ESCUELA TÉCNICA SUPERIOR DE INGENIEROS
INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Técnicas de procesado de señal aplicadas a la espectroscopía laser

(Signal processing techniques applied to laser spectroscopy)

Para acceder al Título de

Graduado en
Ingeniería de Tecnologías de Telecomunicación

Autor: Beatriz Martínez Calvo

Marzo - 2015

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Beatriz Martínez Calvo
Director del TFG: Adolfo Cobo Garcia
Título: "Técnicas de procesado de señal aplicadas a la espectroscopía laser"
Title: "Signal processing techniques applied to laser spectroscopy"

Presentado a examen el día:

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Adolfo Cobo García
Secretario (Apellidos, Nombre): Jesús Ramón Pérez López
Vocal (Apellidos, Nombre): María Ángeles Quintela Incera

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente	Fdo.: El Secretario
Fdo.: El Vocal	Fdo.: El Director del TFG (sólo si es distinto del Secretario)
Vº Bº del Subdirector	Trabajo Fin de Grado Nº (a asignar por Secretaría)

Trabajo realizado parcialmente con los medios aportados por el proyecto "OFS4" (TEC2013-47264-C2-1-R)

Índice de contenidos

1. Introducción

1.1 Contexto

Este trabajo gira en torno a la técnica LIBS (Laser-induced breakdown spectroscopy) o espectroscopia de plasma inducido por láser, utilizando dicha técnica como método para determinar la composición de la materia.

LIBS consiste en generar un plasma mediante ablación láser y analizar espectroscópicamente su emisión. En principio, LIBS pueden analizar cualquier materia, independientemente de su estado físico, ya sea sólido, líquido o gas. Debido a que todos los elementos emiten luz de frecuencias características cuando se excita a temperaturas suficientemente altas. Si se conocen los constituyentes de un material a ser analizado, LIBS se pueden usar para evaluar la abundancia relativa de cada elemento constituyente.

Para llevar a cabo esta técnica se usa un pulso láser de corta duración y alta potencia centrado en un área pequeña de la superficie de la muestra, causando la ablación de una cantidad muy pequeña de la muestra, en el rango de nanogramos a picogramos, lo que genera una pluma de gas parcialmente ionizado conocido como plasma, que alcanza temperaturas de entre 5000k-20000k, esta excitación de los átomos produce líneas de emisión atómica que son las que debemos medir.

1.2 Motivación

Cuando se utiliza la técnica LIBS en aplicaciones y problemas reales, surgen necesidades concretas que dependen de cada aplicación. Una necesidad muy común es la de automatizar las medidas. Ya que en numerosas aplicaciones es necesario muestrear exhaustivamente en diferentes puntos de la muestra, esto requiere una automatización de los movimientos a realizar para tomar las medidas, ya que manualmente es un trabajo tedioso y requiere más tiempo.

Una de las grandes aplicaciones de LIBS es obtener la composición química de una muestra, para obtener dicha información se lleva a cabo un estudio minucioso de los espectros generados, para aumentar la precisión de estas medidas es necesario eliminar el ruido de fondo ya que provoca que los picos espectrales estén más o menos elevados respecto a su posición original.

Además estudiar los espectros obtenidos es un trabajo muy tedioso debido a la cantidad de espectros obtenidos en una sola muestra, por lo que conviene realizar una

organización más visual. Para ello se crea una GUI con matlab, organizando así los espectros y también disponer de los datos más usados a simple vista.

1.3. Objetivos del trabajo

Los objetivos de este trabajo fin de grado se centran en dos aspectos concretos. En primer lugar, el procesado de imágenes de muestra por ordenador mediante la herramienta Matlab, para llevar a cabo el guiado del láser a lo largo de una trayectoria muestreando diferentes puntos de una muestra, utilizando la técnica de libs. Para poder determinar la trayectoria que debe seguir el láser lo primero que se ha de conseguir es detectar los bordes de la capa que se quiere estudiar, la capa de interés será determinada por el usuario, que también impondrá el punto de partida, la distancia entre dos puntos consecutivos y la profundidad respecto al borde de la capa.

El segundo objetivo de este trabajo es implementar un algoritmo de eliminación del background, propuesto en 2009 por Lanxiang Sun y Haibin Yu. Para estimar el background el algoritmo utiliza los diferentes mínimos del espectro y un umbral calculado a partir de la desviación estándar y la media de la varianza de los mismos.

Además se quiere realizar una interfaz gráfica de matlab para poder visualizar los espectros obtenidos así como algún dato de ellos que sea importante para lo que se desee estudiar con dichos espectros.

2. Implementación

2.1 Instrumentación

El montaje que tenemos en el laboratorio para llevar a cabo la técnica LIBS es el siguiente:

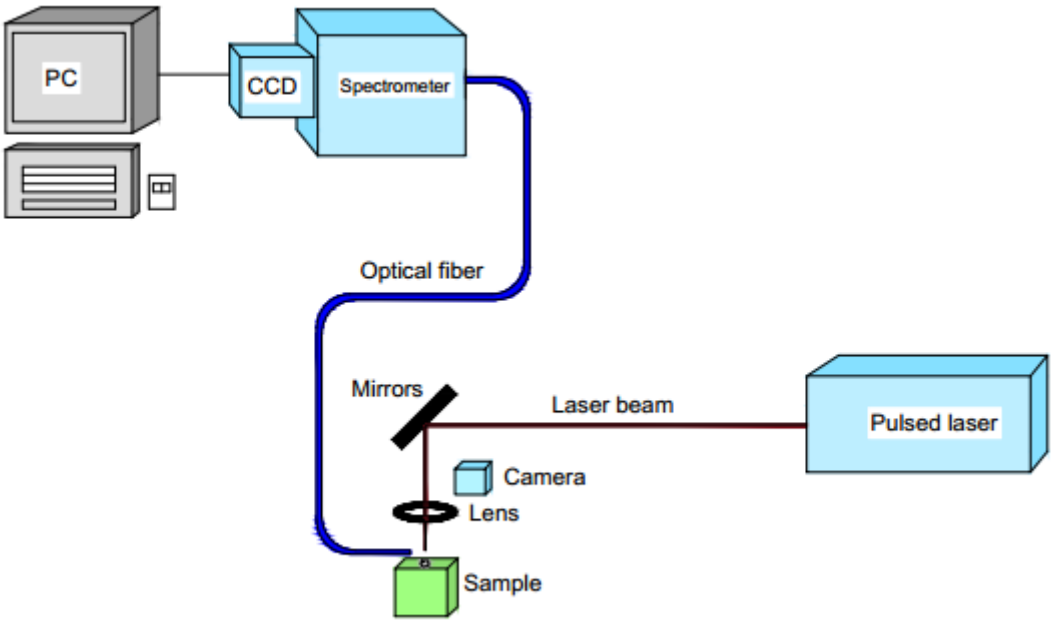


Ilustración 1 Montaje laboratorio

En primer lugar con un láser de Nd:YAG como fuente de excitación. Se utiliza el láser LOTIS LS-2147, que emite a una longitud de onda de 1064nm, en el infrarrojo cercano. Este láser permite obtener pulsos muy estrechos temporalmente y con elevada energía que es lo que se busca en la técnica LIBS. Esto se consigue con la técnica de Q-switching, en la que el láser almacena una gran cantidad de energía a partir de la lámpara de bombeo, para luego liberarla en un pulso de gran potencia y unos nanosegundos de duración. Para focalizar el pulso en la muestra tenemos una serie de espejos y una lente.

Como se puede apreciar en la figura1 el montaje incluye una cámara (Guppy FC-33 con conexión IEEE1394 y resolución de 640x480 pixeles en color), que es la que

permite recoger la imagen de la muestra sobre la que después trabajaremos para llevar a cabo el guiado del láser. La cámara recoge una imagen de la superficie de la muestra a través de la óptica de focalización del láser (espejo y lente), lo que aporta un gran aumento: un milímetro de superficie de la muestra ocupa 82 píxeles en la imagen.

La recolección de la luz que se genera en el plasma, como consecuencia de la interacción del láser con el material debe ser realizada de la manera más eficiente posible. En nuestro caso se ha utilizado una fibra óptica para recoger directamente la luz del plasma y conducirla al monocromador, un ACTON SP2300. El tiempo que transcurre desde el disparo del láser hasta el momento en el que se produce el plasma y se recoge debe ser determinado para cada caso para poder tener una buena relación entre la señal y el ruido, y es en todo caso de unos pocos microsegundos.

A la salida del monocromador contamos con el CCD PI-MAX3, la función primaria de un detector tipo CCD es la de convertir los fotones de una determinada longitud de onda en una señal eléctrica que pueda ser manipulada por un sistema electrónico digital. Al estar situado el CCD en el plano focal del monocromador, se obtiene una imagen bidimensional cuyo eje horizontal es la longitud de onda, y cuyo eje vertical se corresponde con la posición vertical en la rendija de entrada para la luz. Sumando todos los píxeles verticales se obtienen un espectro “convencional”, en el que cada longitud de onda tiene asociado un valor de intensidad luminosa.

La señal generada es enviada a un ordenador, permitiendo de esta manera visualizar los espectros y trabajar con ellos. Puede encontrarse más información sobre la técnica LIBS en las referencias [3] y [5].

2.2 Guiado del láser

Para llevar a cabo el guiado del láser se han utilizado imágenes de muestra, tomadas por el CCD, de elementos sobre los que se quiere aplicar la técnica LIBS. Un ejemplo de las imágenes de las que disponemos son las mostradas en la figura1.

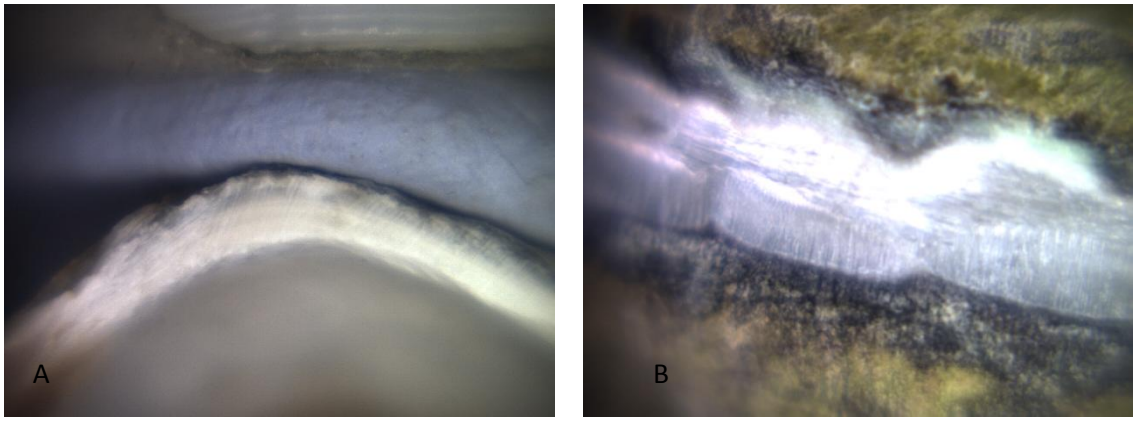


Ilustración 2 Ejemplo imágenes de muestra, una lapa (A) y un caracol (B)

Una vez tenemos las imágenes de muestra tenemos que procesarlas para poder guiar el láser por la capa deseada. Como ya comentamos previamente el programa que vamos a utilizar para procesar la imagen es Matlab.

Para llevar a cabo el guiado el usuario debe en primer lugar cargar la imagen en Matlab, eso se hace con una función muy simple que es *imread.*

```
f=imread('prueba.png');
```

Ahora debemos indicar de alguna forma cual es la capa de interés para poder a partir de ahí sacar la trayectoria del láser. Además debe dar el valor de delta, que será la distancia entre dos puntos consecutivos y la profundidad a la que debe estar el punto respecto a los bordes. La función que vamos a hacer debe tener como datos de entrada delta y profundidad.

```
function [x,y]=trayectory(DELTA,PROFUNDIDAD)
```

En la figura2 se puede observar un ejemplo donde ver bordes, delta y profundidad de una imagen.

