UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

Clasificación de imágenes de satélite de alta dimensionalidad

Autor:
Jorge Alberto Suárez Garea

Directores:
Francisco Argüello Pedreira
Dora Blanco Heras

Grado en Ingeniería Informática

Septiembre 2014

Trabajo de Fin de Grado presentado en la Escola Técnica Superior de Enxeñaría de la Universidade de Santiago de Compostela para la obtención del Grado en Enxeñaría Informática

D. Francisco Argüello Pedreira, Profesor do Departamento de Electrónica e Computación da Universidade de Santiago de Compostela, e **Dna. Dora Blanco Heras**, Profesora do Departamento de Electrónica e Computación da Universidade de Santiago de Compostela,

INFORMAN:

Que a presente memoria, titulada *Clasificación de imágenes de satélite de alta dimensionalidad*, presentada por **D. Jorge Alberto Suárez Garea** para superar os créditos correspondentes ao Traballo de Fin de Grao da titulación de Grao en Enxeñaría Informática, realizouse baixo nosa dirección no Departamento de Electrónica e Computación da Universidade de Santiago de Compostela.

E para que así conste aos efectos oportunos, expiden o presente informe en Santiago de Compostela, a 5-9-2014:

O director, A codirectora, O alumno,

Francisco Argüello Pedreira Dora Blanco Heras Jorge Alberto Suárez Garea

Agradecimientos

A mi mujer y mis hijas por el apoyo que me han dado durante todo este tiempo y a la paciencia que han demostrado tener.

A mi abuelo Ángel, desde donde quiera que nos vea.

A mi abuela Isabel, a la que un mal día el alzheimer se colo en su vida para no marcharse.

Por supuesto, también a mis padres que siempre han sabido darme su apoyo en los momentos difíciles.

A mis compañeros de clase, sin los cuales llegar hasta aquí habría sido una tarea más difícil de lo que ya es.

A mis directores de proyecto por su incondicional disponibilidad y estímulo. He aprendido mucho de ellos y con ellos.

Resumen

En 1972 la *NASA*[1] (*National Aeronautics and Space Administration*) ponía en órbita el *Landsat*[2] 1, originalmente llamado "*Earth Resources Technology Satellite 1*", comenzando así la era de la Teledetección[1]. (En 2013, se lanzaba el *Landsat 8*).

Existen diversos campos de aplicación[2] de las imágenes hiperespectrales correspondientes a teledetección como puede ser el análisis de la cobertura terrestre, con el objetivo de clasificar las distintas zonas de la imagen, o el estudio de los océanos para poder encontrar las relaciones entre la profundidad del agua, el tipo de fondo y otras propiedades ópticas del agua.

La clasificación consiste en la asignación de una determinada etiqueta de clase (tierra, agua, tejado, ...) a cada píxel, y constituye uno de los campos más activos de la teledetección. Las técnicas de teledetección se enfrentan a varios problemas debidos a la escasez de muestras etiquetadas y a la alta dimensionalidad de las imágenes, lo que dificulta el proceso de clasificación.

Debido al gran avance que se está llevando a cabo en el campo de los sensores, cada vez disponemos de más datos por imagen, lo que obliga a mejorar las técnicas de clasificación de las imágenes hiperespectrales, imágenes en las que se dispone de cientos de valores de reflectancia por cada píxel.

Por otra parte, y a pesar del avance de los sensores, la resolución de los mismos no es lo suficientemente alta como para que en un único píxel se encuentre presente solamente un material.

Con este trabajo se pretende desarrollar una aplicación en el cual se puedan analizar las características de las imágenes hiperespectrales, aplicar distintas técnicas de clasificación de imágenes multidimensionales y comparar los resultados de las distintas configuraciones.

[1] http://www.nasa.gov/
[2] http://landsat.gsfc.nasa.gov/

Utilizando la metodología *Scrum* desarrollaremos una aplicación fácil, intuitiva y útil que permita al usuario trabajar con imágenes hiperespectrales y configurar distintos escenarios con el fin de obtener una buena clasificación.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Contextualización

Gracias a todos los avances tecnológicos que se están produciendo en el campo de los sensores, la teledetección ha recorrido un gran camino y ha permitido que se aplique a campos tan dispares como:

- La arqueología, localizando ruinas mayas a partir de la observación de antiguas pistas.
- La oceanografía para, por ejemplo, llevar a cabo estudios que permitan encontrar las relaciones entre la profundidad del agua, el tipo de fondo y otras propiedades ópticas del agua
- La geología donde la mayor parte de los esfuerzos se están centrando en intentar determinar zonas de algún material concreto.

El análisis y la clasificación de imágenes hiperespectrales de teledetección se ha convertido en una tarea importante para muchos investigadores, por eso se hace necesario el desarrollo de herramientas que faciliten los procesos que se deben llevar a cabo para poder extraer la máxima información.

1.2. Motivación y Objetivos

Motivación

El trabajo desarrollado en este Trabajo Fin de Grado (TFG) se ha centrado en el desarrollo de una herramienta de software libre capaz de mostrar las diferentes características detalladas de una imagen hiperespectral, así como aplicar distintas técnicas de clasificación desarrolladas en el grupo de investigación, que facilite la comparación de las mismas, en términos de acierto de la clasificación, de manera sencilla y lo más transparente posible al usuario. El trabajo se ha

aplicado a imágenes hiperespectrales de cobertura terrestre.

Así, el presente Trabajo Fin de Grado pretende el desarrollo de una aplicación que englobe una serie de algoritmos para el tratamiento de imágenes hiperespectrales junto con un conjunto de funcionalidades útiles para la representación y la medición de los resultados ofrecidos por dichos algoritmos. Como ya se ha comentado, la gran ventaja de esta herramienta es que es software libre, permitiendo su libertad de uso, modificación, ampliación y redistribución.

Por otro lado, esta característica elimina las restricciones de uso que imponen las licencias de software propietario como puede ser *Ecognition*[1] y los problemas que pueden surgir con actualizaciones no controladas.

Es pretensión también de este trabajo que los principales fundamentos a la hora de desarrollar la aplicación sean: que tenga una estructura modular que permita la inclusión sencilla de nuevos esquemas de clasificación, así como de otras etapas de procesado, y el manejo sencillo, óptimo y útil de la misma, así como de las funcionalidades que se crean necesarias para complementar este manejo utilizando para ello la biblioteca GTK+ [2].

También se pretende que la aplicación permita de una manera muy intuitiva que el usuario pueda diseñar sus propios algoritmos e incluirlos a los ya existentes. Por defecto se incluye una serie de algoritmos básicos para poder realizar clasificación, en concreto, los algoritmos incluidos en el presente trabajo son: ELM (*Extreme Learning Machine*)[3] y SVM (*Support Vector Machines*)[4]. También se incluyen *Watershed*[6], MV (*Majority Voting*)[7], RQS (*Really Quick Shift*)[8] que son algoritmos que permiten mejorar los resultados de la clasificación obtenida con SVM y ELM.

Objetivos

Este proyecto encierra dos grandes objetivos. El primer objetivo es el de proporcionar una herramienta que permita comparar, de forma gráfica y amigable, diferentes métodos de clasificación de imágenes hiperespectrales los cuales pueden contener algoritmos de compresión, eliminación de ruido, escalado, etc . El segundo gran objetivo es el de suministrar de manera intuitiva para el usuario información en detalle de diferentes características de cada imagen hiperespectral.

De forma más específica los objetivos que se han ido planteando son los siguientes:

- Permitir cargar imágenes hiperespectrales de cobertura terrestre desde un

[1] http://www.ecognition.com/
[2] http://www.gtk.org/

fichero para su posterior procesado, admitiendo para ello el uso de distintos formatos de fichero.

- Clasificar las imágenes mediante técnicas de aprendizaje supervisado basadas en SVM[4] y ELM[3], que han sido desarrolladas por el grupo de investigación.

- El diseño de la aplicación debe permitir incluir nuevas técnicas de clasificación y segmentación de imágenes hiperespectrales de una forma fácil, ofreciendo a los desarrolladores una base completa para minimizar la complejidad de crear nuevos plugins[3].

- Crear una aplicación de complejidad mínima que muestre de manera intuitiva para el usuario las imágenes originales, así como los resultados gráficos y numéricos de clasificación, junto con herramientas útiles para el estudio y comparación de los resultados ofrecidos.

- La interfaz de la aplicación debe ser intuitiva y muy fácil de usar.

- La imagen resultante de la clasificación debe poder compararse visualmente con la imagen original, mostrando ambas en pantalla.

- Permitir almacenar los resultados tanto en imagen clasificada como los resultados numéricos de precisión de la clasificación.

- Visualizar la imagen hiperespectral banda a banda, moviéndose entre las bandas de forma secuencial o aleatoria y permitiendo al usuario almacenar cada banda.

- Realizar un zoom sobre una zona de la imagen de manera que se puedan apreciar claramente los distintos píxeles que componen esa zona aumentada.

- Visualizar la firma espectral de cualquiera de los píxeles que forman la imagen mediante un histograma.

1.3. Estructura de la Memoria

Esta memoria, desarrollada bajo licencia *Creative Commons*[4], consta de 7 capítulos, 3 apéndices y la bibliografía utilizada.

[3]El nombre que reciben cada una de las técnicas que se añaden a la aplicación, dentro de la misma

[4]http://es.creativecommons.org/blog/

Capítulo 1: *Introducción.* En este capítulo se hace una pequeña introducción a los conceptos generales que se van a desarrollar a lo largo de la memoria. Introducimos conceptos como teledetección, imágenes hiperespectrales, distintos tipos de clasificación y segmentación de imágenes, las motivaciones y objetivos del proyecto y su contextualización.

Capítulo 2: *Imágenes Hiperespectrales.* En este capítulo se desarrolla una explicación más profunda de teledetección, imágenes hiperespectrales y clasificación y segmentación de imágenes. También se analizará el software actual para el tratamiento de imágenes hiperespectrales.

Capítulo 3: *Gestión del proyecto.* En este capítulo se desarrolla la gestión del proyecto describiendo el alcance, como se gestionan los riesgos y como se descompone el trabajo en tareas más simples. Se describirá la planificación temporal resultante y los costes derivados del desarrollo del proyecto. Por último se explicará la gestión de la configuración.

Capítulo 4: *Catálogo de Requisitos.* En este capítulo se realizará la elaboración de requisitos de la aplicación, identificando los participantes y describiendo los requisitos y los casos de uso. Se relacionarán los requisitos con los casos de uso a través de una matriz de trazabilidad. Además se establecerán los criterios de aceptación del producto, restricciones y exclusiones del mismo.

Capítulo 5: *Diseño e Implementación.* En este capítulo se recogen todos los aspectos relacionados con el diseño, tanto relativos a la aplicación como a la interfaz, así como los aspectos de implementación del sistema.

Capítulo 6: *Validación y Pruebas.* En este capítulo se describirán las pruebas realizadas sobre la aplicación y como se evaluó la usabilidad y utilidad de la misma. Además se recogerán algunas propuestas de mejora que resultarían interesantes de implementar en otros proyectos.

Capítulo 7: *Conclusiones y Posibles Ampliaciones.* En este capítulo se incluyen las conclusiones, mejoras futuras y lecciones aprendidas.

Apéndice A: *Manual de Usuario.* En este apéndice se describen tanto los pasos para la instalación de la aplicación como para el manejo de la misma.

Apéndice B: *Manual técnico.* En este apéndice se describen algunos detalles para la construcción e inclusión de nuevas técnicas de clasificación a la aplicación.

Apéndice C: *Metodologías ágiles*. Este apéndice es una introducción a las metodologías ágiles y al enfoque *Scrum*. Pretende ser una ayuda para aquellos lectores que no estén familiarizados con dicha metodología.

Bibliografía: Referencias usadas para el desarrollo de este proyecto, así como bibliografía complementaria.

Capítulo 2

Imágenes Hiperespectrales

En el presente capítulo se introducen los conceptos básicos que se van a manejar a lo largo del documento. En primer lugar se introducirá el concepto de teledetección, para posteriormente pasar a describir el concepto de imagen hiperespectral, detallando las características propias de este tipo de imágenes de alta dimensionalidad. También se introducirá el concepto de clasificación de imágenes hiperespectrales y se comentarán algunas técnicas. Para finalizar se analizarán algunos de los paquetes software actualmente disponible para el procesamiento de las imágenes hiperespectrales.

2.1. Teledetección

La teledetección[14] es la técnica de adquisición de datos de la superficie terrestre desde sensores instalados en plataformas espaciales o en aviones. La interacción electromagnética entre el terreno y el sensor, genera una serie de datos que son procesados posteriormente para obtener información de relevancia para el estudio que quiera realizarse: clasificación de terrenos agrícolas según su uso, determinación del avance de la deforestación en distintas zonas, ...

2.2. Imágenes Hiperespectrales

Hoy en día, existe una gran variedad de sensores capaces de medir singularidades espectrales en diferentes longitudes de onda a lo largo de áreas extensas [15]. La disponibilidad de esta instrumentación ha contribuido a una redefinición del concepto de imagen digital a través de la extensión de la idea de píxel. Así en una imagen en escala de grises cada píxel está constituido por un único valor discreto, mientras que, en una imagen hiperespectral, cada píxel contiene un espectro continuo que permite su identificación en base a las propiedades de reflectancia del material que lo compone. Estos valores pueden ser entendidos como vectores N-dimensionales [7], siendo N el número de bandas espectrales en

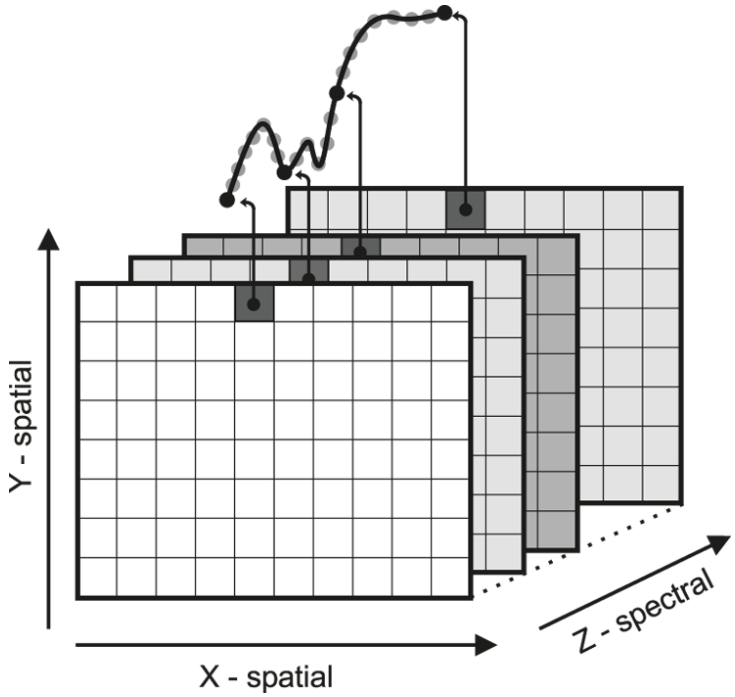

Figura 2.1: Cubo hiperespectral

las que el sensor es capaz de medir la información.

La ampliación del concepto de píxel da lugar a una representación en forma de cubo de los datos, tal y como aparece en la figura 2.1 (fuente: http://biomedicaloptics.spiedigitallibrary.org/article.aspx?articleid=1166695). El número de bandas de la imagen nos permite realizar una distinción a la hora de hablar de imágenes multidimensionales. Así, cuando el valor de N es reducido, típicamente unas cuantas bandas espectrales se habla de imágenes multi-espectrales, mientras que, cuando el orden de magnitud de N es de cientos de bandas se habla de imágenes hiperespectrales.

En este sentido, el análisis hiperespectral se basa en la capacidad de los sensores hiperespectrales para adquirir imágenes digitales en una gran cantidad de canales espectrales muy cercanos entre sí, obteniendo, para cada píxel, una firma espectral característica [15], que se representa como el vector de valores correspondientes al amplio espectro de la luz reflejada. Este proceso facilita la identificación y cuantificación de los materiales en la escena [16].

Tal y como hemos comentado anteriormente, el resultado de la adquisición de datos por parte de un sensor hiperespectral sobre un determinado escenario puede ser representado en forma de cubo de datos, con dos dimensiones para representar la ubicación espacial de un píxel, y una tercera dimensión para representar la

singularidad espectral de cada píxel en diferentes longitudes de onda. La figura 2.2 (fuente: http://www2.brgm.fr/mineo/final.htm) ilustra las características de una imagen hiperespectral mediante un sencillo diagrama donde podemos observar que la adquisición realizada por el sensor se representa mediante un cubo, donde cada lámina representa una banda espectral. Es interesante poder guardar estas bandas para posteriormente realizar un análisis más detallado de las mismas, como por ejemplo en el caso de bandas ruidosas. La firma espectral (*Spectral signature*), también llamada histograma, mostrada en la figura 2.2 se corresponde con los valores de reflectancia a diferentes longitudes de onda de un píxel de la imagen hiperespectral.

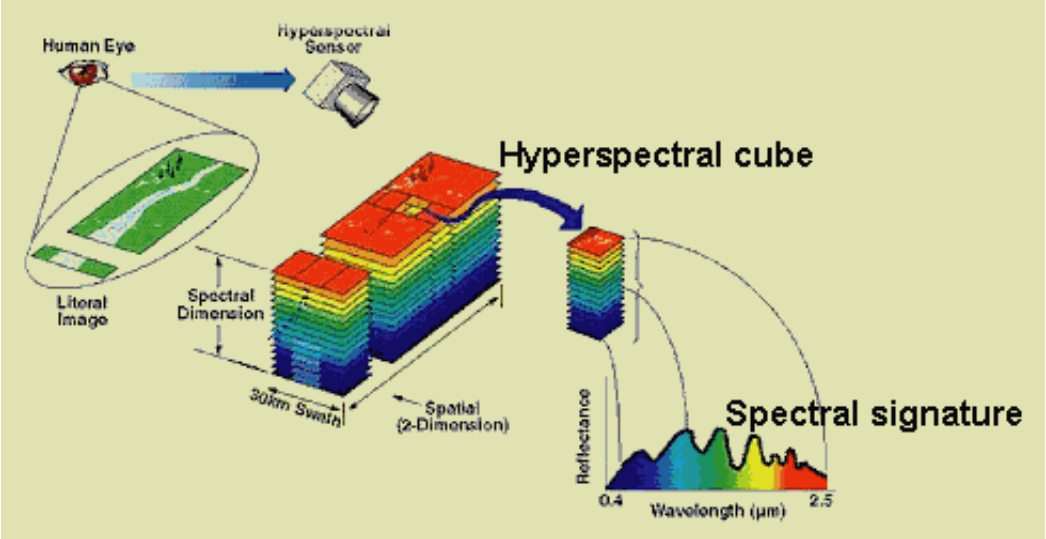

Figura 2.2: Proceso análisis hiperespectral

2.3. Clasificación de las Imágenes Hiperespectrales

Los recientes avances en la tecnología de sensores hiperespectrales remotos permite la adquisición simultánea de cientos de valores de reflectancia a diferentes longitudes de onda por cada píxel de la imagen[7]. Esta detallada información espectral aumenta las posibilidades de discriminar los distintos materiales o tipos de cultivo que componen el escenario con mayor precisión.

En el presente trabajo se han utilizado imágenes reales. Por un lado la imagen de la Universidad de Pavia[9] en Pavia, Italia, obtenida por el sensor óptico

ROSIS-03[1]. Esta imagen tiene una dimensión espacial de 610 x 340 píxeles y 115 bandas de las cuales se han eliminado las 12 bandas mas ruidosas, quedando en un total de 103 bandas disponibles. La otra imagen utilizada es la de Indian Pines[10], obtenida por AVIRIS[2]. En este caso la imagen cuenta con una dimension espacial de 145 x 145 píxeles y un conjunto de 220 bandas disponibles.

En la figura 2.3 (fuente: http://remotesensing.spiedigitallibrary.org/data/Journals/APPRES/929364/JARS_8_1_085094_f004.png) se puede ver el aspecto de la imagen hiperespectral de la Universidad de Pavia, así como el *ground truth* con la información de referencia de dicha imagen. Por otro lado, en la figura 2.3 podemos observar el mapa de clasificación de la imagen hiperespectral de la figura 2.4 utilizando la técnica SVM.

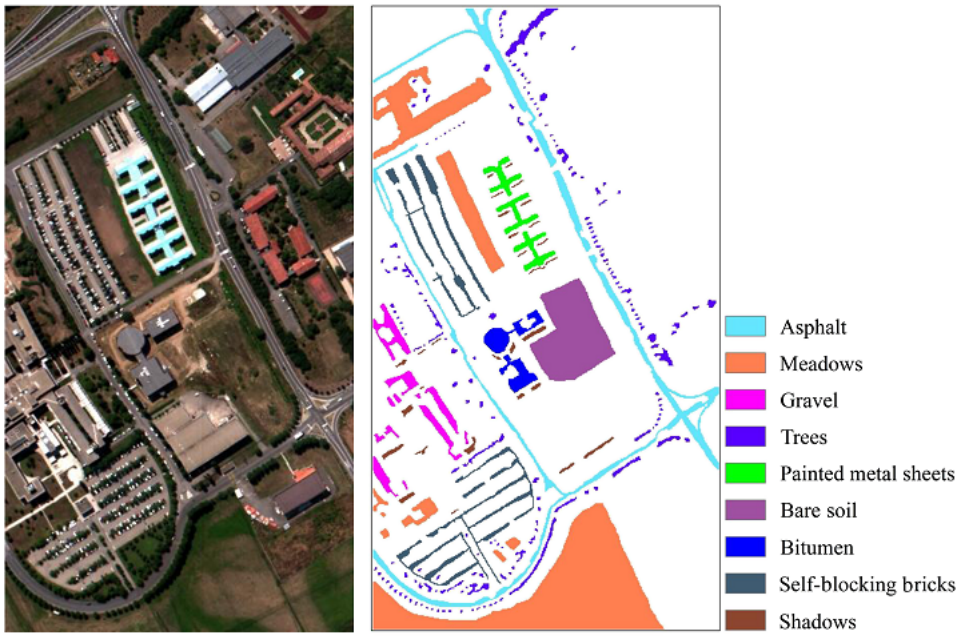

Figura 2.3: Imagen hiperespectral y *ground truth* de la Universidad de Pavia

La clasificación puede ser supervisada o no supervisada [17]. La clasificación supervisada se basa en la disponibilidad de áreas de entrenamiento, mientra que la clasificación no supervisada utiliza algoritmos matemáticos de clasificación automática. El presente trabajo se centra en las técnicas de clasificación supervisada desarrolladas por el grupo de investigación.

Cuando hablamos de clasificación de imágenes hiperespectrales también debemos diferenciar entre espacial y espectral. En el dominio espectral, la clasificación

[1]ROSIS (Reflective Optics System Imaging Spectrometer)
[2]AVIRIS (Airborne Visible - Infrared Imaging Spectrometer)

Figura 2.4: Mapa de clasificación obtenido utilizando *Support Vector Machines*

se basa en el hecho de que todos los materiales presentes en el mundo real reflejan, absorben y emiten energía electromagnética de forma distinta en diferentes longitudes de onda [18], considerando para cada píxel de la imagen toda la información espectral relativa, mientras que en el dominio espacial la clasificación tiene en cuenta la ubicación espacial (en el plano X, Y) de los distintos grupos de píxeles.

Un aspecto importante a tener en cuenta a la hora de clasificar imágenes hiperespectrales es como medir la calidad de las clasificaciones realizadas y así poder compararla con otras clasificaciones. Para este propósito se utilizan, entre otras, las siguientes medidas de precisión:

- *Overall accuracy (OA)* es el porcentaje de píxeles correctamente clasificados.

- *Average accuracy (AA)* es la media de las precisiones específicas de cada clase, por ejemplo, la media del porcentaje de píxeles clasificados correctamente por cada clase.

Clasificación en el Dominio Espectral

La comunidad dedicada a la teledetección ha realizado un gran esfuerzo en la última década para diseñar clasificadores precisos para las imágenes hiperespectrales. En particular, *Extreme Learning Machine* (ELM), que es un método de aprendizaje basado en redes neuronales llamadas *single-hidden layer feed-forward neural netwoks* (SLFNs). Comparado con las técnicas de inteligencia computacional más tradicionales, ELM ha probado ser una alternativa en términos de generalización de rendimiento, velocidad de aprendizaje y escalabilidad computacional[3].

También debemos destacar *Support Vector Machine* (SVM), que ha demostrado un rendimiento notable en cuanto a la precisión de la clasificación cuando el número de muestras de entrenamiento disponible es limitado[19]. SVM realiza una clasificación no lineal de los píxeles basada en toda la información espectral que está unida a la dimensión espectral de la imagen hiperespectral[20]. Sin embargo, SVM clasifica la imagen sin usar información contextual. Por lo tanto, las imágenes hiperespecrtales son tratadas como una lista de medidas espectrales sin organización espacial[21].

Obtención de Información en el Dominio Espacial

Un enfoque para la inclusión de la información espacial en el proceso de clasificación se inicia con la segmentación de imágenes. Los métodos de segmentación particionan una imagen en regiones homogéneas sin solapamiento con respecto a algún criterio de interés o criterio de homogeneidad[22]. Por lo tanto, cada región en el mapa de segmentación define un vecindario espacial para todos los píxeles dentro de esa región. Diferentes técnicas han sido investigadas para la segmentación de imágenes hiperespectrales, tales como *watershed*[6], RQS(*Really Quick Shift*)[8] y *K-means*[23].

Clasificación Espacial-Espectral

Para este propósito necesitamos algoritmos que combinen información desde diversas fuentes. En el caso de la clasificación de imágenes hiperespectrales hablamos de una fuente con información espacial y otra con información espectral, por eso, una vez que hemos realizado la segmentación de la imagen, el siguiente paso es incorporar la información espacial derivada del mapa de segmentación en una clasificación espectral-espacial mediante algoritmos como MV (*Majority Voting*)[11] o *weighted fusion* [24]. Esto permite incluir tanto la información espacial como la espectral para realizar la clasificación final de la imagen.

En la figura 2.5 (fuente: *"Multiple Spectral–Spatial Classification Approach for Hyperspectral Data"*) podemos ver un ejemplo ilustrativo de la combinación de información espectral y espacial usando el método *majority voting*. En dicha figura, el cuadro superior izquierdo representa el mapa de segmentación de la imagen, donde se etiquetan las distintas regiones detectadas en la imagen. En el cuadro superior derecho podemos ver el resultado de la clasificación de la imagen, donde a cada píxel se le asigna una clase (cemento, agua, campo, ...). En el cuadro intermedio podemos ver el resultado de superponer el mapa de segmentación al mapa de clasificación, lo que nos permite ver el número de clases distintas en cada región. Por último, el cuadro inferior nos muestra el resultado final del *majority voting*, donde a cada región se le asigna la clase mayoritaria de dicha región.

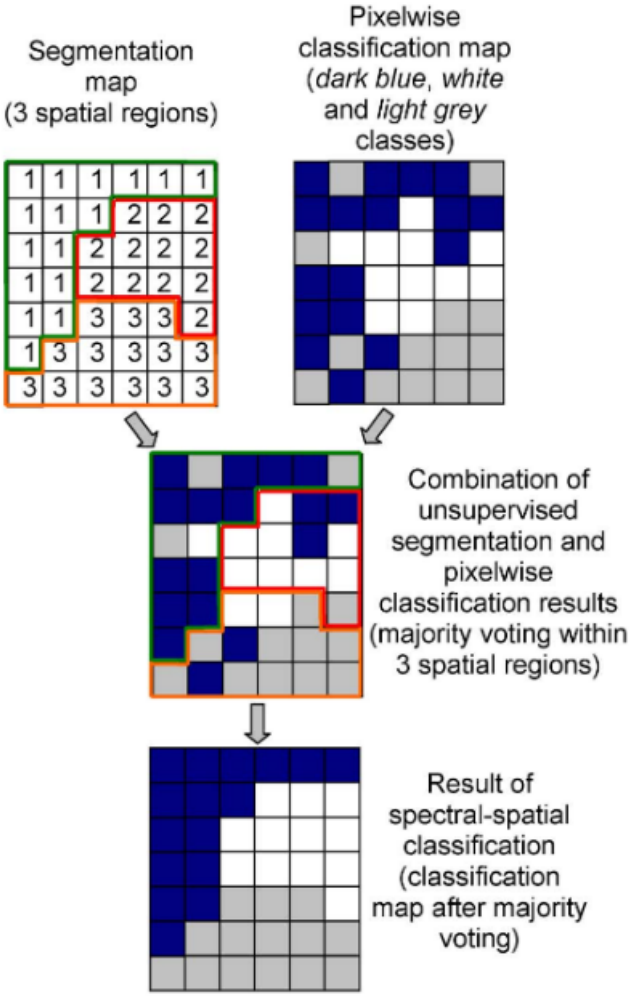

Figura 2.5: Esquema ejemplo de clasificación espacial-espectral usando *majority voting*.

2.3.1. Ejemplo de Clasificación

Partiendo de una imagen hiperespectral y de su ground truth vamos a explicar un ejemplo de clasificación que se podría realizar con la aplicación desarrollada en este proyecto. En la figura 2.6 podemos observar el proceso completo de clasificación usado para este ejemplo.

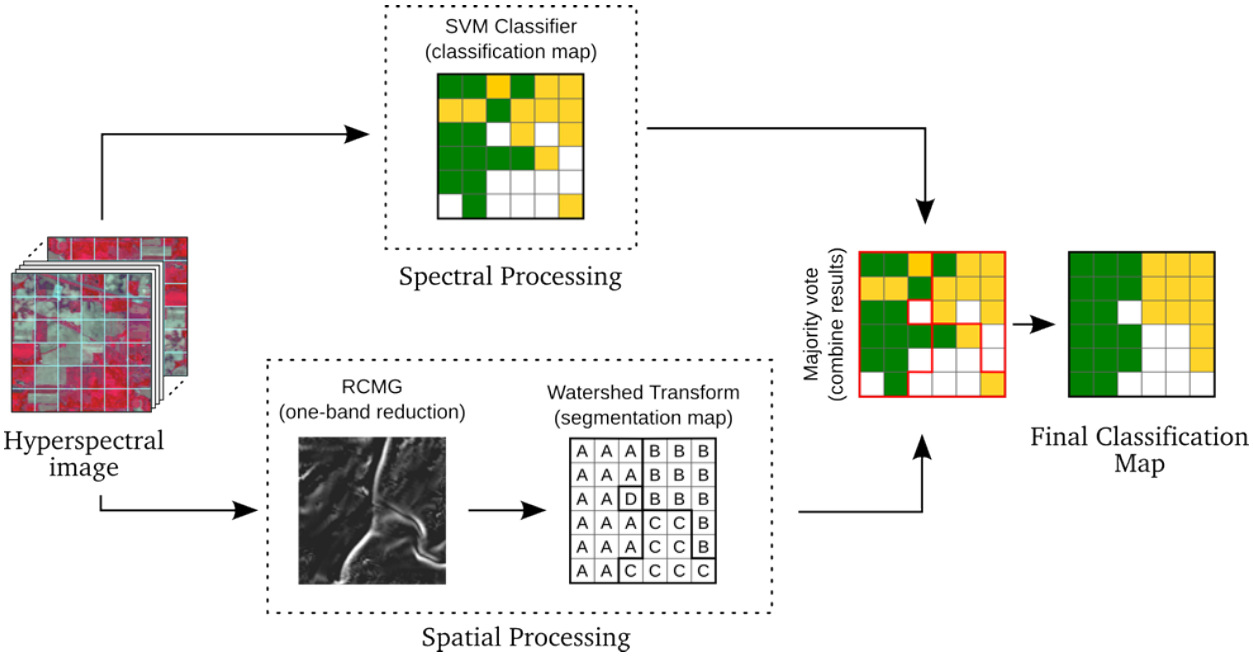

Figura 2.6: Ejemplo de clasificación.

El software desarrollado permite incluir plugins en distintas fases. Estas fases funcionan como una hoja de ruta, indicándole al programa en que orden debe ejecutar los distintos plugins. Para poder entender mejor el proceso de clasificación realizado por la aplicación, vamos a explicar las distintas fases en las que se permiten incluir plugins:

Preprocesado: En esta fase se podrán incluir todos aquellos plugins que modifiquen de alguna manera la imagen hiperespectral, como por ejemplo, un plugin de eliminación de ruido.

Camino 1: Esta fase realiza la clasificación espectral de la imagen y podrá incluir todos aquellos plugins diseñados para esa tarea, permitiendo también incluir cualquier otro plugin que modifique la imagen hiperespectral, ya sea eliminado ruido, reduciendo bandas, etc.

Camino 2: En esta fase se realiza una clasificación espacial y deberá incluir plugins adecuados para esa tarea, además de permitir, al igual que en la fase "camino 1", incluir cualquier otro plugin que modifique la imagen hiperespectral.

Unión: Como su propio nombre indica, en esta fase se incluirá un único plugin encargado de combinar la información espectral proveniente de la fase "camino 1" con la información espacial obtenida de la fase "camino 2". Solo se hará uso de esta fase si incluimos plugins en la fase de "camino 1" y en la fase de "camino 2" al mismo tiempo.

Postprocesado: Esta fase puede incluir todos aquellos plugins que modifiquen el resultado obtenido de la fases anteriores.

Como en este caso no se va a realizar ni preprocesado ni postprocesado de la imagen, no se aplicará ningún algoritmo en estas fases. Sin embargo, podemos observar que si se realizan tanto una clasificación espectral (*Spectral Processing*), utilizando SVM, como una espacial (*Spatial Processiong*), mediante el algoritmo *watershed*, al que se le ha aplicado el algoritmo RCMG de reducción de bandas.

Para la clasificación espectral incluiremos el plugin SVM en la fase "camino 1" mientras que para la clasificación espacial incluiremos el plugin *watershed* en la fase "camino 2". Debemos aclarar que el plugin *watershed* desarrollado para este proyecto ya incluye el algoritmo RCMG, en caso contrario también deberíamos incluirlo en la fase "camino 2" y posicionarlo antes que el algoritmo *watershed*.

Para finalizar, solo tenemos que insertar el plugin MV en la fase "unión" para combinar la información espectral y la espacial y así obtener la clasificación final de la imagen.

En la figura 2.7 podemos ver de forma esquemática la posición de los distintos plugins dentro de cada fase y el resultado de cada una de estas fases tras la ejecución de los plugins que contiene. En la parte superior de la imagen podemos ver tanto la imagen hiperespectral como el *ground truth* que serán utilizados por los plugins para alcanzar la clasificación final. En la fase "camino 1" se ejecuta el plugin *SVM* y a su izquierda podemos ver el resultado obtenido. A continuación se ejecuta el plugin de la fase "camino 2", es decir, *watershed*, y a su derecha tenemos el resultado de esta fase. Por último ejecutamos el plugin *MV* que se encuentra en la fase "unión" y dado que no hay postprocesado, el resultado de esta fase será el mismo que obtenemos al finalizar la ejecución de las fases restantes.

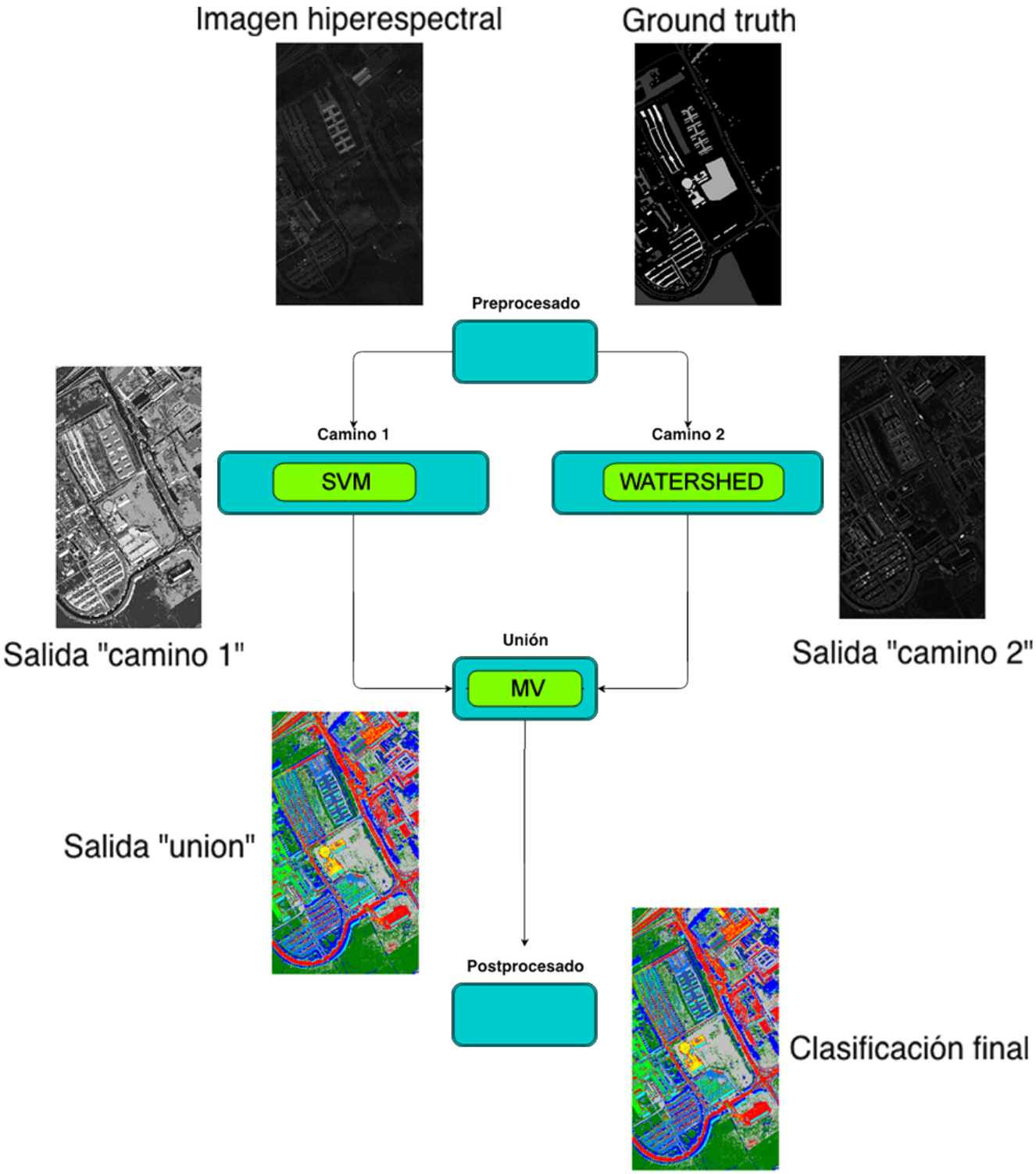

Figura 2.7: Esquema del proceso de clasificación y los distintos resultados generados.

2.4. Software Existente

En esta sección pondremos algunos ejemplos tanto de software comercial como de software libre que están disponibles actualmente para procesar imágenes hiperespectrales y en particular clasificarlas.

La ventaja de nuestro software sobre los que aquí se van a mencionar es que permite al grupo de investigación incorporar de forma fácil los distintos algoritmos de clasificación desarrollados dentro de dicho grupo.

2.4.1. Software Comercial

En la actualidad existen varios software comerciales dedicados al tratamiento de imágenes hiperespectrales. En este apartado vamos a analizar dos de ellos, eCognition[3] y ENVI[4]:

- *eCognition* es un software que combina todo tipo de datos geoespaciales para realizar diversos análisis y comparaciones. Puede importar imágenes (aéreas o de satélite), nubes de puntos, información vectorial, datos provenientes de un SIG, etc.

