ALGORITMOS COMPLEJOS PARA LA REDUCCIÓN DE RUIDO EN DATOS HIPERESPECTRALES DE IMÁGENES CON BAJO SNR

AUTOR: SERGIO SORIA VILLALVA

TUTOR: DANIELE CERRA

MÁSTER EN TECNOLOGÍAS DE LA INFORMACIÓN GEOGRÁFICA

TRABAJO DE FIN DE MÁSTER

SEPTIEMBRE 2013

ÍNDICE

1. INTRODUCCIÓN

El presente trabajo se centra en el limpiado de ruido en imágenes hiperespectrales. Actualmente existe una gran variedad de métodos que abordan este problema de diversas maneras, entre los que destacan, por ser los más extendidos, el filtrado de las imágenes mediante la utilización de filtros de paso bajo como la media. Sin embargo, como es sabido esta técnica presenta ciertos inconvenientes tanto a la hora de mantener los valores originales de la imagen como a la de mantener su resolución espacial original. Esto resulta bastante inconveniente en imágenes hiperespectrales, ya que éstas ya suelen presentar inicialmente bajas resoluciones espaciales, además, la conservación de los valores originales de la imagen es imprescindible para un gran número de aplicaciones, entre las que caben citar aplicaciones acuáticas, por requerir gran precisión en sus datos.

En el *Deutsches Zentrum für Luft- und Raumfahrt e.V*. -o DLR en adelante- se ha desarrollado un nuevo procedimiento de limpieza del ruido -o *denoising*-, basado en el *spectral unmixing*, un conjunto de técnicas propias de la teledetección hiperespectral empleadas para la determinación de la composición interna de cada píxel. Hasta ahora, el spectral unmixing y el limpiado de ruido se han considerado problemas totalmente independientes dentro del tratamiento digital de imágenes hiperespectrales, sin embargo, con el método desarrollado por el DLR, se demuestra que el spectral unmixing puede llegar a aportador buenos resultados para reducir el ruido incluso en imágenes muy corrompidas con ratios de señal/ruido -*Signal to Noise Ratio*, o SNR- muy bajos.

A pesar de esto, aunque esta técnica, llamada *Unmixing-based denoising* -UBD en adelante-, presenta grandes ventajas frente a otra serie de técnicas más extendidas, también muestra algunos problemas que han de solucionarse antes de generalizar el procedimiento. Para ello, en el presente trabajo se ha pretendido abordarlos e intentar solucionarlos, de modo que se ha complementado el algoritmo inicial desarrollando unos procedimientos alrededor de éste que los automatizan y lo implementan para tratar de resolver algunas de sus carencias más relevantes. Finalmente, el algoritmo desarrollado se ha probado con datos reales, llegando a obtener resultados bastante positivos.

2. ANTECEDENTES TEÓRICOS

2.1. IMÁGENES HIPERESPECTRALES

A diferencia de la teledetección multiespectral, la teledetección hiperespectral se caracteriza por trabajar con imágenes de la superficie terrestre que presentan un gran número de bandas a lo largo del espectro electromagnético. Con ello se consigue que cada uno de los píxeles de una imagen hiperespectral represente un espectro continuo, al que se denomina firma espectral. Cada material presenta una firma espectral propia -de ahí el término- que lo hace único y diferenciable del resto. Este hecho se aprovecha para identificar los materiales presentes en la superficie, así como su distribución dentro de una imagen. La siguiente figura ilustra este concepto:

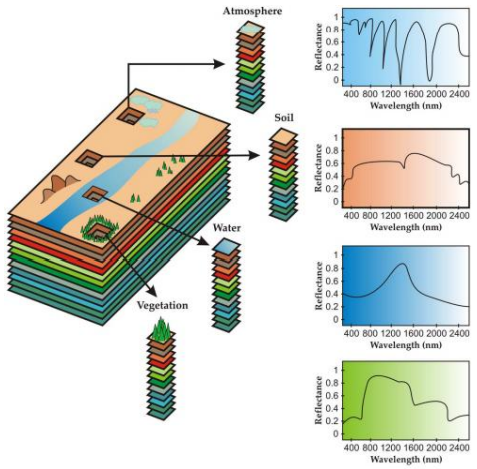

Figura I: Cada píxel presenta una firma espectral, característica de cada material.

El gran número de bandas que presentan estas imágenes se consigue gracias a que estos sensores presentan una alta resolución espectral, captando cada una de las bandas que los componen no más que un estrecho margen de la longitud de onda de la radiación que lo alcanza. Esto implica que la cantidad de radiación electromagnética -o lo que es lo mismo, el número de fotones- que entra en cada una de las bandas del sensor sea muy baja. Para solucionar esto, lo que se suele hacer es aumentar el tamaño del píxel, lo que conlleva una disminución de la resolución espacial de la imagen, pero que a la vez hace que en el sensor puedan entrar más fotones, al provenir éstos de un área mayor. Otra consecuencia de este hecho es que, además, estos sensores suelen ser aeroportados, más que transportados por satélites en órbita, precisamente para compensar su baja resolución espacial.

Por otro lado, una de las principales implicaciones prácticas de un tamaño de píxel grande es que la probabilidad de que dentro de cada uno esté presente un único material es muy baja. En muchos casos, dentro de la superficie cubierta por cada píxel hay presente más de un único material, con lo que el espectro que corresponde a ese píxel procederá de los distintos materiales presentes en el interior de cada píxel, resultando de la combinación de las firmas espectrales de tales materiales.

Este último aspecto resulta clave para entender el presente trabajo ya que el UBD, la técnica desarrollada por el DLR, se basa en los procedimientos que se emplean para determinar la composición interna de cada píxel, un conjunto de técnicas que, como se indicó en la introducción, se denominan *spectral unmixing* y permiten determinar de qué componentes y en qué proporción se compone la radiación proveniente de cada píxel.

2.2. SPECTRAL UNMIXING

Por lo tanto, en una imagen hiperespectral el espectro de cada píxel suele típicamente consistir en una combinación de los espectros puros de los materiales presentes en ese píxel. Estos espectros puros son los llamados *Endmembers* -en adelante se hará referencia a éstos con uno u otro nombre indiferentemente-. El *spectral unmixing*, por tanto, se encarga de determinar cuáles son los *Endmembers* que hay dentro de cada píxel, para lo que toma la imagen en su conjunto y considera cada píxel como una ecuación de la que su espectro es conocido -los sucesivos valores de cada banda en ese píxel- y cuyas incógnitas son las proporciones de los materiales presentes en la imagen de los que este espectro se compone. Estas proporciones vienen determinadas por la fracción del área que ocupa cada uno de ellos. La técnica del UBD asume que estas combinaciones de espectros son lineales -esto es, combinaciones ponderadas y directas de los diferentes *Endmembers*- que la suma de las proporciones que cada *Endmember* ocupa en cada píxel resulta en 1, esto es, en el total de la superficie de cada píxel, y que un *Endmember* no puede ocupar un área negativa. De este modo es posible presentar un sistema de ecuaciones con tantas ecuaciones como

píxeles componen la imagen, lo que resulta en un gran número de ecuaciones que en cualquier caso es mucho mayor que el número de incógnitas del sistema, que son las proporciones ocupadas en cada píxel por cada *Endmember*. Esta clase de sistemas se puede resolver mediante el ajuste de mínimos cuadrados, que proporcionará la solución óptima, o lo que es lo mismo, la combinación de *Endmembers* que explica de mejor manera el espectro de cada uno de los píxeles de la imagen. La siguiente figura muestra esta idea de forma sintética, donde se observa que a partir de la combinación lineal de los materiales α_1, α_2 y α_3, ponderados por sus superficies relativas en el píxel m_1, m_2 y m_3 se obtiene el espectro del píxel en cuestión *y*.

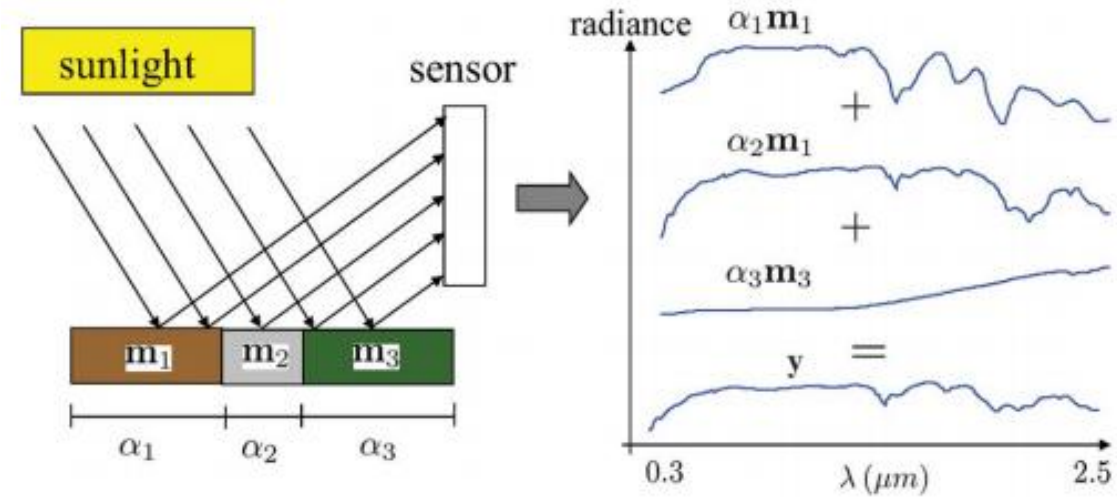

Figura II: Fundamento del spectral unmixing: $y = \alpha_1 m_1 + \alpha_2 m_2 + \alpha_3 m_3$.

En una situación ideal, cada uno de los píxeles de la imagen consistirá, por tanto, en una combinación de los espectros puros de los materiales de referencia presentes en la imagen. Sin embargo, uno de los requerimientos, y a la vez, uno de los principales inconvenientes a la hora de determinar la composición de cada píxel es que el método de aproximación por mínimos cuadrados requiere conocer perfectamente cuáles son estos *Endmembers*, lo que en muchos casos no es posible, o resulta difícil. Existe, no obstante, una gran variedad de técnicas para la obtención de *Endmembers* en una imagen (*Nirmal Keshava, 2003*) que se pueden utilizar previamente a la realización del *spectral unmixing*.