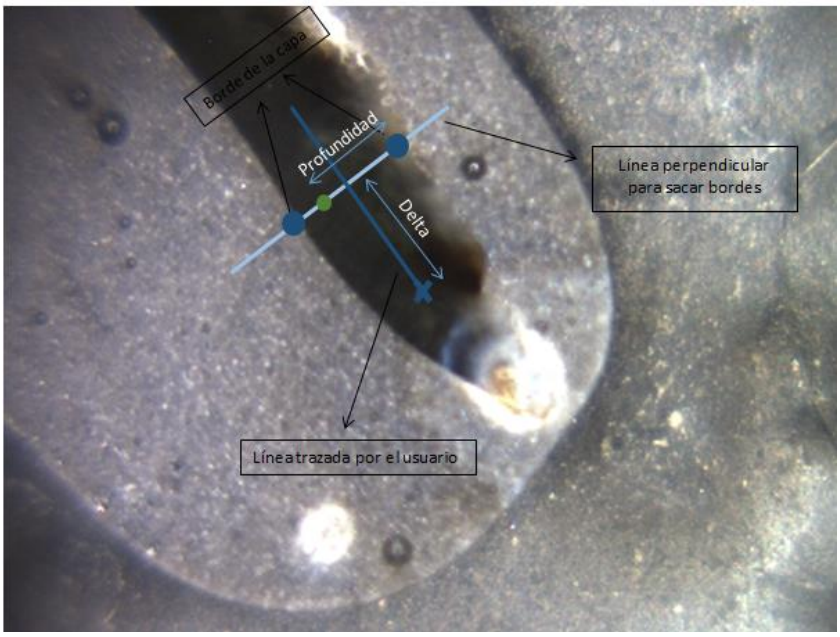


Ilustración 3 Diferentes parámetros del guiado

El usuario nos indica la capa de interés mediante la función de Matlab *ginput,* esta función permite que mediante el cursor del ratón el usuario determine una línea, esta línea nos indicara por donde va la capa elegida. *Ginput* nos proporciona las coordenadas de los puntos que ha dado el usuario para trabajar con ellos.

```
[x_sig,y_sig]=ginput(2)
```

En este caso hacemos un *ginput* 2 porque son los puntos que necesitamos para saber cuál es la capa y el punto de inicio. En x_sig tenemos las coordenadas x de los puntos que ha dado el usuario y en y_sig las de y. Asi en tenemos el punto inicial del láser en el punto (x_sig(1),y_sig(1)).

En un primer intento se intentó determinar los bordes de las capas sacando el mínimo y/o máximo de la intensidad de la imagen en la línea perpendicular a la trazada por el usuario. Para ello se utilizó la función de Matlab *improfile* que te indica la intensidad de los pixeles en las coordenadas dadas. Se pensó en sacar mínimos o máximos porque las capas están definidas por un cambio de intensidad o color. El problema fue que el mayor cambio de intensidad entre dos valores consecutivos no tenía por qué producirse en el borde de la capa sino que podía ocurrir en otro punto.

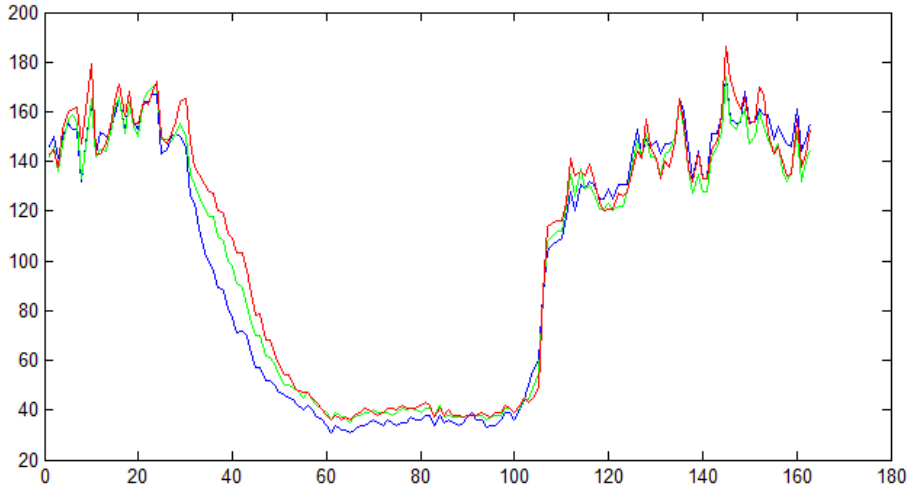


Ilustración 4 intensidad línea sacar bordes

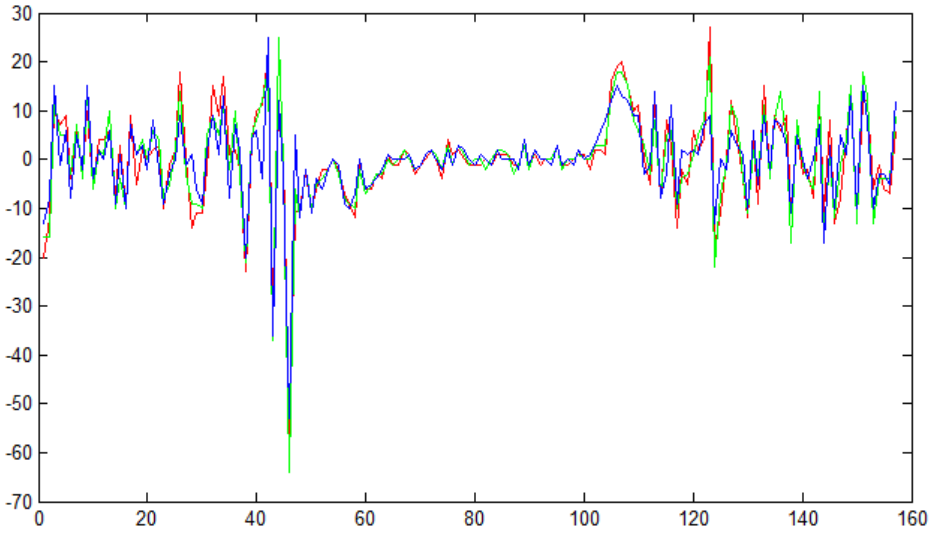


Ilustración 5 Derivada de la intensidad

En la primera imagen se ven las intensidades de los colores en la línea trazada para obtener los bordes de la capa. En la segunda imagen se ha aplicado la derivada para encontrar los máximos o mínimos de intensidad que indicasen el borde. El problema se basa en encontrar cual de esos mínimos y máximos nos indica donde están los límites de la capa, como no se encontró una fácil solución se recurrió a buscar otro método.

Este método se basa en el concepto de *magic wand* que utilizan otros programas como Photoshop. Esta herramienta nos permite seleccionar todos los pixeles del color del primer pixel seleccionado, cuenta con un parámetro que es la tolerancia que es el grado de similitud de color que tiene que encontrar Photoshop para seleccionar o no un píxel. Cuanto más bajo sea el número menos píxeles incluirá la selección.

Para llevar cabo esto en Matlab primero debemos coger unos píxeles de referencia, los píxeles que cogemos en este caso son alguno que pertenezca a la línea indicada por el usuario ya que esos píxeles seguro que pertenecen a la capa. Lo que vamos a hacer ahora es pasar a tener un vector con todos los píxeles similares a los de referencia que cumplen con una tolerancia establecida. Para ello utilizamos la diferencia de color entre los píxeles de referencia y el resto de píxeles de la imagen y comparamos con el valor de tolerancia proporcionado.

$$(C_r - Ref_r)^2 + (C_g - Ref_g)^2 + (C_b - Ref_b)^2 \leq Tolerancia^2$$

En Matlab sería lo siguiente:

```
color_mask = color_mask | ...
((c_r - ref_r) .^ 2 + (c_g - ref_g) .^ 2 + ...
    (c_b - ref_b) .^ 2) <= tolerancia ^ 2;
```

Color_mask se trata simplemente de una matriz de 0´s lógicos para poder hacer una or con la diferencia de color y tener una matriz binaria, en la que tendremos 1 donde pueda estar la capa y 0 en el resto.

A partir de ahí mediante la utilización de la función *bwlabel* sacamos los diferentes objetos que componen la imagen.

```
[objects, count] = bwlabel(color_mask, 8);
```

La función *bwlabel* en este caso busca objetos presentes en la imagen con conectividad 8, en objects tendremos una matriz, de las mismas dimensiones de color_mask, en la que tenemos etiquetados los distintos objetos. Count nos indica el número de objetos que se han encontrado.

Ahora se trata de comprobar cuál de los objetos anteriores pertenece a la capa deseada y crear una máscara en blanco y negro como la que se muestra en la figura 3.

Ilustración 6 En la imagen de la izquierda observamos la original y en la derecha después de aplicar la mascara

Una vez tenemos la imagen en blanco y negro, blanco para la zona de interés y negro para el resto, ya solo se trata de ir encontrando los bordes de esa zona blanca, para ello sacamos la línea perpendicular a la línea dada por el usuario ya que donde deberíamos encontrar los bordes. Esta línea perpendicular la hallamos a una distancia delta porque es donde debe estar el próximo punto.

Ahora como sabemos que la intensidad del color blanco es 1 y la del negro es 0, aplicamos un *improfile* para sacar las intensidades de los píxeles de la línea perpendicular. Así sacamos los bordes con solo saber cuál son las posiciones del primer y último valor a 1 de la intensidad en dicha línea.

```
[CX,CY,C,xi,yi] =improfile(f,xi,yi)
```

Donde en C tenemos las intensidades de los puntos y en CY y CX las coordenadas.

Con los valores de los bordes ya solo queda sacar donde debe estar el próximo punto utilizando la profundidad. Simplemente tenemos que multiplicar la distancia entre los bordes por la profundidad para así saber a qué distancia está el punto.

Ya solo queda repetir encontrar el resto de puntos a los que se va a dirigir el láser, para ello solamente hay que repetir los pasos de sacar la recta perpendicular a la trayectoria, es decir la línea que une los 2 puntos anteriores que ha seguido el láser. A partir de aquí todo es una repetición de los pasos anteriores para encontrar los bordes y el punto exacto.

En la figura 7 se muestra una imagen en la que podemos apreciar la trayectoria de puntos que sigue el láser en color azul, los bordes de la capa en rojo y verde y por último en amarillo el punto actual.

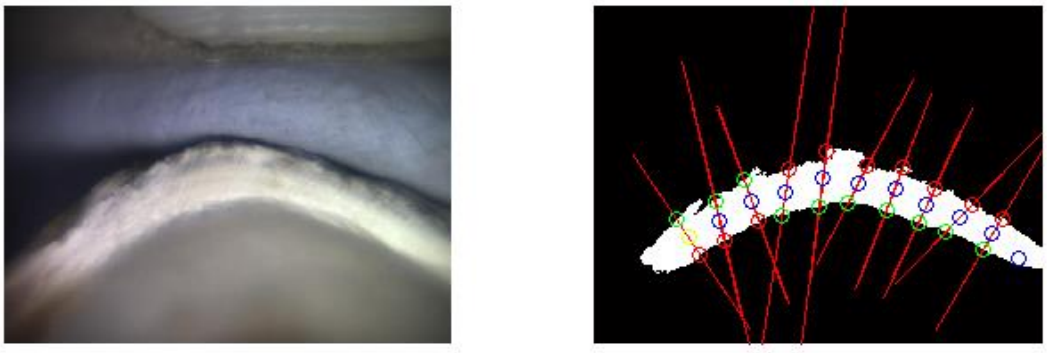
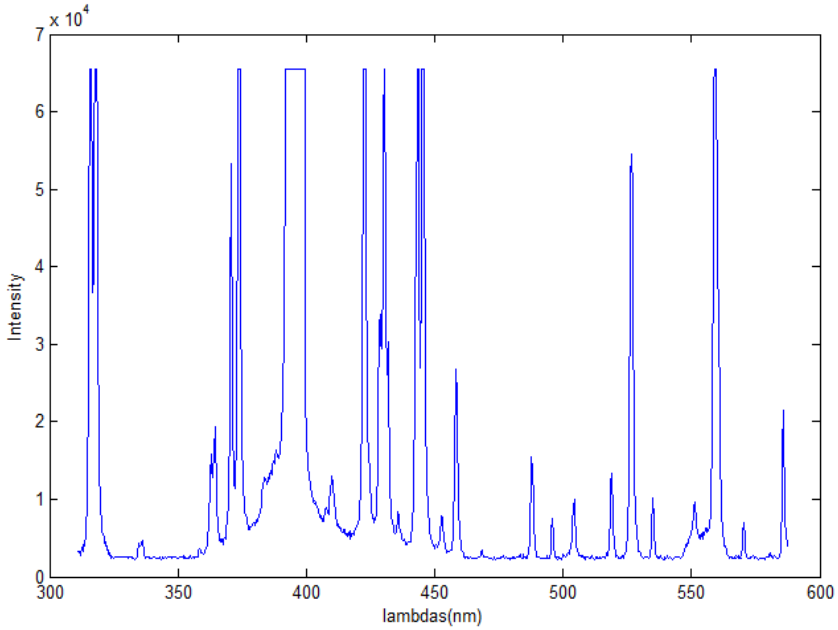


Ilustración 7 Trayectoria del láser

2.3 Eliminación del ruido de fondo

Una de las aplicaciones de la técnica LIBS es determinar la composición de la materia mediante los picos de los espectros recogidos, el problema surge en que estos espectros tiene un alto ruido de fondo. La eliminación de este ruido no es un trabajo fácil porque depende de la longitud de onda, de la muestra y de las condiciones de formación del plasma. Podemos ver un ejemplo de diferentes espectros recogidos con LIBS en la figura4.

