FACULTAD DE INFORMÁTICA
UNIVERSIDAD POLITÉCNICA DE MADRID

UNIVERSIDAD POLITÉCNICA DE MADRID
FACULTAD DE INFORMÁTICA

TRABAJO FIN DE CARRERA

IJOBJECT: HERRAMIENTA PARA LA GENERACIÓN Y CARACTERIZACIÓN DE OBJETOS EN IMÁGENES DE SATÉLITE

AUTOR: ELENA RODRÍGUEZ ZAZO
TUTOR: CONSUELO GONZALO MARTÍN
FECHA DE PRESENTACIÓN: 11 JULIO 2013

ÍNDICE

ÍNDICE DE FIGURAS

AGRADECIMIENTOS

Me gustaría agradecer, en primer lugar, a mis padres por todos estos años de paciencia en los que siempre han estado ahí para animarme en los peores momentos cuando todo se hacía cuesta arriba y siempre han confiado en que lograría terminar esta carrera llena de obstáculos.

Mención especial para *El Quillo*, que sin su ayuda e implicación; sin sus ánimos y su compañía, hubiera sido muy difícil llevar a cabo este proyecto.

Y a Chelo por querer en su momento, dirigirme en esta etapa final.

Gracias, gracias y gracias.

1. INTRODUCCIÓN

La teledetección consiste en la adquisición de información en torno a un objeto sin estar en contacto físico con él. Se basa en la detección y registro de la energía reflejada o emitida, y el procesado, análisis y aplicación de esa información [Arqu 03]. El término se refiere de manera general al uso de tecnologías de sensores remotos para adquisición de imágenes [Wikia].

Las imágenes obtenidas por los satélites de teledetección ofrecen una perspectiva única de la Tierra, sus recursos y el impacto que sobre ella ejercen los seres humanos. La teledetección por satélite ha demostrado ser una fuente rentable de valiosa información para numerosas aplicaciones, entre las que cabe citar la planificación urbana, vigilancia del medio ambiente, gestión de cultivos, prospección petrolífera, exploración minera, desarrollo de mercados, localización de bienes raíces y muchas otras.

El valor práctico y la multiplicidad de aplicaciones de las imágenes continúan aumentando a medida que se lanzan nuevos satélites, que se suman a los que ya están en órbita. Al haber más satélites se dispone de imágenes en una cantidad creciente de resoluciones espectrales, frecuencias de paso y detalles espaciales.

Cabe preguntarse qué ventajas tiene el utilizar imágenes de satélite cuando existen muchas otras fuentes de datos geográficos, como fotografías aéreas, estudios sobre el terreno y mapas sobre papel. Para la mayoría de las aplicaciones, la respuesta más sencilla es que las imágenes de satélite son más rápidas, mejores y más baratas. La imagen del satélite es con frecuencia el medio más práctico para adquirir información geográfica aprovechable. Dada su naturaleza digital, las imágenes satélite se procesan, manipulan y realzan para extraer de ellas detalles e informaciones que otras fuentes no detectarían. Se pueden interpretar estas imágenes utilizando sistemas de tratamiento de imágenes que analizan y tipifican los rasgos del terreno basándose en el valor digital que proporciona determinadas gráficas, como la firma espectral, que se explicará más adelante [SRGIS].

El procesado de las imágenes basado en *píxeles* implica un coste computacional muy alto comparado al procesado de una estructura de datos llamada *segmentación*, que consiste en dividir una imagen digital en varias partes (objetos), con el fin de simplificar y/o cambiar la representación de una imagen en otra más fácil de analizar. La segmentación se usa tanto para localizar objetos (zonas con características comunes) como para encontrar los límites de éstos dentro de una imagen. Más precisamente, la segmentación de la imagen es el proceso de asignación de una etiqueta a cada *píxel* de la imagen de forma que los *píxeles* que compartan la misma etiqueta también tendrán ciertas características visuales similares. Cada uno de los *píxeles* de una región son similares en alguna característica, como el color, la intensidad o la textura. Regiones adyacentes son significativamente diferentes con respecto a la(s) misma(s) característica(s). Algunas de las aplicaciones prácticas de la segmentación de imágenes son en pruebas médicas (localización de tumores y otras patologías, cirugía guiada por ordenador, diagnóstico, medida de volúmenes de tejido…), sensor de huella digital, reconocimiento de caras e iris, sistemas de control de tráfico, visión por computador... [Wikie].

En esta investigación, se propone un estudio de localización de objetos en imágenes satélite, que es otra de las aplicaciones de la segmentación. Concretamente, el objetivo planteado ha sido el diseño y realización de una herramienta, llamada *IJObject*, basada en el algoritmo y metodología propuestos en el artículo “Caracterización multiescala de objetos como herramienta para la clasificación de imágenes de alta resolución espacial” [Gonz 12].

La memoria de este trabajo, constituye el proyecto fin de carrera de su autor, y se ha estructurado en siete capítulos, los cuales se detallan a continuación.

En el primer capítulo (Introducción) se expone de forma clara la motivación, objetivos y contenido del proyecto anteriormente citado.

En el segundo capítulo (Fundamentos) se enumeran y explican algunos de los conceptos básicos de la teledetección.

Por otro lado, en el tercer capítulo de este trabajo (Metodología), se describen los algoritmos de segmentación que se van a utilizar para este trabajo además de una serie de atributos que la herramienta va a calcular de los objetos obtenidos.

En el cuarto capítulo (Implementación de la herramienta), se detalla la metodología *software* llevada a cabo para el desarrollo de esta herramienta, además de los casos de prueba realizados para asegurar el correcto funcionamiento de la aplicación.

En el quinto capítulo (Resultados), se evalúan los resultados obtenidos tras la ejecución completa de la metodología propuesta.

En el sexto capítulo (Conclusiones), se dan a conocer las ventajas e inconvenientes de los algoritmos empleados y se propone una línea de trabajo futura en el área de la clasificación de objetos, identificando mediante distintos algoritmos qué es cada objeto detectado.

En el séptimo capítulo (Referencias), se incluye la bibliografía utilizada durante la realización de este trabajo.

A continuación, se incluye un apéndice donde se puede encontrar una guía de usuario para la utilización de la herramienta, instrucciones de cómo generar un *plugin* para *ImageJ*, y una explicación de la interfaz de los algoritmos descritos.

Al final del trabajo se incluye un anexo a modo de tutorial en el que se detallan paso a paso la ejecución de todos los tipos de algoritmos de clasificación de imágenes multiespectrales en el programa de procesamiento de imágenes *ENVI*. Resulta muy útil para determinar qué tipo de algoritmo es más apropiado para clasificar la imagen multiespectral de partida gracias a los resultados que se obtienen con respecto a qué *píxeles* forman cada objeto.

La herramienta se compone de ficheros *JAVA* que son exportados como un *plugin* para que pueda ser utilizado desde el programa de procesamiento de imágenes *ImageJ*, el cual es un programa de procesamiento digital de dominio público programado en *JAVA* y desarrollado en el *National Institutes of Health*, con amplia difusión en el ámbito del procesamiento de imágenes. *ImageJ* es un *framework* de desarrollo, cuya *API* facilita enormemente el desarrollo y testeo de los algoritmos que se implementan en *IJObject*.

2. FUNDAMENTOS

2.1. Imágenes de satélite y áreas de estudio

Las imágenes de satélite son ficheros ráster, formadas por una matriz regular o rejilla de celdas, a cada una de las cuales, denominada *píxel* (*Picture Element*, Elemento de Imagen, es la menor unidad homogénea en color que forma parte de una imagen digital, que es susceptible de ser procesado), se le asigna un *valor digital*, que corresponde al valor que traduce numéricamente la intensidad radiométrica recibida por un sensor óptico-electrónico [Insti]. La Figura 2.1 muestra un ejemplo de *píxeles* en una imagen monocroma captada a través de un sensor.

Figura 2.1: Ejemplo de *píxeles* en una imagen

- **Imagen Pancromática**

Las imágenes pancromáticas se captan mediante un sensor digital que mide la reflectancia de energía en una amplia parte del espectro electromagnético (con frecuencia, tales porciones del espectro reciben el nombre de bandas). Para los sensores pancromáticos más modernos, esta única banda suele abarcar lo parte visible y de infrarrojo cercano del espectro. Los datos pancromáticos se representan por medio de imágenes en blanco y negro [SRGIS].

En el desarrollo de este proyecto se va a utilizar para realizar las pruebas del programa, la imagen pancromática que aparece en la Figura 2.2. Entre sus

características cabe destacar que es una imagen *Quickbird* (es el satélite comercial con mayor resolución) registrada el día 18 de Febrero de 2005 en la región del Maule, Valle de Peumo, Chile (34º18'6''S; 71º19'11''O). El área registrada es de 9.43 hectáreas, correspondientes a 512 x 512 *píxeles*, está codificada en 16 bits, y su tamaño en disco es de 512 K. La zona de estudio corresponde a un área rural con plantaciones de naranjo y aguacate. No es una imagen grande por lo que facilita el tiempo de respuesta a la hora de hacer pruebas.

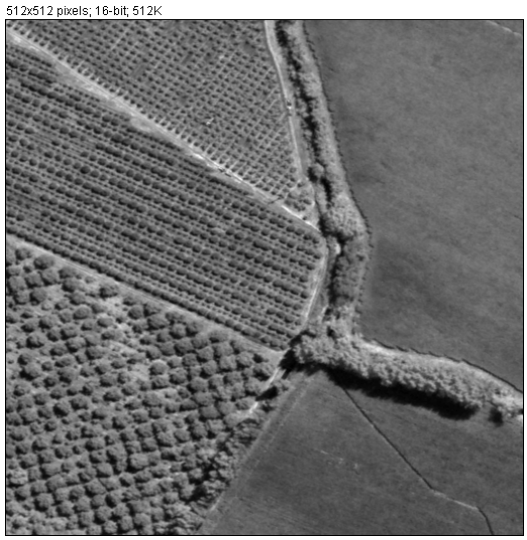


Figura 2.2: Imagen pancromática para el proyecto

- **Imagen Multiespectral**

Una imagen *multiespectral* o *multibanda* es un conjunto de imágenes, con las mismas propiedades geométricas, cada una de las cuales recoge el valor de reflectancia en un determinado intervalo de longitud de onda del espectro electromagnético [Insti]. Por tanto, un mismo *píxel* tendrá un valor digital diferente para cada banda.

Como imagen multibanda, para este proyecto se van a utilizar para realizar las pruebas del programa las 4 bandas que aparecen en la Figura 2.3 (la imagen multibanda es el resultado de la composición de estas 4 bandas). Esta imagen corresponde a la misma zona geográfica que la imagen pancromática de la Figura 2.2.

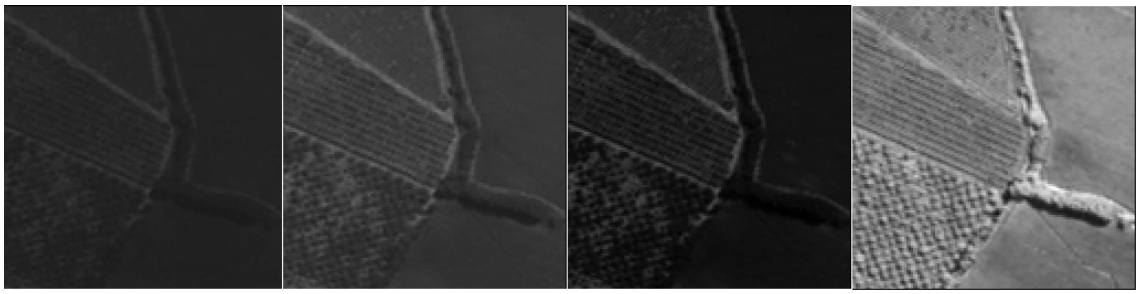


Figura 2.3: Bandas que componen la imagen multiespectral

- **Transformada *wavelet***

La transformada *wavelet* es una transformada multi-resolución que permite separar el detalle de las imágenes de su información de fondo para diferentes escalas, es decir, para las diferentes dimensiones espaciales a las que las entidades, patrones y procesos se pueden observar y medir.

La definición de los segmentos desde el punto de vista morfológico y textural se suele llevar a cabo a partir de la imagen de mayor resolución espacial, la imagen pancromática. Sin embargo, y con objeto de eliminar su información espectral, se propone realizar este proceso, no sobre la imagen directamente, sino sobre los coeficientes obtenidos mediante la transformada multi-resolución *wavelet*.

Una forma de pensar en los coeficientes *wavelet* es plantearse cómo miran los ojos de las personas. En el mundo real, se puede observar un bosque desde muchas perspectivas que son, de hecho, distintas escalas de resolución. Desde la ventana de un avión a reacción, por ejemplo, el bosque parece una cubierta sólida de verde. Desde la ventana de un automóvil que se encuentre sobre el suelo, la cubierta se transforma en árboles individuales; y saliendo del coche y acercándose, se comienzan a ver ramas y hojas. Tomando entonces una lupa, se podrá encontrar una gota de rocío en el extremo de una hoja. A medida que se acerca a escalas cada vez más pequeñas, se podrán encontrar detalles que no se habían observado antes. La transformada *wavelet* permite comprimir la cantidad de datos que se utilizan para almacenar una imagen, permitiendo almacenar una imagen más detallada en un espacio menor [Natio].

Las transformaciones *wavelet* permiten, en el ámbito del análisis multi-resolución, extraer el detalle espacial que se pierde al pasar de una resolución espacial a otra menor. La aproximación discreta de la transformada *wavelet* puede realizarse a partir de distintos algoritmos, como el de *Mallat* y *à trous* (con hoyos). Desde el punto de vista espacial, el algoritmo de *Mallat* presenta visualmente en las imágenes fusionadas, una menor calidad espacial que las obtenidas aplicando el algoritmo *à trous* [Gonz 04].

En este proyecto se van a utilizar los 3 coeficientes *wavelet* que muestra la Figura 2.4, obtenidos con el algoritmo *à trous* debido a que las conclusiones lo señalaban como el método más exacto y/o fiable. Están enumerados de mayor a menor escala, es decir, el coeficiente 1 es el que tiene más escala (con la ventaja de poder tener una visión a grandes rasgos de la imagen) mientras que el coeficiente 3 es el que menos escala tiene (y por tanto, más nivel de detalle).

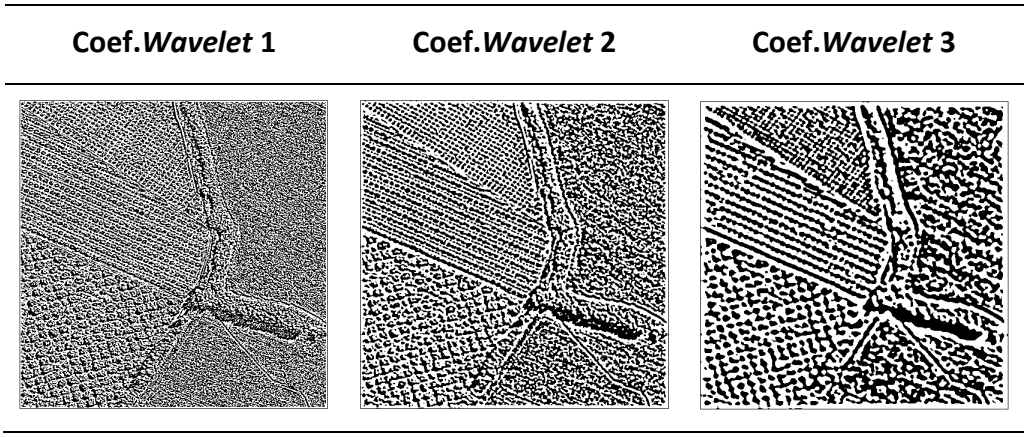


Figura 2.4: Coeficientes *wavelet* con diferentes escalas

2.2. Otros conceptos

A continuación se van a explicar unos conceptos relacionados con las imágenes satélite con el fin de ayudar a entender el resto del documento.

- **Firma espectral**

Cada *píxel* de una imagen multiespectral viene caracterizado espectralmente por un vector de características (firma espectral), que contiene el nivel digital asociado a ese *píxel* en cada una de las bandas de la imagen multibanda, según la resolución espectral [Arqu 03]. Depende de las características físicas o químicas del objeto que interaccionan con la energía electromagnética, y varía según las longitudes de onda [Insti]. Si se construye una gráfica en la que el eje de las *x* representa las bandas (con una longitud de onda determinada), y el eje de las *y* representa el valor digital asociado a cada *píxel* en esa banda, a esta gráfica se le conoce como *firma espectral*. La Figura 2.5 muestra un ejemplo de cuál es la reflectancia para cada una de las 4 cubiertas terrestres (nieve, suelo, vegetación y agua) para el espectro completo (que será un número discreto de bandas, el que determine el sensor), donde la vegetación posee una muy alta reflectividad en la banda del infrarrojo y el suelo mantiene valores constantes en todas las bandas [Wikib].

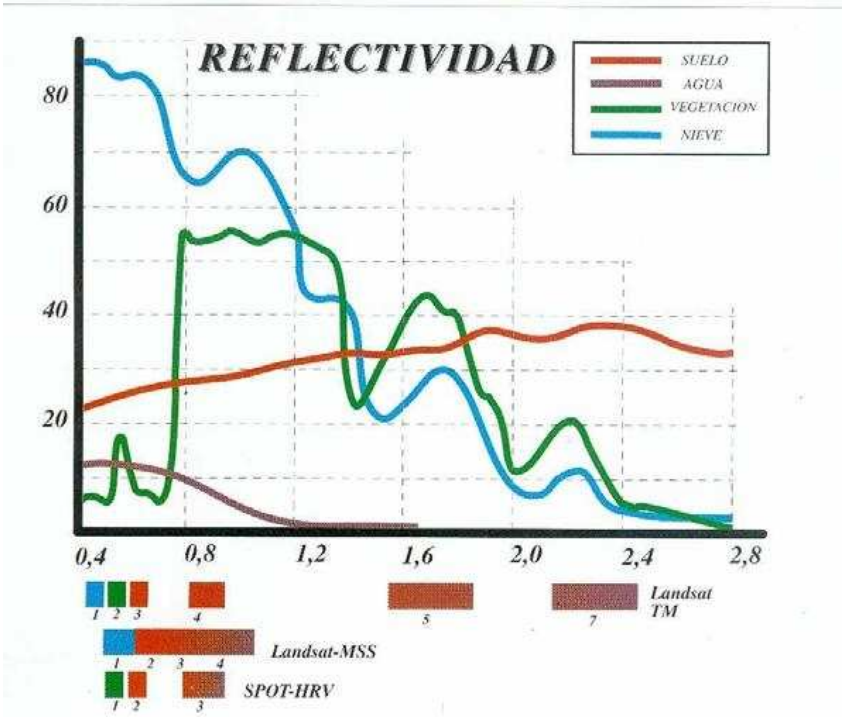


Figura 2.5: Ejemplo en el espectro completo (diferentes longitudes de onda) para 4 tipos de cubiertas [Wikib]

- **Extracción de características: Clases espectrales**

Los *píxeles* con características espectrales similares, por medio de los diferentes métodos de clasificación existentes, se van a ir agrupando en nubes o conglomerados que van a constituir las *clases espectrales*. Observando las curvas que representan la respuesta espectral de las clases de interés, se deduce la gran variabilidad que aparece en los datos reales. Las clases van a formar una nube de puntos que sugieren que patrones de una misma clase (son similares) se representan cercanos en el espacio de representación (banda del visible y del infrarrojo). Al contrario ocurre con clases diferentes, que se representan lejanos en ese espacio [Arqu 03]. La Figura 2.6 esquematiza una representación de un dominio espectral *2D (Bi, Bj)* donde aparecen 3 *clusters* que representan a las clases espectrales 1, 2 y 3.

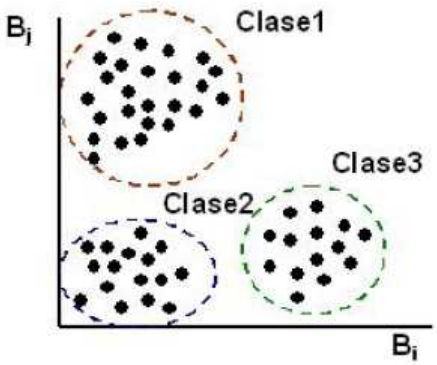


Figura 2.6: Representación 2D en el domino espectral *Bi, Bj* de clases espectrales [Arqu 03]

3. METODOLOGÍA

Tal y como se ha comentado en el capítulo introductorio del presente documento, este trabajo se centra en el estudio e implementación de una herramienta para la generación y caracterización de objetos en imágenes. Concretamente, este capítulo se centrará en dar a conocer la metodología utilizada durante este estudio, abordando todos y cada uno de los pasos que dicha metodología propone para obtener los resultados deseados.

3.1. Introducción

Disponer de imágenes de alta resolución espectral y espacial es importante cuando se abordan estudios en zonas urbanas, forestales heterogéneas o agrícolas muy parceladas. Una alta resolución espacial permite delimitar de forma más precisa la superficie ocupada por cada una de ellas [Gonz 04].

El origen del procesado de imágenes digitales basado en objetos se basa en un matiz humano: el sistema visual de las personas funciona interpretando las imágenes mediante el reconocimiento de los objetos que la componen, objetos que a su vez se caracterizan por una serie de atributos: color, tamaño, forma, textura,… El sistema visual humano no concibe la percepción de los puntos individuales, *píxeles*, que forman la imagen, puntos que están caracterizados única y exclusivamente por el valor digital que tienen asociado y los define. Esta estrategia (percepción de los *píxeles* de la imagen) es la causante de la mayoría de las debilidades detectadas en los métodos de procesado de imágenes digitales basados en *píxeles*, problemas que se ven acrecentados cuando la estrategia se aplica a imágenes de muy alta resolución espacial y baja resolución espectral. Dicho esto, no debería extrañar el hecho de que la clasificación basada en objetos, y utilizando diferentes tipos de atributos, minimiza considerablemente la alta variabilidad de los diferentes tipos de cubiertas presentes en las imágenes tomadas por satélite, consiguiéndose así mejorar de forma considerable la precisión de los resultados obtenidos. Identificar y caracterizar los objetos represente el paso más crítico e importante en estas metodologías. Dicha identificación se lleva a cabo mediante un

proceso de segmentación. La segmentación consiste en agrupar *píxeles* vecinos con atributos comunes. La complejidad de la cubierta terrestre implica que la segmentación deba ser muti-escala, para así poder adaptarse al tamaño de los diferentes objetos presentes en ellas. Entiéndase multi-escala como las diferentes dimensiones espaciales en las que las entidades, patrones y procesos se pueden observar y medir.

La bibliografía propone muchas y variadas formas de modelar este carácter multi-escala. Las transformadas multi-resolución son una de las más comunes y extendidas y, entre ellas, concretamente, la transformada *Wavelet*, la cual permite separar el detalle de las imágenes de su información de fondo, para diferentes escalas. Es necesario caracterizar los segmentos con una serie de atributos, permitiendo de esta manera determinar su similitud o diferencias respecto a otros, siendo este proceso de caracterización el que transforma segmentos en objetos. El objetivo de este trabajo es implementar la metodología propuesta en el artículo “Caracterización multiescala de objetos como herramienta para la clasificación de imágenes de alta resolución espacial” [Gonz 12]. Dicha metodología, de forma sencilla y asequible, propone los algoritmos y métodos necesarios para la definición y caracterización de objetos multi-escala en imágenes de alta resolución espacial [Gonz 12].

En la Figura 3.1 se puede observar el esquema completo de la metodología propuesta en [Gonz 12] para la definición de segmentos y su caracterización para generar los objetos clasificables. En los próximos apartados del documento se detallan las diferentes etapas del proceso completo, exponiéndose un breve resumen a continuación.

Se parte de dos imágenes: una imagen pancromática y otra multiespectral. El trabajo comienza con el estudio de la extracción de las características texturales de la imagen pancromática a través de la transformada *Wavelet* (en el ámbito del análisis multi-resolución, se permite extraer el detalle espacial que se pierde al pasar de una resolución espacial a otra menor, eliminando su información espectral).

A continuación, la metodología propone extraer los objetos texturales homogéneos de la imagen pancromática aplicando algún algoritmo de segmentación (*Otsu*, *Watershed*, etc.), y los objetos espectrales de la imagen multiespectral aplicando algún algoritmo de *clustering* (*REF K-Means*, *Fuzzy C-Means* o *Spatial Fuzzy C-Means*). Ambos objetos se integran a continuación, y dichos objetos serán referidos como *conjuntos* en el presente texto.

Cada conjunto tendrá un vector de atributos asociado donde estarán reflejados los valores de ciertas funciones como la media, perímetro, entropía, etc. Finalmente, se procede a la espacialización de dichas imágenes y sus correspondientes atributos.

La herramienta *IJObject* implementa una metodología ya propuesta en [Gonz 12] y usa los algoritmos de segmentación *Otsu* y *Watershed* y los algoritmos espectrales *REF K-Means*, *Fuzzy C-Means* y *Spatial Fuzzy C-Means*, todos ellos ya implementados en *ImageJ*. Por su parte, *IJObject* implementa la integración de los objetos texturales y espectrales, calcula los atributos sobre los distintos conjuntos detectados, y pinta las imágenes espacializadas para los distintos atributos.

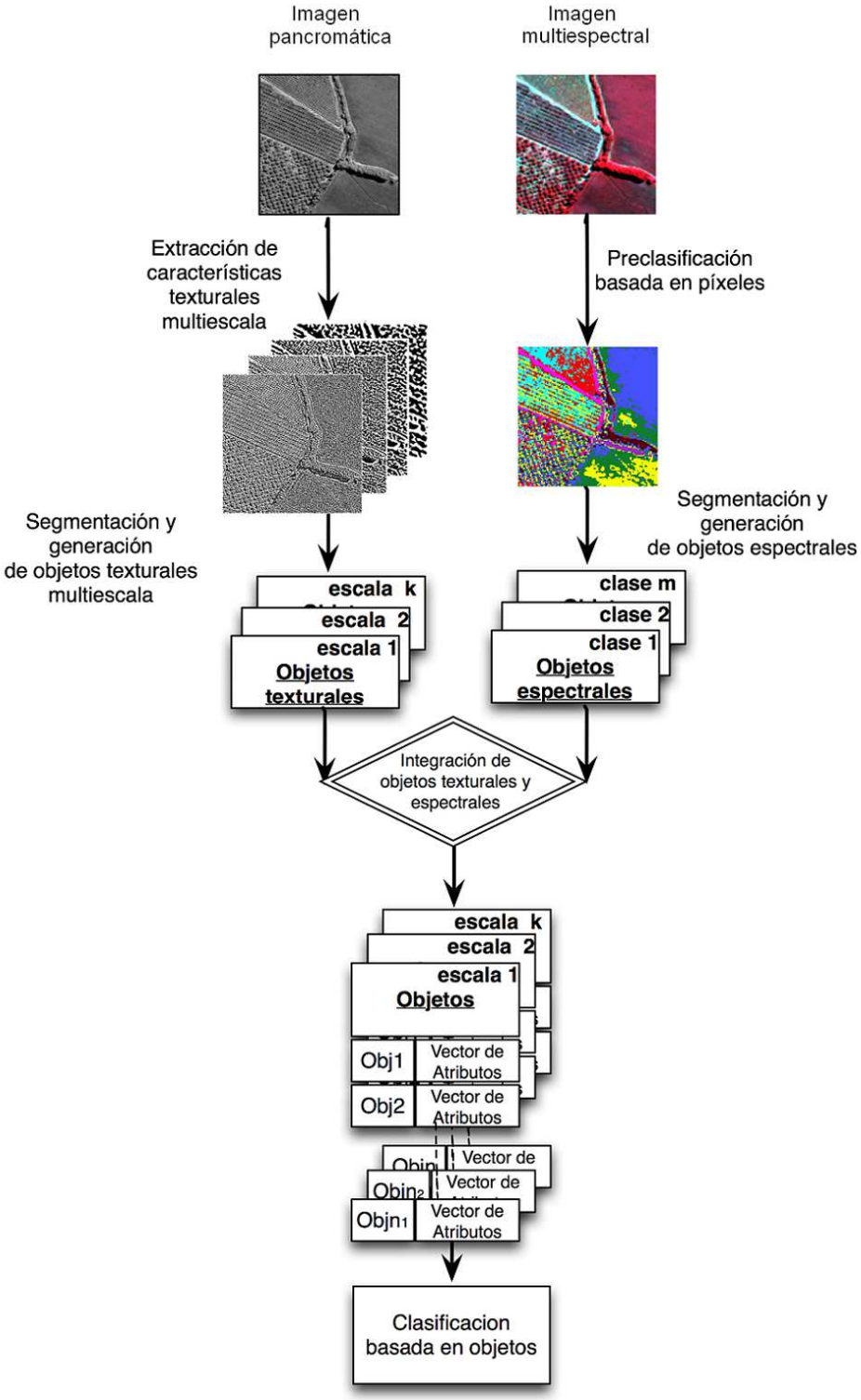


Figura 3.1: Esquema de la metodología propuesta por [Gonz 12] para la obtención de conjuntos e implementada en este trabajo

3.2. Generación de los segmentos multi-escala

Desde un punto de vista morfológico y textural, la definición de los segmentos se lleva a cabo a partir de la imagen pancromática. Esta eventualidad se debe a que la imagen pancromática es la imagen de mayor resolución espacial de entre las dos imágenes fuente. El proceso se realiza sobre los coeficientes obtenidos mediante la transformada multi-resolución *wavelet à trous*, en lugar de realizarse sobre la imagen pancromática directamente. De esta forma, se consigue eliminar la información espectral contenida en la imagen. La Figura 3.2 ilustra cómo [Gonz 12] obtiene dichos coeficientes *wavelet* para la imagen pancromática (*PAN*), los cuáles se van a utilizar ya calculados de esta forma en el presente trabajo, por lo que no ha sido necesario llevar a cabo esta transformada. Si se repitiese este proceso, se obtendrían los diferentes filtros para cada uno de los niveles de la transformada. De esta forma, se podrían definir segmentos a diferentes escalas. Nótese que la escala a utilizar dependerá de la imagen concreta con la que se pretenda trabajar, así como el propósito final de la aplicación.

En primer lugar, en este trabajo se obtienen los objetos texturalmente homogéneos mediante la aplicación de un algoritmo de segmentación por umbralización, como *Otsu* o *Watershed*, a los coeficientes *wavelet*. Por otro lado, se determinan los objetos espectralmente homogéneos a partir de las clases encontradas en la imagen multiespectral. Para ello es necesario haber clasificado con anterioridad dicha imagen mediante un algoritmo clasificador no supervisado basado en *píxeles*, como *REF K-Means*, *Fuzzy C-Means* o *Spatial Fuzzy C-Means*. Finalmente, se procede a la integración de los dos tipos de segmentos detectados (texturalmente homogéneos y espectralmente homogéneos), lo que sumado a su posterior caracterización (ver próximo apartado para conocer más detalles), permite obtener objetos definidos por un patrón de comportamiento espectral y textural diferenciador [Gonz 12].

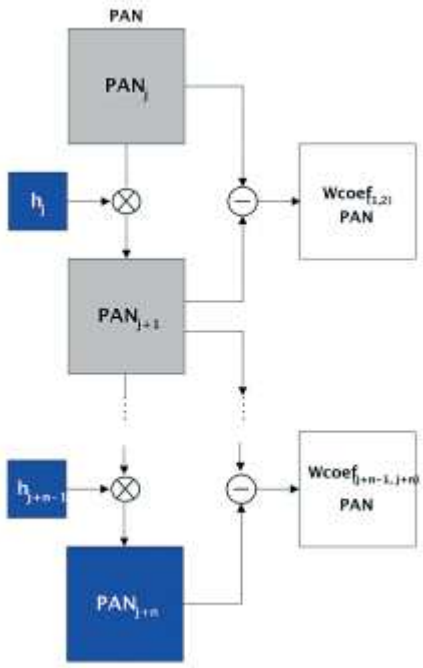


Figura 3.2: Esquema de la obtención de los coeficientes de la imagen pancromática mediante la transformada *wavelet à trous* definida por [Gonz 12]

3.3. Caracterización de los segmentos

Una vez se han identificado los objetos, es necesario caracterizarlos. Dicho proceso se lleva a cabo mediante la asociación de los atributos texturales y espectrales que se consideren más adecuados para la escena objeto de estudio. Es decir, cada objeto identificado pasaría a estar caracterizado por un vector de atributos. El vector, cuyo tamaño lo determina el número de atributos asociados al segmento, proporciona una representación simplificada y conjunta de las dos imágenes utilizadas como fuente (imagen pancromática e imagen multiespectral). Adicionalmente, cada objeto debe estar definido por las coordenadas de los *píxeles* que lo forman. De esta forma, una vez sean procesados, se podrá proceder a realizar la espacialización de dichos objetos. Por lo tanto, finalmente, se obtendría una lista de objetos con sus correspondientes atributos. Los costes computacionales de procesar este tipo de estructuras de datos son notablemente inferiores a los costes que supondrían el procesado de imágenes basado en *píxeles* [Gonz 12].

3.4. Objetos texturalmente homogéneos: Segmentación

Con los **métodos de segmentación** se trata de asignar cada *píxel* a un cierto grupo, llamado comúnmente *segmento*. La imagen que se debe segmentar está compuesta por valores numéricos. La pertenencia de un *píxel* a un cierto segmento se decide mediante la comparación de su nivel de gris (u otro valor unidimensional) con un cierto valor umbral. El nivel de gris de un *píxel* equivale a su nivel de luminosidad; el resto de la información sobre el color no se tiene en cuenta [Wikid]. El punto clave es la elección del valor umbral más adecuado, por lo que se necesita un método que permita calcular el mejor valor umbral automáticamente.

Los métodos del valor umbral son un grupo de algoritmos cuya finalidad es segmentar gráficos rasterizados, es decir, separar los objetos de una imagen que interesen del resto. Con la ayuda de los métodos de valor umbral, en las situaciones más sencillas se puede decidir qué *píxeles* conforman los objetos que se buscan y qué *píxeles* son sólo el entorno de estos objetos [Wikid].

3.4.1. Algoritmo de *Otsu*

El método del Algoritmo de *Otsu* calcula el valor umbral de forma que la dispersión de los niveles de gris dentro de cada segmento sea lo más pequeña posible, pero al mismo tiempo la dispersión sea lo más alta posible entre segmentos diferentes. Para ello, se calcula el cociente entre ambas variancias y se busca un valor umbral para el que este cociente sea máximo.

Como punto de partida, se toman dos segmentos de puntos ($K_0(t)$ y $K_1(t)$), que serán definidos a partir del valor umbral t. Donde t es la variable a buscar, y los dos segmentos son el resultado deseado en la segmentación.

Sea $p(g)$ la probabilidad de ocurrencia del valor de gris $0 < g < G$ (G es el valor de gris máximo). Entonces la probabilidad de ocurrencia de los *píxeles* en los dos segmentos se calcula como se indica en las ecuaciones (1):

(1) K_0: $P_0(t) = \sum_{g=0}^{t} p(g)$ K_1: $P_1(t) = \sum_{g=t+1}^{G} p(g) = 1 - P_0(t)$

Si se toman dos segmentos (o sea un sólo valor umbral), la suma de estas dos probabilidades dará evidentemente 1.

Si $\overline{g}$ es la media aritmética de los valores de gris en toda la imagen, y $\overline{g_0}$ y $\overline{g_1}$ los valores medios dentro de cada segmento, entonces se pueden calcular las variancias dentro de cada segmento como se indica en las ecuaciones número (2):

(2) $\sigma_0^2(t) = \sum_{g=0}^{t}(g - \overline{g_0})^2 \mathrm{p(g)}$ $\sigma_1^2(t) = \sum_{g=t+1}^{G}(g - \overline{g_1})^2 p(g)$

La meta es mantener la variancia dentro de cada segmento lo más pequeña posible y conseguir que la variancia entre los dos segmentos sea lo más grande posible. Así se obtiene (3):

(3) $Q(t) = \frac{\sigma_{zw}^2(t)}{\sigma_{in}^2(t)}$

La variancia entre los segmentos se calcula mediante (4):

(4) $\sigma_{zw}^2(t) = P_0(t) \cdot (\overline{g_0} - \bar{g})^2 + P_1(t) \cdot (\overline{g_1} - \bar{g})^2$

La variancia dentro de los segmentos se obtiene de la suma de ambas (5):

(5) $\sigma_{in}^2(t) = P_0(t) \cdot \sigma_0^2(t) + P_1(t) \cdot \sigma_1^2(t)$

El valor umbral *t* se elige de manera que el cociente *Q(t)* sea máximo. *Q(t)* es, por lo tanto, la medida buscada. De esta forma se elige un valor umbral que optimiza los dos segmentos en términos de variancia [Wikid].

3.4.2. Algoritmo de *Watershed*

La transformación ***Watershed*** es una técnica morfológica de segmentación de imágenes de niveles de gris. Es un método de segmentación basado en regiones, que divide todo el dominio de la imagen en conjuntos conexos. El concepto de *Watershed* procede del campo de la topografía: en un relieve topográfico, las líneas *watershed* son las fronteras de separación entre las cuencas de deyección de ríos y lagos. Además, cada cuenca está asociada a un mínimo local de relieve. La transformación *Watershed* se puede aplicar a imágenes en escala de grises (multinivel), teniendo en cuenta que la intensidad de un punto representa una altura equivalente en un relieve topográfico asociado. Este algoritmo asigna una etiqueta especial a los *píxeles* integrantes de las líneas *watershed*, que constituyen la frontera de separación entre las cuencas o regiones *watershed*. Adicionalmente también se etiqueta cada una de estas cuencas con un valor identificativo, para facilitar el tratamiento posterior. Con el objeto de separar zonas homogéneas de la imagen, no se suele trabajar con la transformación *Watershed* de la imagen original, sino que se opera a partir de una imagen gradiente. De esta manera se espera que los bordes de alto valor de gradiente se correspondan siempre con líneas *watershed*. El problema de este algoritmo es el gran número de mínimos locales que presenta una imagen digital ruidosa, cada uno de ellos asociado a una cuenca *watershed*, luego este operador morfológico produce una gran sobresegmentación en pequeñas regiones cuando se aplica a una imagen sin preprocesar [UPM].