  Tiene un gran número de aplicaciones como, por ejemplo, detección de cambios, usos del suelo, detección de masas arbóreas, tipos de vegetación, etc.

  Dispone de infinidad de algoritmos predefinidos, que el usuario puede combinar y ajustar a sus necesidades y de acuerdo al tipo de datos que esté utilizando. Los usuarios avanzados pueden además desarrollar sus propios algoritmos o añadir nuevas capacidades ya que el software incorpora un SDK (*Software Development Kit*).

  Utiliza análisis basado en objetos, esto es, agrupa píxeles de imágenes con características similares de acuerdo a una parametrización dada.

  Existen tres versiones diferentes de *eCognition*:

  - *eCognition Developer*: Es el módulo que permite combinar los algoritmos existentes y definir las estrategias que se aplicarán sobre los datos geográficos utilizados.

[3] http://www.ecognition.com/
[4] http://www.exelisvis.com/ProductsServices/ENVIProducts/ENVI.aspx

- *eCognition Architect*: El módulo *Architect* permite estructurar de una forma mas amigable las estrategias que se aplicarán sobre los datos geográficos utilizados facilitando al usuario la parametrización y la ejecución de los procesos.
- *eCognition Server*: Es el módulo que permite procesar grandes conjuntos de datos. Puede dividir el proyecto en partes más pequeñas que se ejecutan individualmente en distintos servidores o computadores y que se unen posteriormente.

El precio de este software va desde los 2.000 € hasta los 4.000 € , en función de la versión . En la figura 2.8 podemos ver una captura del la aplicación en funcionamiento.

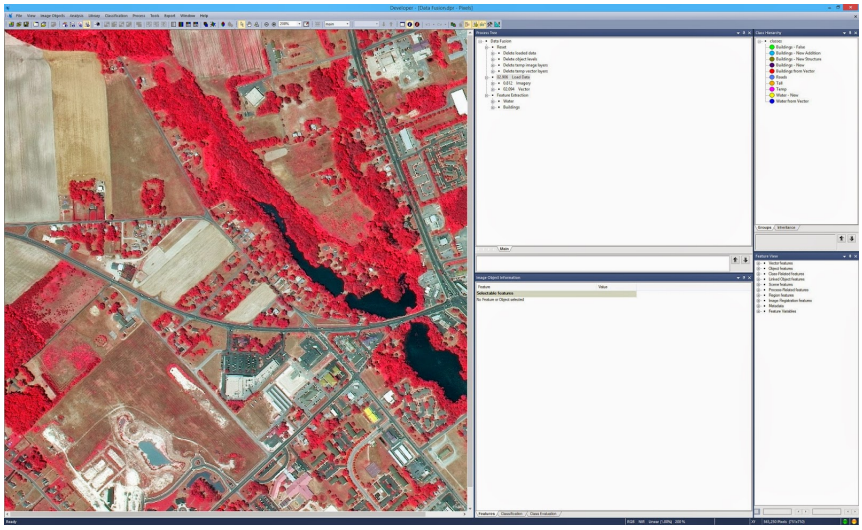

Figura 2.8: Captura de la aplicación *eCognition*

- *ENVI* es un software para el procesamiento y análisis de imágenes geoespaciales que puede ser utilizado por cualquiera, desde profesionales GIS hasta analistas de imágenes y científicos. Desarrollado por *Exelis Visual Information Solutions*, ENVI combina procesamientos de las imágenes espectrales más recientes con tecnología de análisis de imagen mediante una interfaz intuitiva y fácil de usar que ayuda a obtener información significativa de las imágenes tratadas. Posee herramientas automatizadas para trabajar con cualquier tamaño de imagen.

  Fue desarrollado usando el lenguaje (IDL), el cual está especializado en el manejo de datos multidimensionales y su visualización. Se diferencia de otros programas similares en que contiene funciones especialmente adaptadas al trabajo con información territorial o geográfica. ENVI se caracteriza

por ser multiplataforma, existiendo versiones para Windows, Linux y varias versiones de UNIX. En la figura 2.9 podemos ver una captura del la aplicación en funcionamiento.

Ambas herramientas proporcionan un amplio abanico de funcionalidades. Por supuesto, el hecho de que sean software propietario implica que su uso y funcionalidades dependen del soporte dado por las respectivas compañías que los desarrollan.

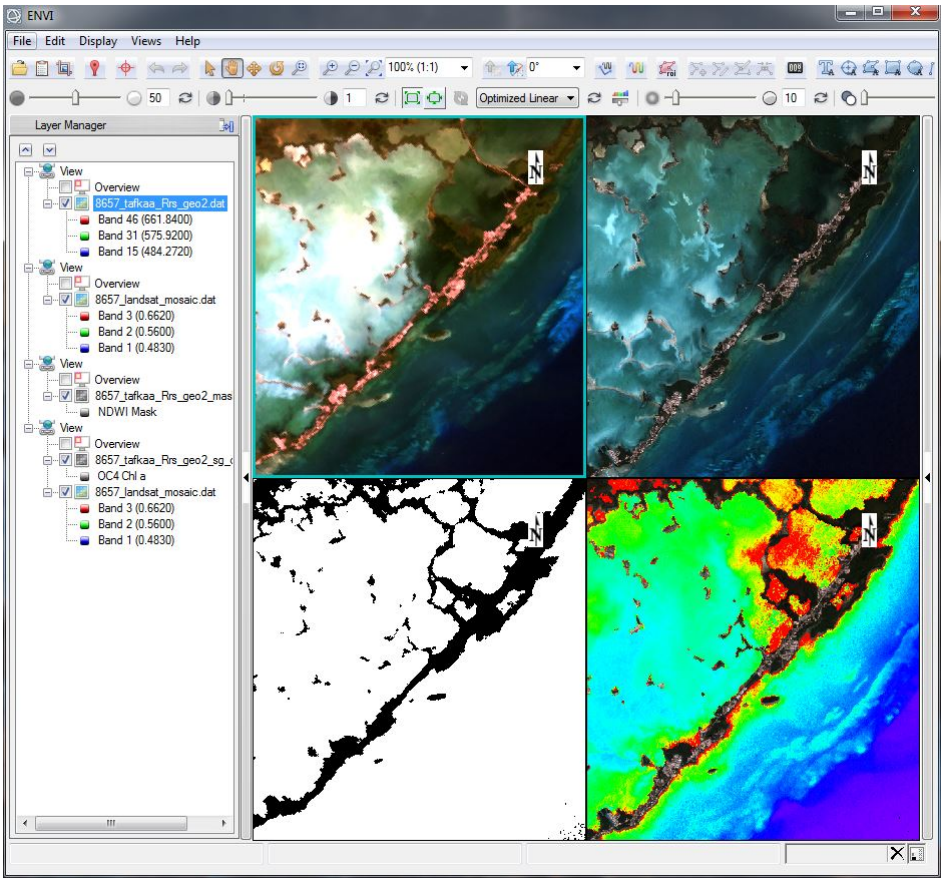

Figura 2.9: Captura de la aplicación *ENVI*

2.4.2. Software Libre

En esta sección vamos a enumerar varios paquetes software para el análisis de imágenes hiperespectrales clasificados como software libre.

- *Web Map Processing Server*(WMPS)[12] es una aplicación web para el procesamiento de imágenes obtenidas desde repositorios de imágenes satélites

disponibles de forma gratuita en Internet. Esta aplicación utiliza el repositorio facilitado por *Google Maps* TM[5]. La Aplicación proporciona acceso a cualquier imagen disponible desde *Google Maps* y permite, posteriormente, que estas puedan ser procesadas y clasificadas según ciertos algoritmos de clasificación. Actualmente están disponibles los algoritmos o procesos*Kmeans* e *Isodata*.

- *HyperMix*[13] es una herramienta disponible online para su descarga y evaluación. Se compone de diferentes algoritmos que cubren la cadena de desmezclado hiperespectral estándar. Esta cadena se compone de tres partes principales: 1) la reducción de dimensiones, 2) selección de firmas o *endmembers* espectrales puros, y 3) la estimación de la abundancia de cada *endmember* en cada píxel de la escena. En la figura 2.10 podemos ver la apariencia de la aplicación.

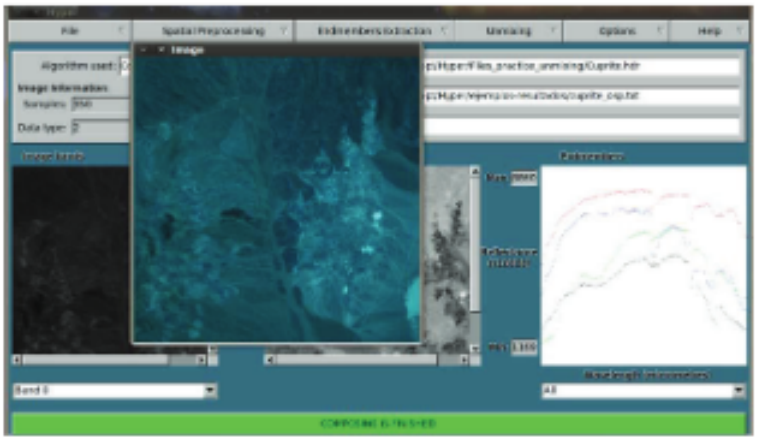

Figura 2.10: Captura de la aplicación *HyperMix*

- *Hyperspectral data analysis in R*[6] es un paquete que contiene clases y funciones para manejar, analizar y simular datos hiperespectrales. Este software está diseñado para manejar imágenes de alta dimensionalidad y contiene además funciones para el trazado de datos espectrales.

- *HyperSpec*[7] es un software que facilita el análisis de los espectros utilizando el software estadístico *R* [8]. *Hyperspec* actúa como una interfaz para el manejo de los espectros, usando el poderoso análisis de datos proporcionada por *R* y otros paquetes.

[5]https://www.google.es/maps/preview?source=newuser-ws
[6]http://sourceforge.net/projects/hsdar/
[7]http://hyperspec.r-forge.r-project.org/
[8]http://www.r-project.org

- *GRASS GIS*[9], comunmente referenciado como GRASS (*Geographic Resources Analysis Support System*), es un software de código abierto usado para la gestión y análisis de datos geoespaciales, procesamiento de imágenes, generación de gráficas y mapas, modelado espacial y visualización. GRASS cuenta con más de 350 módulos para el renderizado de imágenes y mapas; manipular tramas y vectores de datos, incluyendo redes de vectores; permite procesar datos de imágenes multiespectrales; y crear, gestionar y almacenar datos espaciales. GRASS ofrece tanto una interfaz gráfica como una línea de comandos para facilitar las operaciones. También puede interactuar con impresoras, plotters, digitalizadores, y bases de datos para desarrollar nuevos datos, así como gestionar los datos existentes. En la figura 2.11 podemos ver una captura de la aplicación.

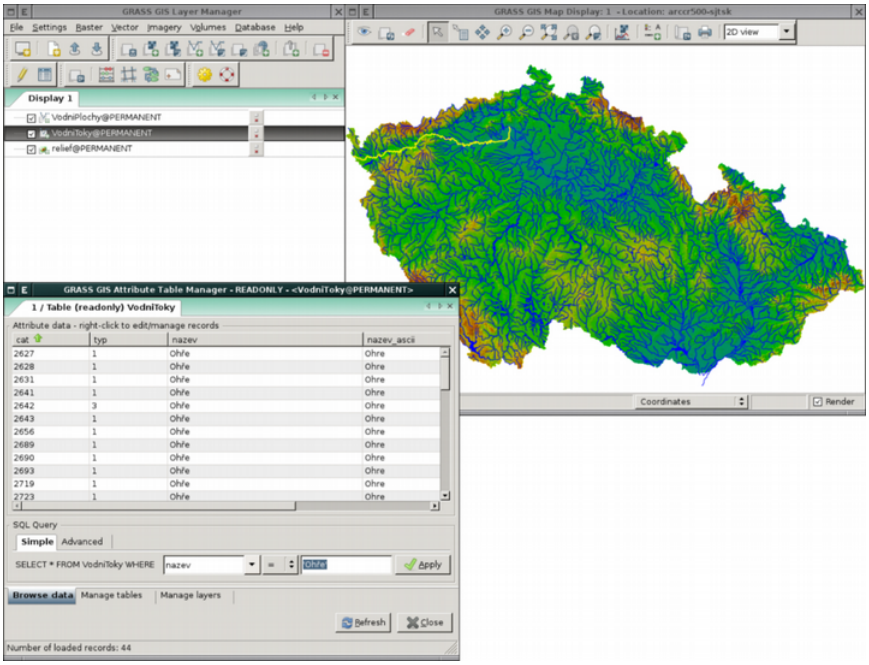

Figura 2.11: Captura de la aplicación GRASS GIS

[9]http://grass.osgeo.org/

Capítulo 3

Gestión del Proyecto

La Gestión de Proyectos tiene como finalidad principal la planificación, el seguimiento y control de las actividades y de los recursos humanos y materiales que intervienen en el desarrollo de un Sistema de Información. Como consecuencia de este control es posible conocer en todo momento qué problemas se producen y resolverlos o paliarlos de manera inmediata.

En la sección III del PMBOK, se encuentran definidos todos los aspectos que nos permitirán desarrollar nuestro proyecto.

3.1. Gestión del Alcance del Proyecto

La gestión del alcance del proyecto incluye los procesos necesarios para garantizar que el proyecto incluya todo (y únicamente todo) el trabajo requerido para dar por finalizado el proyecto de forma satisfactoria. El objetivo principal de la gestión del alcance del proyecto es definir y controlar qué se incluye y qué no se incluye en el proyecto.

En esta sección se seleccionará una metodología de desarrollo, se explicarán las técnicas de extracción de requisitos y se definirá la Estructura de Desglose del Trabajo (EDT).

3.1.1. Metodología de Desarrollo

El ciclo de vida del proyecto define las fases que conectan el inicio de un proyecto con su fin [25]. La elección del modelo debe adaptarse a la naturaleza del proyecto y de los productos, así como a los aspectos relacionados.

La metodología de desarrollo escogida para la realización de este trabajo es una metodología ágil, que trata de minimizar el esfuerzo asociado a la metodología, de

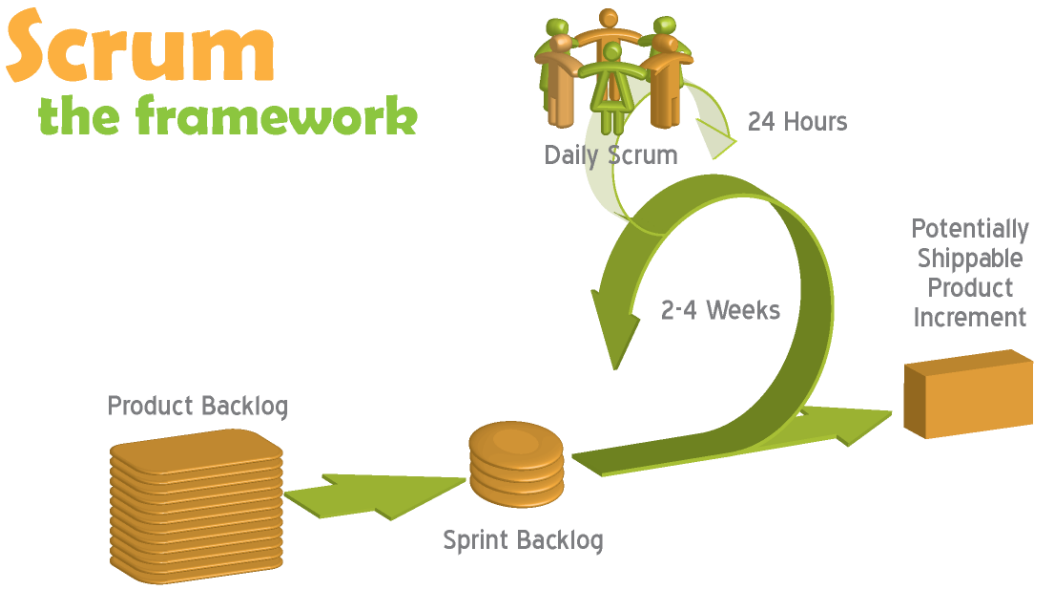

Figura 3.1: Ciclo de la metodología Scrum.

tal forma que se agilice el desarrollo y la capacidad de adaptación a los cambios. Concretamente se a seleccionado Scrum, cuyo ciclo de vida se refleja en la figura 3.1 (fuente: http://www.diegosalama.com/2009/08/21/scrum-funciona/) y que posee los siguientes beneficios [29]:

- **Entrega mensual (o quincenal) de resultados**, lo cual proporciona las siguientes ventajas:

  - **Gestión regular de las expectativas del cliente:**
    El cliente establece sus expectativas indicando el valor que le aporta cada requisito del proyecto y cuando espera que esté completado.

    El cliente comprueba de manera regular si se van cumpliendo sus expectativas, da *feedback*, ya desde el inicio del proyecto puede tomar decisiones informadas a partir de resultados objetivos y dirige estos resultados del proyecto, iteración a iteración, hacia su meta. Se ahorra esfuerzo y tiempo al evitar hipótesis.

  - **Resultados anticipados:**
    El cliente puede empezar a utilizar los resultados más importantes del proyecto antes de que esté finalizado por completo.

    Siguiendo la ley de Pareto [1] (el 20 % del esfuerzo proporciona el 80 % del valor), el cliente puede empezar antes a recuperar su inversión

[1] http://es.wikipedia.org/wiki/Principio_de_Pareto

(y/o autofinanciarse) comenzando a utilizar un producto al que sólo le faltan características poco relevantes, puede sacar al mercado un producto antes que su competidor, puede hacer frente a urgencias o nuevas peticiones de clientes, etc.

- **Flexibilidad y adaptación:**
  De manera regular el cliente redirige el proyecto en función de sus nuevas prioridades, de los cambios en el mercado, de los requisitos completados que le permiten entender mejor el producto, de la velocidad real de desarrollo, etc.

  Al final de cada iteración el cliente puede aprovechar la parte de producto completada hasta ese momento para hacer pruebas de concepto con usuarios o consumidores y tomar decisiones en función del resultado obtenido.

- **Mitigación de riesgos:**
  Desde la primera iteración el equipo tiene que gestionar los problemas que pueden aparecer en una entrega del proyecto. Al hacer patentes estos riesgos, es posible iniciar su mitigación de manera anticipada. "Si hay que equivocarse o fallar, mejor hacerlo lo antes posible". El *feedback* temprano permite ahorrar esfuerzo y tiempo en errores técnicos.

  La cantidad de riesgo a que se enfrenta el equipo está limitada a los requisitos que se puede desarrollar en una iteración. La complejidad y riesgos del proyecto se dividen de manera natural en iteraciones.

- **Productividad y calidad:**
  De manera regular el equipo va mejorando y simplificando su forma de trabajar.

  Los miembros del equipo sincronizan su trabajo diariamente y se ayudan a resolver los problemas que pueden impedir conseguir el objetivo de la iteración. La comunicación y la adaptación a las diferentes necesidades entre los miembros del equipo son máximas (se van ajustando iteración a iteración), de manera que no se realizan tareas innecesarias y se evitan ineficiencias.

- **Alineamiento entre cliente y equipo:**
  Los resultados y esfuerzos del proyecto se miden en forma de objetivos y requisitos entregados al negocio. Todos los participantes en el proyecto conocen cuál es el objetivo a conseguir. El producto se enriquece con las aportaciones de todos.

- **Equipo motivado:**
  Las personas están más motivadas cuando pueden usar su creatividad para

resolver problemas y cuando pueden decidir organizar su trabajo.

Aunque el enfoque *Scrum* es un método ágil general, su enfoque está en la administración iterativa del desarrollo, y no en enfoques técnicos específicos para la ingeniería de software ágil.

Scrum consta de tres fases bien diferenciadas[30]:

Primera Fase Scrum.

Esta fase consta de:

- Planificación del bosquejo.
- Establecer objetivos generales del proyecto.
- Diseño de la arquitectura de software.

Segunda Fase Scrum.

En esta fase se llevan a cabo una serie de ciclos *sprint*, donde cada ciclo desarrolla un incremento del sistema. Un *sprint* de *Scrum* es una unidad de planificación en la que se valora el trabajo que se va a realizar, se seleccionan las particularidades por desarrollar y se implementa el software. Al final de un *sprint*, la funcionalidad completa se entrega a los participantes. Las características clave de este proceso son las siguientes:

1. Los *sprint* tienen una longitud fija, por lo general de dos a cuatro semanas.
2. El punto de partida para la planificación es la cartera del producto (*Product Backlog*) que es la lista de requisitos por realizar en el proyecto. Durante la fase de valoración del *sprint*, esto se revisa, y se asignan prioridades y riesgos. El cliente interviene estrechamente en este proceso y al comienzo de cada *sprint* puede introducir nuevos requisitos o tareas.
3. En la fase de selección se implica a todo el equipo del proyecto que trabaja con el cliente, con la finalidad de seleccionar las características y la funcionalidad a desarrollar durante el *sprint*.
4. Una vez acordado, el equipo se organiza para desarrollar el software. Con el objetivo de revisar el progreso, y si es necesario, revisar las prioridades de los trabajos, se realizan reuniones diarias breves con todos los integrantes del equipo. Durante esta etapa, toda la comunicación entre el cliente o la organización y el equipo de desarrollo se canaliza a través del "maestro de scrum" que debe proteger al equipo de desarrollo de distracciones externas.

A diferencia de otras metodologías, scrum no hace sugerencias específicas sobre como escribir requisitos, desarrollar las pruebas, etc.

5. Al final del *sprint*, se revisa el trabajo y se presenta a los participantes.

Tercera Fase Scrum.

En la última fase de scrum se realiza el cierre del proyecto, completando la documentación requerida, los marcos de ayuda del sistema y los manuales del usuario, y se valoran las lecciones aprendidas en el proyecto.

Podemos encontrar más información sobre *Scrum* y metodologías ágiles en el apéndice C.

3.1.2. Gestión de Requisitos

Tal y como se especifica en la metodología *scrum*, antes de comenzar con los ciclos *sprint* debemos disponer de una lista de requisitos (llamados en *Scrum* historias de usuario) priorizada que se podrán modificar a medida que se consumen los distintos ciclos *sprint* que contenga el proyecto.

Para la adquisición inicial de requisitos se han utilizado los casos de uso, que son una técnica de descubrimiento de requisitos que se introdujo por primera vez en el método *Objectory*[2]. En su forma más sencilla, un caso de uso identifica a los actores implicados en una iteración, y nombra el tipo de iteración. Entonces, esto se complementa con información adicional que describe la iteración con el sistema. Los casos de uso se documentarán con el diagrama de casos de uso de alto nivel que se muestra en la figura 4.1.

3.1.3. Estructura de Descomposición del Trabajo

Según el PMBOK el EDT es *"Una descomposición jerárquica [...] del trabajo que será ejecutada por el equipo del proyecto, para lograr los objetivos del proyecto y crear los productos entregables requeridos"*. El EDT ayuda a descomponer en porciones de trabajo más pequeñas el proyecto que resulta en una serie de "paquetes de trabajo" que se deben realizar para finalizar con éxito el producto.

La figura 3.2 representa el EDT de este trabajo. Cabe señalar que debido a que utilizamos una metodología ágil como es *scrum*, no es posible definir un EDT completo al inicio del proyecto, pues es en cada iteración o ciclo *sprint* donde se decide que requisitos de la lista de productos (*product backlog*) son los que se van ha seleccionar para ser desarrollados en esa iteración. Es por eso que el diseño

[2]Jacobson et al., 1993

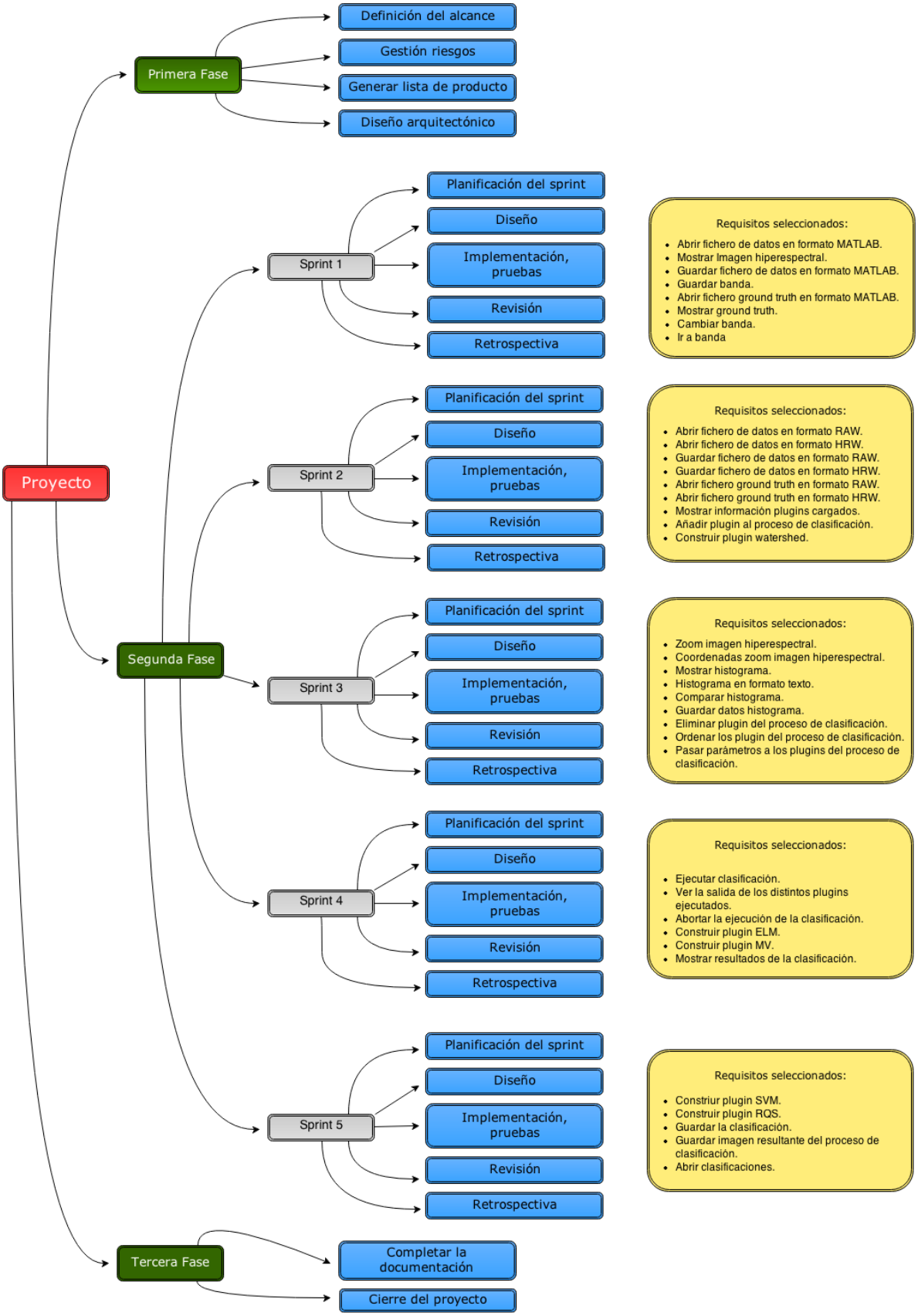

Figura 3.2: Estructura de Descomposición del Trabajo (EDT).

de este EDT se ha realizado en la tercera fase del proyecto, y aunque su elaboración no es obligatoria en este tipo de metodología, hemos considerado que sería interesante poder ver, de una forma gráfica, como se ha estructurado el proyecto.

La planificación incluida en el anteproyecto, y que se puede observar en el EDT de la figura 3.3, hacia referencia a una metodología en cascada. Tras analizar más en profundidad el alcance del proyecto y apreciar que el número de requisitos era bastante elevado, se considero más apropiado optar por una metodología ágil que permitiese dedicar un mayor tiempo al desarrollo de la aplicación a cambio una documentación menos elaborada, permitiendo cumplir con los requisitos del proyecto dentro de un plazo razonable para un trabajo fin de grado.

3.2. Gestión de Riesgos

La gestión de riesgos del proyecto incluye los procesos relacionados con la planificación de la gestión de riesgos, la identificación y el análisis de riesgos, las respuestas a los riesgos, y el seguimiento y control de riesgos de un proyecto. La mayoría de estos procesos se actualizan durante el proyecto. Los objetivos de la gestión de riesgos son aumentar la probabilidad y el impacto de los eventos positivos, y disminuir la probabilidad y el impacto de los eventos adversos para el proyecto [26].

La gestión de riesgos implica prever y gestionar los riesgos que pueden influir en la planificación temporal o en la calidad del producto desarrollado y tomar las acciones necesarias para, según su naturaleza, evitarlos o minimizar su impacto. La tabla 3.1 objetiviza la probabilidad de que ocurra un riesgo.

Nivel de ocurrencia del riesgo	Probabilidad
$\geq 80\,\%$ (Casi seguro que ocurre)	Alta
Entre $30\,\%$ y $79\,\%$ (Muy probable que ocurra)	Media
$< 30\,\%$ (Poco probable)	Baja

Tabla 3.1: Valoración de la probabilidad de la ocurrencia de un riesgo.

Para poder valorar objetivamente el coste de la ocurrencia de un riesgo, se tendrá en cuenta la tabla 3.2 que describe el impacto que tendría en el proyecto.

La especificación de los riesgos que se lleva a cabo a continuación se realiza siguiendo la metodología del PMBOK [27]. La tabla 3.3 objetiviza la probabilidad de que ocurra un riesgo. Para gestionar debidamente los riesgos, únicamente serán considerados aquellos con un nivel de exposición alto según se refleja en la tabla 3.3.

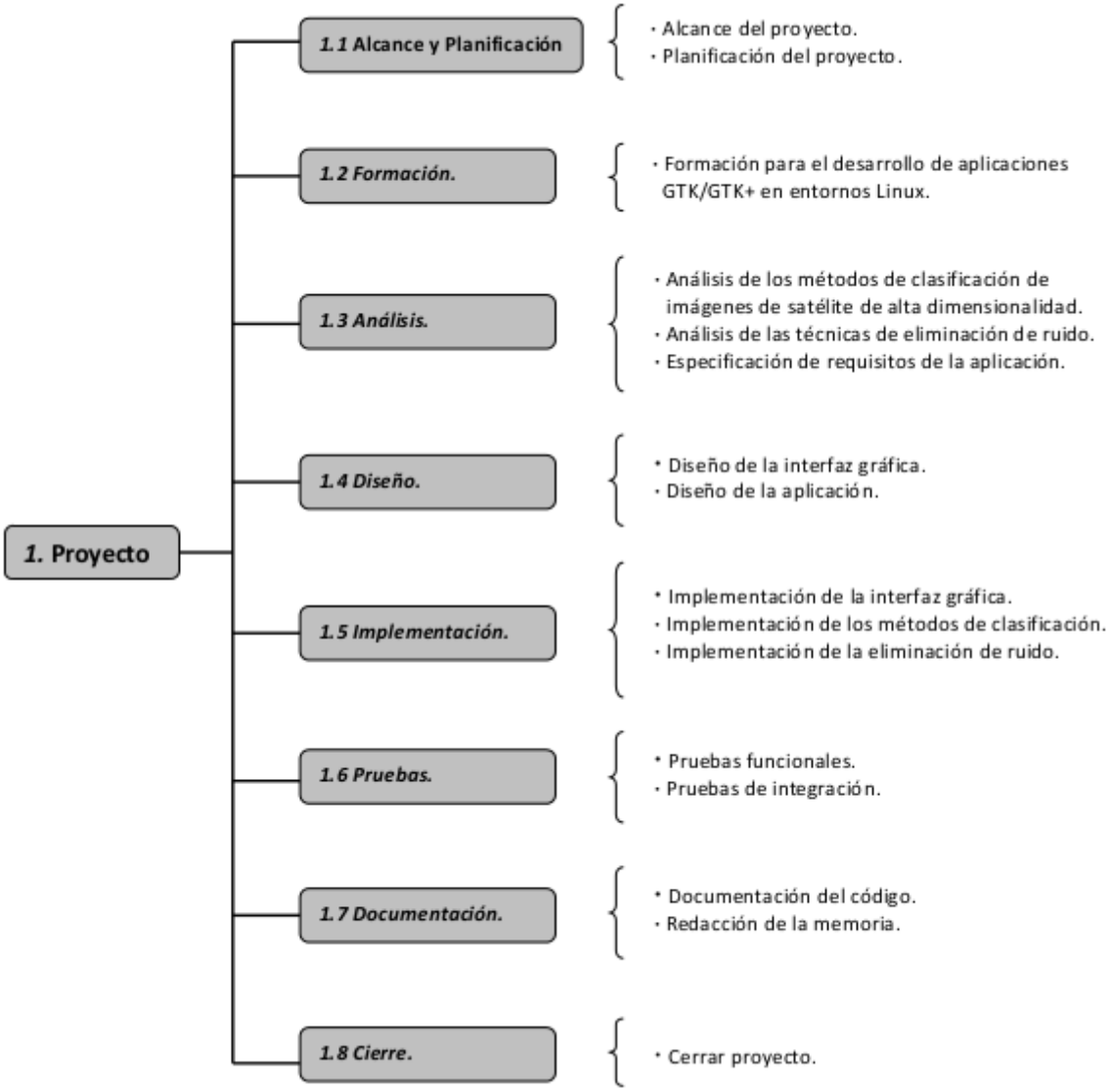

Figura 3.3: Estructura de Descomposición del Trabajo (EDT) del anteproyecto.

Repercusión en Plazo/Esfuerzo	Impacto
$\leq 25\,\%$	Alta
Entre $10\,\%$ y $24\,\%$	Media
$< 10\,\%$	Baja

Tabla 3.2: Valoración del impacto sobre el plazo de entrega o el esfuerzo requerido de un riesgo

Probabilidad / Impacto	Alta	Media	Baja
Alto	Alto	Alto	Medio
Medio	Alto	Medio	Bajo
Bajo	Medio	Bajo	Bajo

Tabla 3.3: Matriz probabilidad/impacto.

3.2.1. Especificación de Riesgos.

Desde la tabla 3.4 hasta la tabla 3.11 se describen aquellos riesgos considerados en el proyecto por alcanzar un nivel de exposición alto.

Identificador	Riesgo-01
Descripción	Desconocimiento de las tecnologías que se van a usar en el proyecto, así como de cualquier otro concepto nuevo que se maneje en el proyecto.
Probabilidad	Alta
Impacto	Alto
Exposición	Alta
Plan de prevención	Tenerlo en cuenta a la hora de planificar los *sprint* para que en cada uno de ellos se pueda reservar un espacio de tiempo a la formación del desarrollador ajustando dicha formación a las necesidades de cada ciclo. Por eso se deben seleccionar los requisitos de cada ciclo de modo que se repartan entre los ciclos los requisitos que impliquen nuevos conocimientos que adquirir.
Plan de corrección	Considerar eliminar algún requisito menor para cuadrar los tiempos.

Tabla 3.4: Riesgo-01.

Identificador	Riesgo-02
Descripción	Pérdida de información relativa al proyecto.
Probabilidad	Media
Impacto	Alto
Exposición	Alta
Plan de prevención	Poner énfasis en la Gestión de la Configuración. Implementar una política de backups. Utilizar un sistema de almacenamiento remoto tipo DropBox[3].
Plan de corrección	Utilizar las copias de seguridad para restaurar los daños ocasionados.

Tabla 3.5: Riesgo-02.

Identificador	Riesgo-03
Descripción	Retraso en alguno de los incrementos.
Probabilidad	Alta
Impacto	Alto
Exposición	Alta
Plan de prevención	En las reuniones diarias realizadas durante los ciclos *sprint* se debe revisar la planificación temporal de dicho *sprint* para realizar los ajustes necesarios.
Plan de corrección	Considerar eliminar algún requisito menor para cuadrar los tiempos o la realización de horas extras si no es posible eliminar ningún requisito.

Tabla 3.6: Riesgo-03.

Identificador	Riesgo-04
Descripción	Error en la estimación de horas para cumplir los requisitos.
Probabilidad	Alta
Impacto	Alto
Exposición	Alta
Plan de prevención	En la retroalimentación que se lleva a cabo al finalizar cada *sprint*, revisar las estimaciones de tiempo de los requisitos para detectar posibles desfases en las mismas.
Plan de corrección	Considerar eliminar algún requisito menor para cuadrar los tiempos o la realización de horas extras si no es posible eliminar ningún requisito.

Tabla 3.7: Riesgo-04.

Identificador	Riesgo-05
Descripción	Indisponibilidad de los clientes.
Probabilidad	Media
Impacto	Alto
Exposición	Alto
Plan de prevención	Buscar otras personas en las que los clientes principales, en nuestro caso los tutores del proyecto, puedan delegar sus funciones en caso de ausencia. Por ejemplo algún investigador que esté relacionado con el tema de la clasificación de imágenes hiperespectrales.
Plan de corrección	Contactar con las personas en las que el cliente a delegado sus funciones.

Tabla 3.8: Riesgo-05.

Identificador	Riesgo-06
Descripción	Imposible incluir las técnicas de clasificación en la aplicación de forma que se puedan cargar dinámicamente, es decir, que se puedan incorporar a la ejecución sin tener que volver a compilar la aplicación. Esto se puede deber a que las librerías de las que dependen los códigos de estas técnicas de clasificación solo permitan generar una librería estática y no dinámica, que es lo que se busca.
Probabilidad	Media
Impacto	Alto
Exposición	Alta
Plan de prevención	Realizar las pruebas de integración entre la aplicación y los distintos plugins en ciclos de *sprint* tempranos.
Plan de corrección	Integrar las técnicas dentro de la propia aplicación.

Tabla 3.9: Riesgo-06.

Identificador	Riesgo-07
Descripción	Incremento excesivo del número de requisitos añadidos en las fases de planificación de los ciclos de *sprint*.
Probabilidad	Media
Impacto	Alto
Exposición	Alta
Plan de prevención	Para cada una de las modificaciones o solicitudes de requisitos nuevos, discutir la viabilidad de los mismos sin comprometer el proyecto.
Plan de corrección	Eliminar aquellos requisitos que sean prescindibles en el proyecto hasta cuadrar los tiempos.

Tabla 3.10: Riesgo-07.

Identificador	Riesgo-08
Descripción	El diseño de la interfaz no es del agrado del cliente.
Probabilidad	Alta
Impacto	Medio
Exposición	Alta
Plan de prevención	En la primera fase de scrum, realizar un boceto del aspecto que tendrá la aplicación y presentárselo al cliente.
Plan de corrección	Tener en cuenta para el siguiente ciclo de scrum que se tiene que reservar tiempo para la modificación de la interfaz.

Tabla 3.11: Riesgo-08.

3.3. Gestión Temporal

Teniendo en cuenta que la metodología *scrum* no planifica la duración ni el contenido de cada uno de los ciclos *sprint* que componen el proyecto, sino que son el producto de un acuerdo al que se llega en cada una de las reuniones previas al inicio de cada ciclo, tal y como ya se ha explicado en la sección 3.1.1, el EDT de la figura 3.2 es el resultado final de las distintas actualizaciones que ha ido sufriendo dicho documento a medida que se conocían la duración y el contenido de cada una de las iteraciones que forman el proyecto.

Siguiendo el EDT mostrado en la figura 3.2 se describirán cada una de las etapas y su duración.