A



B

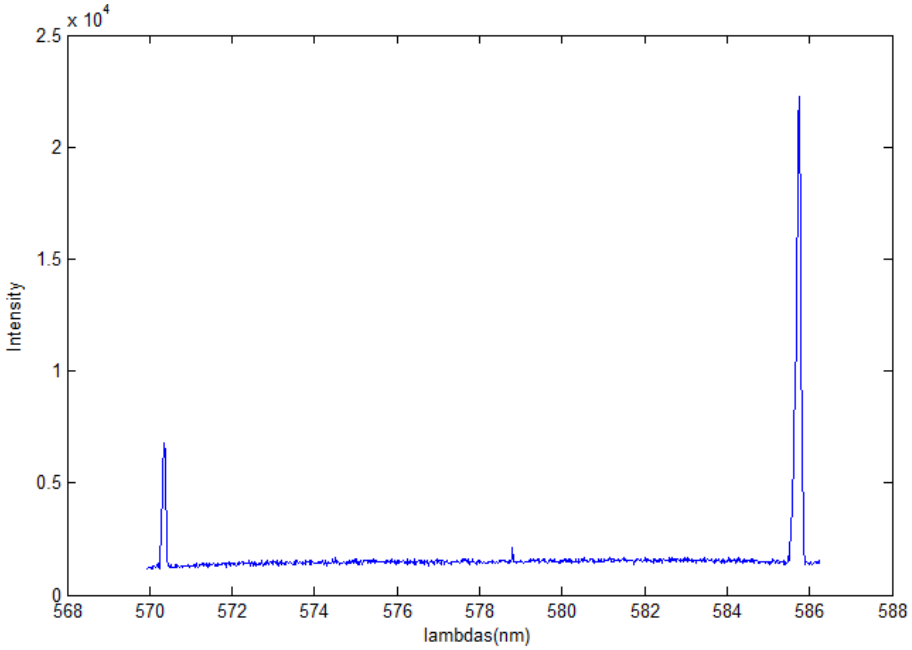


Ilustración 8 Ejemplos espectros LIBS utilizados a) espectro complejo y b) espectro simple

Se han publicado varios artículos sobre posibles métodos de reducirlo, en nuestro caso vamos a usar el método que proponen Lanxiang Sun y Haibin Yu en 2009. En este caso también utilizamos Matlab para implementar dicho método.

Disponemos de varios espectros tomados con LIBS, para probar el funcionamiento del método. Una vez tenemos uno de dichos espectros cargado con sus diferentes intensidades a diferentes lambdas tenemos que encontrar todos los mínimos. Hemos sacado los mínimos simplemente utilizando un ciclo for para ir comparando un valor con los otros 2 valores próximos a él y así saber si se trata de un mínimo o no.

```
for k=2:length(sp)-1
if sp(k)<sp(k-1) && sp(k)<sp(k+1)
lamda(i)=lambdas(k);
Imin(i)=sp(k);
i=i+1;
end
end
```

En lambdas tenemos las longitudes de onda y en sp las intensidades respectivas a las lambdas. Luego cargamos los mínimos en Imin y en lamda su respectiva longitud de onda.

Una vez tenemos los mínimos debemos calcular el umbral que es el que nos va a ayudar a determinar si esos mínimos forman parte del background o no. En primer lugar calculamos la varianza entre mínimos adyacentes.

$$r_j = |(I_{j+1} - I_j)/(\lambda_{j+1} - \lambda_j)|$$

```
for j=1:length(lamda)-1
r_j(j)=abs((Imin(j+1)-Imin(j))/(lamda(j+1)-lamda(j))); %varience
ratio entre dos minimos próximos
end
```

El umbral propuesto para este método depende de la desviación estándar, de la media de la varianza de los mínimos y de un factor de estimación del background.

$$\theta = k_t.\bar{r}$$

$$k_t = \frac{K_\delta}{RSD}$$

Donde $K_\delta$es la estimación del background es un parámetro que depende de la complejidad del espectro tomara unos valores más grandes o más pequeños. RSD es la desviación estándar relativa de r_j (varianza de 2 mínimos adyacentes) y $\bar{r}$ es la media de r_j. Así en este umbral el único término que no queda totalmente definido es la estimación del background, por lo que dependiendo de qué valor se le dé será una mejor o peor estima.

```
RSD=std(r_j)/mean(r_j); %relative standard desviation
kt=k/RSD; %RSD-> desviacion de r_j y k-> background stimation
factor
tetha=kt*mean(r_j);% threshold
```

El algoritmo sigue comprobando de izquierda a derecha si los mínimos son mayores o menores que el umbral, en caso de sobrepasarlo ese mínimo se descarta. Una vez tenemos todos los mínimos por la izquierda, se pasa a calcular por la derecha si dichos mínimos son menores que $-\theta$, si son menores se descartan y si no pasan a formar parte de los puntos que consideraremos de background.

```
for j=1:length(lamda)-1
r_j(j)=(Imin(j+1)-Imin(j))/(lamda(j+1)-lamda(j));
if r_j(j)>tetha
    Imin(j+1)=Imin(j); lamda(j+1)=lamda(j);
else
    l=l+1;
    I_L(l)=Imin(j+1); lamda_l(l)=lamda(j+1);
end
j=j+1;
end
```

En I_L guardamos los mínimos que por la izquierda se ajustan al umbral.

```
for q=l:-1:2
r_j_l(q)=(I_L(q-1)-I_L(q))/(lamda_l(q-1)-lamda_l(q)); %varience ratio
entre dos mínimos próximos

if r_j_l(q)<-tetha
    I_L(q-1)=I_L(q); lamda_l(q-1)=lamda_l(q);
else
    p=p+1;
    I_R(p)=I_L(q-1); lamda_r(p)=lamda_l(q-1);

end
q=q-1;
end
```

En I_R guardamos los mínimos que por la derecha se ajustan al umbral y en lamda_r guardamos sus lambdas correspondientes. Estos son los puntos que definitivamente vamos a considerar que forman parte del ruido de background. Una vez tenemos dichos puntos lo que debemos hacer es obtener el polinomio que pasa

por esos puntos para después restárselo al espectro original y tener el espectro sin el ruido de fondo.

```
x=lamda_r'; % lambdas que han quedado tras procesar por la derecha
y=I_R';  % las intensidades de los mínimos
% lambdas,sp es el espectro original
```

Para obtener dicho polinomio hemos utilizado diferentes funciones de matlab para comparar cual da mejor resultado. En primer lugar utilizamos *spline* pero no sirve porque se trata de una interpolación cubica y el ruido se comporta como una línea, el resultado con *spline* sería el siguiente:

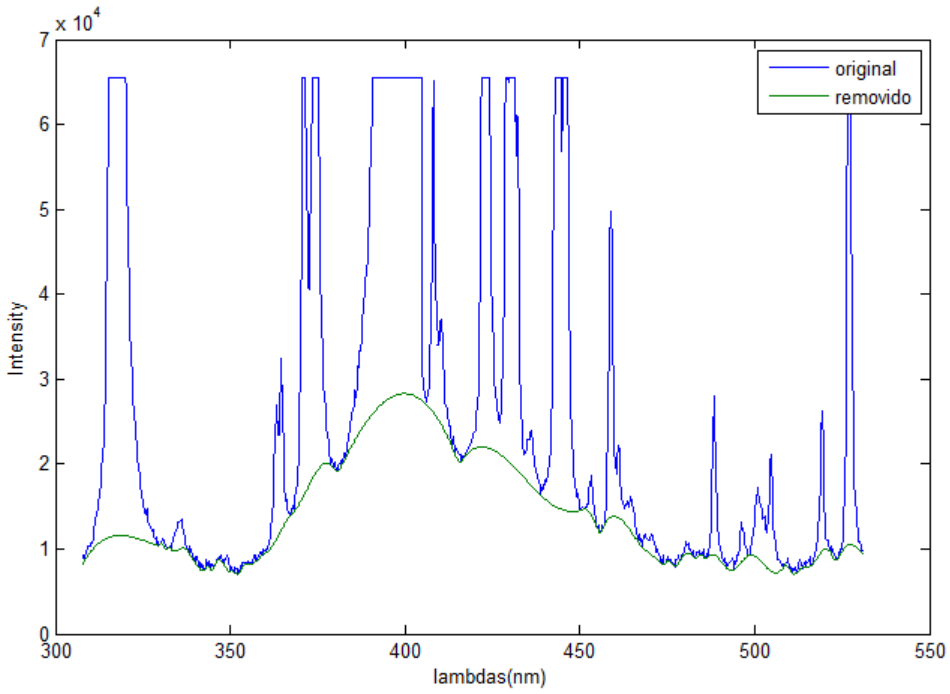


Ilustración 9 Resultado obtenido con spline

Una solución es realizar un *spline* de segundo orden mediante los siguientes comandos:

```
mi_spline =spapi(2,x,y,lambdas);  %esto es una spline de segundo orden
rem = fnval(mi_spline,lambdas);
```

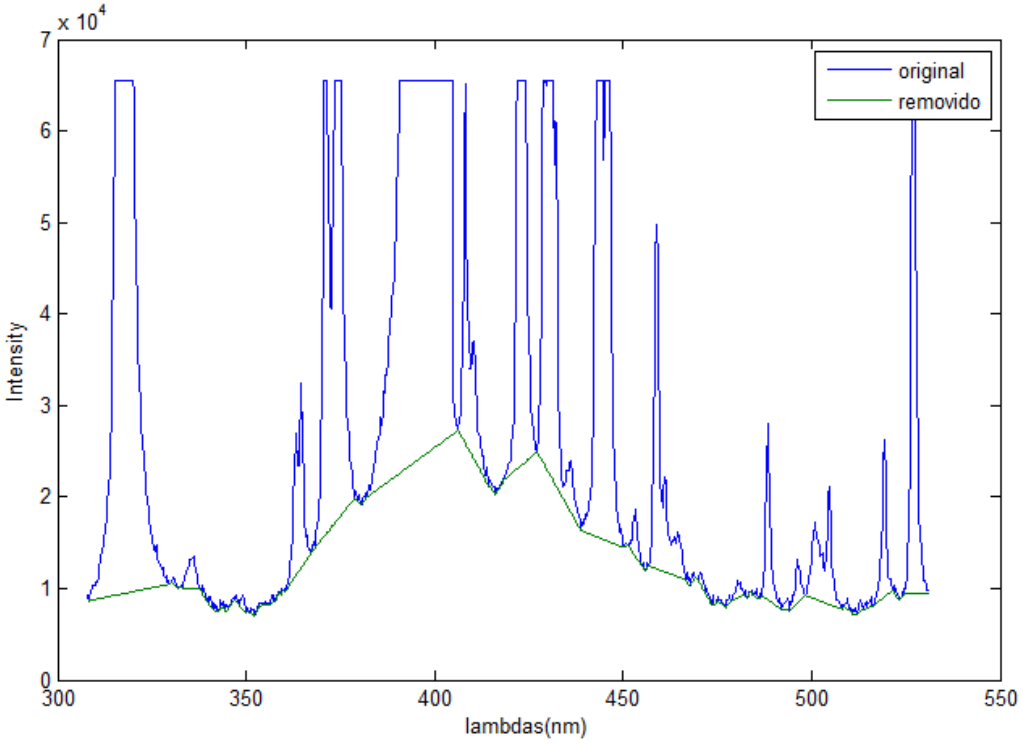


Ilustración 10 Resultado spline segundo orden

La siguiente función con la se ha probado se trata de *fit,* obteniendo el siguiente resultado:

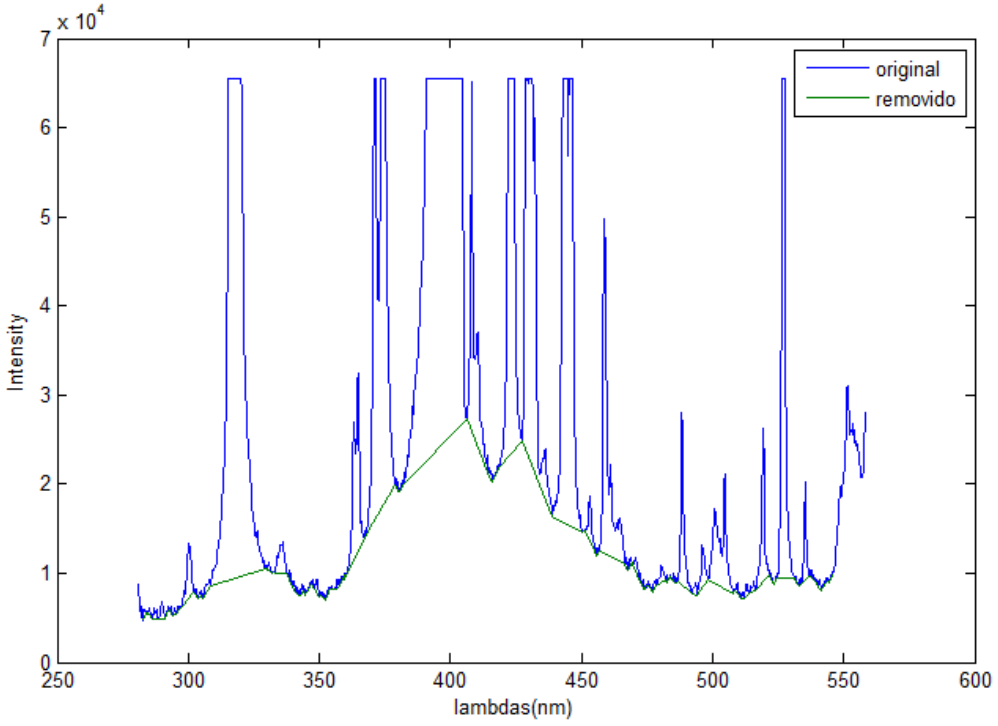


Ilustración 11 Resultado obtenido con funcion fit

Estos resultados son más aproximados a lo que se busca ya que son lineales y no cúbicos. La última función utilizada ha sido *interp1* que proporciona una interpolación lineal de los datos dados.

```
rem=interp1(x,y,lambdas)
```

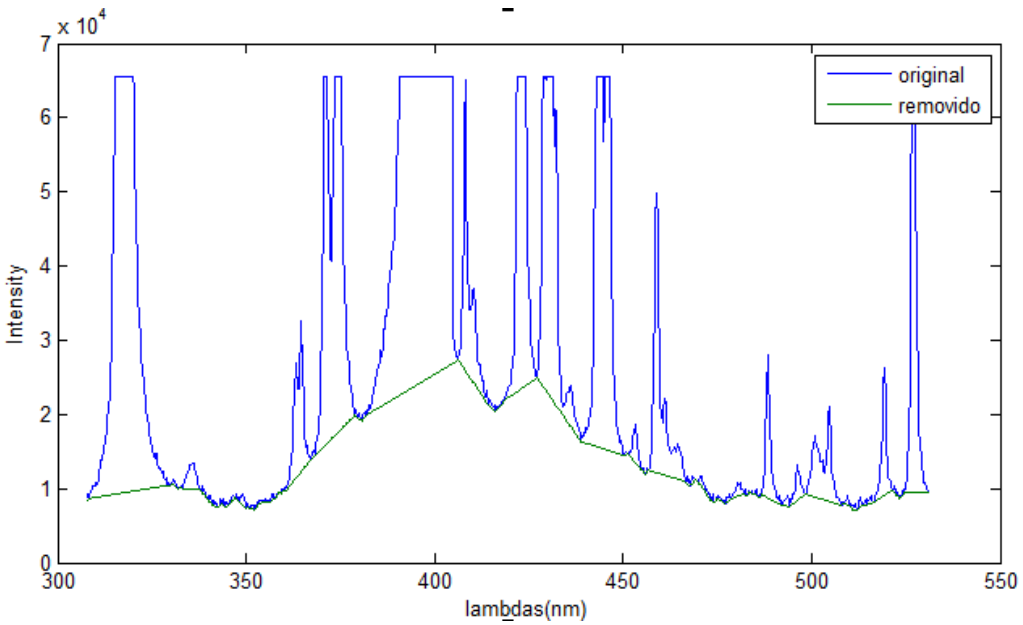


Ilustración 12 Resultado obtenido utilizando interp1

2.4 Interfaz gráfica

Otra línea de trabajo en este proyecto ha sido el desarrollo de un interfaz gráfico para mostrar los espectros capturados en un experimento LIBS. En este experimento se estudian las conchas de lapas y caracoles para saber la temperatura del agua durante el crecimiento del organismo, y para ello se necesita saber que valores de carbonato calcio y de magnesio tienen dichas conchas.

A la hora de realizar el interfaz lo que se busca es en primer lugar poder ir visualizando los espectros obtenidos de los diferentes puntos del espacio de las conchas, para ellos se han puesto las opciones de poner directamente el espectro que se quiere visualizar y de poder moverse a izquierda y derecha de un punto. Tal y como se observa en la imagen, se dispone de botones de izquierda y derecha y de un botón de enter para aceptar cualquier cambio en el número de punto insertado.

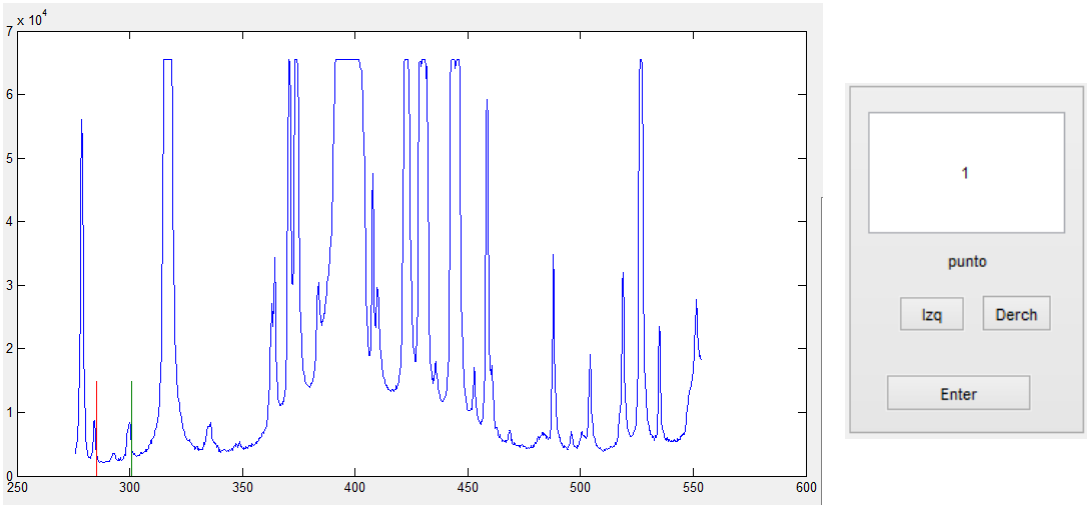


Ilustración 13 Interfaz

En la representación del espectro se dispone de dos marcadores, uno rojo y otro verde para determinar la posición de magnesio y calcio respectivamente.

A continuación se muestra una serie de datos importantes. Primero el número de pulsos y de cleaningshots, que son respectivamente el número total de pulsos y desde pulso empezamos a contabilizarlos. También se muestran las lambdas de calcio y magnesio y el ratio magnesio/calcio. Tanto los datos del número de pulsos/cleaningshots como las lambdas se pueden cambiar simplemente cambiando los valores y dando al ok.

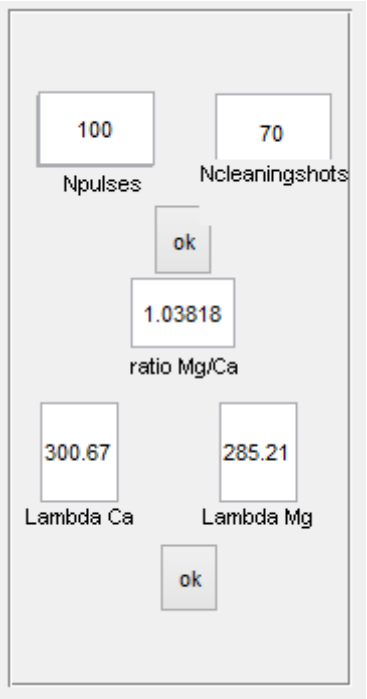


Ilustración 14 Parámetros que muestra el interfaz

Otra opción que da el interfaz es cambiar el directorio de donde se leen los espectros, se dispone de dos opciones, escribir el directorio o elegir el directorio de un menú desplegable.

Ilustración 15 Cambiar directorio

Por último se dispone de otra figura en la que se puede observar el ratio Mg/Ca promedio.

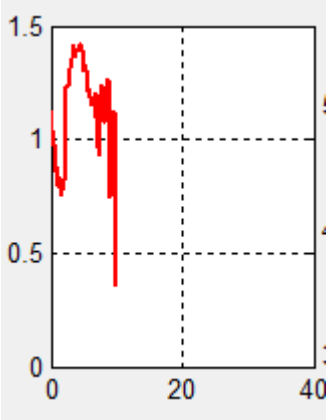


Ilustración 16 Ratio Mg/Ca

En el interfaz se cargan diferentes conjuntos de espectros obtenidos de diferentes iteraciones en diferentes objetos. Por defecto se al abrir el interfaz están cargados los espectro de una ruta determinada, que se puede cambiar, además se carga directamente el espectro del primer punto.