En la realidad, cada píxel además de estar constituido por esta combinación de espectros de los materiales, también presenta un componente de error que hay que considerar, ya que puede llegar a tener un gran peso en relación los valores de la imagen. Tal es el caso de las imágenes que presentan bandas con bajo SNR, es decir, imágenes con mucho ruido en relación a la información que contienen, que hacen estas bandas inservibles en muchos casos, y que se pretenden tratar con el nuevo método desarrollado. En este componente de error cabría englobar toda la información presente en la imagen que no puede ser explicada como combinación de los propios materiales, y que constituye normalmente lo que se denomina ruido proveniente de interferencias atmosféricas, inducido por el propio sensor o por otro tipo de errores originados por el modelo de reconstrucción.

El producto del *spectral unmixing* consiste en un conjunto de tantas imágenes como *Endmembers* presentes en la imagen, mostrando cada una de ellas la abundancia de cada material en cada uno de los píxeles, o lo que es lo mismo, la distribución de los materiales por toda la imagen. Además, de esto, también se obtiene un vector residual en el que se concentra toda la información que no ha podido ser explicada como una combinación de los espectros puros.

2.3. UBD

El UBD es una técnica basada en los conceptos previos que hace uso del *spectral unmixing* para la reducción del ruido en una imagen hiperespectral. Éste es un enfoque nuevo ya que tradicionalmente en teledetección hiperespectral ambos conjuntos de técnicas han sido utilizados para solucionar problemas diferentes que no tenían una relación directa entre ellos.

Habiendo comprendido, pues, los puntos previos, su fundamento resulta bastante sencillo. Tras realizar el *spectral unmixing* es posible volver a recomponer la imagen utilizando sus productos, es decir, sumando las matrices -las imágenes- de distribución de cada uno de los *Endmembers* y el vector residual. Si se supone que este vector residual contiene todo el ruido presente en la imagen, este componente se puede ignorar durante la reconstrucción de modo que la imagen resultante recompuesta resultará únicamente de la combinación de los espectros de los *Endmembers,* quedando libre de ruido y no mostrando otra cosa que la información relevante de la imagen. La idea se resume en las siguientes expresiones:

$$m = \sum_{i=1}^{k} x_i s_i + r \qquad \hat{m} = \sum_{i=1}^{k} x_i s_i$$

Figura III: Izquierda: una imagen hiperespectral "m" consiste en un conjunto de píxeles "i" formados por espectros "x_i" que ocupan una fracción de abundancia "s_i" más un componente de error "r". Derecha: imagen reconstruida "$\dot{m}$" con los componentes anteriores ignorando el vector residual "r".

Como se puede deducir de lo anterior, el UBD requiere tanto de la imagen hiperespectral a ser limpiada como de los *Endmembers* presentes en ella como parámetros de entrada, siendo la reconstrucción tanto mejor cuanto más representativos de los elementos puros presentes en la imagen sean estos *Endmembers.* Para asegurarse de que los éstos están presentes en la imagen, lo que se hace es tomarlos directamente de ésta. Además, es importante que estos espectros no contengan nada de ruido para que éste quede totalmente aislado en el vector residual que va a ser ignorado durante la reconstrucción de la imagen. Si tras realizar ésta con un determinado set de espectros iniciales -o librería espectral- se observa aún ruido, se asume que éste procede de errores en el método, esto es, de lo bien que hayan tomado estos *Endmembers*. Entre los errores posibles se encuentran el considerar un número inadecuado de espectros -de elementos presentes en la imagen- o la inclusión de espectros no puros, es decir, de espectros que resultan como combinaciones de *Endmembers* puros. Este proceso queda reflejado en la siguiente figura.

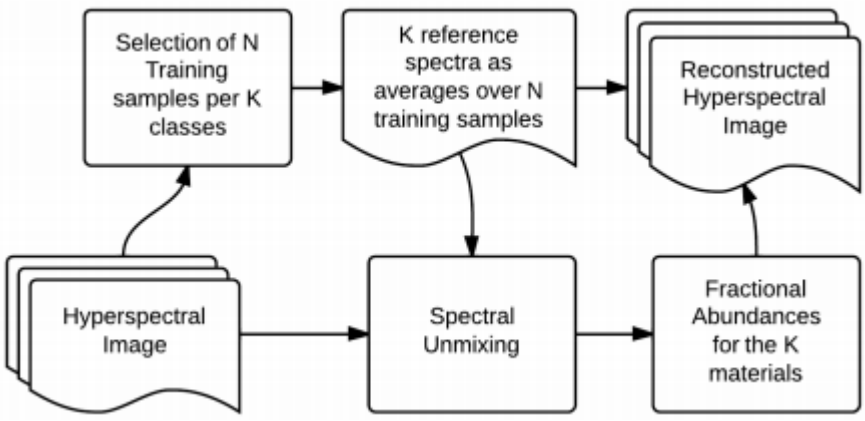

Figura IV: Flujo de trabajo del UBD.

Si bien el método UBD ha demostrado inicialmente que presenta claras ventajas frente a otros como el filtrado de la imagen (*Daniele Cerra et al., 2012*), también presenta algunos claros inconvenientes que han de ser corregidos en la medida de lo posible. Son algunos de estos problemas los que han motivado el desarrollo de un algoritmo que los aborde para pretender solucionarlos.

Así pues, entre las ventajas que presenta el UBD frente a otros procedimientos de reducción del ruido cabe enumerar:

- Una mayor efectividad a la hora de conservar la información relevante sin perjudicar otros aspectos de la imagen.

- Produce resultados basados en cada píxel -pixel-based-, con lo que permite incluso obtener información para píxeles que originalmente carecían de ella.

- Sus resultados presentan un significado físico, ya que se basan en la reconstrucción de imágenes con elementos presentes en la realidad

- Libera bandas anteriormente no utilizables debido a su bajo SNR para nuevas aplicaciones

Por otra parte, entre las desventajas observadas cabe nombrar:

- Los resultados son dependientes de los espectros de referencia iniciales, ya que esto afecta a la cantidad de información presente en el residual vector.

- Se produce una pérdida de información espectral local.

3. PROCEDIMIENTOS

Alguno de los problemas que presenta el UBD, especialmente el relacionado con sus datos de entrada, afecta directa y fuertemente a la calidad de los resultados y requiere, por tanto, de algún tipo de solución. Por ello se ha ideado un procedimiento en forma de algoritmo que se desarrolla alrededor del UBD y lo complementa para que éste pueda ser inicialmente independiente de los *Endmembers* de entrada. Este algoritmo ejecuta de modo iterativo el UBD junto a una serie de operaciones consecutivas que van considerando siempre un número creciente de espectros de entrada en el modelo, lo que hace que la reconstrucción de la imagen sea cada vez más completa.

El algoritmo desarrollado presenta los mismos requerimientos iniciales que los indicados anteriormente para el UBD, esto es, la propia imagen a ser limpiada de ruido y una librería inicial de *Endmembers*. Con ambos elementos se ejecuta inicialmente el UBD, es decir, se realiza el *spectral unmixing* y la reconstrucción ignorando el vector residual. De este modo se obtiene una imagen reconstruida que, al ser dependiente de los espectros iniciales, no tiene por qué quedar completamente libre de ruido. Esto se suele deber, como se indicó anteriormente, bien a que la librería inicial es incompleta o es incorrecta, y hace que tras la reconstrucción de la imagen no haya sido únicamente el ruido aislado en el vector residual, sino que éste también contenga otra información relevante referente a elementos presentes en la imagen.

Si se analizan las diferencias entre la imagen original -con ruido- y la imagen reconstruida es posible observar toda aquella información que no se ha incluido en la reconstrucción. El objetivo es aislar en esta imagen únicamente aquella información perteneciente al ruido de la imagen original, que presenta un comportamiento espacial aleatorio, por tanto, si en esta imagen se observasen distribuciones no aleatorias de la información, es de suponer que no se están teniendo en cuenta correctamente todos los materiales presentes en la imagen y por lo tanto, que es necesario añadir espectros adicionales a la librería inicial de *Endmembers* con el propósito de ejecutar de nuevo un UBD que resulte más satisfactorio.

Este procedimiento se repite de manera iterativa, añadiendo cada vez más información al modelo de reconstrucción de la imagen, hasta considerar que en la imagen residual ya no existe información relevante para la reconstrucción y que los resultados ya no mejoran la calidad de la imagen. En ese momento, se detiene el proceso y se obtiene una imagen limpia de ruido.

Se pretende que el algoritmo desarrollado se ejecute de forma automática, sin necesidad de que sea supervisado. De este modo, se puede decir que queda constituido por tres etapas, esto es, una inicialización, un proceso de iteraciones y un criterio de detención. El esquema conceptual del proceso queda reflejado en *Figura V*.

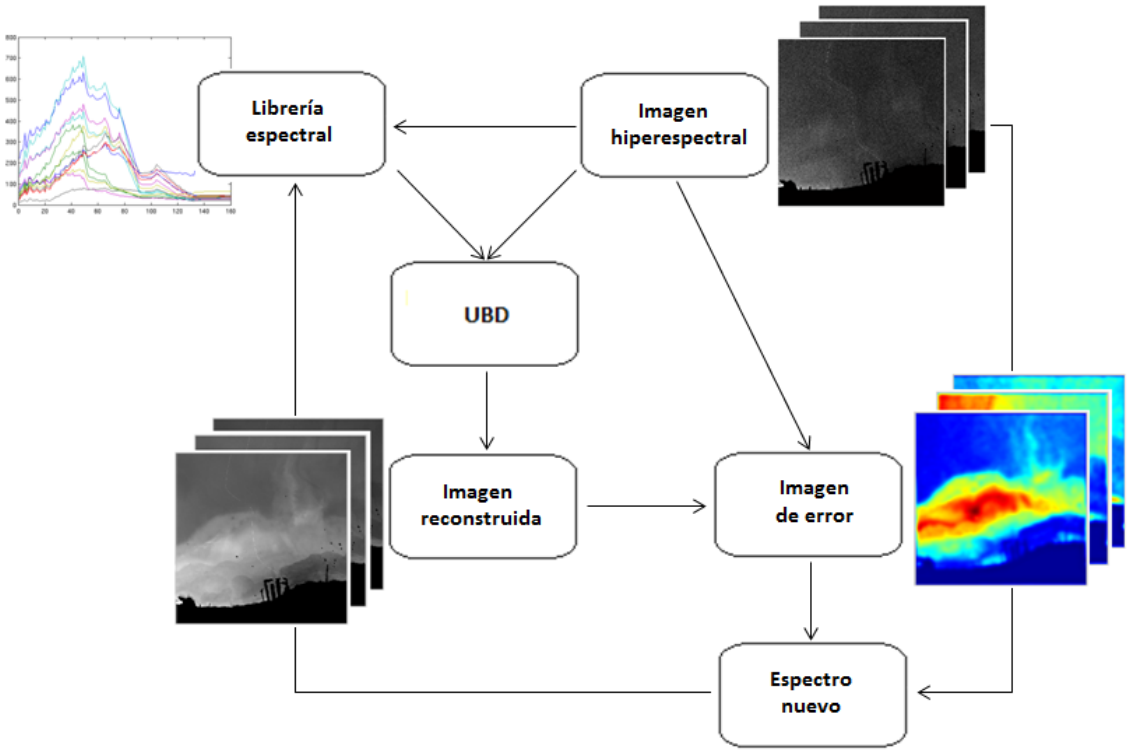

Figura V: Esquema conceptual del proceso seguido por el algoritmo.