3.3.1. Primera Fase

En la primera fase del proyecto se lleva a cabo procesos relacionados con la iniciación del proyecto tales como la gestión del proyecto, la gestión de riesgos, la creación de la lista de producto y el diseño de la arquitectura del sistema.

- **Gestión del proyecto**

En esta etapa se definirá el alcance del proyecto, se creará un EDT inicial y se seleccionarán las herramientas que se consideren necesarias para gestionar, planificar y desarrollar el proyecto de una forma ágil.

Duración: 6 horas

- **Gestión riesgos**

En esta etapa se procede a definir y valorar los posibles riesgos que pueden aparecer a lo largo del proyecto. Tal y como se define en la metodología *scrum*, el documento de riesgos se puede ampliar si en cualquiera de las fases *sprint* apareciesen nuevos riesgos que no se habían tenido en cuenta en la definición inicial.

Duración: 2 horas

- **Generar lista de producto**

En esta etapa se elabora una lista exhaustiva de los requisitos del sistema. Posteriormente se procederá a priorizar dicha lista y estimar el tiempo necesario para cada uno de los requisitos. El resultado final de esta etapa será el *product backlog*[4], que es un documento dinámico que incorpora las necesidades del sistema y que se mantiene durante todo el ciclo de vida.

Duración: 4 horas

- **Diseño arquitectónico**

Esta etapa se realizará el diseño de la arquitectura del sistema en la que se definirá como debe organizarse y como tiene que diseñarse la estructura global del mismo.

Duración: 4 horas

[4] lista de producto

3.3.2. Segunda Fase

En la segunda fase del proyecto se llevará a cabo el desarrollo en si del software objetivo del presente proyecto. Para ello se divide esta fase en los cinco ciclos *sprint* que se definieron en el EDT realizado en la primera fase.

Todos los ciclos *sprint* se descomponen a su vez en las fases de: planificación, diseño, desarrollo y pruebas, revisión y retrospectiva. Debemos tener en cuenta también, que cada día del ciclo *sprint* se debe realizar un scrum diario, que no es más que una reunión entre los miembros del equipo desarrollador para sincronizar las actividades y crear un plan de trabajo para ese día. Esta reunión no debería exceder de 15 minutos, pero en nuestro caso, debido a que el equipo lo compone una sola persona, hemos considerado suficiente definir un tiempo de cinco minutos para estos scrums diarios.

• **Primer ciclo de sprint**

El primer ciclo de *sprint* se compone de:

– **Planificación:** En esta fase se realiza una reunión para analizar cuales son los requisitos, de la lista de productos (*product backlog*), que se van a considerar para este *sprint*. El resultado final es el *sprint backlog*, que no es más que la lista de requisitos seleccionados.

Para este *sprint* se han seleccionado 8 requisitos de los 38 que había en la lista de productos.

Duración: 2 horas

– **Diseño:** En esta fase se realiza el diseño software en función de los requisitos del *sprint backlog*. En este caso se ha diseñado la estructura principal que almacenará los datos de las imágenes hiperespectrales, así como distintas funciones para el acceso a dichos datos.

Duración: 13 horas

– **Desarrollo y pruebas:** En esta fase se realizan tanto el desarrollo como las pruebas unitarias del software de los 8 requisitos que componen el *sprint backlog*.

Duración: 52 horas

– **Revisión:** En esta fase se inspecciona el incremento y se adapta la lista de producto si fuese necesario, por ejemplo, por no haber realizado alguno de los requisitos del *sprint backlog*. En este *sprint* no hizo falta adaptar la lista

de producto puesto que se completaron todos los requisitos seleccionados para el mismo. Así mismo, se mostró el producto final del incremento a los clientes. De esta revisión se observo la necesidad de crear un filtro a la hora de abrir tanto el fichero de datos de la imagen hiperespectral como el fichero del *ground truth*, de forma que por defecto solo se mostrasen los ficheros permitidos (MATLAB, RAW, HRW).

Duración: 1 hora

- **Retrospectiva:** Esta fase es una oportunidad para el equipo de desarrollo de inspeccionarse a si mismos y crear un plan de mejoras que sean abordadas durante el siguiente *sprint*. En esta reunión se propuso utilizar una herramienta de diseño de interfaces gráficas para poder agilizar la creación de las mismas ya que esto permitiría construir un boceto de las ventanas antes de pasar a su construcción, en lugar de utilizar un bucle de construye-prueba donde vamos construyendo la interfaz y visualizando el resultado hasta que conseguimos el aspecto deseado.

  Duración: 15 minutos

- **Scrums diarios:** La duración de este *sprint* fue de 24 días, por lo tanto se han realizado un total de 24 reuniones.

  Duración: 2 horas

• **Segundo ciclo de sprint**

El segundo ciclo de sprint se compone de:

- **Planificación:** Para este *sprint* se han seleccionado 9 requisitos de los 30 restantes que había en la lista de productos.

  Duración: 1 hora

- **Diseño:** Tal y como se especificaba en plan de prevención del "Riesgo-06" en la tabla 3.9, se ha incluido en este *sprint* el desarrollo de uno de los plugins para utilizar en la clasificación, así como otras funcionalidades de la aplicación que permitan verificar el correcto acoplamiento con esta. En el apéndice B podemos ver la estructura que se ha definido para la comunicación entre la aplicación y los plugins.

  Duración: 10 horas

- **Desarrollo y pruebas:** En esta fase se realiza tanto el desarrollo como las pruebas unitarias del software de los 8 requisitos que componen el *sprint*

backlog.

Duración: 75 horas

- **Revisión:** En esta reunión se muestra el resultado del ciclo *sprint* a los clientes, los cuales sugieren cambiar los iconos que aparecen en los botones de la ventana de configuración de la clasificación por el nombre de sus acciones, por ejemplo, cambiar el icono que simboliza añadir por el texto "Añadir". En este *sprint* tampoco hizo falta adaptar la lista de producto, se completaron todos los requisitos esperados para el mismo.

  Duración: 1 hora

- **Retrospectiva:** Tras la construcción del primer plugin, se toma nota de algunas complicaciones que han surgido en su construcción para tenerlas en cuenta a la hora de desarrollar el resto de plugins.

  Duración: 15 minutos

- **Scrums diarios:** La duración de este sprint fue de 30 días, por lo tanto se han realizado un total de 30 reuniones.

  Duración: 2 horas y 30 minutos

• **Tercer ciclo de sprint**

El tercer ciclo de *sprint* se compone de:

- **Planificación:** Para este *sprint* se han seleccionado 9 requisitos de los 21 que quedaban en la lista de productos.

  Duración: 1 hora

- **Diseño:** En este sprint se ha diseñado entre otros, el zoom y el histograma, a la vez que se ha finalizado la herramienta de configuración de plugins. Otro de los aspectos importantes diseñado en este *sprint* ha sido pasar parámetros a los plugins, para lo cual se ha utilizado, entre otras herramientas, el diseñador de interfaces Glade [5]

  Duración: 8 horas

- **Desarrollo y pruebas:** En esta fase se realiza tanto el desarrollo como las pruebas unitarias del software de los 9 requisitos que componen el *sprint*

[5]https://glade.gnome.org/

backlog.

Duración: 63 horas

– **Revisión:** En este *sprint* tampoco fue necesario adaptar la lista de producto ya que se completaron todos los requisitos que componían el *sprint*. El usuario pudo probar el resultado del *sprint* y no se realizaron comentarios sobre el mismo.

Duración: 1 hora

– **Retrospectiva:** En esta reunión se consideró necesario profundizar un poco más en el manejo de *Pango*[6] ya que uno de los requisitos que quedan pendientes en la lista de requisitos se encarga de mostrar los distintos resultados que produzcan los plugins que se van ejecutando en un contenedor como puede ser un "GtkTextViev[7]". Un mayor manejo de *Pango* permitirá una visualización de los datos más clara para el usuario.

Duración: 15 minutos

– **Scrums diarios:** La duración de este *sprint* fue de 26 días, por lo tanto se han realizado un total de 26 reuniones.

Duración: 2 horas y 10 minutos

• **Cuarto ciclo de sprint**

El cuarto ciclo de *sprint* se compone de:

– **Planificación:** Para este *sprint* se han seleccionado 6 requisitos de los 12 restantes en la lista de productos.

Duración: 1 hora

– **Diseño:** El diseño en este *sprint* corresponde al del módulo de ejecución y visualización de la clasificación, así como el desarrollo de más plugins.

Duración: 7 horas

– **Desarrollo y pruebas:** En esta fase se realiza tanto el desarrollo como las pruebas unitarias del software de los 6 requisitos que componen el *sprint*

[6]Pango es una librería para el manejo de texto. *Pango* proporciona las herramientas necesarias para insertar texto en contenedores GTK+ como por ejemplo *GtkTextView*, *GtkLabel*, *GtkEntry*, etc.

[7]Es un contenedor disponible en GTK+, que permite la inclusión de texto.

backlog.

Duración: 40 horas

- **Revisión:** Tras ejecutar el software desarrollado en este ciclo, los clientes comentan que el tamaño de letra de los resultados que se muestran tras la ejecución de la clasificación no es el adecuado y que debería sustituirse por un tamaño más grande. En este *sprint* no se adaptó la lista de producto ya que se completaron todos los requisitos.

  Duración: 1 hora

- **Retrospectiva:** En esta reunión se a tomado nota sobre algunos aspectos que han surgido a la hora del desarrollo del botón "Abortar" como por ejemplo, la liberación de memoria a la hora de abortar una ejecución.

  Duración: 15 minutos

- **Scrums diarios:** La duración de este *sprint* fue de 17 días, por lo tanto se han realizado un total de 17 reuniones.

  Duración: 1 hora y 25 minutos

• **Quinto ciclo de sprint**

El quinto ciclo de *sprint* se compone de:

- **Planificación:** Para este *sprint* se han seleccionado 5 requisitos de los 6 que había en la lista de productos, dejando uno de los requisitos de la lista de producto sin desarrollar, para poder ajustar el tiempo del proyecto, tal y como especificaba en el plan de prevención del "Riesgo-04" 3.7. El requisito desestimado es un requisito opcional, por lo tanto no tendrá una gran impacto en la calidad final del desarrollo.
  Duración: 1 hora

- **Diseño:** El diseño en este *sprint* corresponde al del módulo de almacenamiento del resultado de la clasificación, así como el desarrollo de los últimos plugins.

  Duración: 4 horas

- **Desarrollo y pruebas:** En esta fase se realiza tanto el desarrollo como las pruebas unitarias del software de los 5 requisitos que componen el *sprint backlog.*

  Duración: 49 horas

– **Revisión:** Los clientes pudieron probar el software final, quedando bastante satisfechos con el resultado. Al ser el último *sprint*, no fue necesario adaptar la lista de producto.

Duración: 1 hora

– **Retrospectiva:** En esta reunión no se han tenido en cuenta mejoras para el siguiente *sprint* ya que este no existe, pero si se han tenido en cuenta algunos detalles referentes a la implementación de plugins que han surgido durante el desarrollo del requisito funcional 31 y del requisito funcional 33.

Duración: 15 minutos

– **Scrums diarios:** La duración de este *sprint* fue de 19 días, por lo tanto se han realizado un total de 19 reuniones.

Duración: 1 hora y 35 minutos

3.3.3. Tercera Fase

En la última fase del proyecto se lleva a cabo procesos relacionados con finalización del mismo tales como completar la documentación y el cierre del proyecto.

• Completar la documentación

En esta sección se completa la documentación del proyecto y se construye el manual de usuario.

Duración: 90 horas

• Cierre del proyecto

El cierre del proyecto hace referencia a la entrega al cliente del software y la documentación generada.

Duración: 30 minutos

3.4. Gestión del Coste

Debido al carácter de Trabajo de Fin de Grado, los costes manejados en este apartado son teóricos.

- Consideraremos que no se dispone ni del *software* ni del *hardware* necesario para llevar a cabo el proyecto.

  - Ordenador con un mínimo de 4 GB de memoria RAM y un procesador de 64 bits.
  - Una versión de Ubuntu de 64 bits.
  - Entorno de desarrollo Netbeans 7.4
  - Conexión a Internet.

- Dada la condición de estudiante de la USC del desarrollador del proyecto, se usará la conexión a internet que la universidad provee a sus estudiantes.

- La USC no afrontará gastos adicionales, más allá del suministro de conexión a internet y de proveer un puesto de trabajo temporal en el CITIUS[8].

- Los usuarios que utilizarán el *software* disponen del material necesario para su uso.

  - Ordenador con sistema operativo Linux.

- El desarrollador del proyecto no se hará cargo de los gastos de consumibles tales como impresión de documentos, papel, bolígrafos, etc. . . , que se considerarán gastos imputables al proyecto.

- Los costes imputados finalmente al proyecto serán los relativos a la adquisición del material necesario para el desarrollo del proyecto y la contratación de recursos necesarios.

3.4.1. Costes Relativos al Personal

En este apartado se detallarán los costes correspondientes a los recursos humanos necesarios para el desarrollo del proyecto.

Tal y como se explica en el sección C.2.2 del apéndice C, en la metodología *Scrum* podemos diferenciar varios roles, a saber, *product owner*, *scrum master* y el equipo de desarrolladores, dentro del cual encontraremos a su vez programadores y analistas-programadores. En la tabla 3.12 podemos ver el salario bruto de cada uno de los roles anteriormente enumerados, según la información obtenida de la página web de Infojobs[34].

Tomando como referencia los datos ofrecidos por la Seguridad Social[35], debemos tener en cuenta la siguiente información para calcular los costes de personal:

Grupo de cotización: Todos los roles que intervienen en el proyecto se encuentran en el grupo de cotización uno.

[8] https://citius.usc.es/

Rol	Base de Cotización
Scrum Master	33.375,00 €
Product Owner	29.946,00 €
Analista-Programador	23.288,00 €
Programador	16.433,00 €

Tabla 3.12: Tabla salarial.

Base de cotización: La base de cotización mínima y máxima para el grupo de cotización uno es de 1.051,50 € y 3.597,00 € respectivamente.

Contingencias comunes: La empresa paga por el empleado el 23,60 % sobre la base de cotización.

Desempleo: Suponiendo un empleado con un contrato fijo estaríamos en el caso de un tipo general, por lo que la empresa paga por el empleado el 5.50 % sobre la base de cotización.

FOGASA[9]**:** A la empresa le corresponde pagar por el empleado un 0.20 % sobre la base de cotización.

Formación: En este caso la empresa tiene que pagar por el empleado el 0.60 % sobre la base de cotización.

En la tabla 3.13 podemos ver el coste que la empresa tiene que soportar mensualmente, por cada uno de los roles que intervienen en el proyecto. La cotización del 29,9 % viene dada por la suma de: la formación, FOGASA, desempleo y las contingencias comunes.

Rol	Salario Bruto Mes	Cotización (29.9 %)	Total
Scrum Master	2.781,25 €	831,59 €	3.612,84 €
Product Owner	2.495,50 €	746,15 €	3.241,65 €
Analista-Programador	1.940.66 €	580,26 €	2.520,92 €
Programador	1.369,41 €	409,45 €	1.778,87 €

Tabla 3.13: Coste para la empresa por rol.

Tal y como se puede apreciar en los datos de la tabla 3.13, todos los roles tienen su base de cotización entre el mínimo y máximo establecido por la ley

[9] http://www.empleo.gob.es/fogasa/

para el grupo de cotización uno.

Para el cálculo del coste por horas de cada rol, se ha considerado una jornada de 8 horas diarias y 22 días trabajados por mes, lo que hacen un total de 176 horas/mes. En la tabla 3.14 podemos ver el resultado de los cálculos.

Rol	Coste Mensual para la Empresa	Coste por Horas
Scrum Master	3.612,84 €	20,53 €
Product Owner	3.241,65 €	18,42 €
Analista-Programador	2.520.92 €	14,32 €
Programador	1.778,87 €	10,11 €

Tabla 3.14: Coste para la empresa en horas por rol.

Antes de proceder al cálculo del coste final, debemos tener en cuenta que debido a las características de este trabajo, y a la forma de trabajar en *scrum*, en muchas ocasiones se dará la situación en la que varios roles deban trabajar a la vez, tal y como ocurre en las reuniones, revisiones y retrospectivas. Para ese tipo de situaciones se dividirá el tiempo total de la tarea entre los distintos roles implicados en la misma.

También debemos apuntar que se ha considerado a cada uno de los tutores del proyecto como clientes de la aplicación, por lo cual no se les imputará ningún gasto, a pesar de que para el proyecto se deben incluir 11,25 horas de tutorías y evaluación. Se estima que el tiempo invertido en las reuniones con los clientes realizadas a lo largo de las distintas iteraciones, compensan las horas de tutorías, ya que en el contexto de este proyecto ambas tienen la misma finalidad.

Finalmente, una vez conocido el coste total por horas de cada rol, procedemos a calcular el coste del proyecto, teniendo en cuenta la gestión temporal de la sección 3.3.

El coste final del proyecto relativo al personal asciende a **5.303,00 €**, tal y como se refleja en la tabla 3.15.

3.4.2. Costes Relativos al Material

Hardware

El único elemento *hardware* que se va a necesitar para el desarrollo del proyecto es un ordenador completo. El precio de un ordenador con las características necesarias es de 650 €. El tiempo de vida útil de un ordenador va desde los 4

Tarea	Duración (h)	Coste (€)
Primera Fase	16	245,52 €
Segunda Fase	342,88	3.759,47 €
Primer Ciclo Sprint	70,25	793,13 €
Segundo Ciclo Sprint	89,75	974,35 €
Tercer Ciclo Sprint	75,40	819,15 €
Cuarto Ciclo Sprint	50,66	561,21 €
Quinto Ciclo Sprint	56,82	611,63 €
Tercera Fase	90,50	1.298,01 €
TOTAL	**449,38**	**5.303,00 €**

Tabla 3.15: Tabla de costes de personal.

hasta los 6 años, una media de 5 años. Esta medida se utilizará para conocer cual es el coste por horas de su utilización:

$$\frac{precio\ (€)}{horas\ vida\ util\ (hora)} \tag{3.1}$$

Para realizar los cálculos vamos a suponer 8 horas laborables al día y 22 días al mes. Por lo tanto el número de horas de vida útil del *hardware* es:

$$5\ años \times 12\ meses \times 22\ dias\ laborales\ al\ mes \times 8\ horas\ día = 10{,}560{,}00\ horas \tag{3.2}$$

Dado que el proyecto ha tenido una duración de 449,38 horas, el precio del ordenador para este proyecto es de:

$$\frac{650\ €}{10{,}560\ horas} \times 449{,}38\ horas = 27{,}66\ € \tag{3.3}$$

Por lo tanto el coste final del ordenador para este proyecto es de **27,66 €**.

Otros materiales

A continuación se detallan otros materiales que se han necesitado para el desarrollo y finalización del proyecto:

- Un paquete de folios: 3,50 €.
- Bolígrafos varios: 2,00 €.
- 4 copias impresas de la memoria: 130,00 €.
- 4 CDs: 4,00 €.

La suma del coste de estos materiales asciende a **139,50 €**

3.4.3. Costes Relativos al Software

Todo el *software* utilizado para el desarrollo del proyecto es de uso gratuito.

3.4.4. Financiamiento

Ya que se trata de un proyecto de fin de grado, no resulta necesario detallar el flujo de caja del mismo.

3.4.5. Total Costes

En la tabla 3.16 podemos observar el resumen del coste de este proyecto, que en total es de **5.470,16 €**.

Descripción	Coste (h)
Costes relativos al personal	5.303,00 €
Costes relativos al hardware	27,66 €
Costes relativos otros materiales	139,50 €
TOTAL	**5.470,16 €**

Tabla 3.16: Costes totales.

3.5. Gestión de la Configuración

La gestión de la configuración[28] es un proceso cuyo propósito es establecer y mantener la integridad del trabajo a través de:

- La identificación de los elementos/productos que van a ser controlados.
- La definición de un procedimiento para el control de los productos.
- El registro/informe del estado de los productos.
- Las auditorías de configuración.

Además la etapa de documentación se alarga durante todo el periodo de vida del proyecto y resulta imprescindible poder recuperar la información generada durante esta etapa en caso de producirse algún problema.

3.5.1. Gestión del Código Fuente

Es necesario asegurar la validez del producto, esto se consigue realizando un control de cambios adecuado que tiene como finalidad la disponibilidad de una versión estable en cada etapa de la implementación.

La gestión de la configuración del código fuente se llevará a cabo utilizando la herramienta Git[10], un software de código abierto que está ya integrado en el entorno de desarrollo que se va a utilizar, Netbeans[11]. Se creará un repositorio para cada una de los proyectos que se creen en Netbeans. Dentro de cada repositorio se creará un trunk y todos los branch necesarios para asegurar la integridad de los datos. A su vez todos los repositorios estarán almacenados en una cuenta de DropBox.

3.5.2. Gestión de la Documentación

Para Gestionar la documentación se utilizarán herramientas disponibles gratuitamente en Internet, como puede ser DropBox [12]. El control de versiones proporcionado por esta herramienta resulta suficiente para esta tarea.

[10]http://gitref.org/
[11]https://netbeans.org/
[12]https://www.dropbox.com/

Capítulo 4

Catálogo de Requisitos

Este capítulo representa la definición de requisitos que serán soportados por el sistema resultante del proyecto. Se establecen los requisitos necesarios para que el sistema a desarrollar cumpla con las necesidades del cliente, utilizando dichos requisitos como referencia para el desarrollo del sistema, de manera que permita revisar el cumplimiento de la funcionalidad acotando su alcance.

4.1. Glosario

4.1.1. Definiciones

Proceso de clasificación: En el contexto de este proyecto es el proceso mediante el cual se determina para cada píxel a qué clase pertenece de entre un conjunto de clases disponibles.

IEEE830: Estándar para la especificación de requisitos software.

Ciclos de sprint: Cada una de las iteraciones en las que se divide el desarrollo del código de la aplicación.

Imagen Hiperespectral: Imagen en la que cada píxel contiene un espectro continuo que permite su identificación en base a las propiedades de reflectancia del material que lo compone.

Ground Truth: Archivo que contiene la clasificación real de los píxeles de una determinada imagen hiperespectral y que es usado como referencia para comprobar la calidad de la clasificación obtenida mediante los distintos métodos.

Watershed: Es una técnica no supervisada de segmentación de imágenes basada en identificar regiones especialmente adecuadas para imágenes de bajo contraste. Se basa en aplicar conceptos de topografía ya que la imagen en escala de grises se representa como un perfil topográfico donde la altura del píxel está directamente relacionada con su nivel de gris. Se simula la inundación del perfil topográfico de modo que las líneas divisorias entre cuencas geográficas corresponden a las llamadas líneas de *watershed*, líneas que separan las diferentes regiones.

Histograma: En este proyecto llamaremos histograma a la representación gráfica de los componentes de reflectancia espectral de un píxel de la imagen hiperespectral.

Banda de una imagen: La banda de una imagen se corresponde con los valores de reflectancia a una determinada longitud de onda de todos los píxeles de la imagen hiperespectral.

Plugin: Es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. En este caso se aplica a cada una de las técnicas de clasificación que se pueden añadir al programa.

Criterio de Validación: Funcionalidad, característica o respuesta esperada de un requisito para ser considerado válido.

Proceso de Clasificación: Es el recorrido que debe seguir la aplicación, ejecutando los distintos plugins configurados previamente, para clasificar la imagen hiperespectral.

4.1.2. Acrónimos

IEEE: Del inglés *"Institute of Electrical and Electronic Engineers"*. Es la mayor asociación profesional del mundo dedicada al avance de la innovación tecnológica y excelencia en beneficio de la humanidad[31].

ELM: Del inglés *"Extreme Learning Machine"*. Es un método de aprendizaje basado en redes neuronales llamadas *single-hidden layer feedforward neural networks* (SLFNs). El algoritmo ELM se caracteriza porque permite elegir de manera aleatoria algunos parámetros de las redes neuronales. Es un algoritmo mucho más rápido que otros algoritmos de aprendizaje y es sencillo de computar. Para su computación hacen falta una fase de entrenamiento y una fase posterior que en nuestro caso permitirá clasificar los píxeles de la imagen hiperespectral.

SVM: Del inglés *"Suppport Vector Machines"*. Es una técnica de aprendizaje estadística supervisada para clasificación de conjuntos de datos. Consiste en una etapa de entrenamiento a partir de muestras cuyas clases son conocidas y de una etapa posterior de clasificación de los restantes datos del conjunto. En su forma más simple SVMs son clasificadores lineales binarios que asignan a cada muestra una clase de entre dos clases posibles. A partir de los clasificadores binarios se pueden clasificar conjuntos de datos donde más de dos clases están presentes. Esta es una técnica muy adecuada para clasificación de imágenes hiperespectrales de teledetección ya que la clasificación con SVM es de buena calidad aunque el número de muestras de que se disponga sea pequeño.

MV: Del inglés *"Majority Voting"*. Es un algoritmo que permite combinar información que proviene de diferentes fuentes basándose en el cálculo de mayorías. En nuestro contexto puede permitir combinar información de clasificación proveniente de diferentes clasificadores o bien combinar información espacial derivada, por ejemplo, de un mapa de segmentación con información espectral generada por alguna técnica de clasificación como por ejemplo SVM, obteniendo de esa forma una clasificación que integre información espacial e información espectral.

RQS: Del inglés *"Really Quick Shift"*. Es un algoritmo de segmentación no paramétrico similar al *Mean Shift*[32] que computa para cada píxel un valor de densidad considerando una ventana alrededor de dicho píxel, conectando posteriormente cada punto con el más próximo que tenga el valor de densidad más alto. El conjunto de conexiones alrededor de cada píxel constituye un árbol y da lugar a una región de segmentación.

GTK+: Del inglés "GIMP *Toolkit"*. Es una herramienta para la creación de interfaces de usuario gráficas.

GPL: Del inglés *"General Public License"*. Se refiere a una licencia que garantiza a los usuarios finales la libertad de usar, compartir, estudiar y modificar el software.

PGM: Del inglés *"Portable Gray Map Image"*. Se refiere a un formato de almacenamiento de imágenes.

AA: Del inglés *"Average Accuracy"*. El porcentaje de píxeles clasificados correctamente por cada clase.

OA: Del inglés *"Overall Accuracy"*. El porcentaje de píxeles clasificados correctamente.

HRW: Extensión que se aplica a los fichero que almacenan tanto los datos de las imágenes hiperespectrales como los datos del *ground truth* en el nuevo formato que se ha diseñado para ser utilizado por esta aplicación. Al contrario de lo que pasa con el formato RAW, HRW almacena toda la información necesaria para que el usuario no necesite proporcionar ningún dato a la aplicación para que esta sea capaz de interpretarla. Una imagen en formato HRW contiene una cabecera, las características de la imagen (dimensiones, formato y tipo de los datos, ...) y finalmente los datos de la imagen, todo ello comprimido en un único fichero.

MATLAB: Formato de los ficheros que maneja la aplicación MATLAB.

RAW: Uno de los tres formatos que es capaz de interpretar la aplicación y que contiene las dimensiones de la imagen seguidas da los datos de la misma.

4.2. Participantes en el Proyecto

En la tabla 4.1 podemos ver los distintos participantes del proyecto.

Participante	Función	Organización
Alberto Suárez	Desarrollar el proyecto	ETSE
Francisco Argüello	Director del proyecto/Cliente	DEC
Dora Blanco	Directora del proyecto/Cliente	CiTiUS

Tabla 4.1: Participantes del proyecto.

4.3. Objetivos del Sistema

Una vez el proyecto se encuentre en su etapa de explotación, debe cumplir los objetivos descritos en las tablas 4.2 a 4.10.

4.4. Actores

Hay un único actor que interaccionará con las funcionalidades de la aplicación tal y como podemos ver en la tabla 4.11.

Obj.01	Cargar imágenes.
Descripción	Debe ser posible cargar los ficheros que contienen tanto la imagen hiperespectral como su *ground truth* correspondiente para su posterior clasificación. Se debe admitir el manejo tanto de fichero en formato MATLAB como en formato RAW. También se debe crear un formato alternativo al formato RAW en el que el usuario no necesite conocer las características de la imagen para poder abrir el fichero.

Tabla 4.2: Objetivo del sistema Obj.01

Obj.02	Clasificar imágenes.
Descripción	Clasificar las imágenes hiperespectrales utilizando para ello las distintas técnicas disponibles en la aplicación.

Tabla 4.3: Objetivo del sistema Obj.02

Obj.03	Facilidad de ampliación.
Descripción	El desarrollo, por parte de terceros, de nuevas técnicas de clasificación para incluir en el programa debe ser sencillo.

Tabla 4.4: Objetivo del sistema Obj.03

Obj.04	Complejidad mínima.
Descripción	La aplicación debe contar con las herramientas indispensables para el estudio y comparación de los resultados ofrecidos, así como una interfaz agradable e intuitiva.

Tabla 4.5: Objetivo del sistema Obj.04

Obj.05	Visualizar resultados.
Descripción	La imagen resultante de la clasificación debe poder compararse visualmente con la imagen original, mostrando ambas en pantalla.

Tabla 4.6: Objetivo del sistema Obj.05

Obj.06	Almacenar resultados.
Descripción	Permitir almacenar los resultados gráficos y numéricos de la clasificación.

Tabla 4.7: Objetivo del sistema Obj.06

Obj.07	Visualizar bandas.
Descripción	Permitir al usuario desplazarse entre las distintas bandas de la imagen, así como ofrecer la posibilidad de almacenar la banda visualizada.

Tabla 4.8: Objetivo del sistema Obj.07

Obj.08	Zoom.
Descripción	Debe ser posible ampliar zonas de la imagen para apreciar mejor los distintos píxeles que la forman.

Tabla 4.9: Objetivo del sistema Obj.08

Obj.09	Histograma.
Descripción	Visualizar la firma espectral de los píxeles que forman la imagen y ofrecer herramientas que permitan su análisis y la comparación de las firmas espectrales de distintos píxeles.

Tabla 4.10: Objetivo del sistema Obj.09

Act.01	Usuario.
Descripción	Persona que hace uso de la aplicación.

Tabla 4.11: Actor 01

4.5. Catálogo de Requisitos de la Aplicación

Describe funcionalidades que debe desenvolver el producto. Se trata de requisitos fruto del análisis de los casos de uso que establecen el comportamiento de la aplicación y son la base de los distintos ciclos de sprint llevados a cabo durante el desarrollo del proyecto. Según su naturaleza, los requisitos serán catalogados en alguna de las siguientes categorías, según se especifica en el estándar IEEE830:

Requisitos Funcionales: Son las características que debe presentar la aplicación para dotarla de la funcionalidad requerida.

Requisitos No Funcionales: Requisitos que no otorgan funcionalidad a la aplicación.

- **Requisitos de Plataforma:** Representan requisitos independientes de la funcionalidad, pero dependientes de la plataforma sobre la que se va a ejecutar la aplicación o alguna plataforma específica con la que interaccionará alguna de las funcionalidades de la aplicación.
- **Requisitos de Interface de Usuario:** Requisitos que describen como debe ser representada la aplicación al usuario y como este interaccionará con ella.
- **Requisitos de Datos:** Describe como deben ser manejados los datos con los que trabaja la aplicación.
- **Restricciones:** Restricciones que presenta la aplicación.

4.5.1. Requisitos Funcionales

RF.01 Abrir fichero de datos en formato MATLAB.

RF.02 Abrir fichero de datos en formato RAW.

RF.03 Abrir fichero de datos en formato HRW.

RF.04 Mostrar imagen hiperespectral.

RF.05 Guardar fichero de datos en formato MATLAB.

RF.06 Guardar fichero de datos en formato RAW.

RF.07 Guardar fichero de datos en formato HRW.

RF.08 Zoom imagen hiperespectral.

RF.09 Coordenadas zoom imagen hiperespectral.

RF.10 Guardar banda.

RF.11 Abrir fichero *ground truth* en formato MATLAB.

RF.12 Abrir fichero *ground truth* en formato RAW.

RF.13 Abrir fichero *ground truth* en formato HRW.

RF.14 Mostrar *ground truth*.

RF.15 Zoom *ground truth*.

RF.16 Cambiar banda.

RF.17 Ir a banda.

RF.18 Mostrar histograma.

RF.19 Histograma en formato texto.

RF.20 Comparar histogramas.

RF.21 Guardar los datos del histograma.

RF.22 Mostrar información plugins cargados.

RF.23 Añadir plugin al proceso de clasificación.

RF.24 Eliminar plugin del proceso de clasificación.

RF.25 Ordenar los plugins del proceso de clasificación.

RF.26 Pasar parámetros a los plugins del proceso de clasificación.

RF.27 Ejecutar clasificación.

RF.28 Ver la salida de los distintos plugins ejecutados.

RF.29 Abortar la ejecución de la clasificación.

RF.30 Construir plugin ELM.

RF.31 Construir plugin SVM.

RF.32 Construir plugin MV.

RF.33 Construir plugin RQS.

RF.34 Construir plugin *Watershed*.

RF.35 Mostrar los resultados de la clasificación.

RF.36 Guardar la clasificación.

RF.37 Guardar imagen resultante del proceso de clasificación.

RF.38 Abrir clasificación.

4.5.2. Requisitos No Funcionales

Requisitos de Plataforma

RP.01 Desarrollada para el Sistema Operativo Linux.

Requisitos de Interfaz de Usuario

RI.02 Interfaz intuitiva y fácil de usar.

RI.03 Las opciones del menú tendrán teclas de atajo.

RI.04 Incluir ayuda.

RI.05 Barra de herramientas en la ventana principal.

Requisitos de Datos

RD.01 Diseñar formato nuevo(HRW) para almacenar datos.

Restricciones

R.01 Desarrollado sobre GTK+.

R.02 Desarrollado con lenguaje de programación C.

R.03 Licencia GPL.

4.6. Especificación de Requisitos

Descripción detallada de los requisitos enumerados en el apartado 4.5. Se detallará el requisito, su relevancia según la tabla 4.12 y el criterio de validación.

Relevancia	Descripción.
Esperado	El requisito resulta de gran importancia para conseguir el objetivo principal del proyecto.
Deseado	Conseguir el requisito mejora la calidad del proyecto, pero se puede conseguir el objetivo principal del proyecto prescindiendo del requerimiento.
Opcional	Requisito que mejora la calidad del proyecto en un grado menor que un requisito deseable. Únicamente se llevará a cabo si se dispone de tiempo suficiente.

Tabla 4.12: Descripción de las opciones de relevancia de un requisito.

4.6.1. Requisitos Funcionales

Entre la tabla 4.13 y la tabla 4.50 se describen todos los requisitos funcionales del proyecto.

RF. 01	Abrir fichero de datos en formato MATLAB.
Descripción	El usuario podrá abrir el fichero de datos de una imagen hiperespectral en formato MATLAB.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si la aplicación es capaz de abrir e interpretar los datos contenidos en el fichero de datos de la imagen hiperespectral en formato MATLAB.

Tabla 4.13: Requisito RF.01: Abrir fichero de datos en formato MATLAB.

RF. 02	Abrir fichero de datos en formato RAW.
Descripción	El usuario podrá abrir el fichero de datos de una imagen hiperespectral en formato RAW.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si la aplicación es capaz de abrir e interpretar los datos contenidos en el fichero de datos de la imagen hiperespectral en formato RAW.

Tabla 4.14: Requisito RF.02: Abrir fichero de datos en formato RAW.

RF. 03	Abrir fichero de datos en formato HRW.
Descripción	El usuario podrá abrir el fichero de datos de una imagen hiperespectral en formato HRW.
Relevancia	Deseado.
Criterio de validación	El requisito se considerará cumplido si la aplicación es capaz de abrir e interpretar los datos contenidos en el fichero de datos de la imagen hiperespectral en formato HRW.

Tabla 4.15: Requisito RF.03: Abrir fichero de datos en formato HRW.

RF. 04	Mostrar imagen hiperespectral.
Descripción	La aplicación mostrará una de las bandas de la imagen hiperespectral en una ventana.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si una vez abierto el fichero de datos de la imagen hiperespectral se muestra la banda seleccionada de dicha imagen.

Tabla 4.16: Requisito RF.04: Mostrar imagen hiperespectral.

RF. 05	Guardar fichero de datos en formato MATLAB.
Descripción	La aplicación debe ser capaz de almacenar en un fichero con formato MATLAB, la imagen hiperespectral que se encuentre abierta en la aplicación en ese momento.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si la aplicación es capaz de almacenar la imagen hiperespectral en formato MATLAB, permitiendo posteriormente abrir dicha imagen de forma satisfactoria.

Tabla 4.17: Requisito RF.05: Guardar fichero de datos en formato MATLAB.

RF. 06	Guardar fichero de datos en formato RAW.
Descripción	La aplicación debe ser capaz de almacenar en un fichero con formato RAW, la imagen hiperespectral que se encuentre abierta en la aplicación en ese momento.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si la aplicación es capaz de almacenar la imagen hiperespectral en formato RAW, permitiendo posteriormente abrir dicha imagen de forma satisfactoria.

Tabla 4.18: Requisito RF.06: Guardar fichero de datos en formato RAW.

RF. 07	Guardar fichero de datos en formato HRW.
Descripción	La aplicación debe ser capaz de almacenar en un fichero con formato HRW, la imagen hiperespectral que se encuentre abierta en la aplicación en ese momento.
Relevancia	Deseado.
Criterio de validación	El requisito se considerará cumplido si la aplicación es capaz de almacenar la imagen hiperespectral en formato HRW, permitiendo posteriormente abrir dicha imagen de forma satisfactoria.

Tabla 4.19: Requisito RF.07: Guardar fichero de datos en formato HRW.

RF. 08	Zoom imagen hiperespectral.
Descripción	El usuario podrá visualizar en una ventana aparte, el zoom realizado sobre una zona de la banda de la imagen hiperespectral mostrada en pantalla. La zona sobre la que se realizará el zoom viene determinada por el puntero del ratón cuando se desplaza por encima de la banda de la imagen hiperespectral mostrada en pantalla.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si el usuario puede realizar un zoom , guiado por el puntero del ratón sobre la banda de la imagen hiperespectral mostrada en pantalla.

Tabla 4.20: Requisito RF.08: Zoom imagen hiperespectral.

RF. 09	Coordenadas zoom imagen hiperespectral.
Descripción	El usuario podrá visualizar en la ventana de zoom correspondiente a la banda de la imagen hiperespectral mostrada en pantalla, las coordenadas del puntero sobre las que se encuentra el ratón en cada momento mientra dicha ventana esté operativa.
Relevancia	Deseado.
Criterio de validación	El requisito se considerará cumplido si la aplicación muestra en la ventana de zoom correspondiente a la banda de la imagen hiperespectral mostrada en pantalla, las coordenadas del puntero sobre las que se encuentra el ratón en cada momento, mientra dicha ventana esté operativa.

Tabla 4.21: Requisito RF.09: Coordenadas zoom imagen hiperespectral.

RF. 10	Guardar banda.
Descripción	El usuario podrá guardar, en un fichero con formato PGM, la banda de la imagen hiperespectral que se encuentra en ese momento en pantalla.
Relevancia	Deseado.
Criterio de validación	El requisito se considerará cumplido si la aplicación almacena la banda de la imagen hiperespectral visualizada por pantalla en un fichero con formato PGM de forma correcta, permitiendo posteriormente abrir dicho fichero con cualquier programa que sepa interpretar el formato PGM.

Tabla 4.22: Requisito RF.10: Guardar banda.

RF. 11	Abrir fichero *ground truth* en formato MATLAB.
Descripción	El usuario podrá abrir el fichero que contiene los datos del *ground truth*, de la imagen hiperespectral, en formato MATLAB.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si la aplicación es capaz de abrir e interpretar los datos contenidos en el fichero correspondiente al *ground truth* de la imagen hiperespectral, en formato MATLAB.

