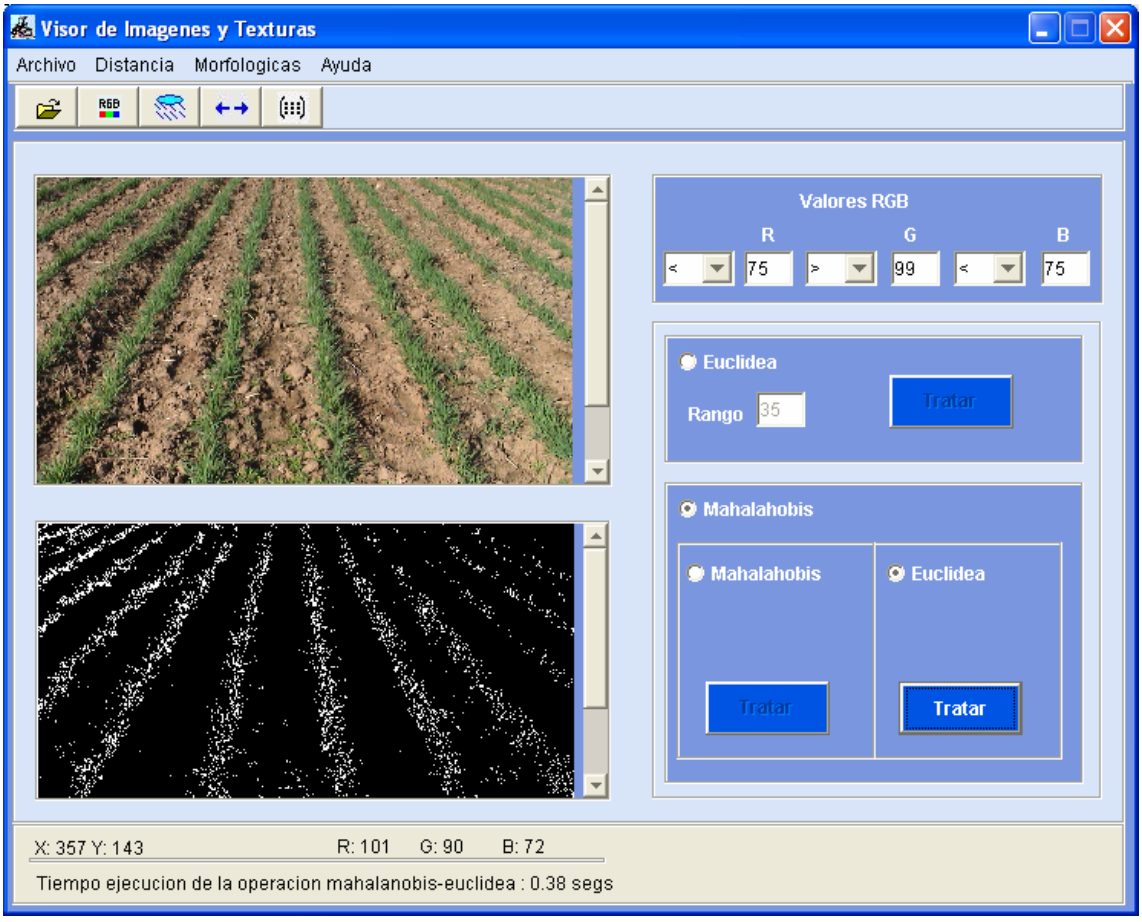
- Seleccionar la distancia Mahalanobis
- Seleccionar distancia Euclidea
- Pulsar botón “Tratar”

Resultados esperados:

- Obtención de una binarización de la imagen correcta con poco ruido, sin necesidad de introducir rangos de confianza
- Previsualización de dicha imagen
- Previsualización del tiempo de tratamiento de la imagen

Resultados Observados:

- Binarización correcta de la imagen debido a los rangos y valores obtenidos.
- Previsualización en orden al tamaño de la imagen original cargada.
- Aparición de una cantidad menor de ruido en la imagen.
- Observación del tiempo de tratamiento de 0,38 seg.



5.-Aplicación de la distancia Mahalanobis

Descripción:

- Aplicación de la distancia Mahalanobis para la obtencion de la binarización de la imagen.

Pasos seguidos:

- Carga de la imagen
- Introduccion de valores RGB para la consideración del color:

	Red	Green	Blue
<	75		75
>		99	
=			

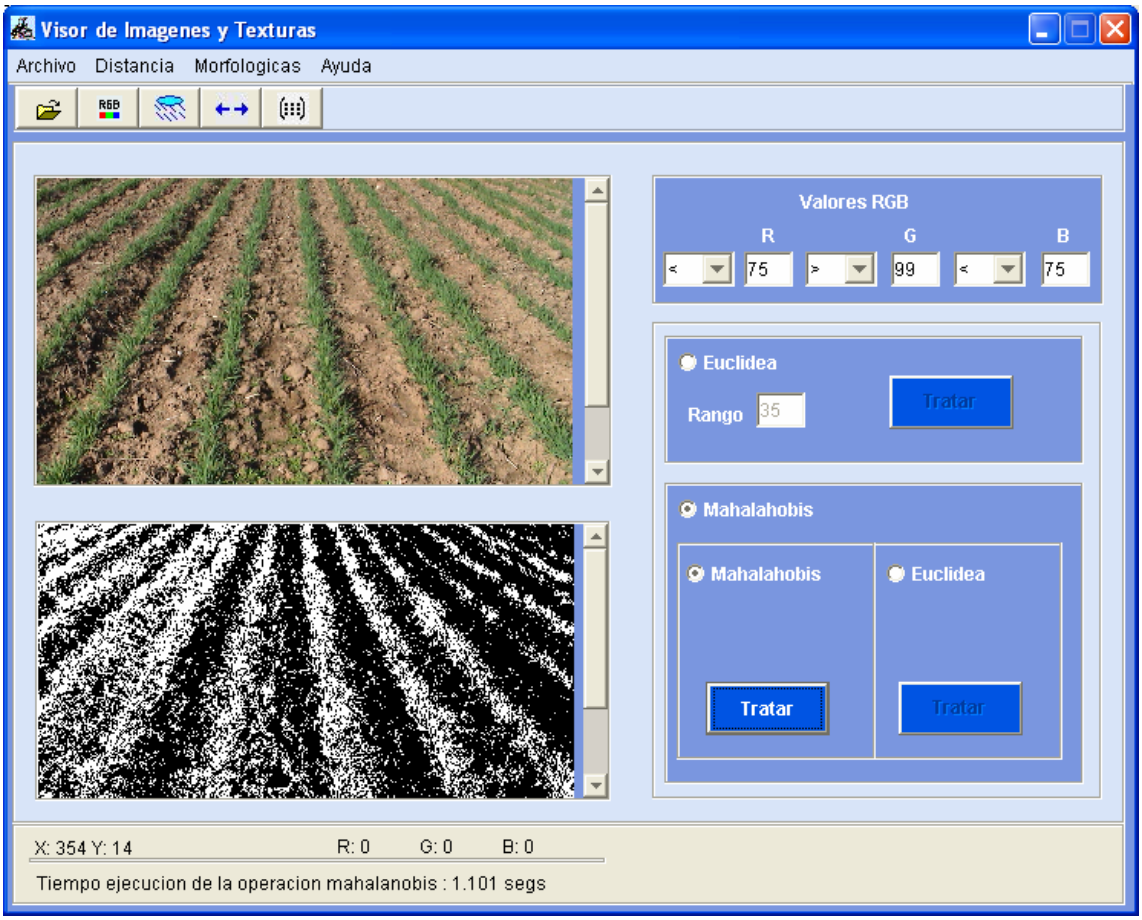
- Seleccionar la distancia Mahalanobis
- Seleccionar distancia Mahalanobis
- Pulsar botón “Tratar”

Resultados esperados:

- Obtención de una binarización de la imagen correcta, sin necesidad de introducir rangos de confianza
- Previsualización de dicha imagen
- Previsualización del tiempo de tratamiento de la imagen

Resultados Observados:

- Binarización correcta de la imagen según los rangos y valores obtenidos.
- Previsualización en orden al tamaño de la imagen original cargada.
- Aparición de gran cantidad de ruido en la imagen. (se podría probar con otros rangos de valores).
- Observación del tiempo de tratamiento de 1,101 seg.



6.- Aplicación de valores de RGB que excluyan muchos pixeles

Descripción:

- Seleccionar rangos para valores de RGB para que existan pocos pixeles durante la binarización.