La elaboración de las funciones y los algoritmos se ha realizado íntegramente en Matlab, que es un lenguaje de programación especialmente adecuado para el análisis y el tratamiento digital de imágenes debido a su facilidad de trabajo matricial y a las completas librerías de funciones que ya presenta, entre ellas, una librería de funciones para el tratamiento de imágenes hiperespectrales. Como se ha señalado anteriormente, una imagen hiperespectral no es una más que una matriz de tres dimensiones, correspondientes a las filas, las columnas y al número de bandas de la imagen. Matlab es capaz de acceder a cualquier banda o píxel de la imagen y hacer operaciones de toda clase con ellos. Además, presenta un ambiente de programación y una interfaz gráfica muy intuitivas que facilita mucho el trabajo con las variables -por ejemplo, los archivos de imagen- y las funciones. De este modo, es posible llamar a una imagen hiperespectral con la que se quiere trabajar, guardarla en una variable y visualizarla previamente, crear las funciones necesarias para tratarla, ejecutar los algoritmos sobre ella y analizar los resultados.

De este modo, se han desarrollado en código Matlab una serie de funciones (ver anexo) que operan de forma conjunta llamándose unas a otras, y que se integran en el algoritmo el par de algoritmos finales "*ti_total*" y *ti3_total*"" (ver anexo). La primera se encarga de reducir el ruido en toda la imagen mientras que la segunda es específica de una banda en concreto. Aunque las funciones realicen tareas sencillas de manera individual, éstas, junto a la librería

de funciones hiperespectrales de Matlab, son capaces de realizar el procedimiento iterativo descrito previamente.

A continuación, se procederá a describir cada una de las fases de las que se compone el algoritmo de forma separada.

3.1. INICIALIZACIÓN

Como se indicó anteriormente, existe una gran variedad de algoritmos de identificación de *Endmembers* basados en diferentes técnicas (*Nirmal Keshava, 2003*). Cualquiera de ellos podría ser empleado, en principio, para tomar un set inicial de *Endmembers* más o menos representativo de los materiales presentes en el área de estudio. Si bien esto es cierto, el método funciona de manera satisfactoria si se utiliza un algoritmo de extracción de *Endmembers* que obtenga un número menor de espectros que los presentes en la propia imagen. Sin embargo, es necesario para la reconstrucción que los espectros obtenidos por este algoritmo tengan sentido físico, esto es, aunque obtenga un número menor de espectros iniciales, éstos deben corresponder a materiales reales presentes en la imagen, y no a combinaciones de éstos.

Cabe señalar que durante los trabajos desarrollados se observó, primeramente, que en principio, el algoritmo funcionaba correctamente independientemente del set inicial de espectros introducidos, esto es, aportaba resultados buenos con números muy bajos de espectros e incluso si estos espectros ha sido obtenidos manualmente sin supervisión. Este aspecto es importante ya que aporta cierta flexibilidad inicial al proceso. En cualquier caso, durante las prácticas se trabajó mayormente con el “Outliner detector method” -ODM- desarrollado por Charoula Andreou, de la Universidad de Atenas (*Charoula Andreou, 2013*).

3.2. ITERACIÓN

Al contrario de la inicialización, el proceso de iteración es el punto clave del proceso, el que mayores dificultades presenta y sobre el que se ha realizado un mayor esfuerzo. En este punto se van añadiendo de forma paulatina, esto es, uno a uno, nuevos espectros a la librería inicial obtenida durante la inicialización. La robustez de este punto determinará la validez final del método, de modo que si los espectros que se van añadiendo no son lo suficientemente representativos de los materiales presentes en la imagen la reconstrucción de la imagen no será adecuada, y por lo tanto la imagen no llegará a quedar suficientemente limpia de ruido. Esto repercutirá, lógicamente, en aplicaciones futuras.

La imagen de error:

El proceso de iteración se centra en la imagen de error, que es sobre la que se trabaja directamente para detectar los nuevos espectros. Como se observa en la *Figura* VI, esta imagen se obtiene a partir de la substracción de cualquiera de las imágenes reconstruidas a la imagen original con ruido.

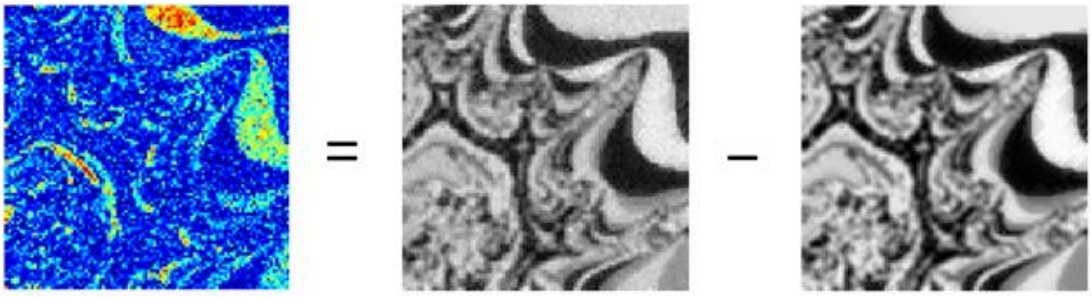

Figura VI: Obtención de la imagen de error: Imagen de error = imagen original con ruido - imagen reconstruida.

Como cabe esperar, esta imagen de error contiene la información del vector residual, explicado anteriormente, esto es tanto el ruido a eliminar como información relevante para el modelo que no se ha considerado en la librería de espectros utilizada durante la reconstrucción inicial de la imagen. Se puede deducir que siempre que se observe un comportamiento espacial no aleatorio de información presente en esta imagen aún existen atributos, es decir, *Endmembers*, presentes en la imagen que no se han considerado durante la reconstrucción previa. Puesto que se trata de información relevante no considerada durante la reconstrucción, sería necesario añadir un espectro representativo de ésta a la librería inicial de espectros, de modo que quede más completa. En esto es en lo que se basa realmente trata el proceso iterativo, en la búsqueda y obtención de los espectros de los materiales presentes en la imagen y su posterior adición a la librería inicial de espectros. De este modo los atributos no considerados en iteraciones anteriores que aún están presentes en la imagen se van reduciendo y la reconstrucción resulta cada vez más completa. En principio se espera que la información presente en esta imagen sea siempre menor, y que ésta presente una homogeneidad espacial mayor a medida que el proceso avanza con las iteraciones.

Desde luego, es un proceso delicado que implica la correcta elección de los nuevos espectros. El criterio básico que se ha llevado a cabo para la extracción de un nuevo *Endmember* ha sido la identificación del píxel con máximo valor absoluto en la imagen de error. Así pues, el primer paso es la obtención de la imagen de error, donde se muestra todo aquello que el proceso de UBD no ha sido capaz de reconstruir, ya sea a causa del ruido o de fallos en los datos de entrada del modelo por cualquier característica que no se haya considerado. Lo que se pretende es encontrar un punto representativo de estas características del que sea posible obtener un espectro, lo que, debido a la gran cantidad de ruido que presentan las imágenes de bajo SNR (*Figura VII izquierda*) presenta dificultades y hace requerir un procesado previo de la imagen.

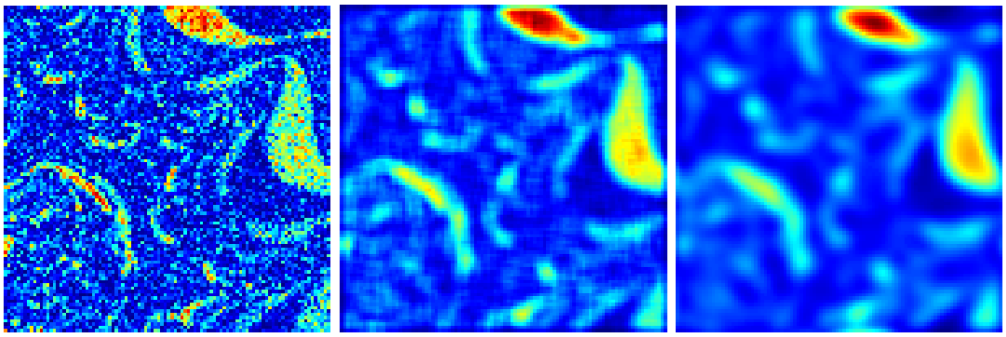

Figura VII: Procesado de la imagen de error para la identificación del valor máximo.