Tabla 4.23: Requisito RF.11: Abrir fichero *ground truth* en formato MATLAB.

RF. 12	**Abrir fichero *ground truth* en formato RAW.**
Descripción	El usuario podrá abrir el fichero que contiene los datos del *ground truth*, de la imagen hiperespectral, en formato RAW.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si la aplicación es capaz de abrir e interpretar los datos contenidos en el fichero correspondiente al *ground truth* de la imagen hiperespectral, en formato RAW.

Tabla 4.24: Requisito RF.12: Abrir fichero *ground truth* en formato RAW.

RF. 13	**Abrir fichero *ground truth* en formato HRW.**
Descripción	El usuario podrá abrir el fichero que contiene los datos del *ground truth*, de la imagen hiperespectral, en formato HRW.
Relevancia	Deseado.
Criterio de validación	El requisito se considerará cumplido si la aplicación es capaz de abrir e interpretar los datos contenidos en el fichero correspondiente al *ground truth* de la imagen hiperespectral, en formato HRW.

Tabla 4.25: Requisito RF.13: Abrir fichero *ground truth* en formato HRW.

RF. 14	**Mostrar *ground truth*.**
Descripción	La aplicación mostrará los datos del *ground truth* en color en una ventana.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si una vez abierto el fichero de datos del *ground truth* de la imagen hiperespectral se muestran dichos datos en color.

Tabla 4.26: Requisito RF.14: Mostrar *ground truth*.

RF. 15	**Zoom *ground truth*.**
Descripción	El usuario podrá visualizar en una ventana aparte, el zoom realizado sobre una zona del *ground truth*. La zona sobre la que se realizará el zoom viene determinada por el puntero del ratón.
Relevancia	Opcional.
Criterio de validación	El requisito se considerará cumplido si el usuario puede realizar un zoom , guiado por el puntero del ratón, sobre el *ground truth*.

Tabla 4.27: Requisito RF.15: Zoom *ground truth*.

RF. 16	**Cambiar banda.**
Descripción	El usuario podrá visualizar las distintas bandas de la imagen hiperespectral, utilizando la rueda del ratón.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si el usuario puede visualizar las distintas bandas de la imagen hiperespectral, usando la rueda del ratón para cambiar de banda.

Tabla 4.28: Requisito RF.16: Cambiar banda.

RF. 17	**Ir a banda.**
Descripción	El usuario podrá indicar una banda determinada, de la imagen hiperespectral, a la que desplazarse.
Relevancia	Deseado.
Criterio de validación	El requisito se considerará cumplido si el usuario puede indicarle al programa una banda, de la imagen hiperespectral, a la que desplazarse y este muestra dicha banda.

Tabla 4.29: Requisito RF.17: Ir a banda.

RF. 18	Mostrar histograma.
Descripción	El usuario podrá visualizar el histograma de un píxel seleccionado, mediante el ratón, sobre la banda de la imagen hiperespectral mostrada en pantalla. El histograma estará compuesto por los valores en las coordenadas del píxel en todas las bandas de la imagen hiperespectral.
Relevancia	Deseado.
Criterio de validación	El requisito se considerará cumplido si el usuario pude visualizar el histograma de todos los datos de las distintas bandas que componen la imagen hiperespectral en el píxel seleccionado.

Tabla 4.30: Requisito RF.18: Mostrar histograma.

RF. 19	Histograma en formato texto.
Descripción	El usuario podrá observar los distintos datos que conforman el histograma, en formato texto.
Relevancia	Deseado.
Criterio de validación	El requisito se considerará cumplido si el usuario puede visualizar los datos que componen el histograma, en formato texto.

Tabla 4.31: Requisito RF.19: Histograma en formato texto.

RF. 20	Comparar histogramas.
Descripción	El usuario podrá visualizar a la vez los histogramas de dos píxeles, quedando uno de los histogramas fijo y pudiendo variar el otro seleccionando otro píxel distinto.
Relevancia	Deseado.
Criterio de validación	El requisito se considerará cumplido si la aplicación permite visualizar el histograma de dos píxeles a la vez, permitiendo variar uno de los histogramas.

Tabla 4.32: Requisito RF.20: Comparar histogramas.

RF. 21	Guardar los datos del histograma.
Descripción	El usuario podrá almacenar los datos que forman el histograma en un fichero de texto.
Relevancia	Opcional.
Criterio de validación	El requisito se considerará cumplido si la aplicación permite almacenar los datos que forman el histograma en un fichero de texto.

Tabla 4.33: Requisito RF.21: Guardar los datos del histograma.

RF. 22	Mostrar información plugins cargados.
Descripción	El usuario dispondrá de una ventana donde se listarán todos los plugins disponibles en el directorio "plugins". De cada plugin que seleccione el usuario se mostrarán los siguientes datos: Nombre del plugin, autor, versión y una breve descripción.
Relevancia	Deseado.
Criterio de validación	El requisito se considerará completo si el usuario es capaz de visualizar la información anteriormente mencionada de cada uno de los plugins cargados del directorio "plugins"

Tabla 4.34: Requisito RF.22: Mostrar información plugins cargados.

RF. 23	Añadir plugin al proceso de clasificación.
Descripción	Se debe permitir al usuario añadir, seleccionándolo de una lista, los plugins que considere oportuno en cada de las fases de las que se compone el proceso de clasificación.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará completo si el usuario puede añadir plugins al proceso de clasificación en cada una de las fases que componen dicho proceso.

Tabla 4.35: Requisito RF.23: Añadir plugin al proceso de clasificación.

RF. 24	Eliminar plugin del proceso de clasificación.
Descripción	Se debe permitir al usuario eliminar los plugins que considere oportuno en cada de las fases de las que se compone el proceso de clasificación, que previamente se habían añadido a dicho proceso.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará completo si el usuario puede eliminar plugins del proceso de clasificación en cada una de las fases que componen dicho proceso, donde previamente se había añadido un plugin.

Tabla 4.36: Requisito RF.24: Eliminar plugin del proceso de clasificación.

RF. 25	Ordenar los plugins del proceso de clasificación.
Descripción	Se debe permitir al usuario ordenar los plugins que se han configurado en cada una de las fases de las que se compone el proceso de clasificación.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará completo si el usuario puede ordenar el listado de plugins del proceso de clasificación en cada una de las fases que componen dicho proceso.

Tabla 4.37: Requisito RF.25: Ordenar los plugins del proceso de clasificación.

RF. 26	Pasar parámetros a los plugins del proceso de clasificación.
Descripción	Se debe permitir al usuario configurar los parámetros que cada plugin necesite para su correcto funcionamiento, o usar los valores por defecto que propone la aplicación.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará completo si el usuario puede configurar los parámetros de todos los plugins que se han configurado para el proceso de clasificación, siempre y cuando dichos plugins necesiten parámetros.

Tabla 4.38: Requisito RF.26: Pasar parámetros a los plugins del proceso de clasificación.

RF. 27	Ejecutar clasificación.
Descripción	El usuario podrá ejecutar la clasificación, de la imagen hiperespectral cargada en el programa, procesando todos los plugins previamente configurados para dicha clasificación.
Relevancia	Esperado.
Criterio de validación	El requisito se considera completo si el usuario, después de haber configurado los plugins del proceso de clasificación, puede ejecutar dicha clasificación, llegando el proceso a su fin.

Tabla 4.39: Requisito RF.27: Ejecutar clasificación.

RF. 28	Ver la salida de los distintos plugins ejecutados.
Descripción	El usuario podrá visualizar la salida que los distintos plugins vayan generando según se van ejecutando en el proceso de clasificación.
Relevancia	Esperado.
Criterio de validación	El requisito se considera completo si se visualiza la salida que los distintos plugins vayan generando a medida que se ejecutan dentro del proceso de clasificación.

Tabla 4.40: Requisito RF.28: Ver la salida de los distintos plugins ejecutados.

RF. 29	Abortar la ejecución de la clasificación.
Descripción	El usuario podrá abortar la ejecución del proceso de clasificación.
Relevancia	Esperado.
Criterio de validación	El requisito se considera completo si tras la ejecución del proceso de clasificación, el usuario puede abortar dicho proceso.

Tabla 4.41: Requisito RF.29: Abortar la ejecución de la clasificación.

RF. 30	Construir plugin ELM.
Descripción	Se adaptará un código previamente desarrollado por el grupo de investigación, que aplica la técnica de clasificación ELM a una imagen hiperespectral, para que pueda ser utilizada por la aplicación.
Relevancia	Esperado.
Criterio de validación	El requisito se considera completo si el usuario puede aplicar la técnica ELM dentro del proceso de clasificación.

Tabla 4.42: Requisito RF.30: Construir plugin ELM.

RF. 31	Construir plugin SVM.
Descripción	Se adaptará un código previamente desarrollado por el grupo de investigación, que aplica la técnica de clasificación SVM a una imagen hiperespectral, para que pueda ser utilizada por la aplicación.
Relevancia	Esperado.
Criterio de validación	El requisito se considera completo si el usuario puede aplicar la técnica SVM dentro del proceso de clasificación.

Tabla 4.43: Requisito RF.31: Construir plugin SVM.

RF. 32	Construir plugin MV.
Descripción	Se adaptará un código previamente desarrollado por el grupo de investigación, que aplica el algoritmo MV, para que pueda ser utilizada por la aplicación.
Relevancia	Esperado.
Criterio de validación	El requisito se considera completo si el usuario puede aplicar el algoritmo MV dentro del proceso de clasificación.

Tabla 4.44: Requisito RF.32: Construir plugin MV.

RF. 33	Construir plugin RQS.
Descripción	Se adaptará un código previamente desarrollado por el grupo de investigación, que aplica la técnica de segmentación RQS a una imagen hiperespectral, para que pueda ser utilizada por la aplicación.
Relevancia	Esperado.
Criterio de validación	El requisito se considera completo si el usuario puede aplicar la técnica RQS dentro del proceso de clasificación.

Tabla 4.45: Requisito RF.33: Construir plugin RQS.

RF. 34	Construir plugin *Watershed*.
Descripción	Se adaptará un código previamente desarrollado por el grupo de investigación, que aplica la técnica de segmentación *Watershed* a una imagen hiperespectral, para que pueda ser utilizada por la aplicación.
Relevancia	Esperado.
Criterio de validación	El requisito se considera completo si el usuario puede aplicar la técnica *Watershed* dentro del proceso de clasificación.

Tabla 4.46: Requisito RF.34: Construir plugin *Watershed*.

RF. 35	Mostrar resultados de la clasificación.
Descripción	El usuario visualizará los resultados de la clasificación, que están compuestos por: el valor de OA, AA y la propia imagen clasificada.
Relevancia	Esperado.
Criterio de validación	El requisito se considera completo si el usuario puede visualizar la imagen resultado del proceso de clasificación y los valor OA y AA.

Tabla 4.47: Requisito RF.35: Mostrar resultados de la clasificación.

RF. 36	Guardar la clasificación.
Descripción	El usuario, una vez finalizado el proceso de clasificación, podrá almacenar en un fichero toda la información relativa el proceso de clasificación, esto es: todos los plugins que intervinieron en el proceso así como sus parámetros y en que fases estaban incluidos, los valores OA y AA y la imagen resultante de la clasificación.
Relevancia	Deseado.
Criterio de validación	El requisito se considerará completo si el fichero almacenado puede ser utilizado posteriormente para realizar una comparación entre clasificaciones.

Tabla 4.48: Requisito RF.36: Guardar la clasificación.

RF. 37	Guardar imagen resultante del proceso de clasificación.
Descripción	El usuario, una vez finalizado el proceso de clasificación, podrá almacenar en un fichero PGM la imagen resultante de dicho proceso.
Relevancia	Opcional.
Criterio de validación	El requisito se considerará completo si la aplicación almacena la imagen en formato PGM de forma correcta, permitiendo posteriormente abrir dicho fichero con cualquier programa que sepa interpretar el formato PGM.

Tabla 4.49: Requisito RF.37: Guardar imagen resultante del proceso de clasificación.

RF. 38	Abrir clasificación.
Descripción	El usuario podrá visualizar los resultados de una clasificación previamente almacenada en un fichero. Se mostrará en pantalla tanto la imagen resultante de la clasificación como los valores OA y AA y la configuración de los plugins implicados.
Relevancia	Opcional.
Criterio de validación	El requisito se considerará completo si la aplicación permite al usuario abrir una clasificación almacenada previamente en disco, mostrando en pantalla tanto la imagen resultante de la clasificación como los valores OA y AA y la configuración de los plugins, implicados dicha clasificación.

Tabla 4.50: Requisito RF.38: Abrir clasificación.

4.6.2. Requisitos no Funcionales

En esta sección se detallarán todos los requisitos no funcionales de la aplicación.

Requisitos de Plataforma

En la tabla 4.51 se detalla el único requisito de plataforma que debe cumplir la aplicación.

RP. 01	Desarrollado para el Sistema Operativo Linux.
Descripción	La aplicación debe funcionar sobre el Sistema Operativo Linux.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si la aplicación se ejecuta sobre el Sistema Operativo Linux.

Tabla 4.51: Requisito RP.01: Desarrollado para el Sistema Operativo Linux.

Requisitos de Interfaz de Usuario

Desde la tabla 4.52 hasta la tabla 4.55 se detallan los requisitos que debe cumplir la interfaz de usuario.

RI. 01	Interfaz intuitiva y fácil de usar.
Descripción	La aplicación debe funcionar sobre el Sistema Operativo Linux.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si la aplicación se ejecuta sobre el Sistema Operativo Linux.

Tabla 4.52: Requisito RI.01: Interfaz intuitiva y fácil de usar.

RI. 02	Las opciones del menú tendrán teclas de atajo.
Descripción	Las opciones de los menús deben tener teclas de atajo.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si la aplicación permite acceder a las distintas opciones utilizando combinaciones de teclas.

Tabla 4.53: Requisito RI.02: Las opciones del menú tendrán teclas de atajo.

RI. 03	Incluir ayuda.
Descripción	La aplicación debe incluir un menú de ayuda.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si la aplicación dispone de una opción de ayuda entre las opciones del menú.

Tabla 4.54: Requisito RI.03: Incluir ayuda.

RI. 04	Barra de herramientas en la ventana principal.
Descripción	La aplicación incluirá una barra de herramientas en la cual las opciones de los menús deben tener su correspondiente icono.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si la aplicación cuenta con una barra de herramientas con las opciones de los menús.

Tabla 4.55: Requisito RI.04: Barra de herramientas en la ventana principal.

Requisitos de Datos

Los requisitos de datos se muestran en la tabla 4.56.

RD. 01	Diseñar formato nuevo (HRW) para almacenar datos.
Descripción	Se debe diseñar el formato de un fichero (HRW) para almacenar una imagen hiperespectral de forma que el usuario no tenga que proporcionar ninguna información referente a la misma a la hora de abrirla en la aplicación.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si la aplicación es capaz de abrir los ficheros del nuevo formato (HRW) sin necesidad de que el usuario proporcione ninguna información relativa a la imagen hiperespectral que contienen.

Tabla 4.56: Requisito RD.01: Diseñar formato nuevo (HRW) para almacenar datos.

Restricciones

Entre la tabla 4.57 y la tabla 4.59 se describen las restricciones del proyecto.

R.01	Desarrollado sobre GTK+.
Descripción	La aplicación de debe desarrollar con el conjunto de bibliotecas multiplataforma GTK+.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si la aplicación se desarrolla sobre la biblioteca GTK+.

Tabla 4.57: Requisito R.01: Desarrollado sobre GTK+.

R.02	Desarrollado con lenguaje de programación C.
Descripción	La aplicación debe estar desarrollada utilizando el lenguaje de programación C.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si la aplicación se ha desarrollado con el lenguaje de programación C.

Tabla 4.58: Requisito R.02: Desarrollado con lenguaje de programación C.

R.03	Licencia GPL.
Descripción	La aplicación debe distribuirse bajo licencia GPL.
Relevancia	Esperado.
Criterio de validación	El requisito se considerará cumplido si la aplicación se distribuye bajo licencia GPL y esto se refleja en la aplicación.

Tabla 4.59: Requisito R.03: Licencia GPL.

4.7. Casos de Uso

En este apartado se presentará el diagrama principal de casos de uso de la aplicación, además se especificará cada uno de estos casos de uso siguiendo el formato de la tabla 4.60. Los casos de uso representan unidades funcionales de un sistema o subsistema en los que una o más actores interaccionan con el sistema para realizar acciones definidas. Su principal ventaja es la facilidad para su interpretación, lo que hace que sean especialmente útiles en la comunicación con el cliente.

4.7.1. Diagrama de Casos de Uso

El diagrama de casos de uso pretende esquematizar lo que se puede esperar de la aplicación a desarrollar con un simple golpe de vista, tal y como se puede observar en la figura 4.1. En ella podemos distinguir 2 subsistemas:

Subsistema Tratamiento Imágenes

Este subsistema es el encargado del tratamiento visual de las imágenes. Su cometido es dar funcionalidad a la apertura y almacenamiento de las imágenes y a su visualización en pantalla, tanto de la propia imagen como de los datos que la componen. De la tabla 4.61 a la tabla 4.71 podemos ver los distintos casos de uso que forman este subsistema.

Identificador	Clave que identifica un caso de uso. Se representará mediante el código CU.x, donde x representa el número de caso de uso.
Nombre	Nombre asociado al caso de uso
Propósito	Explicación general del caso de uso.
Actores	Nombre del actor(es) que participa(n) en el caso de uso.
Precondición	Condición que se debe verificar para que se pueda realizar la acción del caso de uso.
Postcondición	Define el estado en el que debe quedar el sistema una vez el caso de uso se completa correctamente.
Escenario Principal	Comportamiento del sistema en una situación válida del caso de uso.
Escenario Alternativo	Comportamiento del sistema para una situación distinta al curso normal de los eventos. El inicio de la numeración del escenario alternativo comienza en el punto siguiente al que se produce el desvio del escenario alternativo. Si existe alguna opción común a los casos alternativos, esta se marcará con un asterisco (*)
Prioridad	Define el grado de importancia del caso de uso dentro del sistema. Se contemplarán tres niveles de prioridad: ▪ Alta: Proporciona funcionalidad imprescindible relacionada con los objetivos del proyecto. ▪ Media: Proporciona funcionalidad deseable que mejora la calidad del proyecto. ▪ Baja: Proporcionan funcionalidad adicional no demasiado significativa para el proyecto.
Subsistema	Subsistema al que pertenece el caso de uso.

Tabla 4.60: Formato descripción caso de uso.

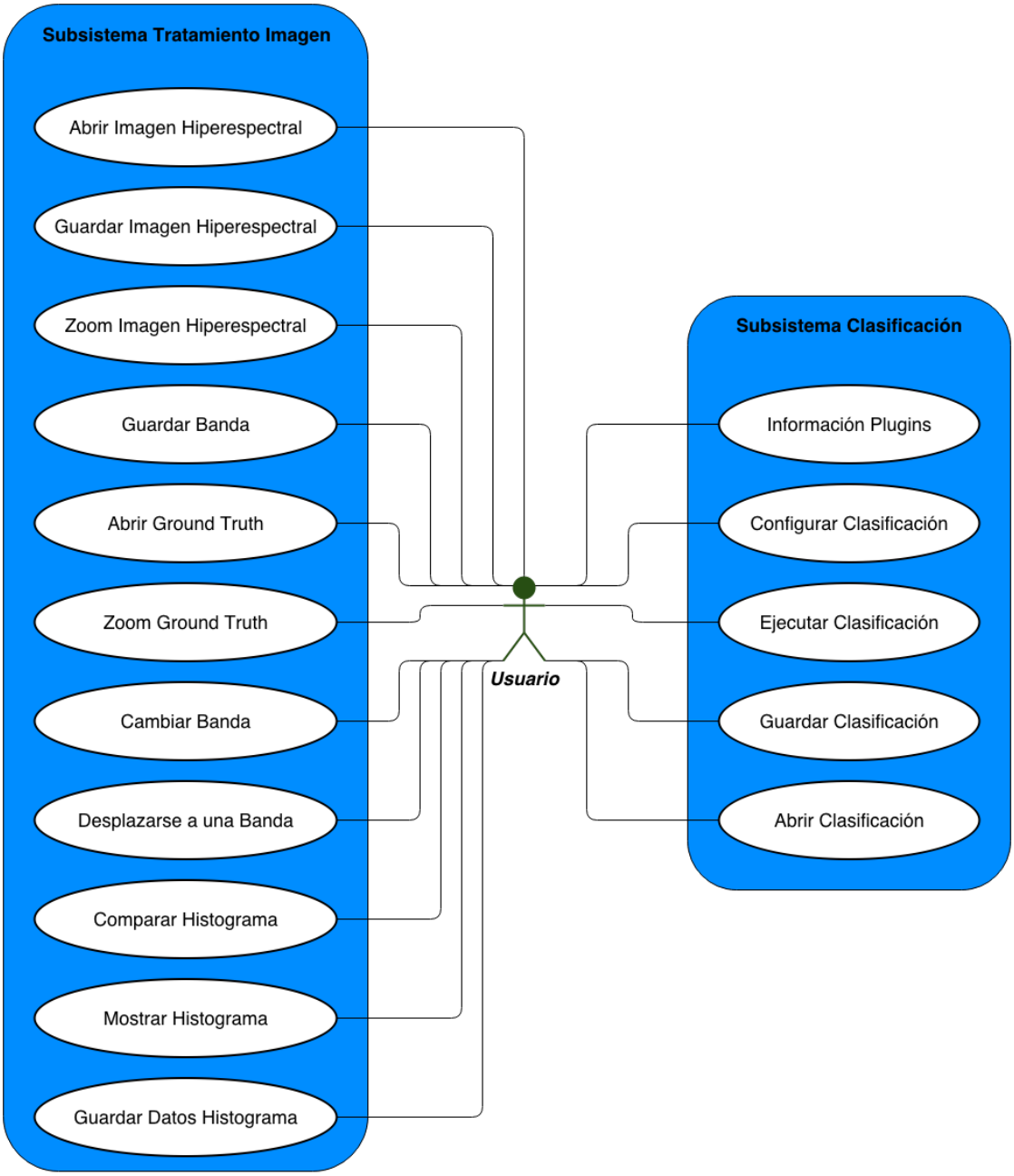

Figura 4.1: Diagrama de casos de uso global

Identificador	CU.01
Nombre	Abrir imagen hiperespectral.
Propósito	Abrir una imagen hiperespectral y mostrar su primera banda por pantalla.
Actores	Usuario.
Precondición	Ninguna.
Postcondición	Se almacenan en memoria los datos de la imagen hiperespectral y se muestra en pantalla la primera banda de dicha imagen hiperespectral.
Escenario Principal	1. El usuario selecciona la opción "Abrir hiperespectral" del menú "Archivo". 2. Se muestra una ventana donde el usuario puede seleccionar el fichero que contiene la imagen. 3. Se pulsa el botón "Aceptar" y se cierra la ventana de selección de fichero. 4. Se muestra la imagen por pantalla.
Escenario Alternativo	1.a El usuario pulsa el botón de abrir imagen hiperespectral situado en la barra de herramientas. 2.a El usuario modifica el tipo de extensión prefijada para poder abrir un fichero con una extensión distinta o que simplemente no tiene extensión. 3.a El fichero está en formato RAW o en un formato desconocido, con lo cual se muestra una ventana donde el usuario debe rellenar un formulario, indicando algunos datos de la imagen. * En cualquier momento el usuario puede cancelar la operación cerrando la ventana o pulsando en el botón cancelar.
Prioridad	Alta
Subsistema	Subsistema Tratamiento Imágenes.

Tabla 4.61: Caso de Uso CU.01: Abrir imagen hiperespectral.

Identificador	CU.02
Nombre	Guardar imagen hiperespectral.
Propósito	Guardar en un fichero una imagen hiperespectral.
Actores	Usuario.
Precondición	Debe haber una imagen hiperespectral abierta en la aplicación.
Postcondición	La imagen queda almacenada en un fichero.
Escenario Principal	1. El usuario selecciona la opción "Guardar hiperespectral" del menú "Archivo". 2. Se abre una ventana donde el usuario escribe el nombre del fichero, selecciona su ubicación y elige el tipo de fichero en el que se guardarán los datos de la imagen. 3. El usuario pulsa el botón "Aceptar", la ventana se cierra y el fichero se almacena en el formato seleccionado por el usuario. 4. Se muestra la imagen por pantalla.
Escenario Alternativo	1.a El usuario pulsa el botón de abrir imagen hiperespectral situado en la barra de herramientas. 2.a El usuario modifica el tipo de extensión prefijada para poder guardar los datos en alguno de los formatos disponibles. 3.a Ya existe un fichero con el mismo nombre. El sistema se lo indica y pregunta si quiere sobrescribirlo. * En cualquier momento el usuario puede cancelar la operación cerrando la ventana o pulsando en el botón cancelar.
Prioridad	Alta
Subsistema	Subsistema Tratamiento Imágenes.

Tabla 4.62: Caso de Uso CU.02: Guardar imagen hiperespectral.

Identificador	CU.03
Nombre	Zoom imagen hiperespectral.
Propósito	Mostrar en una ventana el resultado de realizar un zoom sobre una zona de la imagen hiperespectral. La zona a aumentar está indicada por el puntero del ratón al pasar sobre la imagen hiperespectral.
Actores	Usuario.
Precondición	Debe haber una imagen hiperespectral abierta en la aplicación.
Postcondición	Se creará una ventana donde se muestra el zoom sobre una zona de la imagen hiperespectral.
Escenario Principal	1. El usuario selecciona la opción "Zoom Hiperespectral" del menú "Herramientas". 2. Se muestra la ventana con el zoom sobre la zona superior derecha de la imagen hiperespectral. 3. El usuario se mueve por la imagen hiperespectral para cambiar el foco del zoom.
Escenario Alternativo	1.a El usuario pulsa el botón del zoom situado en la barra de herramientas.
Prioridad	Alta
Subsistema	Subsistema Tratamiento Imágenes.

Tabla 4.63: Caso de Uso CU.03: Zoom imagen hiperespectral.

Identificador	CU.04
Nombre	Guardar banda hiperespectral.
Propósito	Guardar la banda, de la imagen hiperespectral, mostrada en pantalla en un fichero con formato PGM.
Actores	Usuario.
Precondición	Debe haber una imagen hiperespectral abierta en la aplicación.
Postcondición	Se genera un fichero en formato PGM que contiene la banda, de la imagen hiperespectral, mostrada en pantalla.
Escenario Principal	1. Pulsamos en la opción "Guardar banda" del menú "Herramientas". 2. Especificamos el nombre y la ubicación del fichero que contendrá los datos de la banda. 3. Pulsamos el botón de guardar. 4. Se guarda el fichero y se cierre la ventana de selección de fichero.
Escenario Alternativo	1. Pulsamos sobre el botón que ejecuta la acción de guardar los datos del histograma situado en la barra de herramientas. 4.a El fichero ya existe, el programa nos informa del error y nos permite modificar el nombre y la ubicación nuevamente. * En cualquier momento el usuario puede cancelar la operación cerrando la ventana o pulsando en el botón cancelar.
Prioridad	Media
Subsistema	Subsistema Tratamiento Imágenes.

Tabla 4.64: Caso de Uso CU.04: Guardar banda hiperespectral.

Identificador	CU.05
Nombre	Abrir *ground truth*.
Propósito	Abrir el *ground truth* de la imagen hiperespectral.
Actores	Usuario.
Precondición	Debe haber una imagen hiperespectral abierta en la aplicación.
Postcondición	Se almacenan en memoria los datos del *ground truth* de la imagen hiperespectral y se muestran en pantalla en forma de imagen.
Escenario Principal	1. El usuario selecciona la opción "Abrir ground gruth" del menú "Archivo". 2. Se muestra una ventana donde el usuario puede seleccionar el fichero que contiene la imagen. 3. Se pulsa el botón "Aceptar" y se cierra la ventana de selección de fichero. 4. Se muestra la imagen por pantalla.
Escenario Alternativo	1.a El usuario pulsa el botón de abrir *ground truth* situado en la barra de herramientas. 2.a El usuario modifica el tipo de extensión prefijada para poder abrir un fichero con una extensión distinta o que simplemente no tiene extensión. 3.a El fichero está en formato RAW o en un formato desconocido, con lo cual se muestra una ventana donde el usuario debe rellenar un formulario, indicando algunos datos de la imagen. 3.b El fichero contiene una imagen que no concuerda en dimensiones con la imagen hiperespectral abierta en ese momento. Se muestra un mensaje por pantalla avisando del contratiempo y se permite seleccionar otro fichero distinto. * En cualquier momento el usuario puede cancelar la operación cerrando la ventana o pulsando en el botón cancelar.
Prioridad	Alta
Subsistema	Subsistema Tratamiento Imágenes.

Tabla 4.65: Caso de Uso CU.05: Abrir *ground truth*.

Identificador	CU.06
Nombre	Zoom *ground truth*.
Propósito	Mostrar en una ventana el resultado de realizar un zoom sobre una zona del *ground truth* de la imagen hiperespectral. La zona a aumentar está indicada por el puntero del ratón al pasar sobre el*ground truth*.
Actores	Usuario.
Precondición	Debe haber un *ground truth* abierto en la aplicación.
Postcondición	Se creará una ventana donde se muestra el zoom sobre una zona del *ground truth*.
Escenario Principal	1. El usuario selecciona la opción "Zoom ground truth" del menú "Herramientas". 2. Se muestra la ventana con el zoom sobre la zona superior derecha del *ground truth*. 3. El usuario cierra la ventana de zoom.
Escenario Alternativo	1.a El usuario pulsa el botón del zoom situado en la barra de herramientas. 3a. El usuario se mueve por el *ground truth* para cambiar el foco del zoom.
Prioridad	Baja
Subsistema	Subsistema Tratamiento Imágenes.

Tabla 4.66: Caso de Uso CU.06: Zoom *ground truth*.

Identificador	CU.07
Nombre	Cambiar banda.
Propósito	Que el usuario pueda visualizar las distintas bandas que componen la imagen hiperespectral, cambiando de una a otra de forma secuencial usando la rueda del ratón.
Actores	Usuario.
Precondición	Debe haber una imagen hiperespectral abierta en la aplicación.
Postcondición	Se cambia la banda, de la imagen hiperespectral, mostrada en pantalla.
Escenario Principal	1. El usuario gira la rueda del ratón para cambiar la banda.
Escenario Alternativo	Ninguno.
Prioridad	Alta
Subsistema	Subsistema Tratamiento Imágenes.

Tabla 4.67: Caso de Uso CU.07: Cambiar banda.

Identificador	CU.08
Nombre	Desplazarse a una banda.
Propósito	Que el usuario pueda desplazarse directamente a una banda, de la imagen hiperespectral.
Actores	Usuario.
Precondición	Debe haber una imagen hiperespectral abierta en la aplicación.
Postcondición	Se muestra la nueva banda seleccionada.
Escenario Principal	1. Seleccionamos la opción "Ir a banda" del menú Herramientas. 2. Indicamos la banda a la que deseamos cambiar y pulsamos el botón "Aceptar". 3. Se muestra la banda seleccionada. * En cualquier momento el usuario puede cancelar la operación cerrando la ventana o pulsando en el botón cancelar.
Escenario Alternativo	1.a Pulsamos el botón que ejecuta la opción de ir a la banda de la barra de herramientas. 3.a La banda seleccionada no existe, y por lo tanto permanecemos en la misma banda.
Prioridad	Media
Subsistema	Subsistema Tratamiento Imágenes.

Tabla 4.68: Caso de Uso CU.08: Desplazarse a una banda.

Identificador	CU.09
Nombre	Mostrar histograma.
Propósito	Mostrar los valores de un píxel, seleccionado por el usuario, en cada una de las bandas que forman la imagen hiperespectral.
Actores	Usuario.
Precondición	Debe haber una imagen hiperespectral abierta en la aplicación.
Postcondición	Se muestra en una pantalla la información del píxel seleccionado.
Escenario Principal	1. El usuario selecciona la opción "Histograma" del menú "Herramientas". 2. Se muestra la ventana con el histograma del píxel 0, situado en la parte superior izquierda de la imagen. 3. El usuario cierra el histograma.
Escenario Alternativo	1.a El usuario pulsa el icono del histograma situado en la barra de herramientas. 3.a El usuario selecciona otro punto de la imagen hiperespectral.
Prioridad	Media
Subsistema	Subsistema Tratamiento Imágenes.

Tabla 4.69: Caso de Uso CU.09: Mostrar histograma.

Identificador	CU.10
Nombre	Comparar histogramas.
Propósito	Añadir una nueva gráfica a la ventana del histograma para poder comparar, gráficamente, el histograma de dos píxeles. Uno de los píxeles será el que se encuentre seleccionado cuando se active la opción de comparar histogramas, quedando este fijo, mientra que el otro píxel lo decidirá el usuario al seleccionarlo de la imagen hiperespectral.
Actores	Usuario.
Precondición	La ventana del histograma debe estar visible.
Postcondición	En la ventana del histograma podremos comparar gráficamente dos píxeles de la imagen hiperespectral.
Escenario Principal	1. El usuario selecciona el botón de comparación de histogramas. 2. El usuario selecciona un píxel de la imagen hiperespectral.
Escenario Alternativo	2.a El usuario pulsa el botón de comparación de histogramas para cerrar dicha comparación.
Prioridad	Media.
Subsistema	Subsistema Tratamiento Imágenes.

Tabla 4.70: Caso de Uso CU.10: Comparar histogramas.

Identificador	CU.11
Nombre	Guardar datos histograma.
Propósito	Guardar en un fichero los datos que forman el histograma de un píxel.
Actores	Usuario.
Precondición	La ventana del histograma debe estar visible.
Postcondición	Se genera un fichero que contiene los datos del histograma.
Escenario Principal	1. Pulsamos sobre el botón que ejecuta la acción de guardar los datos del histograma situado en la ventana del histograma. 2. Especificamos el nombre y la ubicación del fichero que contendrá los datos del histograma . 3. Pulsamos el botón de guardar. 4. Se guarda el fichero y se cierre la ventana de selección de fichero.
Escenario Alternativo	4.a El fichero ya existe, el programa nos informa del error y nos permite modificar el nombre y la ubicación nuevamente. * En cualquier momento el usuario puede cancelar la operación cerrando la ventana o pulsando en el botón cancelar.
Prioridad	Baja
Subsistema	Subsistema Tratamiento Imágenes.

Tabla 4.71: Caso de Uso CU.11 Guardar datos histograma.

Subsistema Clasificación

Este subsistema es el encargado de la clasificación de la imagen hiperespectral. Su cometido es dar funcionalidad a la configuración de la ejecución de la clasificación manejando los plugins disponibles para la misma, así como generar dichos plugins. Los casos de uso que forman este subsistema se detallan entre la tabla 4.72 hasta la tabla 4.76.

Identificador	**CU.12**
Nombre	Información plugins.
Propósito	Mostrar información referente a los distintos plugins disponibles. La información a mostrar será: nombre, versión, autor y comentarios.
Actores	Usuario.
Precondición	Ninguna.
Postcondición	Se visualiza en pantalla la información de los plugins seleccionados.
Escenario Principal	1. Seleccionamos la opción "Info plugin" del menú "Plugins". 2. Escogemos un plugin de la lista de plugins disponibles y se muestra la información referente al mismo. 3. Cerramos la ventana.
Escenario Alternativo	1.a Seleccionamos la opción de la barra de herramientas. 2.a No hay ningún plugin disponible.
Prioridad	Media.
Subsistema	Subsistema Clasificación.

Tabla 4.72: Caso de Uso CU.12: Información plugins.

Identificador	CU.13
Nombre	Configurar clasificación.
Propósito	Configurar el proceso de clasificación que consta de las siguientes fases: pre-procesado, camino1, camino2, unión y post-procesado. Para llevar a cabo este proceso se configurarán plugins a las distintas fases, según el procesado que deseemos realizar.
Actores	Usuario.
Precondición	Ninguna.
Postcondición	El proceso de configuración queda configurado.
Escenario Principal	1. Seleccionamos la opción "Config. Clasificación" del menú "Clasificación". 2. Seleccionamos las fases a utilizar y en cada una de ellas configuramos los plugins a utilizar. 3. Aceptamos la configuración.
Escenario Alternativo	1.a Seleccionamos la opción de la barra de herramientas.
Prioridad	Alta
Subsistema	Subsistema Clasificación.

Tabla 4.73: Caso de Uso CU.13: Configurar clasificación.

Identificador	CU.14
Nombre	Ejecutar clasificación.
Propósito	Ejecutar los pasos definidos en el proceso de configuración de la clasificación.
Actores	Usuario.
Precondición	Debe haber tanto una imagen hiperespectral como el *ground truth* de dicha imagen, abiertas en la aplicación y se debe haber configurado el proceso de clasificación seleccionando los plugins a ejecutar en las fases adecuadas.
Postcondición	Se muestran en pantalla los resultados de la clasificación.
Escenario Principal	1. Seleccionamos el botón ejecutar de la ventana de configuración. 2. Se muestra el resultado en pantalla.
Escenario Alternativo	1.a Seleccionamos la opción "Clasificar" del menú "Clasificación". 1.b Seleccionamos la opción de la barra de herramientas.
Prioridad	Alta
Subsistema	Subsistema Clasificación.

Tabla 4.74: Caso de Uso CU.14: Ejecutar clasificación.

Identificador	CU.15
Nombre	Guardar clasificación.
Propósito	Almacenar tanto la configuración de cada una de las fases de clasificación, como el resultado final de la clasificación que incluye la eficiencia de la clasificación y la clase asignada a cada píxel de la imagen.
Actores	Usuario.
Precondición	Se ha ejecutado una clasificación.
Postcondición	Se almacena la información de la clasificación en un fichero.
Escenario Principal	1. Al finalizar la ejecución de la clasificación se pulsa el botón de guardar clasificación. 2. Se abre una ventana donde el usuario escribe el nombre del fichero y selecciona su ubicación. 3. El usuario pulsa el botón "Aceptar", la ventana se cierra y el fichero se almacena.
Escenario Alternativo	3.a El fichero ya existe, el sistema muestra un mensaje de aviso y nos permite modificar el nombre del fichero o la ubicación. * En cualquier momento el usuario puede cancelar la operación cerrando la ventana o pulsando en el botón cancelar.
Prioridad	Media
Subsistema	Subsistema Clasificación.

Tabla 4.75: Caso de Uso CU.15: Guardar Clasificación.

Identificador	CU.16
Nombre	Abrir clasificación.
Propósito	Abrir una clasificación previamente almacenada.
Actores	Usuario.
Precondición	Existe, al menos, una clasificación almacenada en fichero.
Postcondición	Se visualizará tanto la configuración de la clasificación como la propia imagen clasificada y los valores de eficiencia de dicha clasificación.
Escenario Principal	1. Pulsamos la opción "Abrir clasificación" del menú Clasificación. 2. Seleccionamos el fichero que contiene la clasificación. 3. Se muestra en pantalla la configuración de la clasificación, los valores de eficiencia y la imagen clasificada. 4. Cerramos la ventana.
Escenario Alternativo	1.a Seleccionamos de la barra de herramientas la opción que permite abrir clasificaciones. 3.a El fichero seleccionado no tiene el formato correcto. Se muestra un mensaje por pantalla. * En cualquier momento el usuario puede cancelar la operación cerrando la ventana o pulsando en el botón cancelar.
Prioridad	Media.
Subsistema	Subsistema Clasificación.