Una definición algorítmica de la transformada *Watershed* por inmersión fue dada por *Vincent and Soille* [Roerd 01]. Dada la función $f: D \rightarrow N$ que representa ser un valor digital gris de la imagen, con $h_{mín}$ y $h_{máx}$ siendo el mínimo y máximo valor de *f*. Define una recursividad con el nivel de gris *h* incrementándose desde $h_{mín}$ y $h_{máx}$ en la que los valores mínimos asociados a los mínimos de *f* son sucesivamente expandidos. Sea *Xh* la denotación de la unión de los conjuntos de puntos mínimos procesados al nivel *h*. Un componente asociado al conjunto de umbrales T_{h+1} en el nivel *h+1* puede ser o bien un nuevo mínimo o una extensión de los puntos mínimos en *Xh*: en el último caso se procesa la influencia sobre las zonas curvas de X_h dentro de T_{h+1}, con el

resultado de actualizar X_{h+1}. MIN_h denota la unión de todas las regiones mínimas a la altura *h. Watershed* por inmersión define la siguiente recursividad:

$$\begin{cases} X_{h_{min}} = \{p \in D | f(p) = h_{min}\} = T_{h_{min}} \\ X_{h+1} = MIN_{h+1} \cup IZ_{T_{h+1}}(X_h), h \in [h_{min}, h_{max}) \end{cases}$$

El *Watershed Wshed(f)* de *f* es el complemento de $X_{hmáx}$ en *D*:

$$W_{shed}(f) = D \backslash X_{h_{max}}$$

Según la recursión, en el caso de que en el nivel *h+1* todos los *píxeles* excluídos del conjunto de puntos mínimos (por ejemplo, todos los *píxeles* en T_{h+1} excepto aquellos en X_h) son potencialmente candidatos a ser asignados a la zona de influencia de los puntos mínimos en el paso *h+1*. Por tanto, la definición permite a los *píxeles* con valor de gris $h' \leq ,h$ los cuales no son aún parte de un conjunto de puntos mínimos, después de ser procesados en el nivel *h*, mezclarlos con algunos puntos mínimos del nivel más alto *h+1*. Los *píxeles* que en una iteración son equidistantes con al menos dos conjuntos de puntos mínimos más cercanos, serán provisionalmente etiquetados como *píxeles watershed* asignándoles la etiqueta *W* (refiriéndose a ellos como *píxeles W*). Sin embargo, en la siguiente iteración esta etiqueta cambiará de nuevo. Un etiquetado definitivo como *píxel watershed* puede sólo suceder después de haberse procesado todos los niveles.

Vincent ahd Soille llevaron a cabo una implementación de la recursión *Watershed* por inmersión. En este algoritmo hay dos pasos: uno en el que se ordenan los *píxeles* con respecto al incremento del nivel de gris, para tener un acceso directo a los *píxeles* de un cierto nivel de gris; otro en que se procede nivel a nivel empezando por los mínimos. La implementación utiliza una cola *FIFO* de *píxeles*, es decir, una estructura de datos en el que el primero que llega es el primero que sale, sobre la cual las siguiente operaciones se pueden llevar a cabo: *fifo_add(p, queue)* añade el *píxel p* al final de la cola, *fifo_remove(queue)* devuelve y elimina el primer elemento de la cola, *fifo_init(queue)* inicializa una cola vacía, y *fifo_empty(queue)* es una comprobación que devuelve verdadero si la cola está vacía y falso en otro caso. El algoritmo asigna una

etiqueta distinta *lab[]* a cada mínimo. Todos los nodos con nivel de gris *h* son los primeros a los que se les asigna la etiqueta *MASK*. Entonces aquellos nodos que tienen vecinos etiquetados de la iteración anterior se introducen en la cola, y las zonas curvas de estos *píxeles* son propagadas en el conjunto de *píxeles* enmascarados. Si un *píxel* es adyacente a dos o más diferentes conjuntos de puntos mínimos, se marca como nodo *watershed* con la etiqueta *WSHED*. Si el *píxel* sólo puede ser alcanzado desde nodos los cuales tienen la misma etiqueta, el nodo es incorporado al conjunto de puntos mínimos correspondiente. *Píxeles* que al final aún tengan el valor *MASK*, pertenecen a un conjunto de nuevos mínimos en el nivel *h*, cuyos componentes conectados tendrán una nueva etiqueta. Como se ve en la Figura 3.3, la complejidad en tiempo del algoritmo es linear en el número de *píxeles* de la imagen de entrada.

```
1: procedure Watershed-by-Immersion
2: INPUT: digital grey scale image G = (D, E, im).
3: OUTPUT: labelled watershed image lab on D.
4: #define INIT − 1                    (* initial value of lab image *)
5: #define MASK − 2                   (* initial value at each level *)
6: #define WSHED 0                    (* label of the watershed pixels *)
7: #define FICTITIOUS (−1, −1)        (* fictitious pixel ∉ D *)
8: curlab ← 0                         (* curlab is the current label *)
9: fifo_init(queue)
10: for all p ∈ D do
11:   lab[p] ← INIT ; dist[p] ← 0     (* dist is a work image of distances *)
12: end for
13: SORT pixels in increasing order of grey values (minimum h_min, maximum h_max)
14:
15: (* Start Flooding *)
16: for h = h_min to h_max do          (* Geodesic SKIZ of level h − 1 inside level h *)
17:   for all p ∈ D with im[p] = h do  (* mask all pixels at level h *)
18:             (* these are directly accessible because of the sorting step *)
19:     lab[p] ← MASK
20:     if p has a neighbour q with (lab[q] > 0 or lab[q] = WSHED) then
21:       (* Initialize queue with neighbours at level h of current basins or watersheds *)
22:       dist[p] ← 1 ; fifo_add(p, queue)
23:     end if
24:   end for
25:   curdist ← 1 ; fifo_add(FICTITIOUS, queue)
26:   loop          (* extend basins *)
27:     p ← fifo_remove(queue)
28:     if p = FICTITIOUS then
29:       if fifo_empty(queue) then
30:         BREAK
31:       else
32:         fifo_add(FICTITIOUS, queue) ; curdist ← curdist + 1 ;
```

```
33:           p ← fifo_remove(queue)
34:         end if
35:       end if
36:       for all q ∈ N_G(p) do            (* labelling p by inspecting neighbours *)
37:         if dist[q] < curdist and (lab[q] > 0 or lab[q] = WSHED) then
38:           (* q belongs to an existing basin or to watersheds *)
39:           if lab[q] > 0 then
40:             if lab[p] = MASK or lab[p] = WSHED then
41:               lab[p] ← lab[q]
42:             else if lab[p] ≠ lab[q] then
43:               lab[p] ← WSHED
44:             end if
45:           else if lab[p] = MASK then
46:             lab[p] ← WSHED
47:           end if
48:         else if lab[q] = MASK and dist[q] = 0 then          (* q is plateau pixel *)
49:           dist[q] ← curdist + 1 ; fifo_add(q, queue)
50:         end if
51:       end for
52:     end loop
53:     (* detect and process new minima at level h *)
54:     for all p ∈ D with im[p] = h do
55:       dist[p] ← 0              (* reset distance to zero *)
56:       if lab[p] = MASK then            (* p is inside a new minimum *)
57:         curlab ← curlab + 1 ;          (* create new label *)
58:         fifo_add(p, queue) ; lab[p] ← curlab
59:         while not fifo_empty(queue) do
60:           q ← fifo_remove(queue)
61:           for all r ∈ N_G(q) do          (* inspect neighbours of q *)
62:             if lab[r] = MASK then
63:               fifo_add(r, queue) ; lab[r] ← curlab
64:             end if
65:           end for
66:         end while
67:       end if
68:     end for
69:   end for
70:   (* End Flooding *)
```

Figura 3.3: El Algoritmo de *Watershed* por *Vincent-Soille*

De hecho, el algoritmo de *Vincent-Soille* no implementa la recursividad matemática expuesta al principio, por las siguientes razones (los números de línea mencionados van a referirse al pseudocódigo de la Figura 3.3):

- En el nivel *h* sólo los *píxeles* con valor de gris *h* son enmascarados (línea 17), en lugar de todos los *píxeles* que no pertenecen al conjunto de mínimos de valores menores o iguales que *h*, como la definición requería.
- No sólo las etiquetas del conjunto de valores mínimos son propagadas, sino también las etiquetas de los *píxeles WSHED* (línea 20). La necesidad de esto es

consecuencia del punto anterior. Ya que el algoritmo intenta clasificar *píxeles* como *píxeles WSHED* en el actual nivel de gris, las etiquetas *watershed* tienen que propagarse porque puede darse el caso de que *píxeles* con valor de gris h sólo tienen *píxeles WSHED* en sus vecinos.

- Un *píxel* que es adyacente a dos conjuntos de mínimos diferentes, y por tanto, inicialmente están etiquetados con *WSHED*, se permite sobreescribirlo al nivel actual de gris por la etiqueta de otro *píxel* vecino, si ese *píxel* es parte de un conjunto de puntos mínimos (líneas 40-41). La motivación dada en la Figura 3.3 no es otra que el hecho que puedan divergir las líneas *watershed*. Esta sentencia probablemente está basada en una esperada intuición para el caso de funciones en espacio continuo. Una valoración de lo correcta que es una implementación debería estar basada únicamente de acuerdo con la definición [Roerd 01].

3.5. Objetos espectralmente homogéneos: *Clustering*

En este apartado se van a describir 3 técnicas de *clustering* usadas en la segmentación de imágenes multiespectrales. Los 3 algoritmos que se abordarán, dado que están incluidos en *ImageJ*, son *K-Means*, *Fuzzy C-Means* y *Spatial Fuzzy C-Means*.

El *Clustering* de datos ayuda a discernir la estructura y simplifica la complejidad de cantidades masivas de datos. Es una técnica común y se utiliza en diversos campos, donde la distribución de la información puede ser de cualquier tamaño y forma. La eficiencia de los algoritmos de *clustering* es extremadamente necesaria cuando se trabaja con enormes bases de datos y tipos de datos de grandes dimensiones como, por ejemplo, trabajar con imágenes de alta resolución.

Un *cluster* es una colección de objetos que son similares entre sí según un determinado criterio de similitud y distintos a los objetos que pertenecen a otros *clusters*. Los algoritmos basados en el agrupamiento particional tienen como objetivo minimizar la varianza *intracluster* y maximizar la varianza *intercluster*. Estos métodos descomponen el conjunto de objetos en un conjunto de *clusters* disjuntos, minimizando

una función criterio que enfatiza la estructura local de los objetos, asignando *clusters* a máximos locales en la estructura global.

La segmentación por *clustering* presenta ventajas con respecto a las otras técnicas de segmentación de imágenes. La diferencia de color es el indicador de similitud más adecuado para la ejecución de los algoritmos tratados. La heurística de análisis de frecuencia de color para la inicialización de los centroides de los algoritmos vistos es la que proporciona los resultados más acertados.

3.5.1. Métodos No Supervisados

Los algoritmos de *clustering* que se van a ver en este proyecto están englobados en los llamados métodos no supervisados. Su principal característica es que no parten de información previa, hacen una búsqueda automática de grupos de valores homogéneos dentro de la imagen. Lo importante aquí es identificar agrupaciones de *píxeles* con características espectrales similares. El objetivo es definir clases espectrales como las vistas en la Figura 2.6 [Arqu 03].

El usuario fija el número inicial de clases que desea obtener y luego el *software* asigna los *píxeles* automáticamente a las distintas clases en base a operaciones estadísticas.

A continuación se ilustra en la Figura 3.4 el resultado de clasificar de forma no supervisada una imagen satélite donde se detectan automáticamente clases homogéneas [Laotra]. En este caso se obtienen 16 clases reconocidas y una sin clasificar donde los *píxeles* no comparten características con ninguna de las 16 anteriores.

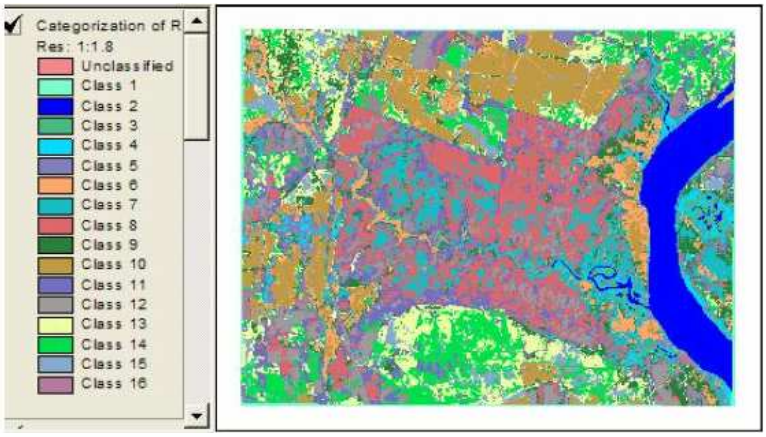


Figura 3.4: Ejemplo de clasificación no supervisada [Laotra]

En cada paso se identifican los dos individuos más próximos, se hace una clase con ellos y se sustituyen por el centroide de la clase resultante. De este modo, cada paso analiza un individuo menos que el anterior ya que los individuos van siendo sustituidos por clases. El proceso se detiene cuando se ha alcanzado un número de clases igual al número de clases que había sido establecido a priori [UM].

El resultado final de un proceso de *clustering* suele ser un dendrograma (Figura 3.5) en el que puede verse cómo los diversos individuos se aglutinan en clases, primero los que están a un menor distancia (los más parecidos), y cómo posteriormente las clases se unen entre sí. A partir de un dendrograma se puede elegir el número de clases que se quiera mantener en función de diferentes criterios [UM].

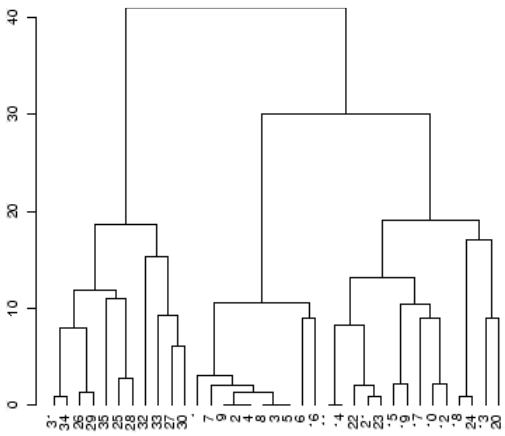


Figura 3.5: Dendograma obtenido a partir de los *píxeles* de las áreas de entrenamiento [UM]

El dendrograma de la Figura 3.5 se ha construido con los valores que aparecen en la Figura 3.6. Pueden verse claramente los 3 grupos que se han identificado en aquella figura [UM].

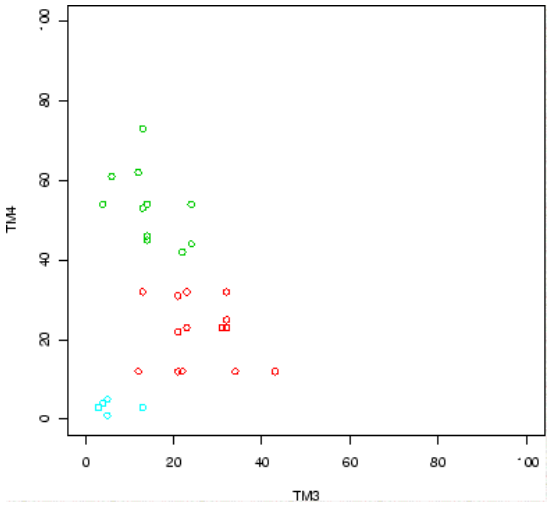


Figura 3.6: Distribución de los *píxeles* de las diferentes áreas de entrenamiento [UM]

En este proceso se clasifican todos los *píxeles*, por tanto la salida no puede ser un dendrograma por razones prácticas. La salida es un mapa en el que los *píxeles* aparecen adjudicados a las diferentes clases. Por tanto, hay que elegir a priori el número de clases que se desee. Este número debe ser elevado ya que siempre se puede a posteriori unir aquellas clases que no tengan sentido mantener separadas [UM].

3.5.2. REF K-Means

K-Means es un método particional que intenta encontrar un número específico de grupos, los cuales están representados por sus centroides, aplicable a un grupo de objetos en un espacio continuo *n-dimensional*. Es uno de los algoritmos de *clustering* más antiguos y ampliamente usados [Lorca 10].

El nombre de *K-Means* viene porque representa cada uno de los *cluster* por la media (o media ponderada) de sus puntos, es decir, por su centroide. La representación mediante centroides tiene la ventaja de que tiene un significado gráfico y estadístico inmediato. Cada *cluster*, por tanto, es caracterizado por su centro o centroide (ver Figura

3.7) que se encuentra en el centro o el medio de los elementos que componen el *cluster*. *K-Means* es traducido como *K-medias* [Garc 08-09].

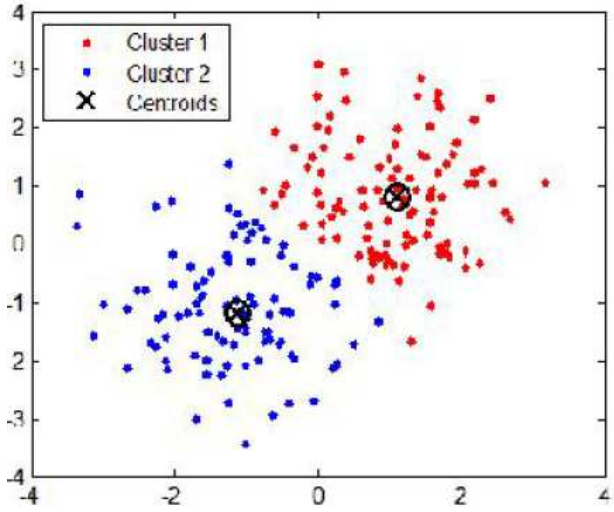


Figura 3.7: Ejemplo de 2 *clusters* (azul y rojo) con sus respectivos centroides [Garc 08-09]

Uno de los inconvenientes principales de *K-Means*, además del hecho de que sea necesario realizar en sucesivas ocasiones el algoritmo para así tener el resultado más óptimo posible, es la necesidad de inicializar el número de prototipos al principio de la ejecución. Esto perjudica la eficacia del algoritmo ya que en la práctica no se conoce a priori el número de *cluster* final. Este defecto le perjudicará al compararlo con otros algoritmos, ya que en muchos la inicialización del número de *clusters* no es necesaria. También *K-Means* es susceptible a valores extremos porque distorsionan la distribución de los datos [Garc 08-09].

La técnica general de *clustering K-Means* es muy simple. A continuación se presenta la descripción del algoritmo básico [Lorca 10]:

1. Seleccionar *K* centroides, donde *K* es el número de *clusters* deseado.
2. Asignar cada punto al centroide más cercano y cada colección de puntos asignados a un centroide es un *cluster* (Región de *Voronoi*).
3. Actualizar los centroides de cada *cluster*, basados en los puntos asignados al *cluster*.
4. Repetir el proceso de asignación y actualización hasta que ningún punto cambie

de *cluster*, o lo que es lo mismo, hasta que los centroides permanezcan iguales.

5. Fin

Sea el conjunto de datos *X = {x1, x2,…, xn}*. Dado un centroide (representante de una agrupación) *yi*, el conjunto de puntos de *X* que está más cercano a *yi* que a cualquier otro centroide según la medida *d (x, y)*, se denomina región de *Voronoi* de *yi* y se denota por (1).

$$(1)\ Vi = \{x \in X : d(x, yi) < d(x,yj), i \neq j\}$$

El número de vectores en una región de *Voronoi* se representa por |*Vi*|. El centroide de los vectores de una región de *Voronoi* viene dado por (2).

$$(2)\frac{1}{|V_i|}\sum_{x \in V_i} x$$

El algoritmo descrito busca minimizar la siguiente función objetivo (3) donde *SSE* es la suma del cuadrado de los errores, *Ci* es el *i-ésimo cluster* de la partición, *d (x,Ci)* es la medida de disimilitud o distancia entre el elemento *x* y el *cluster Ci.*

$$(3)SSE = \sum_{i=1}^{k}\sum_{x \in C_i} d\left(x, C_i\right)^2$$

3.5.3. Fuzzy C-Means

Fuzzy C-Means (también llamado *Fuzzy K-Means*) es una extensión del *K-Means*. Mientras *K-Means* encuentra particiones para las que un punto pertenece a un solo *cluster*, *Fuzzy C-Means* es un método estadísticamente formalizado que encuentra *K clusters* donde un punto puede pertenecer a más de un *cluster* con cierto valor de pertenencia. Tiene su base en la teoría de conjuntos imprecisos o poco definidos (*Fuzzy Set*). La teoría *Fuzzy Set* es una generalización del álgebra Booleana, por lo que la función de pertenencia de un elemento a los grupos se encuentra en el intervalo [0,1]; es decir, un elemento puede pertenecer totalmente a una clase, a todas o a ninguna [Lorca 10].

Como el *K-Means, Fuzzy C-Means* trabaja con aquellos objetos que pueden ser representados en un espacio *n-dimensional* con una medida de distancia definida. El procedimiento *Fuzzy C-Means* minimiza la siguiente función objetivo:

$$SSE(M,C) = \sum_{i=1}^{n}\sum_{j=1}^{k} m_{ij} d_{ij}^2$$

$$(1) \qquad \text{Sujeto a: } \sum_{j=1}^{k} m_{ij} = 1 \quad \text{j = 1,2,...,n}$$

$$\sum_{i=1}^{n} m_{ij} > 0 \quad \text{j = 1,2,...,k } m_{ij} \in [0,1]$$

Donde (1) *SSE (M, C)* es la suma del cuadrado de los errores dentro de las clases, *M* es la matriz (*n x k)* de pertenencia a los grupos ($m_{ij} = 0$ es lo contrario), *C* es la matriz (*k x p)* de centro de las clases siendo *p* el número de componentes del espacio, ϕ es el grado de imprecisión de la solución (*fuzziness* exponente) y d^2_{ij} es el cuadrado de la distancia entre el elemento *i* y el centro representativo del *cluster j*. El algoritmo de solución de la función objetivo consta de las siguientes etapas iterativas [Lorca 10]:

1. Seleccionar el número de clases *k*, con $1 < k < n$. Si *k* es 1 ó *n*, el análisis no es necesario.

2. Seleccionar el valor de *fuzziness* exponente ϕ con $\phi > 1$. Los valores comúnmente usados están en el rango 1.1 a 2.

3. Seleccionar la definición de distancia en el espacio variable. Las distancias más usadas son la *Euclídea* y *Mahalanobis*. Seleccionar un valor del criterio de detención, $e = 0.001$, da una convergencia razonable. Iniciar con $M = M_0$, por ejemplo, con una agrupación aleatoria o con agrupación de partición rígida (*K-Means*).

4. En las iteraciones $i = 1, 2, 3, \ldots$ re-calcular $C = C_i$ usando M_{i-1} con la ecuación (2).

$$(2) C_j = \frac{\sum_{i=1}^{n} m_{ij} x_i}{\sum_{i=1}^{n} m_{ij}}$$

5. Re-calcular M = M1 usando Ci y la ecuación (3).

$$(3) m_{ij} = \frac{d_{ij}^{2/(\Phi-1)}}{\sum_{r=1}^{k} d_{ir}^{2/(\Phi-1)}}$$

Si $||Mi - Mi\text{-}1|| < e$ entonces parar, sino retornar al paso 5.

Donde $||Mi - Mi\text{-}1||$ es el mayor valor absoluto de la diferencia entre los elementos de la matriz *Mi* y sus correspondientes elementos de la matriz *Mi-1*.

Según el criterio de detención *e*, el algoritmo converge con mayor o menor número de iteraciones. Sin embargo, la solución no es siempre óptima pues puede converger hacia mínimos locales en función de la estimación inicial. Esta misma desventaja fue vista en el algoritmo *K-Means* aunque este método mejora el problema de convergencia del *K-Means* [Lorca 10].

3.5.4. Spatial Fuzzy C-Means

Spatial Fuzzy C-Means es una variante de *Fuzzy C-Means* que tiene en cuenta la información espacial asociada alrededor de cada *píxel* de una imagen.

Spatial Fuzzy C-Means cambia la pertenencia de cada *píxel* con respecto a un grupo, teniendo en cuenta el número de miembros de los *píxeles* próximos a él. Se considera un barrio, que es una ventana de radio fijo y centrada en el *píxel* para calcular la información espacial en la forma de una función *h* (ver (1)) [Capo 11].

$$(1) h_{ij} = \sum_{c \in NB(x_i)} u_{cj}$$

La información contenida en *hij* se refiere a la probabilidad de que el *píxel i-ésimo* pertenezca al *cluster j-ésimo* y pueda ser estimado como la suma de los miembros de los

vecinos del *píxel i-ésimo* respecto al *cluster j-ésimo*. Cuantos más *píxeles* en la zona pertenecen al mismo grupo, mayor será el valor de *hij* [Capo 11].

3.5.5. Atributos

Un *atributo* o *medida* de una imagen es una función matemática que asigna un número a dicha imagen tomando como valores los niveles de grises de cada uno de los *píxeles* que forman la imagen. [Wikii]

Las medidas se calculan para cada conjunto, no para el total de la imagen. Se van a agrupar en tres grandes grupos: medidas de forma, medidas espectrales y medidas texturales.

3.5.5.1. Medidas de forma

Área

El área del conjunto es igual al número de *píxeles* que lo forman.

Perímetro

El cálculo del perímetro parte de la premisa de que para un conjunto formado por un sólo *píxel*, el perímetro sería igual a 4. Dicho esto, el cálculo del perímetro se calcula mediante el siguiente algoritmo:

Se itera sobre todos los *píxeles* que forman el conjunto y se parte de un contador inicializado a cero. Para cada *píxel* del conjunto:

Se comprueba si el *píxel* superior pertenece al conjunto. En caso negativo o de que el *píxel* forme parte del borde superior de la imagen, se incrementa en uno el contador.

Se comprueba si el *píxel* inferior pertenece al conjunto. En caso negativo o de que el *píxel* forme parte del borde inferior de la imagen, se incrementa en uno el contador.

Se comprueba si el *píxel* situado a la izquierda pertenece al conjunto. En caso negativo o de que el *píxel* forme parte del borde izquierdo de la imagen, se incrementa en uno el contador.

Se comprueba si el *píxel* situado a la derecha pertenece al conjunto. En caso negativo o de que el *píxel* forme parte del borde derecho de la imagen, se incrementa en uno el contador.

Tras haber iterado sobre todos los *píxeles* del conjunto, el valor del contador dará el valor del perímetro de la imagen.

Centro de masa

Se devuelven dos valores: *XM* e *YM* (centro de masa para las coordenadas *X* e *Y*).

$$XM = \frac{\sum_{i=1}^{n} x_i \, m_i}{\sum_{i=1}^{n} m_i}$$

$$YM = \frac{\sum_{i=1}^{n} y_i \, m_i}{\sum_{i=1}^{n} m_i}$$

Es decir, *XM* es la suma de los pesos (valor de tono de gris) de cada *píxel* por el valor de su coordenada *x*, entre el sumatorio de todos los pesos (valores de tonos de gris de cada *píxel*) del conjunto. *YM* es igual, sólo que considerando la coordenada *y* en lugar de la *x*.

Hay un caso especial a considerar: se trata del caso en el que el sumatorio de todos los pesos es igual a 0 (lo que significa que todos los *píxeles* del conjunto son de color negro). En dicho caso, *XM* e *YM* valen las coordenadas *x* e *y*, respectivamente, del punto intermedio del conjunto.

Descriptores de Forma

Se devuelven 3 valores: circularidad, relación de aspecto y redondez.

- *Circularidad*

Se calcula acorde a la siguiente fórmula:

$$circ = \frac{4\,\pi\ area}{perimeter^2}$$

El cálculo tanto del área como del perímetro del conjunto ya han sido explicados con anterioridad.

- *Relación de aspecto*

Se calcula según la siguiente fórmula:

$$AR = \frac{major\ axis}{minor\ axis}$$

donde: $major\ axis = \frac{width^2}{\sqrt{\pi}}$

$$minor\ axis = \frac{height^2}{\sqrt{\pi}}$$

El cálculo de la anchura y de la altura del conjunto es sencillo. Para la anchura se resta al mayor valor de coordenada *x* del conjunto el menor *x* del mismo. Para el cálculo de la altura se hace el mismo proceso, pero teniendo en cuenta los valores de la coordenada *y*.

- *Redondez*

Para su cálculo se sigue la fórmula siguiente:

$$round = \frac{4\ area}{\pi\ \ major\ axis^2}$$

Los cálculos del área y del *major axis* ya han sido comentados con anterioridad.

Bordes del rectángulo

Se definen mediante 4 valores: *BX, BY, width* y *height*. La idea es la de encontrar el rectángulo más pequeño capaz de encerrar todos los *píxeles* del conjunto.

BX es el mínimo valor de coordenada *x* de todo el conjunto.

BY es el mínimo valor de coordenada *y* de entre todos los *píxeles* del conjunto.

Los cálculos de w*idth* y *height* ya han sido explicados con anterioridad en el presente apartado del documento.

3.5.5.2. Medidas espectrales

Media

El cálculo de la media es sencillo. Se suman todos los valores de nivel de gris de todos los *píxeles* del conjunto, y dicho valor se divide entre el número de *píxeles* que forman el conjunto.

Mínimo y máximo valor de gris

Se retornan dos valores: el mínimo valor de gris de todos los *píxeles* que conforman el conjunto y el máximo valor de gris.

Desviación estándar

La deviación estándar se calcula de acuerdo a la fórmula que se presenta a continuación: $$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})}$$

Donde:

N es el número total de *píxeles* del conjunto.

X_i es el valor de tono de gris del *píxel* considerado en cada momento.

$\bar{x}$ es la media del conjunto (explicada con anterioridad).

Mediana

La mediana se calcula como la media de los dos valores (tono de gris) centrales de entre todos los que forman el conjunto en cuestión. El algoritmo a seguir para su cálculo sería el siguiente:

Se obtiene un listado con todos los valores de nivel de gris del conjunto. A continuación, se ordena dicha lista en orden ascendente. Finalmente, se toman los dos valores centrales y se calcula su media aritmética, obteniendo así la mediana del conjunto.

3.5.5.3. Medidas texturales

GLCM

Para el cálculo de las medidas texturales es necesario calcular, previamente, la denominada *Gray Level Correlation Matrix* (*GLCM* a partir de ahora). Para el cálculo de dicha matriz, son necesarios dos parámetros elegidos por el usuario: tamaño del *step* en *píxeles* y dirección del *step* (0, 90, 180 y 270 grados, y media).

A continuación se explica el cálculo de *GLCM* dependiendo de la dirección del *step* elegida:

- *0 Grados*

Se consideran todos los *píxeles* que forman el conjunto. Se genera una matriz de tamaño 257x257. Para todos y cada uno de ellos, se calculan dos números, que se llamarán *a* y *b*. El valor de *a*, se calcula como la operación lógica *AND* de -1 y el valor del *píxel* en cuestión. Para el cálculo de *b*, depende de si el *píxel* que ocupa la posición (*x+step,y*) pertenece o no al conjunto en cuestión. Si pertenece, *b* se calcula como la función lógica *AND* de -1 y el valor del *píxel* (*x+step,y*), siendo *x* e *y* las coordenadas del *píxel* que se están considerando en este momento. En caso de no pertenecer al conjunto, *b* se calcula como la función lógica *AND* de -1 y 0. Una vez calculados *a* y *b*, se incrementa en 1 el valor de la matriz para las posicionas *a,b* y *b,a*.

- *90 Grados*

Se consideran todos los *píxeles* que forman el conjunto. Se genera una matriz de tamaño 257x257. Para todos y cada uno de ellos, se calculan dos números, que se llamarán *a* y *b*. El valor de *a* se calcula como la operación lógica *AND* de -1 y el valor del *píxel* en cuestión. Para el cálculo de *b*, depende de si el *píxel* que ocupa la posición (*x*,*y-step*) pertenece o no al conjunto en cuestión. Si pertenece, *b* se calcula como la función lógica *AND* de -1 y el valor del *píxel* (*x*,*y-step*), siendo *x* e *y* las coordenadas del *píxel* que se están considerando en este momento. En caso de no pertenecer al conjunto, *b* se calcula como la función lógica *AND* de -1 y 0. Una vez calculados *a* y *b*, se incrementa en 1 el valor de la matriz para las posicionas *a*,*b* y *b*,*a*.

- *180 Grados*

Se consideran todos los *píxeles* que forman el conjunto. Se genera una matriz de tamaño 257x257. Para todos y cada uno de ellos, se calculan dos números, que se llamarán *a* y *b*. El valor de *a* se calcula como la operación lógica *AND* de -1 y el valor del *píxel* en cuestión. Para el cálculo de *b*, depende de si el *píxel* que ocupa la posición (*x-step*,*y*) pertenece o no al conjunto en cuestión. Si pertenece, *b* se calcula como la función lógica *AND* de -1 y el valor del *píxel* (*x-step*,*y*), siendo *x* e *y* las coordenadas del *píxel* que se está considerando en este momento. En caso de no pertenecer al conjunto, *b* se calcula como la función lógica *AND* de -1 y 0. Una vez calculados *a* y *b*, se incrementa en 1 el valor de la matriz para las posicionas *a*,*b* y *b*,*a*.

- *270 Grados*

Se consideran todos los *píxeles* que forman el conjunto. Se genera una matriz de tamaño 257x257. Para todos y cada uno de ellos, se calculan dos números, que se llamarán *a* y *b*. El valor de *a* se calcula como la operación lógica *AND* de -1 y el valor del *píxel* en cuestión. Para el cálculo de b, depende de si el *píxel* que ocupa la posición (*x*,*y+step*) pertenece o no al conjunto en cuestión. Si pertenece, *b* se calcula como la función lógica *AND* de -1 y el valor del *píxel* (*x*,*y+step*), siendo *x* e *y* las coordenadas del *píxel* que se está considerando en este momento. En caso de no pertenecer al

conjunto, *b* se calcula como la función lógica *AND* de -1 y 0. Una vez calculados *a* y *b*, se incrementa en 1 el valor de la matriz para las posicionas *a,b* y *b,a*.

- *Media*

Esta matriz se calcula como la media de las otras cuatro. Finalmente, hay que dividir cada elemento de la matriz entre el número total de *píxeles* del conjunto.

Una vez explicado cómo se obtiene *GLCM*, se procede a desarrollar las distintas medidas texturales que se describen a continuación.

Segundo momento angular

Es la suma de los cuadrados de cada uno de los valores de la matriz *GLCM*.

Contraste

Sea *Cab* el valor de la matriz *GLCM* para la fila a columna *b*. El contraste es el sumatorio del cuadrado de (*a* – *b*) multiplicado por *Cab* para todos los elementos de la matriz.

Correlación

Sea *Cab* el valor de la matriz *GLCM* para la fila *a* y columna *b*. En primer lugar, se calculan dos valores, que se denominan *px* y *py*, siendo *px* el sumatorio de *a* multiplicado por *Cab* y *py* el sumatorio de *b* multiplicado por *Cab*, para todos los elementos de la matriz. A continuación, se calculan dos desviaciones estándares, que se denominan *stdevx* y *stdevy*. *Stdevx* se calcula como el sumatorio del cuadrado de (*a* – *px*) multiplicado por *Cab* sobre todos los elementos de la matriz *GLCM*. *Stdevy* se calcula como el sumatorio del cuadrado de (*y* – *px*) multiplicado por *Cab* sobre todos los elementos de la matriz *GLCM*. Finalmente, la correlación se calcula como el sumatorio de (*a* – *px*) multiplicado por (*b* – *py*) multiplicado por *Cab* y entre *stdevx* multiplicado por *stdevy*.

Homogeneidad

Sea *Cab* el valor de la matriz *GLCM* para la fila *a* y columna *b*. Para todos los elementos de la matriz, *IDM* se calcula como el sumatorio *Cab* entre 1 más el cuadrado de (*a* – *b*).

Entropía

Sea *Cab* el valor de la matriz *GLCM* para la fila *a* y columna *b*. La entropía se calcula como el sumatorio de menos *Cab* por el logaritmo neperiano de *Cab,* para todos aquellos *Cab* distintos de cero.

4. IMPLEMENTACIÓN DE LA HERRAMIENTA

4.1. Desarrollo *Software*

4.1.1. Introducción

IJObject es un producto *software* que satisface unos requisitos propuestos en el ámbito del procesamiento de imágenes. Como cualquier otro *software*, cumple los siguientes criterios:

- Funcionamiento → El *software* debe funcionar siempre.
- Funcionalidad → El *software* debe cubrir las funcionalidades que publica.
- Atributos de calidad → El *software* debe satisfacer unas cualidades o propiedades.

Los atributos de calidad que todo *software* debe cumplir son los siguientes:

- Confiabilidad
    - Fiabilidad → Capacidad de ofrecer los mismos resultados bajo las mismas condiciones.
    - Robustez (tolerancia a fallos) → Capacidad de operar en situaciones excepcionales sin poseer un comportamiento catastrófico.
    - Seguridad (integridad) → Capacidad de protegerse ante ataques o fallos accidentales o deliberados.
    - Disponibilidad → Capacidad de proporcionar servicios siempre que se solicita.
- Rendimiento
    - Eficiencia → Utilización óptima de los recursos de la máquina.
    - Capacidad de respuesta → Capacidad para dar la sensación de estar siempre activos reduciendo los tiempos de respuesta.

  - Comportamiento temporal (determinismo) → Capacidad de ejecutar alguna tarea generalmente crítica en un tiempo determinista, dentro de un plazo de tiempo definido.

- Usabilidad
  - Ayuda y documentación.
  - Uso simple e intuitivo.
  - Mínimo esfuerzo en su uso.
- Soporte
  - Internacionalización.
  - Facilidad para cambiar la configuración.
  - Adaptabilidad → Capaz de modificar alguna función sin que afecte a sus actividades.
  - Facilidad de mantenimiento → Capacidad con la que se puede corregir un error, adaptarse a un entorno cambiante o cambio de requisitos.
  - Interoperabilidad → Esfuerzo necesario para acoplar un sistema en otro.
- Portabilidad → Capaz de integrarse en entornos distintos con el mismo esfuerzo.
- Reutilización → Capacidad para reutilizar elementos de desarrollo previos.

Como posteriormente se irá desarrollando, *IJObject* cumple con la mayoría de los criterios de calidad que se acaban de enumerar.