De estas tareas se encargan las funciones "*imresmedfou2*" y "*imfourier*" (ver anexo). La primera se encarga deobtener y procesar la imagen de error, para lo que realiza tres operaciones: primeramente substrae de la imagen original la imagen reconstruida píxel a píxel y ajusta su valor mínimo a cero, seguidamente pasa un filtro de media -paso bajo- a la imagen para reducir el peso del ruido, y por último realiza a partir de esta imagen filtrada la transformada de Fourier, que tiene como efecto práctico la discretización de los valores de unas frecuencias determinadas. Este último paso, que presenta cierta complejidad, se realiza haciendo una llamada a "*imfourier*", cuyo cometido es el de realizar la transformación de Fourier. Para ello se efectúa un proceso estándar que consiste en, primeramente, transformar la imagen del dominio del espacio al de las frecuencias. Para que la visualización sea mejor, se mueven a continuación las frecuencias bajas, que se encuentran originalmente en las esquinas de la imagen, al centro de ésta. Después se filtra la imagen con un filtro Butterworth, para lo que se crea una malla adaptada a las dimensiones de ésta. El parámetro que controla el radio del filtro Butterworth en la función es ajustable por el usuario, y es un valor que depende del las dimensiones de la imagen. Durante el desarrollo

de estas funciones, se ha determinado este valor únicamente de manera empírica, mediante la simple observación visual del aspecto de los resultados y la adecuación de éstos con las sucesivas operaciones. El filtrado de la imagen de frecuencias lo que hace es seleccionar sólo ciertas frecuencias, situadas en el centro de la imagen, con cuyos valores se va a trabajar posteriormente. Una vez hecho esto, lo último que hace la función es trasformar este producto filtrado de nuevo al dominio del espacio, deshaciendo las transformaciones anteriores y obteniendo una imagen en la que se identifican mucho más claramente los elementos de importancia presentes en ella.

De esta imagen es de la que se identifica, a continuación, la posición del valor máximo, presuntamente representativo de una de las características no consideradas anteriormente en el modelo. Para ello se identifican primeramente las coordenadas del píxel de la imagen con máximo valor. De esto se encargan las funciones "*maxhyp_bands*" y "*maxhyp2*" (ver anexo), que devuelven la fila, la columna -y la banda con "*maxhyp_bands*" de este punto. La primera función se encarga de encontrar el máximo en toda la imagen -en todas las bandas-, mientras que la segunda es específica de una banda en concreto.

A continuación, desde las propias funciones "*ti_total*" y "*ti3_total*" se obtiene el espectro de ese píxel de valor máximo utilizando las coordenadas obtenidas anteriormente. Si bien el método para obtener el espectro es sencillo, hay que tener en cuenta un par de aspectos importantes. Por un lado, como se hizo alusión anteriormente, el espectro se obtiene de la imagen inicial puesto que es en ésta donde se encuentra la información original. Si bien esto es cierto, sólo lo es de manera parcial, ya que requiere una matización, y es que no se debe olvidar que la imagen presenta mucho ruido, lo que lógicamente afecta a la toma directa de espectros en esta imagen -éstos tendrán también ruido-. Este ruido puede reducirse mediante la toma del espectro a partir de un valor medio centrado en las coordenadas obtenidas anteriormente.

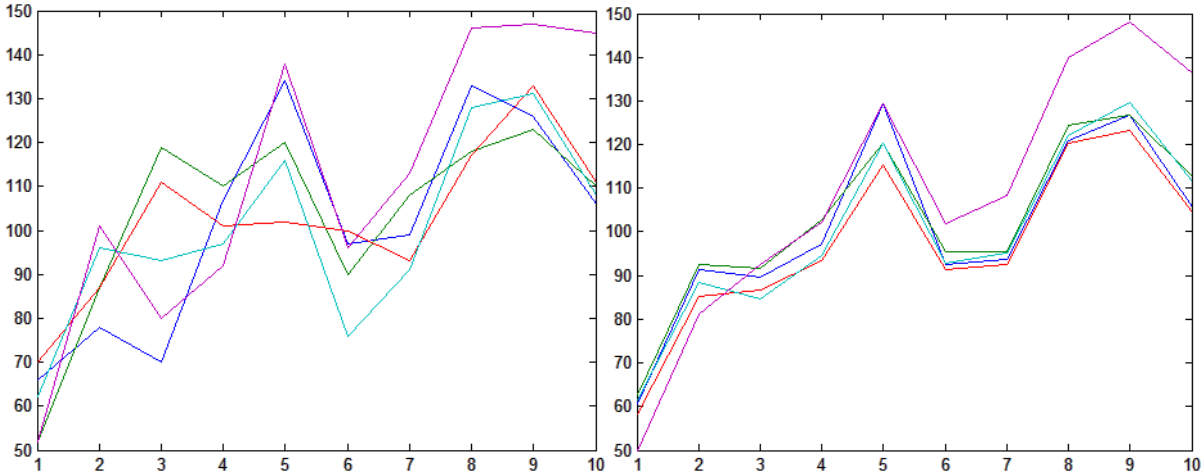

Figura VIII: Diez bandas consecutivas de cinco espectros aleatorios. Izquierda: antes del filtrado. Derecha: tras el filtrado.

Como se observa en la *Figura VIII*, un espectro medio tomado en una zona homogénea, es capaz de eliminar el posible ruido presente, pero ésta no es la única ventaja de tomar espectros medios en lugar de directamente del píxel en particular. Hacerlo de este modo también permite obtener espectros más representativos del elemento presente en el área. Por esto el método toma un espectro medio centrado en las coordenadas obtenidas en el paso anterior mediante un filtro de media dinámico -este punto se abordará de nuevo en la explicación del criterio de detención- que hace una ventana de filtrado mayor o menor en función de la heterogeneidad presente en la zona, de este modo, si el área donde se encuentra el espectro a tomar es muy homogénea, el tamaño de la ventana podrá ser mayor que si el área presenta valores muy heterogéneos, en cuyo caso será de menor tamaño.

Esta homogeneidad espacial se ha medido con el *Spectral Angel Mapper*, o SAM. El SAM es una medida de similitud entre un par de espectros basada en el ángulo que forman sus vectores en un espacio con tantas dimensiones como bandas presentan los espectros. A menor SAM, mayor similitud habrá entre un par de espectros. Una de las ventajas de esta medida es que no se ve afectada por condiciones de iluminación diferentes (*Figura IX*). El método, simplemente, establece un límite -como en otros casos, empíricamente y configurable por el usuario- por encima del cual se toma el espectro medio de una ventana de 3x3, y por encima, el espectro de una ventana de 5x5 (ver anexo, referencia a "*Getting new spectra*" en "*ti_total*" y "*ti3_total*"). De este modo se evita hacer la media de espectros de elementos diferentes en las zonas más heterogéneas.

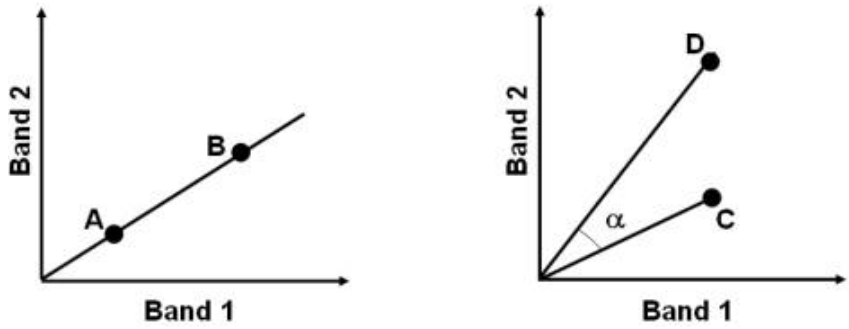

Figura IX: Concepto del Spectral Angle Mapping Izquiera: A y B se corresponden al mismo material bajo diferentes niveles de iluminación. Derecha: C y D son diferentes materiales. El ángulo α indica la semejanza entre ambos materiales.

Una vez obtenido el espectro, que no es más que un vector con tantos valores como bandas tenga la imagen, se emplea la función "*addspec"* (ver anexo) para añadirlo a la librería inicial de espectros. Esta función lo único que hace es añadir los valores de cada banda por separado a una nueva columna en la tabla de la librería espectral.

Éste es básicamente el proceso de iteración. Una vez se ha obtenido y añadido el nuevo *Endmember* a la librería se guardan los resultados de la iteración. Posteriormente, tras comprobar que no se cumplen las condiciones de salida del algoritmo, se vuelve a ejecutar el proceso nuevamente, esta vez, con un espectro más.

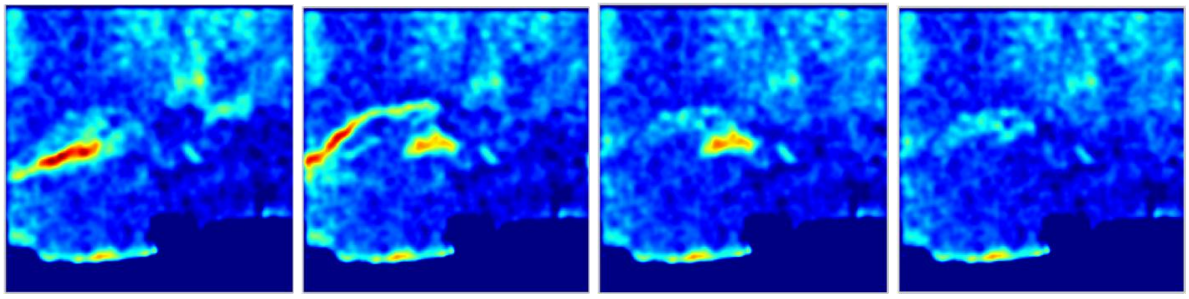

Figura X: Comportamiento continuo del proceso de iteración sobre la imagen de error. El error se va reduciendo.

3.3. FINALIZACIÓN

Debido a la particular forma que tiene el proceso iterativo de búsqueda de nuevos *Endmembers*, tomando máximos, el método debería, teóricamente, de continuar añadiendo nuevos espectros a la librería de forma indefinida ya que siempre va a existir un punto máximo en la imagen, esto es, un píxel con mayor valor que otro. Sin embargo, esto no puede ser así, ya que en la imagen sólo existe un número determinado de *Endmembers*, y no infinitos. Así pues, el proceso debería de terminar idealmente en el momento en el que se

hayan considerado todas las sustancias presentes en la imagen antes de comenzar a modelar ruido, de modo que se observe que la distribución final del ruido ya se comporta de forma aleatoria. Idealmente, esto se consigue cuando se ha añadido el número exacto de *Endmembers* presentes en la imagen.

Por ello, al proceso hay que añadirle algún criterio de detención que sea capaz de parar el flujo iterativo en el momento oportuno. Para ello, se han elaborado y probado varias alternativas basadas en el mismo criterio, y es que se ha observado que existen varios parámetros de la imagen que tienden a mostrar un comportamiento asintótico a medida que el proceso avanza.