Pasos seguidos:

- Carga de la imagen
- Selección de rangos RGB muy excluyentes:

	Red	Green	Blue
<	196		191
>		228	
=			

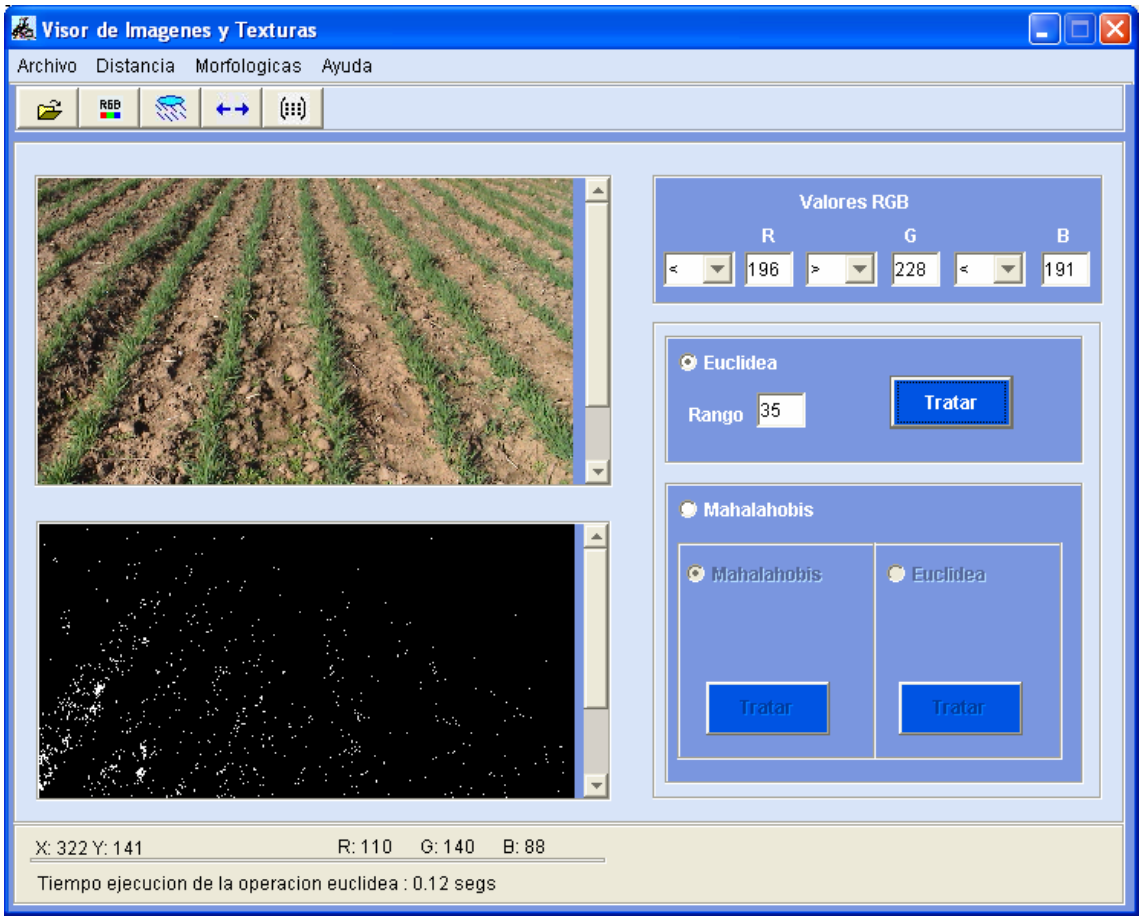
- Aplicación de diversas distancias.

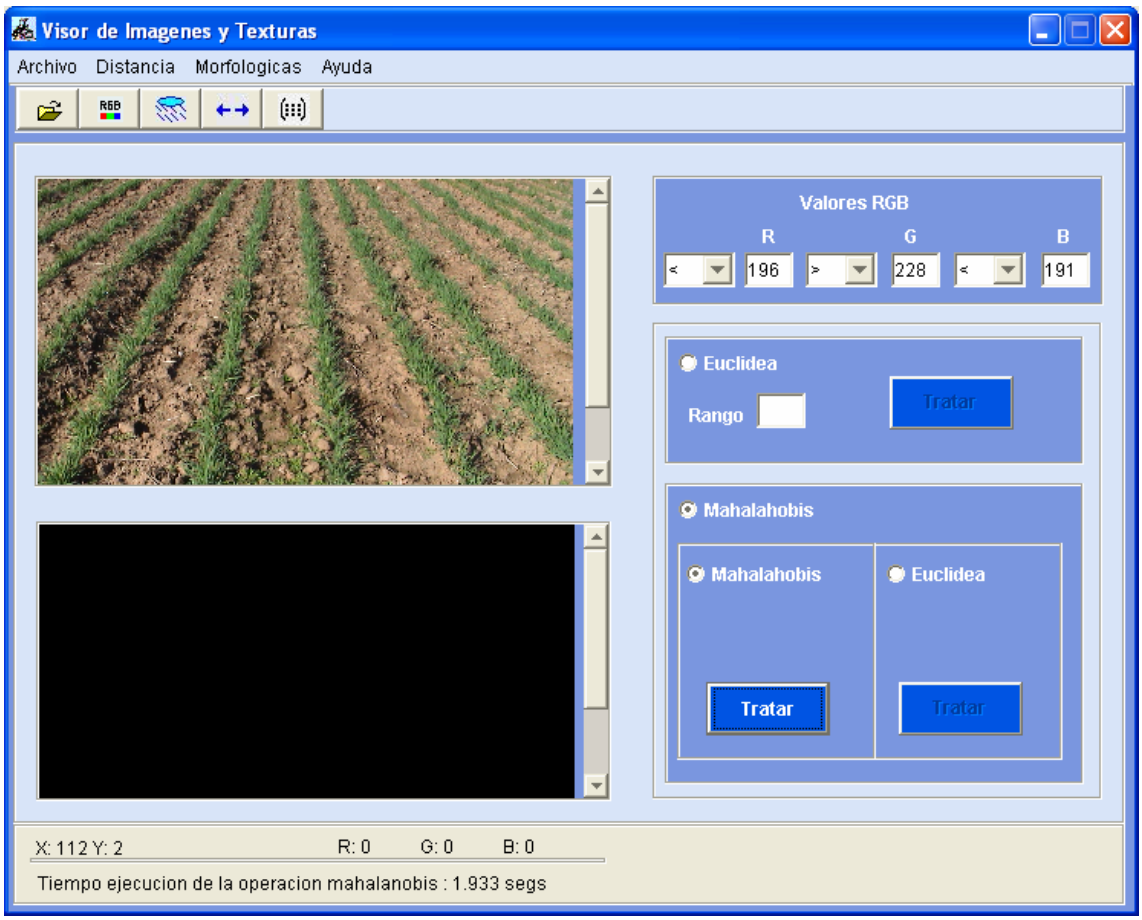
Resultados esperados:

- Binarización de imagen mala, casi toda la imagen negra.
- Observación de dicha binarización.
- Tiempo de tratamiento normal

Resultados observados:

- Binarización de baja calidad no útil para su procesamiento en caso de la distancia euclidea
- En caso de la distancia de Mahalanobis (en cualquiera de sus dos versiones) no se consigue ver nada en la imagen binarizada.
- Tiempo de procesamiento bajo debido a la gran cantidad de pixeles excluidos para el estudio en caso de la distancia euclidea.
- El tiempo de procesamiento es alto en caso de la distancia de Mahalanobis.





7.-Aplicación de erosion

Descripción:

- Aplicar erosión a la imagen binarizada con el fin de obtener una cierta eliminación del ruido, aplicando la matriz por defecto.

Pasos seguidos:

- Carga de la imagen.
- Selección del rango RGB:

	Red	Green	Blue
<	75		75
>		99	
=			

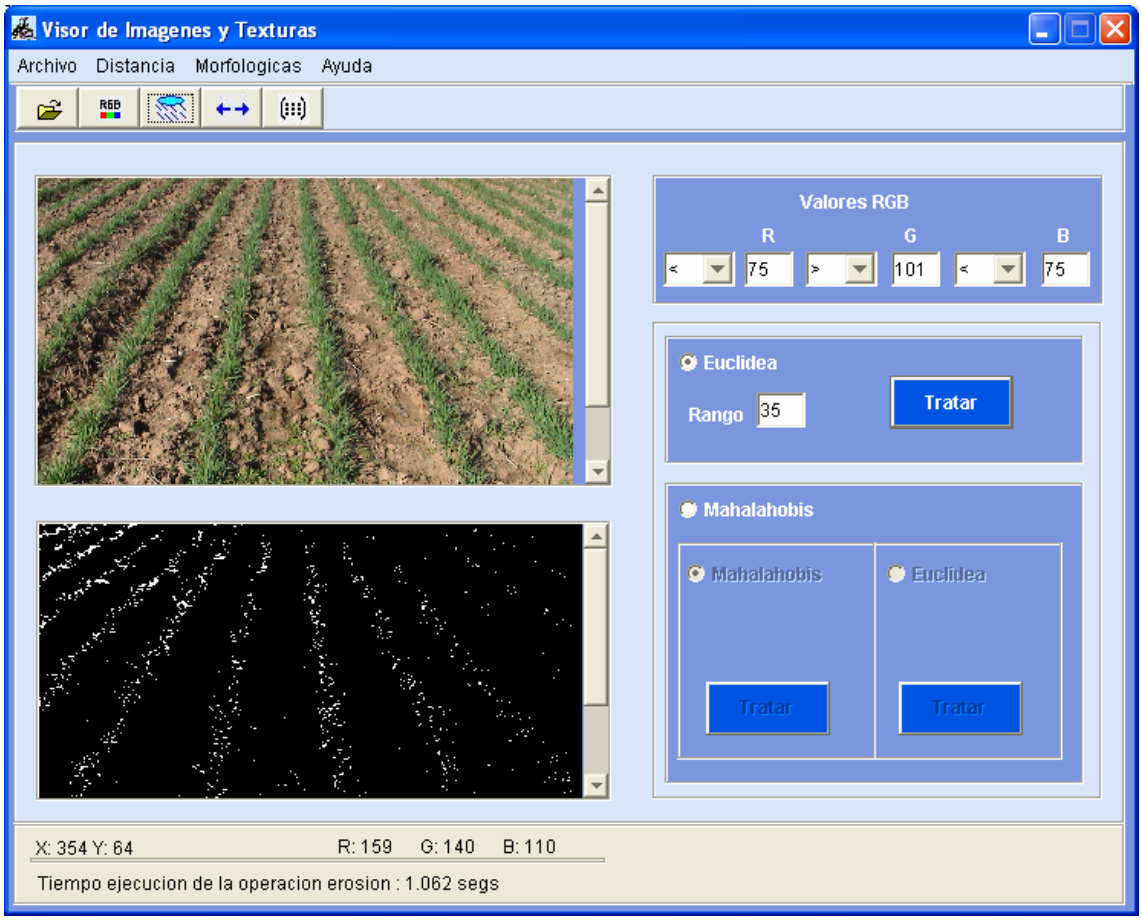
- Pulsar botón distancia euclidea e introducir un rango de 35.
- Pulsar el boton erosión.