El desarrollo de un producto *software* consta de las siguientes etapas:

- Planificación → Etapa de la fase de definición del producto *software* en el que se describe el alcance funcional del *software*, se asignan los recursos, se estiman los costes y se definen las tareas y la agenda a seguir. *IJObject* es un producto

software especial (se trata de un Proyecto Fin de Carrera), por lo que no hay costes derivados ni se puede presuponer una dedicación. Por tanto, se decidió no establecer una planificación a la hora de abordar el proyecto.

- Modelado de requisitos → El cliente y los desarrolladores definen el propósito del sistema.
- Análisis → Los desarrolladores formalizan la especificación de requisitos obtenida en el modelado de requisitos en modelos que describen completamente el sistema.
- Diseño → Transformación del modelo de análisis a un modelo de diseño en el dominio de la solución seleccionando estrategias adecuadas para la construcción del sistema.
- Implementación → Se transforma el modelo de diseño en código fuente.
- Pruebas → Es el proceso de encontrar diferencias entre el comportamiento esperado del sistema y el comportamiento observado en la ejecución del sistema implementado.
- Mantenimiento → Modificación de un producto *software* después de haber sido entregado para corregir fallos, mejorar su funcionamiento u otros atributos, o para adaptarlo a cambios producidos en el entorno. Al igual que para la etapa de planificación, al tratarse de un Proyecto Fin de Carrera, no se contempla un mantenimiento para *IJObject*. No obstante, ha sido diseñado e implementado pensando en que alguien, algún día, podría continuar con el trabajo aquí desarrollado.

Para abordar la tarea de desarrollo de *software* existen múltiples y variados modelos: modelo clásico, modelo iterativo (se trata de un proceso incremental), modelo evolutivo,… Tras analizar concienzudamente los diferentes modelos propuestos en la bibliografía de Ingeniería del *Software*, se optó por el modelo clásico, en el cual se

abordan las tareas de captura de requisitos, análisis, diseño, implementación y pruebas respectivamente.

A continuación, se abordan las diferentes etapas del proceso de desarrollo de *software* para *IJObject*, comenzando con un análisis de Riesgos.

4.1.2. Análisis de riesgos

En todo proceso de Ingeniería del *Software*, es fundamental hacer un buen proceso de análisis de riesgos. Este proceso puede ser determinante en la práctica para obtener un entregable en producción en los plazos planificados.

A continuación se identifican los riesgos que se pueden encontrar en el proyecto.

4.1.2.1. Riesgos del proyecto

La estimación del tamaño puede ser significativamente baja.

La fecha de entrega está muy ajustada.

El cliente cambia los requisitos.

Enfermedad del personal disponible.

4.1.2.2. Riesgos técnicos

La tecnología no alcanza la expectativa.

Falta de formación en las herramientas.

Tiempos de ejecución muy elevados.

4.1.3. Requisitos

4.1.3.1. Captura de Requisitos

Durante esta fase, se adquieren, reúnen y especifican las características funcionales y no funcionales que deberá cumplir el futuro programa o sistema a desarrollar [Wikif].

Esta etapa involucra en gran medida al usuario o al cliente del sistema. Por ello, no existe una técnica "estrictamente adecuada" a seguir. La habilidad del analista para interactuar con el cliente durante esta etapa es fundamental. La situación habitual es la de un cliente con un objetivo o problema que resolver, y sin conocimientos en el área de la informática

Los casos de uso suponen una herramienta de vital importancia y la más usada en el proceso de captura de requisitos funcionales. A su vez, los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas [Wikig].

Para la elaboración del presente documento se ha optado por detectar y redactar los casos de uso, así como dibujar el diagrama de casos de uso, abordando de esta forma ampliamente la fase de captura de requisitos. El Sistema desarrollado involucra a 4 Actores y 4 Casos de uso, los cuales se redactan en el subapartado de Requisitos Funcionales:

4.1.3.2. Requisitos Funcionales

A continuación se muestran los requisitos funcionales de *IJObject* en la siguiente Figura 4.1.

Nombre del caso	*Obtain sets and draw them on Image*
Resumen	El sistema obtiene los conjuntos homogéneos que definen a las imágenes pancromática y multiespectral causa de la ejecución, apoyándose en un algoritmo de segmentación primero, y en otro

	espectral después. Posteriormente, dibuja el contorno de dichos conjuntos sobre una imagen con fondo blanco.
Dependencias	-
Actores	*User, Segmentation Algorithm Process, Spectral Algorithm Process*
Precondiciones	El usuario debe disponer en disco de una imagen pancromática y otra multiespectral (multibanda), pues ambas imágenes son los datos de entrada del programa.
Pos condición	Se habrán generado en los directorios pertinentes, tomando como directorio raíz el directorio elegido por el usuario en el paso 5, los ficheros que *IJObject* es capaz de entender y manejar y que definen los conjuntos descritos por el fichero de texto *Matlab*.
Escenario Principal	1. El usuario aplica el algoritmo de segmentación deseado a la imagen pancromática. 2. a. El sistema solicita al usuario que elija el directorio en disco en el que pretende que se creen los objetos y conjuntos, así como el resto de ficheros de apoyo que el sistema necesita para sus ejecuciones. 3. a. El usuario elige la opción del sistema destinada a la detección de objetos sobre la imagen resultado del paso 1. 4. El sistema procesa la imagen y obtiene los objetos que la definen. 5. El usuario aplica el algoritmo espacial que desee sobre la imagen multiespectral. 6. a. El usuario elige la opción del sistema destinada a la obtención de conjuntos sobre la imagen resultado del paso 4. 7. El sistema procesa la imagen y obtiene los conjuntos. Cuando finaliza, informa del usuario de ello. 8. El sistema presenta una imagen con los contornos de los conjuntos detectados dibujados sobre fondo blanco.
Escenarios Alternativos	2. b. Si el directorio elegido contuviese directorios con el mismo nombre que los que el sistema pretende usar para la generación de sus ficheros, el sistema procedería al borrado de su contenido. 3. b. Si se detecta que a la imagen no se le ha aplicado un algoritmo de segmentación, el sistema informa pertinentemente al usuario y no

	continúa con la detección de objetos. 6. b. Si el sistema detecta que a la imagen no se le ha aplicado un algoritmo espacial, se informa pertinentemente al usuario y no se procede a la detección de conjuntos.
Observaciones	El usuario no debe manipular bajo ningún concepto el contenido de los ficheros que el sistema genere en el directorio elegido para dicho fin.
Requisitos no Funcionales Específicos	Es necesario contar con el espacio suficiente en disco para albergar los ficheros que definen los conjuntos así como diversos ficheros de soporte que el sistema usa para su propio beneficio. Por otro lado, la obtención de conjuntos y objetos es un proceso que conlleva cálculos complejos, luego unas buenas especificaciones *hardware* tendrá beneficios favorables en los tiempos de respuesta.

Nombre del caso	*Import Sets from a Matlab File*
Resumen	Eventualmente, un usuario puede preferir importar los conjuntos homogéneos que definen las imágenes desde un fichero de texto obtenido con la aplicación informática *Matlab*.
Dependencias	-
Actores	*User, Matlab*
Precondiciones	El usuario debe haber generado con la herramienta *Matlab* el fichero de texto que define los conjuntos. Igualmente, debe disponer de las imágenes pancromática y multiespectral usadas para la generación de dicho fichero. El usuario debe disponer de *ImageJ* con el *plugin IJObject* instalado en él.
Pos condición	Se habrán generado en los directorios pertinentes, tomando como directorio raíz el directorio elegido por el usuario en el paso 5, los ficheros que *IJObject* es capaz de entender y manejar y que definen los conjuntos descritos por el fichero de texto *Matlab*.
Escenario	1. El usuario elige la opción del sistema destinada a la importación de conjuntos desde el fichero *Matlab*. El sistema

Principal	solicita al usuario que elija la imagen pancromática usada por *Matlab* para la detección de los conjuntos que se pretenden importar en *IJObject*. 2. El sistema solicita al usuario que elija la imagen multiespectral usada por *Matlab* para la detección de los conjuntos que se pretenden importar en *IJObject*. 3. El sistema solicita al usuario que elija el fichero de texto que *Matlab* proporciona con la definición de los conjuntos detectados para las imágenes pancromática y multiespectral elegidas. 4. a. El sistema solicita al usuario que indique el directorio en disco en el cual *IJObject* creará su estructura de directorios y ficheros con las definiciones de los conjuntos que se pretenden importar, y en el formato que *IJObject* es capaz de manejar y entender. 5. El sistema procesa los ficheros de imagen y de texto que el usuario ha indicado, e importa los conjuntos definidos en ellos. 6. El sistema ofrece un mensaje de confirmación indicando que el proceso de importación ha finalizado correctamente.
Escenarios Alternativos	4. b. Si el directorio elegido contuviese directorios con el mismo nombre que los que el sistema pretende usar para la generación de sus ficheros, el sistema procedería al borrado de su contenido.
Observaciones	El usuario no debe manipular bajo ningún concepto el contenido de los ficheros que el sistema genere en el directorio elegido para dicho fin.
Requisitos no Funcionales Específicos	Es necesario contar con el espacio suficiente en disco para albergar los ficheros que definen los conjuntos.

Nombre del caso	*Draw Attribute's Spacialization on Images*
Resumen	El usuario visualiza los atributos espacializados que desea de entre las opciones posibles, las cuales dependen de las medidas que haya calculado con anterioridad.
Dependencias	Caso de uso *Calculate Measurements*

Actores	*User*
Precondiciones	El usuario tiene que haber ejecutado exitosamente la opción de cálculo de medidas.
Pos condición	Se han presentado imágenes con los atributos espacializados que el usuario ha decidido visualizar.
Escenario Principal	1. El usuario elige la opción del sistema implementada para la espacialización de atributos. 2. a. El sistema solicita al usuario el directorio raíz donde *IJObject* contiene los ficheros que definen los conjuntos, así como el resto de ficheros de respaldo que *IJObject* necesita. 3. El sistema presenta una pantalla con las posibles opciones, dependiendo de las medidas calculadas en el paso de cálculo de medidas y de las imágenes seleccionadas en dicho paso para el cálculo. 4. El sistema genera las imágenes con los atributos espacializados y las presenta por pantalla.
Escenarios Alternativos	2. b. Si el sistema detecta que el directorio elegido no contiene la estructura que espera, o los ficheros que necesita, o bien no se ha ejecutado con anterioridad la opción de cálculo de medidas, informa de ello al usuario y finaliza.
Observaciones	Los ficheros del directorio donde el sistema mantiene su estructura de directorios y ficheros no deben ser manipulados.
Requisitos no Funcionales Específicos	Es necesario contar con el espacio suficiente en disco para albergar los ficheros que contienen las imágenes que se van a generar.

Nombre del caso	*Calculate Measurements*
Resumen	El usuario obtiene las medidas deseadas para cada conjunto, así como sus valores normalizados, calculadas sobre las imágenes pancromática y las diferentes bandas de la imagen multibanda multiespectral, y para todos los conjuntos detectados.

Dependencias	Casos de uso *Import Sets from a Matlab File* y *Obtain sets and draw them on Image*
Actores	*User*
Precondiciones	Bien a través de la opción de importación que el sistema ofrece, o bien a través de la opción que el propio sistema proporciona, se tienen que haber obtenido los conjuntos homogéneos que definen a las imágenes pancromática y multiespectral deseadas. De lo contrario, el sistema informará e impedirá proceder al cálculo de medidas.
Pos condición	El usuario dispondrá de una tabla de medidas y otra tabla de medidas normalizadas, calculadas para todos los conjuntos, según las medidas seleccionadas y las imágenes sobre las que ha querido proceder al cálculo.
Escenario Principal	1. El usuario elige la opción del sistema destinada al cálculo de medidas. 2. a. El sistema solicita al usuario el directorio raíz donde *IJObject* contiene los ficheros que definen los conjuntos, así como el resto de ficheros de respaldo que *IJObject* necesita. 3. El sistema presenta al usuario una pantalla para que seleccione las medidas que desee calcular, así como las imágenes sobre las que quiere hacer el cálculo. Las imágenes que el usuario podrá seleccionar son la pancromática y las distintas bandas que conforman la imagen multibanda. Las medidas que el usuario puede elegir son las siguientes: *area, center of mass, bounding rectangle, shape descriptors, perimeter, mean value, standard deviation, min & max gray value, median, angular second moment, contrast, correlation, inverse difference moment, entropy*. 4. a. El usuario selecciona las imágenes y las medidas deseadas. 5. El sistema procesa todos los datos que tiene, y presenta al usuario la tabla de medidas y la de medidas normalizadas.
Escenarios Alternativos	2. b. Si el sistema detecta que el directorio seleccionado no es válido o no contiene los ficheros que espera, se presenta una ventana de error informando de ello. 4. b. Si el usuario no selecciona al menos una imagen y una medida, se le presenta un mensaje de error informando de ello.

Observaciones	El usuario no debe manipular bajo ningún concepto el contenido de los directorios que el sistema usa para la generación de los ficheros involucrados en el proceso.
Requisitos no Funcionales Específicos	El cálculo de atributos implica un proceso algorítmico relativamente complejo. Dado que el cálculo es sobre cada uno de los conjuntos detectados, y dependiendo de la magnitud de la imagen, el número de conjuntos detectado puede ser considerablemente alto. Por lo tanto, un equipo con muy buenas especificaciones *hardware* es aconsejable para reducir el tiempo de ejecución.

Figura 4.1: Requisitos funcionales de *IJObject*

El Diagrama de Casos de Uso se presenta a continuación en la Figura 4.2:

Figura 4.2: Diagrama de casos de uso para *IJObject*

4.1.3.3. Requisitos no Funcionales

Un Requisito No Funcional es todo requisito que ni describe información a guardar, ni funciones a realizar. Por tanto, una vez descritos en el apartado anterior los distintos requisitos funcionales del sistema, a continuación se enumeran los requisitos no funcionales del sistema:

- El sistema debe ser un sistema multientorno, es decir, ejecutable en cualquier entorno (*Windows*, *UNIX*, *OSX*,…).
- La interfaz del sistema será amigable. No todos los usuarios del sistema serán expertos informáticos.
- El sistema debe ser robusto a errores.
- El sistema debe ofrecer un rendimiento aceptable, con tiempos finitos de ejecución, por lo que se debe invertir tiempo de análisis y desarrollo en la optimización de los algoritmos involucrados en los cálculos.
- Las distintas opciones del sistema deben tener menús de ayuda, con indicaciones precisas para que el usuario tenga toda la información que necesita para poder ejecutar correctamente las opciones del sistema.
- El sistema se programará en lenguaje *JAVA*, y el entregable será un *plugin* de la herramienta de procesamiento de imágenes *ImageJ*.

4.1.4. Análisis

La fase de Análisis se centra en comprender y describir el sistema en el dominio del problema. Se estructuran y se formalizan los requerimientos obtenidos en el modelado de requisitos y se trabaja con abstracciones del mundo real, obteniendo objetos, situaciones o procesos del mundo real.

Una forma de modelar el análisis es mediante diagramas de colaboración, que determinan los objetos que participan en cada caso de uso, y sus interacciones.

Los diagramas de colaboración describen cómo se lleva a cabo y se ejecuta una operación del sistema dentro del contexto de un caso de uso en términos de interacciones de las clases de análisis y de sus objetos. Determina cómo se reparte la responsabilidad asociada a cada operación del sistema entre las distintas clases colaboradoras. Capturan dos aspectos:

- Contexto de la colaboración: Proporciona una descripción de la organización estructural de los objetos involucrados, incluyendo sus relaciones.
- Interacción: Proporciona una descripción de la secuencia de mensajes enviados y recibidos que se intercambian entre los objetos en un formato de grafo o red.

A continuación, se exponen en las Figuras 4.3, 4.4, 4.5, 4.6 y 4.7 los diagramas de colaboración que modelan el sistema *IJObject*:

1. Diagrama de colaboración para la operación *Import Sets*:

Figura 4.3: Diagrama de colaboración para la operación *Import Sets*

2. Diagrama de colaboración para la operación *Calculate Measurements*:

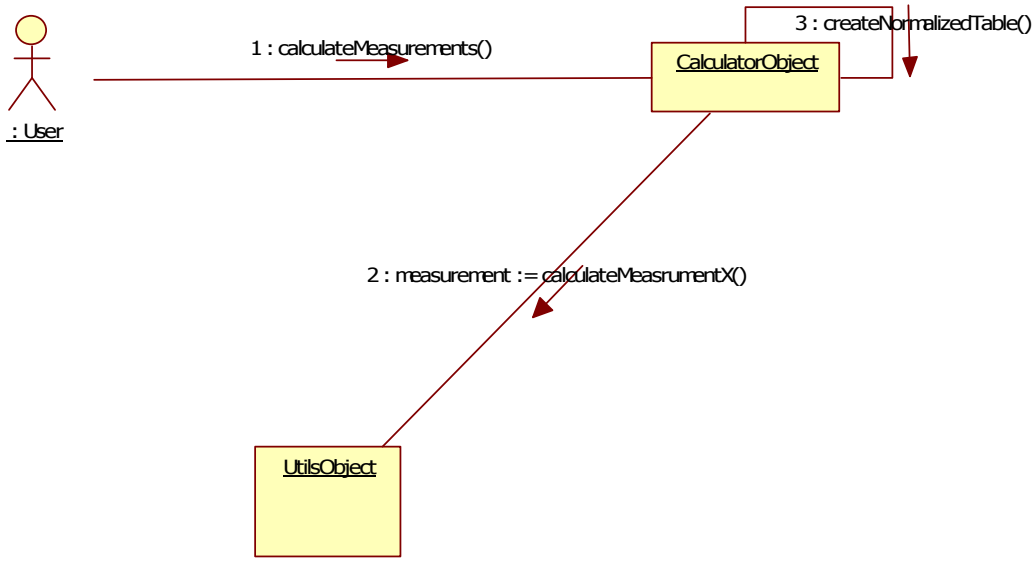


Figura 4.4: Diagrama de colaboración para la operación *Calculate Measurements*

3. Diagrama de colaboración para la operación *Spacialized Attributes*:

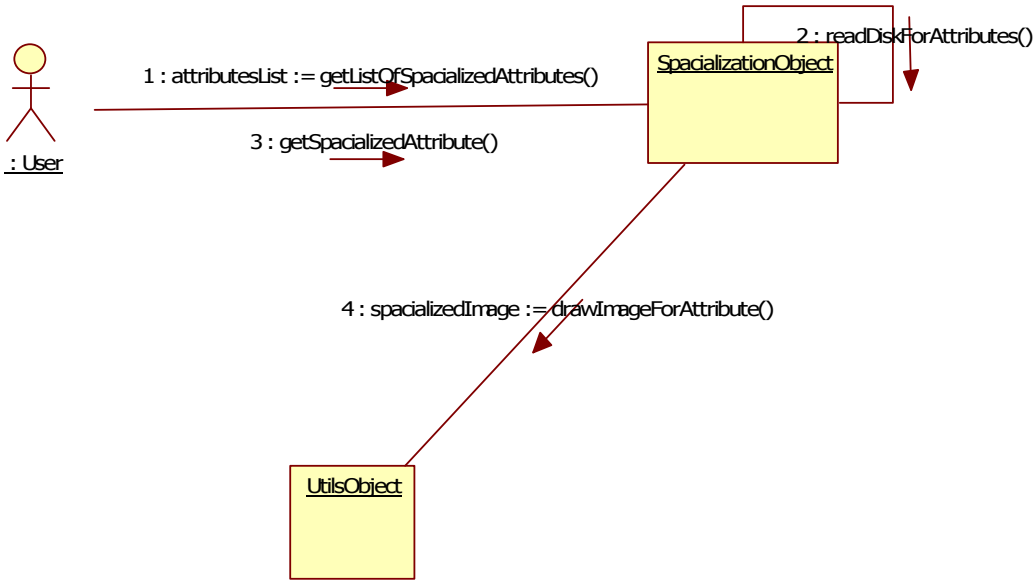


Figura 4.5: Diagrama de colaboración para la operación *Spacialized Attributes*

4. Diagrama de colaboración para la operación *Draw Sets on Image*:

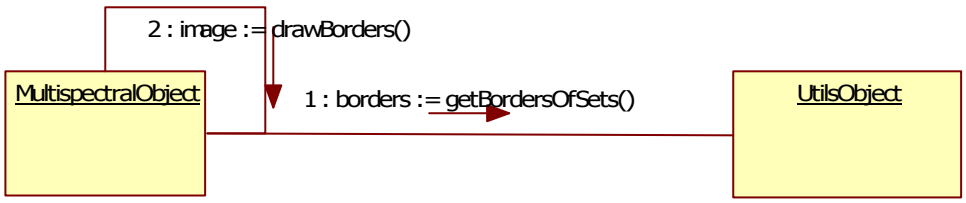


Figura 4.6: Diagrama de colaboración para la operación *Draw Sets on Image*

5. Diagrama de colaboración para la operación *Calculate Sets*:

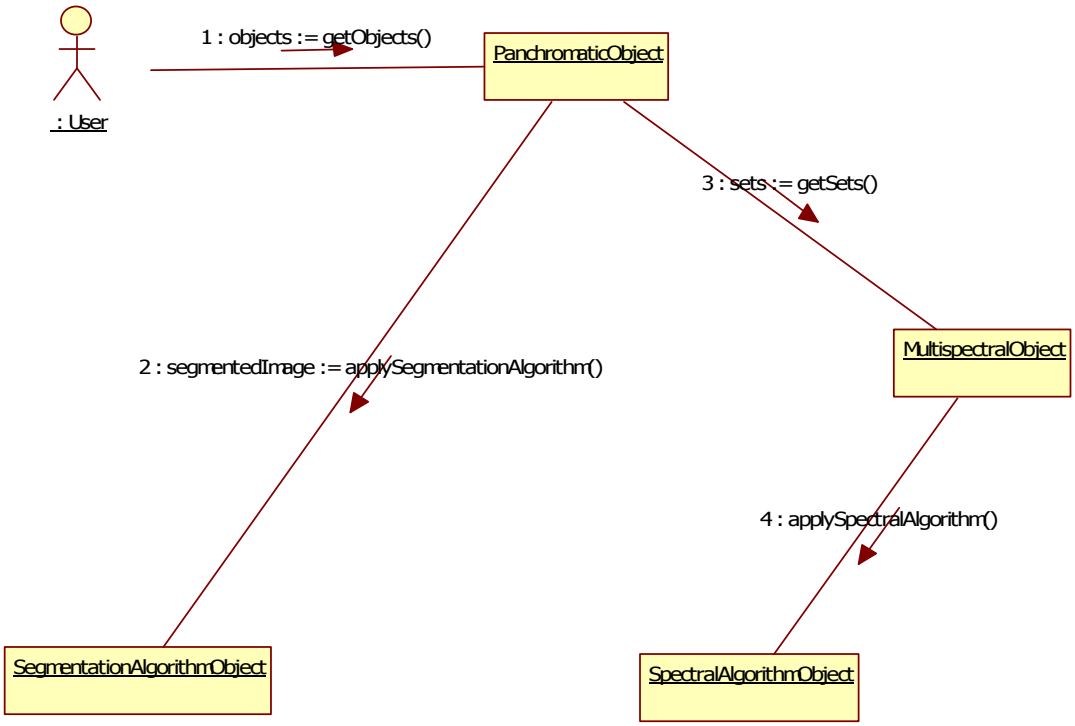


Figura 4.7: Diagrama de colaboración para la operación *Calculate Sets*

4.1.5. Diseño

El diseño es la actividad de aplicar diferentes técnicas y principios con el propósito de definir un sistema con el suficiente detalle para permitir su implementación. Integra el paso del ¿Qué? al ¿Cómo? (ver Figura 4.8).

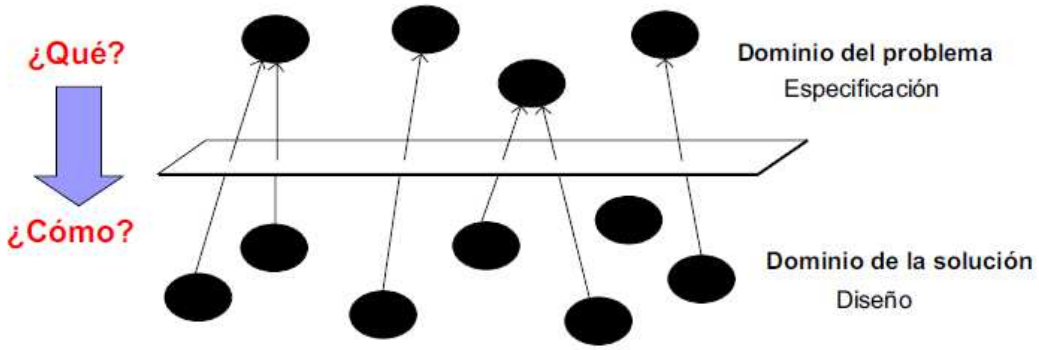


Figura 4.8: Fase de diseño en un proceso *software* [Garv 11-12]

Durante la fase de diseño es primordial definir la Arquitectura del Sistema. La Arquitectura del Sistema es una descripción de los subsistemas de un sistema *software*, sus propiedades y las relaciones entre ellos.

Es necesario definir un modelo arquitectónico por los siguientes motivos:

- Facilita la comprensión a cualquier miembro del equipo.
- Permite que cada miembro del equipo pueda trabajar en los subsistemas de forma individual.
- Prepara al sistema para su extensión.
- Facilita la reutilización del sistema.

La bibliografía recoge los siguientes estilos arquitectónicos:

- Arquitectura multicapa (*Microkernel*) → Modela la interacción entre los subsistemas organizando un sistema en una serie de capas.
- Arquitectura cliente-servidor → Modelo de sistemas distribuidos que muestran cómo datos y procesamiento se distribuyen a lo largo de varios procesadores.
- Arquitectura par a par (*Peer to Peer*) → Generalización de la arquitectura cliente-servidor en la que los subsistemas clientes pueden ser servidores y los subsistemas servidores pueden ser clientes.
- Arquitectura de depósito (*Repository*) → Arquitectura en la que todos los datos se ubican en una base de datos central a la que acceden todos los subsistemas.
- Arquitectura Modelo-Vista-Controlador (*Model-View-Controller*) →Ayuda a separar la capa de interfaz de usuario de otras partes del sistema.

El diseño de *IJObject* cumple con los patrones propuestos por una arquitectura multicapa. Cada capa presta servicios a la capa inmediatamente superior y actúa como cliente sobre las capas más internas. La principal ventaja de este enfoque es, en primer lugar, el hecho de que todas las capas pueden probarse independientemente y, en

segundo lugar, la reutilización, pues otro sistema externo podría pedir servicios a cualquiera de las capas que modelan el sistema. En *IJObject* se encuentran las siguientes capas bien diferenciadas:

- *ImageJ* → Proporciona la capa más baja, y su *API* ofrece múltiples servicios y funcionalidades.
- *IJObject Utils* → Es la capa que apoyándose en la capa anterior, ofrece las distintas funcionalidades que *IJObject* es capaz de ofrecer.
- *IJObject User Interface* → Define la interfaz de usuario y, haciendo uso de las capas más inferiores, ofrece al usuario los resultados esperados.

La principal herramienta usada en la fase de diseños es el diagrama de clases. Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Crean el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro [Wikih]. A continuación se expone en la Figura 4.9 el diagrama de clases diseñado para *IJObject*:

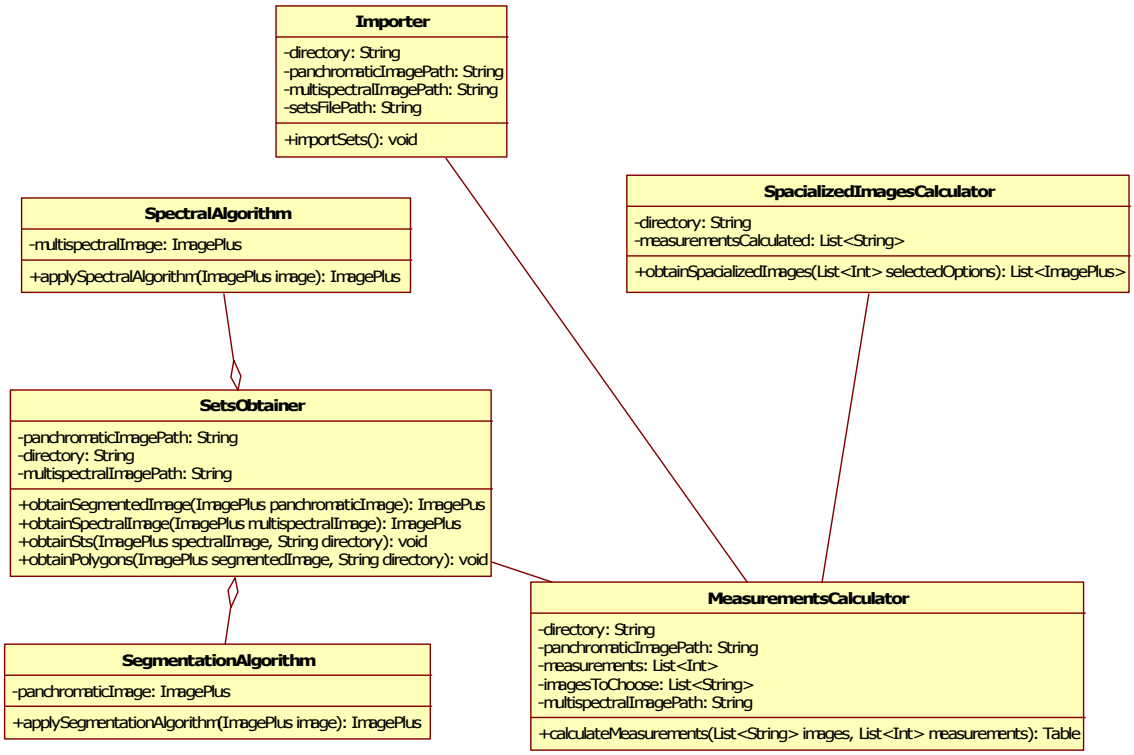


Figura 4.9: Diagrama de clases para *IJObject*

4.1.6. Pruebas

La fase de pruebas en el desarrollo de *software* es aquella cuyo objetivo es probar que el *software* desarrollado no tiene errores, y que el programa realiza lo que se supone debe hacer. De esta forma, al final de la fase de pruebas, el equipo de trabajo puede hacerse una idea objetiva de la calidad del producto. La satisfacción de clientes y usuarios del *software* desarrollado depende, en buena medida, de haber realizado esta etapa correctamente.

Un buen *software testing* se basa en los siguientes principios:

1. Una parte necesaria de un *test* es la definición de los resultados esperados.
2. Un programador debe evitar probar su propio código desarrollado.
3. Una compañía no debe probar sus propios desarrollos.
4. Los resultados de los *tests* deben ser revisados en profundidad.
5. Los *tests* deben incluir entradas inválidas e inesperadas así como las válidas y esperadas.
6. Revisar un programa para verificar que hace lo que se espera que haga supone sólo la mitad de la prueba; la otra mitad consiste en comprobar que no haga lo que no se espera.
7. No planear esfuerzos de pruebas asumiendo que no se encontrarán errores.
8. La probabilidad de encontrar errores en una sección de un programa es proporcional al número de errores ya encontrados en esa sección.
9. El *testing* constituye una tarea creativa e intelectualmente desafiante [ITM].

A lo largo del proceso de análisis, diseño y desarrollo de *IJObject* han sido múltiples y variadas las pruebas realizadas con el fin de asegurar su robustez y calidad. Finalmente, el resultado de dichas pruebas es satisfactorio. La idea subyacente era la de repetir todas las pruebas tras cada cambio, asegurando así que el *software* seguía siendo

robusto a errores y que los resultados obtenidos eran los esperados. A continuación se describen las pruebas más significativas diseñadas para *IJObject*:

4.1.6.1. Pruebas de funcionalidad

- **Detección de objetos y polígonos** → *IJObject* detecta los objetos y polígonos que delimitan la imagen pancromática y exporta todos los puntos del contorno del objeto y del polígono en cuestión, a ficheros de texto. Por otro lado, la detección de objetos también es realizada por *ImageJ*, de forma correcta se supone, pero sin la funcionalidad de exportación en el formato deseado que aporta *IJObject*. La prueba consiste en calcular objetos y polígonos para la imagen con *IJObject* y, a continuación, obtener los objetos mediante la funcionalidad que para ello aporta *ImageJ*. Por último, comprobar que los puntos de los objetos detectados por *IJObject* coinciden con los puntos que *ImageJ* dibuja sobre la imagen resultado de su analizador de objetos.

- **Detección de conjuntos** → *IJObject* detecta los conjuntos resultado de la unión de los polígonos detectados de la imagen pancromática y la imagen multiespectral. La prueba consiste en obtener los conjuntos, en primer lugar, y posteriormente comprobar manualmente que los puntos que conforman los distintos conjuntos efectivamente se corresponden con la superposición de los polígonos de la imagen pancromática y las distintas regiones de la imagen multiespectral. Se trata de un proceso tedioso, pero trivial.

- **Representación del contorno de los conjuntos en una imagen con fondo blanco** → Para representar los conjuntos sobre una imagen ha sido necesario implementar el algoritmo descrito en el apartado correspondiente, dado que la *API* de *ImageJ* no permitía dicha representación teniendo en cuenta que los conjuntos están descritos en ficheros de texto con un formato específico de *IJObject*. La prueba consiste, sencillamente, en comprobar de forma manual que los contornos pintados se corresponden a los puntos contenidos en los diversos ficheros de texto que describen a cada uno de los conjuntos. Vuelve a ser un proceso laborioso, pero trivial.

- **Cálculo de medidas** → Las medidas se calculan independientemente para todos y cada uno de los conjuntos que *IJObject* tenga definidos en ese momento, hayan sido calculados, o bien, importados. Por otro lado, todas las medidas que *IJObject* ofrece al usuario son también calculadas por *ImageJ* o por algún *plugin* de terceros de forma correcta, se supone. Pero con la diferencia de que se aplican sobre toda la imagen y no sobre un conjunto. La prueba consiste en seleccionar una medida y calcularla mediante *IJObject* prestando atención al valor obtenido para un conjunto de forma rectangular. Acto seguido, en *ImageJ* (o mediante el *plugin* de terceros), obtener el valor de dicha medida sobre una imagen que coincida con el conjunto rectangular que se ha observado. Los cálculos deberían coincidir.

- **Espacialización de atributos** → La obtención de imágenes espacializadas por atributos no es más que representar una imagen en la que cada conjunto tiene el valor de gris determinado por el valor normalizado de la medida seleccionada. Para realizar la prueba es tan simple como obtener una imagen espacializada por un atributo cualquiera y, para cada conjunto, comprobar manualmente que el valor de gris usado coincide con el valor de la medida normalizada.

- **Importación de conjuntos desde fichero *Matlab*** → *Matlab* es una herramienta de *software* matemático ampliamente usada en el ámbito del tratamiento de imágenes. La detección de conjuntos homogéneos sobre imágenes pancromáticas y multiespectrales es una funcionalidad abordada por dicha herramienta. El formato con el que *Matlab* exporta dichos conjuntos difiere sensiblemente al de *IJObject*. Para empezar, *Matlab* usa un solo fichero y no especifica las coordenadas del punto en el fichero, como sí hace *IJObject*. La prueba consiste en importar los conjuntos definidos en el fichero *Matlab* y, a continuación, comprobar manualmente que efectivamente los conjuntos definidos por *IJObject* corresponden a los definidos en el fichero *Matlab*.

4.1.6.2. Pruebas de error

- **Directorio de soporte de *IJObject* incorrecto** → *IJObject* siempre solicita al usuario que seleccione el directorio en el que montar su estructura de ficheros o directorios en el caso de un paso inicial, o el directorio en el que ya ha montado su estructura, para una funcionalidad que no pertenezca a un paso inicial (se considera a la importación de conjuntos y a la detección de objetos y polígonos los dos pasos iniciales). La prueba consiste en seleccionar un directorio en el que no se pueda escribir, por no tener permisos de escritura, por ejemplo, para un paso inicial, o un directorio distinto al usado por *IJObject* para un paso distinto a los iniciales. *IJObject* responderá pertinentemente y mediante mensajes de error apropiados.

- **Algoritmo de segmentación no aplicado a imagen pancromática** → Para el cálculo de objetos y polígonos, *IJObject* espera una imagen pancromática procesada por un algoritmo de segmentación (*Otsu* o *Watershed*). La prueba consiste en usar como entrada de la citada opción de *IJObject* una imagen incorrecta. *IJObject* informa pertinentemente de la eventualidad.

- **Algoritmo espectral no aplicado a imagen multibanda** → Para el cálculo de conjuntos, *IJObject* espera una imagen multiespectral procesada por un algoritmo espacial (*K-Means*, *Fuzzy* o *Spatial Fuzzy*). La prueba consiste en usar como entrada de la citada opción de *IJObject* una imagen incorrecta. *IJObject* informa pertinentemente de la eventualidad.

- **Cálculo de conjuntos sin haber generado polígonos previamente** → El cálculo de conjuntos implica haber generado los polígonos previamente. La prueba consiste en intentar generar directamente los conjuntos sin haber generado previamente los polígonos. Un mensaje de error acorde a las circunstancias se presentará al usuario y, obviamente, la ejecución no prosigue.

- **Cálculo de medidas sin haber generado los conjuntos previamente** → Para calcular las medidas, previamente tienen que haberse hallado los conjuntos. La

prueba consiste en intentar calcular medidas sin haber procedido con anterioridad a la detección de los conjuntos. *IJObject* finalizará la ejecución e informará del motivo.

- **Obtención de imágenes de atributos espacializados sin cálculo de medidas previo** → La obtención de las imágenes de atributos espacializados depende de un cálculo previo de medidas. La prueba consiste en intentar presentar imágenes de atributos espacializados sin haber calculado alguna medida previamente. Nuevamente, *IJObject* informará de la eventualidad al usuario de la aplicación.

- **Usar una imagen unibanda como entrada de los algoritmos espectrales** → La prueba consiste en usar una imagen que no es multibanda como entrada de alguno de los algoritmos espectrales. Cualquiera de ellos detectará si la imagen de entrada no es multibanda y reportará el mensaje pertinente.