Para empezar, éste es el caso del valor medio de la primera derivada para toda la imagen de error: o lo que es lo mismo, el valor medio de la cantidad de ruido reducida con cada iteración. Como se espera, en cada ciclo del proceso, el ruido que se reduce al considerar siempre un modelo más completo va decreciendo paulatinamente, siendo las reducciones de ruido al comienzo del algoritmo muy altas y al final casi inapreciables -piénsese que se van eliminando máximos cada vez más pequeños-, de modo que se llega a un punto en el que la calidad de la imagen no se ve mejorada al añadir más *Endmembers.* De este paso se encarga la función "*res_res2*" (ver anexo), que da la opción de calcular el máximo de la diferencia, esto es, el valor del píxel que ha sufrido una reducción mayor de su valor, o la media de esta diferencia. La función forma parte del algoritmo principal, desde el que se la llama de modo que si su valor es menor que un número determinado, el algoritmo completo finaliza. Los valores de este parámetro tienden a cero en cualquier imagen, con lo que resulta bastante intuitivo determinar el valor de salida. Aunque éste haya sido el criterio utilizado por este motivo, este criterio presenta un problema, y es que el valor de la reducción del ruido no es siempre menor a la anterior, con lo que el proceso corre el riesgo de detenerse antes de completarse.

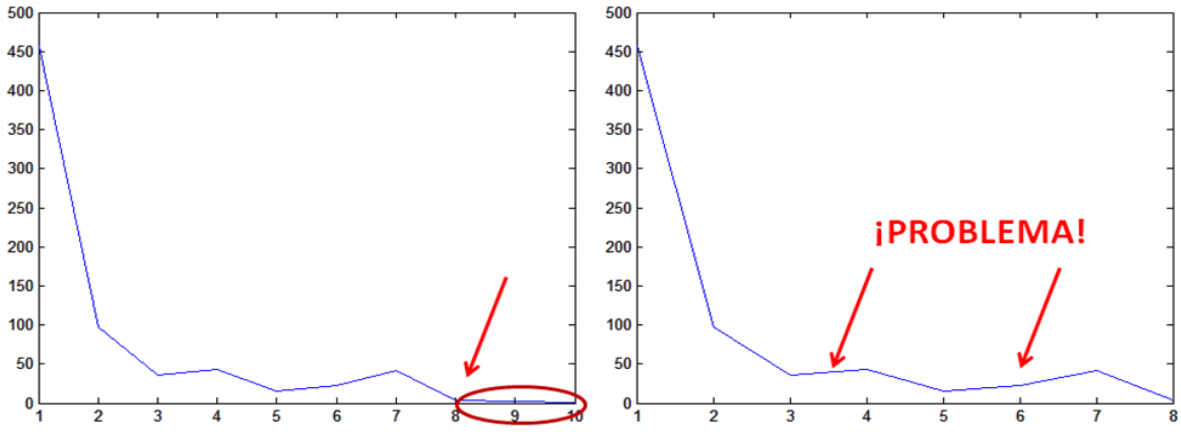

Figura XI: Comportamiento asintótico como criterio de stop. Izquierda: momento ideal de detención para una imagen sintética de la que se conoce su número de Endmembers. Derecha: problema presentado por este criterio.

Los otros dos criterios utilizados se comportan de manera semejante y no presentan el problema del parámetro anterior. Estos son: el valor medio del SAM para toda la imagen entre cada uno de los píxeles de la imagen original reconstruida y la original y el error medio cuadrático -RMSE- realizado igualmente entre la imagen reconstruida y la original. En ambos casos, estos criterios se comportan de manera asintótica hacia un valor concreto de modo que la imagen reconstruida es cada ciclo más diferente a la original, y ambos parámetros tienden a un valor concreto que es diferente de 0 y que dependerá de los valores de cada imagen en particular. En cualquier caso, el valor de corte se aplica a la derivada de la curva obtenida, es decir, a su pendiente, con lo que no es un valor absoluto, sino relativo al total de la imagen y que efectivamente tiende a 0. En estos casos, el método

empleado para detener el algoritmo sería el mismo, esto es, establecer un límite de modo que si se traspasa el proceso se detiene al considerase que ya no se reduce más el ruido o incluso que comienza a modelarse. En este caso la curva cambiaría su pendiente, haciéndose creciente al hacer la imagen reconstruida cada vez más parecida a la original.

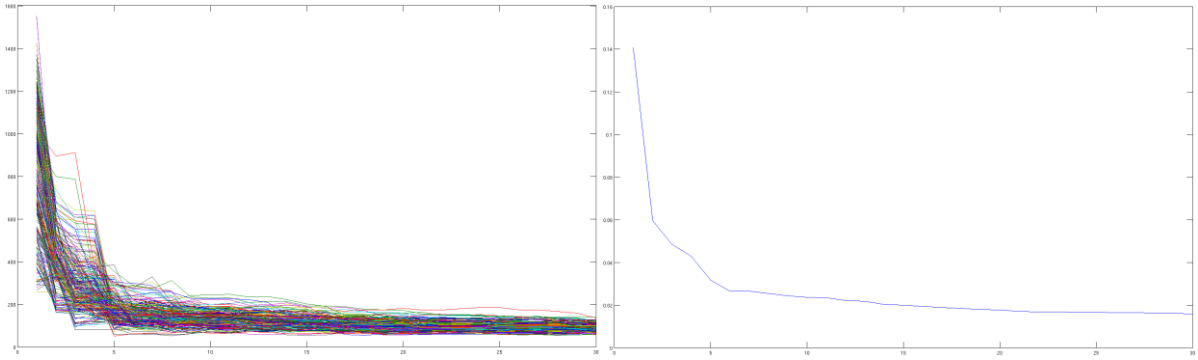

Figura XII: Izquierda: variación del RMSE entre cada banda de la imagen original y la reconstruida a lo largo de las iteraciones. Derecha: SAM medio de todos los píxeles de la imagen entre la original y la reconstruida a lo largo del proceso.

Además de estos tres criterios de detención del proceso, se ha integrado otro en el algoritmo que, aunque no se trate de un criterio de detención propiamente dicho, sí que finaliza el proceso. Esto se basa al hecho de que existe un problema inherente al método que no ha podido llegar a solucionarse y que por motivos computacionales hace necesario detener el proceso antes de su finalización, por lo tanto, afectando de este modo al proceso de stop. El problema aparece debido a que el método de tomar nuevos *Endmembers* afecta indirectamente a la reconstrucción. Como se indicó anteriormente, el espectro que se añade con cada iteración a la librería espectral utilizada para la reconstrucción no es el del píxel que presenta un mayor valor en la imagen de error, sino el de un área homogénea alrededor de éste obtenido a partir del espectro medio de una ventana de píxeles de tamaño variable alrededor de este máximo. Utilizar este espectro y no el del píxel máximo directamente tiene como consecuencia que tras la reconstrucción de éste, su valor varíe y por lo tanto, que su valor en la imagen de error sea diferente de 0 -si un píxel se reconstruye con su mismo espectro, este píxel será una combinación del 100% de este espectro y el 0% del resto-. Esto produce que el algoritmo pueda llegar a detectar este mismo píxel en más de una ocasión como máximo, y por lo tanto, que añada dos veces el mismo espectro, lo que ocasiona errores de computación y la pérdida del proceso ejecutado hasta el momento que hacen necesario evitar este hecho. Por ello, el algoritmo se detiene antes de llegar a este punto, de modo que si resulta que se añaden dos veces el mismo espectro a la librería, el proceso se detiene antes de ejecutar de nuevo el UBD. Aunque es un punto a clarificar y mejorar en el futuro, en el algoritmo se encarga de esto la función "*stopp1*" (ver anexo).

4. RESULTADOS

En el presente trabajo se ha trabajado principalmente con una imagen tomada por el sensor HySpex, que presenta las principales especificaciones siguientes (*www.hyspex.no*):

Rango espectral	0.4–1.0 µm
Número de píxeles por banda	1600
Número de bandas	160
Digitalización	12 bits

Figura XIII: Algunas especificaciones del sensor HySpex.

El algoritmo se aplicó sobre dos sets de imágenes diferentes. Por un lado, se comprobaron sus capacidades y se analizó sobre una imagen sintética de la cual se conocían perfectamente los componentes que la conformaban. Por otro, se analizó sobre una imagen real tomada por el sensor aeroportado HySpex sobre el lago de Starnberg -Starnberger See-, situado al sudoeste de la ciudad de Múnich, en el estado de Baviera, en el sudeste de Alemania (*Figura XIV*).

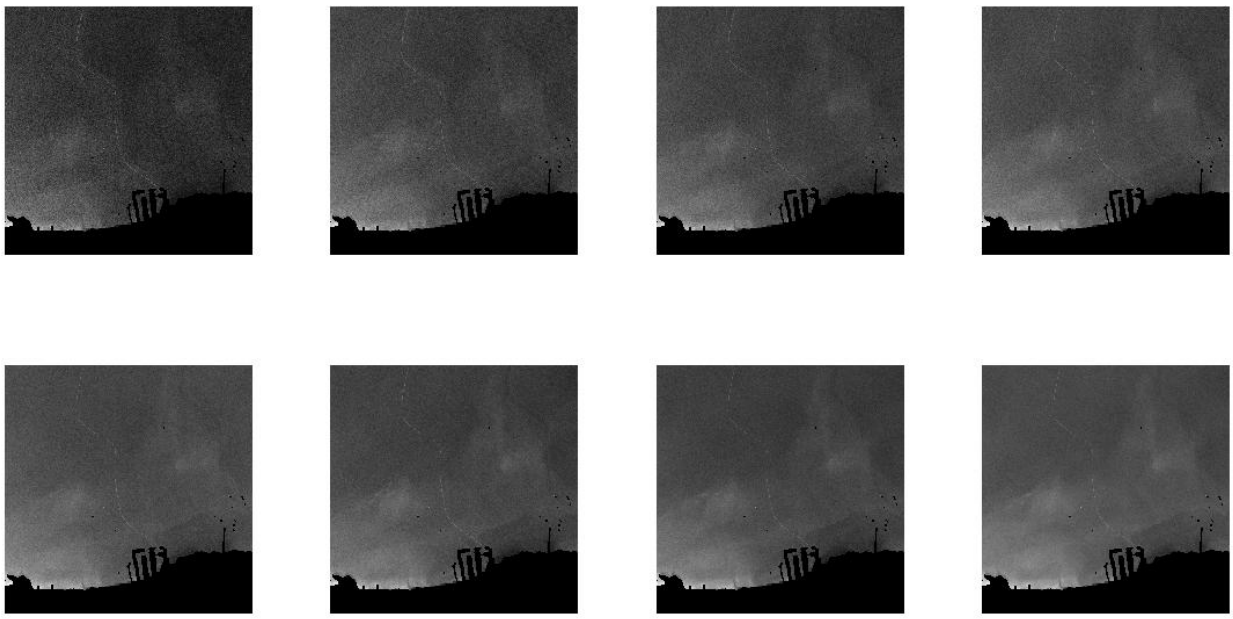

Figura XIV: Primeras 8 bandas de la imagen del Starnberger See, correspondientes al rango del azul, donde el SNR es menor.