Resultado esperados:

- La previsualización de una imagen binarizada conforme a la anterior pero con la presencia de menos ruido
- Observación del tiempo de procesamiento.

Resultados observados:

- Se produce una visualización de la imagen con la desaparición de 8una gran cantidad de puntos blancos que representan a los valores correctos de verde obteniendo así la eliminación de ruidos.
- El valor de procesamiento observado es de 1,061 seg.



8.-Aplicación de dilatación

Descripción:

- Aplicar dilatación a la imagen binarizada con el fin de obtener una agrupación de puntos blancos que representan el color verde.

Pasos seguidos:

- Carga de la imagen
- Selección de rango RGB:

	Red	Green	Blue
<	75		75
>		99	
=			

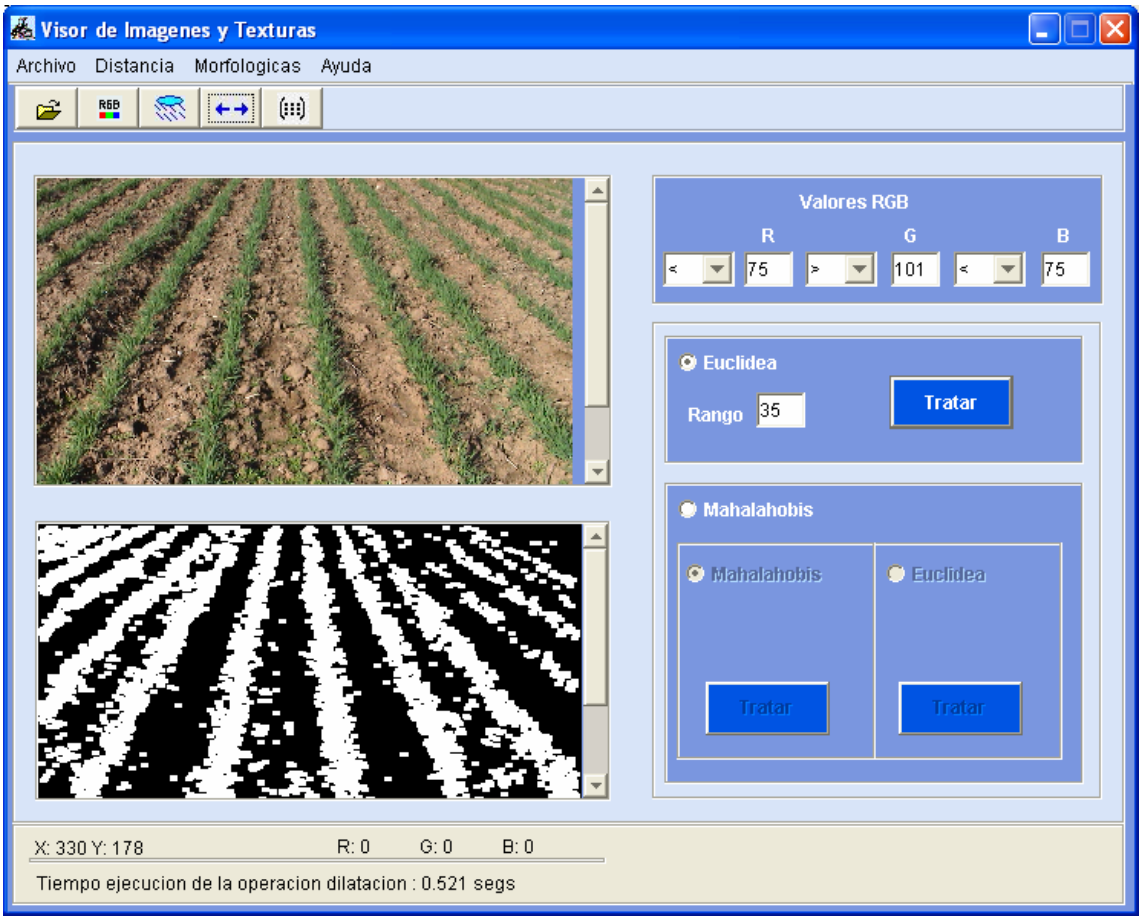
- Pulsar el boton “euclidea”.
- Introducir un rango de 35.
- Pulsar el boton dilatación.

Resultados esperados:

- Obtención de una imagen con una mayor cantidad de puntos blancos, sin la presencia de valores de negro entre ellos.
- Observación de la imagen tratada.
- Observación del tiempo de procesamiento.

Resultados observados:

- Imagen con una gran cantidad de puntos blancos debido a la dilatación ofrecida, se produce una agrupación de puntos blancos.
- Un tiempo de procesamiento de 0,521 seg.



9.-Aplicación de una erosión, seguida de una dilatación

Descripcion:

- Aplicación de estas dos operaciones morfológicas sobre la imagen binarizada para obtener así lo surcos correspondientes.

Pasos seguidos:

- Carga de la imagen
- Introduccion de los valores de RGB:

	Red	Green	Blue
<	75		75
>		99	
=			

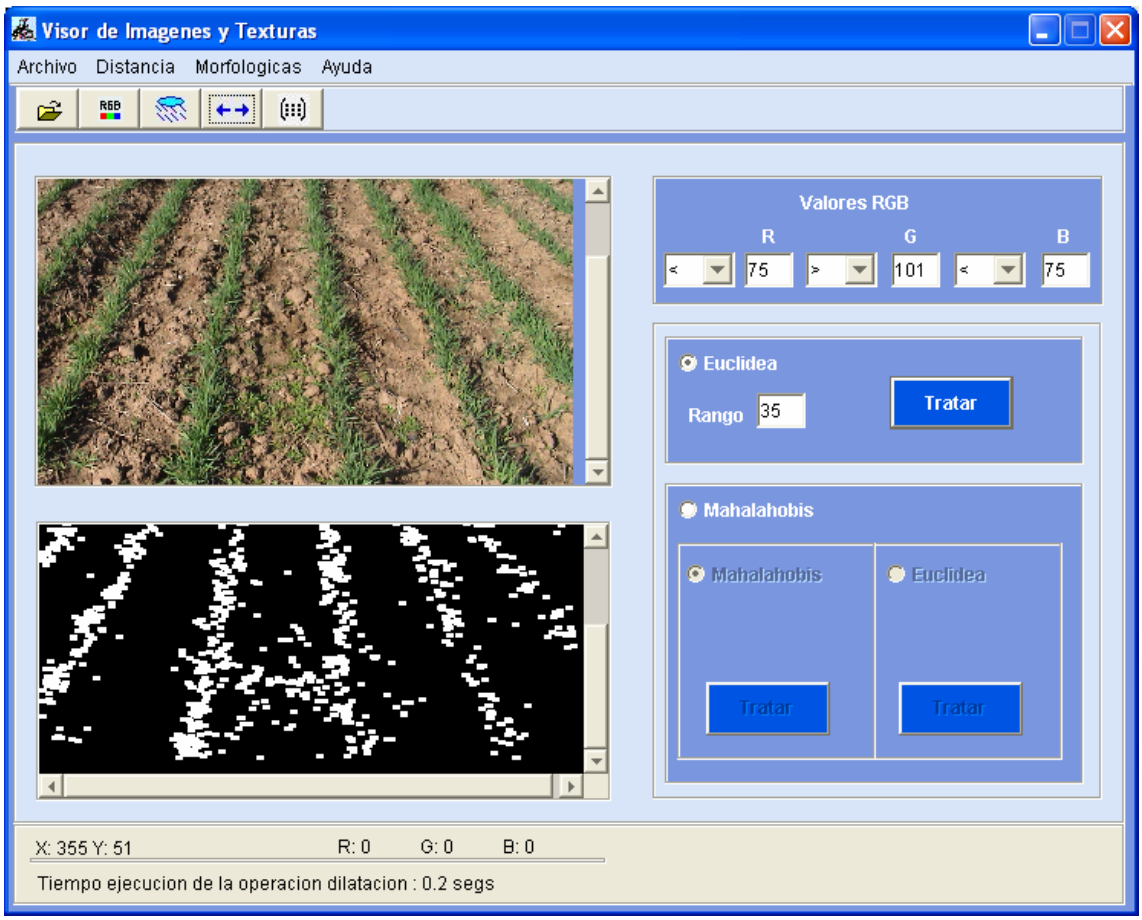
- Pulsar el boton “euclidea”
- Introducir como valor del rango 35.
- Pulsar “Tratar”
- Pulsar el boton “erosión”
- Pulsar el boton “dilatacion”

Resultado esperados:

- Obtencion de una imagen binarizada con una eliminación casi completa de ruidos
- Previsualización de la imagen
- Observacion del tiempo de tratamiento.