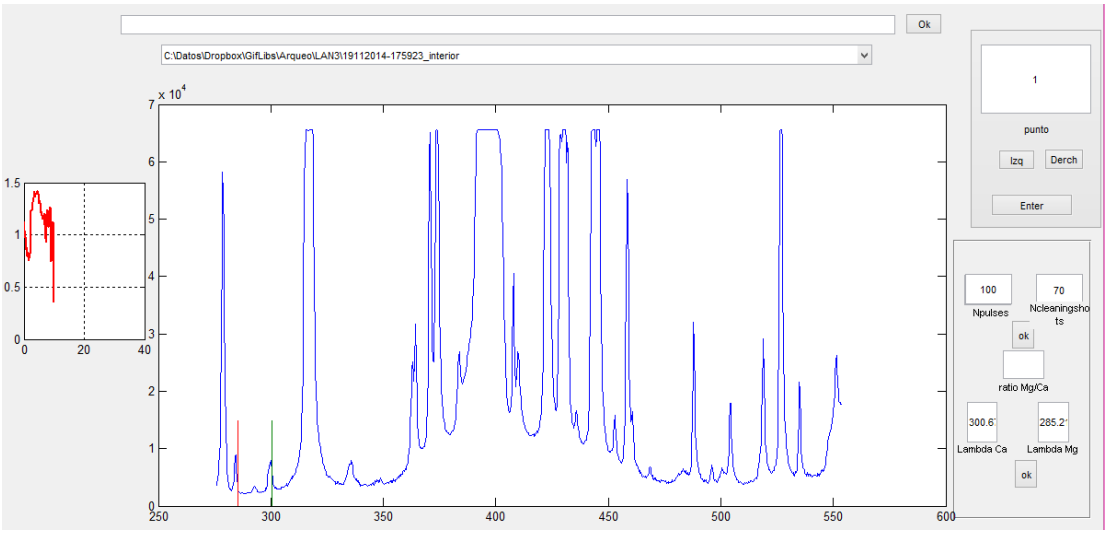


Ilustración 17 Interfaz

Se ha probado el funcionamiento con diferentes conjuntos de espectros, así como cambiando el número de pulsos y las lambdas de Ca y Mg, los resultados obtenidos en todos los casos son los esperados.

3. Resultados Obtenidos

3.1 Resultados del guiado láser

Para probar el funcionamiento del algoritmo se han usado varias imágenes de lapas, caracoles y algún trozo de plástico.

La utilización de lapas y caracoles se basa en el interés que tiene saber los valores de carbonato calcio y magnesio que componen sus conchas aporta información muy valiosa sobre la temperatura del agua durante el crecimiento del organismo. Así dos posibles aplicaciones de estos conocimientos pueden ser detectar los meses del año en el que se recogieron y reconstruir la evolución de la temperatura en épocas pasadas.

La primera imagen es la de una concha de lapa *(Patella Vulgata)* que se ha cortado transversalmente, observándose el perfil de la concha que permite discernir las diferentes capas que lo componen, con ligeras diferencias de composición. La imagen de la derecha corresponde a otro ejemplar de lapa, que en este caso ha sido embutido en resina epoxy y cordado posteriormente. El aspecto negruzco se debe a que las conchas presentan dos capas diferenciadas del mismo material pero en diferente estado de cristalización (aragonito y calcita), y en ciertos ejemplares ésta última puede tener este color.

A la hora de elegir la tolerancia se ha intentado poder solamente variar entre dos valores haciendo así más fácil para el usuario, los valores que se han elegido son 0.2, para casos en el que las imágenes tienen la capa bien definida por un cambio de intensidad fuerte del color. El otro valor es 0.5 para casos en que la capa esta menos definida.

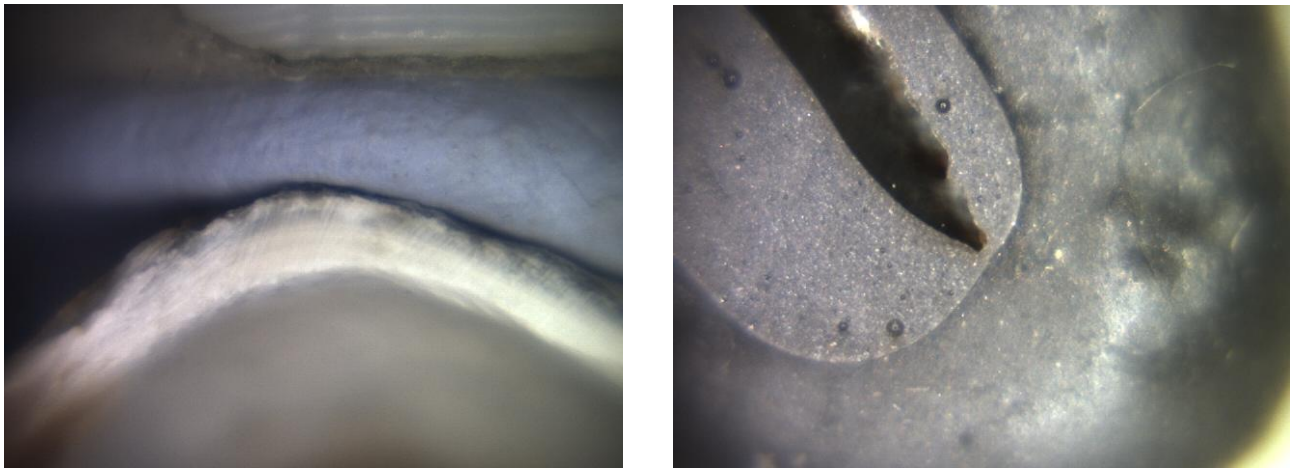


Ilustración 18 Concha de un lapa de perfil y concha de lapa en resina de epoxy

En las siguientes imágenes se puede observar el resultado que da el programa a las lapas anteriores. La primera imagen de la fila corresponde a la imagen original con la que se trabaja, la siguiente es el resultado con lo que se considera la capa de interés (la zona blanca de la imagen) ,después de ser procesada por matlab. En la última imagen los puntos azules representan la trayectoria seguida por el láser y los rojos y verdes los límites de la capa considerados en cada iteración del láser.

Ilustración 19 Resultado para la lapa de perfil

Ilustración 20 Resultado lapa en resina

En ambos casos se ha utilizado una profundidad del 50% por ese motivo el láser recorre una trayectoria por el medio de la capa. En la lapa de perfil se ha usado una tolerancia de 0.5 porque los colores que limitan la capa son muy, en la segunda se ha usado un 0.2 porque los límites son más claros.

A continuación se muestra el resultado obtenido para el de juguete de plástico esta vez con un 75% y un 0.2 de tolerancia.

Ilustración 21 Resultado trozo de plástico

A la hora de probar el programa con imágenes de conchas de caracol los resultados son buenos dependiendo de la imagen. En este caso se prueba tanto en imágenes de conchas sueltas como en imágenes que están compuestas por varias imágenes unidas. Las dos primeras imágenes se realizan al 50% de profundidad, la primera imagen al tener colores de capa más definidos se utiliza tolerancia de 0.5 mientras que en la segunda un 0.2.

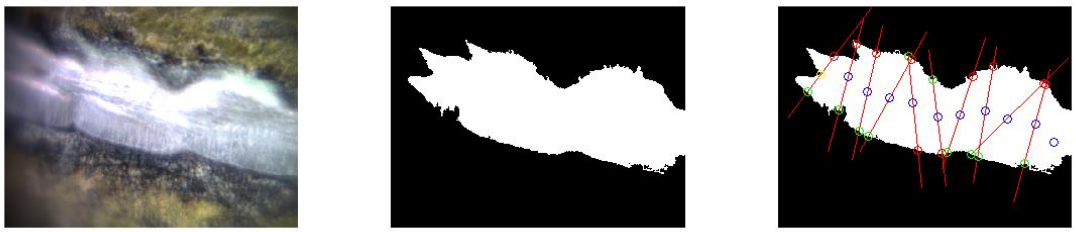


Ilustración 22 Resultado caracol 1

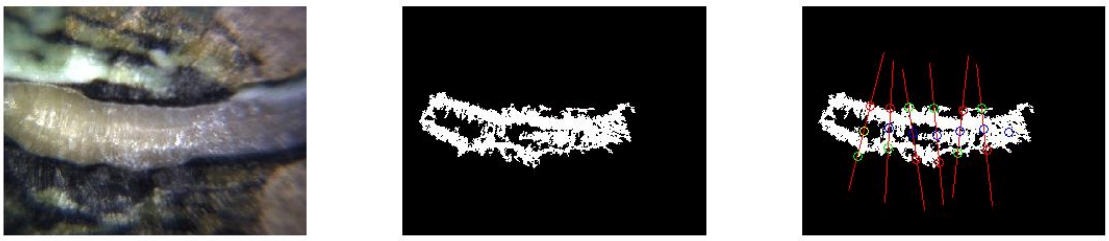


Ilustración 23 Resultado caracol 2

En la segunda imagen del caracol el programa no puede distinguir la capa desde el principio ya que empieza la capa con un color muy similar al resto de la concha, se prueba con diferentes valores de tolerancia para ver si se obtiene un buen resultado pero no se consigue ningún valor óptimo. Si se comienza el guiado desde el principio el resultado sería el siguiente:

Ilustración 24 Resultado caracol 2 desde principio de la imagen

En estas imágenes se observa que no se ha conseguido definir la capa que se buscaba sin embargo alguna vez si se consigue pero no es lo habitual.

Ilustración 25 Resultado caracol 2 desde el principio de la imagen

Las dos siguientes imágenes se tratan de varias imágenes de dos caracoles pudiendo así probar el programa en imágenes más largas. En la primera imagen se utiliza una tolerancia del 0.5 porque los bordes están bien marcados, el resultado que se obtiene es igual de bueno que cuando solo había una imagen.

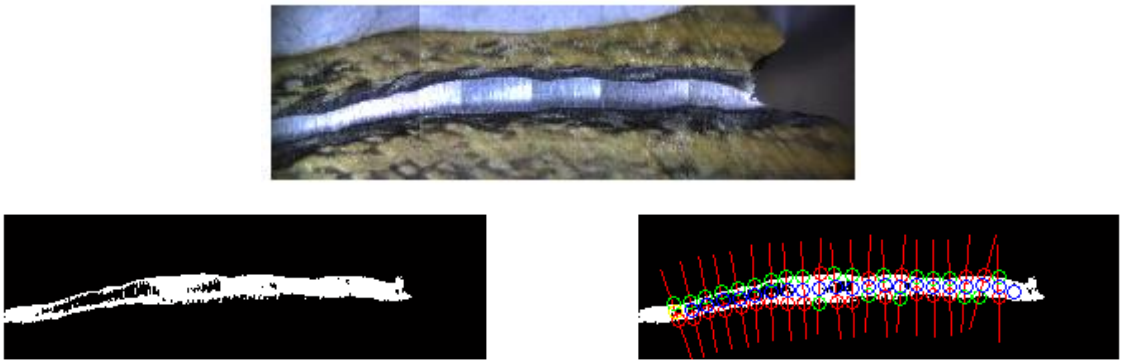


Ilustración 26 Resultado varias imágenes caracol 1

En la siguiente imagen el programa tampoco funciona como debería porque al igual que en la otra imagen de caracol no hay mucha diferencia de intensidad de color entre la capa y el resto. La única parte de la imagen de la que se puede sacar la capa es de la intermedia porque se puede observar que ahí está más definida.

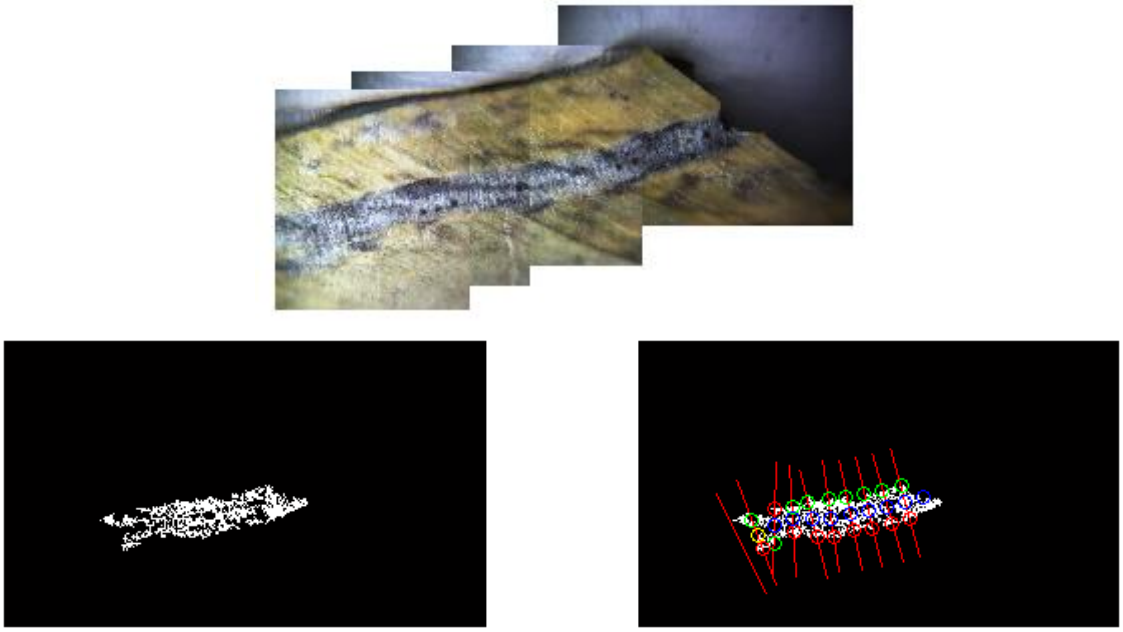


Ilustración 27 Resultado varias imágenes caracol 2

Sin embargo si se intenta diferenciar la capa desde el principio de la imagen los resultados son los siguientes.