Tabla 4.76: Caso de Uso CU.16: Abrir Clasificación.

4.8. Matriz de Trazabilidad

Tal y como se especifica en el PMBOK [33], la matriz de trazabilidad es una tabla que vincula los requisitos con su origen y los monitoriza a lo largo del ciclo de vida del proyecto. En la tabla 4.77 podemos ver la matriz de trazabilidad del proyecto.

4.9. Enunciado del Alcance del Proyecto

Se incluirá en este apartado aquellos procesos necesarios para garantizar que el proyecto incluya el trabajo requerido para completarlo con éxito.

4.9.1. Definición del Alcance del Proyecto

El proyecto pretende desarrollar una aplicación que tiene como último fin poder clasificar imágenes hiperespectrales utilizando para ello distintas técnicas de clasificación. Con esto también queremos conseguir que la herramienta sirva de apoyo en el desarrollo de nuevas técnicas de clasificación, sirviendo de mesa de pruebas para aquellos usuarios que deseen verificar sus nuevas creaciones.

4.9.2. Criterios de Aceptación del Producto

La aplicación debe ser de utilidad para el usuario, más allá de la clasificación de imágenes. Para eso debe cumplir con los requisitos que se reflejan en el apartado 4.3 del presente documento. Para la aceptación del producto se desarrollarán casos de prueba para los diferentes casos de uso descritos en el apartado 4.7.

Para la aceptación del proyecto, las pruebas se deberán ejecutar sin producir errores.

4.9.3. Entregables del Proyecto

Tal y como se especifica en el artículo 25 del reglamento del Trabajo Fin de Grado[1], se entregarán cuatro copias de esta memoria en papel y cuatro copias en formato digital de la siguiente documentación:

- Memoria completa del Trabajo Fin de Grado.
- Código fuente del producto de software terminado.
- Manual de instalación en formato PDF.

[1] http://www.usc.es/etse/files/u1/RegulamentoTFG_GrEI_CG_30xan2014.pdf

	CU.01	CU.02	CU.03	CU.04	CU.05	CU.06	CU.07	CU.08	CU.09	CU.10	CU.11	CU.12	CU.13	CU.14	CU.15	CU.16
RF.01	•															
RF.02	•															
RF.03	•															
RF.04	•															
RF.05		•														
RF.06		•														
RF.07		•														
RF.08			•													
RF.09			•													
RF.10				•												
RF.11					•											
RF.12					•											
RF.13					•											
RF.14					•											
RF.15						•										
RF.16							•									
RF.17								•								
RF.18									•							
RF.19									•							
RF.20										•						
RF.21											•					
RF.22												•				
RF.23													•			
RF.24													•			
RF.25													•			
RF.26													•			
RF.27														•		
RF.28														•		
RF.29														•		
RF.30														•		
RF.31														•		
RF.32														•		
RF.33														•		
RF.34														•		
RF.35														•		
RF.36															•	
RF.37															•	
RF.38																•

Tabla 4.77: Matriz de trababilidad.

- Manual de usuario en formato PDF.

4.9.4. Exclusiones del Proyecto

No se incluye ningún tipo de requisito adicional a los descritos en el apartado 4.5, ningún tipo de trabajo de mantenimiento de la aplicación una vez entregada, ni procesos de formación o asistencia a los usuarios, entendiendo como suficientes los manuales de instalación y utilización.

4.9.5. Restricciones del Proyecto

La aplicación deberá ser desarrollada bajo la licencia GPL y la documentación como Creative Commons[2].

El desarrollo se realizara para el sistema operativo Linux utilizando la librería GTK+ y empleando el lenguaje de programación C.

4.9.6. Supuestos del Proyecto

Suponemos que los códigos de técnicas que se van a adaptar como plugins en nuestra aplicación, ya se encuentran desarrollados y son completamente funcionales de forma independiente, por lo que solo tendremos que adaptarlos para que se acoplen con nuestra aplicación, suponiendo también que se nos suministrarán todas las librerías que dichos códigos necesiten. Por otra parte, también suponemos que el cliente dispone de imágenes hiperespectrales y sus correspondientes ficheros *ground truth* para poder probar la aplicación.

[2]http://creativecommons.org

Capítulo 5

Diseño e Implementación

En esta sección se detalla el diseño y la implementación de la aplicación.

Siguiendo la filosofía de la metodología *Scrum*, el proceso de diseño se lleva a cabo de abajo hacia arriba, comenzando a nivel de arquitectura y terminando con el diseño detallado.

5.1. Diseño de la Arquitectura

La solución adoptada pretende que el sistema esté dotado de una gran modularidad, permitiendo que la mayoría de los componentes funcionen de forma independiente fuera de la aplicación y sean fácilmente sustituibles.

Todos los procesos que actúan sobre los datos de las imágenes hiperespectrales lo hacen utilizando un módulo que se ha diseñado simulando una clase de Java. Este módulo contiene una estructura que permite albergar todos los datos de una imagen hiperespectral y esta dotado de funciones para el acceso y modificación de dichos datos sin que el usuario del módulo necesite saber la estructura de datos que se está utilizando.

En la figura 5.1 podemos ver el diseño de la arquitectura que se realizó en la primera fase de *Scrum* y como desde el principio quedó clara la división del sistema en dos grandes subsistemas como son el de tratamiento de imágenes y el de clasificación de imágenes..

5.2. Diseño del Sistema

En este punto vamos a representar el sistema utilizando para ello distintas técnicas de modelado. Es importante dejar claro que todos los modelos mostrados describen un único sistema y por eso se han utilizado modelos que nos

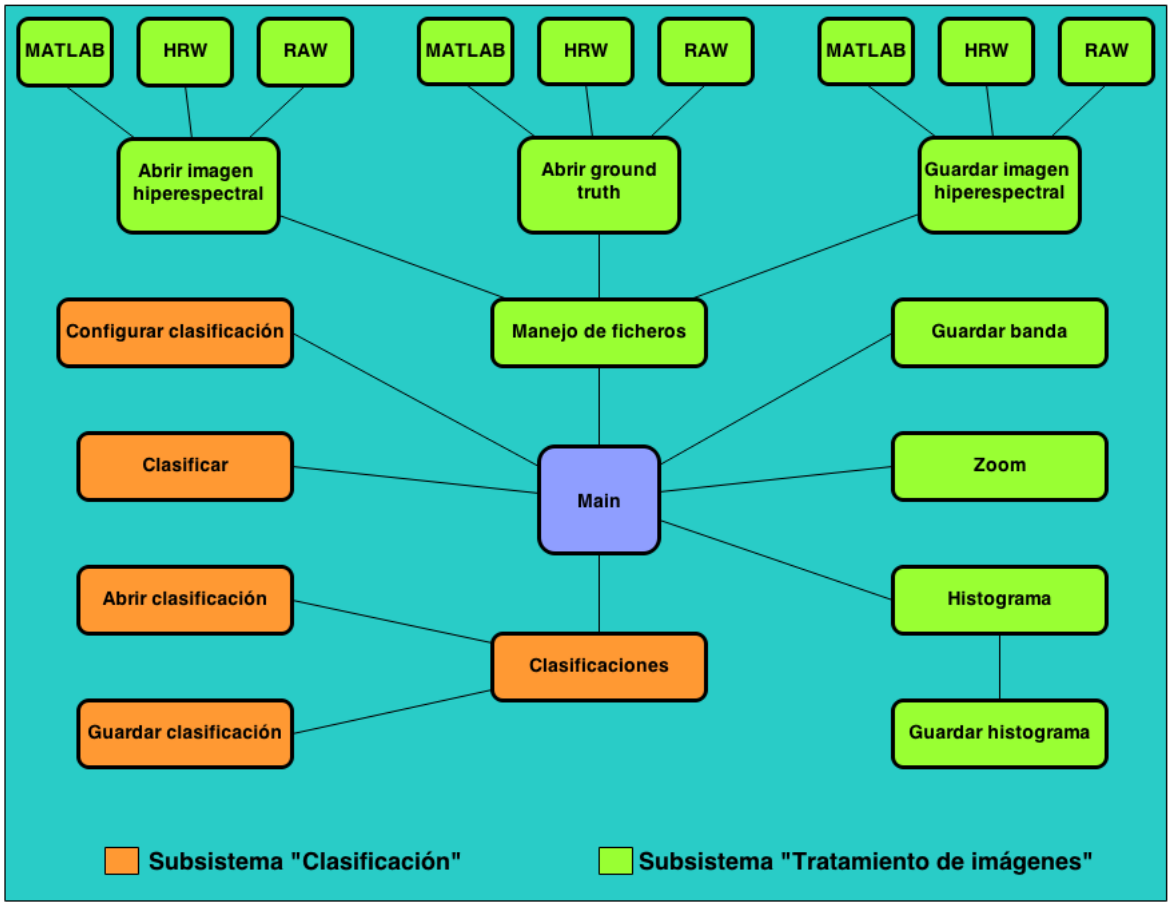

Figura 5.1: Diseño de la arquitectura

permitan relacionar las distintas representaciones.

Debido a las restricciones del proyecto en cuanto al lenguaje a utilizar en el desarrollo de la aplicación, no ha sido posible un enfoque orientado a objetos, lo que nos hubiese permitido incluir en esta sección elementos comunes a ese tipo de desarrollo como pueden ser diagramas de clase, diagramas de secuencia, diagramas de paquetes, etc. En su lugar se optó por la utilización de diagramas de flujo de datos y diagramas de flujo control, que nos van a permitir describir el sistema desde el punto de vista del proceso y su comportamiento.

5.2.1. Diccionario de Datos

El diccionario de datos contiene las características de los datos que se van a utilizar en el sistema que estamos desarrollando, permitiendo una mejor comprensión de los modelos que se emplean para representar ese sistema. Este diccionario de datos se complementa con el glosario de la sección 4.1.

Explosión de proceso: Consiste en describir con más detalle un proceso, dando lugar a un nuevo diagrama de flujo de datos.

Estructura general: Es una estructura de datos que se utiliza para almacenar tanto los datos de la imagen como todas sus características. Existirá una estructura general para la imagen hiperespectral y otra para su *ground truth*.

Librería MATIO: Librería para la lectura y escritura de ficheros MATLAB.

Row-Major order: Indica el orden que se debe seguir a la hora de leer/escribir una matriz, que en este caso va leyendo primero los datos de la 1ª fila, luego de la 2ª, ...

5.2.2. Diagramas de Flujo de Datos

Los diagramas de flujo de datos definen el sistema desde el punto de vista del proceso, describiéndolo como un conjunto de operaciones de proceso de información. Estas operaciones reciben unos flujos de datos de entrada y los transforman en flujos de datos de salida. Para describir el sistema desde este punto de vista utilizaremos tanto diagramas de flujo de datos como especificaciones de procesos.

Diagrama de Contexto

En la figura 5.2 podemos ver el diagrama de contexto, que representa el sistema al más alto nivel de abstracción. Incluye un único proceso que identifica cuál es la función principal del sistema.

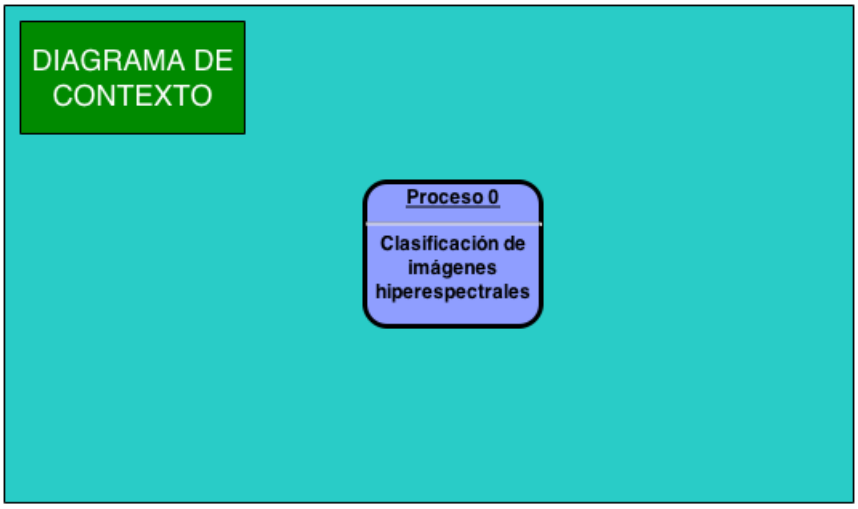

Figura 5.2: Diagrama de contexto.

DFD 0 - Clasificación de Imágenes Hiperespectrales

En la figura 5.3 podemos ver el resultado de la explosión del proceso "Clasificación de imágenes hiperespectrales" incluido en el diagrama de contexto de la figura 5.2.

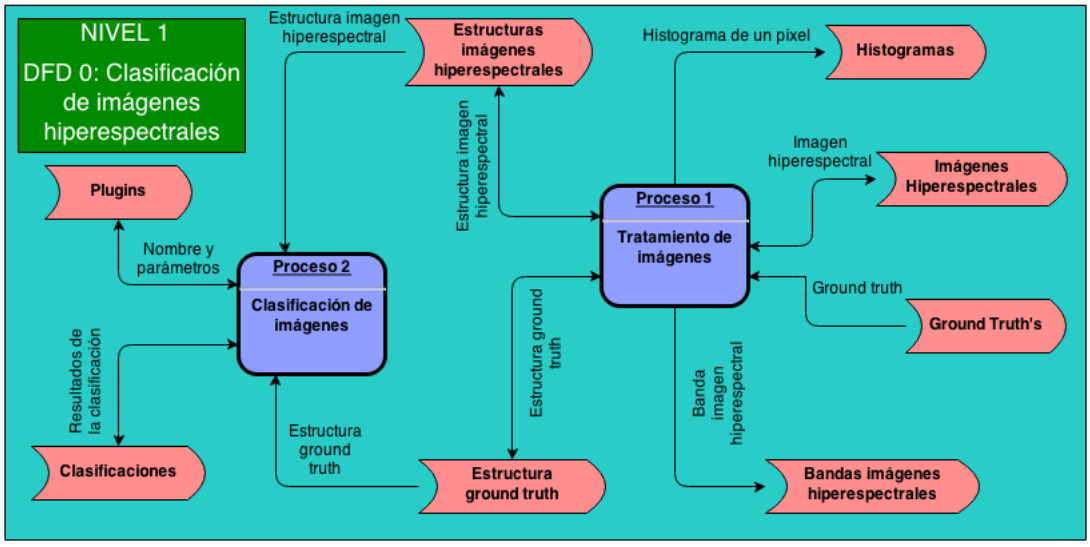

Figura 5.3: DFD 0. Clasificación de imágenes hiperespectrales

DFD 1 - Tratamiento de Imágenes

En la figura 5.4 podemos ver el resultado de la explosión del proceso "Tratamiento de imágenes" incluido en el diagrama de flujo de datos 0 de la figura 5.3.

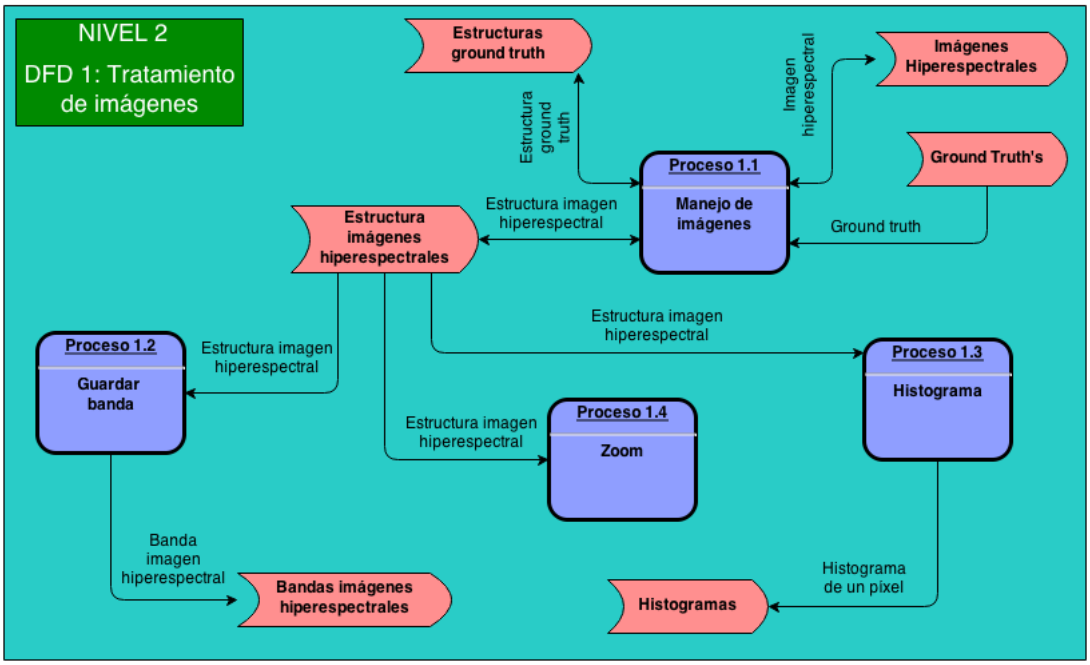

Figura 5.4: DFD 1. Tratamiento de imágenes

Especificación de procesos:

- Proceso 1.2: Guardar banda.
  - Precondición: Debe existir una imagen hiperespectral abierta en el programa.
  - Postcondición: Se genera un fichero que contiene una banda de la imagen hiperespectral.
  - Descripción: Este proceso muestra una ventana de diálogo que permite al usuario indicar el nombre y la ubicación del fichero que contendrá la banda. Si el usuario introduce los datos y acepta la ventana, el proceso accede a los datos de la banda que se encuentran almacenados en la estructura general y los almacena en el fichero con el nombre y en la ubicación especificada.
- Proceso 1.4: Zoom.
  - Precondición: Debe existir una imagen hiperespectral abierta en el programa.
  - Postcondición: Se crea una nueva ventana donde se visualiza el zoom de la imagen hiperespectral.

  - Descripción: Este proceso verifica que no existe ninguna ventana de zoom previa, en cuyo caso la destruye. Posteriormente construye la ventana que contendrá el zoom de la imagen hiperespectral, obtiene las coordenadas de la imagen donde debemos realizar el zoom de la estructura general, y a continuación accede a los datos de la misma para extraer tanto el píxel que va a ampliar como sus vecinos, hasta una distancia de 10 vecinos, es decir, obtiene un cubo de datos de 10 x 10. Por último muestra los datos en la pantalla y activa la señal que permite actualizar el zoom al mover el ratón por encima de la imagen hiperespectral.

DFD 1.1 - Manejo de Imágenes

En la figura 5.5 podemos ver el resultado de la explosión del proceso "Manejo de imágenes" incluido en el diagrama de flujo de datos 1 de la figura 5.4.

Especificación de procesos:

- Proceso 1.1.2: Mostrar imagen hiperespectral.
  - Precondición: Los datos de una imagen hiperespectral deben estar cargado en la estructura general.
  - Postcondición: Se muestra por pantalla la primera banda de la imagen.
  - Descripción: El proceso crea la ventana que contendrá la primera banda de la imagen hiperespectral. A continuación lee los datos de la banda a mostrar de la estructura general y los visualiza por pantalla.
- Proceso 1.1.4: Mostrar *ground truth*.
  - Precondición: Los datos del *ground truth* de una imagen hiperespectral deben estar cargados en su estructura general.
  - Postcondición: Se muestra por pantalla el *ground truth*.
  - Descripción: El proceso crea la ventana que contendrá el *ground truth*. A continuación lee los desde la estructura general y los muestra en pantalla.

DFD 1.1.1 - Abrir Imagen Hiperespectral

En la figura 5.6 podemos ver el resultado de la explosión del proceso "Abrir imagen hiperespectral" incluido en el diagrama de flujo de datos 1.1 de la figura 5.5.

Especificación de procesos:

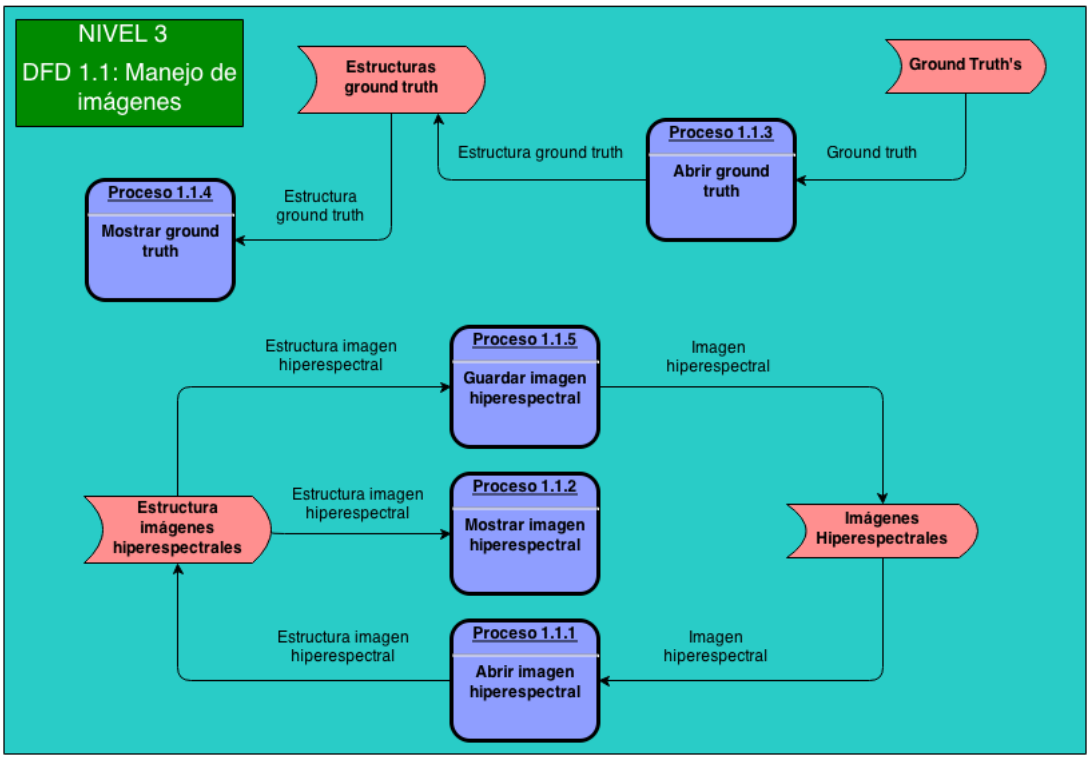

Figura 5.5: DFD 1.1. Manejo de imágenes

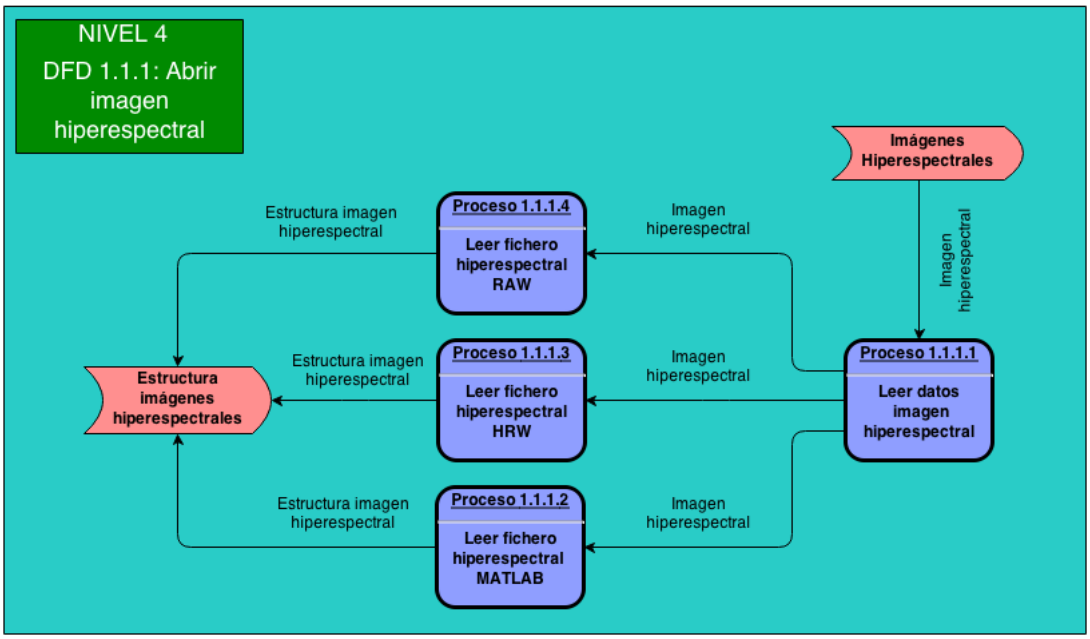

Figura 5.6: DFD 1.1.1. Abrir imagen hiperespectral

- Proceso 1.1.1.1: Leer datos imagen hiperespectral.
  - Precondición: Ninguna.
  - Postcondición: Se conoce el nombre y la ubicación del fichero que contiene la imagen hiperespectral que queremos abrir.
  - Descripción: En primer lugar el proceso muestra la ventana de diálogo al usuario para que este seleccione el nombre y la ubicación del fichero que contiene la imagen hiperespectral. Después, se encarga de llamar a la función correspondiente al tipo de fichero que se ha seleccionado para que lo abra.
- Proceso 1.1.1.2: Leer fichero hiperespectral MATLAB.
  - Precondición: Se ha seleccionado un fichero MATLAB para ser abierto.
  - Postcondición: Se carga en la estructura general los datos de la imagen hiperespectral.
  - Descripción: Este proceso utiliza funciones de la librería MATIO [1] para realizar la apertura del los ficheros MATLAB. En primer lugar lee el fichero y posteriormente rellena la estructura general con los datos correspondiente a la imagen leída. Entre los datos que almacena se encuentran las dimensiones y la primera banda de la imagen.
- Proceso 1.1.1.3: Leer fichero hiperespectral HRW.
  - Precondición: El usuario ha indicado un fichero HRW para su apertura.
  - Postcondición: Se carga en la estructura general los datos de la imagen hiperespectral.
  - Descripción: Este proceso se encarga de interpretar los fichero con formato HRW. En primer lugar lee la cabecera de los mismos para extraer los datos referentes al tamaño de la imagen, para posteriormente leer los datos de la imagen, descomprimir esos datos y rellenar la estructura general con los valores leídos, incluyendo las dimensiones de la imagen y la primera banda de la misma.
- Proceso 1.1.1.4: Leer fichero hiperespectral RAW.
  - Precondición: Se ha seleccionado un fichero RAW para ser abierto.
  - Postcondición: Se carga en la estructura general los datos de la imagen hiperespectral.

[1] http://sourceforge.net/projects/matio/

– Descripción: Este proceso, antes de interpretar el fichero, necesita que se le faciliten las características de la imagen, es decir, dimensiones, tipo de datos, tamaño de cabecera, etc. Para ese cometido crea una ventana con una serie de campos que el usuario debe rellenar. Una vez rellenado los datos de forma correcta, el proceso continua con la interpretación del fichero, para posteriormente proceder a almacenar los datos leídos en la estructura general.

DFD 1.1.3 - Abrir Ground Truth

En la figura 5.7 podemos ver el resultado de la explosión del proceso "Abrir ground truth" incluido en el diagrama de flujo de datos 1.1 de la figura 5.5.

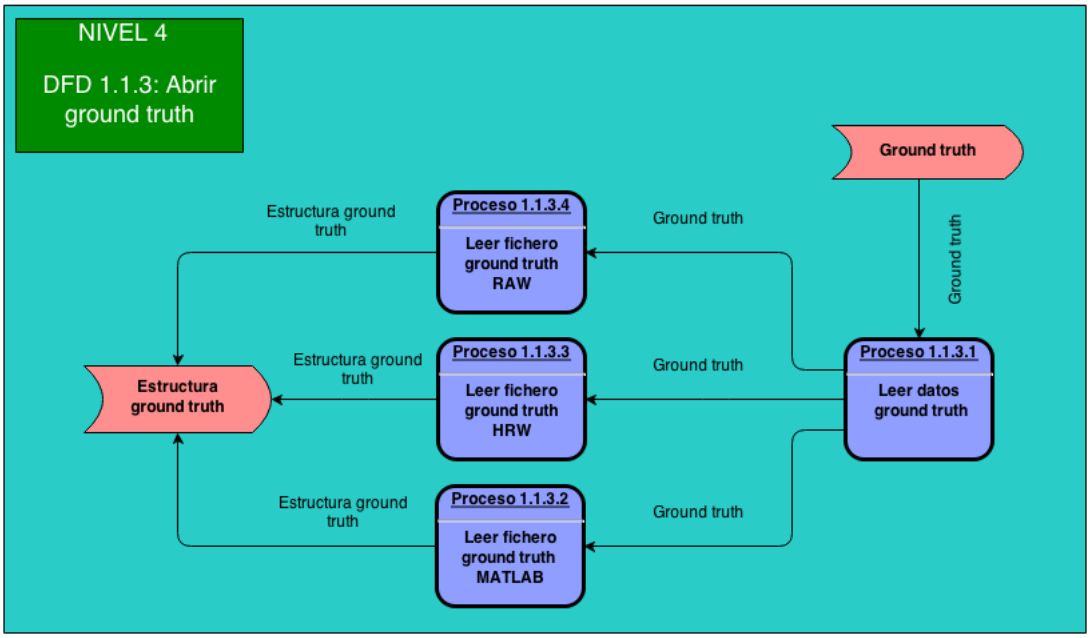

Figura 5.7: DFD 1.1.3. Abrir *ground truth*

Especificación de procesos:

- Proceso 1.1.3.1: Leer datos *ground truth*.
  - Precondición: Debe existir una imagen hiperespectral abierta en la aplicación.
  - Postcondición: Se conoce el nombre y la ubicación del fichero que contiene el *ground truth* que queremos abrir.
  - Descripción: El proceso muestra la ventana de diálogo al usuario para que este seleccione el nombre y la ubicación del fichero que contiene

el *ground truth*. Después, se encarga de llamar a la función correspondiente al tipo de fichero que se ha seleccionado para que lo abra.

- Proceso 1.1.3.2: Leer fichero *ground truth* MATLAB.
  - Precondición: Se ha seleccionado un fichero MATLAB para ser abierto.
  - Postcondición: Se carga en la estructura general los datos del *ground truth*.
  - Descripción: Este proceso utiliza funciones de la librería "MATIO" para realizar la apertura del los ficheros MATLAB. En primer lugar lee el fichero y posteriormente rellena la estructura general con los datos correspondiente a la imagen abierta.
- Proceso 1.1.3.3: Leer fichero *ground truth* HRW.
  - Precondición: El usuario ha indicado un fichero HRW para su apertura.
  - Postcondición: Se carga en la estructura general los datos del *ground truth*.
  - Descripción: Este proceso se encarga de interpretar los fichero con formato HRW. En primer lugar lee la cabecera de los mismos para extraer los datos referentes al tamaño de la imagen, para posteriormente leer los datos de la imagen y rellenar la estructura general con los mismos, incluyendo las dimensiones de la imagen.
- Proceso 1.1.3.4: Leer fichero *ground truth* RAW.
  - Precondición: Se ha seleccionado un fichero RAW para ser abierto.
  - Postcondición: Se carga en la estructura general los datos del *ground truth*.
  - Descripción: Este proceso, antes de interpretar el fichero, necesita que se le faciliten las características de la imagen, es decir, tamaño de cabecera, tipo de datos, formato, etc. Para ese cometido crea una ventana con una serie de campos que el usuario debe rellenar. Una vez rellenado los datos de forma correcta, el proceso continua con la interpretación del fichero, para posteriormente proceder a almacenar los datos leídos en la estructura general.

DFD 1.1.5 - Guardar Imagen Hiperespectral

En la figura 5.8 podemos ver el resultado de la explosión del proceso "Guardar imagen hiperespectral" incluido en el diagrama de flujo de datos 1.1 de la figura 5.5.

Especificación de procesos:

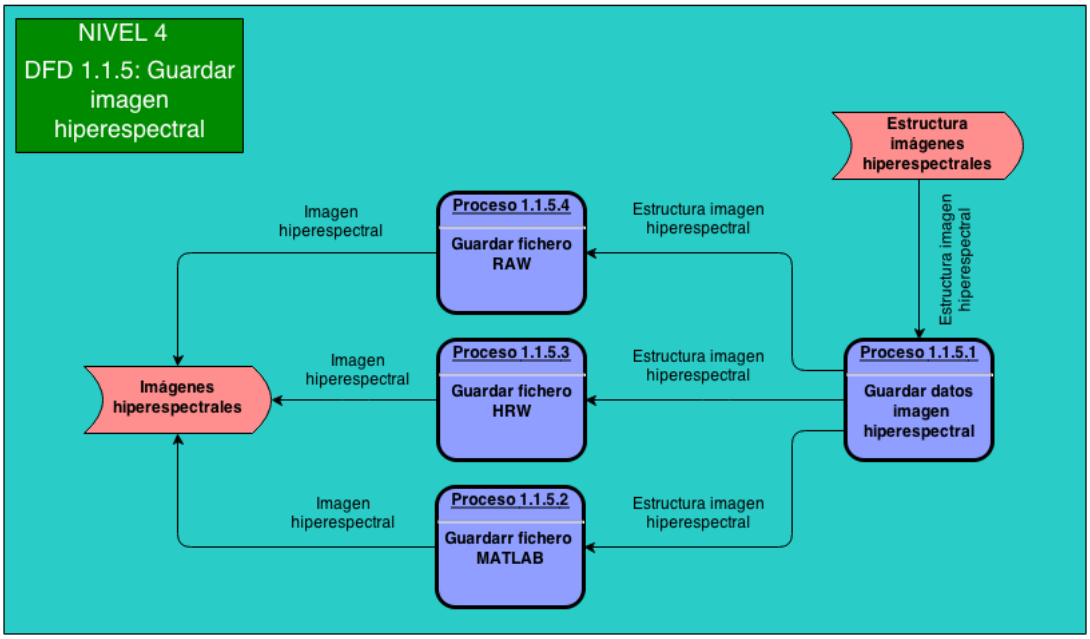

Figura 5.8: DFD 1.1.5. Guardar imagen hiperespectral

- Proceso 1.1.5.1: Guardar datos imagen hiperespectral.
  - Precondición: Debe haber una imagen hiperespectral abierta en el programa.
  - Postcondición: Se conoce el nombre y la ubicación del fichero que contendrá la imagen a almacenar.
  - Descripción: El proceso muestra una ventana de diálogo en la que el usuario indica la ubicación y el nombre del fichero que contendrá la imagen. Posteriormente, en función del filtro que el usuario seleccionó, se invoca a la función correspondiente para que almacene los datos.
- Proceso 1.1.5.2: Guardar fichero MATLAB.
  - Precondición: Debe haber una imagen hiperespectral abierta en el programa.
  - Postcondición: Se almacena la imagen hiperespectral en un fichero con formato MATLAB.
  - Descripción: Este proceso utiliza funciones de la librería "MATIO" para almacenar los datos en ficheros con formato MATLAB. En primer lugar se organizan los datos de la imagen en "row-mayor order", que es el orden que usa MATLAB para almacenar matrices. Después, se guardan las dimensiones de la imagen y finalmente se almacenan los propios datos de la imagen.

- Proceso 1.1.5.3: Guardar fichero HRW.
  - Precondición: Debe haber una imagen hiperespectral abierta en el programa.
  - Postcondición: Se almacena la imagen hiperespectral en un fichero con formato HRW.
  - Descripción: El proceso crea un nuevo fichero al que incluye como cabecera, las dimensiones de la imagen y el formato de la imagen para posteriormente comprimir los datos de la imagen hiperespectral y finalmente almacenarlos en el fichero de salida.
- Proceso 1.1.5.4: Guardar fichero RAW.
  - Precondición: Debe haber una imagen hiperespectral abierta en el programa.
  - Postcondición: Se almacena la imagen hiperespectral en un fichero con formato RAW.
  - Descripción: El proceso crea un nuevo fichero al que incluye como cabecera, las dimensiones de la imagen, para a continuación añadir los datos de la imagen hiperespectral al fichero.

DFD 1.3 - Histograma

En la figura 5.9 podemos ver el resultado de la explosión del proceso "Histograma" incluido en el diagrama de flujo de datos 1 de la figura 5.4.

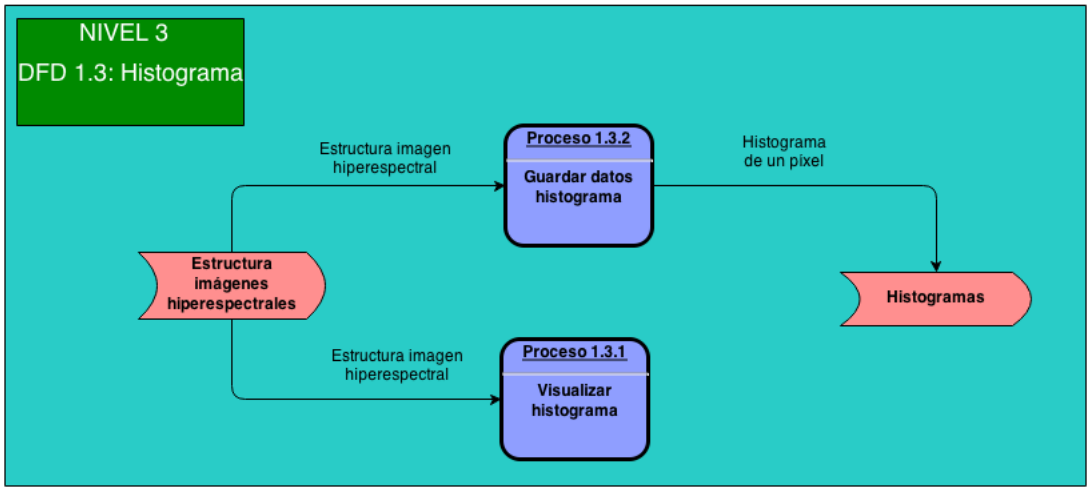

Figura 5.9: DFD 1.3. Histograma

Especificación de procesos:

- Proceso 1.3.1: Visualizar histograma.
  - Precondición: Existe una imagen hiperespectral cargada en la aplicación.
  - Postcondición: Se visualiza la firma espectral de un píxel de la imagen hiperespectral.
  - Descripción: Este proceso verifica que no existe ninguna ventana de histograma previa, en cuyo caso la destruye. Después construye la ventana que contendrá la firma espectral de los píxeles de la imagen hiperespectral, y a continuación accede a los datos de la misma para proceder a dibujar dicha firma por pantalla.
- Proceso 1.3.2: Guardar datos histograma.
  - Precondición: La ventana del histograma está visible.
  - Postcondición: Se almacenan en un fichero los datos que forman la firma espectral del píxel seleccionado.
  - Descripción: El proceso abre un ventana de dialogo en la cual el usuario proporciona un nombre de fichero y una ubicación donde almacenarlo. A continuación, guarda los valores utilizados para construir la firma espectral en el fichero indicado por el usuario.

DFD 2 - Clasificación de Imágenes

En la figura 5.10 podemos ver el resultado de la explosión del proceso "Clasificación de imágenes" incluido en el diagrama de flujo de datos 0 de la figura 5.3.

Especificación de procesos:

- Proceso 2.2: Abrir clasificación.
  - Precondición: Ninguna.
  - Postcondición: Se muestran en pantalla la configuración de la clasificación, los resultados de precisión y la imagen clasificada.
  - Descripción: Mediante una ventana de diálogo el usuario indica la ubicación y el nombre del fichero que contiene la clasificación. Después se procede a interpretar los datos del mismo, comenzando por los plugins que se utilizaron para la clasificación, continuando con los resultados de precisión y la imagen clasificada. En una ventana mostramos tanto los plugins utilizados como los resultados de precisión y la propia imagen clasificada.
- Proceso 2.3: Guardar clasificación.