Resultados observados:

- Imagen binarizada en la que aparece ruido debido a la existencia de malas hierbas.
- Observación de un tiempo de porcesamiento de 0,35 seg. durante la erosión y de 0,2 seg. durante la dilatación, es decir un tiempo total de 0,55 en el tratamiento dela imagen binarizada.



10.-Aplicación de una dilatación, seguida de una erosion

Descripcion:

- Aplicación de estas dos operaciones morfologicas sobre la imagen binarizada para obtener asi lo surcos correspondientes.

Pasos seguidos:

- Carga de la imagen
- Introducción de los valores de RGB:

	Red	Green	Blue
<	75		75
>		99	
=			

- Pulsar el boton “euclidea”
- Introducir como valor del rango 35.
- Pulsar “Tratar”
- Pulsar el boton “dilatacion”
- Pulsar el boton “erosión”

Resultado esperados:

- Obtención de una imagen binarizada con una eliminación casi completa de ruidos
- Previsualización de la imagen
- Observación del tiempo de tratamiento.

Resultados observados:

- Imagen binarizada en la que aparece ruido debido a la existencia de malas hierbas, pero además debido a que la eliminación de los ruidos intrínsecos a la binarización no han sido eliminados por completo. Se puede decir que el metodo de erosión y posteriormente dilatación actua de mejor manera.
- Observación de un tiempo de procesamiento de 0,46 seg. durante la dilatación y de 0,531 seg. durante la erosión, es decir un tiempo total de 0,96 en el tratamiento de la imagen binarizada.

11.-Aplicación de erosión con una matriz distinta a la proporcionada por defecto

Descripción:

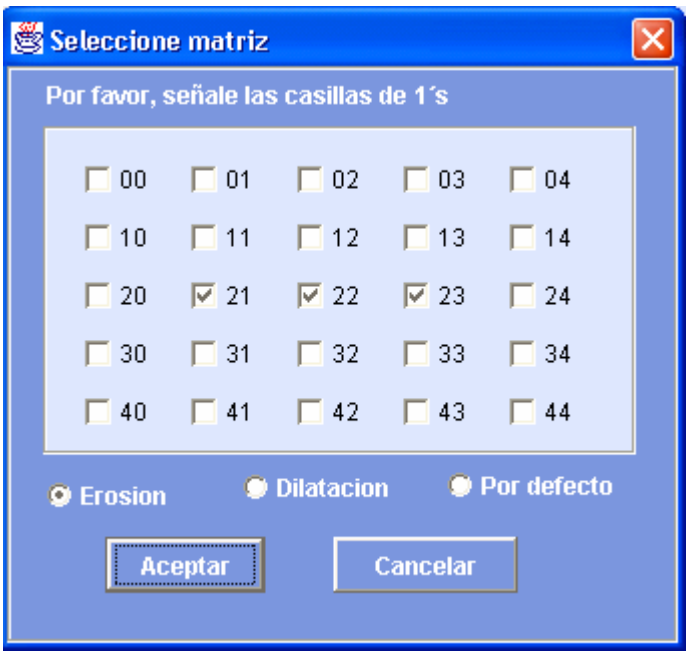
- Aplicar erosión a la imagen binarizada con el fin de obtener una cierta eliminación del ruido, aplicando una matriz diferente a la proporcionada por defecto.

Pasos seguidos:

- Carga de la imagen.
- Selección del rango RGB:

	Red	Green	Blue
<	75		75
>		99	
=			

- Pulsar botón distancia euclidea e introducir un rango de 35.
- Pulsar botón 



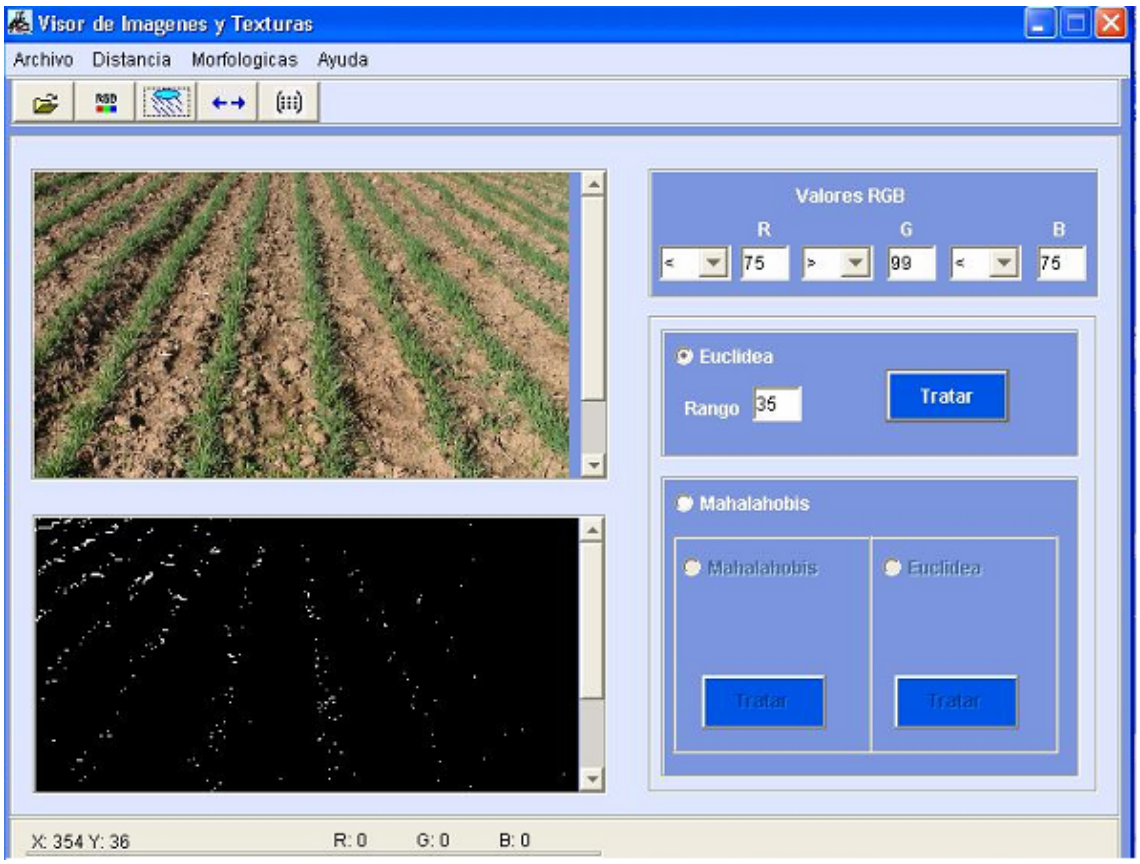
- Escoger la opción “Erosión”
- Pulsar las casillas de la matriz de erosión que se desea que estén a 1
- Aceptar
- Pulsar el botón erosión.

Resultados esperados:

- La previsualización de una imagen binarizada conforme a la anterior pero con la presencia de menos ruido
- La binarización contendrá un mayor o menor número de puntos blancos dependiendo de la matriz elegida para realizar la erosión

Resultados observados:

- Se produce una visualización de la imagen con la desaparición de una gran cantidad de puntos blancos que representan a los valores correctos de verde obteniendo así la eliminación de ruidos.



12.-Aplicación de dilatación con una matriz distinta a la proporcionada por defecto

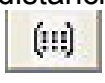
Descripción:

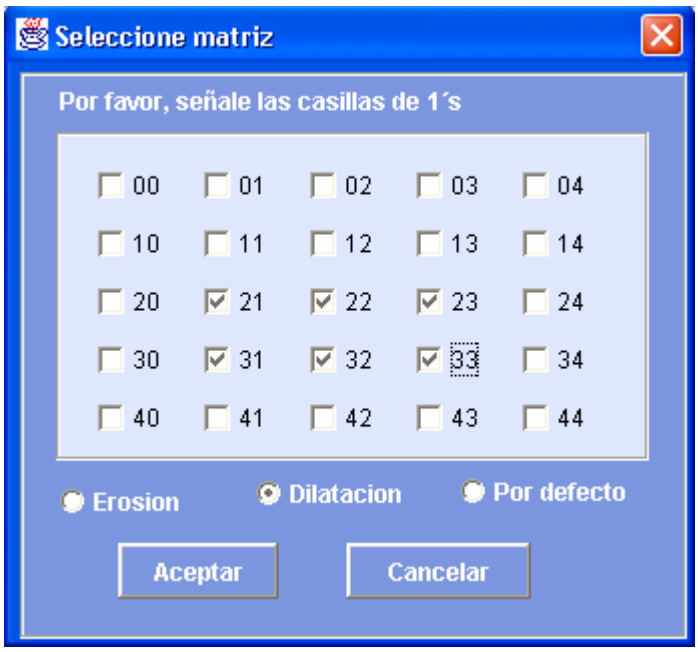
- Aplicar dilatación a la imagen binarizada con el fin de obtener una cierta eliminación del ruido, aplicando una matriz diferente a la proporcionada por defecto.