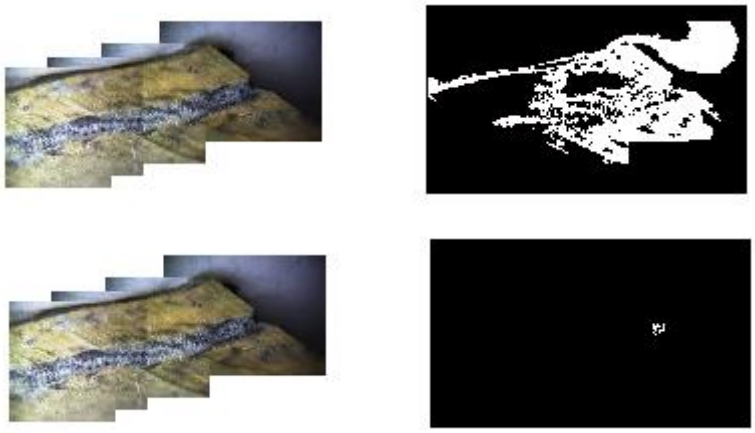


Ilustración 28 Resultado varias imágenes caracol 2 desde principio imagen

Se puede observar que en la primera imagen se coge más de lo necesario y en la segunda no se tiene casi nada.

3.2 Resultados de eliminación del background

Para probar el funcionamiento del algoritmo de eliminación se usan diferentes espectros adquiridos de diferentes conchas mediante LIBS. Se disponen de espectros complejos (con varios picos) y sencillos. En los siguientes ejemplos en las primeras imágenes de cada espectro se puede observar el espectro original (azul) y la aproximación del ruido (verde). En las segundas se tiene el espectro con el ruido eliminado (azul) y el ruido (verde). Se comienza con el caso de espectros sencillos y el parámetro de estimación del background utilizado es $K_\delta = 0.8$.