4.1.6.3. Pruebas de Rendimiento

Se han realizado múltiples pruebas, usando diversas imágenes y de múltiples tamaños. El equipo *hardware* usado se trata de un *Core 2 Duo* a 1,6 *GHz* con 2 *GB* de *RAM* y disco duro de 5400 *rpm*. En todos los casos se han obtenido resultados en tiempos finitos y muy razonables.

4.2. *IJObject*: código fuente

A continuación, se describe punto a punto el código fuente *JAVA* escrito para implementar *IJObject*.

4.2.1. Submenú *2. Spatial Object*

Lo primero que hace es solicitar el directorio donde se quieren almacenar todos los ficheros que se vayan generando. Si no existen, se crean los directorios de Polígonos (*Polygon*), de Conjuntos (*Sets*), de Soporte (*Support*) y de Espacialización (*Spacialization*). En cambio, si existen, se procede a eliminar su contenido.

Como el objetivo realmente es saber qué *píxeles* forman cada objeto, en este apartado se hace una modificación del código del analizador de partículas que viene integrado en *ImageJ* para incluir esta funcionalidad. Del analizador se sacan ciertos *píxeles* que forman el borde del objeto, pero no todos. Por ejemplo, si existen más de dos coordenadas consecutivas en la misma fila o columna, el *plugin* sólo incluía la primera y la última, pero no las intermedias. Teniendo en cuenta este detalle, se procede a obtener todos estos *píxeles* del borde. Si el primer *píxel* es inmediatamente consecutivo al siguiente, sólo estos dos puntos se consideran borde. Pero si entre ellos hay una distancia superior a uno en la coordenada de las *X* o en la coordenada de las *Y*, se recorre un bucle para obtener esos *píxeles* intermedios (además del primero y del último). Se repite esta operación hasta que se recorran todos los puntos del borde que proporciona el analizador de partículas.

Para el cálculo de las coordenadas del interior del objeto, se crea un objeto del tipo *Polygon* en *JAVA* y se le van añadiendo todos los puntos del borde que se obtienen de la forma que se ha descrito. Además, en todo este proceso se van actualizando cuatro variables con las coordenadas *X* e *Y*, máxima y mínima, para todo el borde. A continuación, se crea un bucle que va a recorrer el rectángulo de tamaño el que forman estas cuatro variables, y pregunta por cada punto que está dentro de este rectángulo, si el polígono que se ha creado con los bordes del objeto lo contiene. En ese caso, lo añade al fichero de texto. Por tanto, añade los puntos del borde y todos los del interior. Se puede observar un ejemplo en la Figura 4.10 de cómo quedaría el fichero de texto del polígono completo para el objeto número 20. En rojo están redondeados los puntos del borde. En este caso se está recorriendo el objeto desde la coordenada *X* mínima (257) hasta la *X* máxima (266), y dentro de estos posibles casos se anida un bucle que va desde la *Y* mínima (420) hasta la *Y* máxima (427) como si de una matriz se tratase.

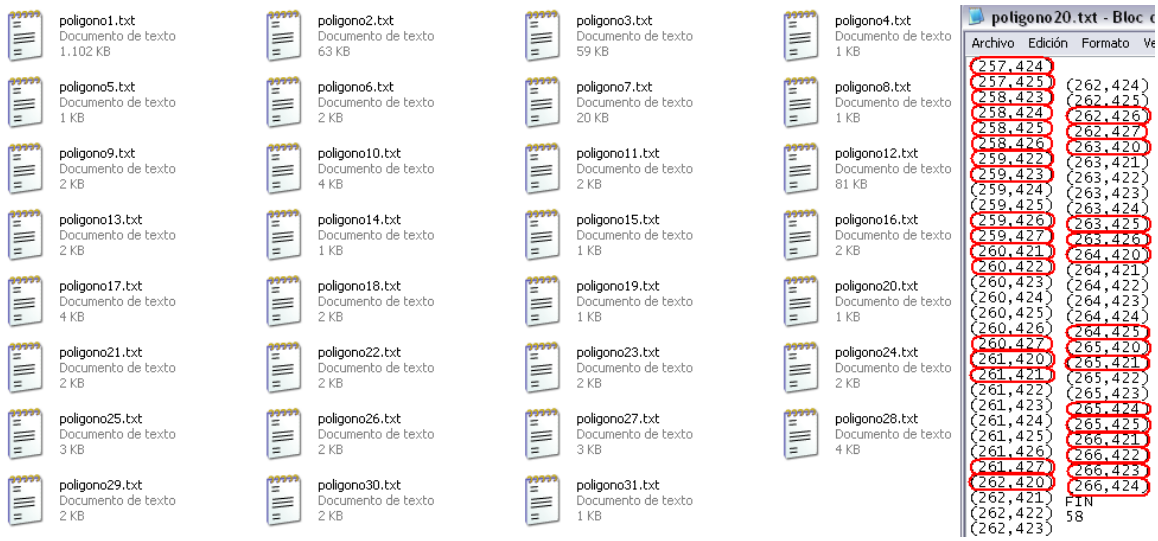


Figura 4.10: Formato de almacenamiento de un objeto en disco

También se recorren todas las imágenes abiertas en ese momento (en principio sólo tendrían que aparecer la pancromática original y la *Watershed* u *Otsu* si todo se ha hecho en el orden correcto) y tomando el directorio de la información del fichero asociado a cada una de ellas, se elige la que lo tenga distinto a “” ó *null*, lo cual significará que es la imagen pancromática original. Obteniendo la ruta en disco de esa imagen, se crea un nuevo fichero (*Path_Panchromatic.txt*) donde se almacenará la ruta, ya que será necesario recuperar dicha imagen en el momento en el que se quieran obtener las medidas (media, desviación estándar…) de ella.

4.2.1.1. Tratamiento de errores

Lo primero que busca el programa es ver si hay alguna imagen abierta sea cual sea, porque si da falso, se muestra por pantalla el siguiente mensaje: *Error capturing open image. Make sure the previous steps have been accomplished.*

A continuación, si existe al menos una imagen abierta, se comprueba que es una imagen segmentada (resultado de aplicar o el algoritmo de *Watershed* o el de *Otsu*) a través del título de la ventana. Si la imagen segmentada no es correcta, se lanza un mensaje con el siguiente texto para informar al usuario: *Error, you have not selected a panchromatic image resulting of applying Otsu or Watershed Algorithm. Apply the desired algorithm and try again.*

4.2.2. Submenú *4. Final Object*

Lo primero que hace es solicitar el directorio donde se quieren almacenar todos los ficheros que se vayan generando. Si no existen, se crean los directorios de Polígonos (*Polygon*), de Conjuntos (*Sets*), de Soporte (*Support*) y de Espacialización (*Spacialization*). En cambio si existen, se procede a eliminar su contenido.

En este punto lo que se hace primero es salvar la imagen multibanda que el usuario ha generado dinámicamente o que ha abierto ya que estaba almacenada en algún directorio (para así agrupar estos dos casos y no perder esta información de ninguna manera), en el directorio de *Support*. Se crea también un nuevo fichero (*Path_Multispectral.txt*) donde se almacenará esa ruta, ya que será necesario recuperar dicha imagen en el momento en el que se quieran obtener las medidas (media, desviación estándar…) de ella.

La integración de los conjuntos de la imagen pancromática y de la multiespectral se lleva a cabo de la siguiente forma. Se recorre cada fichero de texto del directorio *Polygon* generado de la imagen pancromática y que corresponde a cada uno de sus objetos, leyendo cada punto que lo contiene. Por cada punto de cada objeto, se consultará su valor correspondiente en la imagen multibanda segmentada y se irán generando en un nuevo directorio, *Set*, un fichero de texto agrupando por cada valor diferente que se localice en la imagen multibanda, los puntos del objeto de la pancromática que correspondan con ese valor. Por lo que un objeto de la pancromática se dividirá en tantos objetos como esos valores les correspondan en la multibanda. El fichero *polygon1.txt* puede generar por ejemplo los ficheros *set1_2.txt*, *set1_5.txt* y *set1_6.txt*, donde el primer número corresponde al objeto pancromático y el número que está a continuación del subguión corresponde con el objeto multibanda.

Una vez generados los conjuntos resultado de la integración de las dos segmentaciones, se procede a dibujarlos en una imagen. Pero lo que está almacenado en el directorio *Set* no son los bordes de cada conjunto, sino el conjunto entero, por lo que se implementa una función que dado un conjunto, me devuelva únicamente el perímetro del mismo. Para calcular si un *píxel* es borde, es tan simple como comprobar si está

totalmente rodeado por *píxeles* del conjunto. En el caso en el que no esté rodeado por los cuatro sitios (arriba, abajo, izquierda y derecha) se considerará que el *píxel* forma parte del borde. Esto se hará para cada conjunto y el resultado será una imagen donde los bordes de todos los conjuntos estarán coloreados de negro.

4.2.2.1. Tratamiento de errores

Lo primero que hace es ver si existe el fichero *Path_Panchromatic.txt* en el directorio *Support* que ha tenido que ser generado en el punto 2 del menú. De no ser así, la ejecución se detendría mostrando el siguiente texto: *Error, you have not executed previously the Panchromatic Algorithm, thus the objects have not been created.*

Comprueba que existe una imagen multibanda abierta, ya que sino no podría llevarse a cabo este punto, y le aparecería al usuario el siguiente mensaje: *There is not a multiband image open or generated. Please do open one or generate one in order to complete the process.*

También mira que a la imagen multibanda se le ha aplicado alguno de los algoritmos de segmentación de los que se dispone (*K-Means*, *Fuzzy*...) porque sino no se podrían sacar los valores de los *píxeles*, por tanto se comprueba que el título de la ventana de la imagen actual contiene o la palabra *Clusters* o *Gray Scale* tal y como la generan los algoritmos. Si no es así, se obtendría el siguiente texto: *Error, you have not selected an image resulting of applying a multispectral algorithm like REF K-Means, Fuzzy C-Means or Spatial Fuzzy C-Means. Apply the chosen algorithm and try the Final Object option again.*

Cuando va a pintar la imagen que contiene la integración de los conjuntos, primero comprueba que existe el fichero *Path_Panchromatic.txt* en el directorio *Support*, ya que de no ser así se mostraría este mensaje: *You must have executed previously the algorithms IJObject Panchromatic and Multispectral.*

4.2.3. Submenú *5. Measurements*

Lo primero que hace es solicitar el directorio donde están almacenados todos los ficheros que han ido generando y que contiene a los directorios de Polígonos (*Polygon*), de Conjuntos (*Sets*), de Soporte (*Support*) y de Espacialización (*Spacialization*). A través de este directorio se obtienen la imagen pancromática y la imagen multiespectral, ya que de esta última se van a obtener sus diferentes bandas por separado haciendo que cada una sea una imagen distinta.

La ventana que aparece nada más ejecutar este submenú contiene en la parte superior la imagen pancromática y todas las bandas que tenga la imagen multiespectral por separado. A continuación aparecerán todas las medidas texturales, espectrales y de forma, las cuales se pueden obtener de las imágenes previas.

Para cada conjunto, se obtendrá una serie de medidas que elija el usuario para las imágenes que haya seleccionado. Estos resultados se visualizarán en una tabla la cual se almacena automáticamente en formato Excel. También se generará otra tabla con los mismos resultados normalizados, y de igual forma se almacenará (en el directorio que indique el usuario) en formato Excel.

Una vez se tienen estos datos, se elimina el contenido del directorio *Spacialization*, y se almacenan ahí para las imágenes y medidas seleccionadas, su imagen con el valor correspondiente normalizado de cada conjunto como valores de los *píxeles* (además de un fichero de texto que contiene qué medidas y qué imágenes se han elegido). Estas imágenes podrán visualizarse a petición del usuario en el submenú 6.

4.2.3.1. Tratamiento de errores

Para poder ejecutar este submenú es condición necesaria que exista tanto el fichero *Path_Panchromatic.txt* como el fichero *Path.Multispectral.txt* en el directorio *Support*, ya que querrá decir que se han hecho los pasos previos. Si no es así, se mostrará el siguiente mensaje: *You must have executed previously the algorithms IJObject Panchromatic and Multispectral.*

En la ventana de selección, se capturan todos los casos posibles para una correcta ejecución. En el caso de que haya al menos una imagen seleccionada pero no haya ninguna medida marcada, aparecerá este texto: *You must select at least one Measurement in order to proceed.* Si no se selecciona ninguna imagen pero sí alguna de las medidas, el mensaje será: *You must select at least one Image in order to proceed.* Por último, si no se selecciona ninguna medida ni ninguna imagen, el texto será: *You must select at least one Measurement and one Image in order to proceed.*

4.2.4. Submenú *6. Attributes' Spacialization*

Lo primero que hace es solicitar el directorio donde están almacenados todos los ficheros que han ido generando y que contiene a los directorios de Polígonos (*Polygon*), de Conjuntos (*Sets*), de Soporte (*Support*) y de Espacialización (*Spacialization*). A través de este directorio, se obtiene el fichero de texto que se generó en el directorio *Spacialization* donde se encontraban las imágenes y medidas que se habían hallado en el submenú anterior, y con este fichero se genera una ventana que va a dar la opción al usuario a seleccionar qué imágenes quiere visualizar (las cuales están almacenadas en el directorio *Spacialization* por el submenú 5).

4.2.4.1. Tratamiento de errores

Para poder ejecutar este submenú es condición necesaria que exista tanto el fichero *Path_Panchromatic.txt* como el fichero *Path.Multispectral.txt* en el directorio *Support*, ya que querrá decir que se han hecho los pasos previos. Si no es así, se mostrará el siguiente mensaje: *You must have executed previously the algorithms IJObject Panchromatic and Multispectral.*

Si el fichero que se genera en el submenú 5 en el directorio *Spacialization* donde se encuentran las imágenes y medidas seleccionadas por el usuario está corrupto (vacío) entonces se mostrará un aviso al usuario: *The Spacialization File Info is corrupt.*

Si no se selecciona la visualización de ninguna de las imágenes en la ventana de selección, aparece el mensaje *Error, you must select at least one Attribute.*

4.2.5. Submenú *7. Import Sets…*

Lo primero que hace es solicitar la imagen pancromática, a continuación la imagen multiespectral y, por último, el fichero de texto generado por *Matlab* que contiene los conjuntos resultados de la integración de las dos imágenes, en su nomenclatura. Una vez se introducen estos archivos, se solicita al usuario el directorio donde se quieren almacenar todos los ficheros que se vayan generando. Si no existen, se crean los directorios de Polígonos (*Polygon*), de Conjuntos (*Sets*), de Soporte (*Support*) y de Espacialización (*Spacialization*). En cambio si existen, se procede a eliminar su contenido.

Con la ruta de la imagen pancromática y de la imagen multiespectral que el usuario ha introducido en el comienzo, se rellenan el fichero *Path_Panchromatic.txt* y el fichero *Path_Multispectral.txt*, respectivamente, en el directorio *Support.*

El fichero generado por *Matlab* no contiene las dimensiones de la imagen por lo que se establecen obteniendo el ancho y largo de la imagen pancromática. Una vez se tiene este dato, se procede a leer el fichero *Matlab* y generar los conjuntos acorde como trabaja este *plugin*, dejándolo con la misma estructura para poder así obtener sus medidas.

4.2.5.1. Tratamiento de errores

Comprueba que el fichero de texto generado por *Matlab* existe, es decir, el usuario lo ha elegido oportunamente en el momento en el que el programa se lo ha pedido. Si no, se muestra la ventana de texto *There has been some unhandled Error trying to read the file that describes the Sets.*

5. RESULTADOS

5.1. Generación de objetos en la imagen pancromática y coeficientes wavelet

Una vez segmentada la imagen por cada uno de los algoritmos estudiados, el siguiente paso es la **extracción de objetos**. Para ello, se va a utilizar una utilidad que proporciona *ImageJ* para analizar partículas.

Analyze Particles cuenta y mide objetos en imágenes binarias. Funciona mediante el escaneo de una imagen o de una parte seleccionada de ella hasta que encuentra el borde del objeto.

A continuación (en la Figura 5.1) se presenta el cuadro de diálogo que *Analyze Particles* presenta al usuario.

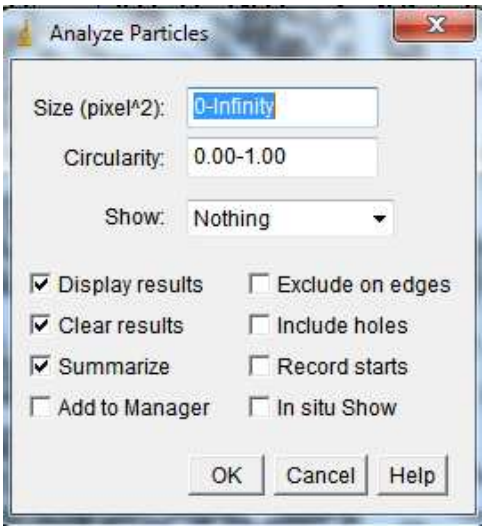


Figura 5.1: Ventana de selección de *Analyze Particles*

Las partículas fuera del rango especificado en el campo *Size* son ignoradas. Si en lugar de un rango se introduce un valor simple en el campo *Size*, aquellas partículas cuyo valor sea menor al especificado serán ignoradas. Partículas con circularidad fuera del rango especificado en el campo *Circularity* también son ignoradas. La fórmula para la circularidad es *4PI(área/perímetro2)*. Un valor de *1.0* indica un círculo perfecto.

Con respecto a la opción *Show*, si se selecciona *Outlines*, *ImageJ* abrirá una imagen conteniendo los objetos detectados dibujados de forma numerada. Si se seleccionase *Masks* se mostrarían los objetos rellenos (no sólo contorno) y si se eligiese *Ellipses* se mostrarían las elipses que mejor se ajusten a los objetos encontrados. Lo que se acaba de describir se ejemplifica claramente en la imagen que se presenta a continuación en la Figura 5.2:

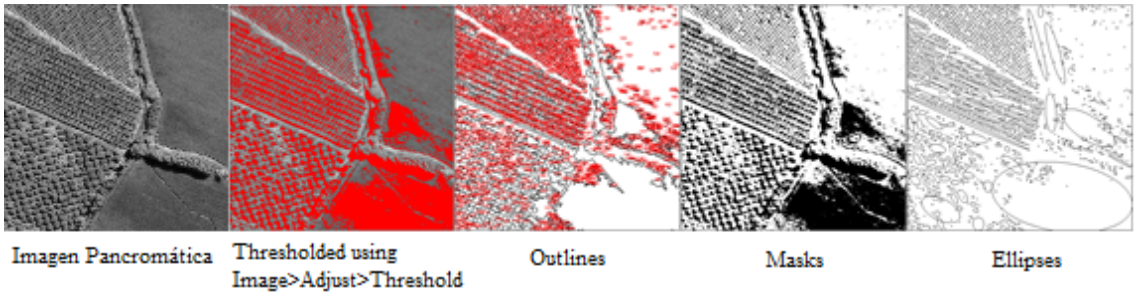


Figura 5.2: Resultado de aplicar a una imagen (pancromática) las distintas opciones del menú *Show* de *Analyze Particles*

Si se activa la opción *Display results* se mostrarán las medidas para cada partícula en la ventana *Results*. Si se selecciona *Clear Results,* se limpiará la tabla de la ventana *Results* antes de presentar los resultados. Si se seleccionase *Summarize*, se mostrará en una ventana separada el número total de partículas, el área total de las partículas, el tamaño medio de las partículas y la fracción de área. Por otro lado, si el usuario se decantase por seleccionar *Exclude on Edges,* se ignorarán las partículas que estén en contacto con el borde de la imagen o la sección seleccionada para el estudio.

Si se selecciona *Include Holes,* se incluyen huecos interiores. Si se desactiva esta opción, se excluyen huecos interiores y se miden partículas incluidas en otras partículas. Se puede comprobar en la Figura 5.3 la diferencia de seleccionar o no estas opciones.

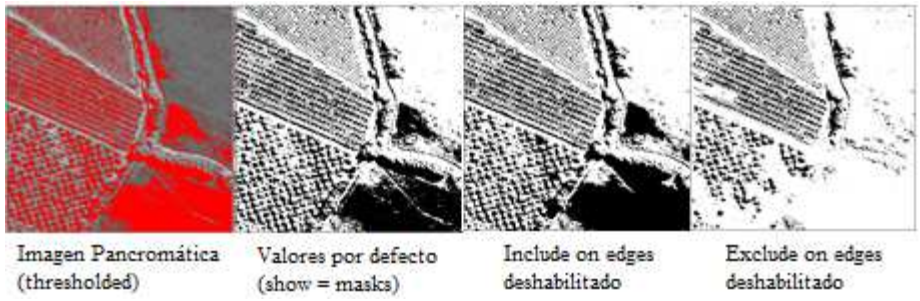


Figura 5.3: Resultado de aplicar unas opciones u otras a una imagen

5.2. Algoritmo de *Otsu* y de *Watershed*: segmentación y generación de segmentos texturales

En este apartado se va a ver cómo funciona la metodología descrita con estos algoritmos de segmentación mediante ejemplos prácticos. La prueba se va a hacer con los coeficientes 1, 2 y 3. Para ello, se abre cada una de estas imágenes y se selecciona este algoritmo en el menú: *Plugins* → *IJObject* → 1. *Segmentation* → *Otsu Thresholding for IJObject;* o bien este otro: *Plugins* → *IJObject* → 1. *Segmentation* → *Watershed Algorithm for IJObject,* con el objeto de segmentar la imagen original.

El algoritmo de *Otsu* genera unas imágenes muy parecidas a las originales. En cambio, la visualización del resultado que ofrece *Watershed* difiere de la imagen original, notándose claramente esa localización de objetos diferentes entre sí. La comparativa de resultados de ambos algoritmos se pueden observar en la Figura 5.4.

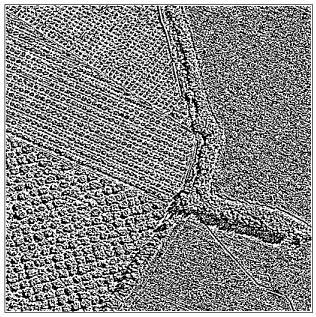
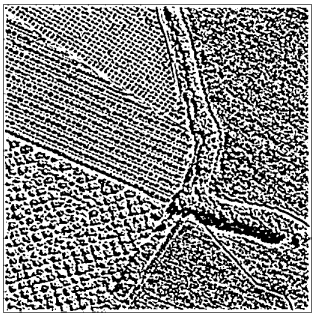
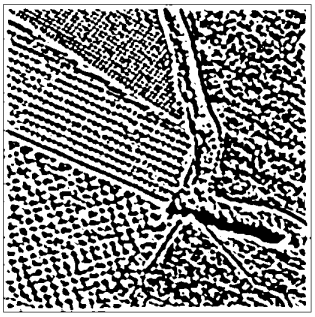
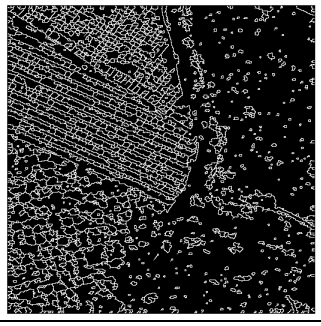
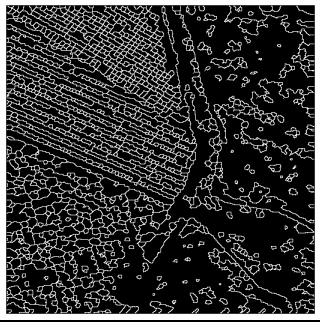
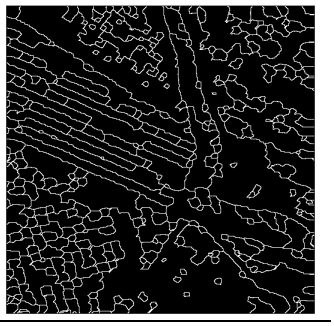
Algoritmo	Coef. *Wavelet* 1	Coef. *Wavelet* 2	Coef. *Wavelet* 3
Otsu			
Watershed			

Figura 5.4: Tabla resumen imágenes resultado de segmentación por algoritmos

Una vez se tiene la imagen segmentada, con el fin de extraer los objetos de la misma, se selecciona en el menú: *Plugins* → *IJObject* → 2. *Spatial Object*. A continuación, en la Figura 5.5 se ve el resultado para las correspondientes imágenes y algoritmos de la figura anterior.

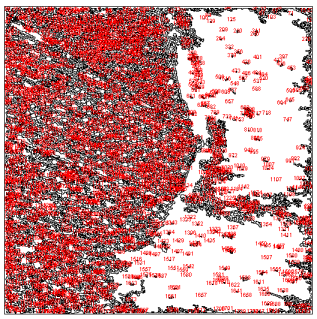
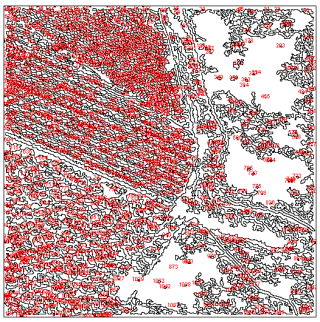
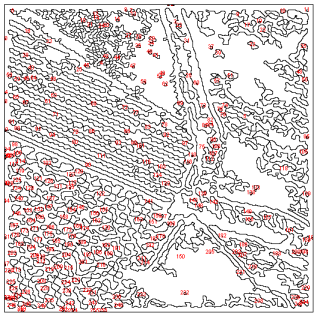
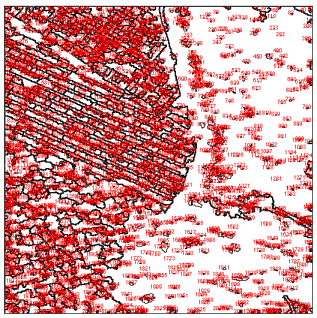
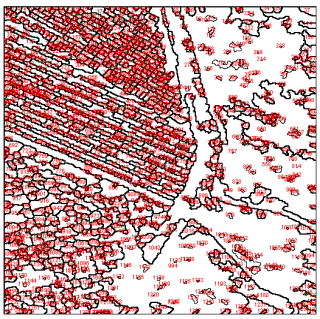
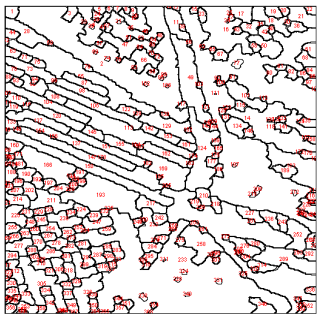
Algoritmo	**Coef. *Wavelet* 1**	**Coef. *Wavelet* 2**	**Coef. *Wavelet* 3**
Otsu			
Watershed			

Figura 5.5: Tabla resumen imágenes resultados de analizar objetos según algoritmos

Complementando a esta última figura, se añade la Figura 5.6 donde se puede ver el número de objetos que ha detectado cada algoritmo para cada uno de los anteriores coeficientes.

Algoritmo	**Coef.*Wavelet* 1**	**Coef.*Wavelet* 2**	**Coef.*Wavelet* 3**
Otsu	1733 objetos	1109 objetos	256 objetos
Watershed	2051 objetos	1281 objetos	368 objetos

Figura 5.6: Tabla resumen número de objetos por algoritmos

En base a estos resultados que se observan en la Figura 5.5 y 5.6, la diferencia es bastante pequeña entre estos dos algoritmos. *Watershed* detecta algún objeto más al considerar que algún grupo en *Otsu* no es tan homogéneo y tiene que dividirse. Pero en esencia, son tremendamente parecidos.

5.3. Algoritmo de *REF K-Means, Fuzzy C-Means* y *Spatial C-Means*: generación de segmentos espectrales

En este apartado se va a ver cómo funciona la metodología descrita con estos algoritmos de clasificación mediante ejemplos prácticos. La prueba se va a hacer con la imagen multiespectral de la Figura 2.3. Para ello, se abre esta imagen y se selecciona este algoritmo en el menú: *Plugins* → *IJObject* → 3. *Spectral Object*→ *REF K-Means Clustering for IJObject;* o bien: *Plugins* → *IJObject* → 3. *Spectral Object*→ *Fuzzy C-Means Clustering for IJObject;* o bien: *Plugins* → *IJObject* → 3. *Spectral Object*→ *Spatial Fuzzy C-Means Clustering for IJObject* con el objeto de segmentar la imagen original.

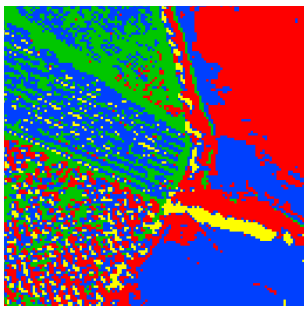
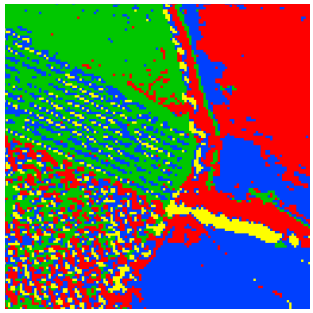
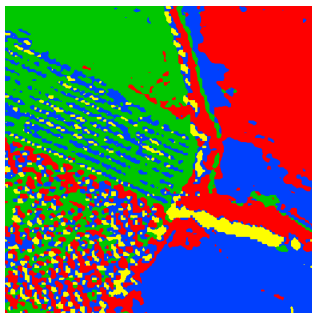
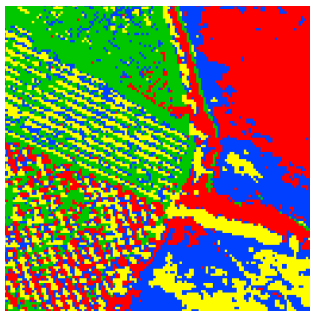
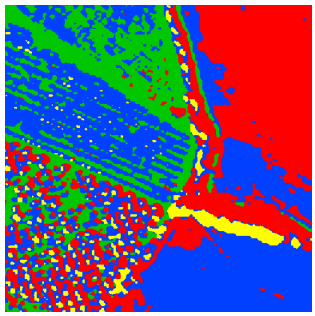
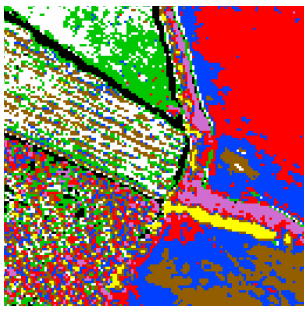
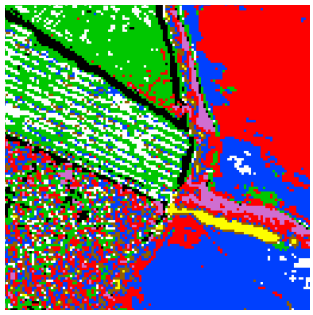
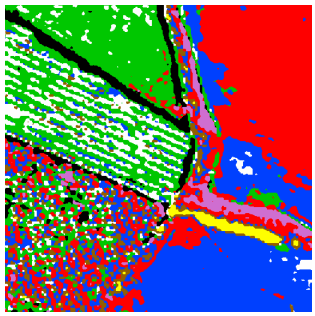
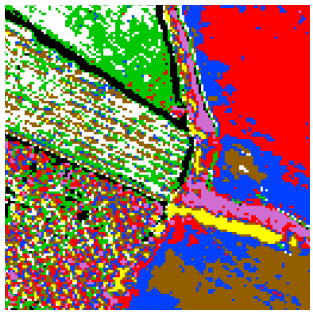
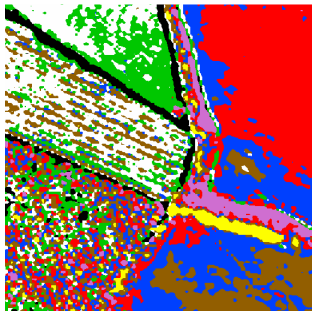
Algoritmo	***REF K-Means***	***Fuzzy C-Means***	***Spatial C-Means***
4 Clusters- 1 iteración			
4 Clusters- 100 iteraciones	Es sólo una iteración		
8 Clusters- 1 iteración			
8 Clusters- 100 iteraciones	Es sólo una iteración		

Figura 5.7: Imágenes clasificadas con 4 y 8 *clusters*, para una y 100 iteraciones, con los algoritmos *REF K-Means, Fuzzy C-Means* y *Spatial C-Means*

Se pueden visualizar y comparar los resultados para los 3 algoritmos en la Figura 5.7, alterando el número de clases y el de iteraciones. Nótese que para todos los resultados de *Spatial C-Means* se ha seleccionado en la ventana de parámetros, un valor de 10.0 para la función peso (*Spatial Function Weight*), con el objetivo de que estos resultados difieran de los del algoritmo *Fuzzy C-Means*. Se ha comprobado que para valores superiores a 200, la imagen resultado pierde propiedades y la segmentación dista mucho de la realidad. Con valores de hasta 200, la imagen resultado es muy parecida a la original. En cualquier caso, la diferencia de aumentar este valor radica en que zonas de poca extensión que apenas contienen *píxeles* y que se encuentran rodeadas de grandes áreas, acaban formando parte de estas zonas amplias y desaparecen como elemento aislado.

Probando para 4 *clusters* y 1 iteración, *REF K-Means* es el que mejor segmenta la zona superior de cultivo. En cambio, no ocurre así con la zona central de plantación, la cual está más conseguida en los casos de *Fuzzy C-Means* y *Spatial C-Means*. Este último perfecciona más los resultados al variar el coeficiente de la función de peso (como se ha indicado previamente) y los objetos quedan más reales al tener un aspecto más redondeado y no pixelado. Para 100 iteraciones, el *Fuzzy C-Means* segmenta mejor que el *Spatial C-Means* ya que este último deja de reconocer objetos más pequeños que pasan a formar parte de zonas amplias predominantes que los acaban incorporando.

En el caso de 8 *clusters* y 1 iteración, el algoritmo que más afina en cuanto a detectar objetos por pequeñas zonas que ocupen, es el *REF K-Means*. Pero como pasa en el caso de 4 *clusters*, el *Spatial C-Means* mejora la visualización de todos los objetos encontrados mostrando una visión buena y global sin entrar en demasiado detalle. Para 100 iteraciones, el *Spatial C-Means* clasifica mejor que el *Fuzzy C-Means* ya que prácticamente identifica los mismos objetos (grandes y pequeños) pero además los dibuja de forma muy parecida a cómo están en la imagen original.

5.4. Integración de segmentos texturales y espectrales

Una vez se obtiene la segmentación de la imagen pancromática y de la imagen multiespectral, el siguiente paso es integrar ambas. Para ello, se va a ver cómo quedaría esta fusión segmentando primero la imagen multiespectral con el algoritmo *Fuzzy C-Means* para 4 *clusters* con 100 iteraciones; y después con el algoritmo *Spatial C-Means* para 8 *clusters* con 100 iteraciones y 10 en la función de peso (ver Figura 5.7). La imagen pancromática, en ambos casos, será el resultado de aplicarle el algoritmo tanto de *Otsu* como de *Watershed*. Se va a realizar este proceso con los 3 coeficientes.

Con la imagen pancromática y multiespectral ya segmentadas y abiertas, se selecciona en el menú: *Plugins* → *IJObject* → 4. *Final Object*. Automáticamente después de hacer los cálculos oportunos, se genera la imagen resultado de esta integración, además de generar los conjuntos oportunos en el directorio correspondiente.

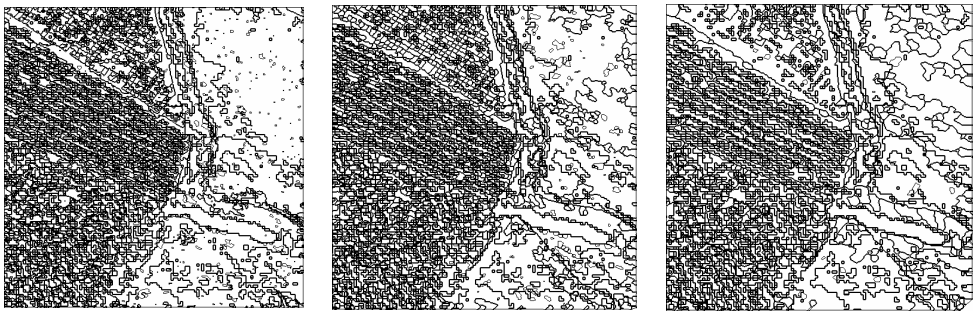
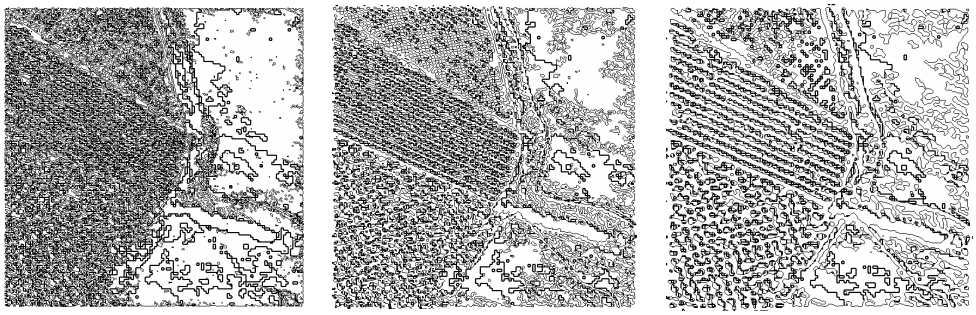
Algoritmo	**Coef.*Wavelet* 1**	**Coef.*Wavelet* 2**	**Coef.*Wavelet* 3**
Watershed			
Otsu			

Figura 5.8: Imágenes resultado de la integración de la segmentación textural y espectral utilizando la clasificación de la imagen multiespectral con 4 *clusters* y 100 iteraciones, con el algoritmo *Fuzzy C-Means* (ver Figura 5.7)

En la Figura 5.8 se pueden visualizar los resultados clasificando la imagen multiespectral con 4 *clusters* y 100 iteraciones por *Fuzzy C-Means*. Con *Otsu* parece que las zonas de cultivo estén más limpias en cuanto a que las hileras de vegetación están mejor definidas, mientras que *Watershed* añade algo más de ruido y se ve todo un poco enmarañado. Esto se debe a que *Watershed* identifica más objetos de los que realmente hay. Se puede ver esta diferencia en la siguiente Figura 5.9.