Pasos seguidos:

- Carga de la imagen.
- Selección del rango RGB:

	Red	Green	Blue
<	75		75
>		99	
=			

- Pulsar botón distancia euclidea e introducir un rango de 35.
- Pulsar botón 



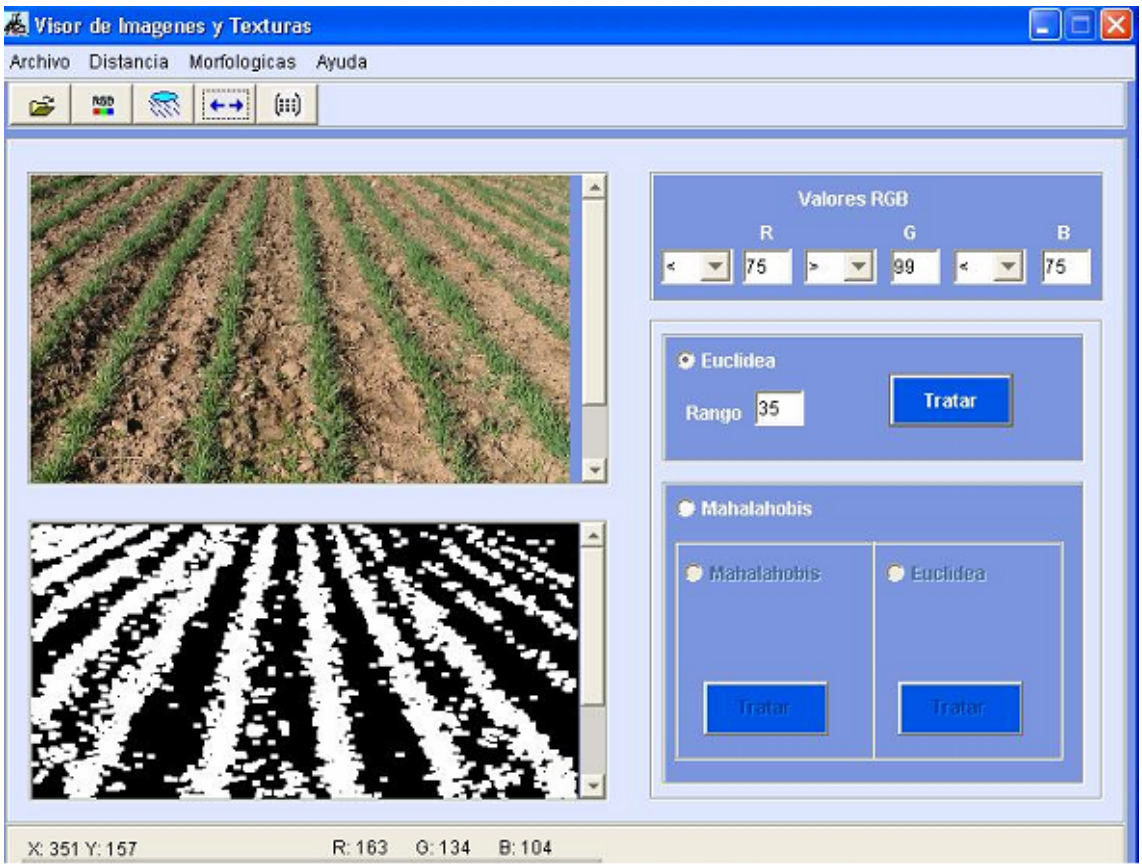
- Escoger la opción “Dilatación”
- Pulsar las casillas de la matriz de erosión que se desea que estén a 1
- Aceptar
- Pulsar el botón dilatación.

Resultados esperados:

- Obtención de una imagen con una mayor cantidad de puntos blancos, sin la presencia de valores de negro entre ellos.
- La binarización contendrá un mayor o menor número de puntos blancos dependiendo de la matriz elegida para realizar la dilatación

Resultados observados:

- Imagen con una gran cantidad de puntos blancos debido a la dilatación ofrecida, se produce una agrupación de puntos blancos.



13.- Comprobación de que los valores RGB mostrados se adaptan a lo esperado

Descripción:

- Comprobar que los valores RGB mostrados en la parte inferior de la ventana van acorde con lo esperado en función de donde se encuentra situado el ratón.

Pasos seguidos:

- Carga de la imagen.
- Selección del rango RGB:

	Red	Green	Blue
<	75		75
>		99	
=			

- Pulsar botón distancia euclidea e introducir un rango de 35.
- Pulsar “Tratar”
- Pasar el ratón sobre la imagen original y comprobar como cambian los valores RGB mostrados en la parte inferior de la ventana.

Resultados esperados:

- Visualización de unos valores RGB que concuerden con el color de la zona sobre la que se encuentra situado el ratón

Resultados observados:

- Los valores RGB que se muestran van variando según nos movemos sobre la imagen, y nos van mostrando valores cuya componente de verde es muy alta cuando nos situamos sobre los surcos, y baja cuando salimos de ellos.

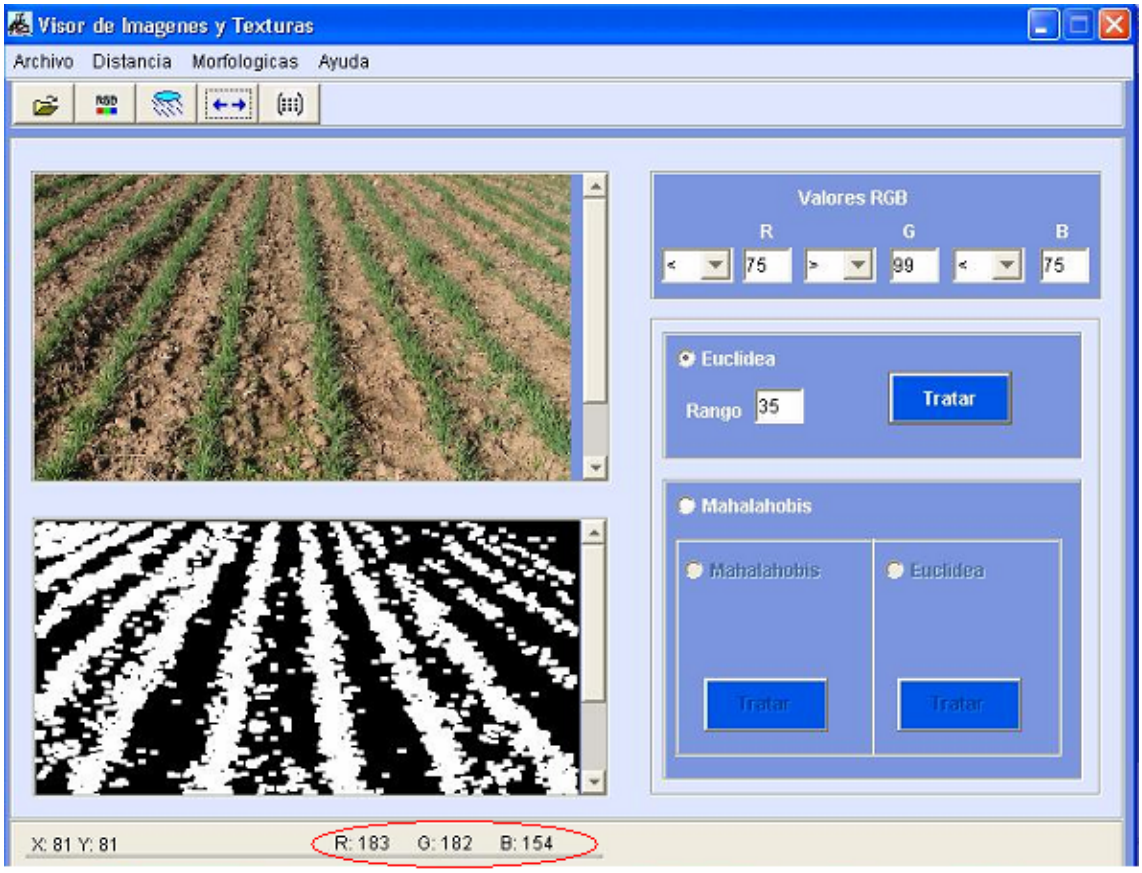


Tabla resumen del resultado de las pruebas:

	Resultado		
Nombre Prueba	**Bien**	**Mal**	**Regular**
Carga de una imagen	✔		
Selección de un rango de color	✔		
Aplicación de la distancia euclídea	✔		
Aplicación de la distancia Mahalanobis-Euclídea	✔		
Aplicación de la distancia Mahalanobis	✔		
Aplicación de valores de RGB que excluyan muchos píxeles	✔		
Aplicación de erosión	✔		
Aplicación de dilatación	✔		
Aplicación de una erosión seguida de una dilatación	✔		
Aplicación de una dilatación seguida de una erosión	✔		
Aplicación de erosión con una matriz distinta a la proporcionada por defecto	✔		
Aplicación de dilatación con una matriz distinta a la proporcionada por defecto	✔		
Comprobación de que los valores RGB mostrados se adaptan a lo esperado	✔		

8. Conclusión

El trabajo realizado en la asignatura Sistemas Informáticos nos ha permitido desarrollar un proyecto Software como preparación para la formación profesional a nivel empresarial.

Se trata de un proyecto real planteado bajo la perspectiva de una amplia investigación en el campo agrícola por parte del grupo de Inteligencia Artificial en el Instituto de Automática del CSIC. Es un proyecto de investigación real cuyas directrices emanan del propio grupo de investigación del CSIC. En este sentido cabe señalar que desde nuestra perspectiva éste sería nuestro cliente, tal y como se entiende en el mundo empresarial.

Así pues, a partir de las especificaciones proporcionadas por el CSIC, se establece un plan de trabajo que contempla todas las fases del *ciclo de vida* de desarrollo de cualquier proyecto software. El mencionado plan de trabajo ha sido una tarea que nos propusimos inicialmente contando con los plazos previstos de desarrollo del proyecto, en este caso, también imitando las pautas de los desarrollos Software.