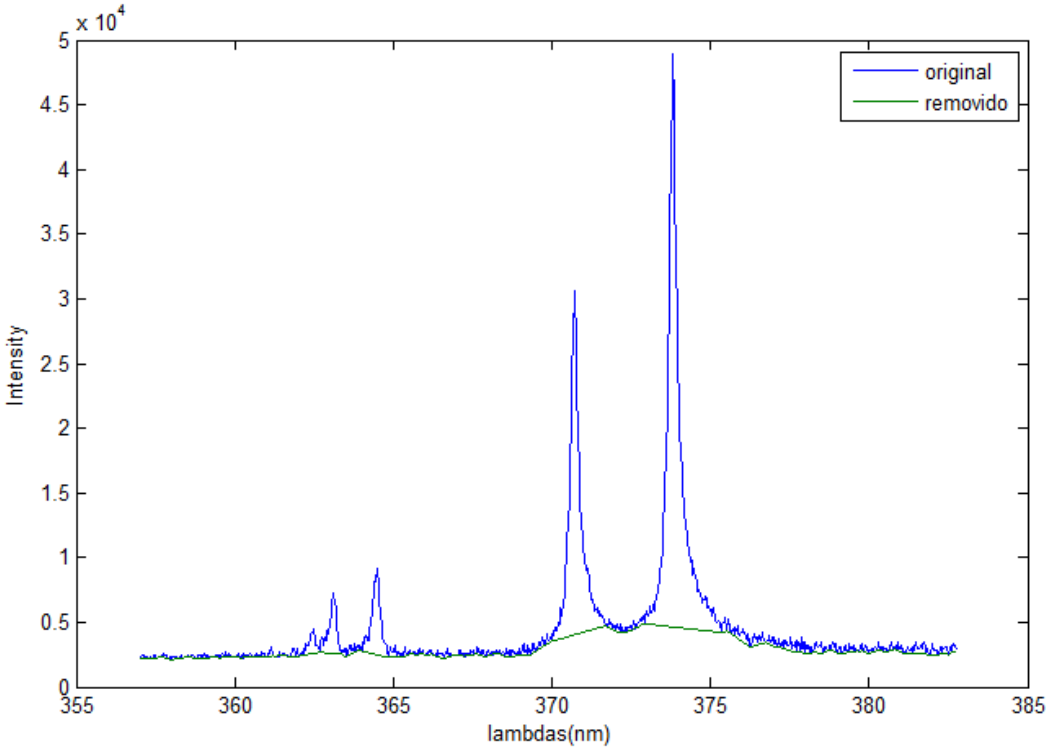


Ilustración 29 Espectro1 original

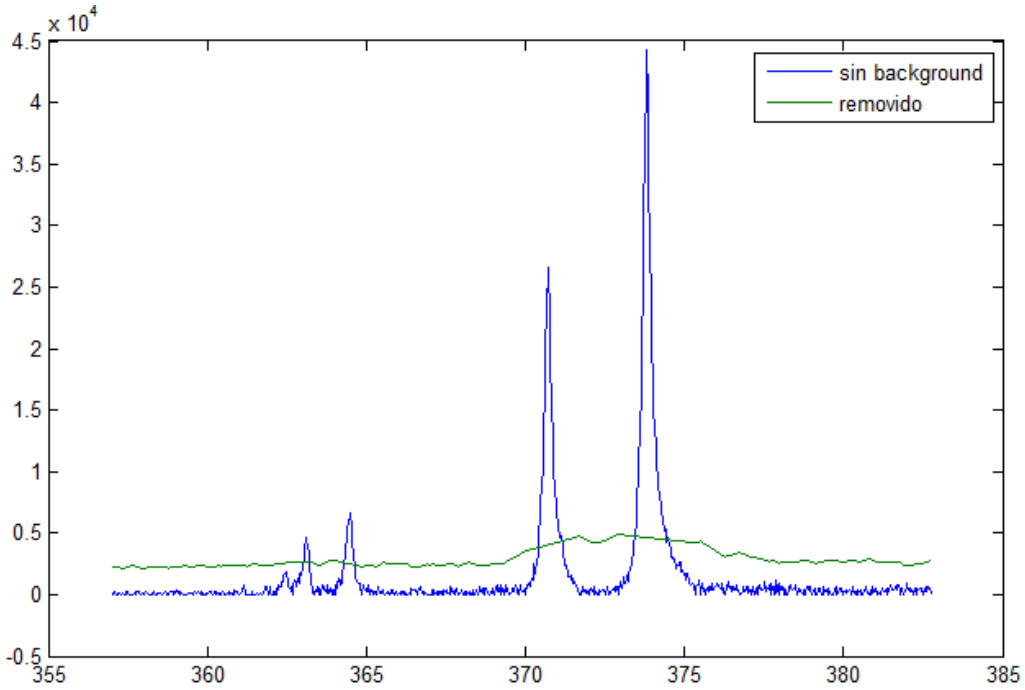


Ilustración 30 Espectro1 final

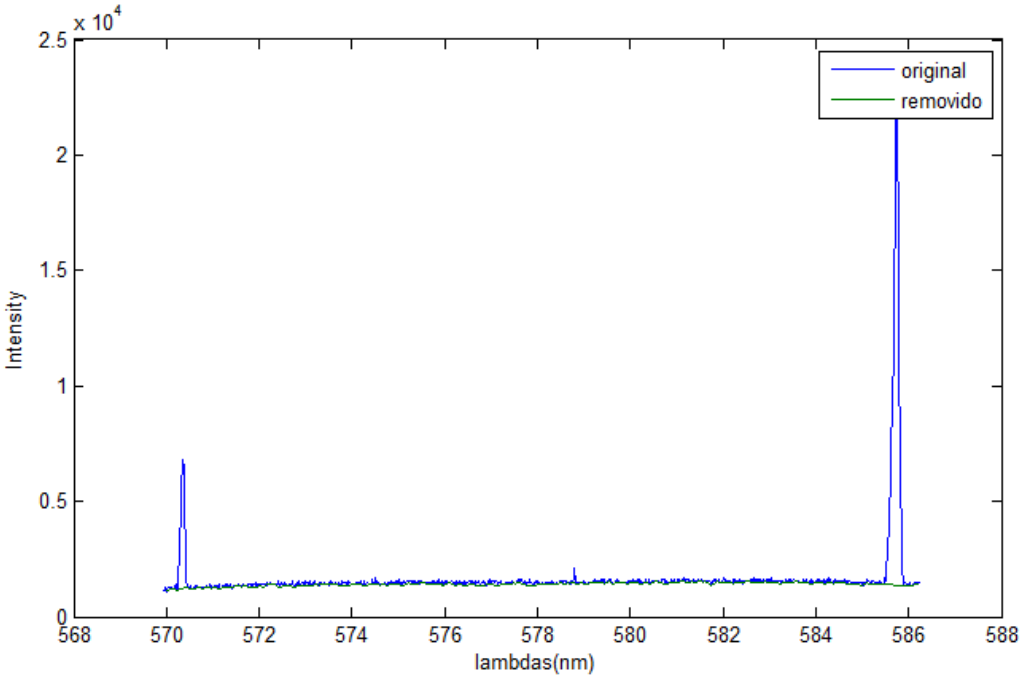


Ilustración 31Espectro2 original

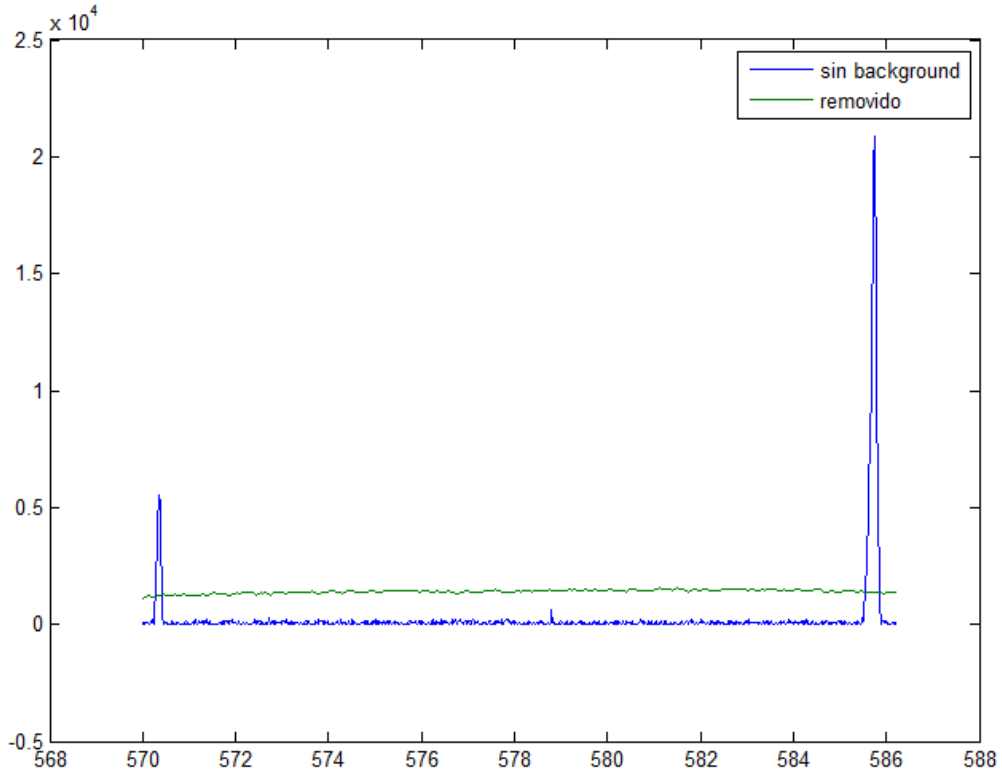


Ilustración 32 Espectro2 final

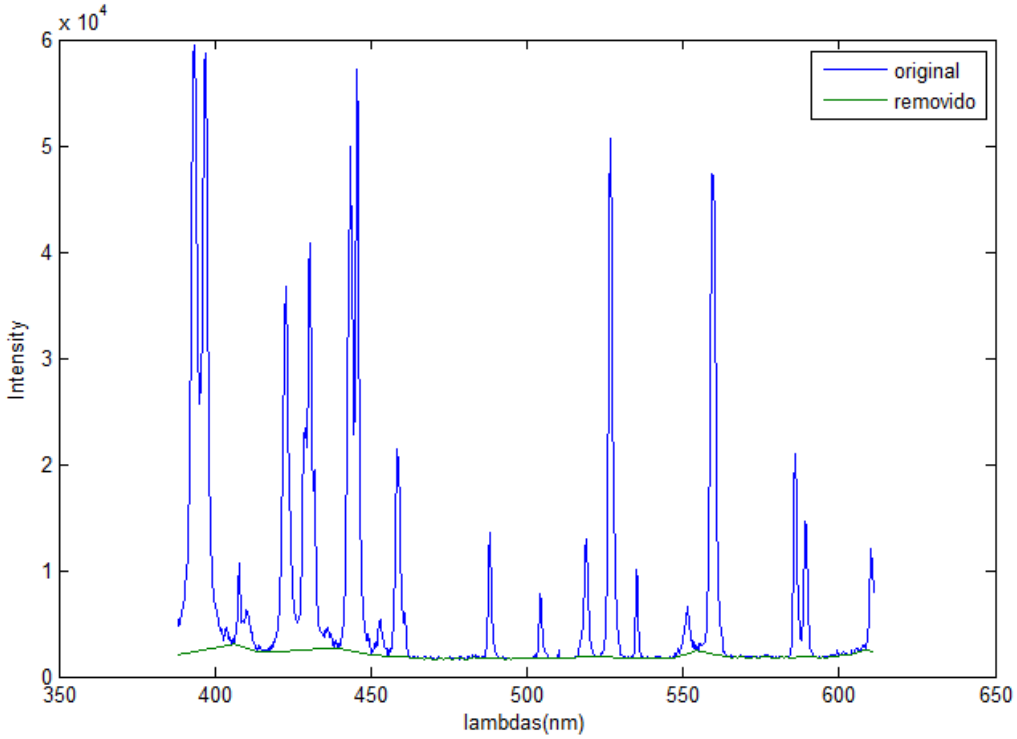


Ilustración 33 Espectro3 original

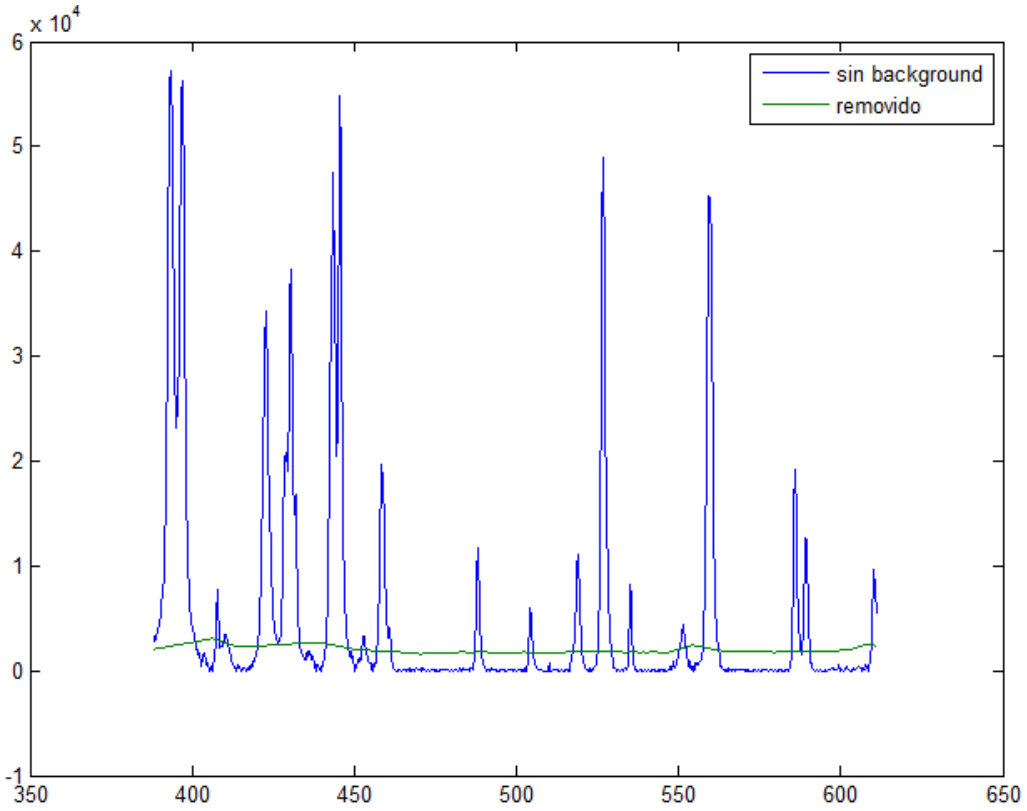


Ilustración 34 Espectro3 final

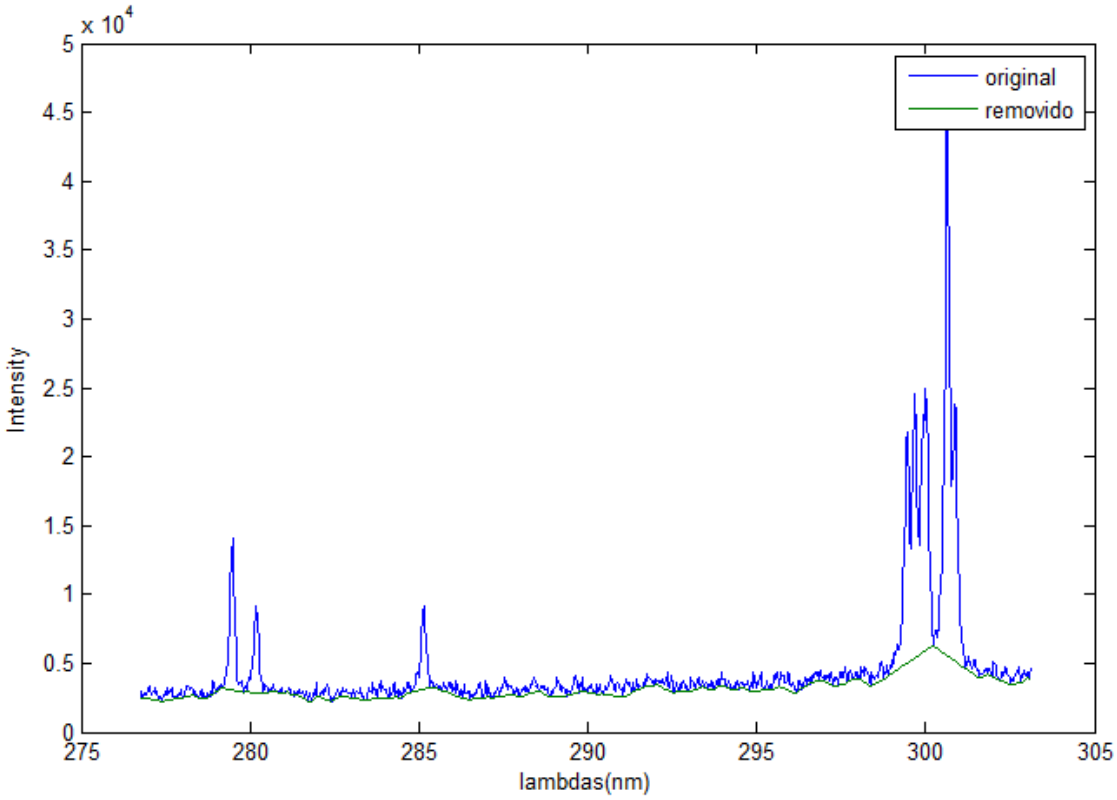


Ilustración 35 Espectro4 original

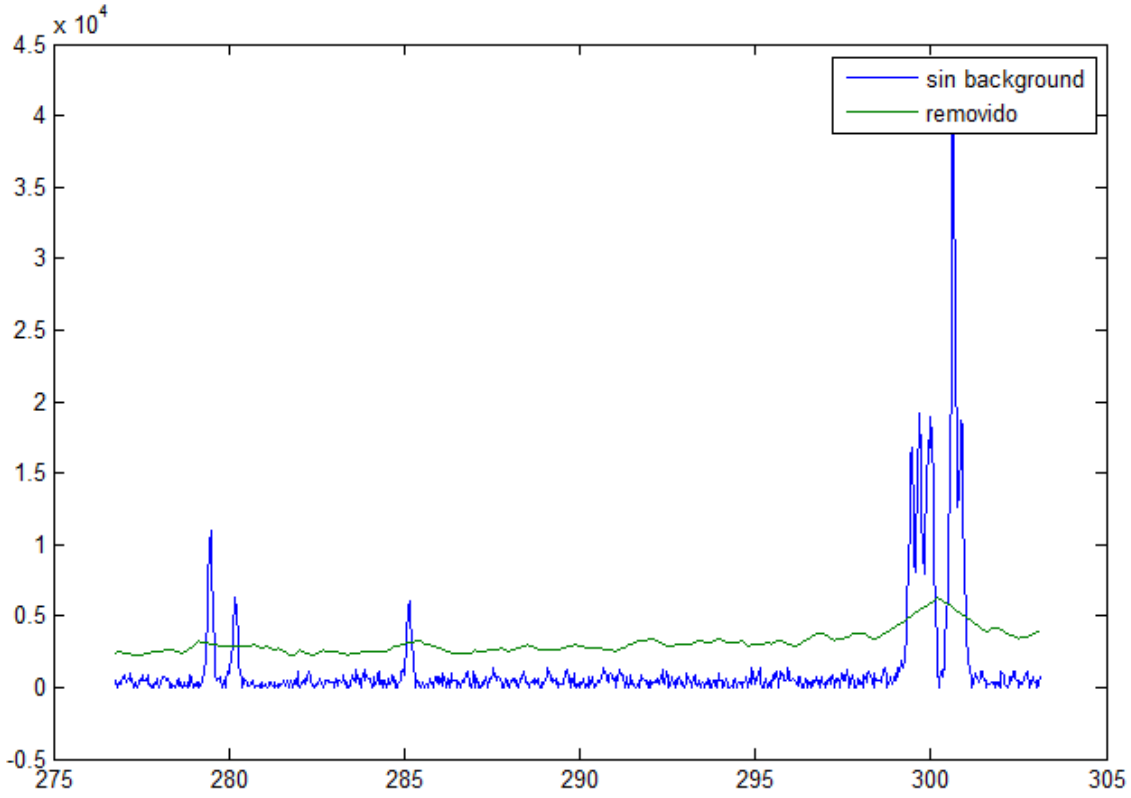


Ilustración 36 Espectro4 final

Por último en los espectros más complejos se ha utilizado un parámetro de estimación del background de $K_\delta = 1.4$.