Algoritmo	**Coef.*Wavelet* 1**	**Coef.*Wavelet* 2**	**Coef.*Wavelet* 3**
Watershed	3833 conjuntos	2793 conjuntos	920 conjuntos
Otsu	3147 conjuntos	2282 conjuntos	658 conjuntos

Figura 5.9: Tabla resumen número de conjuntos por algoritmos de segmentación de la imagen pancromática para el caso de clasificar la imagen multiespectral con 4 *clusters* y 100 iteraciones, con el algoritmo *Fuzzy C-Means*

En la Figura 5.10 se pueden visualizar los resultados clasificando la imagen multiespectral con 8 *clusters* y 100 iteraciones por *Spatial C-Means*. Ocurre lo mismo que en el caso anterior: con *Otsu* parece que las zonas de cultivo estén más limpias en cuanto a que las hileras de vegetación están mejor definidas, mientras que *Watershed* añade algo más de ruido y se ve todo un poco enmarañado.

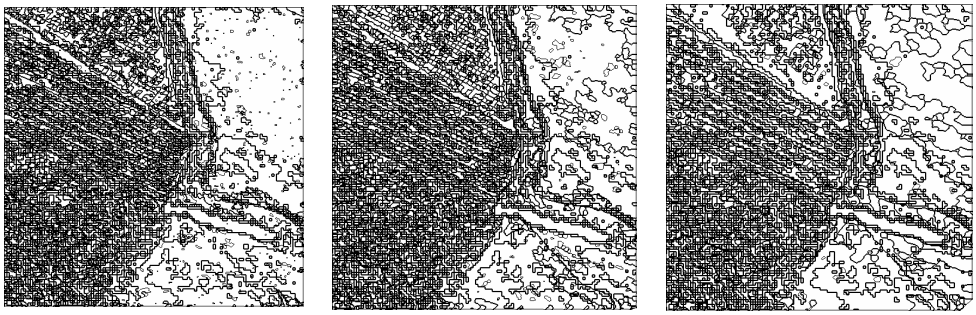
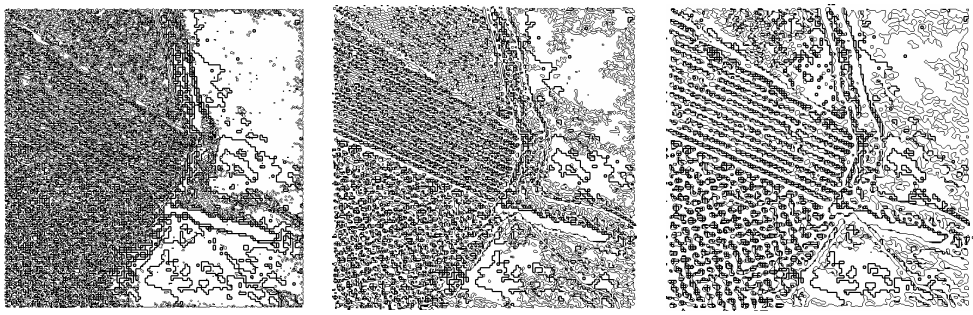
Algoritmo	**Coef.*Wavelet* 1**	**Coef.*Wavelet* 2**	**Coef.*Wavelet* 3**
Watershed			
Otsu			

Figura 5.10: Imágenes resultado de la integración de la segmentación textural y espectral utilizando la clasificación de la imagen multiespectral con 8 *clusters* y 100 iteraciones, con el algoritmo *Spatial C-Means* (ver Figura 5.7)

La diferencia con el anterior caso radica en que ahora hay 4 *clusters* más, por lo que el número de conjuntos identificados va a ser superior en ambos algoritmos (unos 500-700). Se puede ver el número de conjuntos que saldrían para este caso con 8 *clusters*.

Algoritmo	**Coef.*Wavelet* 1**	**Coef.*Wavelet* 2**	**Coef.*Wavelet* 3**
Watershed	4503 conjuntos	3436 conjuntos	1290 conjuntos
Otsu	3671 conjuntos	2769 conjuntos	914 conjuntos

Figura 5.11: Tabla resumen número de conjuntos por algoritmos de segmentación de la imagen pancromática para el caso de clasificar la imagen multiespectral con 8 *clusters* y 100 iteraciones, con el algoritmo *Spatial C-Means*

5.5. Atributos correspondientes a los segmentos resultantes

En este punto se van a mostrar qué medidas se obtienen para los 10 primeros conjuntos (no se reflejan para todos debido a que sería muy extenso). Para llevarlo a cabo, una vez se ha hecho la integración de los segmentos texturales y espectrales (se parte del coeficiente *Wavelet* 1 segmentado por *Otsu* y de la clasificación de la imagen multiespectral con el algoritmo de *Spatial C-Means* para 8 *clusters* con 100 iteraciones y 10 de peso, ver Figuras 5.10 y 5.11), se selecciona en el menú: *Plugins* → *IJObject* → 5. *Measurements*. Una ventana da la posibilidad de poder elegir de qué imagen o imágenes (la imagen pancromática y cada una de las bandas por separado de la imagen multiespectral) se desean sacar las medidas texturales, espectrales o de forma. Se ha escogido, en este caso, todos los atributos correspondientes a la imagen pancromática (para las medidas texturales se ha elegido un tamaño de *step* en *píxeles* a 1 y la dirección a *average*). En la Figura 5.12 se observa una captura de pantalla de la aplicación para las medidas, así como sus valores normalizados, contemplando únicamente los 10 primeros conjuntos.

	Set	Area	Bx	By	Width	Height	XM	YM	Circularity	Roundness	Aspect Ratio	Perimeter
1	set1_4	4	7	1	2	2	7.667	1.667	0.785	1.000	1	8
2	set2_0	28	21	1	8	5	24.976	3.553	0.521	0.438	1.600	26
3	set3_0	13	52	1	5	3	54.286	2.286	0.638	0.520	1.667	16
4	set4_0	380	47	1	48	20	72.502	11.038	0.054	0.165	2.400	297
5	set4_4	29	48	2	21	15	63.455	13.636	0.356	0.066	1.400	32
6	set5_0	4	92	1	2	2	92.667	1.667	0.785	1.000	1	8
7	set6_0	20	129	1	5	5	131.405	3.544	0.628	0.800	1	20
8	set6_4	6	127	1	2	4	127.500	3.500	0.524	1.500	0.500	12
9	set7_0	356	140	1	59	17	169.314	8.356	0.068	0.102	3.471	256
10	set7_3	23	149	13	8	7	152.375	16.500	0.369	0.359	1.143	28

Mean Grey (Panchromatic)	Min Grey (Panchromatic)	Max Grey (Panchromatic)	Median (Panchromatic)	Std Dev (Panchromatic)
191	0	255	255	127.500
106	0	255	0	125.179
137	0	255	255	132.313
108	0	255	0	124.774
96	0	255	0	125.920
191	0	255	255	127.500
111	0	255	0	127.521
85	0	255	0	131.681
103	0	255	0	124.351
88	0	255	0	124.181

Angular SM (Panchromatic)	Contrast (Panchromatic)	Correlation (Panchromatic)	Inverse DM (Panchromatic)	Entropy (Panchromatic)
0.297	40640.625	-1.968E-5	0.375	0
0.280	21951.170	1.611E-5	0.572	0
0.299	27510.577	8.493E-6	0.577	0
0.280	26570.585	6.261E-6	0.558	0
0.352	24104.095	1.005E-5	0.629	0
0.297	40640.625	-1.968E-5	0.375	0
0.279	27152.550	4.421E-6	0.525	0
0.427	29803.125	-2.656E-5	0.542	0
0.286	24395.592	1.179E-5	0.598	0
0.396	26151.359	-3.227E-6	0.598	0

	Set	Mean Grey (Panchromatic)	Min Grey (Panchromatic)	Max Grey (Panchromatic)	Median (Panchromatic)	Std Dev (Panchromatic)
1	set1_4	191.000	NaN	255.000	255.000	180.312
2	set2_0	106.000	NaN	255.000	0.000	177.029
3	set3_0	137.000	NaN	255.000	255.000	187.119
4	set4_0	108.000	NaN	255.000	0.000	176.457
5	set4_4	96.000	NaN	255.000	0.000	178.078
6	set5_0	191.000	NaN	255.000	255.000	180.312
7	set6_0	111.000	NaN	255.000	0.000	180.342
8	set6_4	85.000	NaN	255.000	0.000	186.226
9	set7_0	103.000	NaN	255.000	0.000	175.859
10	set7_3	88.000	NaN	255.000	0.000	175.619

Angular SM (Panchromatic)	Contrast (Panchromatic)	Correlation (Panchromatic)	Inverse DM (Panchromatic)	Entropy (Panchromatic)
53.291	159.375	252.025	95.625	NaN
48.351	86.083	253.145	145.802	NaN
53.848	107.885	252.906	147.115	NaN
48.486	104.198	252.836	142.266	NaN
69.051	94.526	252.955	160.474	NaN
53.291	159.375	252.025	95.625	NaN
48.271	106.481	252.779	133.879	NaN
90.645	116.875	251.809	138.125	NaN
50.185	95.669	253.009	152.609	NaN
81.804	102.554	252.539	152.446	NaN

Figura 5.12: Tabla con el resultado de las medidas (3 tablas superiores) y sus valores normalizados (2 tablas inferiores) para los 10 primeros conjuntos resultado de la integración de segmentar el coeficiente *wavelet* 1 por *Otsu* con la imagen multiespectral clasificada con *Spatial C-Means* con 8 *clusters*, 100 iteraciones y 10 de peso

Hay en un par de atributos (*entropy* y *Min Grey*, normalizados) en los que no se obtiene el valor para la medida normalizada. La explicación es sencilla. Para cálcular cada valor normalizado se usa la siguiente fórmula:

$$valor\ normalizado = \frac{valor\ conjunto\ actual - valor\ mínimo}{valor\ máximo - valor\ mínimo} \cdot 255$$

Donde los valores máximos y mínimos son los valores máximos y mínimo del atributo en cuestión, considerando los valores de todos los conjuntos.

Para este caso concreto, para todos y cada uno de los conjuntos, el valor *Min Grey* es 0. Por tanto, el cálculo para un conjunto concreto queda como se expone a continuación:

$$valor\ normalizado = \frac{0 - 0}{0 - 0} \cdot 255$$

ImageJ devuelve en la tabla de valores normalizados *NaN*, es decir, *Not a Number*. El problema es que *ImageJ* intenta dividir por 0, como en la fórmula anterior se puede ver.

5.6. Imágenes que representan los atributos obtenidos

Una vez se han obtenido las medidas, el último paso es visualizar para cada imagen y para cada medida que se haya seleccionado en el punto anterior del menú, su correspondiente imagen fruto de colorear cada conjunto con el valor normalizado para cada uno de ellos. Se selecciona en el menú: *Plugins* → *IJObject* → 6. *Attributes' Spacialization*. Para el anterior caso (pero ahora se tienen en cuenta todos los conjuntos, no sólo uno), se van a presentar en la Figura 5.13 las imágenes asociadas.

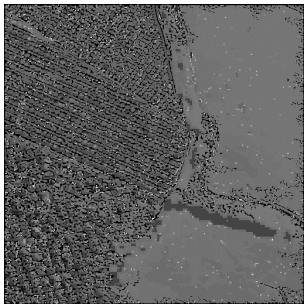
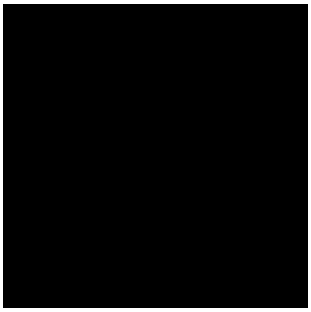
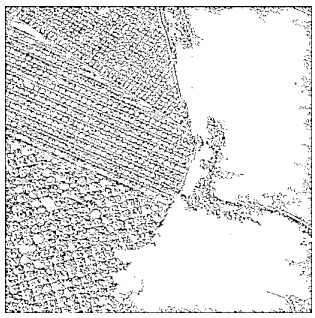
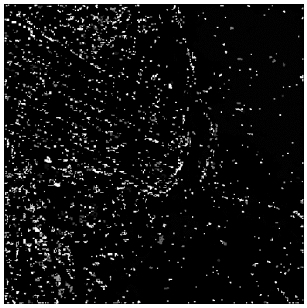
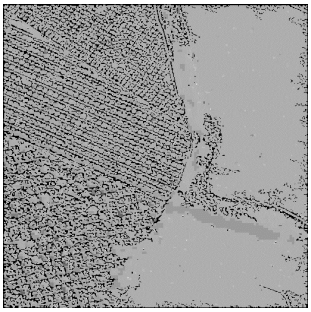
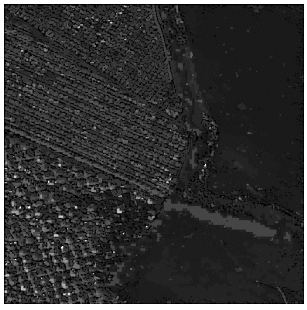
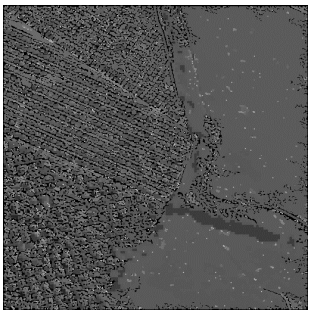
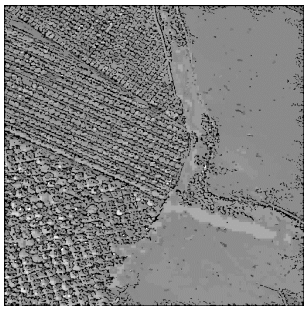
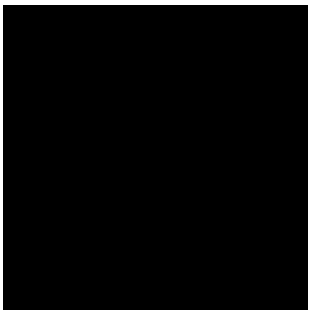
	Mean Grey	***Min Grey***	***Max Grey***
Medidas Espectrales			
	Median	***Standard Deviation***	
Medidas Espectrales			
	Angular SM	***Contrast***	***Correlation***
Medidas Texturales			
	Inverse DM	***Entropy***	
Medidas Texturales			

Figura 5.13: Tabla con el resultado de las imágenes asociadas a las medidas normalizadas para todos los conjuntos resultado de la integración de segmentar el coeficiente *wavelet* 1 por *Otsu* con la imagen multiespectral clasificada con *Spatial C-Means* con 8 *clusters*, 100 iteraciones y 10 de peso

Cabe destacar que para las medidas *Min Grey* y *Entropy* las imágenes salen en color negro (todos los *píxeles* tienen valor 0) debido al problema antes comentado, que el valor mínimo y máximo son 0, por tanto al dividir por 0 no se obtiene valor y se queda negra la imagen. El resto de imágenes manifiestan la diferencia que hay dentro de las medidas para cada conjunto, lo que hace que al colorear la imagen asociada a cada atributo, cada conjunto es identificado por este valor y se dibujan perfectamente todos los contornos quedando una silueta muy parecida a la de la imagen original.

6. CONCLUSIONES

Tal y como se ha mencionado en los primeros capítulos, este trabajo permite la definición y caracterización de objetos asociados a una escena mediante la segmentación, es decir, dividir una imagen digital en varios objetos con el fin de simplificar una imagen en otra más fácilmente analizable y procesable desde un punto de vista computacional. Las aplicaciones prácticas de la segmentación de imágenes son múltiples, abarcando ámbitos tan variados y distintos como la medicina, reconocimiento de caras o iris, sensores de huellas digitales, sistemas de control de tráfico, etc.

Uno de los inconvenientes encontrados durante el proceso, el cual comienza con el procesado de una imagen pancromática, es el de la extracción de las características texturales de dicha imagen. Para ello hay que apoyarse en la denominada transformada *Wavelet*, la cual permite extraer el detalle espacial que se pierde al pasar de una resolución espacial a otra menor y que, además, elimina su información espectral. A continuación se obtienen los objetos espectrales de la imagen multiespectral (segunda imagen que es necesaria en el proceso de segmentación que este trabajo resuelve) y los objetos texturales de la imagen pancromática, los cuales terminan integrándose a continuación. Finalmente, para cada conjunto detectado, se podrá obtener un vector de atributos asociados donde se reflejan los valores de ciertas funciones como la media, desviación estándar, entropía, etc. Esta representación facilita el procesado de dicha información.

Para la obtención de los objetos texturales se han propuesto dos algoritmos: *Otsu* y *Watershed*, después de haber estudiado (y desechado por los malos resultados que se observaron) otros como *Voronoi*, *Mixture Modeling*, *Maximum Entropy*… A tenor de las pruebas realizadas (ver apartado 5 del presente documento para más información), *Otsu* es el que mejores resultados presenta por dos motivos: número de objetos (menor número de objetos detectados que *Watershed*, lo que se traduce en tiempos de ejecución inferiores), y calidad de los objetos detectados. Cabe mencionar que si se desea un fuerte nivel de detalle en cuanto a objetos detectados, sería el algoritmo de *Watershed* el

elegido, teniendo en cuenta que puede correrse el riesgo de que estos objetos "de más" puedan introducir ruido en la imagen.

Por otro lado, para la obtención de los objetos espectrales de la imagen multiespectral elegida, se han propuesto 3 algoritmos (también después de haber desechado otros como *Maximum Likelihood*, *Isodata*, Redes Neuronales…): *REF K-Means*, *Fuzzy C-Means* y *Spatial Fuzzy C-Means*. Decidir con cuál de los 3, y en qué condiciones de ejecución, se obtendrían los mejores resultados es muy subjetivo. Tras estudiar los resultados propuestos en el apartado 5 del documento, se podría concluir que *Spatial Fuzzy C-Means* tiene una ligera ventaja sobre el resto para ejecuciones con muchas iteraciones (al menos 100), configurándolo con 8 *clusters* y con un peso de 10. En cambio, si se prefiere no entrar a tanto nivel de detalle y no clasificar con un número elevado de *clusters* para tener sólo una aproximación general de lo que puede tener la imagen, se optará por el algoritmo *Fuzzy C-Means*.

Los resultados obtenidos tras la integración de los objetos texturales y espectrales muestran un buen comportamiento tanto para zonas de cultivo con texturas regulares como para otras zonas más homogéneas y con patrones de texturas aleatorios.

Para finalizar, la herramienta desarrollada obtendrá los atributos deseados sobre los conjuntos obtenidos en el último paso. Dichos atributos, así como sus valores normalizados, se presentan en sendas tablas por pantalla, y se exportan a fichero *Excel* para así poder ser estudiados con posterioridad. A continuación se representan dichos atributos normalizados en forma de imágenes, traduciéndose estos valores normalizados a un nivel de la escala de grises. De esta manera, de un simple vistazo, se pueden distinguir los conjuntos de una imagen para una medida.

Por último, y teniendo en cuenta todo lo expuesto anteriormente, el trabajo presentado se puede usar como base para futuros trabajos de clasificación de conjuntos. Haciendo un análisis exhaustivo de los atributos (espectrales, texturales y/o morfológicos) que mejor caractericen la cubierta objeto de estudio en cada caso, se podrá identificar qué es: un río, un camino, un campo de cultivo…

7. REFERENCIAS

[Arqu 03] Arquero, A., Gonzalo, C., & Martínez, E. 2003. *Una aproximación desde la superficie al satélite*. Fundación General de la UPM.

[Capo 11] Caponetti, L. 2011. *Fuzzy Clustering per la segmentazione di immagini in ImageJ*. Universidad de Bari. (https://github.com/arranger1044/SFCM/blob/master/SFCM.pdf)

[Fern 08] Fernández, T. 2008. *Clasificación digital*. Universidad de Jaén. (http://coello.ujaen.es/Asignaturas/teledeteccion/tel/tel_tfc_archivos/Tema14.pdf)

[Garc 08-09] García, C. & Gómez, I. 2008-2009. *Algoritmos de Aprendizaje: KNN & KMEANS*. Universidad Carlos III de Madrid. (http://www.it.uc3m.es/jvillena/irc/practicas/08-09/06.pdf)

[Garv 11-12] Garví, E., Holgado, J. & Rodríguez, L. 2011-2012. *Ingeniería del Software II. Diseño.* Universidad de Granada. (http://lsi.ugr.es/~is2/Teoria/Transparencias/Tema6-1112red.pdf)

[Gonz 04] González-Audícana, M., Otazu, X., Fors, O., Seco, A. & García, R. 2004. *Bondad de los algoritmos de descomposición Wavelet de Mallat y 'à trous' para la fusión de imágenes Quickbird.* Revista de Teledetección. Universidad de Navarra. (http://www.aet.org.es/revistas/revista21/AET21-15.pdf)

[Gonz 12] Gonzalo, C. & Lillo, M. 2012. *Caracterización multiescala de objetos como herramienta para la clasificación de imágenes de alta resolución espacial*. Asociación Española de Teledetección, Revista de Teledetección. (http://www.aet.org.es/revistas/revista38/Numero38_02.pdf)

[ImageJ] ImageJ Official Documentation. *Spatial Fuzzy C-Means plugin in ImageJ*. (https://github.com/arranger1044/SFCM/blob/master/doc/SFCM%20micro%20guide%20ENG.pdf)

[Insti] Instituto Nacional de Tecnologías Formativas y de Educación del Profesorado. *Glosario de Teledetección*. (http://concurso.cnice.mec.es/cnice2006/material121/unidad3/glosario.htm)

[ITM] IT-Mentor. *Pruebas de Software*. (http://materias.fi.uba.ar/7548/PruebasSoftware.pdf)

[Itur 98] Iturrate, E. 1998. *Curso básico de Teledetección con ENVI.* Estudio Atlas S.L. (http://www.innovanet.com.ar/gis/TELEDETE/TELEDETE/teledete.htm)

[Laotra] La otra opinión. *Teledetección*. (http://laotraopinion.*net*/satelite_colombia_17.html)

[Lorca 10] Lorca, G., Arzola, J. & Pereira, O. 2010. *Segmentación de Imágenes Médicas Digitales mediante Técnicas de Clustering*. Rev. Aporte Santiaguino. (http://www.scielo.org.pe/pdf/as/v3n1/a15v3n1.pdf)

[Man 10] Man-olo. 2010. *Introducción a la percepción remota.* Scribd. (http://es.scribd.com/doc/36632503/Introduccion-a-La-Percepcion-Remota)

[Natio] The National Academies. *Wavelets: ver el bosque y los árboles*. (http://www7.nationalacademies.org/spanishbeyonddiscovery/mat_008276.html)

[Roerd 01] Roerdink, J. & Meijster, A. 2001. *The Watershed Transform: Definitions, Algorithms and Parallelization Strategies*. IOS Press. (http://www.cs.rug.nl/~roe/publications/parwshed.pdf)

[SRGIS] Sensores Remotos & GIS. *Guía básica sobre imágenes satelitales y sus productos.* Business Image Group & Spot Image. (http://www.srgis.cl/pdf/guia_basica_imagenes_satelitales.pdf)

[Tele 11] Teledetet. 2011 *Estimación de la Matriz de Confusión*. (http://tutorial-percepcion-remota-satelital.blogspot.com/2011/04/matriz-de-confusion-estimacion-de-la.html)

[UM] U. de Murcia. *Fotointerpretación y Teledetección*. (http://www.um.es/geograf/sig/teledet/)

[UPM] (www.die.upm.es/im/papers/Caseib01_Ortuno.pdfSimilares)

[Wikia] Wikipedia. *Teledetección*. (http://es.wikipedia.org/wiki/Teledetecci%C3%B3n)

[Wikib] Wikiagro. *¿Qué es un mapa de productividad*? (http://www.wikiagro.com/es/Qu%C3%A9_es_un_mapa_de_productividad%3F)

[Wikic] Wikipedia. *Matriz de Confusión*. (http://es.wikipedia.org/wiki/Matriz_de_confusi%C3%B3n)

[Wikid] Wikipedia. *Método del valor umbral*. (http://es.wikipedia.org/wiki/M%C3%A9todo_del_valor_umbral)

[Wikie] Wikipedia. *Segmentación (procesamiento de imágenes)*. (http://es.wikipedia.org/wiki/Segmentaci%C3%B3n_(procesamiento_de_im%C3%A1genes))

[Wikif] Wikipedia. *Software*. (https://es.wikipedia.org/wiki/Software#Captura.2C_an.C3.A1lisis_y_especificaci.C3.B3n_de_requisitos)

[Wikig] Wikipedia. *Casos de Uso*. (http://es.wikipedia.org/wiki/Caso_de_uso)

[Wikih] Wikipedia. *Diagrama de clases*. (http://es.wikipedia.org/wiki/Diagrama_de_clases)

[Wikii] Wikipedia. *Teoría de la medida*. (http://es.wikipedia.org/wiki/Teor%C3%ADa_de_la_medida)

I. APÉNDICE

I.1. Guía de Usuario

Se parte de varios coeficientes *à trous* que se obtienen de la imagen pancromática de partida, como se pueden observar en la Figura I.1.

Figura I.1: Diferentes coeficientes de la imagen pancromática con los que parte el *plugin*

Con uno de estos coeficientes abiertos (o incluso la propia imagen pancromática), se selecciona uno de los dos algoritmos de segmentación (*Otsu Thresholding for IJObject* o *Watershed Algorithm for IJObject*) que aparecen en el menú *1. Segmentation* del *plugin IJObject* de *ImageJ*, el cual obtiene directamente su correspondiente imagen *Watershed* u *Otsu* al aplicarle el algoritmo que lleva dicho nombre. Se puede visualizar el resultado para tres coeficientes en la Figura I.2 para el caso de *Watershed*. Aquí ya están los objetos segmentados.

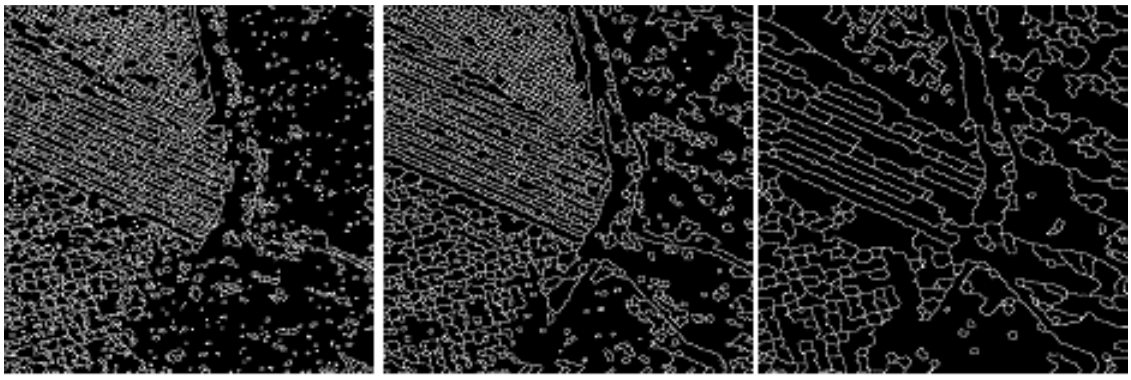


Figura I.2: Resultado de aplicar el algoritmo de *Watershed* a los coeficientes de la Figura 2.4

A continuación, con esta imagen segmentada abierta, se selecciona dentro del *plugin IJObject,* el menú *2. Spatial Object* el cual analiza (con el código del *plugin Analyze Particles* de *ImageJ*, cuyos parámetros de selección adecuados se encuentran marcados en la Figura I.3) todos los objetos y los diferencia unos de otros obteniendo como resultado esa misma imagen con los objetos enumerados y una lista de cada uno de ellos con sus correspondientes área, ancho, alto, desviación estándar, perímetro…

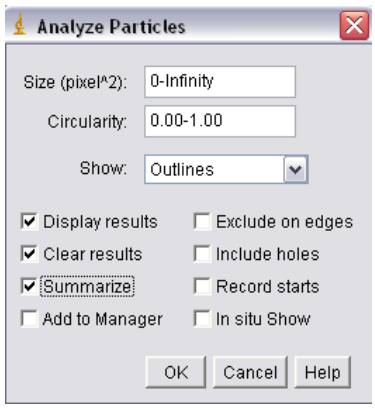


Figura I.3: Ventana de selección de *Analyze Particles*

Se va a crear un fichero de texto por cada objeto que detecte el analizador de objetos, que contiene el objeto completo, es decir, con las coordenadas del borde y las del interior. Se almacenan en una ruta del ordenador, obligatoriamente en una carpeta que se llame *Polygon*. El nombre de cada fichero es del tipo *polygon* + número de objeto detectado (incremental) + *.txt,* y cada uno contiene todas las coordenadas del objeto (por cada fila, una coordenada del tipo *"(" + X + "," + Y + ")"*) y finaliza con el número total de ellas. Lógicamente hay una correspondencia entre el nombre del número de objeto asociado al título de los ficheros polígono. Se puede ver un ejemplo en la Figura I.4.

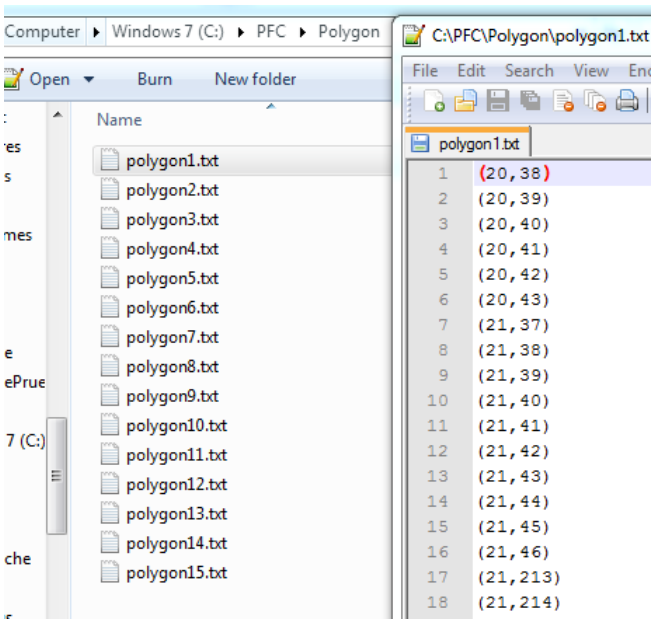


Figura I.4: Ficheros almacenados en disco que tienen los *píxeles* de los objetos en los que se ha segmentado la imagen pancromática

Con esto ya se tienen definidos perfectamente todos los objetos de cualquier imagen pancromática.

A continuación, se abre la imagen multiespectral correspondiente, con la condición de que al menos tenga dos bandas montadas. El ejemplo de la Figura I.5 contiene 4 bandas.

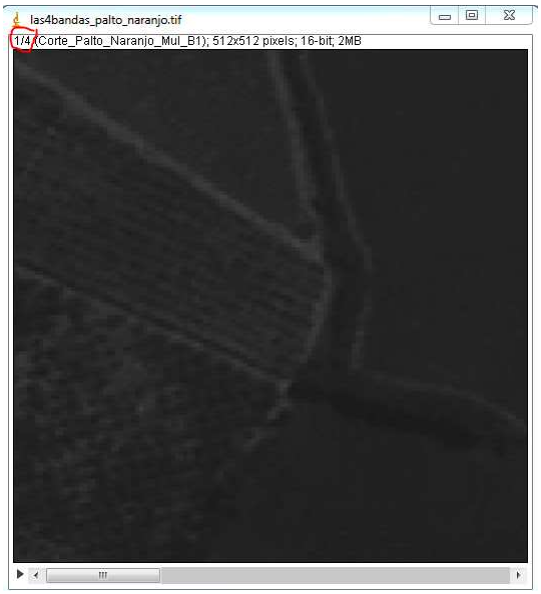


Figura I.5: Imagen multibanda o multiespectral con 4 bandas

Con esta imagen abierta, se selecciona uno de los tres algoritmos de segmentación (*REF K-Means Clustering for IJObject*, *Fuzzy C-Means Clustering for IJObject* o *Spatial Fuzzy C-Means Clustering for IJObject*) que aparecen en el menú *3. Spectral Object* del *plugin IJObject* de *ImageJ*, el cual obtiene directamente su correspondiente imagen segmentada al aplicarle el algoritmo que proceda. Se puede visualizar el resultado para el algoritmo *REF K-Means Clustering for IJObject* en la Figura I.6 seleccionando 7 *clusters* además de elegir la opción de que se visualice con colores RGB aleatorios. Aquí ya están los objetos segmentados.

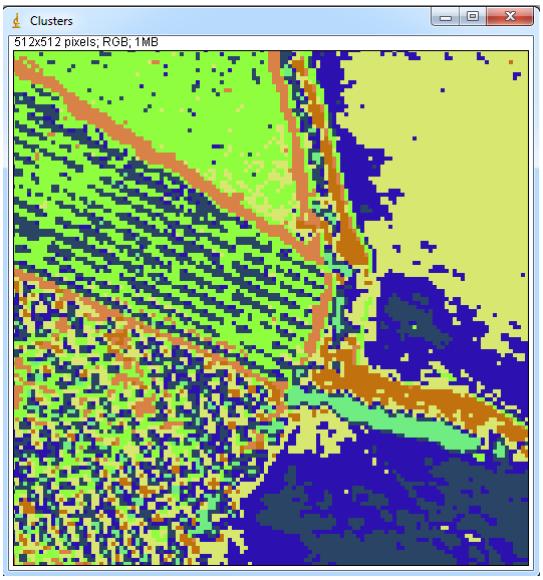


Figura I.6: Imagen multibanda segmentada por el algoritmo REF *K-Means*

Este menú *3. Spectral Object* contiene, además de los tres algoritmos, una ventana de ayuda *Help for Spectral Algorithms* que muestra cómo generar una imagen multibanda y qué parámetros de selección se han de seleccionar en cada algoritmo.

Después de este paso, se irá al menú *4. Final Object* que hará una fusión de estas dos imágenes que se han segmentado y se generará un nuevo directorio *Set* que contendrá una serie de ficheros los cuales corresponderán a cada uno de los objetos resultado de la integración de las dos imágenes. El nombre de cada fichero es del tipo *set* + número de objeto detectado en la imagen pancromática segmentada + _ + número de objeto detectado en la imagen multiespectral + *.txt*, y cada uno contiene todas las

coordenadas del objeto (por cada fila, una coordenada del tipo "(" + X + "," + Y + ")"). En la Figura I.7 se muestra un ejemplo donde el área ocupada por el objeto número 2 detectado en la imagen pancromática segmentada, corresponde con tres de los seis *clusters* de la imagen multiespectral segmentada (con el 0, 2 y 3) por lo que este objeto número 2 pasará a dividirse en tres conjuntos: *set2_0*, *set2_2* y *set2_3*, que contendrán los puntos correspondientes a esa zona. Así sucesivamente se irán fraccionando cada uno de los objetos de la imagen pancromática segmentada.

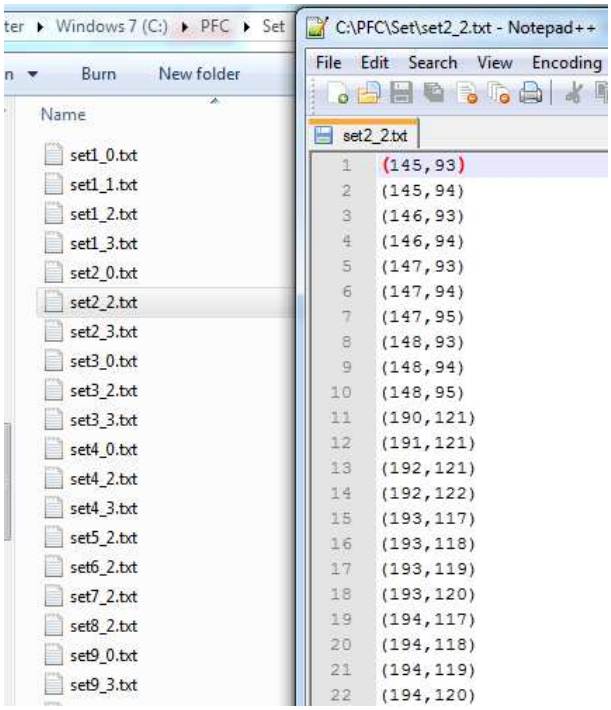


Figura I.7: Ficheros almacenados en disco que contienen los *píxeles* de los conjuntos resultado de la integración de la segmentación pancromática y de la multibanda

Al final de este proceso se mostrará la imagen resultado de esta fusión de imágenes segmentadas con cada conjunto perfectamente delimitado, como se aprecia en la Figura I.8.

Figura I.8: Imagen que muestra los conjuntos resultado de la integración de la segmentación pancromática y de la multibanda

Ahora se selecciona del menú el punto *5. Measurements* con el fin de poder obtener diferentes medidas de estos conjuntos resultados de la integración. La ventana (ver Figura I.9) da la posibilidad de poder elegir de qué imagen o imágenes (la imagen pancromática y cada una de las bandas por separado de la imagen multiespectral) se desean sacar las medidas texturales, espectrales o de forma, y hay que especificarle en qué directorio se desea generar los ficheros con estos datos en forma de tabla (tanto las medidas como su correspondiente normalización) para poder trabajar con ellos posteriormente.