En el marco de las especificaciones se contempla la necesidad de desarrollar técnicas de procesamiento de imágenes bajo la cobertura de un interfaz de usuario. Gracias a lo cual, este proyecto nos ha permitido documentarnos a la vez que buscar recursos a través de Internet, como será menester hacer en el futuro de cara al desarrollo de la actividad profesional en la empresa. En concreto, hemos tenido que estudiar nuevas tecnologías, destacando por su importancia la librería JAI (JAVA Advanced Imaging), tras cuya valoración ha sido necesaria su integración.

Dado el carácter de continuidad del proyecto, hemos previsto un diseño abierto y modular con el fin de incorporar e integrar nuevas funcionalidades en el futuro y como consecuencia de la continuidad del proyecto. Esto implica necesariamente la consideración de un proyecto que ha de abordarse desde la perspectiva de los desarrollos empresariales donde los equipos humanos pueden cambiar, pero los desarrollos son de propiedad empresarial.

9. ANEXO A

A continuación se muestran los diagramas de clases asociados a la aplicación:

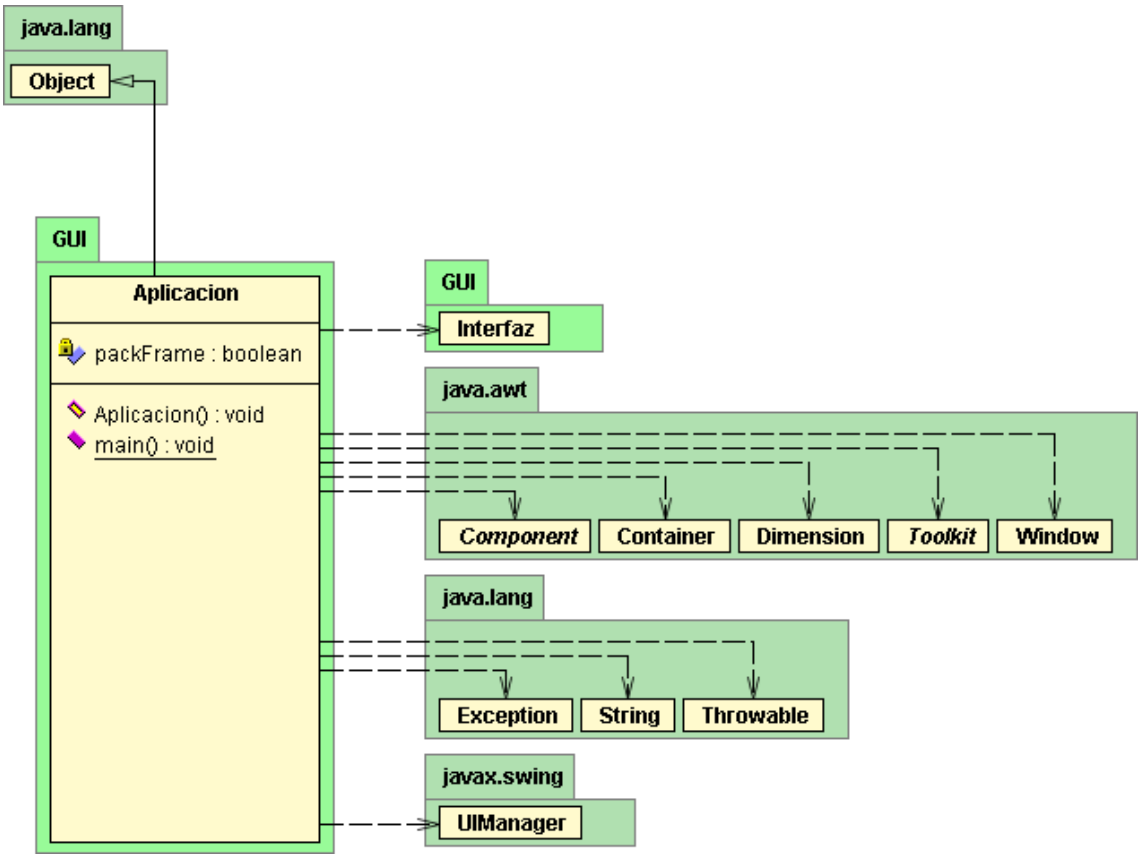


Figura 1 Diseño de la aplicación

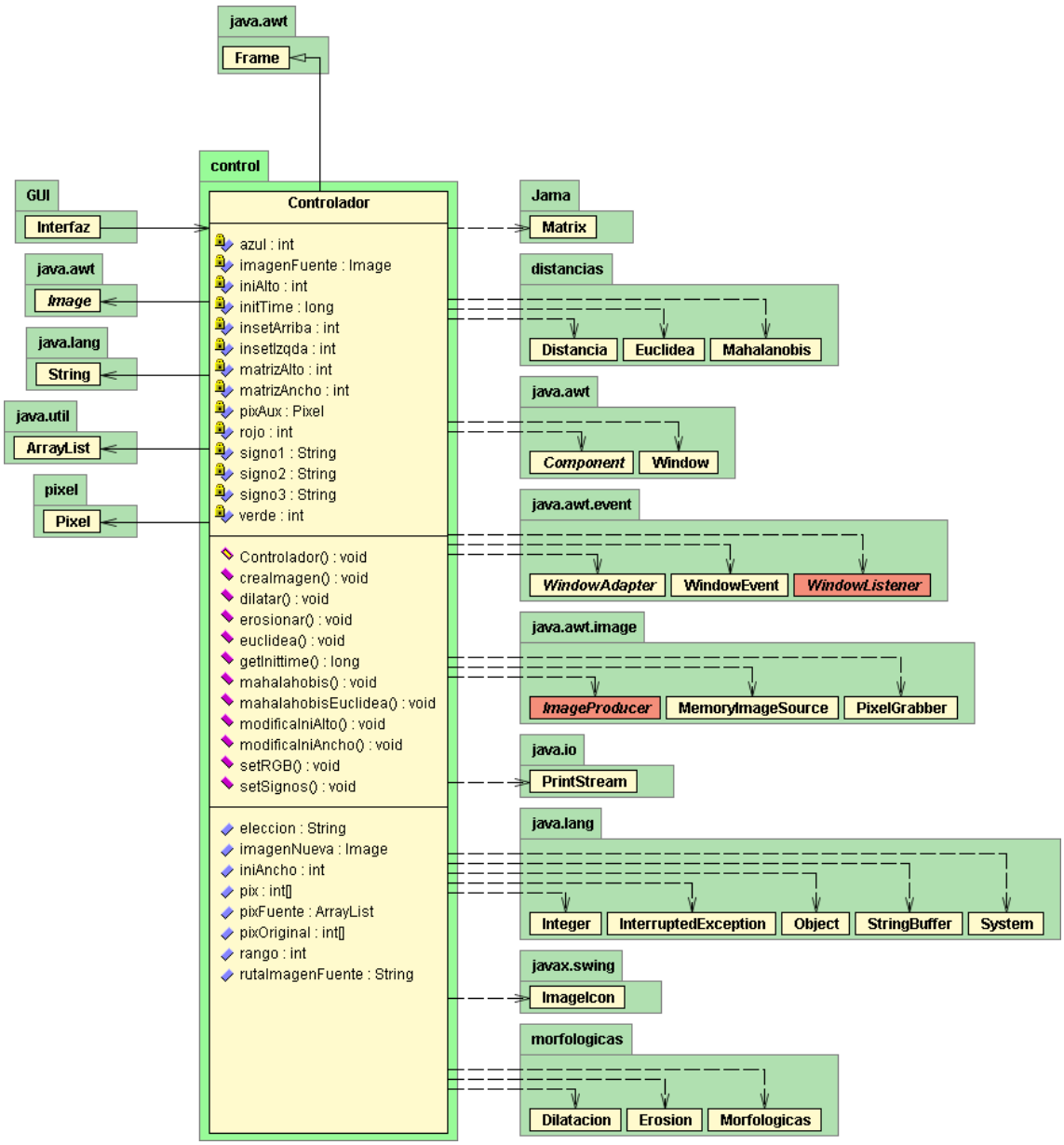


Figura 2 Diseño del Controlador

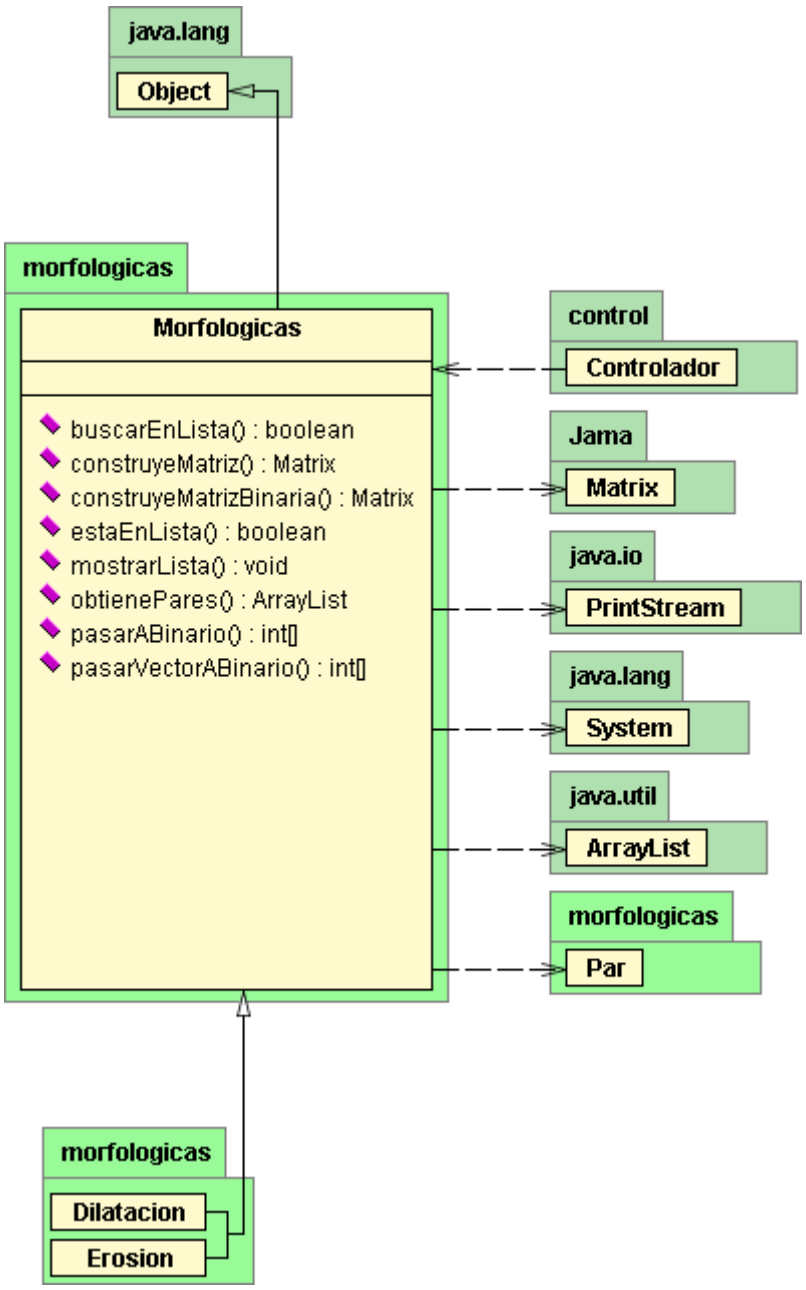


Figura 3 Paquete Morfologicas

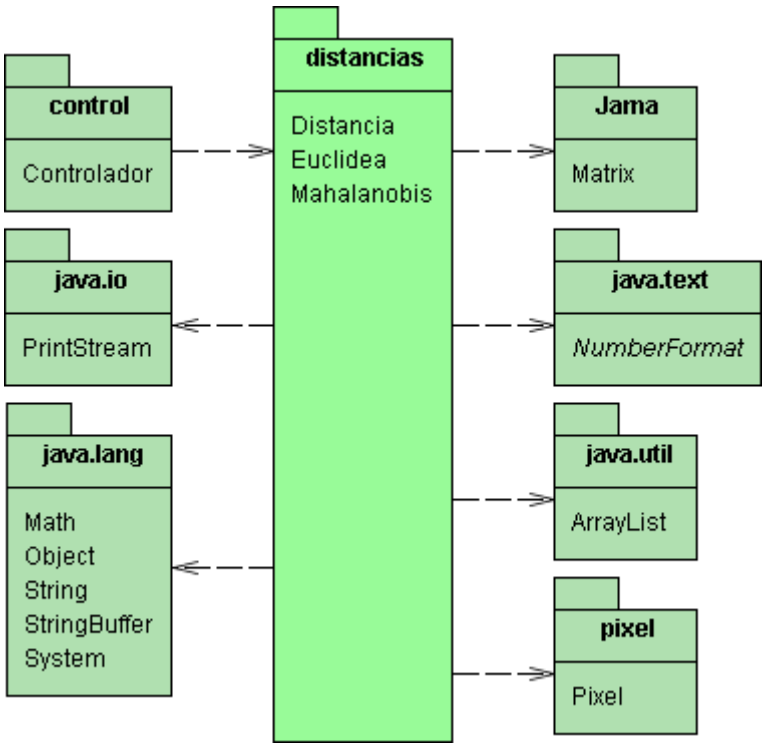