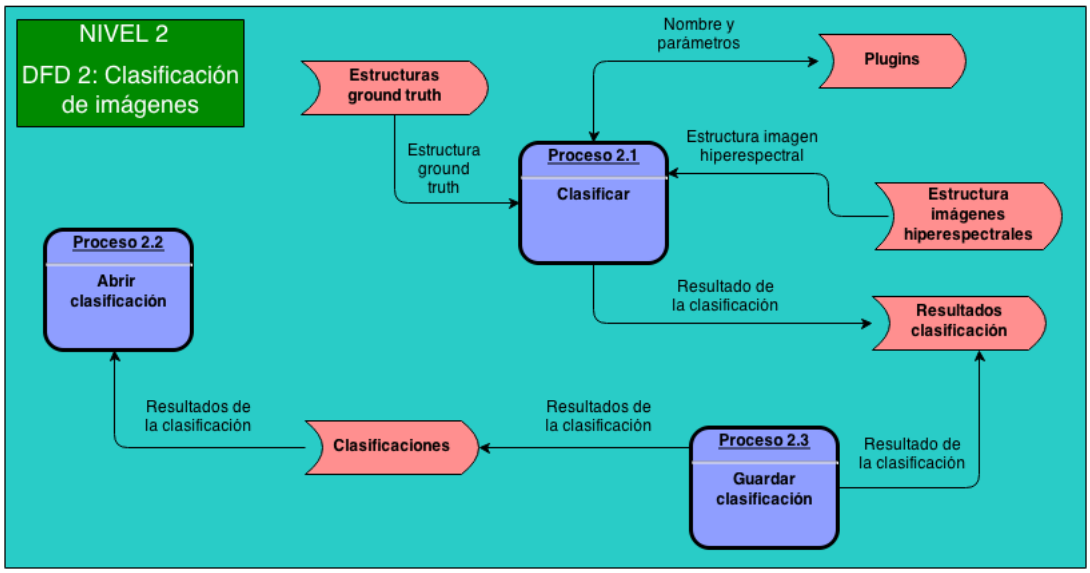

Figura 5.10: DFD 2. Clasificación de imágenes

- Precondición: Se ha realizado una clasificación previa.
- Postcondición: Se almacenan los resultados de una clasificación en un fichero.
- Descripción: En primer lugar este proceso solicita al usuario un nombre de fichero y su ubicación mediante una ventana de dialogo. En segundo lugar se recorre el array que contiene la configuración de los plugins que intervienen en el proceso de clasificación y almacena esos valores en el fichero de salida. Para finalizar, se almacenan tanto los datos de precisión como los datos de la imagen.

DFD 2.1 - Clasificar

En la figura 5.11 podemos ver el resultado de la explosión del proceso "Clasificar" incluido en el diagrama de flujo de datos 2 de la figura 5.10.

- Proceso 2.1.1: Configurar clasificación.
  - Precondición: Ninguna.
  - Postcondición: Se configuran los pasos que se darán en el proceso de clasificación.
  - Descripción: Este proceso crea una nueva ventana con cinco pestañas, una por cada una de las fases que se pueden configurar en el proceso de clasificación. Para cada una de esas pestañas se insertan dos lista, una

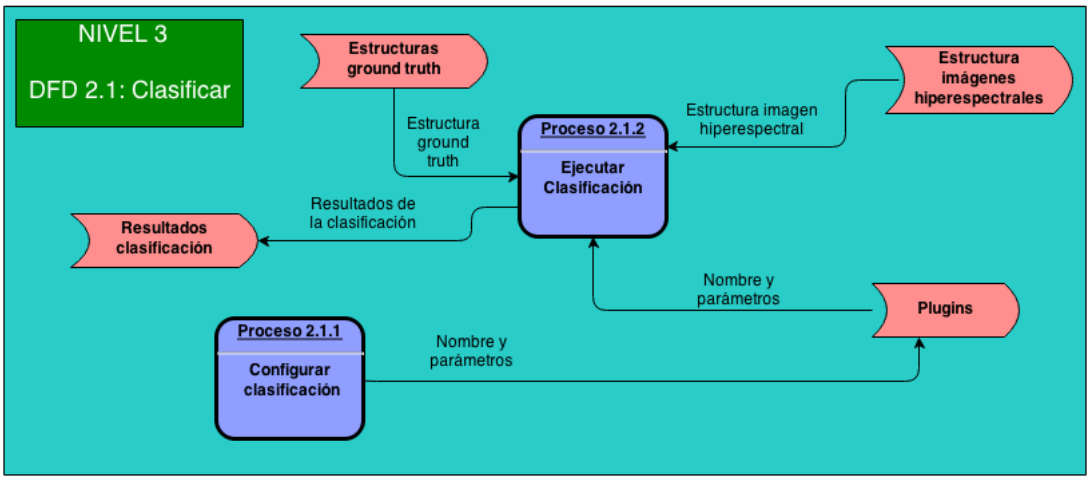

Figura 5.11: DFD 2.1. Clasificar

contendrá los plugins disponibles y la otra los plugins seleccionados. También se incluyen botones para pasar de la lista de disponibles a la lista de seleccionados y otros botones para organizar esta última lista. Todos estos cambios se graban en un array que se encarga de almacenar la configuración de los plugins seleccionados.

- Proceso 2.1.2: Ejecutar clasificación.
  - Precondición: Existen tanto la imagen hiperespectral como su *ground truth* y se han configurado los plugins que intervienen en el proceso de clasificación.
  - Postcondición: Se muestra el resultado de la clasificación y los datos de precisión.
  - Descripción: Este proceso lee el array que contiene los plugins que intervienen en el proceso de clasificación y los ejecuta uno tras otro hasta llegar al último. Después calcula los datos de precisión y los muestra por pantalla junto con la imagen clasificada y la lista de plugins que han intervenido en el proceso.

5.2.3. Diagramas de Flujo de Control

Los diagramas de flujo de control definen el sistema desde el punto de vista del comportamiento, describiéndolo como una sucesión de estados o modos de funcionamiento. Debe indicarse también cuales son las condiciones o eventos que hacen que el sistema pase de un modo a otro. Para describir el sistema desde este punto de vista utilizaremos tanto diagramas de flujo de control como especificaciones de control. Únicamente se representarán diagramas de flujo de control para

aquellos diagramas de flujo de datos en los que alguno de sus procesos necesite una señal o evento para su funcionamiento.

DFC 1.1 - Manejo de Imágenes

En la figura 5.12 podemos ver el diagrama de flujo de control 1.1 correspondiente al diagrama de flujo de datos 1.1 de la figura 5.5.

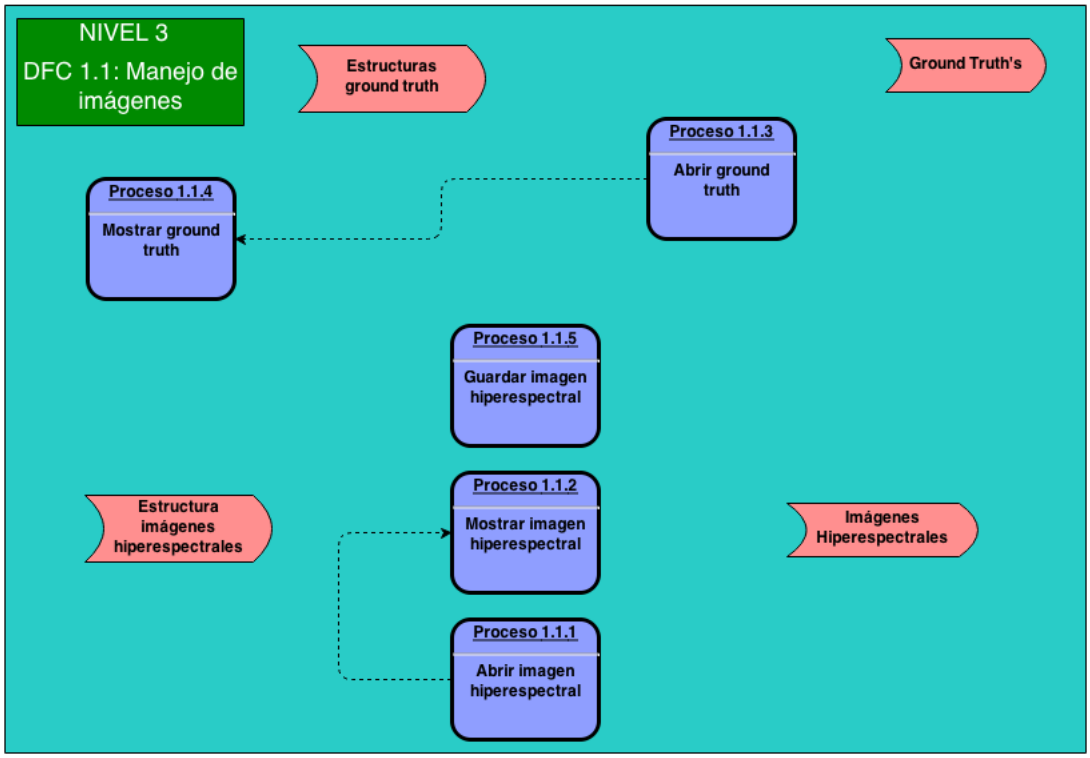

Figura 5.12: DFC 1.1. Manejo de imágenes

Especificación de control:

- Proceso 1.1.1: Abrir imagen hiperespectral.
  - Descripción: Al finalizar la apertura de la imagen hiperespectral, se activa el procedimiento 1.1.4 que visualiza por pantalla dicha imagen.
- Proceso 1.1.3: Abrir *ground truth*.
  - Descripción: Una vez que se ha terminado de cargar los datos del *ground truth*, se llama al procedimiento 1.1.4, encargado de mostrar dichos datos por pantalla.

DFC 1.1.1 - Abrir Imagen Hiperespectral

En la figura 5.13 podemos ver el diagrama de flujo de control 1.1.1 correspondiente al diagrama de flujo de datos 1.1.1 de la figura 5.6.

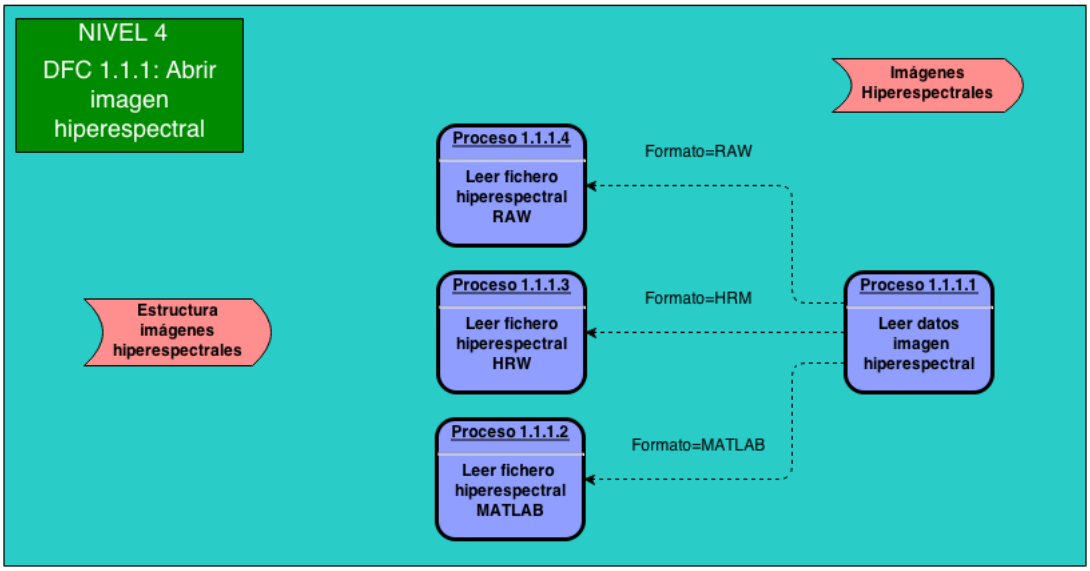

Figura 5.13: DFC 1.1.1. Abrir imagen hiperespectral

Especificación de control:

- Proceso 1.1.1.1: Leer datos imagen hiperespectral.
  - Descripción: Tras la obtención del nombre y la ubicación del fichero, este proceso llama a los procesos 1.1.1.2, 1.1.1.3 y 1.1.1.4, en ese orden, hasta que alguno de ellos responde de forma afirmativa, lo que indicará que ese proceso reconoce el fichero y va a intentar abrirlo. Es entonces cuando se pasa el control al proceso que respondió afirmativamente.

DFC 1.1.3 - Abrir Ground Truth

En la figura 5.14 podemos ver el diagrama de flujo de control 1.1.3 correspondiente al diagrama de flujo de datos 1.1.3 de la figura 5.7.

Especificación de control:

- Proceso 1.1.3.1: Leer datos *ground truth*.

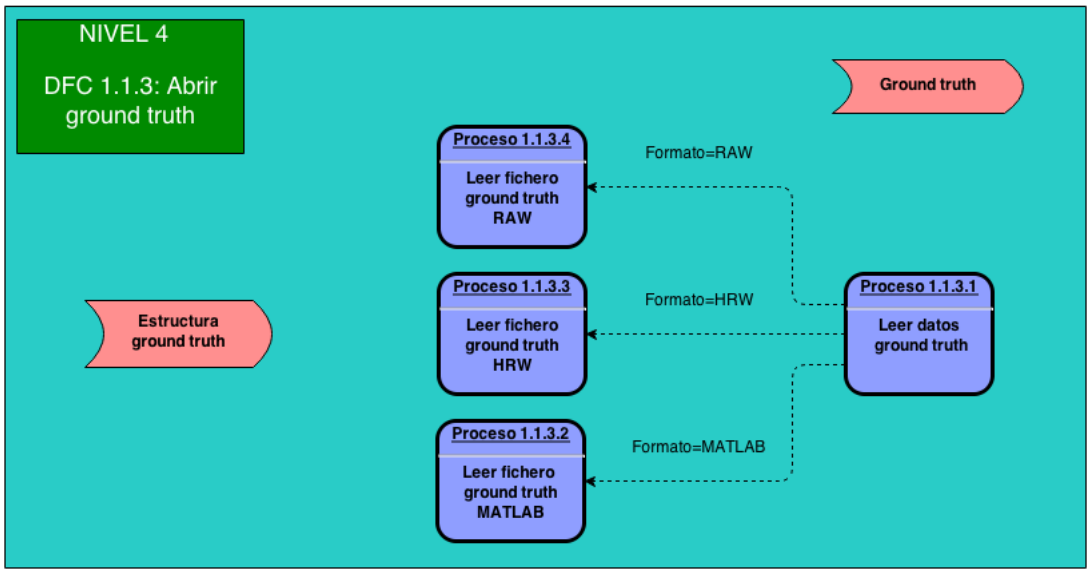

Figura 5.14: DFC 1.1.3. Abrir *ground truth*

– Descripción: Después de solicitar el nombre y la ubicación del fichero al usuario, este proceso llama a los procesos 1.1.3.2, 1.1.3.3 y 1.1.3.4, en ese orden, hasta que alguno de ellos responde de forma afirmativa, lo que indicará que ese proceso reconoce el fichero y va a intentar abrirlo. Es entonces cuando se pasa el control al proceso que respondió afirmativamente.

DFC 1.1.5 - Guardar Imagen Hiperespectral

En la figura 5.15 podemos ver el diagrama de flujo de control 1.1.5 correspondiente al diagrama de flujo de datos 1.1.5 de la figura 5.8.

Especificación de control:

- Proceso 1.1.5.1: Guardar datos imagen hiperespectral.

  – Descripción: En función del tipo de fichero que el usuario seleccionó, se llama al procedimiento 1.1.5.2 (fichero MATLAB), 1.1.5.3 (fichero HRW) o 1.1.5.4 (fichero RAW).

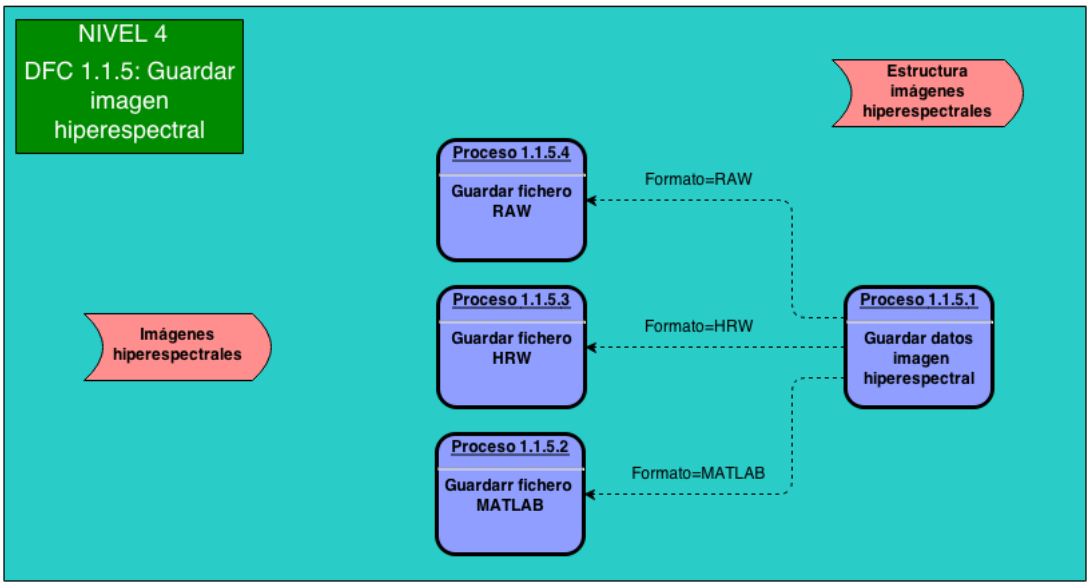

Figura 5.15: DFC 1.1.5. Guardar imagen hiperespectral

Capítulo 6

Validación y Pruebas

El presente proyecto consta de 5 ciclos *sprint* en los que se desarrollaron los dos subsistemas en los que se ha dividido el proyecto. La metodología *Scrum* que hemos usado para este proyecto requiere que se realicen pruebas del software implementado en cada ciclo de *sprint*, validando los requisitos desarrollados en dicho ciclo.

6.1. Validación de Requisitos

En este apartado se describen cuales fueron las pruebas realizadas para cada uno de los ciclos *sprint* de los que se compone el proyecto. Se enumeran los casos de uso implementados de forma total o parcial en cada ciclo y los requisitos que se deben probar. Aclarar que un caso de uso puede ser parcialmente implementado si alguno de los requisitos que lo componen se desarrolla en un ciclo posterior.

6.1.1. Primer Ciclo Sprint

Según la planificación del primer ciclo de *sprint*, estos son los requisitos funcionales que se han implementado:

- Abrir fichero de datos en formato MATLAB (RF.01 tabla 4.13).
- Mostrar imagen hiperespectral (RF.04 4.16).
- Guardar fichero de datos en formato MATLAB (RF.05 4.17).
- Guardar banda (RF.10 4.22).
- Abrir fichero *ground truth* en formato MATLAB (RF.11 4.23).
- Mostrar *ground truth* (RF.14 4.26).
- Cambiar banda (RF.16 4.28).

- Ir a banda (RF.17 4.29).

Según la matriz de trazabilidad de la tabla 4.77, en este ciclo se implementaron de forma parcial o total los siguientes casos de uso:

- Abrir imagen hiperespectral (CU.01 tabla 4.61).
- Guardar imagen hiperespectral (CU.02 tabla 4.62).
- Guardar banda hiperespectral (CU.04 tabla 4.64).
- Abrir *ground truth* (CU.05 tabla 4.65).
- Cambiar banda (CU.07 tabla 4.67).
- Desplazarse a una banda (CU.08 tabla 4.68).

RF.01 Abrir fichero de datos en formato MATLAB.

Estado: Cumplido.

Prueba de funcionamiento: Desde la pantalla principal accedemos a la opción de "Abrir hiperespectral" del menú "Archivo" y seleccionamos un fichero en formato MATLAB[1]. Una vez abierto, imprimimos los valores de algunos píxeles de la primera banda seleccionados aleatoriamente y los cotejamos con los correspondientes valores que obtenemos si abrimos el fichero de datos con el programa MATLAB. Para comprobar que el resto de bandas también se leen, en el código se modifica la banda que debe abrirse por defecto sustituyéndola por la última y por una banda intermedia, repitiendo las pruebas que se hicieron para la primera banda.

RF.04 Mostrar imagen hiperespectral.

Estado: Cumplido.

Prueba de funcionamiento: Desde la barra de herramientas seleccionamos el icono que nos permite abrir una imagen hiperespectral y posteriormente seleccionamos un fichero en formato MATLAB, ya que por ahora es el único que podemos interpretar, para visualizar por pantalla la primera banda de la imagen y compararla con la imagen suministrada por el equipo de investigación. Para comprobar que el resto de bandas también se leen, en el código se modifica la banda que debe abrirse por defecto sustituyéndola por la última y por una banda intermedia, comparándolas también con las suministradas por el equipo de investigación.

[1]http://www.mathworks.es/products/matlab/

RF.05 Guardar fichero de datos en formato MATLAB.

Estado: Cumplido.

Prueba de funcionamiento: Una vez abierta una imagen hiperespectral en formato MATLAB, seleccionamos la opción de "Guardar hiperespectral" del menú "Archivo" y guardamos la imagen seleccionando el formato MATLAB. Posteriormente utilizaremos el programa cmp[2] desde la línea de comandos para comparar el fichero original de la imagen hiperespectral y el fichero almacenado por nuestro programa.

RF.10 Guardar banda.

Estado: Cumplido.

Prueba de funcionamiento: Después de abrir una imagen hiperespectral en formato MATLAB utilizando la opción "Abrir hiperespectral" del menú "Archivo" y que esta se visualizase por pantalla, pulsamos la opción de la barra de herramientas que nos permite almacenar la banda actual. Tras seleccionar una ubicación y un nombre al fichero, pulsamos el botón guardar. Una vez que tenemos la banda almacenada, accedemos a la ubicación de la misma y procedemos a visualizar su contenido con el visor de imágenes que incorpora el sistema operativo Ubuntu.

RF.11 Abrir fichero ground truth en formato MATLAB.

Estado: Cumplido.

Prueba de funcionamiento: Desde la pantalla principal accedemos a la opción de "Abrir ground truth" del menú "Archivo" y seleccionamos un fichero en formato MATLAB que no corresponde con el *ground truth* de la imagen hiperespectral cargada en la aplicación. En este caso el programa muestra un mensaje por pantalla indicando que el *ground truth* no se corresponde y nos permite seleccionar otro fichero. Posteriormente se selecciona el fichero *ground truth* correcto y este se abre. Una vez abierto, imprimimos los valores de algunos píxeles seleccionados aleatoriamente y los cotejamos con los correspondientes valores que obtenemos al abrir el fichero *ground truth* con el programa MATLAB.

RF.14 Mostrar ground truth.

[2]http://linux.about.com/library/cmd/blcmdl1_cmp.htm

Estado: Cumplido.

Prueba de funcionamiento: Se selecciona un fichero en formato MATLAB que contenga el *ground truth* de la imagen hiperespectral abierta en el programa para que se muestre por pantalla y poder compararla con la imagen suministrada por el equipo de investigación.

RF.16 Cambiar banda.

Estado: Cumplido.

Prueba de funcionamiento: Una vez cargada una imagen hiperespectral y que esta se visualiza por pantalla, utilizamos la rueda del ratón para desplazarnos por las distintas bandas. Se prueba que al intentar pasar más allá de la primera y a la última banda el programa no falle.

RF.17 Ir a banda.

Estado: Cumplido.

Prueba de funcionamiento: Una vez cargada una imagen hiperespectral y que esta se visualiza por pantalla, seleccionamos de la barra de herramientas la opción que nos permite saltar a una banda determinada, y en la ventana donde se nos pide la banda a la que saltar, indicamos una banda válida. También se comprueba que la aplicación no permite seleccionar una banda fuera del rango posible, es decir, más allá del número de bandas que tiene la imagen hiperespectral.

6.1.2. Segundo Ciclo Sprint

Según la planificación del segundo ciclo de *sprint*, estos son los requisitos funcionales que se han implementado:

- Abrir fichero de datos en formato RAW (RF.02 tabla 4.14).
- Abrir fichero de datos en formato HRW (RF.03 tabla 4.15).
- Guardar fichero de datos en formato RAW (RF.06 tabla 4.18).
- Guardar fichero de datos en formato HRW (RF.07 tabla 4.19).
- Abrir fichero *ground truth* en formato RAW (RF.12 tabla 4.24).
- Abrir fichero *ground truth* en formato HRW (RF.13 tabla 4.25).

- Mostrar información plugins cargados (RF.22 tabla 4.34).
- Añadir plugin al proceso de clasificación (RF.23 tabla 4.35).
- Construir plugin *Watershed* (RF.34 tabla 4.46).

Según la matriz de trazabilidad de la tabla 4.77, en este ciclo se implementaron de forma parcial o total los siguientes casos de uso:

- Abrir imagen hiperespectral (CU.01 tabla 4.61).
- Guardar imagen hiperespectral (CU.02 tabla 4.62).
- Abrir *ground truth* (CU.05 tabla 4.65).
- Información plugins (CU.12 tabla 4.72).
- Configurar clasificación (CU.13 tabla 4.73).
- Ejecutar clasificación (CU.14 tabla 4.74).

RF.02 Abrir fichero de datos en formato RAW.

Estado: Cumplido.

Prueba de funcionamiento: Desde la pantalla principal accedemos a la opción de "Abrir hiperespectral" del menú "Archivo" y seleccionamos un fichero en formato RAW, introduciendo a continuación las características de la imagen (alto, ancho, bandas, ...) para poder visualizarla, ya que el formato RAW no incluye esta información. Posteriormente utilizamos la opción de "Guardar hiperespectral" del menú "Archivo" para almacenar la imagen en formato MATLAB. Una vez que disponemos de la imagen en formato MATLAB podemos proceder con la prueba de funcionamiento tal y como se hizo para el requisito RF.01, es decir, imprimiendo los valores de algunos píxeles seleccionados aleatoriamente y verificándolos con el programa MATLAB.

RF.03 Abrir fichero de datos en formato HRW.

Estado: Cumplido.

Prueba de funcionamiento: Desde la pantalla principal accedemos a la opción de "Abrir hiperespectral" del menú "Archivo" y seleccionamos un fichero en formato HRW, visualizándose la imagen por pantalla. Posteriormente utilizamos la opción de "Guardar hiperespectral" del menú "Archivo" para almacenar la imagen en formato MATLAB. Una vez que disponemos de la imagen en formato MATLAB podemos proceder con la prueba de funcionamiento tal y como se hizo para los requisitos RF.01 y RF.02, es decir, imprimiendo los valores de algunos píxeles seleccionados aleatoriamente y verificándolos con el programa MATLAB.

RF.06 Guardar fichero de datos en formato RAW.

Estado: Cumplido.

Prueba de funcionamiento: Una vez abierta una imagen hiperespectral en formato RAW, seleccionamos la opción de "Guardar hiperespectral" del menú "Archivo" y guardamos la imagen seleccionando nuevamente el formato RAW. Posteriormente utilizaremos el programa cmp desde la línea de comandos para comparar el fichero original de la imagen hiperespectral y el fichero almacenado por nuestro programa.

RF.07 Guardar fichero de datos en formato HRW.

Estado: Cumplido.

Prueba de funcionamiento: Una vez abierta una imagen hiperespectral en formato HRW, seleccionamos la opción de "Guardar hiperespectral" del menú "Archivo" y guardamos la imagen seleccionando también el formato HRW. Posteriormente utilizaremos el programa cmp desde la línea de comandos para comparar el fichero original de la imagen hiperespectral y el fichero almacenado por nuestro programa.

RF.12 Abrir fichero ground truth en formato RAW.

Estado: Cumplido.

Prueba de funcionamiento: Desde la pantalla principal accedemos a la opción de "Abrir ground truth" del menú "Archivo" y seleccionamos un fichero en formato RAW que no corresponde con el *ground truth* de la imagen hiperespectral cargada en la aplicación. En este caso el programa muestra un mensaje por pantalla indicando que el *ground truth* no se corresponde y nos permite seleccionar otro fichero. Posteriormente se selecciona el fichero *ground truth* correcto y este

se abre. Una vez abierto, imprimimos los valores de algunos píxeles seleccionados aleatoriamente y los cotejamos con los correspondientes valores que obtenemos al abrir el fichero *ground truth* con el editor hexadecimal Khexedit[3].

RF.13 Abrir fichero ground truth en formato HRW.

Estado: Cumplido.

Prueba de funcionamiento: Desde la pantalla principal accedemos a la opción de "Abrir ground truth" del menú "Archivo" y seleccionamos un fichero en formato HRW que no corresponde con el *ground truth* de la imagen hiperespectral cargada en la aplicación. En este caso el programa muestra un mensaje por pantalla indicando que el *ground truth* no se corresponde y nos permite seleccionar otro fichero. Posteriormente se selecciona el fichero *ground truth* correcto y este se abre. Una vez abierto, imprimimos los valores de algunos píxeles seleccionados aleatoriamente y los cotejamos con los correspondientes valores que obtenemos al abrir el fichero *ground truth* con el editor hexadecimal Khexedit.

RF.22 Mostrar información plugins cargados.

Estado: Cumplido.

Prueba de funcionamiento: Se accede desde la opción correspondiente de la barra de herramientas a la ventana que muestra la información de los plugins. Una vez cargada la ventana, seleccionamos un plugin y se muestra por pantalla la información correspondiente al mismo (nombre, autor, versión y comentarios).

RF.23 Añadir plugin al proceso de clasificación.

Estado: Cumplido.

Prueba de funcionamiento: En primer lugar modificamos el código del plugin *Watershed* para que aparezca en todas las pestañas de la ventana de configuración de la clasificación. Después, desde la pantalla principal, accedemos a la opción "Configurar clasificación" del menú "Configurar". A continuación, en cada una de las pestañas existentes seleccionamos el plugin *Watershed* y pulsamos el botón añadir, verificando que se añade a la lista de plugins seleccionados. Finalmente, una vez que hemos añadido el plugin en todas las pestañas, cerramos la ventana y volvemos a entrar para comprobar que se mantienen los cambios.

[3]http://manpages.ubuntu.com/manpages/hardy/man1/khexedit.1.html

RF.34 Construir plugin *Watershed*.

Estado: Cumplido.

Prueba de funcionamiento: Se modifica el código de la aplicación para que una vez que se lea correctamente un fichero *ground truth* (previamente se tubo que leer una imagen hiperespectral) se ejecute el plugin y se retorne a la aplicación principal, sustituyendo los datos del *ground truth* por los de la clasificación devueltos por el plugin y mostrándolos en pantalla. Este resultado se comparará con el resultado suministrado por el grupo de investigación correspondiente a la ejecución del plugin por separado y utilizando los mismos parámetros de entrada, así como las mismas imágenes.

6.1.3. Tercer Ciclo Sprint

Según la planificación del tercer ciclo de *sprint*, estos son los requisitos funcionales que se han implementado:

- Zoom imagen hiperespectral (RF.08 tabla 4.20).
- Coordenadas zoom imagen hiperespectral (RF.09 tabla 4.21).
- Mostrar histograma (RF.18 tabla 4.30).
- Histograma en formato texto (RF.19 tabla 4.31).
- Comparar histogramas (RF.20 tabla 4.32).
- Guardar los datos del histograma (RF.21 tabla 4.33).
- Eliminar plugin del proceso de clasificación (RF.24 tabla 4.36).
- Ordenar los plugins del proceso de clasificación (RF.25 tabla 4.37).
- Pasar parámetros a los plugins del proceso de clasificación (RF.26 tabla 4.38).

Según la matriz de trazabilidad de la tabla 4.77, en este ciclo se implementaron de forma parcial o total los siguientes casos de uso:

- Zoom imagen hiperespectral (CU.03 tabla 4.63).
- Mostrar histograma (CU.09 tabla 4.69).
- Comparar histogramas (CU.10 tabla 4.70).
- Guardar datos histograma (CU.11 tabla 4.71).
- Configurar clasificación (CU.13 tabla 4.73).

RF.08 Zoom imagen hiperespectral.

Estado: Cumplido.

Prueba de funcionamiento: En primer lugar se abre una imagen hiperespectral. Después se activa la ventana del zoom desde el icono correspondiente en la barra de herramientas y se desplaza el ratón por encima de la imagen hiperespectral para verificar que se realiza un zoom de la misma. Se prueban todos los bordes de la imagen, para verificar que no se produce ningún error.

RF.09 Coordenadas zoom imagen hiperespectral.

Estado: Cumplido.

Prueba de funcionamiento: Tras abrir una imagen hiperespectral, activamos la ventana del zoom y comprobamos que aparecen las coordenadas de la zona del zoom en dicha ventana. Movemos el ratón sobre la imagen hiperespectral para verificar que las coordenadas cambian según la posición del puntero.

RF.18 Mostrar histograma.

Estado: Cumplido.

Prueba de funcionamiento: Después de abrir una imagen hiperespectral seleccionamos la opción de "Histograma" del menú "Herramientas". En la nueva ventana debe aparecer la firma espectral del píxel con coordenadas (0, 0). Ahora, utilizamos el ratón para seleccionar distintos píxeles de la imagen hiperespectral y verificar que la firma espectral mostrada en la ventana del histograma cambia.

RF.19 Histograma en formato texto.

Estado: Cumplido.

Prueba de funcionamiento: Se selecciona una imagen hiperespectral y se activa la ventana del histograma. En ese momento debemos visualizar la firma espectral del píxel con coordenadas (0, 0) y a continuación todos los datos que forman dicha firma espectral, en formato texto y de forma ordenada. Ahora, utilizamos el ratón para seleccionar distintos píxeles de la imagen hiperespectral y verificar que los datos varían.

RF.20 Comparar histogramas.

Estado: Cumplido.

Prueba de funcionamiento: Una vez que hemos abierto una imagen hiperespectral y activado la ventana del histograma, esta contiene la firma espectral del píxel con coordenadas (0, 0). Pulsamos el icono situado en la parte superior de la ventana del histograma, para activar la comparación de histogramas. Se debe comprobar que se muestran dos firmas espectrales iguales y que al seleccionar un píxel distinto en la imagen hiperespectral, una de las firmas espectrales permanece fija mientras que la otra muestra la firma espectral del nuevo píxel seleccionado.

RF.21 Guardar los datos del histograma.

Estado: Cumplido.

Prueba de funcionamiento: Se abre una imagen hiperespectral y se activa la ventana del histograma para posteriormente pulsar sobre la opción que permite almacenar los datos del histograma. Una vez que los datos se han guardado, se abre el fichero que los contiene y se cotejan con los datos en pantalla.

RF.24 Eliminar plugin del proceso de clasificación.

Estado: Cumplido.

Prueba de funcionamiento: Dentro de la ventana de configuración de la clasificación se añaden plugins a la lista de plugins seleccionados en todas las pestañas disponibles. Después se utiliza el botón "Quitar" y, pestaña a pestaña, verificamos que se elimina el plugin de la lista de seleccionados. Se intenta también eliminar cuando no hay ningún plugin en la lista de plugins seleccionados.

RF.25 Ordenar los plugins del proceso de clasificación.

Estado: Cumplido.

Prueba de funcionamiento: Después de mostrar la ventana de configuración de la clasificación y tras añadir plugins a la lista de plugins seleccionados en todas las pestañas, utilizamos los botones de "Subir" y "Bajar" para ordenar la lista de plugins. También se intenta bajar el último plugin de la lista y subir el primer plugin de la lista.

RF.26 Pasar parámetros a los plugins del proceso de clasificación.

Estado: Cumplido.

Prueba de funcionamiento: Tras añadir plugins a la lista de plugins seleccionados en todas las pestañas de la ventana de configuración de la clasificación, seleccionamos un plugin de dicha lista y pulsamos el botón "Parámetros" para cada una de las pestañas. En la ventana que se despliega probamos a modificar los parámetros por defecto y verificamos que al volver a entrar estos cambios se mantienen.

6.1.4. Cuarto Ciclo Sprint

Según la planificación del cuarto ciclo de *sprint*, estos son los requisitos funcionales que se han implementado:

- Ejecutar clasificación (RF.27 tabla 4.39).
- Ver la salida de los distintos plugins ejecutados (RF.28 tabla 4.40).
- Abortar la ejecución de la clasificación (RF.29 tabla 4.41).
- Construir plugin ELM (RF.30 tabla 4.42).
- Construir plugin MV (RF.32 tabla 4.44).
- Mostrar los resultados de la clasificación (RF.35 tabla 4.47).

Según la matriz de trazabilidad de la tabla 4.77, en este ciclo se implementó de forma parcial el siguiente caso de uso:

- Ejecutar clasificación (CU.14 tabla 4.74).

RF.27 Ejecutar clasificación.

Estado: Cumplido.

Prueba de funcionamiento: En primer lugar se modifican los distintos plugins para que durante su ejecución impriman un texto por pantalla. Posteriormente, después de haber configurado el proceso de clasificación añadiendo algunos plugins y tras abrir una imagen hiperespectral y su *ground truth* correspondiente, se procede a ejecutar la clasificación, comprobando que los procesos implicados imprimen su texto por pantalla, y verificando que se llame a todos los plugins y en el orden establecido.

RF.28 Ver la salida de los distintos plugins ejecutados.

Estado: Cumplido.

Prueba de funcionamiento: Después de abrir una imagen hiperespectral y su correspondiente *ground truth*, y de haber configurado algunos plugins para el proceso de clasificación, ejecutamos dicho proceso y verificamos que los distintos plugins escriben en la zona reservada para ese propósito.

RF.29 Abortar la ejecución de la clasificación.

Estado: Cumplido.

Prueba de funcionamiento: Preparamos una clasificación abriendo una imagen hiperespectral y su correspondiente *ground truth*, y añadiendo algunos plugins al proceso de clasificación. Antes de que comience la ejecución de la clasificación, debemos anotar la cantidad de memoria libre en el sistema, para lo cual utilizamos el comando *free*[4] de la línea de comandos de Linux. Una vez que comienza la ejecución de la clasificación pulsamos el botón abortar y esperamos a que el proceso finalice, tras lo cual debemos verificar, nuevamente con el comando *free*, que no ha quedado memoria sin liberar.

RF.30 Construir plugin ELM.

Estado: Cumplido.

Prueba de funcionamiento: Se procede a configurar el proceso de clasificación utilizando el plugin ELM y visualizando el resultado por pantalla. Este, al igual que se hizo con el plugin *Watershed*, se contrastará con los resultados suministrados por el equipo de investigación.

RF.32 Construir plugin MV.

Estado: Cumplido.

Prueba de funcionamiento: Para probar este plugin necesitamos configurar el proceso de clasificación con los plugins ELM, *Watershed* y MV de manera que este último agrupe los resultados de los dos primeros. El resultado de esta clasificación se contrastará con los resultados suministrados por el equipo de investigación.

RF.35 Mostrar los resultados de la clasificación.

[4]http://linux.die.net/man/1/free

Estado: Cumplido.

Prueba de funcionamiento: Después de abrir una imagen hiperespectral y su *ground truth*, de haber configurado el proceso de clasificación añadiendo plugins al mismo y de haber ejecutado la clasificación, se comprueba que se muestren por pantalla tanto la clasificación final como los valores de precisión de dicha clasificación.