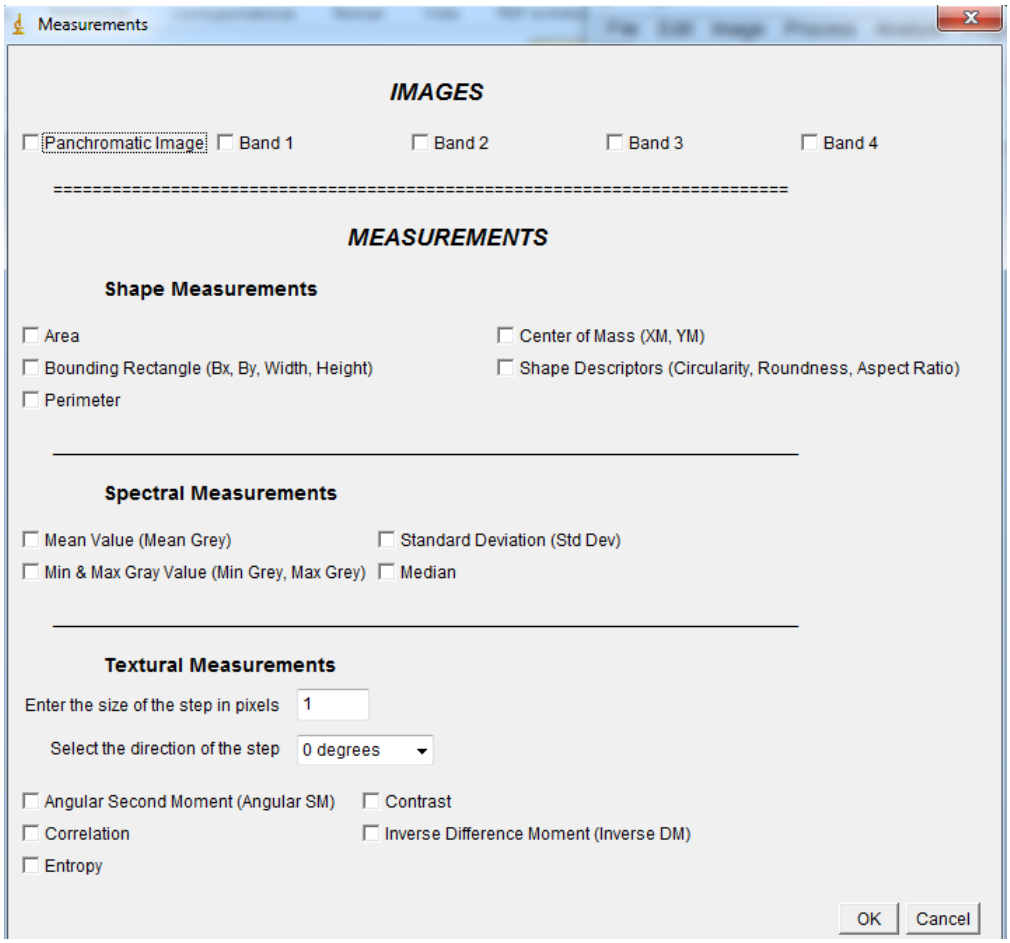


Figura I.9: Ventana de selección de imágenes y medidas

A continuación, aparecerán las ventanas correspondientes de *ImageJ* con estos datos como se puede ver en la Figura I.10.

Results

File Edit Font Results

	Set	Area	Perimeter	Median (Panchromatic)	Entropy (Panchromatic)	Median (Band 3)	Entropy (Band 3)
1	set1_0	12221	6838	0	2.478	231	4.410
2	set1_1	7975	3624	0	0.628	9	4.129
3	set1_2	3882	2318	0	2.071	15	4.155
4	set1_3	28729	10310	0	2.136	28	4.465
5	set2_0	8766	3284	0	1.927	38	3.889
6	set2_2	1038	764	0	1.746	33	3.472
7	set2_3	3811	2226	0	0.711	35	3.647
8	set3_0	43	70	0	1.211	34	2.451
9	set3_2	13864	2460	0	4.026	20	2.125
10	set3_3	12532	1882	0	1.334	20	2.892
11	set4_0	347	294	0	3.331	37	3.329
12	set4_2	12	16	81	3.120	27	0.868
13	set4_3	537	210	0	0.643	35	3.605
14	set5_2	1890	254	0	2.515	20	1.411
15	set6_2	229	68	16	3.911	21	2.305
16	set7_2	4085	582	0	3.501	20	1.411

Normalization

File Edit Font

	Set	Median (Panchromatic)	Entropy (Panchromatic)	Median (Band 3)	Entropy (Band 3)
1	set1_0	0	113.713	255	251.158
2	set1_1	0	26.937	0	231.635
3	set1_2	0	94.646	6.892	233.470
4	set1_3	0	97.656	21.824	255
5	set2_0	0	87.879	33.311	214.920
6	set2_2	0	79.391	27.568	185.942
7	set2_3	0	30.829	29.865	198.140
8	set3_0	0	54.289	28.716	115.002
9	set3_2	0	186.330	12.635	92.314
10	set3_3	0	60.032	12.635	145.634
11	set4_0	0	153.742	32.162	176.035
12	set4_2	142.448	143.847	20.676	4.952
13	set4_3	0	27.651	29.865	195.239
14	set5_2	0	115.440	12.635	42.716
15	set6_2	28.138	180.932	13.784	104.834
16	set7_2	0	161.716	12.635	42.729

Figura I.10: Tabla de resultados y tabla con sus correspondientes valores normalizados

El siguiente punto del menú, el *6. Attributes' Spacialization*, mostrará una ventana (ver Figura I.11) que da la posibilidad de visualizar para cada imagen y para cada medida que se haya seleccionado en el punto anterior del menú, su correspondiente imagen fruto de colorear cada conjunto con el valor normalizado para cada uno de ellos.

Figura I.11: Ventana de selección de imágenes

La siguiente Figura I.12 muestra el resultado para la imagen pancromática con medida mediana, que como se ve en el fragmento de la Figura I.10, hay muchos conjuntos que contienen valor 0, por ese motivo la mayoría de la imagen es negra.

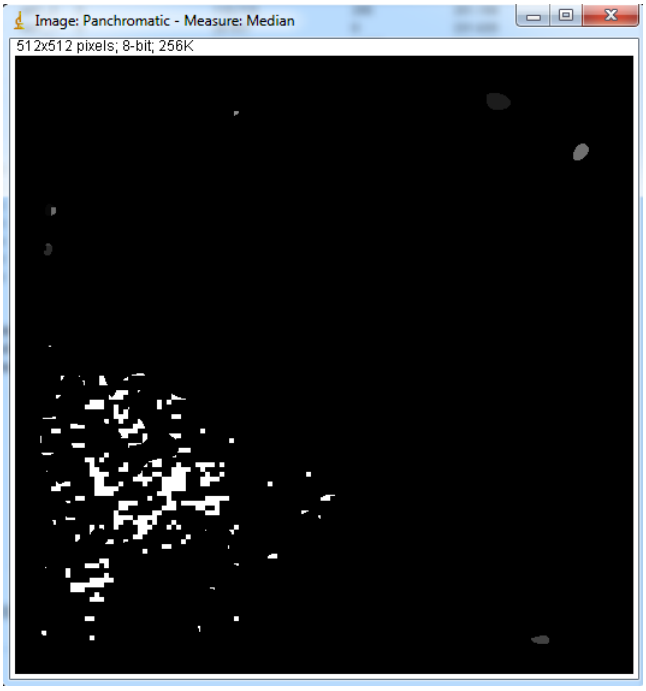


Figura I.12: Imagen resultado para el caso de la medida mediana e imagen pancromática

El último punto del menú, el *7. Import Sets*, es una extensión del *plugin* que sirve para importar una imagen pancromática, una imagen multiespectral y los objetos generados por un programa hecho en *Matlab* resultado de la integración de las dos imágenes (este fichero de texto está hecho de tal forma que contiene números seguidos separados por un blanco, y cada uno de ellos corresponde al número de objeto que pertenece ese *píxel*, es decir, el primer número corresponderá al *píxel* (0, 0), el segundo

al (0, 1) y así sucesivamente). En la Figura I.13 se puede ver un ejemplo donde todos los primeros *píxeles* pertenecen al conjunto 34, los de a continuación serán del conjunto 49. De esta forma, genera en el directorio *Set* los correspondientes ficheros de texto de cada conjunto y sustituye a los pasos del menú 1, 2, 3, y 4, por lo que lo siguiente que habría que hacer es ir al menú *5. Measurements* y continuar como se ha explicado previamente.

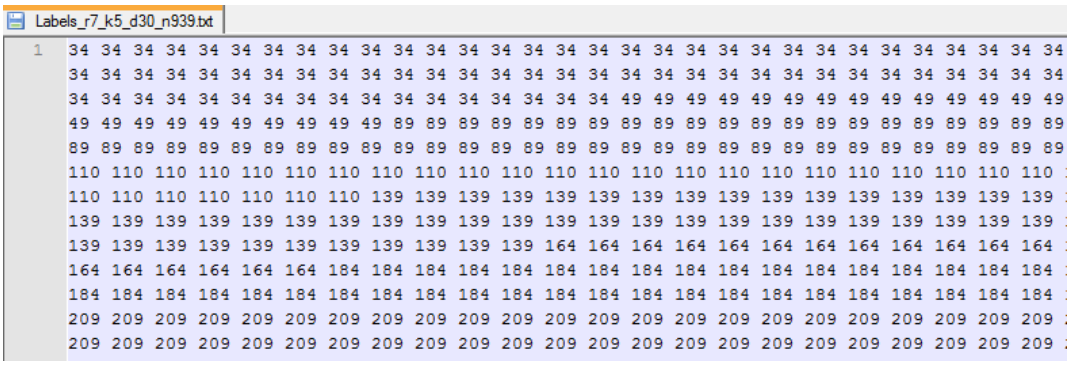


Figura I.13: Formato de la correspondencia conjuntos-*píxeles* en *Matlab*

I.2. Interfaz de usuario algoritmos espectrales

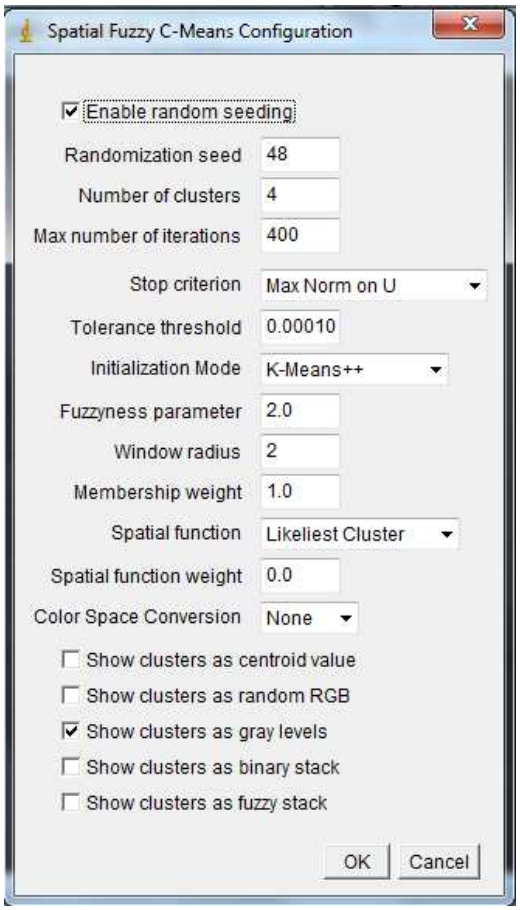


Figura I.14: Ventana de configuración del algoritmo *Spatial Fuzzy K-Means*

Al seleccionar el algoritmo de *Spatial Fuzzy K-Means*, aparecerá la ventana que se puede ver en la Figura I.14. Para los otros dos algoritmos (*REF K-Means* y *Fuzzy C-Means*) sus interfaces son un subconjunto de ésta, por lo que esta explicación es válida también para ambos. A continuación se procede a explicar cada una de las casillas para introducir los parámetros adecuados a la imagen que se quiera segmentar.

Randomization seed → Semilla usada para generar números aleatorios necesarios para la inicialización.

Number of clusters → Número de *clusters* en los que se va a segmentar la imagen.

Max number of iterations → Máximo número de iteraciones permitidas.

Stop criterion → Criterio de parada. Si el criterio elegido supone un valor por debajo del umbral seleccionado, el algoritmo para.

Tolerance threshold → Umbral usado como criterio de parada del algoritmo.

Initialization mode → Criterio de inicialización para las matrices *V* y *U* (*K-Means++*, *Random U* y *Random V*). Cada criterio puede conducir a un resultado diferente.

Fuzzyness parameter → Cuando el valor es cercano a 1, los resultados son similares a los del algoritmo *K-Means.* 2 significa normalización linear.

Window radius → Usado para computar la función espacial. Para un valor de 2 se obtendría una ventana de (*5x5)* centrada en el *píxel* actual.

Memembership weight → Valor usado para actualizar la matriz *U* usada en la función espacial.

Spatial Function → Tipo de función espacial. El *cluster* más probable computará la suma de los miembros de los *píxeles* en el vecindario.

Color space conversion → Conversión de color desde *RGB* hasta *HSB*. Las imágenes en escala de grises no pueden ser convertidas.

Checks parte inferior → Determinan el modo de visualización de la imagen segmentada. Las regiones pueden ser etiquetadas con colores *RGB* aleatorios, escala de grises, como una pila de imágenes binarias, o con una pila de imágenes *fuzzy* [ImageJ].

I.3. Generar un *plugin* para ImageJ

Un *plugin* de *ImageJ* no es más que un fichero *JAR* que contiene los ficheros resultantes de compilar los fuentes *Java* (ficheros *.class*), así como un fichero ***plugins.config*** con una sintaxis y estructura que se abordará a continuación, y que es

usado por *ImageJ* para determinar qué entradas de menú generar, así como los fuentes que hay que ejecutar para cada una de dichas opciones de menú.

Para que *ImageJ* detecte el *plugin*, es tan sencillo como copiar el *JAR* del *plugin* en cuestión al directorio *plugins* del directorio donde esté instalado *ImageJ*. La próxima vez que *ImageJ* se abra, el *plugin* será accesible desde sus opciones de menú definidas en *plugins.config*.

Aclarar que los fuentes dentro del *JAR* del *plugin* deben mantener la misma estructura de directorios que tenían en el entorno de desarrollo usado para su implementación.

Como se ha detallado al comienzo del presente apéndice, todo *plugin* de *ImageJ* debe contener un fichero de configuración, denominado *plugins.config* y situado a nivel de directorio raíz dentro del *JAR*, cuyo cometido es definir la estructura de las opciones de menú que el *plugin* ofrecerá al usuario, así como el fichero al que *ImageJ* llamará para proceder a la ejecución. La estructura de cada una de las líneas de *plugins.config* es la siguiente:

Estructura de Menús, Nombre de la opción de Menú, Clase *Java* a ejecutar (parámetros)

Se puede apreciar que cada entrada de *plugins.config* posee 3 elementos separados por comas:

- La estructura de menús de la que colgará la opción definida en la presente línea. Cada nuevo nivel se separa por el símbolo “>”.
- El nombre que dicha opción tendrá en *ImageJ*.
- Clase *Java* que *ImageJ* invocará (los parámetros a pasarle se definen entre paréntesis).

Se pasa a ver un ejemplo práctico de una línea de *plugins.config* del *plugin IJObject*: *Plugins>IJObject>1. Segmentation, "Watershed Algorithm for IJObject", IJObject_Watershed_Algorithm*

Esta entrada del fichero *plugins.config* define una opción de Menú llamada *Watershed Algorithm for IJObject*, que cuelga de la opción de Menú *1. Segmentation*, que a su vez cuelga de *IJObject*, que a su vez cuelga de la opción *Plugins* de *ImageJ*. La clase a invocar en caso de que el usuario se decantase por este algoritmo de ejecución sería *IJObject_Watershed_Algorithm*. A continuación (ver Figura I.15) se ilustra una captura con la opción de menú que se acaba de describir:

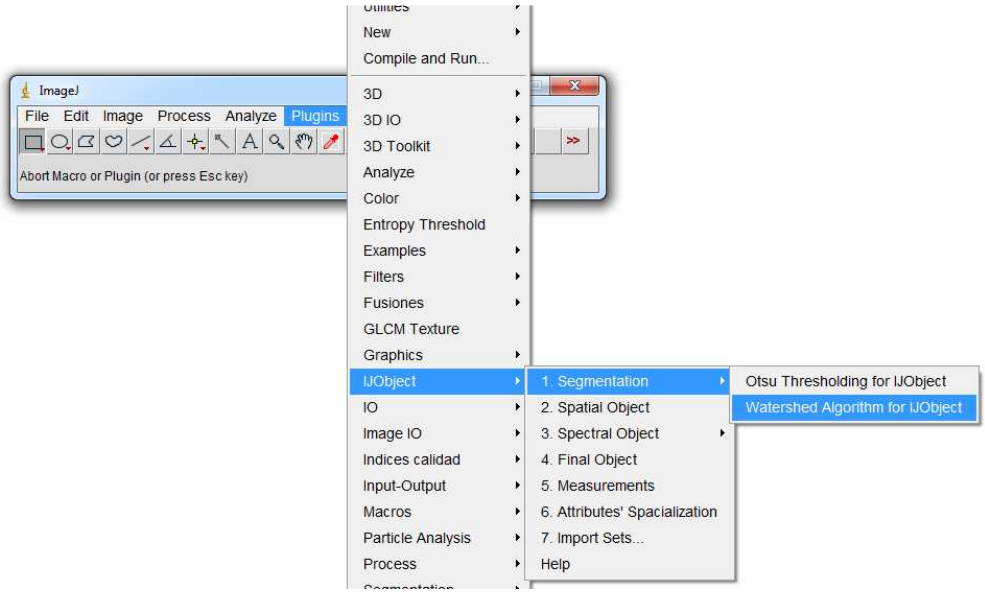


Figura I.15: Ejemplo de cómo generar una entrada a un *plugin* en *ImageJ*

ANEXO A

1. Introducción a *ENVI*

En esta sección se va a llevar a cabo un minucioso trabajo de cómo obtener una clasificación de imágenes multiespectrales con la herramienta *ENVI*. Se consideró hacer esta parte nada más comenzar este trabajo debido al escaso conocimiento en la materia y, gracias al potencial que tiene este programa, ha servido de ayuda para poder desarrollar el objetivo de la aplicación a desarrollar. Se adquiere una idea de lo que es una firma espectral, cómo funcionan diferentes algoritmos de clasificación (supervisados y no supervisados) con las distintas bandas de la imagen multiesprectal, qué es una matriz de confusión, qué es una *ROI* (*Region Of Interest*)… y todo ello de forma visual e interactiva. De este estudio se obtiene la conclusión de que los algoritmos más efectivos para clasificar (y que se incluyen en el *plugin* desarrollado) son los *K-Means* y derivados.

ENVI ofrece muchísimas utilidades y hace partícipe al usuario de todas ellas, permitiendo modificar valores según le convenga en cada momento. Es un gran complemento a *ImageJ* y, en este caso, ha servido de mucha ayuda para decantarse por unos algoritmos o por otros.

Lo primero es seleccionar en el menú principal de *ENVI*, ***File → Open Image File.*** Se escoge una imagen multiespectral almacenada en el PC y se pulsa ***Open***. En este momento aparece una nueva ventana con la lista de bandas disponibles de la imagen. De forma indiferente, hay que emparejar a cada color R, G y B, con una de las bandas. Estas diferentes combinaciones posibles de colores se van a utilizar para localizar e identificar áreas únicas dentro de la imagen con el fin de usarlas para adquirir conjuntos de entrenamientos para la clasificación. Seleccionar ***Load RGB*** y aparece la imagen correctamente cargada (ver Figura A.1).

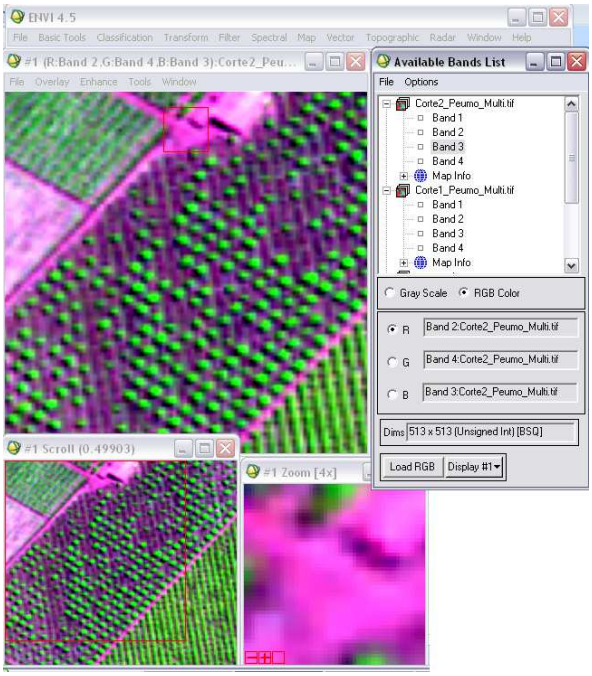


Figura A.1: Ejemplo de imagen multiespectral en *ENVI*

Si se selecciona otra combinación distinta de bandas aparece la imagen con otros tonos diferentes (ver Figura A.2).

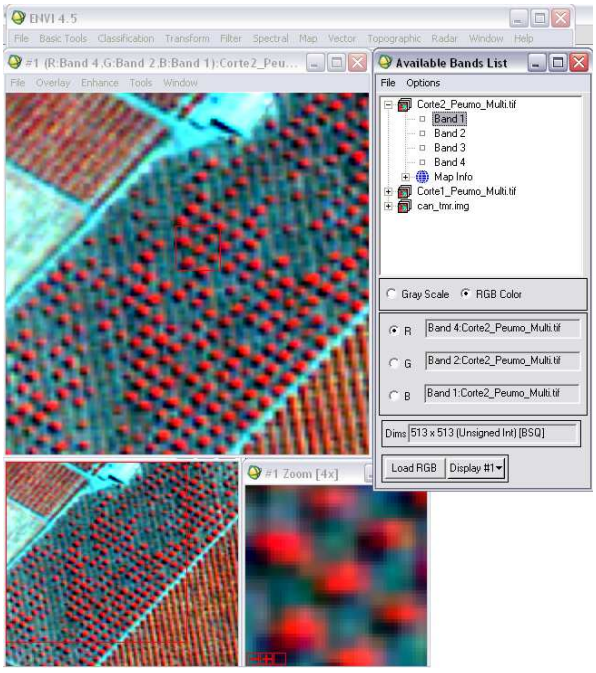


Figura A.2: Imagen multiespectral de la Figura A.1 con otra combinación de bandas

Esta imagen puede utilizarse como guía para la clasificación. Incluso en una simple imagen de tres bandas es sencillo descubrir que hay áreas que tienen características espectrales similares. Como por ejemplo, en esta última imagen, las áreas rojas que tienen forma redondeada corresponden a cultivos.

Otra función que proporciona *ENVI* es la de obtener las coordenadas de un punto. Seleccionando en el menú de la imagen ***Tools → Cursor Location/Value***. Situar el cursor sobre el punto a profundizar y se obtienen los valores del color de la imagen (según se hayan seleccionado las bandas), los valores de los datos reales, y las coordenadas longitud y latitud.

Situándose sobre uno de los cultivos rojos se obtienen los siguientes valores para un punto en concreto:

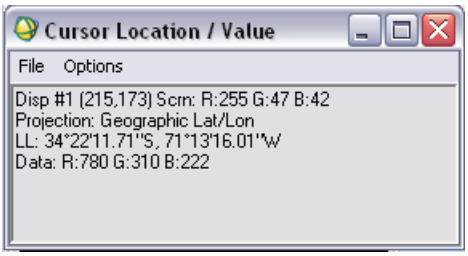


Figura A.3: Valores de varios parámetros en relación a un punto de una imagen multiespectral

2. *ENVI*: Firma espectral

A continuación se va a obtener la firma espectral de un punto de la imagen. Para ello, seleccionar ***Tools → Profiles → Z Profile (spectrum)***.

Se va a llevar a cabo para examinar un punto que pertenece a uno de los cultivos plantados (ver Figura A.4).

Se hace un click en la zona y se arrastra, observando que la firma espectral comienza a variar, es decir, que cada elemento que se encuentra en la superficie tiene una firma espectral diferente. Si hay zonas homogéneas, las firmas espectrales se van a parecer entre ellas.

Para analizar varias firmas espectrales como pueden ser vegetación, agua, suelo… se selecciona en la ventana de la firma, ***options → New Window: Blank***, y aparece una nueva ventana que hace que se pueda combinar varias firmas espectrales y poder analizar cada una de ellas.

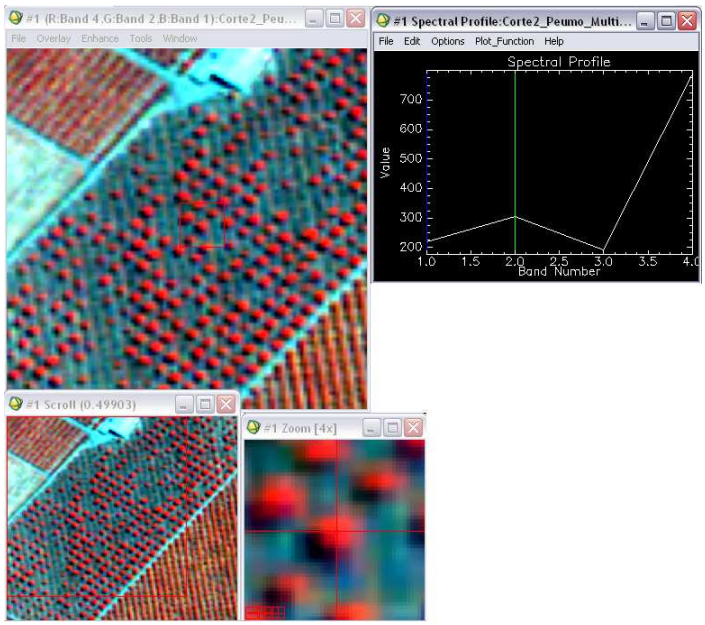


Figura A.4: Firma espectral de un punto concreto (zona de cultivo) de la imagen multiespectral

Si ahora se pulsa con el botón derecho del ratón sobre la firma espectral, seleccionar ***Plot Key***, y aparece en el margen derecho de la firma, las coordenadas *X* e *Y* del punto al que corresponde esa firma espectral. Si se pulsa sobre ese punto y se arrastra a la nueva ventana creada, se cargará ahí dicha firma. Esto se repite tantas veces como puntos se quieran investigar y se puede hacer una primera clasificación de todas las firmas espectrales obtenidas. En la Figura A.5 se pueden observar las firmas espectrales de 4 puntos de la imagen.

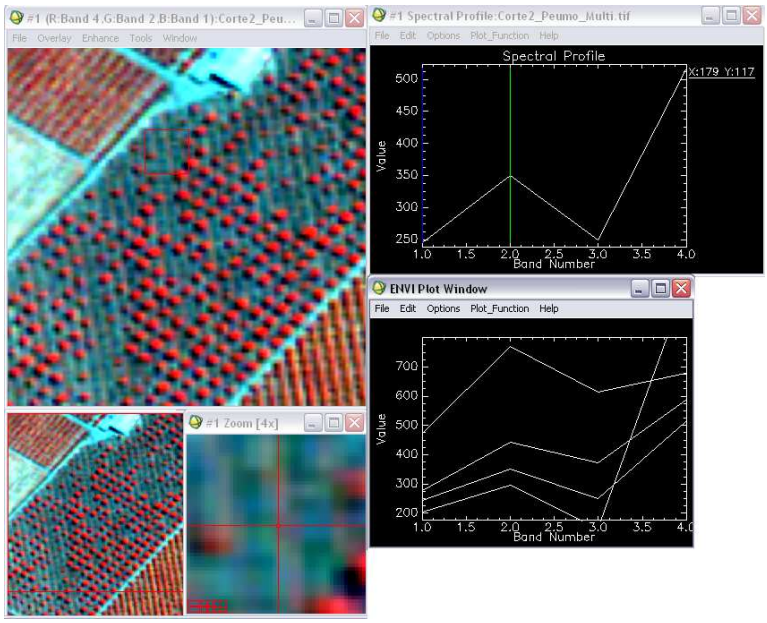


Figura A.5: Combinación de firmas espectrales de la imagen multiespectral en una sola gráfica

De este conjunto de firmas se puede decir que todas se refieren a vegetación, ya que siguen un trazado muy parecido (ver Figura A.5), aunque es variable dependiendo del estado fenológico, forma y humedad, pero en general presentan valores bajos en la zona del visible debidos al efecto absorbente de los pigmentos fotosintéticos de las hojas y a su estructura celular en el *NIR*.

Se pueden cambiar los colores de estas curvas y su grosor pulsando en la ventana de *Plot Window*, la opción ***Edit → Data Parameters***. Ver Figura A.6.

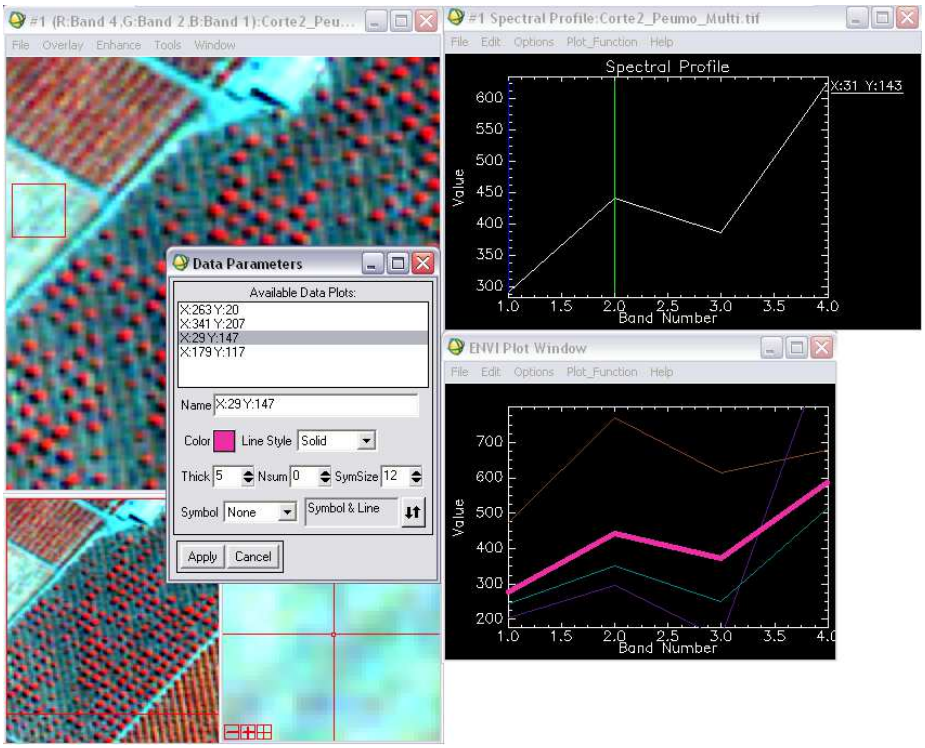


Figura A.6: Asignación de distintos anchos y colores a las firmas espectrales obtenidas

3. Métodos de clasificación no supervisada

3.1. *Isodata*

Seleccionar en el menú el tipo de clasificación que se va a llevar a cabo:

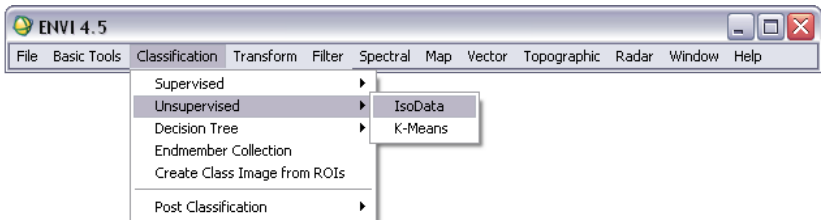


Figura A.7: Selección del método *Isodata* en *ENVI*

Aparece una pantalla (Figura A.8) en la que hay que elegir la imagen a clasificar:

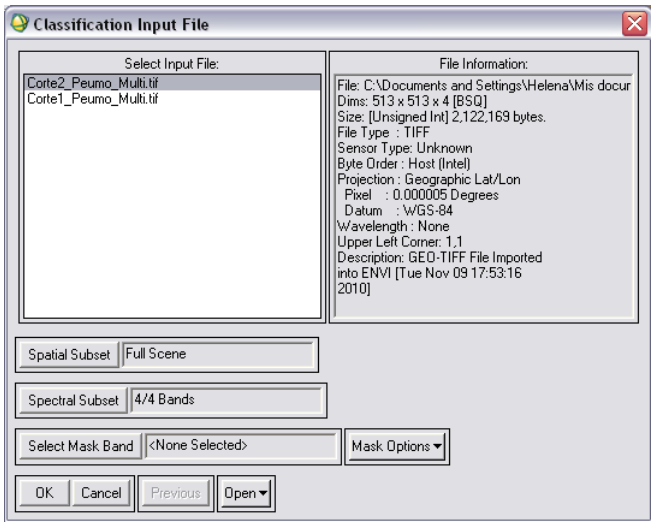


Figura A.8: Selección de la imagen a clasificar en *Isodata* en *ENVI*

Aceptar en ***OK*** y aparece una ventana (Figura A.9) con los parámetros que requiere introducir el algoritmo *Isodata*, y se dejan los que vienen por defecto, redireccionando la salida a memoria (ver Figura A.9).

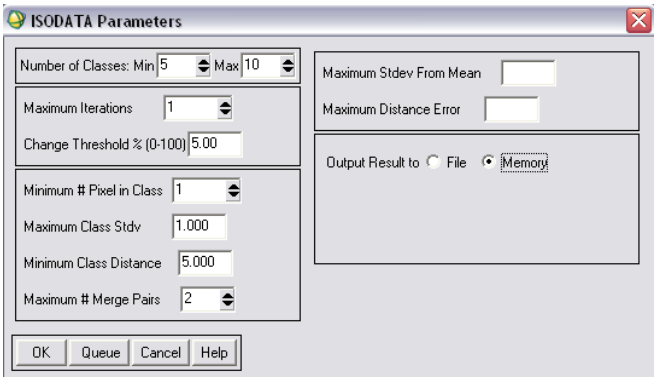


Figura A.9: Parámetros de *Isodata*

Aparece una nueva banda cargada en la ventana de bandas disponibles:

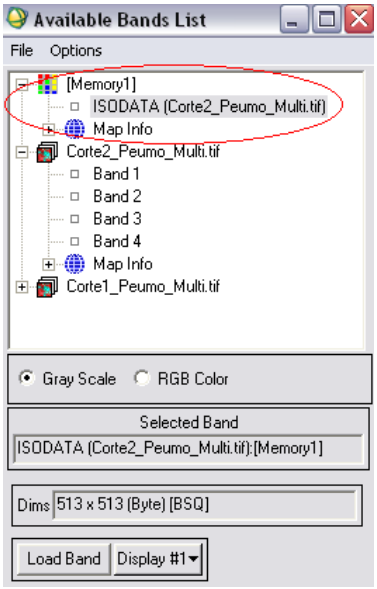


Figura A.10: Bandas disponibles

Ahora se selecciona una nueva ventana para poder visualizarlo de esta forma:

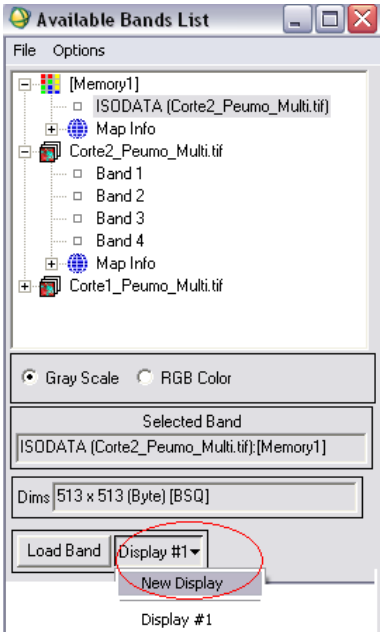


Figura A.11: Selección de nueva ventana para visualización

A continuación seleccionar la banda *ISODATA* y seleccionar en ***Load Band***:

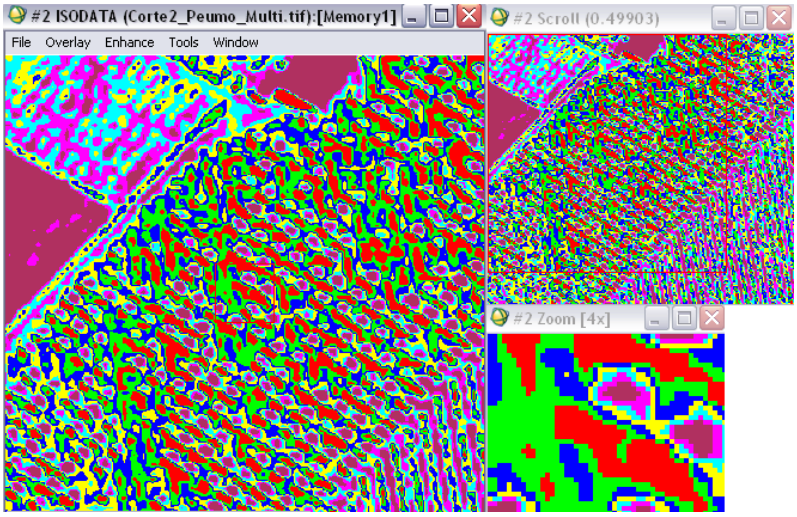


Figura A.12: Resultado de la clasificación *Isodata*

Si se compara esta clasificación con su imagen original se tiene lo siguiente:

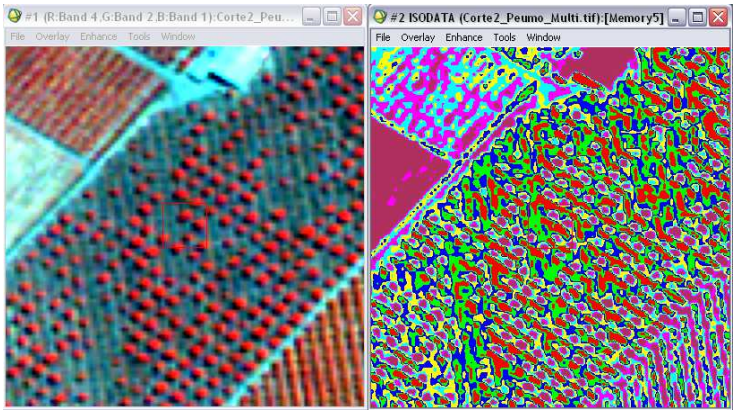


Figura A.13: Imagen multiespectral original y su clasificada por *Isodata*

Si se modifica el número mínimo y máximo de clases a 3 y 6 respectivamente, resulta la siguiente clasificación donde se comprueba que al haber menos clases, hay menos detalle de los elementos de la imagen (ver Figura A.14):