Con esta última imagen, lo primero que se ha hecho ha sido una máscara para eliminar toda la costa. Los valores de las zonas de tierra y de los barcos presentes se han puesto a cero, de modo que únicamente se ha dejado visible el agua. De este modo estas zonas han quedado excluidas de los siguientes cálculos. Además de hacer esto para poder utilizar la imagen correctamente en aplicaciones acuáticas, es especialmente importante que no haya ningún píxel visible en la imagen que no sea de agua para que a la hora de extraer los *Endmembers* de ésta no se extraigan espectros de costa o de embarcaciones que perturben la reconstrucción del agua. Esta potencial perturbación se debería a que también en estos casos los espectros se obtienen con medias espaciales, lo que haría que un píxel aislado de estas zonas hiciera su media con píxeles de agua dando lugar a espectros que realmente no tienen ningún significado físico, afectando a la reconstrucción de la imagen.

Para realizar una máscara se ha de crear la función "*water_masc*" (ver anexo) que discrimina el agua de todo lo que no es agua. Esto se puede hacer sencillamente al considerar que el espectro del agua en longitudes de onda más largas presenta valores de reflectancia muy bajos, apareciendo en estas bandas con colores muy oscuros, algo que contrasta con el resto de las superficies de la imagen (*Figura XV*). Este comportamiento diferencial se puede utilizar para marcar un límite por encima del cual se asignan a todos los píxeles el valor cero. La función "*water_masc*" realiza este proceso aplicando el límite a los valores de la banda 150 del sensor HySpex, que tiene una longitud de onda de 0´9 micrómetros. Para perfilar los límites y asegurarse la exclusión de todos los píxeles que no son agua se emplean filtros morfológicos con una serie de aperturas y cierres que dilatan y erosionan los bordes. A continuación, las pequeñas imperfecciones del método se pueden corregir manualmente, píxel a píxel, en el mismo código. Por último, se aplica esta máscara a toda la imagen, es decir, a todas las bandas, de lo que se encarga la función "*mascall*" (ver anexo).

Figura XV: Banda 150 (0´9 micrómetros), empleada para el proceso de enmascarado. Nótense las diferencias entre el agua y el resto de los elementos de la imagen.

Figura XVI: Banda 3 antes y tras el enmascarado. Las grandes diferencias se deben al stretching.

Más allá del infrarrojo, el agua presenta una reflectancia muy similar y homogénea con valores muy bajos. Este comportamiento homogéneo en las longitudes de onda más largas hace que estas bandas presenten altas correlaciones entre ellas, y que aporten prácticamente la misma información, siendo redundantes. Por ello, pueden eliminarse de modo que los cálculos no se vean afectados negativamente de modo que se puedan hacer más ligeros, un aspecto que en teledetección hiperespectral aún tiene importancia. Además de esto, hay que considerar que la mayoría del ruido se distribuye en la imagen en las primeras bandas, de longitud de onda menor, y que para aplicaciones acuáticas es suficiente trabajar con las bandas más bajas correspondientes al rango visible e infrarrojo cercano -el llamado VINIR-. En el sensor HySpex esto se corresponde entre las bandas 1 y 70, que son las longitudes de onda comprendidas entre los 0´4 y 0´7 micrómetros. Por tanto, es posible hacer un subset de toda la imagen con estas bandas eliminando las últimas 90. La *Figura XVII* muestra este procedimiento.

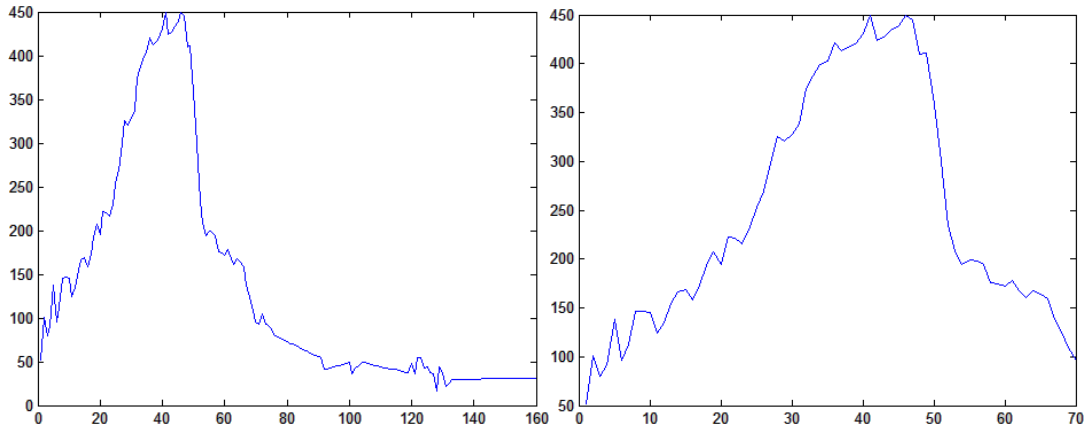

Figura XVII: Izquierda: espectro completo de un píxel de agua. Derecha: subset del VINIR del mismo píxel.

En importante matizar que hay que realizar la máscara antes de hacer este subset, ya que este último elimina las bandas del infrarrojo que se emplean para hacerla. Una vez realizadas ambas operaciones, la imagen ya se encuentra preparada para desarrollar el algoritmo.

Figura XVIII: Banda 2 -4´2 µm- de la imagen del Starnberger See. Izquierda: imagen con ruido. Centro: imagen reconstruida. Derecha: diferencia entre ambas, obsérvese la ausencia de elementos importantes en ésta.

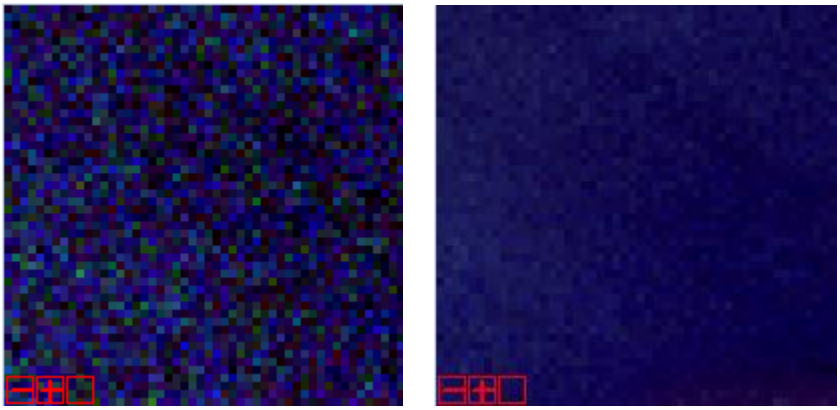

Figura XIX: Comparación de los resultados. Composición RGB del sensor HySpex -a 4´5, 4´4 y 4´3 µm-. Izquierda: imagen sin procesar. Derecha: resultados tras aplicar al algoritmo. Arriba: imagen completa. Abajo: imagen en detalle.

Estas bandas, que anteriormente no habían podido emplearse debido a la gran cantidad del ruido, se probaron en la misma DLR en aplicaciones acuáticas obteniendo buenos resultados prácticos en mediciones de absorción de Gelbstoff, profundidad del agua o brillo del fondo.

Por otro lado, se probó el método con una imagen sintética. En este caso, al tratarse de una imagen de la que se conoce perfectamente su composición se pudo comprobar y cuantificar la bondad del método.

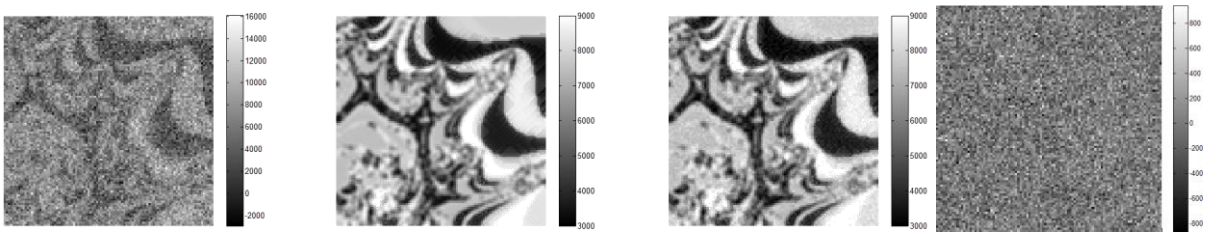

Figura XX: De izquierda a derecha: banda 40 de la imagen sintética con gran cantidad de ruido añadido, misma banda 40 sin ruido -objetivo ideal a alcanzar-, imagen reconstruida tras aplicar el UBD utilizando los 9 espectros originales, imagen de error -diferencia-, que sólo contiene ruido aleatorio.

En la *Figura XX* se puede apreciar visualmente cómo de semejante es la imagen sintética a la imagen original. De forma numérica esto se expresa con el RMSE y el SAM, los cuales se midieron arrojando los siguientes valores: RMSE: 0´2855, SAM: $5´0481e^{-5} + 4´2147e^{-12i}$, que son valores de diferencia muy bajos. Esto significa que el método funciona bastante bien, y que los errores que produce en relación a los valores de la imagen original, con un rango comprendido entre -3.000 y 16.000, pueden seguramente despreciarse sin problemas.

Por último, a parte de los resultados que se pretendían conseguir inicialmente con el desarrollo del algoritmo, se han observado otros resultados totalmente inesperados que parecen ser también bastante interesantes, y es que el proceso que sigue el algoritmo resulta un método que, en principio, parece servir adecuadamente a la hora de extraer los *Endmembers* originales de una imagen. Utilizando la imagen sintética, de la que se conocen inicialmente sus espectros puros, se pudieron comprobar éstos con los espectros extraídos por el método. Se obtuvieron, tanto a partir de imágenes limpias como imágenes muy corrompidas por ruido, espectros muy semejantes. Aunque este aspecto parece bastante prometedor, aún habría que analizarlo más y ofrecer resultados numéricos que apoyen estas afirmaciones. En la *Figura XXI* se observan bien estas semejanzas.