6.1.5. Quinto Ciclo Sprint

Según la planificación del quinto ciclo de *sprint*, estos son los requisitos funcionales que se han implementado:

- Construir plugin SVM (RF.31 tabla 4.43).
- Construir plugin RQS (RF.33 tabla 4.45).
- Guardar la clasificación (RF.36 tabla 4.48).
- Guardar imagen resultante del proceso de clasificación (RF.37 tabla 4.49).
- Abrir clasificación (RF.38 tabla 4.50).

Según la matriz de trazabilidad de la tabla 4.77, en este ciclo se implementaron de forma parcial o total los siguientes casos de uso:

- Ejecutar clasificación (CU.14 tabla 4.74).
- Guardar clasificación (CU.15 tabla 4.75).
- Abrir clasificación (CU.16 tabla 4.76).

RF.31 Construir plugin SVM.

Estado: Cumplido.

Prueba de funcionamiento: Se procede a configurar el proceso de clasificación utilizando el plugin SVM y visualizando el resultado por pantalla. Este, al igual que se hizo con los plugins anteriores, se contrastará con los resultados suministrados por el equipo de investigación.

RF.33 Construir plugin RQS.

Estado: Cumplido.

Prueba de funcionamiento: Se procede a configurar el proceso de clasificación utilizando el plugin RQS y visualizando el resultado por pantalla. Este se contrastará con los resultados suministrados por el equipo de investigación.

RF.36 Guardar la clasificación.

Estado: Cumplido.

Prueba de funcionamiento: Después de haber configurado el proceso de clasificación añadiendo algunos plugins y tras abrir una imagen hiperespectral y su *ground truth* correspondiente, se procede a ejecutar la clasificación, tras la cual utilizamos el botón "Guardar configuración", que se muestra en la ventana de resultados para almacenarla. Posteriormente abrimos el fichero resultante con el editor hexadecimal Khexedit[5] para verificar que los datos se almacenaron correctamente.

RF.37 Guardar imagen resultante del proceso de clasificación.

Estado: Cumplido.

Prueba de funcionamiento: Tras la ejecución de una clasificación, utilizamos el icono correspondiente a la acción de guardar la imagen resultante del proceso de clasificación situado en la misma ventana, para almacenarla. Posteriormente usamos el visor de Ubuntu para verificar que se almacenó correctamente.

RF.38 Abrir clasificación.

Estado: Cumplido.

Prueba de funcionamiento: Desde la pantalla principal accedemos a la opción "Abrir clasificación" del menú "Clasificación", seleccionamos una clasificación guardada con anterioridad y visualizamos los datos por pantalla, verificando que los datos que se muestran se corresponden con los datos que se generaron al ejecutar la clasificación que estamos abriendo.

6.2. Evaluación Heurística

En esta sección se va a realizar un análisis de las principales reglas heurísticas de usabilidad definidas por Jakon Nielsen[6], para así poder validar la usabilidad

[5]http://manpages.ubuntu.com/manpages/hardy/man1/khexedit.1.html
[6]http://www.nngroup.com/people/jakob-nielsen/

de la aplicación.

A continuación se enumeran las principales reglas heurísticas y de que forma se implementan en la aplicación:

1. **Visibilidad del estado del sistema.** El sistema debe siempre mantener a los usuarios informados del estado del sistema, con una realimentación apropiada y en un tiempo razonable.

   – En la aplicación existe una barra de estado en la que se muestran mensajes al usuario informándole del estado del sistema.

2. **Utilizar el lenguaje de los usuarios.** El sistema debe hablar el lenguaje de los usuarios, con las palabras, las frases y los conceptos familiares, en lugar de que los términos estén orientados al sistema. Utilizar convenciones del mundo real, haciendo que la información aparezca en un orden natural y lógico.

   – Aunque la aplicación está dirigida a un perfil más específico, se ha utilizado un lenguaje genérico, permitiendo que cualquier usuario pueda utilizarla.

3. **Control y libertad para el usuario.** Los usuarios eligen a veces funciones del sistema por error y necesitan a menudo *una salida de emergencia claramente marcada*, esto es, salir del estado indeseado sin tener que pasar por un diálogo extendido. Es importante disponer de deshacer y rehacer.

   – A pesar de que no existen opciones de deshacer y rehacer, en cualquier momento el usuario puede cancelar la acción que esté realizando cerrando la ventana correspondiente o pulsando el botón cancelar.

4. **Consistencia y estándares.** Los usuarios no deben tener que preguntarse si las diversas palabras, situaciones, o acciones significan la misma cosa. En general siga las normas y convenciones de la plataforma sobre la que se está implementando el sistema.

   – Al tratarse de una aplicación con unas funcionalidades poco genéricas como puede ser “Guardar banda” o “Mostrar histograma”, es difícil encontrar iconos que sean fácilmente identificables por el usuario. A pesar de eso se han utilizado iconos que representan de la mejor forma posible la acción con la que se asocia.

5. **Prevención de errores.** Es importante prevenir la aparición de errores que mejor que generar buenos mensajes de error.

   - En toda la aplicación se previene la aparición de errores, por ejemplo, deshabilitando las acciones que no se puedan realizar mientra no se de una condición concreta.

6. **Minimizar la carga de la memoria del usuario.** El usuario no debería tener que recordar la información de una parte de diálogo a la otra. Es mejor mantener objetos, acciones, y las opciones visibles que memorizar.

   - En ninguno de los diálogos que se utilizan en la aplicación el usuario debe memorizar información.

7. **Flexibilidad y eficiencia de uso.** Las instrucciones para el uso del sistema deben ser visibles o fácilmente accesibles siempre que se necesiten. Los aceleradores o atajos no vistos por el principiante, mejoran la interacción para el usuario experto de tal manera que el sistema puede servir para usuarios inexpertos y experimentados. Es importante que el sistema permita personalizar acciones frecuentes.

   - La aplicación hace uso de los atajos de teclado en los menús y también se configuran botones por defecto en todas las ventanas de diálogo.

8. **Los diálogos estéticos y diseño minimalista.** No deben contener la información que sea inaplicable o se necesite raramente. Cada unidad adicional de la información en un diálogo compite con las unidades relevantes de la información y disminuye su visibilidad relativa.

   - El diseño de los diálogos contiene el mínimo número de componentes en cada caso, evitando crear diálogos extensos y cargados de campos que rellenar.

9. **Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de los errores.** Que los mensajes de error se deben expresar en un lenguaje claro (no haya códigos extraños), se debe indicar exactamente el problema, y deben ser constructivos.

   - Los mensajes de error que el sistema muestra al usuario utilizan un lenguaje sencillo e intentan guiar al usuario hacia la solución del mismo.

10. **Ayuda y documentación.** Aunque es mejor si el sistema se pueda usar sin documentación, puede ser necesario disponer de ayuda y documentación. Ésta ha de ser fácil de buscar, centrada en las tareas del usuario, tener información de las etapas a realizar y que no sea muy extensa.

    - En la aplicación existe una opción de ayuda en la que se incluye el manual de usuario. También existen botones de ayuda en aquellos diálogos que por su complejidad lo necesitan.

Tras este análisis heurístico se puede concluir que la aplicación cumple, en líneas generales, con los principales reglas de usabilidad de Nielsen. De todas formas queda margen de mejora como puede ser la incorporación de opciones de deshacer y rehacer.

6.3. Evaluación de Usabilidad

6.3.1. Técnicas Utilizadas

Para este proyecto se han utilizado dos técnicas de evaluación de usabilidad. Por un lado se ha realizado un test de usabilidad para recoger las impresiones de los usuarios sobre la aplicación y por otro lado se ha utilizado el *método del conductor*, que consiste en guiar a los usuarios mientras usan el sistema tomando nota de los percances que puedan ocurrir y de los comentarios de los propios usuarios. La decisión de usar el método del conductor se debe a la necesidad de verificar que el test había extraído todas los posibles fallos de usabilidad de la aplicación.

Las pruebas de usabilidad constan de un cuestionario que se le pasará a los individuos de prueba después de que estos se familiaricen con la aplicación mediante la realización de una serie de tareas sobre la misma.

Las tareas propuestas al usuario son las siguientes:

1. Ejecutar la aplicación y familiarizarse con ella. Se dispone de un tiempo máximo de 2 minutos para la realización de esta tarea.

2. Intentar realizar las siguientes acciones:

   - Abrir una imagen hiperespectral en formato MATLAB.
   - Guardar la imagen en formato RAW.
   - Volver a abrir una imagen hiperespectral pero en formato RAW.
   - Realizar un zoom sobre la imagen.
   - Comparar el histograma de dos píxeles.
   - Guardar una banda.
   - Configurar varios plugins para el proceso de clasificación modificando los parámetros por defecto.
   - Ejecutar una clasificación de la imagen.
   - Guardar la clasificación.

6.3.2. Cuestionario

Después de efectuar las tareas descrita en la sección anterior, se cubrirá un cuestionario que pretende valorar la usabilidad de la aplicación. A continuación se enumeran las distintas preguntas de las que se compone el cuestionario.

Valorando entre 1, muy en desacuerdo y 5, muy de acuerdo conteste las siguientes preguntas:

1. El acceso a las distintas opciones me parece fácil.
2. Me siento cómodo/a al utilizarla.
3. Parece que es muy compleja.
4. Siempre se donde me encuentro.
5. La funcionalidad de los botones es obvia a partir de su diseño.
6. Los avisos son breves y no ambiguos.
7. Los campos de los formularios son intuitivos y fáciles de rellenar.
8. Es fácil obtener ayuda en la forma y momento oportuno.
9. La información es presentada en un orden lógico, simple y natural.
10. Mi satisfacción general sobre la aplicación es buena.

El método utilizado para calificarlas respuestas del cuestionario es el siguiente:

- Las valoraciones muy negativas y muy positivas restarán o sumarán 3 respectivamente.
- Las valoraciones intermedias contribuirán con un valor de 0.
- El resto de las valoraciones restarán o sumarán 1 en función de si son negativas o positivas.

6.3.3. Resultados del Cuestionario

Se realizó una prueba a 5 usuario cuyo perfil era el siguiente:

- Usuario que maneja habitualmente el sistema operativo Linux.
- Conocimientos de programación a nivel avanzado.
- Sin conocimientos en procesamiento de imágenes.

- Se enfrentaban por primera vez a la aplicación.

Los resultados obtenidos se pueden observar en la tabla 6.1:

Cuestionario	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
1	5	5	1	4	4	5	5	4	4	5
2	4	5	1	5	4	5	5	5	5	5
3	5	4	1	5	3	5	4	5	5	4
4	4	5	1	5	4	4	5	5	5	4
5	5	5	1	4	4	5	5	5	5	5

Tabla 6.1: Tabla de puntuación del test de usabilidad.

Las respuestas se acercan notablemente al máximo establecido para cada una, es decir, 5 para las de naturaleza positiva y 1 para las de naturaleza negativa, poniendo de manifiesto la satisfacción de los usuarios respecto a la usabilidad.

En la figura 6.1 podemos ver la puntuación media de las respuestas. Cabe destacar que los resultados de la pregunta 5 "*La funcionalidad de los botones es obvia a partir de su diseño*" arrojan una clara necesidad de analizar cuales son los botones que no resultan adecuados en la aplicación.

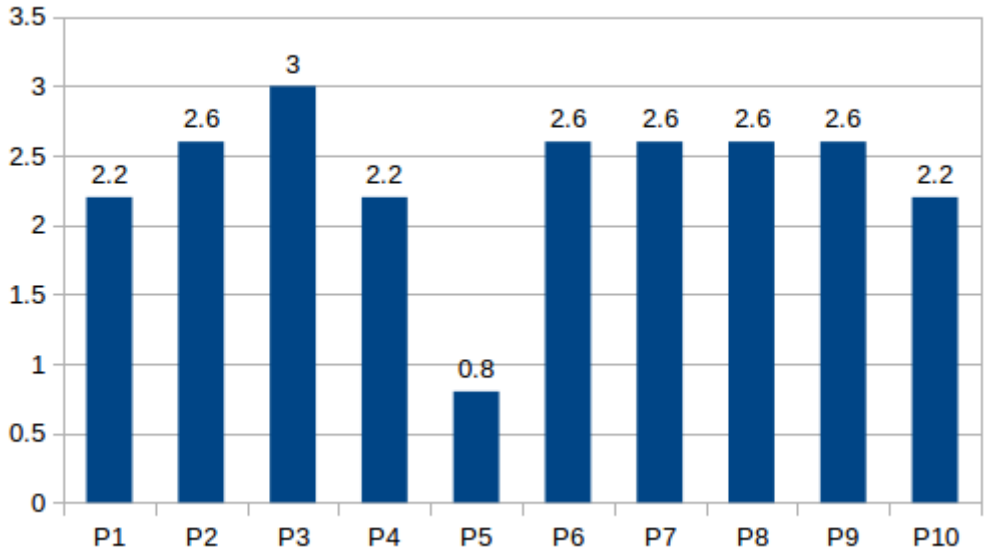

Figura 6.1: Puntuación media del cuestionario de usabilidad

6.3.4. Propuestas de Mejora

Una vez analizados los resultados de la encuesta de usabilidad y los resultados del método del conductor, estas han sido las propuestas:

- Mejorar la barra de herramientas de la aplicación, modificando los iconos actuales por unos más acordes con la acción a la que hacen referencia.
- Ampliar el tamaño de los botones de la barra de herramientas, pasando de una resolución de 20 x 20 píxeles a una de 32 x 32 píxeles.
- Ajustar automáticamente el tamaño de la imagen hiperespectral, de forma que si la esta excede el tamaño máximo de la pantalla, no se muestre la imagen completa y se haga uso de barras de desplazamiento para explorarla.
- Incluir mnemónicos en las opciones de los menús para que el manejo mediante teclado sea más cómodo.
- Incluir la opción de salir del programa en el menú "Archivo" y añadirle un mnemónico y un atajo de teclado.
- Incluir la opción de "Acerca de" en el menú "Ayuda".

Capítulo 7

Conclusiones y Posibles Ampliaciones

En este capítulo se describirán las conclusiones del proyecto y las posibles ampliaciones futuras.

7.1. Conclusiones

En este trabajo hemos desarrollado una herramienta de software libre capaz de mostrar las diferentes características detalladas de una imagen hiperespectral, así como aplicar distintas técnicas de clasificación desarrolladas en el grupo de investigación, facilitando la comparación de las mismas, en términos de acierto de la clasificación, de manera sencilla y lo mas transparente posible al usuario.

A pesar de que el software desarrollado en este proyecto fue diseñado para ser aplicado a imágenes hiperespectrales de superficie terrestre, podría ser utilizado con imágenes hiperespectrales obtenidas en otros ámbitos.

Después de realizar las pruebas de verificación y validación, y los análisis heurísticos y de usabilidad podemos considerar que con este proyecto se ha desarrollado una aplicación que ofrece tanto herramientas para el análisis de las imágenes hiperespectrales como un completo sistema que permite aplicar distintas técnicas de clasificación a las mismas. Además, al haber sido diseñada de modo que el usuario puede configurar el proceso de clasificación seleccionando diferentes etapas de preprocesado, clasificación y postprocesado, es una herramienta fácilmente ampliable. Esta característica convierte al software en una herramienta valiosa para que el grupo de investigación compare las diferentes técnicas de clasificación que vaya desarrollando o que las aplique a distintas imágenes.

Por otro lado, la utilización de la metodología *Scrum* para el desarrollo del pro-

yecto ha sido una experiencia positiva y un aprendizaje constante en todas las etapas del desarrollo del mismo.

Finalmente indicar que los objetivos planteados en el trabajo se han cumplido pues el software desarrollado es completamente funcional.

7.2. Posibles Ampliaciones Futuras

La principal ampliación que se debería acometer es el desarrollo de nuevas técnicas de clasificación, de manera que se amplíe el conjunto de plugins que se puedan utilizar en la aplicación dándole mayor valor a la misma.

Otras posibles ampliaciones pueden ser:

- La incorporación de plugins para las fases de preprocesado y postprocesado.
- Al realizar la comparación de dos histogramas no limitarse únicamente a una comparación gráfica, sino que se muestren los datos de los dos histogramas en forma de texto.
- Ampliar el número de formatos de ficheros que la aplicación puede manejar.
- Permitir realizar zoom sobre el *ground truth*, con la opción de sincronizar este con el zoom de la imagen hiperespectral, visualizando en ambos la misma zona.
- Incorporar acciones de rehacer y deshacer.
- Que la comparación de datos de eficiencia no tenga que hacerla el usuario.
- Búsqueda de píxeles con firmas espectrales iguales o con un porcentaje de diferencia a uno dado.
- Poder mostrar al usuario, en formato HTML, los posibles resultado gráficos intermedios generados por cada uno de los plugins que integran una técnica de clasificación.
- Permitir guardar y cargar sesiones de trabajo para evitar que el usuario tenga que volver a configurar todos los datos de la aplicación cada vez que la cierre y posteriormente la abra.

Apéndice A

Manuales de Usuario

A.1. Manual de Instalación

A.1.1. Compilación del Código

Para la compilación del código solo debemos acceder al directorio "Codigo" del CD que se suministra con la memoria y ejecutar el comando *make*, siendo necesario disponer de un equipo con las siguientes librerías instaladas:

- GTK+-2.0.
- Glade-2.0.
- Gmodule-export-2.0.
- Pango.
- Matio.
- HDF5.
- LAPACKE.
- LAPACK con BLAS.
- Librería de compresión Z.

A.1.2. Código Compilado

En el CD de la memoria también se incluye una versión precompilada del software, realizada sobre una máquina con Ubuntu 10.04 LTS (64-bits).

A.2. Manual de Usuario

A.2.1. Interfaz Gráfica

En las figuras A.1 y A.2 podemos observar la apariencia principal que ofrece la aplicación, así como el aspecto de sus menús. Los elementos que forman esta pantalla son los siguientes:

- Un menú con la mayoría de las opciones que permite la aplicación.
- Una barra de herramientas para acceder rápidamente a las distintas opciones del software.
- Una zona central que será utilizada para ubicar la imagen hiperespectral.
- Una barra de estado, en la parte inferior, para mantener informado al usuario del estado de la aplicación en todo momento.

Figura A.1: Pantalla principal de la aplicación desarrollada en este proyecto

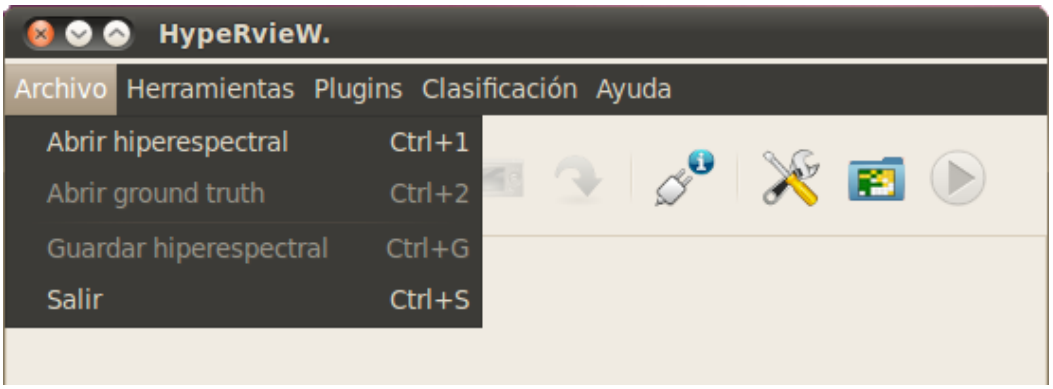

Figura A.2: Aspecto de los menús de la aplicación

A.2.2. Abrir Imagen Hiperespectral

Figura A.3: Icono de la barra de herramientas que permite abrir una imagen hiperespectral

Para abrir una imagen hiperespectral podemos utilizar el icono de la barra de herramientas mostrado en la figura A.3, acceder a la opción "Abrir hiperespectral" del menú "Archivo" o usar la combinación de teclas "Ctrl+1". En la figura A.4 podemos ver el aspecto de la ventana que nos permite seleccionar el fichero de la imagen hiperespectral. En caso de seleccionar un fichero con formato

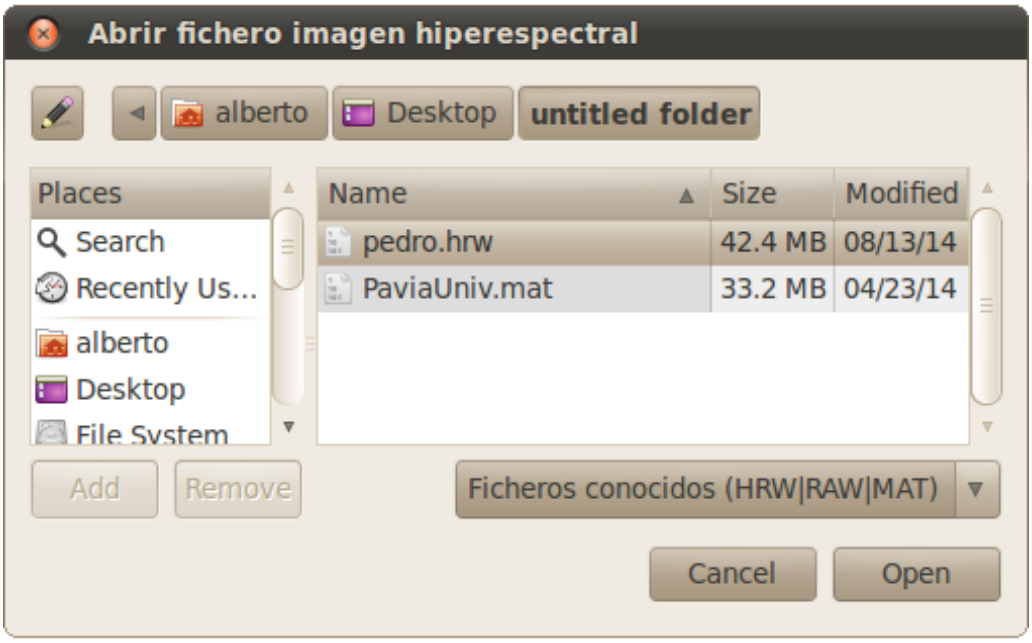

Figura A.4: Selección de una imagen hiperespectral

RAW, deberemos indicarle a la aplicación una serie de datos para que esta pueda interpretar la imagen. En la figura A.5 podemos observar los distintos datos que nos solicita la aplicación y que a continuación pasamos a describir:

- *Tamaño de la imagen*: Especificaremos el alto, ancho y el número de bandas de la imagen.
- *Tamaño de los datos*: Indicar cual es el tamaño en bits de cada dato.
- *Tipo de datos*: Si los datos están almacenados como números enteros o decimales.

Figura A.5: Formulario a cubrir para abrir una imagen hiperespectral en formato RAW

- *Formato de los datos*: Permite especificar el orden en el que están colocados los datos en el fichero. *Pixel-vector* indica que los datos se encuentran almacenados en vectores del mismo tamaño que el número de bandas, donde cada vector hace referencia a todos los datos de un píxel. En este caso primero encontraremos el vector del píxel 1, después el vector del píxel 2, etc. *Bands-vector* indica que los datos se almacenan por bandas, primero todos los píxeles de la banda 1, después todos los píxeles de la banda 2, etc.

- *Tamaño de la cabecera de fichero*: Es el número de bytes que se encuentran antes de los propios datos de la imagen. Podemos usar el botón "Calcular tamaño cabecera" para averiguar el tamaño de la cabecera.

En todo momento un mensaje situado en la parte inferior de la ventana nos indica si la configuración permite o no interpretar el fichero

A.2.3. Guardar Imagen Hiperespectral

El acceso a esta opción se puede hacer mediante el submenú "Guardar hiperespectral" del menú "Archivo", pulsando el icono de la figura A.6 situado en la barra de herramientas o mediante el atajo de teclado "Ctrl+G". Esta acción solo estará disponible si previamente se abrió una imagen hiperespectral.

Figura A.6: Icono de la barra de herramientas que permite guardar una imagen hiperespectral

A la hora de guardar una imagen debemos seleccionar el tipo de formato en el cual queremos hacerlo. En la figura A.7 podemos ver que por defecto está seleccionada la opción de "Guardar como HRW", pudiendola cambiar por "Guardar como RAW" o "Guardar como MATLAB".

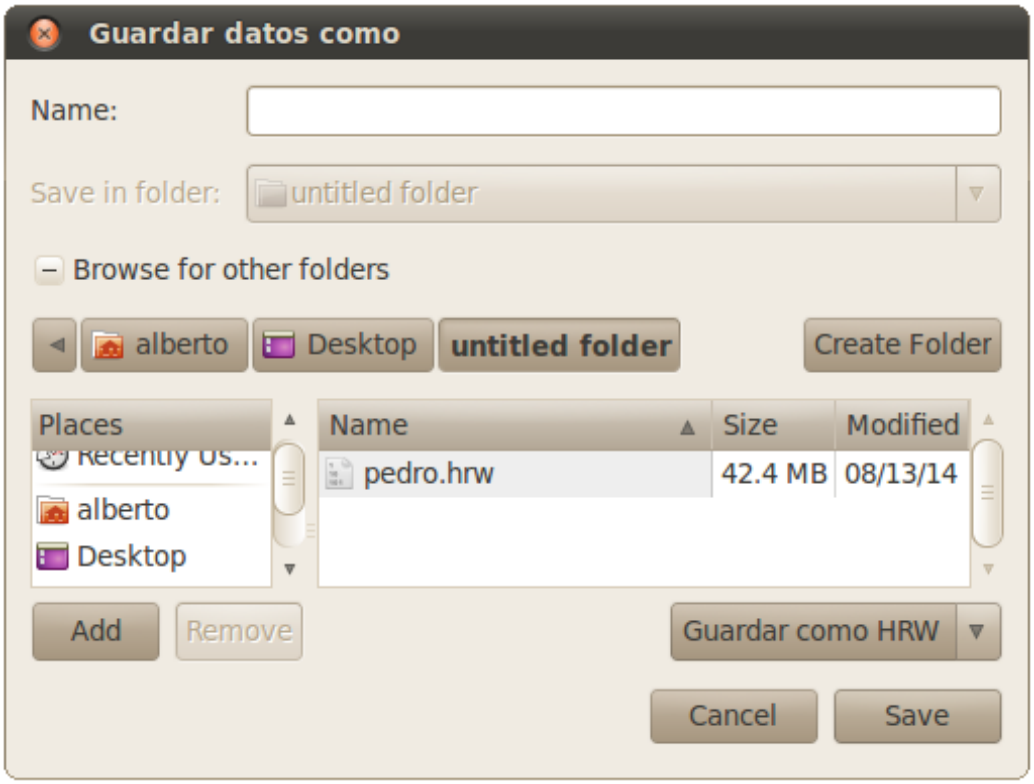

Figura A.7: Guardar la imagen hiperespectral en un fichero

A.2.4. Zoom Imagen Hiperespectral

Esta opción nos permite posicionar el puntero sobre una zona de la imagen hiperespectral y visualizar un zoom de la misma en una pantalla nueva. Debe existir una imagen hiperespectral cargada en la aplicación para que esta opción esté disponible. En la figura A.8 podemos ver el aspecto de dicha ventana.

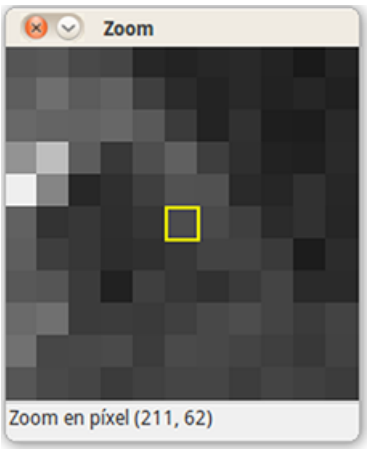

Figura A.8: Zoom sobre la imagen hiperespectral

Para activar/desactivar la ventana de zoom podemos usar el icono de la figura A.9 situado en la barra de herramientas, utilizar el atajo de teclado “Ctrl+Z” o acceder a la opción “Zoom hiperespectral” del menú “Herramientas”.

Figura A.9: Icono de la barra de herramientas que permite activar/desactivar la ventana de zoom

A.2.5. Histograma

El histograma nos va a permitir visualizar los valores de las componentes de reflectancia espectral de un píxel de la imagen hiperespectral. Al igual que pasaba con la la ventana de zoom, solo podremos visualizar el histograma si existe una imagen hiperespectral cargada en la aplicación.

Podemos activar/desactivar la ventana del histograma mediante la opción “Histograma” del menú “Herramientas”, con el atajo de teclado “Ctrl+H” o utilizando el icono de la figura A.10 de la barra de herramientas. En la figura A.11 podemos ver el aspecto de la ventana del histograma.

Figura A.10: Icono de la barra de herramientas que permite activar/desactivar la ventana del histograma

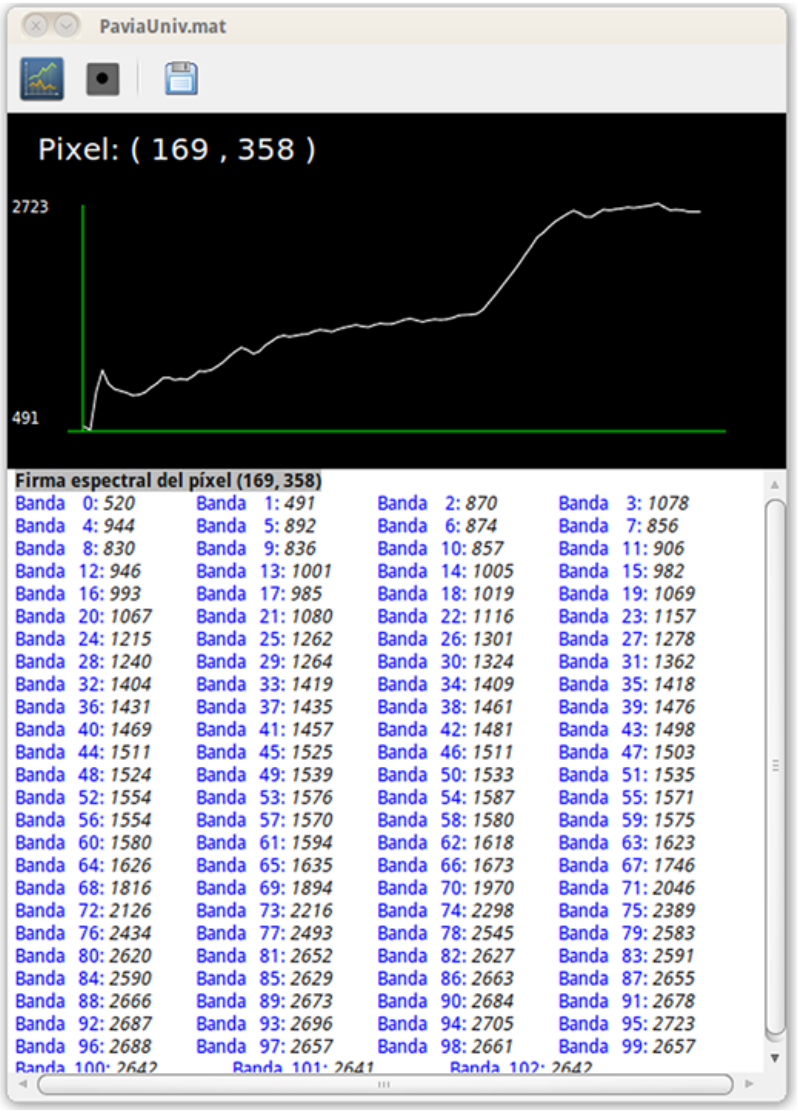

Figura A.11: Ventana con el histograma de un píxel

Una vez activada la ventana, se nos mostrará por defecto el histograma del píxel con coordenadas (0, 0), y debajo de este, todos los datos que forman dicho histograma. Si deseamos visualizar el histograma de un píxel determinado, simplemente deberemos seleccionarlo en la imagen hiperespectral mediante un clic

de ratón.

La ventana del histograma contiene en la parte superior una barra de herramientas con distintas opciones que pasamos a describir por orden de colocación (de izquierda a derecha):

- El primer icono permite comparar gráficamente el histograma de dos píxeles. Al activar esta opción el histograma que se encuentra en la ventana en ese momento permanece fijo y debajo de este aparece un nuevo histograma que podremos modificar seleccionando cualquier otro píxel de la imagen hiperespectral. En la figura A.12 podemos ver la comparación de dos histogramas.

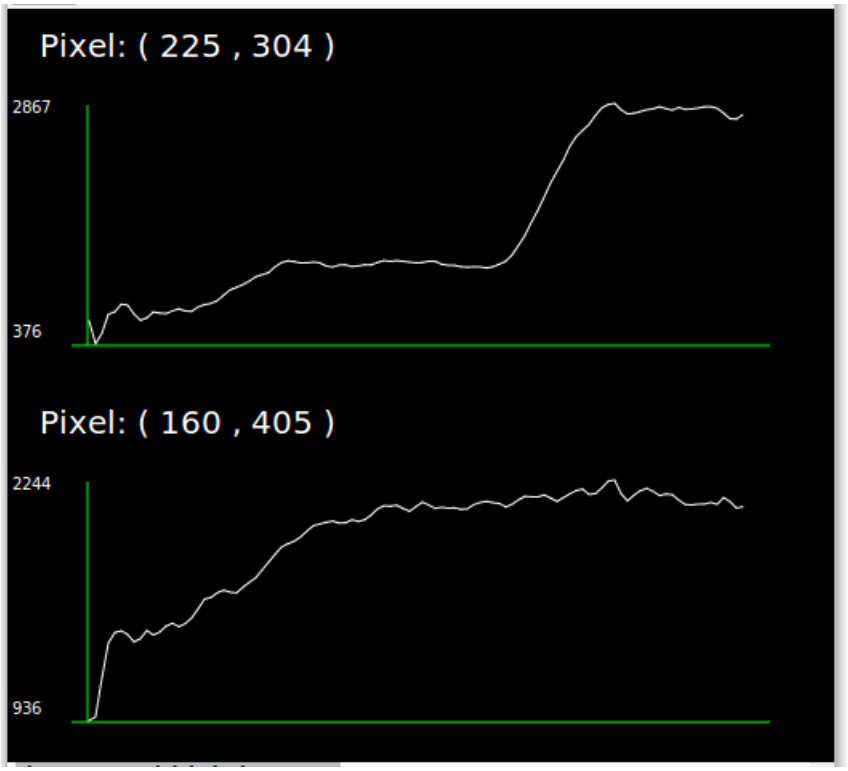

Figura A.12: Comparación gráfica de dos histogramas

- El segundo icono modifica el gráfico del histograma visualizando únicamente los puntos que forman la gráfica. En la figura A.13 podemos ver el aspecto de un histograma dibujado mediante puntos.
- El tercer icono permite almacenar los datos que forman el histograma en un fichero de texto plano. La aplicación nos mostrará una ventana donde podremos indicar el nombre y el destino del fichero que contendrá la información.

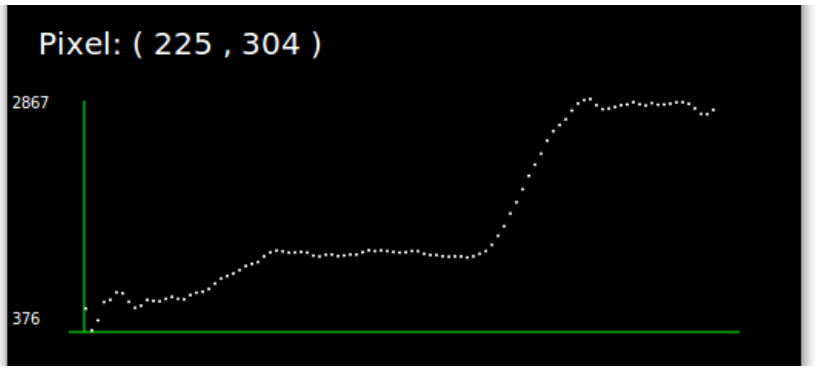

Figura A.13: Histograma de un píxel dibujado mediante puntos

A.2.6. Guardar Banda

Esta opción permite almacenar en formato PGM, la banda que esta siendo visualizada en ese momento en la aplicación. Podremos indicar el nombre y el destino del fichero que almacenará la banda mediante la ventana que muestra el software.

Para ejecutar esta acción accederemos a la opción "Guardar banda" del menú "Herramientas", usaremos el icono de la figura A.14 de la barra de herramientas o con el atajo de teclado "Ctrl+U". Debe existir una imagen hiperespectral cargada en la aplicación para poder guardar una banda.

Figura A.14: Icono de la barra de herramientas que permite guardar la banda de la imagen hiperespectral visualizada en pantalla

A.2.7. Cambiar de Banda

Tenemos dos opciones para cambiar la banda de la imagen hiperespectral que se visualiza en pantalla.

Mediante el Ratón

Podemos cambiar de banda simplemente posicionando sobre la imagen hiperespectral el puntero del ratón y utilizando la rueda de este para desplazarnos secuencialmente entre las distintas bandas.

Ir a Banda

En lugar de cambiar de banda de forma secuencial, podemos utilizar la opción de "Ir a banda" situada en el menú "Herramientas", mediante el icono de la figura A.15 de la barra de herramientas o con el atajo de teclado "Ctrl+B" para saltar a una banda especifica. En la figura A.16 podemos ver el aspecto de la ventana que nos solicita la banda que queremos visualizar. Si indicamos una banda fuera del rango indicado no se producirá ningún cambio.

Figura A.15: Icono de la barra de herramientas que permite cambiar la banda de la imagen hiperespectral visualizada en pantalla

Figura A.16: Ventana que permite cambiar de banda

A.2.8. Abrir Ground Truth

Solo tendremos habilitada la opción de abrir un *ground truth* si previamente hemos abierto una imagen hiperespectral. El *ground truth* que vamos a abrir debe coincidir en dimensiones con las de la imagen hiperespectral abierta, en caso contrario se nos mostrará un mensaje de error indicando que el archivo que estamos intentando abrir no se corresponde con la imagen hiperespectral.

Podemos acceder a la opción de abrir *ground truth* haciendo uso del icono de la figura A.17 de la barra de herramientas, accediendo a la opción "Abrir ground truth" del menú "Archivo" o usando la combinación de teclas "Ctrl+2".

Figura A.17: Icono de la barra de herramientas para abrir un ground truth

En la figura A.18 podemos ver el aspecto de la ventana que nos permite seleccionar un fichero *ground truth*. Tal y como ocurría al abrir una imagen hiperespectral, si seleccionamos un fichero con formato RAW tendremos que facilitar algunos datos sobre la imagen para que el programa pueda interpretarla. En la figura A.19 podemos observar que los datos que se solicitan se corresponden con los descritos en la sección A.2.2.