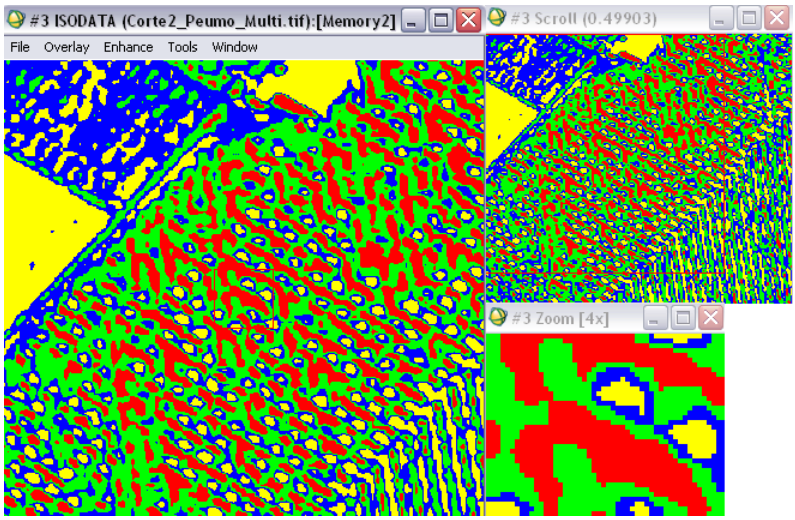


Figura A.14: Clasificación *Isodata* con menor número de clases

3.2. *K-Means*

Actúa de forma muy parecida a *Isodata*, se procede de la siguiente forma (Figura A.15):

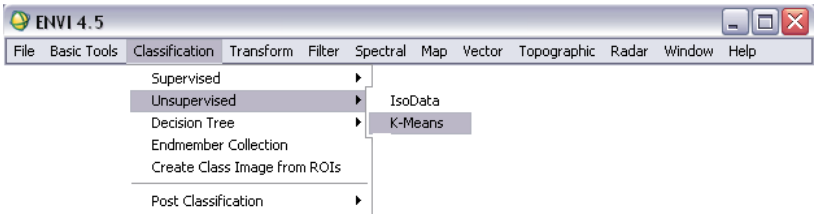


Figura A.15: Selección del método *K-Means* en *ENVI*

Seleccionar la imagen a clasificar y aceptar en ***OK*** (Figura A.16):

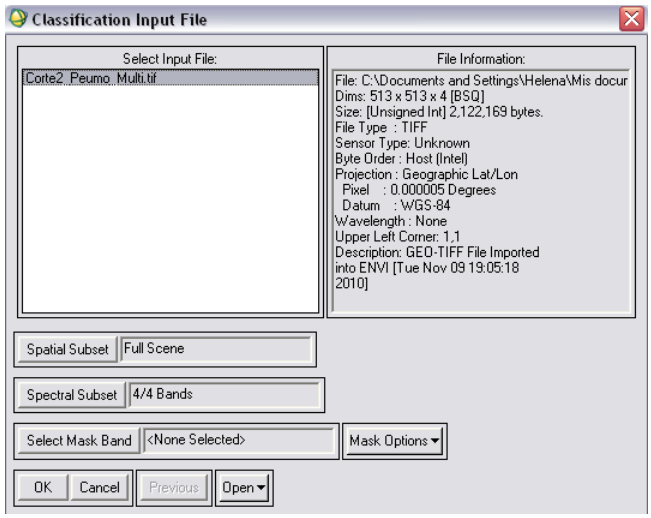


Figura A.16: Selección de la imagen a clasificar en *K-Means* en *ENVI*

Aparece una ventana con los parámetros que requiere introducir el algoritmo *K-Means*, y se dejan los que vienen por defecto, redireccionando la salida a memoria:

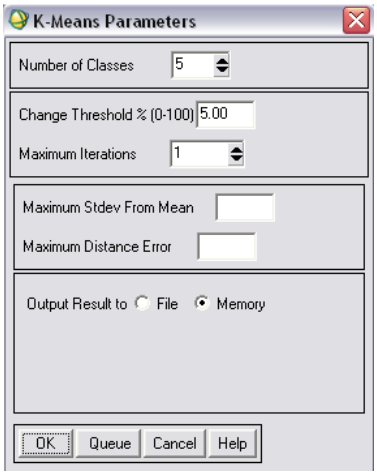


Figura A.17: Parámetros de *K-Means*

Al igual que en *Isodata*, se crea una nueva banda en la lista de bandas disponibles y se procede de igual forma cargándola en una nueva ventana para ver el resultado que obtiene *K-Means* (Figura A.18):

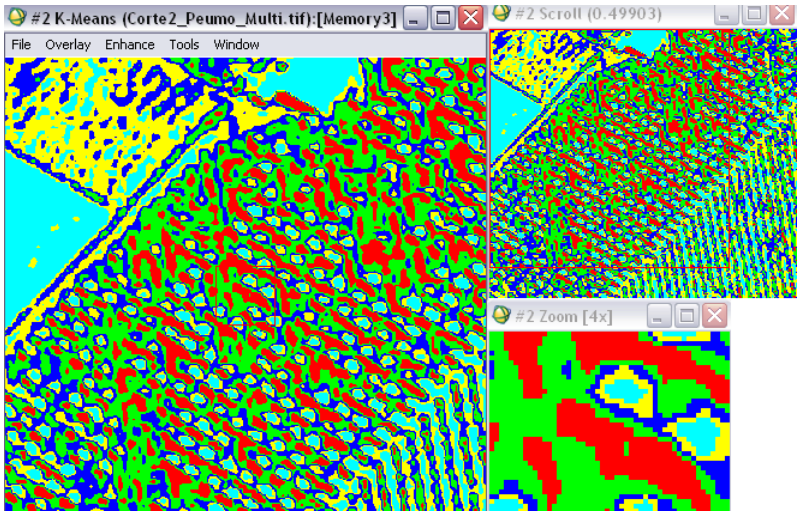


Figura A.18: Resultado de la clasificación *K-Means*

Se puede comparar la imagen original y la clasificada por *K-Means* (Figura A.19):

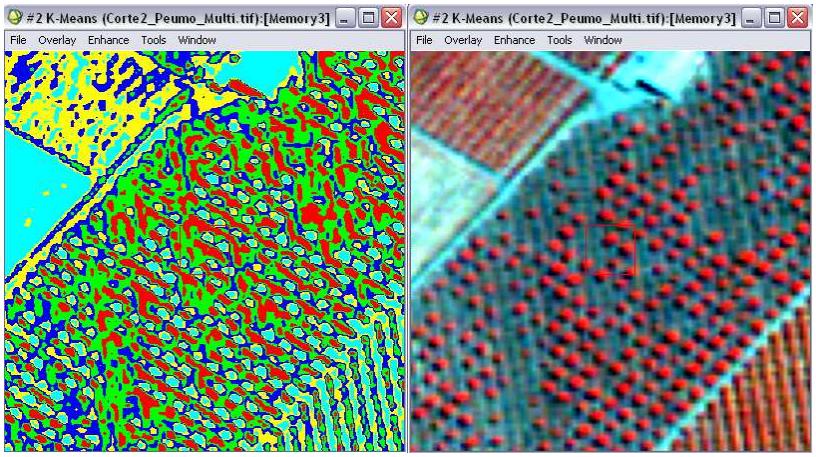


Figura A.19: Imagen multiespectral original y su clasificada por *K-Means*

Comparando la imagen original, *Isodata* y *K-Means*, se obtiene la Figura A.20:

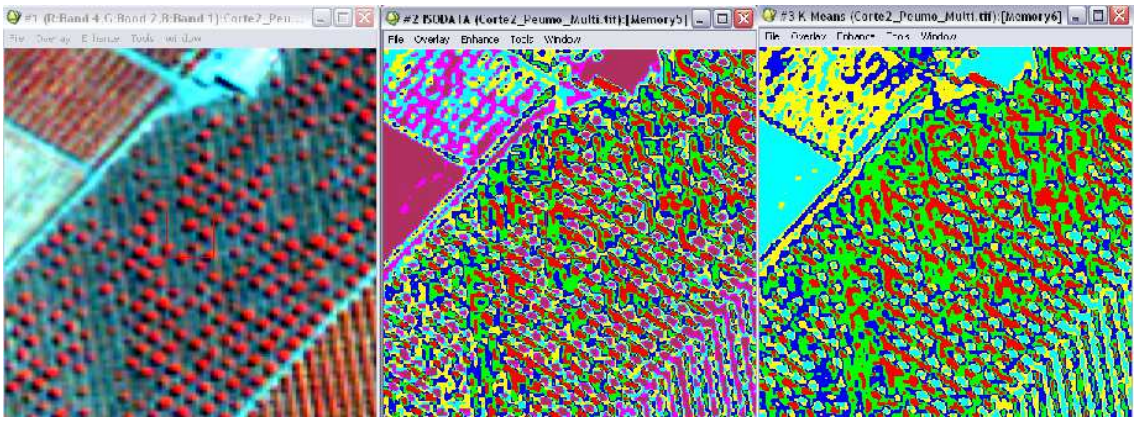


Figura A.20: Imagen multiespectral original, su clasificada por *Isodata* y por *K-Means*

4. Métodos de clasificación supervisada

4.1. *ROI*

Para extraer un segmento de forma regular o irregular se hace a través del uso de la herramienta *ROI* (*Region Of Interest*). La región de análisis puede tener cualquier forma por lo que esta opción se utiliza cuando se quiere analizar, por ejemplo, una cuenca, una comuna, una región o algo similar y no se desea que las características de lo observado sea “contaminado” estadísticamente por los elementos presentes en las áreas aledañas que no forman parte del estudio.

Para generar una región de interés se puede utilizar un archivo vectorial pre-existente o un área definida directamente sobre la imagen desde donde se extraerá el segmento.

Se va a utilizar esta última opción para obtener áreas de interés de una imagen multiespectral (Figura A.21):

Figura A.21: Selección de 3 áreas de interés sobre la imagen multiespectral

En la opción *Window*, seleccionar *Image*. Ésto determinará en cuál de las tres pantallas (*Image, scroll o zoom*) se puede crear la nueva *Region of Interest*.

En *ROI_Type*, seleccionar polígono. Utilizando el botón izquierdo del mouse se marca el área sobre la imagen desde la cual se desea extraer la subescena. Utilizar el botón derecho para terminar la definición del área, y para confirmar que se está conforme, hacer click por segunda vez con el botón derecho. El área marcada será cubierta por un polígono de color sólido (si es el primero que se hace, será rojo) a través del cual se puede verificar que el área es la deseada.

Con estas tres regiones que se han creado, a partir de sus características se van a ampliar con los vecinos que se asemejan a estas características de la siguiente forma. Se comienza, por ejemplo, con la región roja. Seleccionar esta región y, a continuación, pulsar en *Grow* para ampliarla. En este momento el programa pregunta con qué desviación estándar y qué número de vecinos se quiere llevar a cabo (Figura A.22):

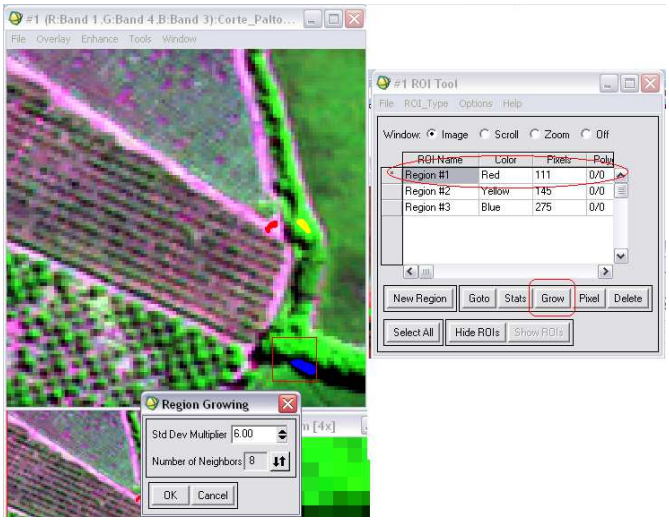


Figura A.22: Proceso de expansión de la *ROI* de color rojo tomándola como partida

Al aceptar se obtiene la Figura A.23:

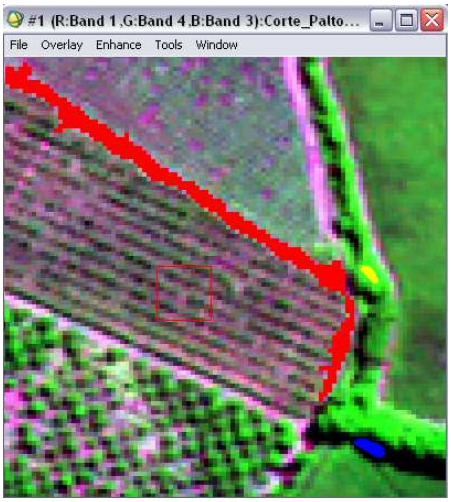


Figura A.23: Expansión de la región roja

Si se procede de igual forma con las otras dos regiones se tiene:

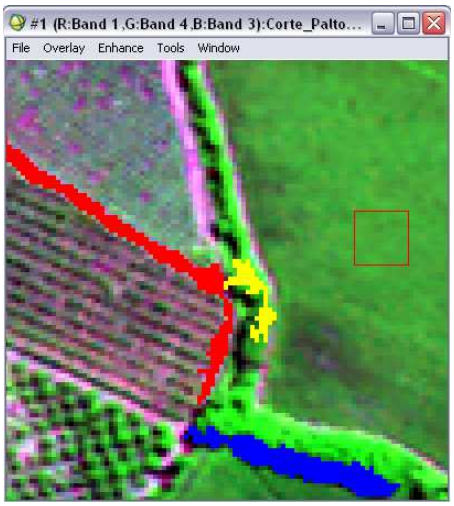


Figura A.24: Resultado de la expansión de las regiones roja, azul y amarilla

Otra gran utilidad es el botón *Stats* que hay en la ventana de *Roi Tool*. Se pueden seleccionar las tres regiones y se obtiene la firma espectral de todas ellas para así poder comprobar la diferencia que hay entre ellas, gracias a que cada gráfica está en el color correspondiente a cada región, como se aprecia en la Figura A.25:

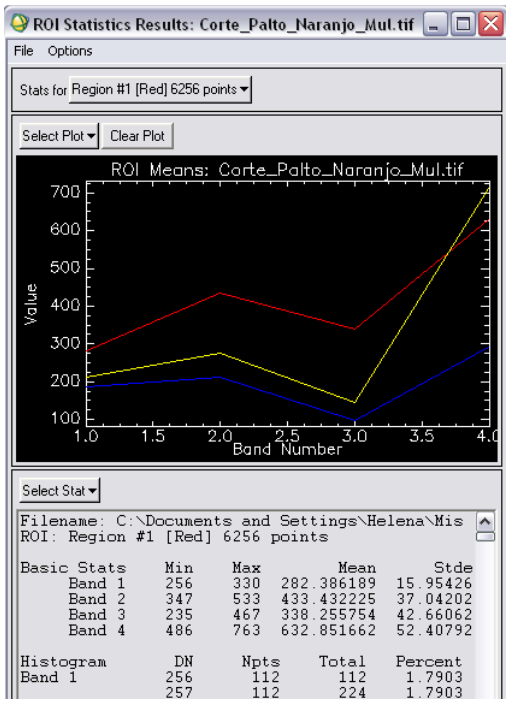


Figura A.25: Firma espectral de las 3 *Roi's*

Otra herramienta muy interesante en clasificación es el *n-D Visualizer* que va a permitir ver la separabilidad entre clases cuando se usan *ROI's* como entradas para la clasificación supervisada. Seleccionar las bandas que se quieran (en este caso la 2 y la 3) y se ven las agrupaciones de *píxeles* por colores según la región a la que pertenezca cada uno. La región roja está muy distanciada de las otras dos, y la azul y la amarilla son similares en una zona, cosa que se puede observar en la imagen inicial (ver Figura A.26).

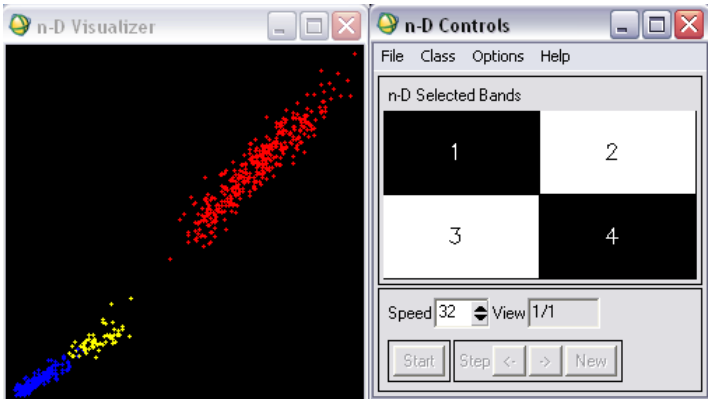


Figura A.26: Respuesta espectral del conjunto de datos en 2 dimensiones

Si se hubieran elegido tres o cuatro bandas, la ventana hubiera permitido seleccionar el botón de *start* y se podría ver cómo se comportan los puntos en un espacio de tres direcciones, como si de una animación se tratase.

4.2. *Maximum Likelihood*

4.2.1. Ejemplo 1

Para poder aplicar este algoritmo, primero se seleccionan un conjunto de *ROI's* de la imagen multiespectral como las que se muestran en la Figura A.27.

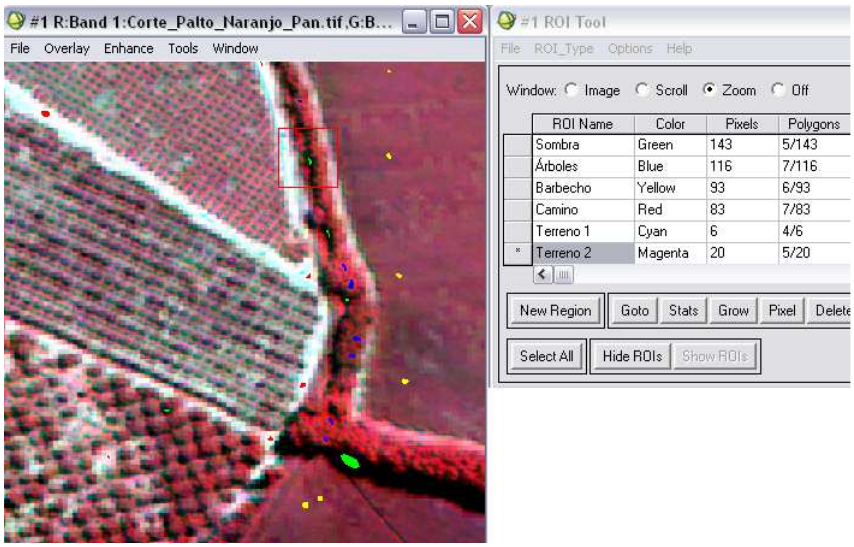


Figura A.27: Selección de un conjunto de *ROI's* de la imagen multiespectral (Ejemplo 1)

Ejecutando el algoritmo de *Maximum Likelihood*, el resultado sería el siguiente:

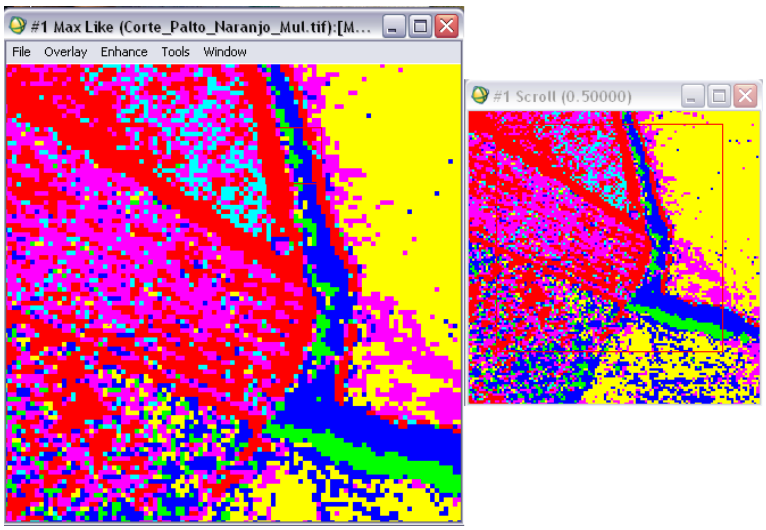


Figura A.28: Resultado de la clasificación *Maximum Likelihood* (Ejemplo 1)

4.2.2. Ejemplo 2

Seleccionar otro conjunto de *ROI's* diferentes como las de la Figura A.29:

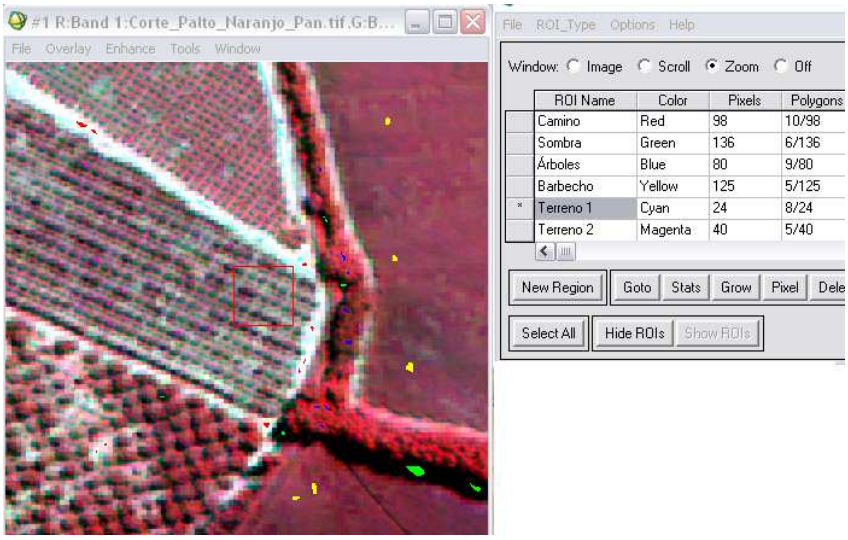


Figura A.29: Selección de un conjunto de *ROI's* de la imagen multiespectral (Ejemplo 2)

Ejecutando el algoritmo de *Maximum Likelihood*, el resultado sería el siguiente:

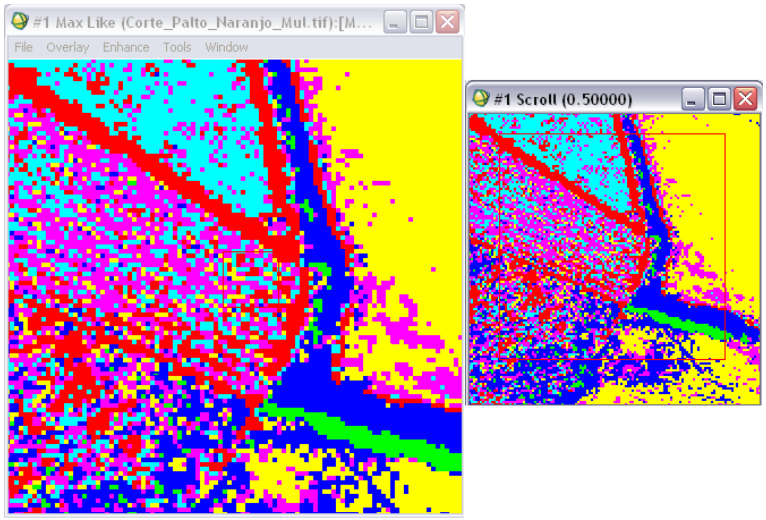


Figura A.30: Resultado de la clasificación *Maximum Likelihood* (Ejemplo 2)

4.2.3. Ejemplo 3

Seleccionar otro conjunto de *ROI's* diferentes, ahora con un gran número de *píxeles* para cada zona, y se obtiene un resultado mucho mejor y más preciso (ver Figura A.31):

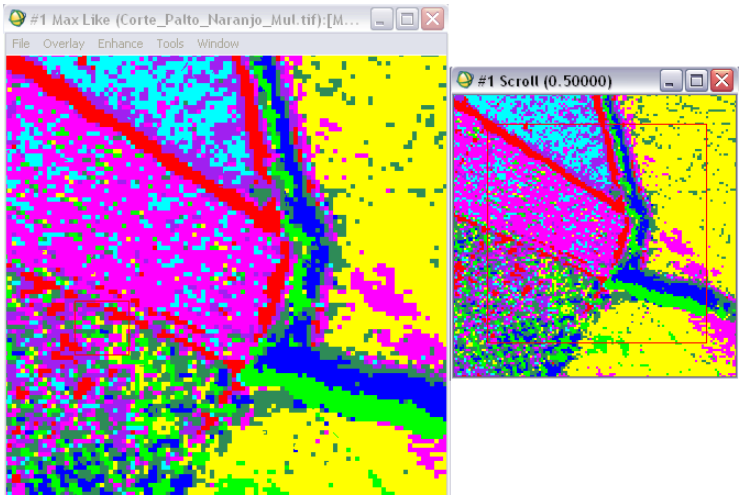


Figura A.31: Resultado de la clasificación *Maximum Likelihood* (Ejemplo 3)

4.2.4. Ejemplo 1, 2, y 3: Comparación de resultados

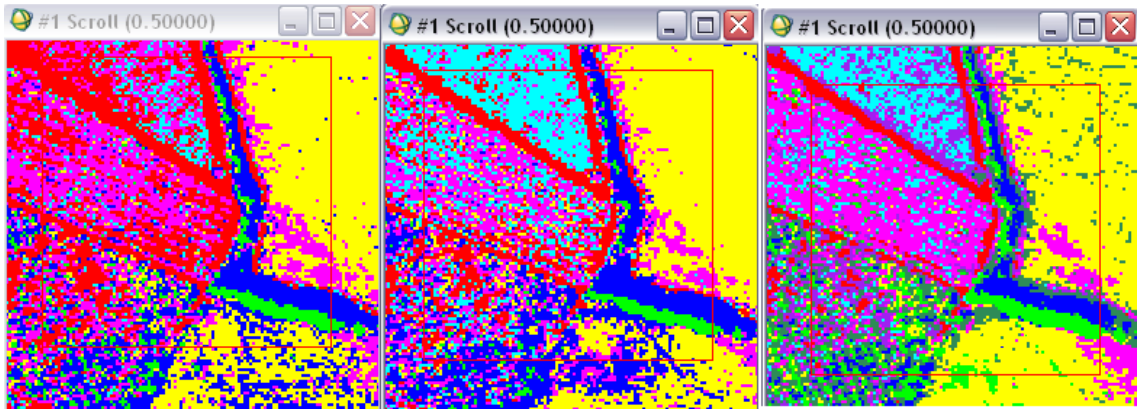


Figura A.32: Comparación resultados de la clasificación *Maximum Likelihood* (Ejemplos 1, 2, 3)

4.3. Redes neuronales

4.3.1. Ejemplo 1

Se parte de la misma imagen multiespectral anterior con las *ROI's* de la Figura A.27 del ejemplo 1 del algoritmo de *Maximum Likelihood*, y el resultado es el siguiente (con función logística) aplicando el algoritmo de *Neural Net* (Figura A.33):

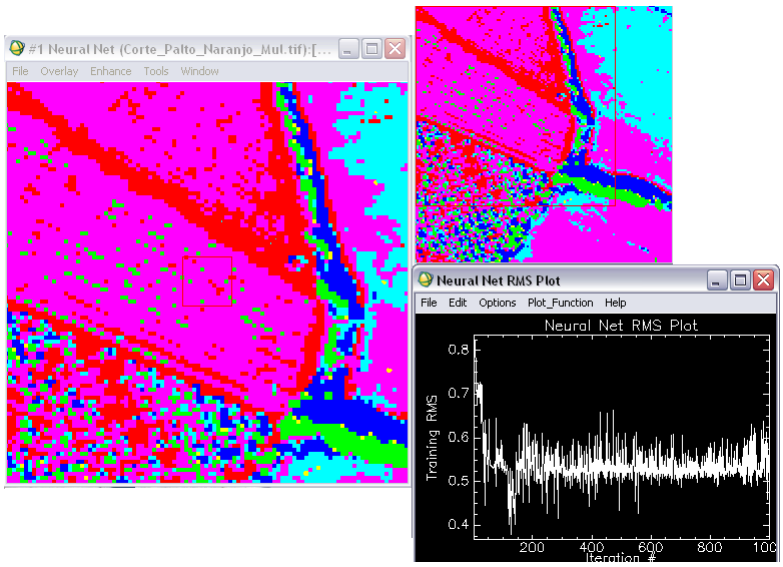


Figura A.33: Resultado de la clasificación *Neural Net* (Ejemplo 1)

Como se observa, difiere mucho con respecto del método de *Maximum Likelihood*, ya que aquí ni si quiera ha detectado la zona de barbecho (amarillo).

4.3.2. Ejemplo 2

Se compara también un segundo caso con las *ROI's* iguales a las de la Figura A.29 del ejemplo 2 de *Maximum Likelihood*, y el resultado es el siguiente aplicando el algoritmo de *Neural Net* (Figura A.34):

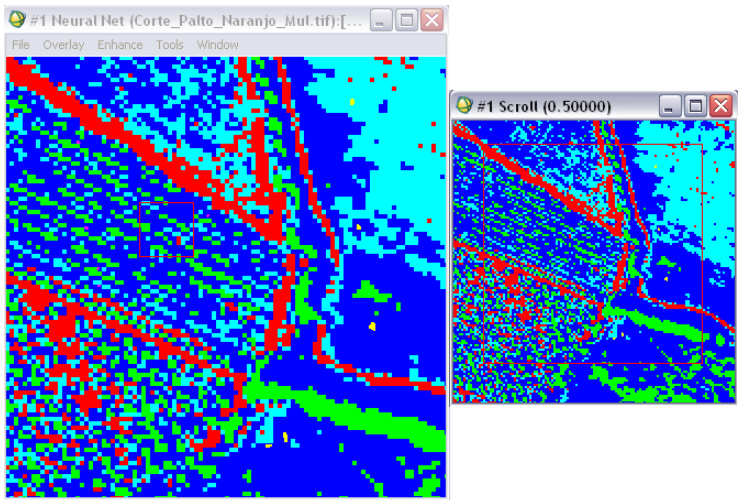
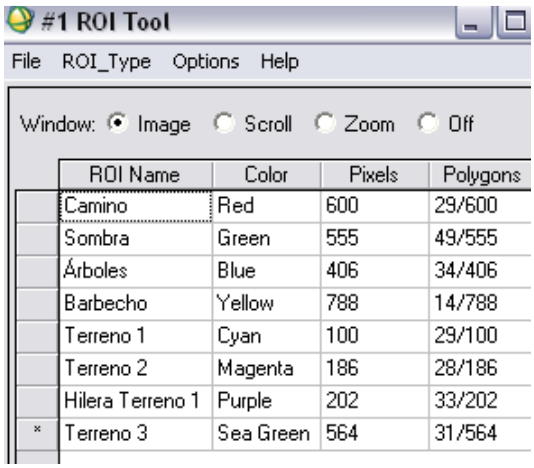


Figura A.34: Resultado de la clasificación *Neural Net* (Ejemplo 2)

Donde no detecta la zona de barbecho ni tampoco el terreno 2 que estaba en color magenta. Con estas *ROI's*, para el algoritmo la mayoría de las zonas son árboles.

4.3.3. Ejemplo 3

Con las *ROI's* iguales a la Figura A.35 (gran número de *píxeles* para cada zona), si se modifica el *Training Threshold Contribution* a 0.4 y se mantiene el número de iteraciones a 1000, se obtiene el Ejemplo 3 de *Neural Net* en la Figura A.36:



	ROI Name	Color	Pixels	Polygons
	Camino	Red	600	29/600
	Sombra	Green	555	49/555
	Árboles	Blue	406	34/406
	Barbecho	Yellow	788	14/788
	Terreno 1	Cyan	100	29/100
	Terreno 2	Magenta	186	28/186
	Hilera Terreno 1	Purple	202	33/202
*	Terreno 3	Sea Green	564	31/564

Figura A.35: Selección de un conjunto de *ROI's* de la imagen multiespectral con gran cantidad de *píxeles*

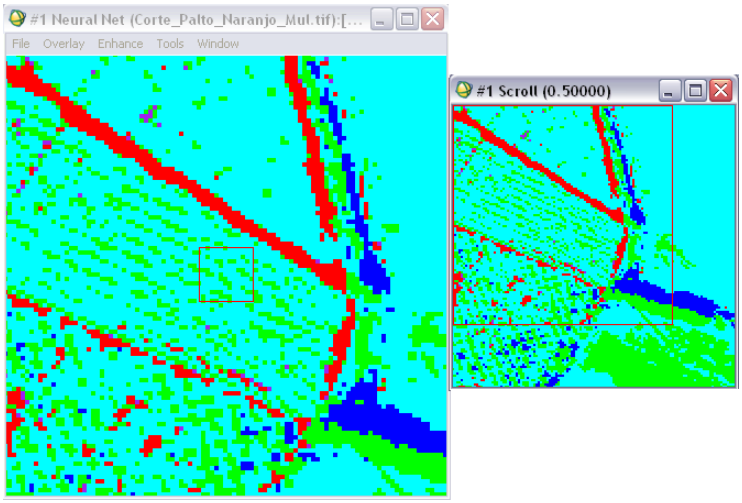


Figura A.36: Resultado de la clasificación *Neural Net* (Ejemplo 3)

4.3.4. Ejemplo 4

Con las mismas *ROI's* de la Figura A.35, si ahora se mantiene el *Training Threshold Contribution* a 0.4 y se modifica el número de iteraciones a 100, se obtiene el Ejemplo 4 de *Neural Net* en la Figura A.37:

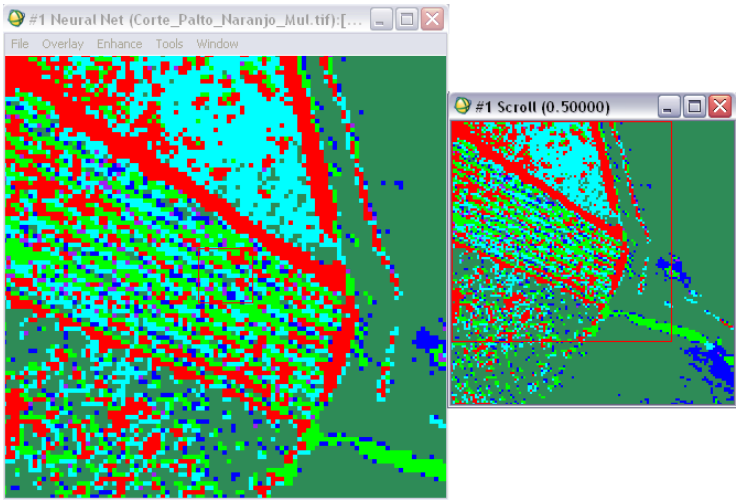


Figura A.37: Resultado de la clasificación *Neural Net* (Ejemplo 4)

4.3.5. Ejemplo 5

Con las mismas *ROI's* de la Figura A.35, con el *Training Threshold Contribution* a 0.2, el número de iteraciones a 100, y se modifica el *Training Rate* a 0.4:

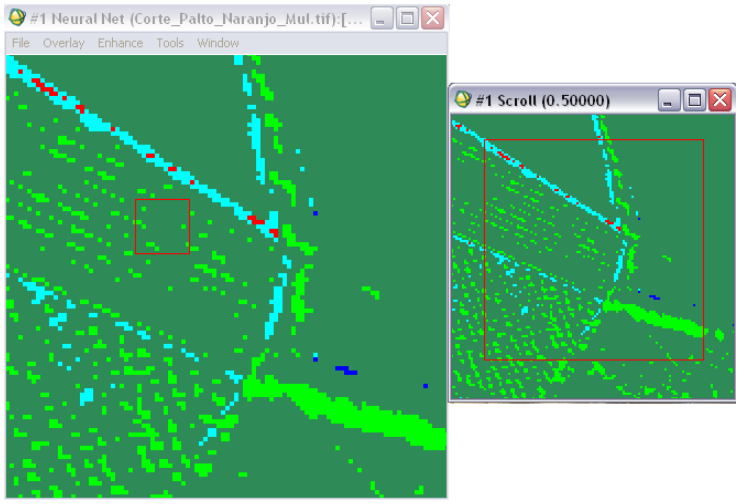


Figura A.38: Resultado de la clasificación *Neural Net* (Ejemplo 5)

4.3.6. Ejemplo 6

Con las mismas *ROI's* de la Figura A.35, el *Training Threshold Contribution* a 0.5, el número de iteraciones a 200, y se modifica el *Training Rate* a 0.2:

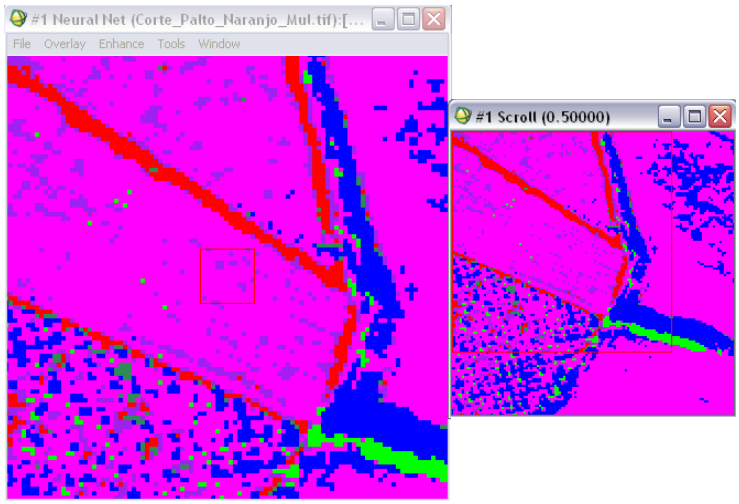


Figura A.39: Resultado de la clasificación *Neural Net* (Ejemplo 6)

4.3.7. Ejemplo 1, 2, 3, 4, 5 y 6: Comparación de resultados

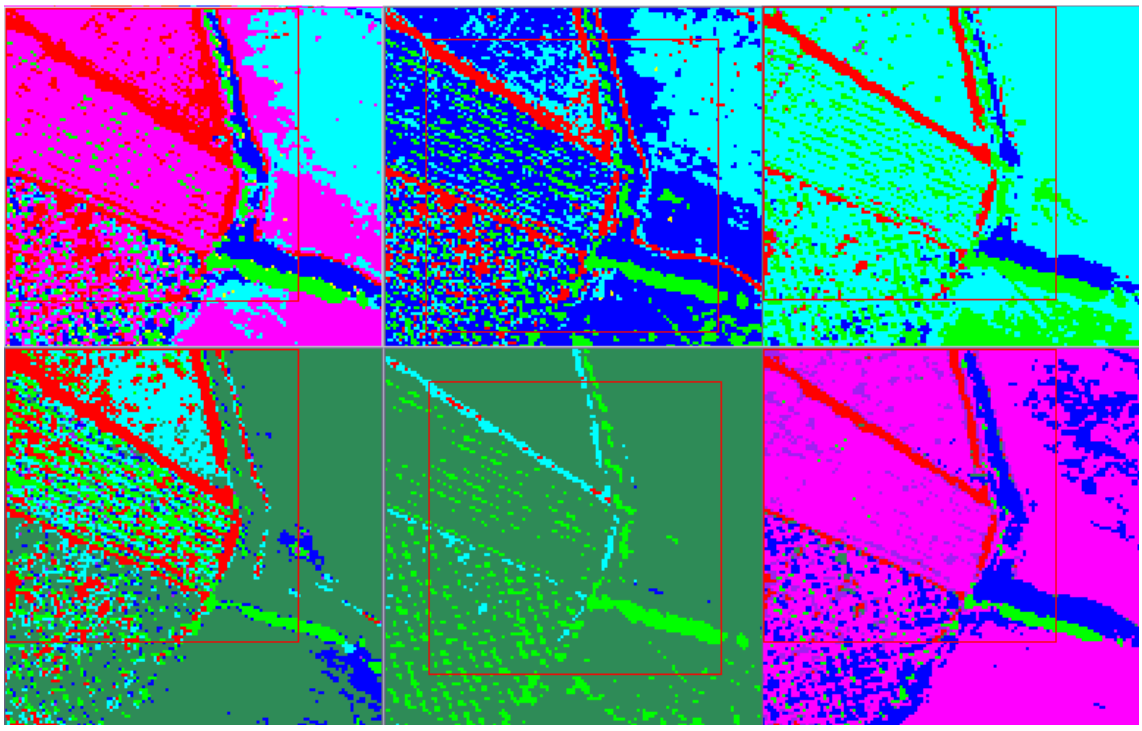


Figura A.40: Comparación resultados de la clasificación *Neural Net* (Ejemplos 1, 2, 3, 4, 5, 67)

5. Matrices de confusión

5.1. Fase de comprobación y verificación de los resultados

Esta fase se dedica a la recopilación de todos los resultados y la generación de un documento gráfico (mapa temático) y tablas estadísticas [Fern 08].