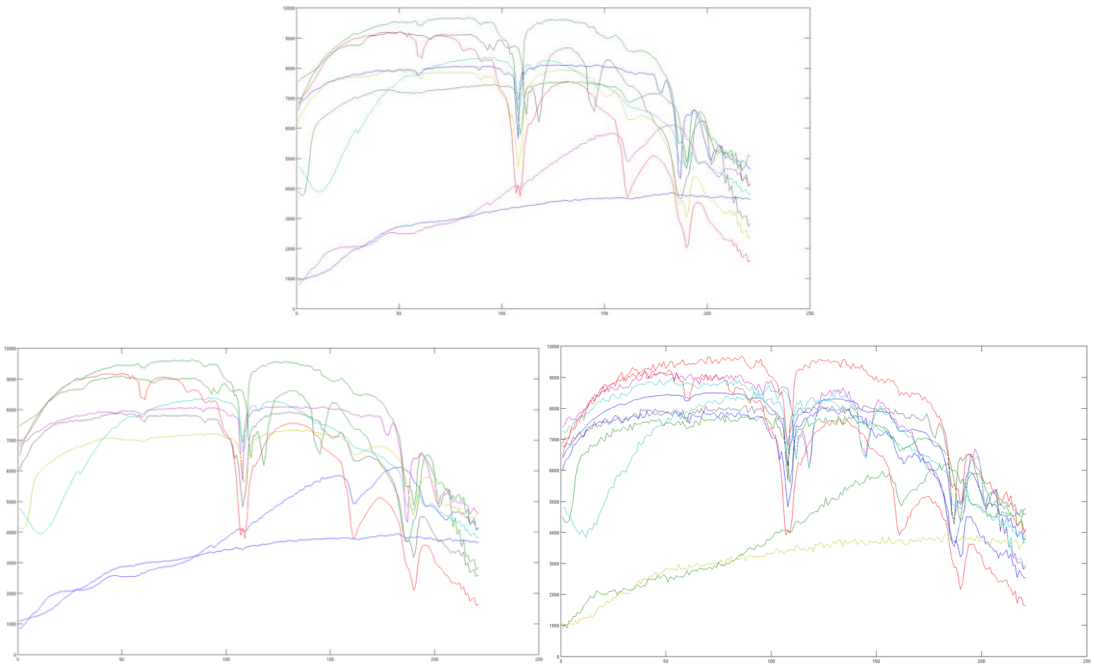

Figura XXI: Arriba: Endmembers originales de la imagen sintética. Abajo: resultados obtenidos a partir del algoritmo desarrollado. Izquierda: Endmembers extraídos de la imagen limpia de ruido. Derecha: Endmembers extraídos de la imagen con baja SNR.

5. CONCLUSIONES

Tras analizarlos, los resultados obtenidos podrían clasificarse en positivos y negativos. Para empezar, el mayor aspecto positivo alcanzado podría decirse que efectivamente se han alcanzado los objetivos que se pretendían conseguir inicialmente con el método, que era el de suplir las carencias iniciales del UBD al obtener un algoritmo capaz de hacer uso de éste independientemente de los espectros iniciales. Además, se ha conseguido automatizar el proceso en gran medida, haciendo que no se requiera mucho control por parte del usuario, aunque quizás habría que mejorar este punto en el futuro. Además, el método continúa preservando tanto la información espectral como espacial gracias al UBD. Por último, no hay que olvidar el "producto secundario" que se ha encontrado en el algoritmo como método de extracción de las firmas espectrales puras de los materiales de la imagen.

Como aspectos negativos, cabe destacar primeramente el hecho de que aún no queda demasiado claro cómo obtener los espectros de referencia como medias sin que esto afecte demasiado durante la reconstrucción de la imagen. Por otro lado, hay aún aspectos del algoritmo que dependen de valores empíricos, con lo que la configuración de éste dependerá de la imagen a tratar. Finalmente, otro aspecto importante es que el método presenta grandes requerimientos computacionales, tanto en tiempo como en memoria, al depender de cálculos bastante pesados que además se efectúan de forma repetida.

6. VALORACIONES

Las prácticas en empresa han resultado ser una experiencia enormemente satisfactoria en prácticamente todos los sentidos. Principalmente debido a dos aspectos. Primeramente, por haberme permitido poder introducirme de una forma suficientemente sólida en el mundo de la programación, un aspecto que no sólo considero personalmente muy interesante, sino que además resulta enormemente relevante y de gran ayuda hoy en día. Tanto técnica como laboralmente son indiscutibles las ventajas que puede ofrecer tan sólo un pequeño conocimiento en la materia para el empleo eficiente de la clase de herramientas con las que alguien que trabaje regularmente con los SIG o con la teledetección puede suponer. No sólo para automatizar tareas, sino como se ha hecho en las prácticas, para desarrollar y probar

método y técnicas nuevas. Antes de las prácticas, ni me hubiera imaginado cómo se podría utilizar la programación en el ámbito dela teledetección. El lado negativo de este aspecto es la reafirmación de una opinión que ya tenía previamente, y es que me parece que el programa del máster debería considerar esta materia, la programación, de forma más seria, tanto por la enorme utilidad de estas técnicas como por las ventajas laborales que éstas pueden ofrecer a los futuros estudiantes.

Por otro lado, el hecho mismo de haber realizado las prácticas en una empresa presenta ya un valor propio. En la DLR me he encontrado con un ambiente muy profesional, y puesto que para mí ésta ha sido la primera experiencia laboral en un campo de mi interés, no puedo ignorar la suerte que he tenido en este aspecto. De este punto, son dos los aspectos más positivos los que me gustaría destacar. Primeramente, el propio ambiente de trabajo, enormemente profesional y competente en una institución investigadora, y, por otro lado y de manera simultánea, muy flexible, relajado y de trabajo bastante independiente. Creo que ambos aspectos ayudan a producir un ambiente laboral tanto productivo como agradable para los trabajadores y que facilita el aprendizaje.

Otro de los aspectos positivos de la experiencia ha sido el poder profundizar en la teledetección hiperespectral, un campo del que apenas conocía nada antes de las prácticas. Parece que éste está sufriendo un gran avance debido a la gran cantidad de aplicaciones que puede ofrecer. Personalmente me ha parecido mucho más flexible que la teledetección multiespectral, aunque también algo más compleja. Aunque me ha gustado conocer ideas nuevas en este ámbito, y además, hacerlo bajo un punto de vista orientado hacia un objetivo, lo que, a diferencia del aprendizaje desarrollado durante el estudio, me ha ayudado a centrarse en la comprensión de un problema determinado y en la búsqueda dde una solución concreta.

7. BIBLIOGRAFÍA

Nirmal Keshava (2003), A Survey of Spectral Unmixing Algorithms. *Lincoln Laboratory Journal*

Daniele Cerra, Rupert Muller, and Peter Reinartz (2012), Noise Reduction in Hyperspectral Images. *IEEE Geoscience and remote sensing letter, Vol. X, No. X, XX 2012*

Daniele Cerra, Rupert Muller, Jakub Bieniarz, and Peter Reinartz (2012), On the application of spectral unmixing for noise reduction. *Remote Sensing Technology Institute, DLR*

Charoula Andreou (2013), Estimation of the number of endmembers using robust outliner detection method. *Journal of selected topics in applied Earth observations and remote sensing.*

David Valencia (2007), Endmember extraction, spectral unmixing and target detection using ENVI and Matlab. *UNEX*

Pablo J. Martínex et al., (2006), Endmember extraction algorithms from hyperspectral images. *Universidad de Extremadura*

A. Plaza, P Martínez, J. Plaza, R. M. Pérez, P. L. Aguilar, M. C. Cantero (2004), Algoritmos de extracción de Endmembers en imágenes hiperespectrales. *Universidad de Extremadura*

8. ANEXO: FUNCIONES MATLAB

Con estas funciones y la librería hiperespectral de Matlab se pueden obtener resultados similares en cualquier PC, para ello, el nombre de los archivos de texto en formato .m que las contengan ha de ser el mismo al de las funciones. El código queda muy modulado al estar tan dividido en funciones individuales que realizan tareas muy concretas. De este modo, aunque individualmente sean relativamente sencillas, al integrarse en la aplicación son capaces de desarrollar una tarea compleja.

1. WATER_MASC

```
function masc = water_masc(y)
masc=y(:,:,150)>60;
b=[0 1 0;1 1 1;0 1 0];
masc=imopen(masc,b);
b=[1 1 1 1 1;1 1 1 1 1;1 1 1 1 1;1 1 1 1 1;1 1 1 1 1];
masc=imclose(masc,b);
masc(418:420,230:232)=1;masc(419,233)=1;masc(302:304,176)=1;masc(335,223)=1
;masc(126:128,321:323)=1;masc(305,425)=1;masc(279,484)=1;masc(315,468)=1;ma
sc(335,464)=1;masc(264,453:454)=1;masc(266,454)=1;masc(379,310)=1;masc(372:
373,348)=1;
imshow(masc);
end;
```

2. MASCALL

```
function y = mascall(y)
a=0;
[~,~,n]=size(y);
for i=1:n
    a=y(:,:,i);
    masc=water_masc(y);
    a(masc)=0;
    y(:,:,i)=a;
end
end
```