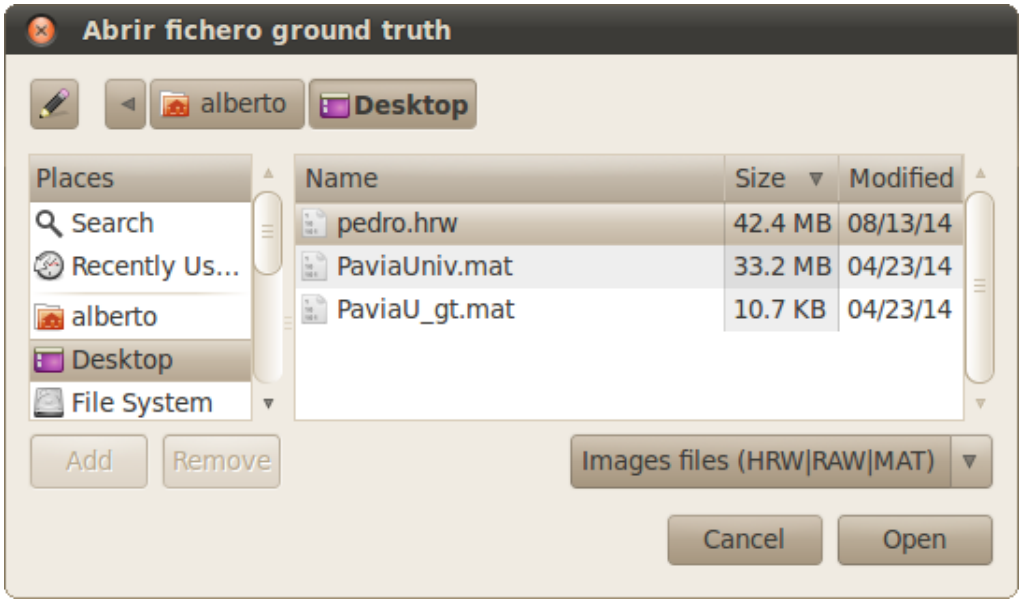

Figura A.18: Ventana para la selección de un ground truth

A.2.9. Información de los Plugins

Con esta opción, tal y como se muestra en la figura A.21, podemos consultar el nombre, la versión, el desarrollador y algunos comentarios de los plugins cargados en la aplicación. Como en los casos anteriores, hay tres formas de acceder a esta ventana, podemos utilizar el icono de la figura A.20 de la barra de herramientas, el atajo de teclado "Ctrl+I" o mediante la opción "Info plugins" del menú "Plugins".

A.2.10. Configurar el Proceso de Clasificación

El software desarrollado permite incluir plugins en distintas fases. Estas fases funcionan como una hoja de ruta, indicándole al programa en que orden debe ejecutar los distintos plugins. Para poder entender mejor como configurar el proceso de clasificación, vamos a explicar las distintas fases en las que se permiten incluir plugins:

Figura A.19: Formulario a cubrir para abrir un ground truth en formato RAW

Figura A.20: Icono de la barra de herramientas que permite mostrar información sobre los plugins disponibles

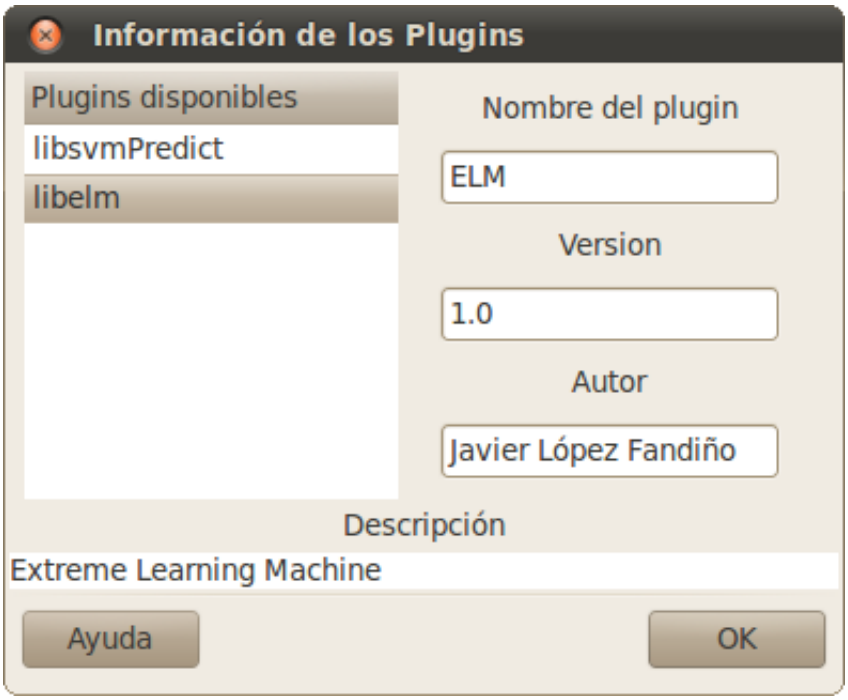

Figura A.21: Ventana con información de los plugins

Preprocesado: En esta fase se podrán incluir todos aquellos plugins que modifiquen de alguna manera la imagen hiperespectral, como por ejemplo, un plugin de eliminación de ruido.

Camino 1: Esta fase realiza la clasificación espectral de la imagen y podrá incluir todos aquellos plugins diseñados para esa tarea, permitiendo también incluir cualquier otro plugin que modifique la imagen hiperespectral, ya sea eliminado ruido, reduciendo bandas, etc.

Camino 2: En esta fase se realiza una clasificación espacial y deberá incluir plugins adecuados para esa tarea, además de permitir, al igual que en la fase "camino 1", incluir cualquier otro plugin que modifique la imagen hiperespectral.

Unión: Como su propio nombre indica, en esta fase se incluirá un único plugin encargado de combinar la información espectral proveniente de la fase "camino 1" con la información espacial obtenida de la fase "camino 2". Solo se hará uso de esta fase si incluimos plugins en la fase de "camino 1" y en la fase de "camino 2" al mismo tiempo.

Postprocesado: Esta fase puede incluir todos aquellos plugins que modifiquen el resultado obtenido de la fases anteriores.

En la figura A.22 podemos observar como existen 5 pestañas que se corresponden con las cinco fases en las que se pueden incluir plugins. Para cada una de las fases existen dos listas, una con los plugins disponibles y otra con los plugins seleccionados. Para pasar un plugin de la lista de disponibles a la lista de seleccionados, debemos marcar el plugin y pulsar el botón "Añadir". Una vez que el plugin está en la lista de plugins seleccionados podemos realizar las siguientes acciones:

Figura A.22: Ventana para la configuración del proceso de clasificación

- Subir: Permite incrementar la preferencia del plugin a la hora de ejecutar los plugins de esa fase.
- Bajar: Permite decrementar la preferencia del plugin a la hora de ejecutar los plugins de esa fase.
- Quitar: Elimina el plugin de la lista de plugins seleccionados.
- Parámetros: Permite configurar los parámetros del plugin según se puede ver en la figura A.23.

A.2.11. Ejecutar Clasificación

En la figura A.24 podemos ver la ventana de seguimiento del proceso de clasificación. Una vez que pulsemos el botón "Ejecutar" comienza el proceso de clasificación y se sustituyen los botones "Ejecutar" y "Cerrar" por el botón "Abortar". Este último botón nos permitirá abortar el proceso de clasificación. Según avance el proceso y se ejecuten los distintos plugins, la barra de progreso superior

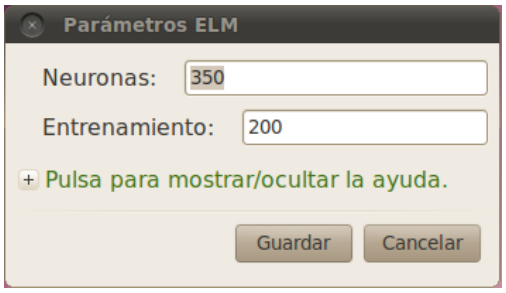

Figura A.23: Ventana para la configuración de los parámetros de un plugin

se irá rellenando. En la figura A.25 podemos ver el proceso de clasificación en funcionamiento. Al finalizar este proceso se nos ofrecerá la opción de guardar los resultados en un fichero mediante el botón "Guardar".

Figura A.24: Ventana de seguimiento del proceso de clasificación

Podemos utilizar la opción "Ejecutar clasificación" del menú "Clasificación", el atajo de teclado "Ctrl+E" o el icono de la figura A.26 de la barra de herramientas para acceder a esta opción.

En la figura A.27 podemos ver el resultado del proceso de clasificación.

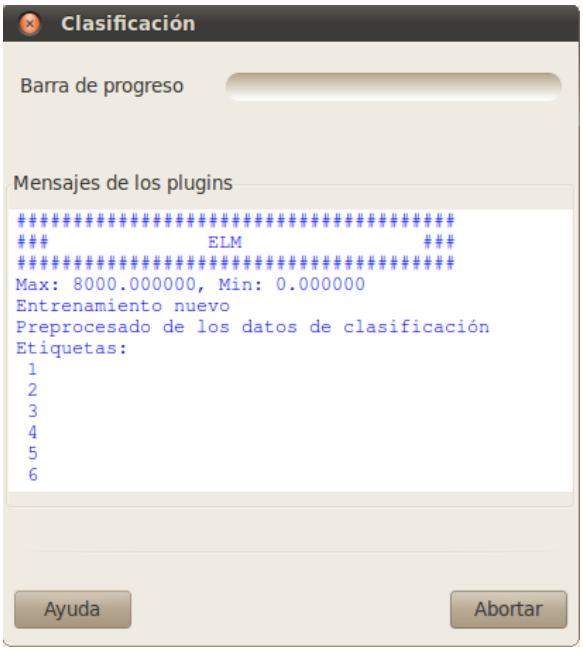

Figura A.25: Proceso de clasificación en funcionamiento

Figura A.26: Icono de la barra de herramientas que permite iniciar el proceso de clasificación

A.2.12. Abrir Clasificación

Para abrir una clasificación previamente almacenada podemos utilizar el icono de la barra de herramientas mostrado en la figura A.28, acceder a la opción "Abrir clasificación" del menú "Clasificación" o usar la combinación de teclas "Ctrl+A".

Figura A.27: Resultado del proceso de clasificación

Figura A.28: Icono de la barra de herramientas para abrir una clasificación

Apéndice B

Manuales Técnicos

Este manual está orientado principalmente a programadores. Contiene información acerca de algunos puntos importantes a tener en cuenta a la hora de desarrollar nuevas técnicas de clasificación para añadirlas como plugins de la aplicación.

B.1. Plugins

Para el manejo de los datos de la imagen hiperespectral y de su ground truth debemos utilizar la librería *global_data* que se suministra con la aplicación. Esta librería ofrece una amplia variedad de funciones que dan acceso a todos los datos de las imágenes, así como funciones para modificar esos datos.

Cada uno de los plugins que se desarrollen para ser utilizados con el software desarrollado en este proyecto deben incluir las siguientes funciones:

- **Función: ambito**
  - **Estructura:** guint32 ambito()
  - **Descripción:** Devuelve el ámbito del plugin, es decir, dentro de la configuración del proceso de clasificación, en cual de las cinco fases disponibles (preprocesado, camino1, camino2, unión, postprocesado) puede estar disponible el plugin. Para indicar el ámbito debemos tener en cuenta que cada una de las fases anteriormente enumeradas tienen un número primo como referencia, así, la fase de preprocesado se corresponde con el 2, la de camino1 con el 3, la de camino2 con el 5, la de unión con el 7 y la de postprocesado con el 11. Una vez identificado el/los ámbito/s que vamos a utilizar, solo nos resta calcular el ámbito total multiplicando sus primos correspondientes. El resultado de esta multiplicación debe ser el valor que retorna la función. Por

ejemplo, si nuestro plugin estará disponible en la fase de camino1 y camino2, la función ambito devolverá el valor 15 (3*5).

- **Parámetros:** Sin parámetros.
- **Devuelve:** El ámbito del plugin.

- **Función: num_param**
  - **Estructura:** guint32 num_param()
  - **Descripción:** Devuelve el número de parámetros que necesita la función.
  - **Parámetros:** Sin parámetros.
  - **Devuelve:** El número de parámetros que necesita la función.

- **Función: parametro_data**
  - **Estructura:** guint8 parametro_data(guint8 param, guchar *data)
  - **Descripción:** Devuelve el valor por defecto de cada parámetro en formato texto.
  - **Parámetros:**
    - **param:** Numero del parámetro por el que se pregunta.
    - **data:** Valor por defecto en formato texto del parámetro indicado.
  - **Devuelve:** 0 en caso de éxito, 1 en caso de fallo.

- **Función: parametros**
  - **Estructura:** guint32 parametros(guint8 param)
  - **Descripción:** Devuelve el tamaño del parámetro indicado. Comienza en 0 el número de parámetro. Se debe tener en cuenta que el tamaño no incluye el indicador de fin de cadena, por lo tanto a la hora de reservar memoria para almacenar los datos, debemos añadir 1 byte más para almacenar el final de cadena.
  - **Parámetros:**
    - **param:** Numero del parámetro por el que se pregunta.
  - **Devuelve:** El tamaño del parámetro indicado.

- **Función: tamano_total_param**
  - **Estructura:** guint32 tamano_total_param()
  - **Descripción:** Devuelve el tamaño total de los parámetros.
  - **Parámetros:**

  - **Devuelve:** El tamaño total de los parámetros.

- **Función: plugin_name**
  - **Estructura:** gboolean plugin_name(gchar *texto)
  - **Descripción:** Devuelve el nombre del plugin.
  - **Parámetros:**
    - **texto:** Variable en la que se retorna el nombre del plugin.
  - **Devuelve:** TRUE.

- **Función: plugin_author**
  - **Estructura:** gboolean plugin_author(gchar *texto)
  - **Descripción:** Devuelve el nombre del desarrollador del plugin.
  - **Parámetros:**
    - **texto:** Variable en la que se retorna el nombre del desarrollador del plugin.
  - **Devuelve:** TRUE.

- **Función: plugin_version**
  - **Estructura:** gboolean plugin_version(gchar *texto)
  - **Descripción:** Devuelve la versión del plugin.
  - **Parámetros:**
    - **texto:** Variable en la que se retorna la versión del plugin.
  - **Devuelve:** TRUE.

- **Función: plugin_description**
  - **Estructura:** gboolean plugin_description(gchar *texto)
  - **Descripción:** Devuelve una descripción del plugin.
  - **Parámetros:**
    - **texto:** Variable en la que se retorna la descripción del plugin.
  - **Devuelve:** TRUE.

- **Función: run**
  - **Estructura:** guint32 run(image_struct *image, image_struct *ground_truth, guint32 **clasificacion, guint32 **segmentacion, guchar *param);
  - **Descripción:** Función principal del plugin. Es la función que se debe llamar para que se ejecute el plugin.

- **Parámetros:**
  - **image:** Estructura que contiene los datos de la imagen hiperespectral.
  - **ground_truth:** Estructura que contiene los datos del ground truth.
  - **clasificacion:** Almacena el resultado de la clasificación.
  - **segmentacion:** Almacena el resultado de la segmentación.
  - **param:** Contiene los parámetros que se pasan al plugin.
- **Devuelve:** TRUE.

B.2. Parámetros de los Plugins

Todos los plugins deben ir acompañados de un fichero generado por el programa *Glade Interface Designer*[1] y con el mismo nombre que el plugin al que está asociado, seguido de la extensión "glade". En el código fuente de la aplicación pueden encontrarse varios ejemplos de este fichero, correspondientes a los plugins desarrollados en este proyecto. Este archivo permite a la aplicación recoger los parámetros que cada uno de los plugins crea conveniente. A continuación se detallarán algunos de los aspectos a tener en cuenta a la hora de construir este documento.

En la parte central de la figura B.1 podemos ver la apariencia de la ventana que solicitará los parámetros al usuario. En esta ventana se pueden apreciar dos cajas de texto y dos botones, así como un desplegable para la ayuda. Podemos incluir tantas cajas de texto como sea necesario, pero todas deben seguir una nomenclatura a la hora de configurar el nombre de cada una ellas, esto es, deben comenzar por "txt_" y continuar con el número de caja correspondiente, empezando por el 0 y continuando con 1, después el 2, etc. En la imagen podemos apreciar como el nombre de la primera caja de texto es "txt_0".

Para el desplegable de ayuda, debemos utilizar un búfer de texto tal y como se observa en la figura B.2, modificando la propiedad "text" en la que se incluirá aquel texto que consideremos oportuno para ayudar al usuario.

B.3. Interfaz Plugins-Aplicación

Todos los plugins que quieran mostrar información al usuario durante su ejecución deben utilizar el fichero de cabecera "interface_plugins.h" que se puede

[1] https://glade.gnome.org/

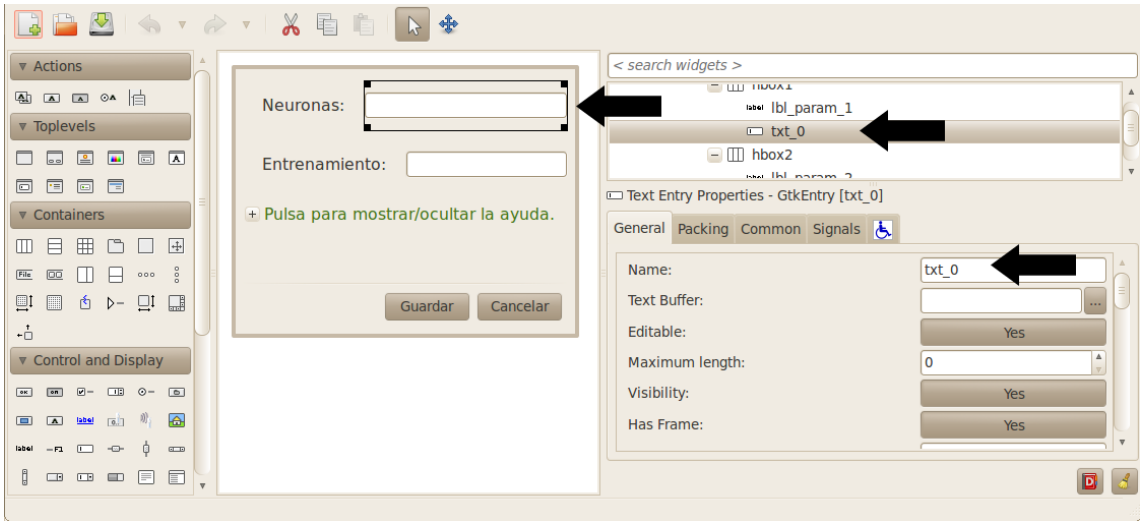

Figura B.1: Construcción de la ventana de captura de parámetros I.

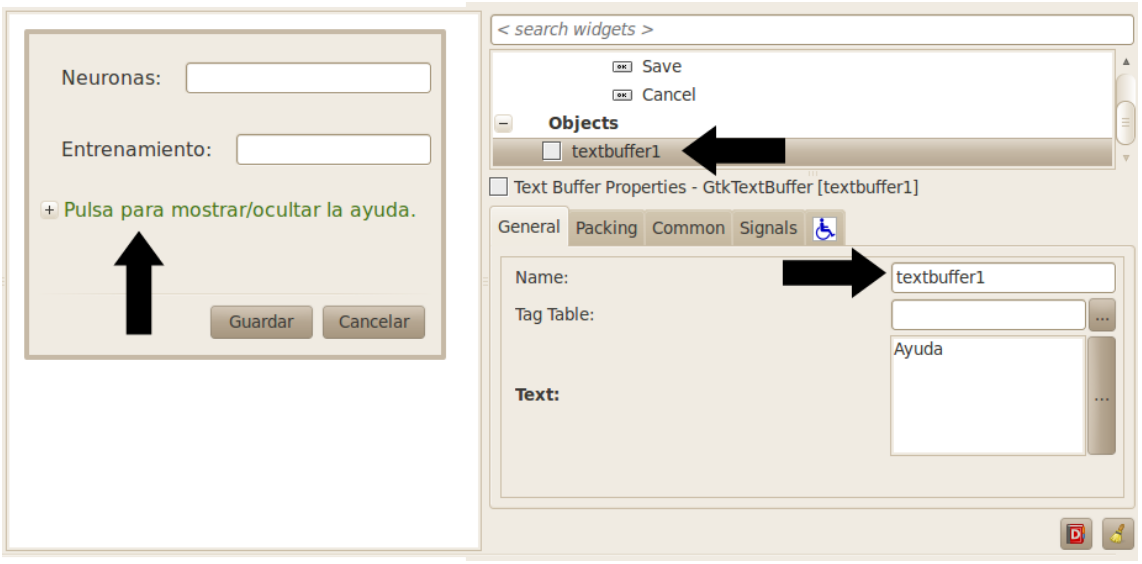

Figura B.2: Construcción de la ventana de captura de parámetros II.

encontrar en el código fuente de la aplicación y que incorpora la siguiente función:

- **Función: imprimir_mensaje**
  - **Estructura:** void imprimir_mensaje(char *mensaje);
  - **Descripción:** Permite al plugin comunicarse con la aplicación para mostrar un mensaje en la ventana de ejecución del proceso de clasificación.
  - **Parámetros:**
    - **mensaje:** Mensaje que se quiere mostrar.

Apéndice C

Metodologías Ágiles

C.1. Métodos Ágiles

El presente apéndice pretende ser una ayuda para aquellos lectores que no estén familiarizados con la metodología ágil y más concretamente con *Scrum*, complementando la información de la sección 3.1.1.

C.1.1. Introducción a los Métodos Ágiles

Cualquiera que alguna vez haya sido responsable de dirigir un proyecto de desarrollo de software sabe que no es nada fácil coordinar y negociar exitosamente con las partes implicadas en el proyecto.

Los problemas más frecuentes para el fracaso de un proyecto de desarrollo software son:

- No se adaptan a los cambios.
- Calidad insuficiente y muy variable.
- Proyectos que exceden sus tiempos y costos.

En base a estos problemas se ha llegado a un acuerdo de los que significa un proyecto de software exitoso, en tres dimensiones:

- A tiempo.
- En presupuesto.
- Cumpliendo el alcance definido.

Está claro que una forma de resolver los problemas de calidad, y otros, de un producto es mejorando la forma de construir tales productos. Y eso en el desarrollo de software es casi equivalente a mejorar la metodología que se sigue para construir los productos.

Durante el transcurso de los años 90, debido al ambiente cambiante surgió la necesidad de desarrollar metodologías livianas y maniobrables que pudieran operar en ese ambiente cambiante. Estas metodologías son conocidas hoy en día como "metodologías ágiles".

C.1.2. Qué es una Metodología Ágil?

Lo ágil se define (por los mismos agilistas) como la habilidad de responder de forma versátil al cambio para maximizar los beneficios. Las metodologías ágiles varían en su forma de responder al cambio, pero en general comparten las siguientes características:

- Los individuos y sus interacciones son más importantes que los procesos y las herramientas.
- El software que funciona es más importante que la documentación exhaustiva.
- La colaboración con el cliente en lugar de la negociación de contratos.
- La respuesta al cambio en lugar de aferrarse a un plan.

Los valores y principios compartidos por toda la metodología ágil fueron enunciados en el "manifiesto ágil", por la "alianza ágil".

C.1.3. La Alianza Ágil

En una reunión celebrada en febrero de 2001 en Utha (EEUU), nace el término "ágil" aplicado al desarrollo de software. En esta reunión participaron un grupo de 17 expertos de la industria del software. Su objetivo fue esbozar los valores y principios que debían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que pueden surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.

Tras esta reunión se creó *The Alliance* [1], una organización dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que los adopten. El punto de partida fue el Manifiesto Ágil [2], un documento que resume la filosofía ágil.

C.1.4. El Manifiesto Ágil

El Manifiesto comienza enumerando los principales valores del desarrollo ágil. Se valora:

- Al individuo y a las interacciones del equipo de desarrollo sobre el proceso y las herramientas.
- Desarrollar software que funciona más que conseguir una buena documentación.
- La colaboración con el cliente más que la negociación de un contrato.
- Responder a los cambios más que seguir estrictamente un plan.

Los valores anteriores inspiran los doce principios del manifiesto:

1. *La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporten un valor.* Un proceso es ágil si a las pocas semanas de empezar ya entrega software que funcione aunque sea rudimentario. El cliente decide si pone en marcha dicho software con la funcionalidad que ahora le proporciona o simplemente lo revisa e informa de posibles cambios a realizar.

2. *Bienvenidos los requisitos cambiantes. Se capturan los cambios para que el cliente tenga una ventaja competitiva.* Los cambios en los registros deben verse como algo positivo. Les va a permitir aprender más, a la vez que logran una mayor satisfacción del cliente. Este principio implica además que la estructura del software debe ser flexible para poder incorporar los cambios sin demasiado coste agregado.

3. *Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.* Las entregas al cliente se insiste en que sean software, no planificaciones, ni documentación de análisis o de diseño.

[1] http://www.agilealliance.org/
[2] http://www.agilealliance.org/the-alliance/the-agile-manifesto/

4. *La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.* El proceso de desarrollo necesita ser guiado por el cliente, por lo que la interacción con el equipo es muy frecuente.

5. *Construir el proyecto en torno a individuos motivados.* Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.

6. *El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.* Los miembros de equipo deben hablar entre ellos, éste es el principal modo de comunicación. Se pueden crear documentos pero no todo estará en ellos.

7. *El software que funciona es la medida principal de progreso.* El estado de un proyecto no viene dado por la documentación generada o la fase en la que se encuentre, sino por el código generado y el funcionamiento. Por ejemplo, un proyecto se encuentra al 50 % si el 50 % de los requisitos ya están en funcionamiento.

8. *Los procesos ágiles promueven un desarrollo sostenible.* Los desarrolladores y usuarios deberían ser capaces de mantener el ritmo de desarrollo durante toda la ejecución del proyecto, asegurando en todo momento que la calidad es máxima.

9. *La atención continua a la calidad técnica y al buen diseño mejora la agilidad.* Producir código claro y robusto es la clave para avanzar más rápidamente en el proyecto.

10. *Simplicidad - el arte de maximizar la cantidad de trabajo no realizado - es esencial.* Tomar los caminos más simples que sean consistentes con los objetivos perseguidos. Si el código producido es simple y de alta calidad será más sencillo adaptarlo a los cambios que puedan surgir.

11. *Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por si mismos.* Todo el equipo es informado de las responsabilidades y éstas recaen sobre todos sus miembros. Es el propio equipo el que decide la mejor forma de organizarse, de acuerdo a los objetivos que se persigan.

12. *En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.* Puesto que el entorno está cambiando continuamente, el equipo también debe ajustarse al nuevo escenario de forma continua. Puede cambiar su organización, sus reglas, sus convenciones, sus relaciones, etc, para seguir siendo ágil.

C.1.5. Metodologías Ágiles vs. Tradicionales

A continuación vamos a enumerar las principales diferencias respecto de las metodologías tradicionales (no ágiles).

En la tabla C.1 se recoge esquemáticamente estas diferencias que no se refieren sólo al proceso en sí, sino también al contexto de equipo y organización que es más favorable a cada una de estas filosofías de desarrollo de software.

Metodologías Ágiles	Metodologías Tradicionales
La planificación del trabajo sólo comprende el ciclo en el que se está trabajando.	Trabajo y gestión guiada por un plan general del proyecto que comprende todo su ciclo de desarrollo.
Descubrimiento progresivo de requisitos, e incorporación de cambios en cualquier iteración del desarrollo.	Conocimiento detallado de los requisitos antes de comenzar el diseño del proyecto.
"Refactorización" de código como modelo de trabajo compatible con el punto anterior.	"Hacerlo bien a la primera". Evitar la re-codificación y el re-trabajo que supone una pérdida de eficiencia.
Comunicación directa entre los integrantes del equipo (incluidos cliente y usuarios) prefiriendo la verbal directa.	Comunicación formal según el plan de comunicación del proyecto.
Equipos auto-gestionados.	Gestión de equipos y personas centralizada en el gestor del proyecto.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
Grupos pequeños (hasta 20 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos roles.	Más roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Tabla C.1: Diferencias entre metodologías ágiles y no ágiles.

Las metodologías ágiles están revolucionando la manera de producir software, y a la vez generando un amplio debate entre sus adeptos y quienes por prejuicio no las ven como alternativa a las metodologías tradicionales.

Aunque los creadores de las metodologías ágiles han suscrito el manifiesto ágil, y todas ellas coinciden con los principios enunciados anteriormente, cada una tiene características propias y hace hincapié en algunos aspectos más específicos.

Las metodologías ágiles más populares son: XP (*Extreme Programming* – Programación Extrema), *Cristal* y *Scrum*. Por ser esta última la elegida para desarrollar el presente trabajo mencionamos los tres principales motivos que nos llevaron a su elección:

- Sirve para gestionar proyectos de cualquier tipo, no solamente tecnológicos.
- Deja un vacío en la parte de definiciones del área de ingeniería lo que permite una elaboración propia para completarla.
- Tiene escasa documentación por lo que se piensa que este trabajo puede resultar en un aporte significativo, debido a la inmensa cantidad de tiempo que se tiene que dedicar a escribir código.

C.2. Scrum

C.2.1. Introducción

Scrum es una metodología ágil de gestión de proyectos cuyo objetivo primordial es elevar al máximo la productividad de un equipo. Reduce al máximo la burocracia y actividades no orientadas a producir software que funcione y produce resultados en periodos muy breves de tiempo. Como método, *Scrum* enfatiza valores y prácticas de gestión, sin pronunciarse sobre requerimientos, prácticas de desarrollo, implementación y demás cuestiones técnicas. Más bien delega completamente en el equipo la responsabilidad de decidir la mejor manera de trabajar para ser lo más productivos posibles.

La palabra *Scrum* procede de la terminología del juego de rugby, donde designa al acto de preparar el avance del equipo en unidad pasando la pelota a uno y otro jugador. Igual que el juego, *Scrum* es adaptable, ágil, auto-organizante y con pocos tiempos muertos.

Scrum fue desarrollado por Jeff Sutherland y elaborado más formalmente por Ken Schwaber. Poco después Sutherland y Schwaber se unieron para refinar y extender *Scrum*. Se la ha llegado a conocer como una herramienta de hiperproductividad. Schwaber se dio cuenta entonces de que un proceso necesita aceptar el cambio, en lugar de esperar predictibilidad. Se enfoca en el hecho de que procesos

definidos y repetibles sólo funcionan para atacar problemas definidos y repetibles con gente definida y repetible en ambientes definidos y repetibles. Toma el cambio como una forma de entregar al final del desarrollo algo más cercano a la verdadera necesidad del Cliente. Puede ser aplicado teóricamente a cualquier contexto en donde un grupo de gente necesita trabajar junta para lograr una meta común.

Se basa en los principios ágiles:

- Privilegiar el valor de la gente sobre el valor de los procesos.
- Entregar software funcional lo más pronto posible.
- Predisposición y respuesta al cambio.
- Fortalecer la comunicación y la colaboración.
- Comunicación verbal directa entre los implicados en el proyecto.
- Simplicidad; supresión de artefactos innecesarios en la gestión del proyecto.

C.2.2. Elementos de Scrum

Roles

La dimensión del equipo total de *Scrum* no debería ser superior a veinte personas. Si hay más, lo más recomendable es formar varios equipos.

Scrum tiene una estructura muy simple. Todas las responsabilidades del proyecto se reparten en 3 roles:

- **Product Owner**
  Representa a todos los interesados en el producto final. Sus áreas de responsabilidad son:
  - Financiación del proyecto.
  - Requisitos del sistema.
  - Retorno de la inversión del proyecto.
  - Lanzamiento del proyecto.

  Es el responsable oficial del proyecto, gestión, control y visibilidad de la lista de acumulación o lista de retraso del producto (*Product Backlog*). Toma las decisiones finales de las tareas asignadas al registro y convierte sus elementos en rasgos a desarrollar.

- **Scrum Master (Líder del proyecto).**
  Responsable del proceso *Scrum*, de cumplir la meta y resolver los problemas. Así como también, de asegurarse que el proyecto se lleve a cabo de acuerdo con las prácticas, valores y reglas de *Scrum* y que progrese según lo previsto.

  Interactúa con el cliente y el equipo. Coordina los encuentros diarios, y se encarga de eliminar eventuales obstáculos. Debe ser miembro del equipo y trabajar a la par.

- **Team (Equipo).**
  Responsable de transformar el *Backlog* de la iteración en un incremento de la funcionalidad del software. Tiene autoridad para reorganizarse. Esa auto-gestionado, auto-organizado y multi-funcional.

Bibliografía

[1] Qian Du, Liangpei Zhang, Bing Zhang, Xiaohua Tong, Peijun Du, Jocelyn Chanussot, "Foreword to the Special Issue on Hyperspectral Remote Sensing: Theory, Methods and Applications", *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol.6, no 2, Abril 2013.

[2] Teledetección Terrestre. Web Universidad de Jaén(*http://www.ujaen.es/huesped/pidoceps/telap/Inves_tel/terrestre.htm*). Consultada el 3 de Agosto de 2014.

[3] Dora B.Heras, Francisco Argüello, and Pablo Quesada, "Exploring ELM-based spatial-spectral classification of hyperspectral images", *International Journal of Remote Sensing*, vol.35, no 2, pp. 401-423, Junio 2013.

[4] Giorgos Mountrakis, Jungho Im and Caesar Ogole, "Support vector machine in remote sensing: A review", *ISPRS Journal of Photogrammetry and Remote Sensing*, no 66, pp. 247-259, Diciembre 2010

[5] Chih-Chung Chang, Chin-Jen Lin. Librería SVM (libsvm). Disponible [21 Mayo 2014] en http://www.csie.ntu.edu.tw/ cjlin/libsvm/

[6] Y. Tarabalka, J.Chanussot, J.A. Benediktsson, "Segmentation and clasification of hyperspectral images using watershed transformation", *Pattern Recognition*, vol. 43, pp. 2367-2379, Enero 2010.

[7] Mathieu Fauvel, Yuliya Tarabalka, Jón Atli Benediktsson, Jocelyn Chanussot and James C. Tilton, "Advances in Spectral-Spatial Clasificaction of Hyperspectral Images", *Proceedings of the IEEE*, vol. 101, pp. 652 - 675, Marzo 2013.

[8] Brian Fulkerson, Stefano Soatto, "Really Quick Shift: Image Segmentation on a GPU", *Lecture Notes in Computer Science* vol. 6554, pp 350-358, 2012.

[9] ROSIS. 2013. "University of Pavia dataset". *http://www.ehu.es/ccwintco/uploads/e/ee/PaviaU.mat* (imagen hiperespectral) y *http://www.ehu.es/ccwintco/uploads/5/50/PaviaU_gt.mat* (ground truth). Consultada el 15 Febrero de 2014.

[10] AVIRIS. 2013. "Indian Pine dataset". *http://www.ehu.es/ccwintco/uploads/2/22/Indian_pines.mat* (imagen hiperespectral) y *http://www.ehu.es/ccwintco/uploads/c/c4/Indian_pines_gt.mat* (ground truth). Consultada el 15 de Febrero de 2014.

[11] Y. Tarabalka, J. A. Benediktsson, and J. Chanussot, "Spectral-spatial classification of hyperspectral imagery based on partitional clustering techniques", *IEEE Trans. Geosci.Remote Sens.*, vol. 47, no. 9, pp. 2973–2987, Septiembre 2009.

[12] A. Ferran, S. Bernabe, P. G. Rodriguez and A. Plaza. "A Web-Based System for Classification of Remote Sensing Data". *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6, no. 4, pp. 1934-1948, Agosto 2013.

[13] L. I. Jimenez, G. Martin and A. Plaza. "HyperMix: A New Tool for Quantitative Evaluation of Endmember Identification and Spectral Unmixing Techniques", *IEEE Geoscience and Remote Sensing Symposium (IGARSS'12)*, Munich, Germany, 2012.

[14] Teledetección. Web Instituto Geográfico Nacional(*http://www.ign.es/ign/layoutIn/teledeteccionQueEs.do*). Consultada el 7 de Julio de 2014.

[15] D. Landgrebe, "Hyperspectral Image Data Analysis", *IEEE Signal Processing Magazine*, vol. 19, no. 1, pp. 17-28, Enero 2002.

[16] G. Shaw, D. Manolakis, "Signal processing for hyperspectral image exploitation", *IEEE Signal Processing Magazine*, vol. 19, pp. 12-16, Enero 2002.

[17] C. Chang, *Hyperspectral Imaging. Techniques for Spectral Detection and Classification.* Norwell, MA: Kluwer, 2003.

[18] Clark, R.N., *Spectroscopy of Rocks and Minerals, and Principles of Spectroscopy*, John Wiley and Sons, New York, 1999.

[19] M. Fauvel, J. Chanussot, and J. A. Benediktsson, "Evaluation of kernels for multiclass classification of hyperspectral remote sensing data", *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, May 2006, vol. 2, p. II, Mayo 2006.

[20] F. Melgani and L. Bruzzone, "Classification of hyperspectral remote sensing images with support vector machines", *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 8, pp. 1778–1790, Agosto 2004.

[21] S. Tadjudin and D. A. Landgrebe, "Classification of high dimensional data with limited training samples", *ECE Technical Reports*, paper 56, Enero 1998

[22] *Digital Image Processing* R. C. Gonzalez and R. E. Woods, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2002

[23] Frery, A. C., de Araujo, C. C., Alice, H., Cerqueira, J., Loureiro, J.A., de Lima, M.E., and Horta, M. M. (2003, September). "Hyperspectral images clustering on reconfigurable hardware using the k-means algorithm". *In Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings. 16th Symposium on* (pp. 99-104), Septiembre 2003.

[24] Benediktsson, J. A., and Kanellopoulos, I. . "Classification of multisource and hyperspectral data based on decision fusion". *Geoscience and Remote Sensing, IEEE Transactions* vol. 37(3), pp. 1367-1377, Mayo 1999.

[25] *Guía de los Fundamentos de la Dirección de Proyectos (Guía del PMBOK)*, pp. 19-22, 3ª edición, Project Management Institute, Four Campus Boulevard, Newtown Square, 2004.

[26] *Guía de los Fundamentos de la Dirección de Proyectos (Guía del PMBOK)*, pp. 249-238, 3ª edición, Project Management Institute, Four Campus Boulevard, Newtown Square, 2004.

[27] *Guía de los Fundamentos de la Dirección de Proyectos (Guía del PMBOK)*, pp. 237-264, 3ª edición, Project Management Institute, Four Campus Boulevard, Newtown Square, 2004.

[28] INTECO, *Guía Práctica de Gestión de Configuración*, 2008

[29] Beneficios de Scrum. Web proyectos ágiles (*http://www.proyectosagiles.org /beneficios-de-scrum*). Consultada el 3 de Junio del 2014.

[30] Sommerville, Ivan, *Ingeniería de Software*, pp.72-77 9ª edición, Pearson, Mexico, 2011.

[31] Acerca de la IEEE. Web de la IEEE(*http://www.ieee.org/about/index.html*). Consultada el 15 de Junio de 2014.

[32] Comaniciu, D. and Meer, P. "Mean shift analysis and applications", *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference* vol. 02, pp. 1197-1203, Septiembre 1999

[33] *Guía de los Fundamentos de la Dirección de Proyectos (Guía del PMBOK)*, pp. 101, 4ª edición, Project Management Institute, Four Campus Boulevard, Newtown Square, 2008.

[34] Salarios. Web InfoJobs(*http://www.plandecarrera.infojobs.net*). Consultada el 17 de Junio del 2014.

[35] Bases y tipos de cotización 2014. Web de la Seguridad Social(*http://www.segsocial.es/Internet_1/Trabajadores/CotizacionRecaudaci10777/Basesytiposdecotiza36537/index.htm*). Consultada el 30 de Julio del 2014.

Metodología Ágil

Introducción a los métodos de desarrollo de software. *http://ocw.um.es/ingenierias/fundamentos-de-ingenieria-del-software/material-de-clase-1/capitulo08.pdf*. Consultado el 3 de Marzo 2014.

Manifesto for Agile Software Development. *http://www.agilemanifesto.org* Consultada el 3 de Marzo de 2014.

Agile-Spain. *http://www.agile-spain.org*. Consultada el 8 de Marzo de 2014.

Megapolis.net. *http://www.navegapolis.net/content/blogcategory/83/62/*. Consultada el 9 de Marzo de 2014.