En un sentido estricto, ninguna clasificación puede considerarse completa hasta que su grado de exactitud sea evaluado. Éste puede definirse como el grado de concordancia entre las clases asignadas por el clasificador y sus ubicaciones correctas según datos de tierra recolectados por el usuario y considerados como datos de referencia a tomar en el conjunto de datos de entrenamiento [Tele 11].

Se puede evaluar una estimación teórica del error en función de las características del algoritmo de clasificación o analizar una serie de áreas test obtenidas del mismo

modo que las áreas de entrenamiento. El segundo modo de proceder permite obtener una estimación más realista de los errores mientras la muestra de *píxeles* para la estimación del error sea lo suficientemente grande y representativo. Un método simple y apropiado de evaluaciones de los errores es utilizar la **matriz de confusión** de clases. Con este tipo de análisis, se obtiene no sólo una caracterización del error cometido, sino también una medida sobre la adecuación de las clases consideradas a la realidad y de los parámetros utilizados para caracterizarlas. Puede por tanto utilizarse para definir un nuevo conjunto de clases para realizar una clasificación [UM].

Dicha matriz muestra la relación entre dos series de medidas correspondientes al área en estudio. La primera serie corresponde a datos de referencia adquiridos de observaciones de campo, inspección de estadísticas agrícolas, interpretación de fotos aéreas y otras fuentes similares. La segunda corresponde a la categorización de los *píxeles* realizada por el clasificador para las clases de interés. En una matriz de confusión las columnas corresponden a los datos de referencia, mientras que las filas corresponden a las asignaciones del clasificador [Tele 11]. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real [Wikic]. Cuanto mayor sean los valores contenidos en la diagonal principal con respecto a los del resto de la matriz, más fiable será la clasificación realizada. Al final los resultados se expresan en porcentajes de acierto para cada clase y totales [Itur 98]. Uno de los beneficios de las matrices de confusión es que facilitan ver si el sistema está confundiendo dos clases [Wikic].

Si en los datos de entrada el número de muestras de clases diferentes cambia mucho la tasa de error del clasificador, no es representativa de lo bien que realiza la tarea el clasificador. Si por ejemplo hay 990 muestras de la clase 1 y sólo 10 de la clase 2, el clasificador puede tener fácilmente un sesgo hacia la clase 1. Si el clasificador clasifica todas las muestras como clase 1 su precisión será del 99%. Esto no significa que sea un buen clasificador, pues tuvo un 100% de error en la clasificación de las muestras de la clase 2 [Wikic].

A partir de la matriz de confusión se pueden calcular otras medidas de interés desde el punto de vista de la exactitud [Man 10]:

- **Exactitud global (*overall accuracy*):**

  Se calcula dividiendo el número total de *píxeles* correctamente clasificados por el número total de pixeles de referencia y expresándolo como porcentaje.

- **Exactitud del usuario (*user's accuracy*):**

  Se calcula dividiendo el número de *píxeles* correctamente clasificados en cada categoría por el número total de *píxeles* que fueron clasificados en dicha categoría (total de la fila).

- **Exactitud de productor (*producer's accuracy*):**

  Resulta de dividir el número de *píxeles* correctamente clasificados en cada categoría por el número de *píxeles* de referencia utilizados para dicha categoría (total de la columna).

Es una herramienta de visualización que se emplea en aprendizaje supervisado.

Se tienen dos versiones equivalentes de la matriz: una cuyas entradas se expresan en *píxeles* y otra en que se expresan como porcentajes. Obsérvese que además de las clases de interés se introduce una columna y fila correspondiente a los *píxeles* que no pudieron ser clasificados [Man 10].

5.2. Matriz de confusión para Ejemplo 2 de *Maximum Likelihood*

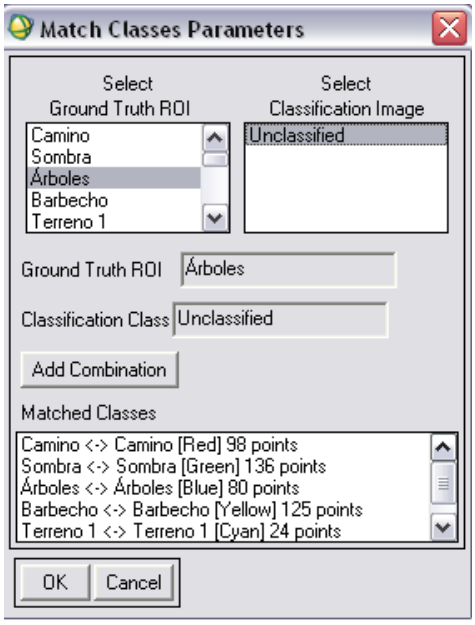
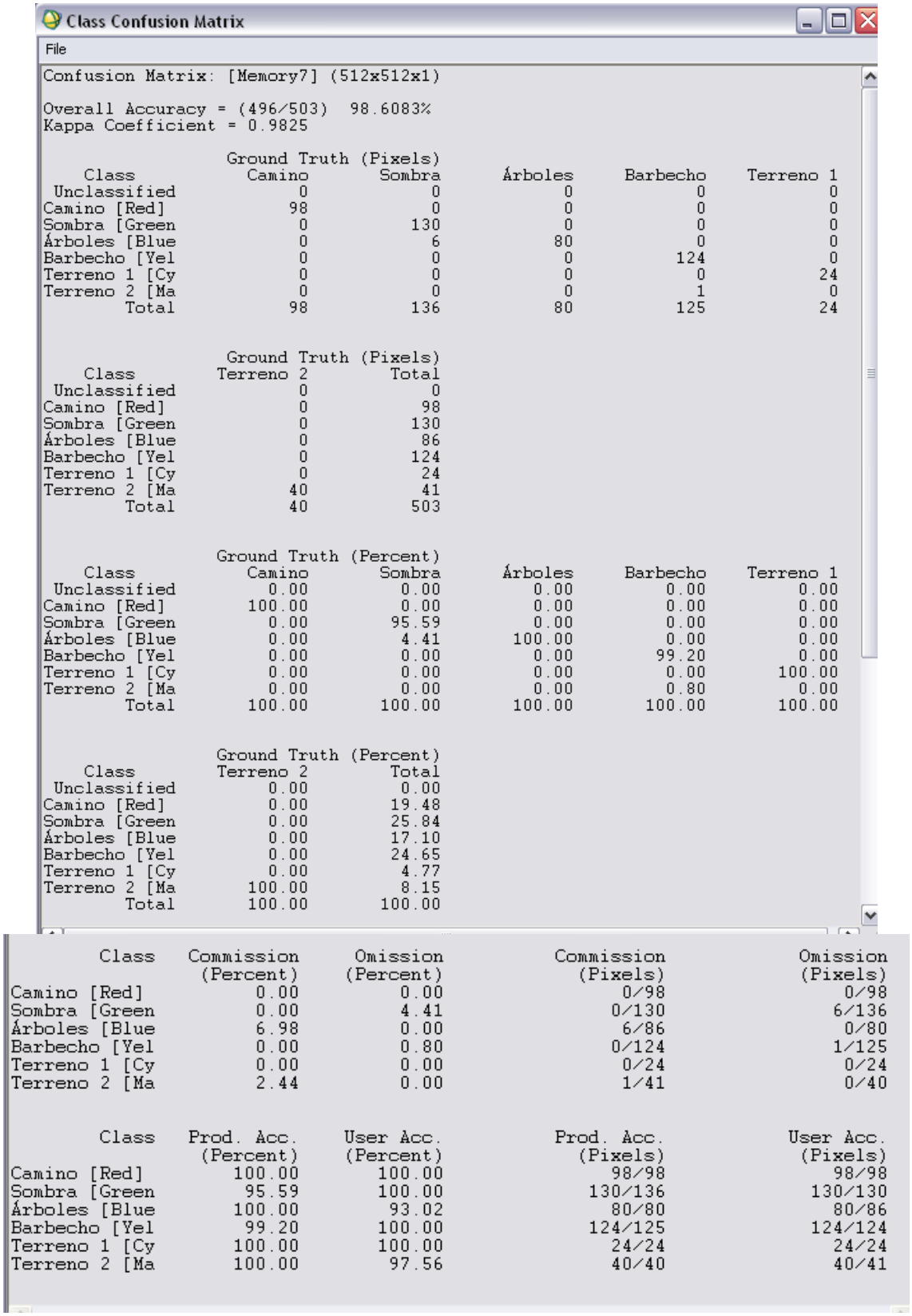


Figura A.41: Parámetros de la matriz de confusión

Estableciendo los parámetros para el cálculo de la matriz en la Figura A.41, se puede observar en la Figura A.42 que la zona de árboles y el terreno 2 es donde más dificultades ha tenido para detectarlos ya que han confundido respectivamente, con sombra y barbecho. Pero por lo general los resultados son buenos.



Class Confusion Matrix

File

Confusion Matrix: [Memory7] (512x512x1)

Overall Accuracy = (496/503) 98.6083%
Kappa Coefficient = 0.9825

Class	Ground Truth (Pixels) Camino	Sombra	Árboles	Barbecho	Terreno 1
Unclassified	0	0	0	0	0
Camino [Red]	98	0	0	0	0
Sombra [Green	0	130	0	0	0
Árboles [Blue	0	6	80	0	0
Barbecho [Yel	0	0	0	124	0
Terreno 1 [Cy	0	0	0	0	24
Terreno 2 [Ma	0	0	0	1	0
Total	98	136	80	125	24

Class	Ground Truth (Pixels) Terreno 2	Total
Unclassified	0	0
Camino [Red]	0	98
Sombra [Green	0	130
Árboles [Blue	0	86
Barbecho [Yel	0	124
Terreno 1 [Cy	0	24
Terreno 2 [Ma	40	41
Total	40	503

Class	Ground Truth (Percent) Camino	Sombra	Árboles	Barbecho	Terreno 1
Unclassified	0.00	0.00	0.00	0.00	0.00
Camino [Red]	100.00	0.00	0.00	0.00	0.00
Sombra [Green	0.00	95.59	0.00	0.00	0.00
Árboles [Blue	0.00	4.41	100.00	0.00	0.00
Barbecho [Yel	0.00	0.00	0.00	99.20	0.00
Terreno 1 [Cy	0.00	0.00	0.00	0.00	100.00
Terreno 2 [Ma	0.00	0.00	0.00	0.80	0.00
Total	100.00	100.00	100.00	100.00	100.00

Class	Ground Truth (Percent) Terreno 2	Total
Unclassified	0.00	0.00
Camino [Red]	0.00	19.48
Sombra [Green	0.00	25.84
Árboles [Blue	0.00	17.10
Barbecho [Yel	0.00	24.65
Terreno 1 [Cy	0.00	4.77
Terreno 2 [Ma	100.00	8.15
Total	100.00	100.00

Class	Commission (Percent)	Omission (Percent)	Commission (Pixels)	Omission (Pixels)
Camino [Red]	0.00	0.00	0/98	0/98
Sombra [Green	0.00	4.41	0/130	6/136
Árboles [Blue	6.98	0.00	6/86	0/80
Barbecho [Yel	0.00	0.80	0/124	1/125
Terreno 1 [Cy	0.00	0.00	0/24	0/24
Terreno 2 [Ma	2.44	0.00	1/41	0/40

Class	Prod. Acc. (Percent)	User Acc. (Percent)	Prod. Acc. (Pixels)	User Acc. (Pixels)
Camino [Red]	100.00	100.00	98/98	98/98
Sombra [Green	95.59	100.00	130/136	130/130
Árboles [Blue	100.00	93.02	80/80	80/86
Barbecho [Yel	99.20	100.00	124/125	124/124
Terreno 1 [Cy	100.00	100.00	24/24	24/24
Terreno 2 [Ma	100.00	97.56	40/40	40/41

Figura A.42: Matriz de confusión para el ejemplo 2 de *maximum likelihood*

5.3. Matriz de confusión para Ejemplo 3 de *Maximum Likelihood*

Este resultado es mucho más realista según se observa en los resultados obtenidos en la Figura A.43:

Confusion Matrix: [Memory6] (512x512x1)

Overall Accuracy = (2474/2928) 84.4945%
Kappa Coefficient = 0.8124

	Ground Truth (Pixels)				
Class	Sombra	Árboles	Barbecho	Terreno 1	Terreno 2
Unclassified	0	0	0	0	0
Sombra [Green	519	10	0	0	2
Árboles [Blue	12	333	7	0	0
Barbecho [Yel	13	2	753	0	8
Terreno 1 [Cy	0	0	0	85	13
Terreno 2 [Ma	1	0	0	1	157
Hilera Terren	0	0	0	14	0
Terreno 3 [Se	10	61	28	0	6
Camino [Red]	0	0	0	0	0
Total	555	406	788	100	186

	Ground Truth (Pixels)			
Class	Hilera Terren	Terreno 3	Camino	Total
Unclassified	0	0	0	0
Sombra [Green	0	6	0	537
Árboles [Blue	0	170	0	522
Barbecho [Yel	0	20	0	796
Terreno 1 [Cy	32	5	0	135
Terreno 2 [Ma	5	8	0	172
Hilera Terren	162	17	0	193
Terreno 3 [Se	0	338	0	443
Camino [Red]	3	0	127	130
Total	202	564	127	2928

	Ground Truth (Percent)				
Class	Sombra	Árboles	Barbecho	Terreno 1	Terreno 2
Unclassified	0.00	0.00	0.00	0.00	0.00
Sombra [Green	93.51	2.46	0.00	0.00	1.08
Árboles [Blue	2.16	82.02	0.89	0.00	0.00
Barbecho [Yel	2.34	0.49	95.56	0.00	4.30
Terreno 1 [Cy	0.00	0.00	0.00	85.00	6.99
Terreno 2 [Ma	0.18	0.00	0.00	1.00	84.41
Hilera Terren	0.00	0.00	0.00	14.00	0.00
Terreno 3 [Se	1.80	15.02	3.55	0.00	3.23
Camino [Red]	0.00	0.00	0.00	0.00	0.00
Total	100.00	100.00	100.00	100.00	100.00

	Ground Truth (Percent)			
Class	Hilera Terren	Terreno 3	Camino	Total
Unclassified	0.00	0.00	0.00	0.00
Sombra [Green	0.00	1.06	0.00	18.34
Árboles [Blue	0.00	30.14	0.00	17.83
Barbecho [Yel	0.00	3.55	0.00	27.19
Terreno 1 [Cy	15.84	0.89	0.00	4.61
Terreno 2 [Ma	2.48	1.42	0.00	5.87
Hilera Terren	80.20	3.01	0.00	6.59
Terreno 3 [Se	0.00	59.93	0.00	15.13
Camino [Red]	1.49	0.00	100.00	4.44
Total	100.00	100.00	100.00	100.00

Class	Commission (Percent)	Omission (Percent)	Commission (Pixels)	Omission (Pixels)
Sombra [Green	3.35	6.49	18/537	36/555
Árboles [Blue	36.21	17.98	189/522	73/406
Barbecho [Yel	5.40	4.44	43/796	35/788
Terreno 1 [Cy	37.04	15.00	50/135	15/100
Terreno 2 [Ma	8.72	15.59	15/172	29/186
Hilera Terren	16.06	19.80	31/193	40/202
Terreno 3 [Se	23.70	40.07	105/443	226/564
Camino [Red]	2.31	0.00	3/130	0/127

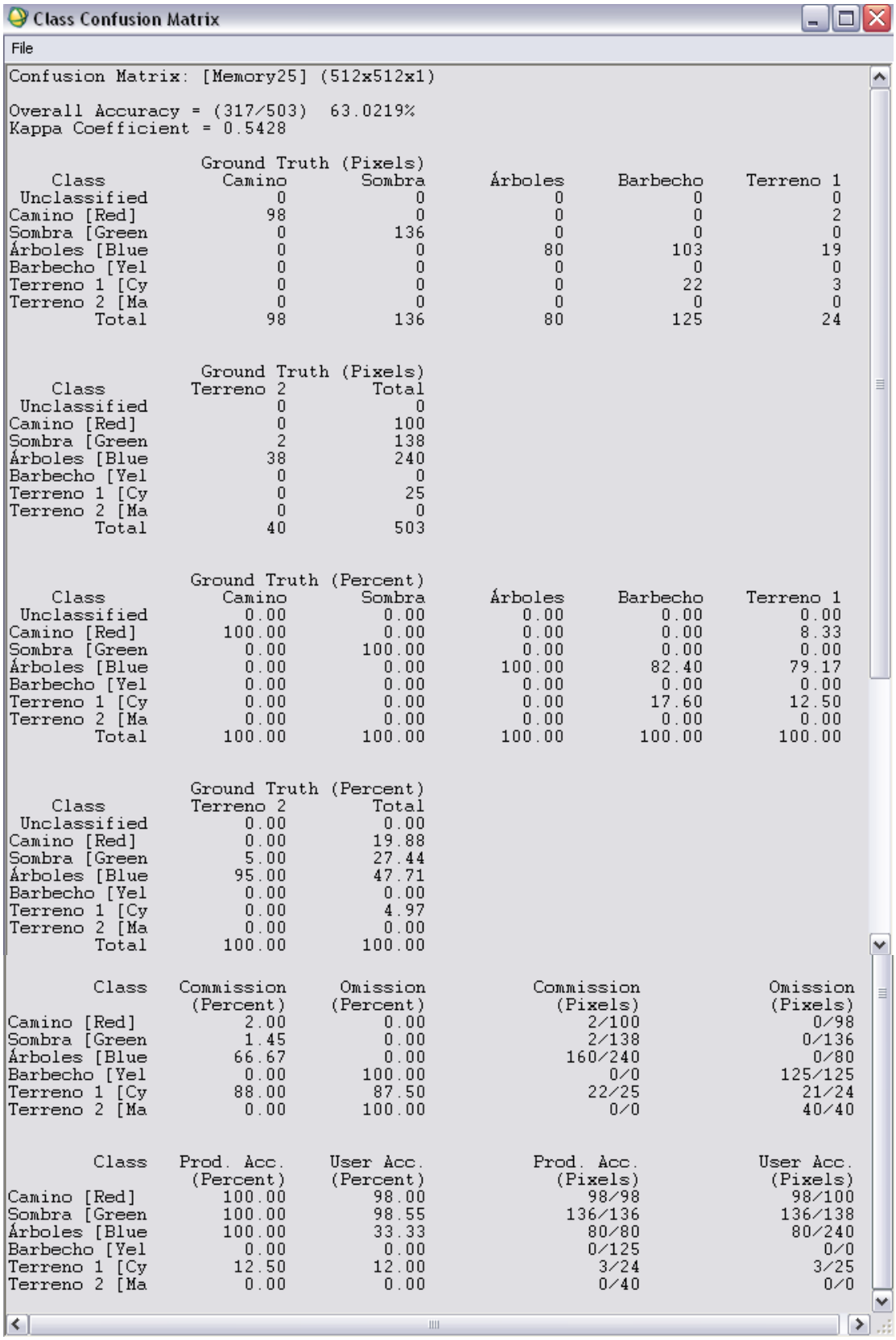
Class	Prod. Acc. (Percent)	User Acc. (Percent)	Prod. Acc. (Pixels)	User Acc. (Pixels)
Sombra [Green	93.51	96.65	519/555	519/537
Árboles [Blue	82.02	63.79	333/406	333/522
Barbecho [Yel	95.56	94.60	753/788	753/796
Terreno 1 [Cy	85.00	62.96	85/100	85/135
Terreno 2 [Ma	84.41	91.28	157/186	157/172
Hilera Terren	80.20	83.94	162/202	162/193
Terreno 3 [Se	59.93	76.30	338/564	338/443
Camino [Red]	100.00	97.69	127/127	127/130

:

Figura A.43: Matriz de confusión para el ejemplo 3 de *maximum likelihood*

5.4. Matriz de confusión para Ejemplo 2 de *Neural Net*

Se procede de igual forma con el algoritmo de *Neural Net* en ejemplo 2 visto previamente, haciendo coincidir la *ROI* de clasificación con la *ROI* de postclasificación (*roi_de_palto.roi*), y se obtiene el resultado de la Figura A.44:



Class Confusion Matrix

File

Confusion Matrix: [Memory25] (512x512x1)

Overall Accuracy = (317/503) 63.0219%
Kappa Coefficient = 0.5428

	Ground Truth (Pixels)				
Class	Camino	Sombra	Árboles	Barbecho	Terreno 1
Unclassified	0	0	0	0	0
Camino [Red]	98	0	0	0	2
Sombra [Green	0	136	0	0	0
Árboles [Blue	0	0	80	103	19
Barbecho [Yel	0	0	0	0	0
Terreno 1 [Cy	0	0	0	22	3
Terreno 2 [Ma	0	0	0	0	0
Total	98	136	80	125	24

	Ground Truth (Pixels)	
Class	Terreno 2	Total
Unclassified	0	0
Camino [Red]	0	100
Sombra [Green	2	138
Árboles [Blue	38	240
Barbecho [Yel	0	0
Terreno 1 [Cy	0	25
Terreno 2 [Ma	0	0
Total	40	503

	Ground Truth (Percent)				
Class	Camino	Sombra	Árboles	Barbecho	Terreno 1
Unclassified	0.00	0.00	0.00	0.00	0.00
Camino [Red]	100.00	0.00	0.00	0.00	8.33
Sombra [Green	0.00	100.00	0.00	0.00	0.00
Árboles [Blue	0.00	0.00	100.00	82.40	79.17
Barbecho [Yel	0.00	0.00	0.00	0.00	0.00
Terreno 1 [Cy	0.00	0.00	0.00	17.60	12.50
Terreno 2 [Ma	0.00	0.00	0.00	0.00	0.00
Total	100.00	100.00	100.00	100.00	100.00

	Ground Truth (Percent)	
Class	Terreno 2	Total
Unclassified	0.00	0.00
Camino [Red]	0.00	19.88
Sombra [Green	5.00	27.44
Árboles [Blue	95.00	47.71
Barbecho [Yel	0.00	0.00
Terreno 1 [Cy	0.00	4.97
Terreno 2 [Ma	0.00	0.00
Total	100.00	100.00

Class	Commission (Percent)	Omission (Percent)	Commission (Pixels)	Omission (Pixels)
Camino [Red]	2.00	0.00	2/100	0/98
Sombra [Green	1.45	0.00	2/138	0/136
Árboles [Blue	66.67	0.00	160/240	0/80
Barbecho [Yel	0.00	100.00	0/0	125/125
Terreno 1 [Cy	88.00	87.50	22/25	21/24
Terreno 2 [Ma	0.00	100.00	0/0	40/40

Class	Prod. Acc. (Percent)	User Acc. (Percent)	Prod. Acc. (Pixels)	User Acc. (Pixels)
Camino [Red]	100.00	98.00	98/98	98/100
Sombra [Green	100.00	98.55	136/136	136/138
Árboles [Blue	100.00	33.33	80/80	80/240
Barbecho [Yel	0.00	0.00	0/125	0/0
Terreno 1 [Cy	12.50	12.00	3/24	3/25
Terreno 2 [Ma	0.00	0.00	0/40	0/0

Figura A.44: Matriz de confusión para el ejemplo 2 de *neural net*

5.5. Matriz de confusión para Ejemplo 4 de *Neural Net*

Ahora para el ejemplo 4 de *Neural Net*, se parte de la siguiente *ROI* (Figura A.45) y se obtiene la imagen (Figura A.45) con la clasificación del algoritmo:

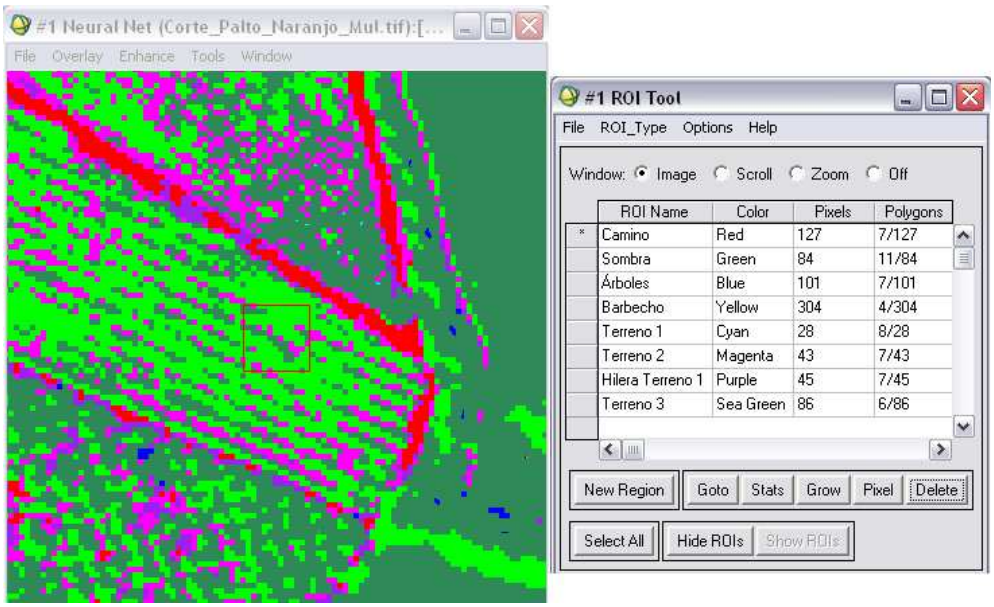


Figura A.45: Imagen clasificada mediante *Neural Net* junto con la *ROI* empleada

En este caso, se ha seleccionado para la *ROI* de la clasificación (*roi_postclasificacion.roi)* un número de *píxeles* para cada zona relativamente inferior al que se va a utilizar ahora para hacer la postclasificación, en el que se van a tomar muchas más muestras de *píxeles* (*roi_completa.roi*) como se puede observar en la Figura A.46.

La matriz de confusión queda por tanto, de la siguiente manera (Figura A.47):

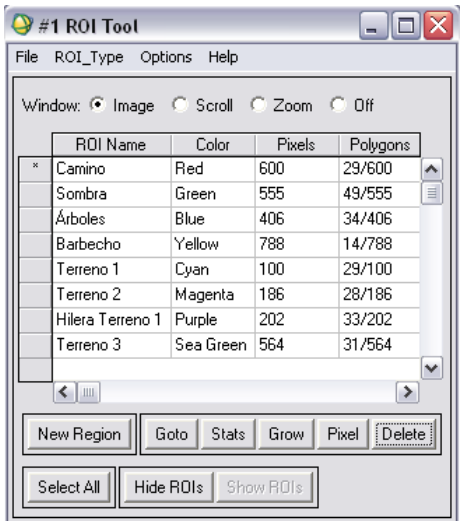


Figura A.46: *ROI* empleada para la postclasificación

Class Confusion Matrix

File

Confusion Matrix: [Memory19] (512x512x1)

Overall Accuracy = (1610/3401) 47.3390%
Kappa Coefficient = 0.3722

Ground Truth (Pixels)

Class	Camino	Sombra	Árboles	Barbecho	Terreno 1
Unclassified	0	0	0	0	0
Camino [Red]	536	0	0	0	0
Sombra [Green	0	494	1	0	19
Árboles [Blue	0	0	0	0	0
Barbecho [Yel	0	0	0	0	0
Terreno 1 [Cy	0	0	0	0	0
Terreno 2 [Ma	7	0	0	0	19
Hilera Terren	57	0	0	0	0
Terreno 3 [Se	0	61	405	788	62
Total	600	555	406	788	100

Ground Truth (Pixels)

Class	Terreno 2	Hilera Terren	Terreno 3	Total
Unclassified	0	0	0	0
Camino [Red]	0	0	0	536
Sombra [Green	54	4	11	583
Árboles [Blue	0	0	0	0
Barbecho [Yel	0	0	0	0
Terreno 1 [Cy	0	0	0	0
Terreno 2 [Ma	26	72	0	124
Hilera Terren	0	1	0	58
Terreno 3 [Se	106	125	553	2100
Total	186	202	564	3401

Ground Truth (Percent)

Class	Camino	Sombra	Árboles	Barbecho	Terreno 1
Unclassified	0.00	0.00	0.00	0.00	0.00
Camino [Red]	89.33	0.00	0.00	0.00	0.00
Sombra [Green	0.00	89.01	0.25	0.00	19.00
Árboles [Blue	0.00	0.00	0.00	0.00	0.00
Barbecho [Yel	0.00	0.00	0.00	0.00	0.00
Terreno 1 [Cy	0.00	0.00	0.00	0.00	0.00
Terreno 2 [Ma	1.17	0.00	0.00	0.00	19.00
Hilera Terren	9.50	0.00	0.00	0.00	0.00
Terreno 3 [Se	0.00	10.99	99.75	100.00	62.00
Total	100.00	100.00	100.00	100.00	100.00

	Ground Truth (Percent)			
Class	Terreno 2	Hilera Terren	Terreno 3	Total
Unclassified	0.00	0.00	0.00	0.00
Camino [Red]	0.00	0.00	0.00	15.76
Sombra [Green	29.03	1.98	1.95	17.14
Árboles [Blue	0.00	0.00	0.00	0.00
Barbecho [Yel	0.00	0.00	0.00	0.00
Terreno 1 [Cy	0.00	0.00	0.00	0.00
Terreno 2 [Ma	13.98	35.64	0.00	3.65
Hilera Terren	0.00	0.50	0.00	1.71
Terreno 3 [Se	56.99	61.88	98.05	61.75
Total	100.00	100.00	100.00	100.00

Class	Commission (Percent)	Omission (Percent)	Commission (Pixels)	Omission (Pixels)
Camino [Red]	0.00	10.67	0/536	64/600
Sombra [Green	15.27	10.99	89/583	61/555
Árboles [Blue	0.00	100.00	0/0	406/406
Barbecho [Yel	0.00	100.00	0/0	788/788
Terreno 1 [Cy	0.00	100.00	0/0	100/100
Terreno 2 [Ma	79.03	86.02	98/124	160/186
Hilera Terren	98.28	99.50	57/58	201/202
Terreno 3 [Se	73.67	1.95	1547/2100	11/564

Class	Prod. Acc. (Percent)	User Acc. (Percent)	Prod. Acc. (Pixels)	User Acc. (Pixels)
Camino [Red]	89.33	100.00	536/600	536/536
Sombra [Green	89.01	84.73	494/555	494/583
Árboles [Blue	0.00	0.00	0/406	0/0
Barbecho [Yel	0.00	0.00	0/788	0/0
Terreno 1 [Cy	0.00	0.00	0/100	0/0
Terreno 2 [Ma	13.98	20.97	26/186	26/124
Hilera Terren	0.50	1.72	1/202	1/58
Terreno 3 [Se	98.05	26.33	553/564	553/2100

Figura A.47: Matriz de confusión para el ejemplo 4 de *neural net* (a)

Ahora con este último ejemplo, se va a hacer en sentido inverso, es decir, se va a llevar a cabo la clasificación con un número elevado de *píxeles* (*roi_completa.roi*) (ver Figura A.48) y la postclasificación se va a hacer con un menor número de *píxeles* (*roi_postclasificacion.roi*):

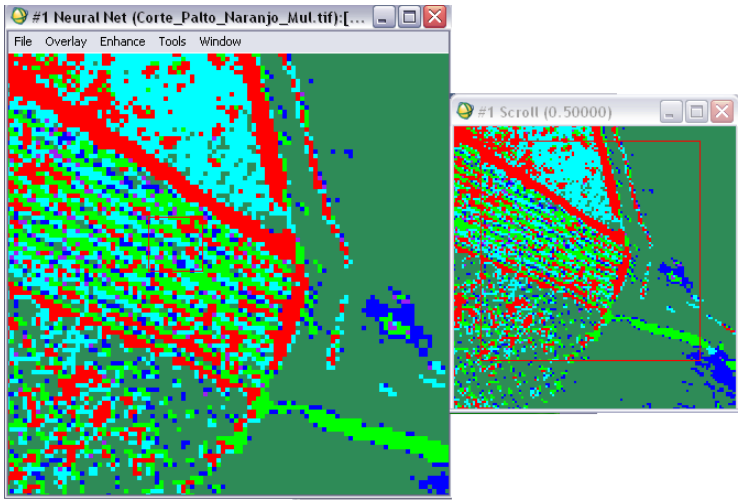


Figura A.48: Imagen clasificada mediante *Neural Net* con la *ROI* de la Figura A.46

Class Confusion Matrix

File

Confusion Matrix: [Memory9] (512x512x1)

Overall Accuracy = (295/818) 36.0636%
Kappa Coefficient = 0.2858

Ground Truth (Pixels)

Class	Camino	Sombra	Árboles	Barbecho	Terreno 1
Unclassified	0	0	0	0	0
Camino [Red]	127	0	0	0	0
Sombra [Green	0	62	0	0	0
Árboles [Blue	0	12	0	0	0
Barbecho [Yel	0	0	0	0	0
Terreno 1 [Cy	0	0	0	0	23
Terreno 2 [Ma	0	0	0	0	0
Hilera Terren	0	0	0	0	0
Terreno 3 [Se	0	10	101	304	5
Total	127	84	101	304	28

Ground Truth (Pixels)

Class	Terreno 2	Hilera Terren	Terreno 3	Total
Unclassified	0	0	0	0
Camino [Red]	0	1	0	128
Sombra [Green	0	0	0	62
Árboles [Blue	7	0	0	19
Barbecho [Yel	0	0	0	0
Terreno 1 [Cy	31	38	3	95
Terreno 2 [Ma	0	0	0	0
Hilera Terren	3	0	0	3
Terreno 3 [Se	2	6	83	511
Total	43	45	86	818

Ground Truth (Percent)

Class	Camino	Sombra	Árboles	Barbecho	Terreno 1
Unclassified	0.00	0.00	0.00	0.00	0.00
Camino [Red]	100.00	0.00	0.00	0.00	0.00
Sombra [Green	0.00	73.81	0.00	0.00	0.00
Árboles [Blue	0.00	14.29	0.00	0.00	0.00
Barbecho [Yel	0.00	0.00	0.00	0.00	0.00
Terreno 1 [Cy	0.00	0.00	0.00	0.00	82.14
Terreno 2 [Ma	0.00	0.00	0.00	0.00	0.00
Hilera Terren	0.00	0.00	0.00	0.00	0.00
Terreno 3 [Se	0.00	11.90	100.00	100.00	17.86
Total	100.00	100.00	100.00	100.00	100.00

	Ground Truth (Percent)			
Class	Terreno 2	Hilera Terren	Terreno 3	Total
Unclassified	0.00	0.00	0.00	0.00
Camino [Red]	0.00	2.22	0.00	15.65
Sombra [Green	0.00	0.00	0.00	7.58
Árboles [Blue	16.28	0.00	0.00	2.32
Barbecho [Yel	0.00	0.00	0.00	0.00
Terreno 1 [Cy	72.09	84.44	3.49	11.61
Terreno 2 [Ma	0.00	0.00	0.00	0.00
Hilera Terren	6.98	0.00	0.00	0.37
Terreno 3 [Se	4.65	13.33	96.51	62.47
Total	100.00	100.00	100.00	100.00

Class	Commission (Percent)	Omission (Percent)	Commission (Pixels)	Omission (Pixels)
Camino [Red]	0.78	0.00	1/128	0/127
Sombra [Green	0.00	26.19	0/62	22/84
Árboles [Blue	100.00	100.00	19/19	101/101
Barbecho [Yel	0.00	100.00	0/0	304/304
Terreno 1 [Cy	75.79	17.86	72/95	5/28
Terreno 2 [Ma	0.00	100.00	0/0	43/43
Hilera Terren	100.00	100.00	3/3	45/45
Terreno 3 [Se	83.76	3.49	428/511	3/86

Class	Prod. Acc. (Percent)	User Acc. (Percent)	Prod. Acc. (Pixels)	User Acc. (Pixels)
Camino [Red]	100.00	99.22	127/127	127/128
Sombra [Green	73.81	100.00	62/84	62/62
Árboles [Blue	0.00	0.00	0/101	0/19
Barbecho [Yel	0.00	0.00	0/304	0/0
Terreno 1 [Cy	82.14	24.21	23/28	23/95
Terreno 2 [Ma	0.00	0.00	0/43	0/0
Hilera Terren	0.00	0.00	0/45	0/3
Terreno 3 [Se	96.51	16.24	83/86	83/511

Figura A.49: Matriz de confusión para el ejemplo 4 de *neural net* (b)

La matriz de confusión queda de la siguiente manera (ver Figura A.49), con peores valores para el *overall* y el *kappa*.