Figura 4 Paquete Distancias

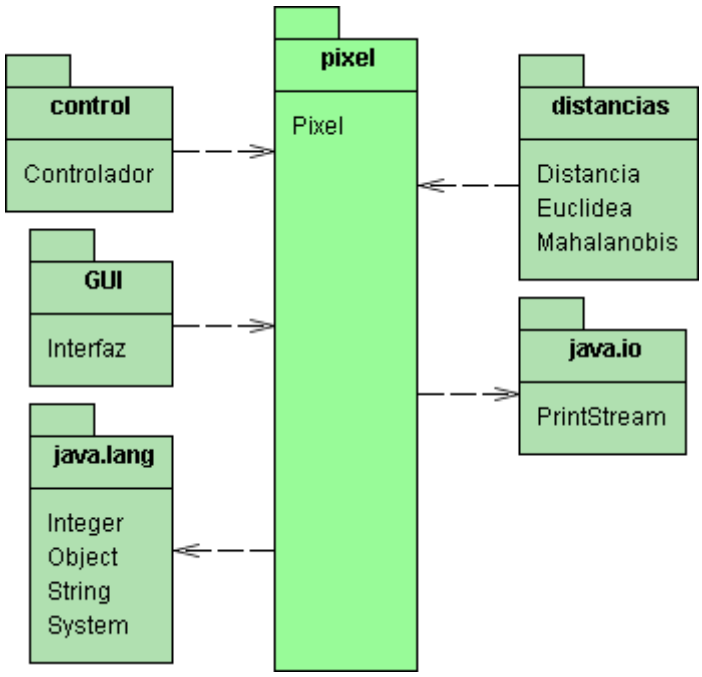


Figura 5 Paquete Pixel

10. ANEXO B

Clasificador de mínima distancia Euclídea

Si $\delta_E(A, B)$ es la distancia euclídea entre *A* y *B*, entonces:

$$\delta_E^2(A, B) = \|A\text{-}B\|^2 = (A\text{-}B)^T(A - B)$$

Supongamos que *A* y *B* son dos puntos del espacio definido por las variables X_1 y X_2. Su expresión vectorial sería:

$$A = \begin{bmatrix} A_{X_1} \\ A_{X_2} \end{bmatrix} \qquad B = \begin{bmatrix} B_{X_1} \\ B_{X_2} \end{bmatrix}$$

Los vectores $(A - B)$ y $(A\text{-}B)^T$ serán, por lo tanto,

$$(A - B) = \begin{bmatrix} A_{X_1} - B_{X_1} \\ A_{X_2} - B_{X_2} \end{bmatrix} \qquad (A\text{-}B)^T = \begin{bmatrix} A_{X1} - B_{X1}, A_{X2} - B_{X2} \end{bmatrix}$$

Así, $(A\text{-}B)^T(A - B)$ es un valor escalar, de expresión:

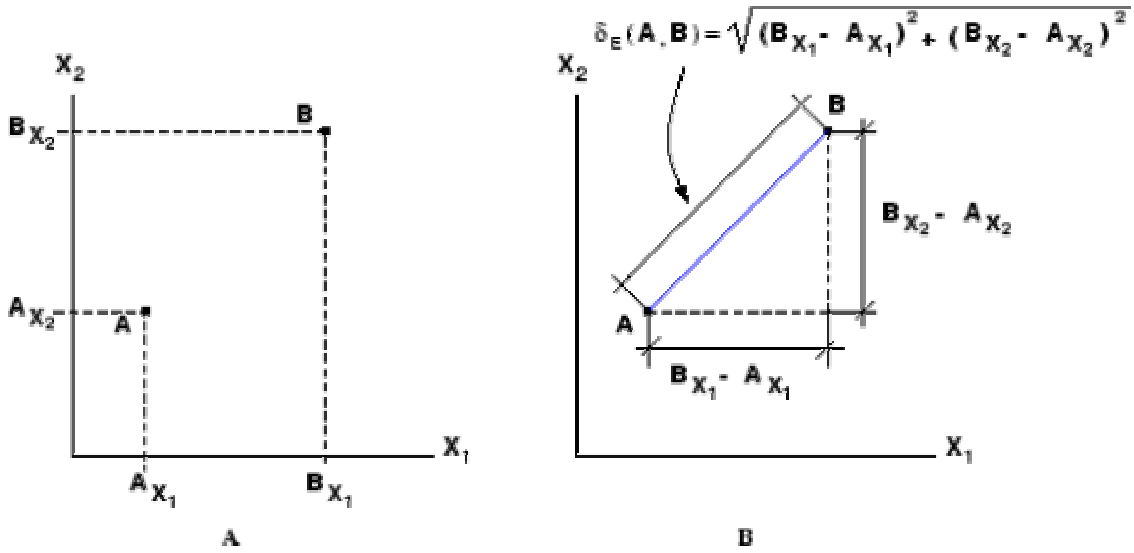
$$(A\text{-}B)^T(A - B) = (A_{X1} - B_{X1})(A_{X1} - B_{X1}) + (A_{X2} - B_{X2})(A_{X2} - B_{X2})$$

Por otra parte, si consideramos que

$$\delta_E(A, B) = \sqrt{(A_{X_1} - B_{X_1})^2 + (A_{X_2} - B_{X_2})^2}$$

resulta evidente que

$$\delta_E^2(A, B) = (A_{X1}\text{-}B_{X1})^2 + (A_{X2}\text{-}B_{X2})^2 = (A\text{-}B)^T(A - B)$$



Cálculo de la distancia Euclídea

En este caso, la regla de decisión óptima, que da lugar al ***clasificador de mínima distancia Euclídea*** se formula de esta manera:

Si $\delta_E^2(X, \mu_i)$ *es la distancia Euclídea (al cuadrado) de X al centro de la clase i,*

$$d(X) = \omega_c \quad \text{si} \quad \delta_E^2(X, \mu_c) = \min_{i=1,2,\ldots,J} \delta_E^2(X, \mu_i)$$

donde $\delta_E^2(X, \mu_i) = \| X - \mu_i \|^2 = (X - \mu_i)^T (X - \mu_i)$

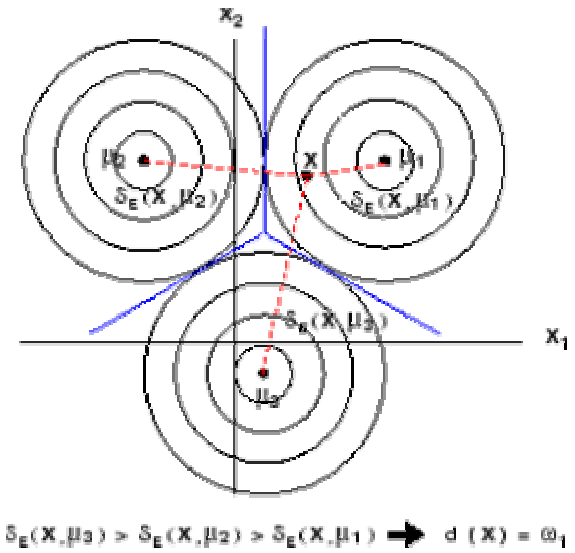
Observar que estamos ``resumiendo'' una clase por su valor medio: toda la información de interés de una clase (para la clasificación) está concentrada en su media. Así, suele hablarse de *la distancia de X a la clase* ω_i en lugar de la expresión *la distancia de X al centro de la clase* ω_i, más correcta desde un punto de vista formal.

Ejemplo

En la figura mostramos cómo se efectúa la clasificación de un patrón *X* en un problema con tres clases suponiendo un espacio Euclídeo. Las líneas azules indican las fronteras de decisión. En este ejemplo se observa claramente que *d*

$$\delta_E^2(X, \mu_1) < \delta_E^2(X, \mu_2) < \delta_E^2(X, \mu_3)$$

(*X*) = ω_1 ya que



Un clasificador Euclídeo para tres clases

Para simplificar los cálculos, vamos a expresar de manera diferente este clasificador. En primer lugar, resulta evidente que

$$\delta_E^2(X,\mu_i) = (X-\mu_i)^T(X-\mu_i) = X^TX - 2\,\mu_i^T X + \mu_i^T\mu_i$$

Como el objetivo es encontrar la clase para la que $\delta_E^2(X,\mu_i)$ sea mínima, podemos descartar el término X^TX al ser constante (igual para toda *i*), Así,

$$\min_{i=1,2,\ldots,J} \delta_E^2(X,\mu_i) = \min_{i=1,2,\ldots,J}\{-2\,\mu_i^T X + \mu_i^T\mu_i\} = \max_{i=1,2,\ldots,J}\{\mu_i^T X - \tfrac{1}{2}\,\mu_i^T\mu_i\}$$

que expresado en forma de funciones discriminantes,

$$g_i(X) = \mu_i^T X - \tfrac{1}{2}\,\mu_i^T\mu_i = W_i^TX + w_{i0}$$

donde:

$$W_i = \mu_i$$

$$w_{i0} = \tfrac{1}{2}\,\mu_i^T\mu_i$$

Finalmente, el resultado obtenido en la ecuación puede escribirse de una manera aún más compacta como el producto de dos vectores:

$$g_i(X) = W_i^T X$$

donde:

$$W_i^T = [\ \mu_{i_1}, \mu_{i_2}, \ldots, \mu_{i_d}, -\tfrac{1}{2}\ \mu_i^T\ \mu_i\ ]$$
$$X^T = [X_1, X_2, \ldots, X_d, 1\ ]$$

Los vectores W_i y X no son más que los vectores μ_i y X a los que se les ha añadido una nueva componente. En el caso de W_i se supone que el valor $-\frac{1}{2}\ \mu_i^T \mu_i$ se ha calculado previamente a su inclusión. Como resultado obtenemos las siguientes funciones:

$$g_i(X) = W_i^T X = \left[\ \mu_{i_1}, \mu_{i_2}, \ldots, \mu_{i_d}, -\tfrac{1}{2}\ \mu_i^T\ \mu_i\ \right] \begin{bmatrix} X_1 \\ X_2 \\ \cdots \\ X_d \\ 1 \end{bmatrix} =$$

$$= \underbrace{\mu_{i_1} X_1 + \mu_{i_2} X_2 + \ldots + \mu_{i_d} X_d}_{\mu_i^T X} - \tfrac{1}{2}\ \mu_i^T\ \mu_i$$

Clasificador de mínima distancia de Mahalanobis

La *distancia de Mahalanobis (al cuadrado)* de X a μ_i, a la que hemos llamado $\delta_M^2(X, \mu_i)$, se expresa como:

$$\delta_M^2(X, \mu_i) = (X - \mu_i)^T\ \Sigma^{-1} (X - \mu_i)$$

Esta distancia se emplea en situaciones en las que las matrices de covarianza son iguales (en la práctica, basta con ser similares) y si además, el espacio de representación no es Euclídeo al estar las variables incorreladas ser sus varianzas diferentes. En la expresión de la distancia de Mahalanobis interviene la inversa de la matriz de covarianza para tener en cuenta la distinta dispersión de las variables en el espacio.

En este caso, la regla de decisión óptima, que da lugar al ***clasificador de mínima distancia de Mahalanobis*** se formula de esta manera:

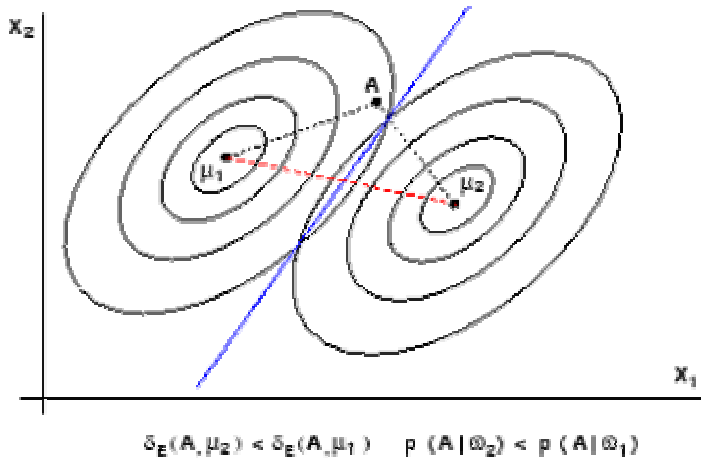
Si $\delta_M^2(X, \mu_i)$ es la distancia de Mahalanobis (al cuadrado) de X al centro de la clase i,

$$d(X) = \omega_c \quad \text{si} \quad \delta_M^2(X, \mu_c) = \min_{i=1,2,\ldots,J} \delta_M^2(X, \mu_i)$$

donde $\delta_M^2(X, \mu_i) = (X - \mu_i)^T \Sigma^{-1} (X - \mu_i)$

Ejemplo

En la figura mostramos un claro ejemplo de un problema del caso 2 (dos clases con iguales matrices de covarianza y en el que las variables están correladas). Suponemos que $\pi_1 = \pi_2$.



Distancia de Mahalanobis frente a distancia Euclídea

Si la clasificación del punto *A* se hiciera en base al clasificador de mínima distancia Euclídea se etiquetaría como de clase 2 ya que como puede verse, $\delta_E^2(A, \mu_2) < \delta_E^2(A, \mu_2)$ Sin embargo, de la representación gráfica puede verse fácilmente que *A* está en la región de decisión de la clase 1 por lo que debería etiquetarse de clase 1. Veamos porqué.

Recordemos que las elipses indican los puntos con igual densidad de probabilidad. Si consideramos la elipse más externa asociada a las dos clases, *A* está dentro de la elipse asociada a la clase 1, mientras que está fuera de la misma elipse asociada a la clase 2. En términos de probabilidades esto significa que $P(A|\omega_1) > P(A|\omega_2)$ por lo que una clasificación óptima asignará el punto *A* a la clase 1. En la figura mostramos (con trazos punteados) dos nuevas elipses (una por clase) que corresponden a los puntos para los que su densidad de probabilidad es igual a $P(A|\omega_1)$.

Todos estos puntos *X* verifican que $\delta_M^2(A, \mu_1) = \delta_M^2(X, \mu_1) = \delta_M^2(X, \mu_2)$

En este ejemplo puede verse claramente que $\delta_M^2(A, \mu_1) < \delta_M^2(A, \mu_2)$

11. BIBLIOGRAFÍA

REF 1: VISION PO COMPUTADOR: IMÁGENES DIGITALES Y APLICACIONES

PAJARES, GONZALO; DE LA CRUZ, JESÚS;

REF2: IMÁGENES DIGITALES: PROCESAMIENTO PRACTICO CON JAVA

PAJARES, GONZALO; DE LA CRUZ, JESÚS; MOLINA, JOSÉ; CUADRADO JUAN Y LOPEZ ALEJANDRO

www-etsi2.ugr.es/depar/ccia/ rf/www/tema2_00-01_www/node5.html

java.sun.com/products/**java**-media/ **jai**

java.sun.com/**j2se**/1.4.2/ docs/**api/java**/

www.itapizaco.edu.mx/paginas/ JavaTut/froufe/parte15/cap15-10.html