3. TI_TOTAL

```
% New values taken from the whole image.
% Stopping with first derivative of error to repect of the previous
iteration.
function [rec_array, res_array, spec] = ti_total(y,spec)
% Iteration number: set high number.
n = 30;
% Preallocating space for the outputs and filters.
[rows, columns, bands] = size(y);
rec_array = zeros(rows, columns, bands, n);
res_array = zeros(rows, columns, bands, n);
y_f3 = zeros(rows, columns, bands);
y_f5 = zeros(rows, columns, bands);
% Filters used:
f3 = ones(3) / (3*3);
for j = 1:bands;
    y_f3(:, :, j) = filter2(f3,y(:, :, j));
end
f5 = ones(5) / (5*5);
for j = 1:bands;
    y_f5(:, :, j) = filter2(f5, y(:, :, j));
end
```

```
% SAM-image used for the "dynamic filter".
[im_Sam, ~] = Sam_mean(y, y_f5);
for i = 1:n
    % Iterations: UBD and getting new coordinates.
    [rec] = UBD_2(y, spec);
    [res_fou] = imresmedfou2(y, rec);
    [r, c, b] = maxhyp_bands(res_fou);
    coord(i, 1) = r; coord(i, 2) = c; coord(i,3)=b;
    % Getting new spectra.
    a = coord(i, 1); b = coord(i, 2);
    if im_Sam(a, b) > 0.1
        spectra = squeeze(y_f3(r, c, :));
    else
        spectra = squeeze(y_f5(r, c, :));
    end
    spec = addspec(spec, spectra);
    % Outputs: reconstructed and error image.
    for k = 1:bands;
        rec_array(:, :, k, i) = rec(:, :, k);
        res_array(:, :, k, i) = res_fou(:, :, k);
    end
    % Regular exit:
    if i > 1;
        [~, medi] = res_res2(res_array);
        if medi(i-1) < 0; % Empirical value!
            disp('No more to go!');
            return
        end
    end
    % Emergency exit: in case the coordenates repeat.
    x = stopp1(spec);
    if x == true;
        disp('Endmember repeated!');
        return
    end
end
end
```

4. TI3_TOTAL

```
% New values taken from third band.
% Stopping with first derivative of error to repect of the previous
% iteration.
function [rec_array, res_array, spec] = ti3_total(y,spec)
% Iteration number: set high number.
n = 30;
% Preallocating space for the outputs and filters.
[rows, columns, bands] = size(y);
rec_array = zeros(rows, columns, bands, n);
res_array = zeros(rows, columns, bands, n);
y_f3 = zeros(rows, columns, bands);
y_f5 = zeros(rows, columns, bands);
% Filters used:
f3 = ones(3) / (3*3);
for j = 1:bands;
    y_f3(:, :, j) = filter2(f3,y(:, :, j));
end
f5 = ones(5) / (5*5);
for j = 1:bands;
    y_f5(:, :, j) = filter2(f5, y(:, :, j));
end
```

```
% SAM-image used for the "dynamic filter".
[im_Sam, ~] = Sam_mean(y, y_f5);
for i = 1:n
    % Iterations: UBD and getting new coordinates.
    [rec] = UBD_2(y, spec);
    [res_fou] = imresmedfou2(y, rec);
    [r, c] = maxhyp2(res_fou(:, :, 3));
    coord(i, 1) = r; coord(i, 2) = c;
    % Getting new spectra.
    a = coord(i, 1); b = coord(i, 2);
    if im_Sam(a, b) > 0.1
        spectra = squeeze(y_f3(r, c, :));
    else
        spectra = squeeze(y_f5(r, c, :));
    end
    spec = addspec(spec, spectra);
    % Outputs: reconstructed and error image.
    for k = 1:bands;
        rec_array(:, :, k, i) = rec(:, :, k);
        res_array(:, :, k, i) = res_fou(:, :, k);
    end
    % Regular exit:
    if i > 1;
        [~, medi] = res_res2(res_array);
        if medi(i-1) < 0; % Empirical value!
            disp('No more to go!');
            return
        end
    end
    % Emergency exit: in case the coordenates repeat.
    x = stopp1(spec);
    if x == true;
        disp('Endmember repeated!');
        return
    end
end
end
```

5. SAM_MEAN

```
% Makes a SAM image and calculates its mean between two hyperspectral
images.
% y: Original image.
% rec: reconstructed image.
function [im_Sam, mean] = Sam_mean (y, rec)
[m, n, ~, it] = size (rec);
im_Sam = zeros (m, n, it);
mean = zeros (it);
for k = 1:it;
    for i = 1:m;
        for j = 1:n;
            im_Sam (i, j, k) = hyperSam (squeeze (y (i, j, :)), squeeze
(rec (i, j, :, k)));
            im_Sam (isnan (im_Sam)) = 0; % Removes the NaN.
            mean (k) = real (sum (sum (im_Sam (:, :, k))) / (m*n));
        end
    end
end
end
```

6. UBD_2

```
% It just calculates the reconstructed image.
function [reconstructed] = UBD_2(M, spectra)
dims = size(M);
x = hyperConvert2d(double(M));
abundances = hyperNnls(x,spectra);
reconstructed = hyperConvert3d(spectra * abundances,
dims(1),dims(2),dims(3));
return;
end
```

7. IMRESMEDFOU2

```
% 1. Substracts one image -reconstructed- to another -original-.
% 2. Then runs a 5*5 average filter.
% 3. And it makes the Fourier transform.
function [err_fourier] = imresmedfou2(x,y)
% x = original image.
% y = reconstructed image.
% Dimensions and space preallocation.
dims_x = size(x);
dims_y = size(y);
resta = zeros(dims_x(1),dims_x(2),dims_x(3));
resta_av = zeros(dims_x(1),dims_x(2),dims_x(3));
err_fourier = zeros(dims_x(1),dims_x(2),dims_x(3));
% Substraction.
if dims_x(3) ~= dims_y(3)
    disp('Same band number required!')
else
    for i = 1:dims_x(3)
        resta(:,:,i) = abs(x(:,:,i)-y(:,:,i));
    end
minimo = min(min(resta(:,:,i)));
for i = 1:dims_x(3)
    resta(:,:,i) = resta(:,:,i)-minimo;
end
end
% Filtering.
filt = ones(5)/(5*5);
for i = 1:dims_x(3)
    resta_av(:,:,i) = filter2(filt,resta(:,:,i));
end
% Fourier transform.
for j=1:dims_x(3)
    err_fourier(:,:,j) = real(imfourier(resta_av(:,:,j)));
end
end
```

8. IMFOURIER

```
% Makes the Fourier transform of an image, filters it with Butterworth and
% comes back to space domain.
function yfourier = imfourier(y)
[~,n] = size(y);
fsize = 0.75*n; % Set Butterworth filter size as a proportion of the image
size.
y_fft = fft2(y);
y_ffts = fftshift(y_fft);
[f,c] = meshgrid(-n/2:-1+n/2,-n/2:-1+n/2);
bf = 1./(1+((f.^2+c.^2)/fsize).^2);
```

```
ybf = y_ffts.*bf;
ybfs = fftshift(ybf);
yfourier = ifft2(ybfs);
end
```

9. MAXHYP_BANDS

```
% Gives the corrdinates of the maximum value of a 4-D Matrix.
function [f,c,b] = maxhyp_bands(y)
[m,n,bands] = size(y);
f = 1; c = 1; b = 1;
max = y(1,1);
for k = 1:bands
    for i = 1:m
        for j = 1:n
            if y(i,j,k) > max
                max = y(i,j,k);
                f = i;
                c = j;
                b = k;
            end
        end
    end
end
end
```

10. MAXHYP2

```
% Gives the coordinates of the maximum value of a matrix.
function [f,c] = maxhyp2(y)
[m,n] = size(y);
f = 1; c = 1;
max = y(1,1);
for i = 1:m
    for j = 1:n
        if y(i,j) > max
            max = y(i,j);
            f = i;
            c = j;
        end
    end
end
end
```

11. ADDSPEC

```
% Add a new spectrum to an already existing spectral library.
% x: spectral library.
% y: new spectrum.
function x = addspec(x,y)
[a, b] = size(x);
[c, d] = size(y);
if a ~= c
    disp ('Dimensions must agree!');
else
    for i = 1:d
        x(:, b+i) = y(:, i);
    end
end
end
```

12. RES_RES2

```
% Calculates de average and the maximum of the differences between two
consecutive images.
% Used as stopping criterium with the error images.
function [maxi, medi] = res_res2(x)
[m,n,~,it] = size(x);
maxi = zeros(it-1);
medi = zeros(it-1);
y = zeros(m,n,it-1);
if ndims(x) == 4;
for i = 1:it-1;
    y(:,:,i) = x(:,:,3,i+1)-x(:,:,3,i);
    maxi(i) = abs(max(max(y(:,:,i))));
    medi(i) = abs(mean2(y(:,:,i)));
end
elseif ndims(x) == 3;
    [~,~,it] = size(x);
for i = 1:it-1;
    y(:,:,i) = x(:,:,i+1) - x(:,:,i);
    maxi(i) = abs(max(max(y(:,:,i))));
    medi(i) = abs(mean2(y(:,:,i)));
end
else
    print ('Check dimensions!');
end
end
```

13. STOPP1

```
% Says when two spectra are the same one.
function x = stopp1(spec)
[~,l] = size(spec);
count = 0;
for i = 1:l;
    for j = 1:l;
        if spec(:,i) == spec(:,j);
            count = count+1;
        end
    end
if count == 1; % All spectra from a spectral library are different.
    x = false;
else
    x = true; % At least two spectra are the same one.
    return
end
count = 0;
end
end
```

14. N_RMSE

```
% Makes RMSE along all the iterations of an hyperspectral image with
respect to the original.
% y: original image.
% rec: whole reconstructed image.
function [r, r_mean] = n_rmse(rec,y)
[~,~,bands,it] = size(rec);
r = zeros(it,bands);
```

```
r_mean = zeros(it);
for i = 1:bands;
    for j = 1:it;
        r(j,i) = rmse(rec(:,:,i,j),y(:,:,i));
        r_mean(j) = sum(r(j,:))/(bands);
    end
end
end
```

15. RMSE

```
function r=rmse(data,estimate)
% Function to calculate root mean square error from a data vector or matrix
% and the corresponding estimates.
% Usage: r=rmse(data,estimate)
% Note: data and estimates have to be of same size
% Example: r=rmse(randn(100,100),randn(100,100));
% delete records with NaNs in both datasets first
I = ~isnan(data) & ~isnan(estimate);
data = data(I); estimate = estimate(I);
r=sqrt(sum((data(:)-estimate(:)).^2)/numel(data));
```

16. SAMMATRIX

```
% Makes a SAM matrix between all the spectra of a library.
function [matrix] = Sammatrix(speclib)
[m,n]=size(speclib);
matrix = zeros(m,n);
for i=1:n
    for j=1:n
        matrix(i,j)=hyperSam(speclib(:,i),speclib(:,j));
    end
end
end
```