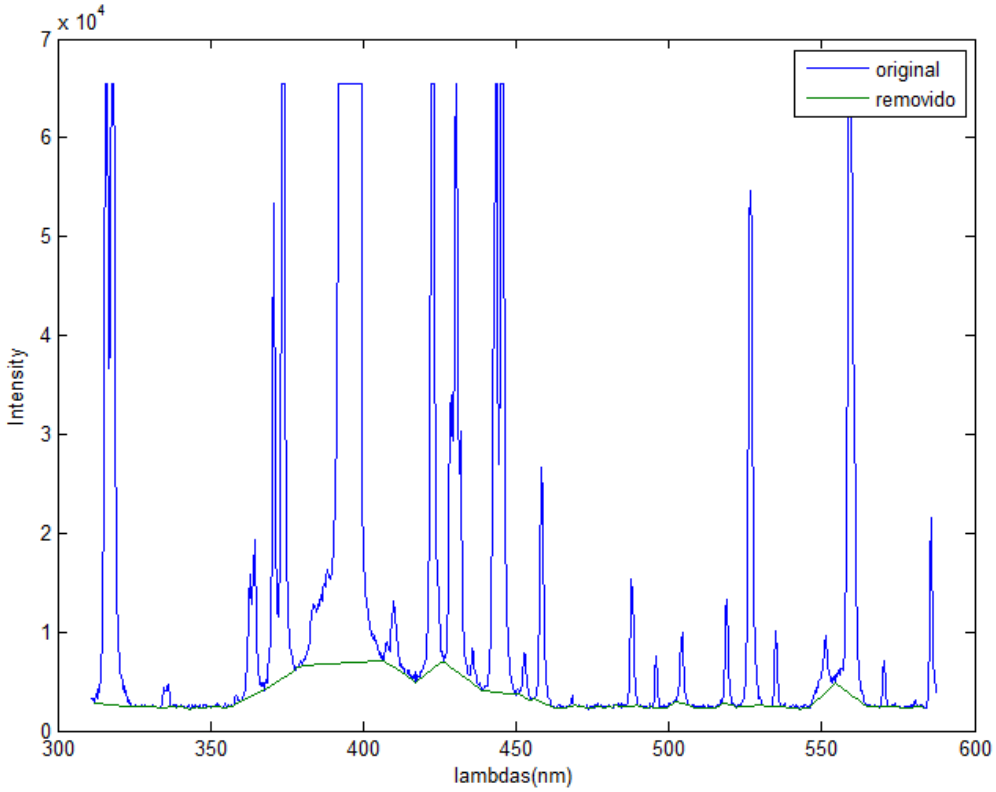


Ilustración 37 Espectro5 original

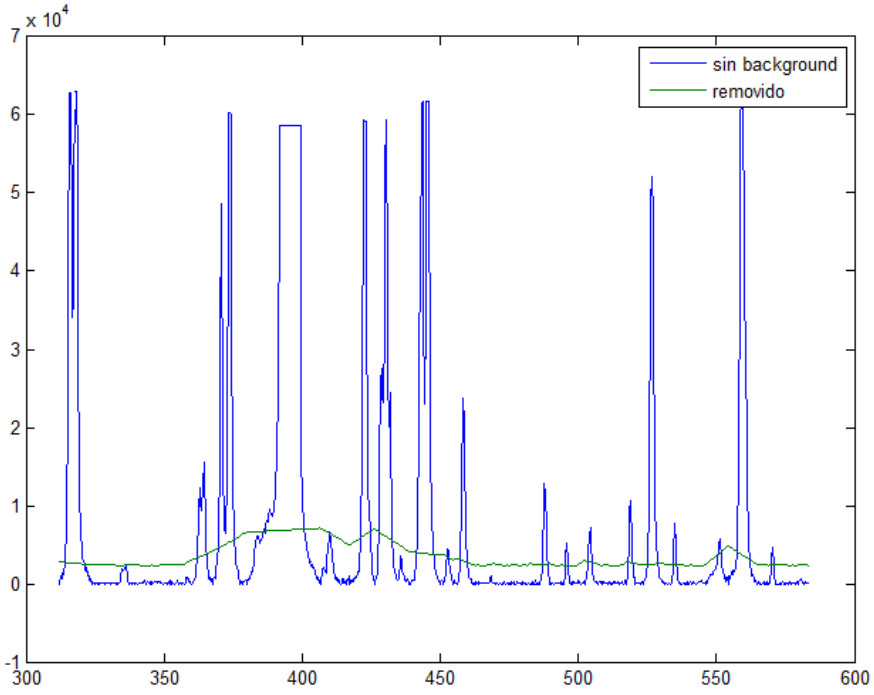


Ilustración 38 Espectro5 final

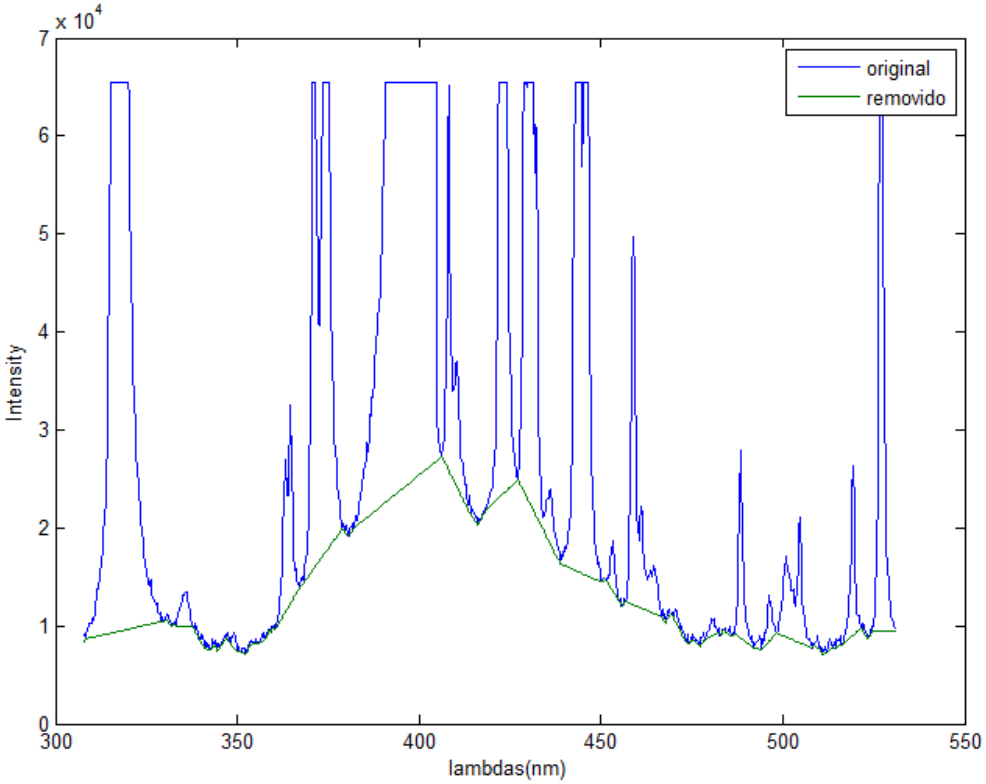


Ilustración 39 Espectro6 original

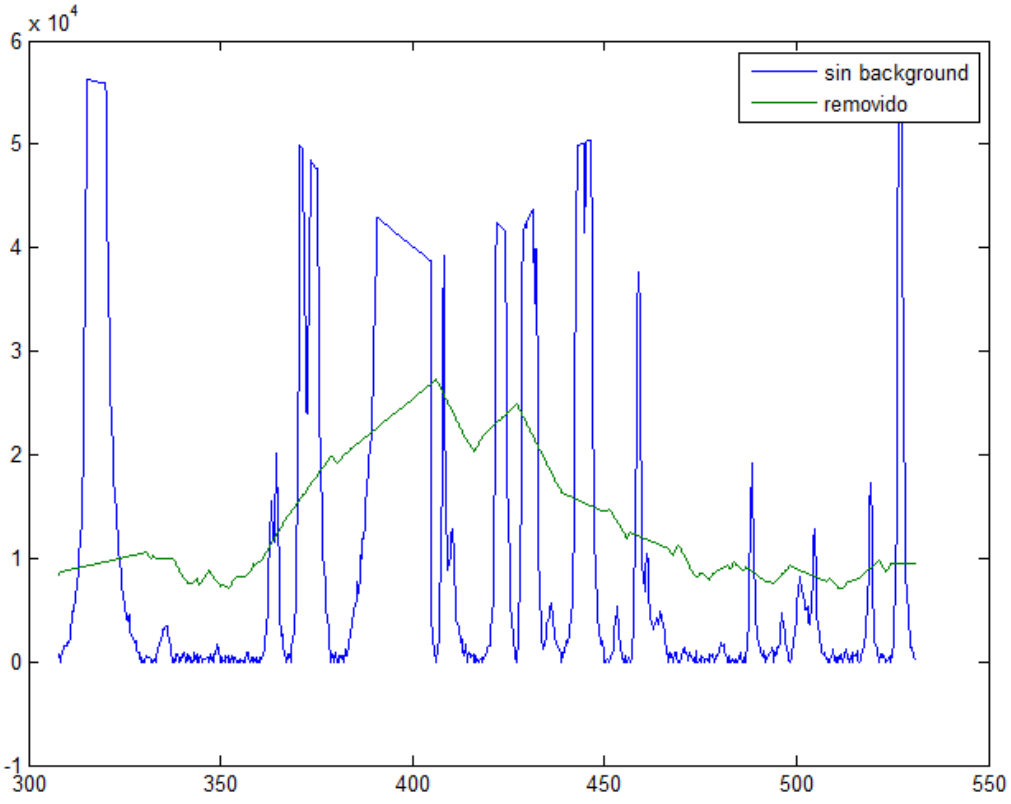


Ilustración 40 Espectro6 final

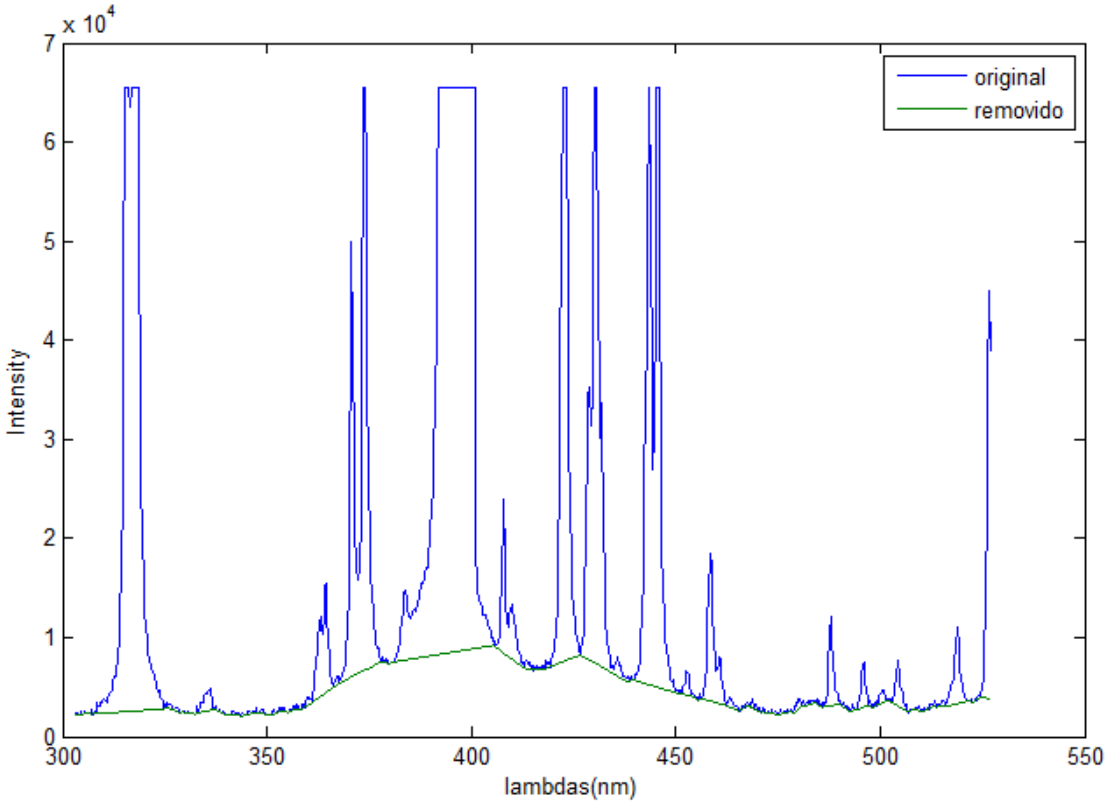


Ilustración 41 Espectro7 original

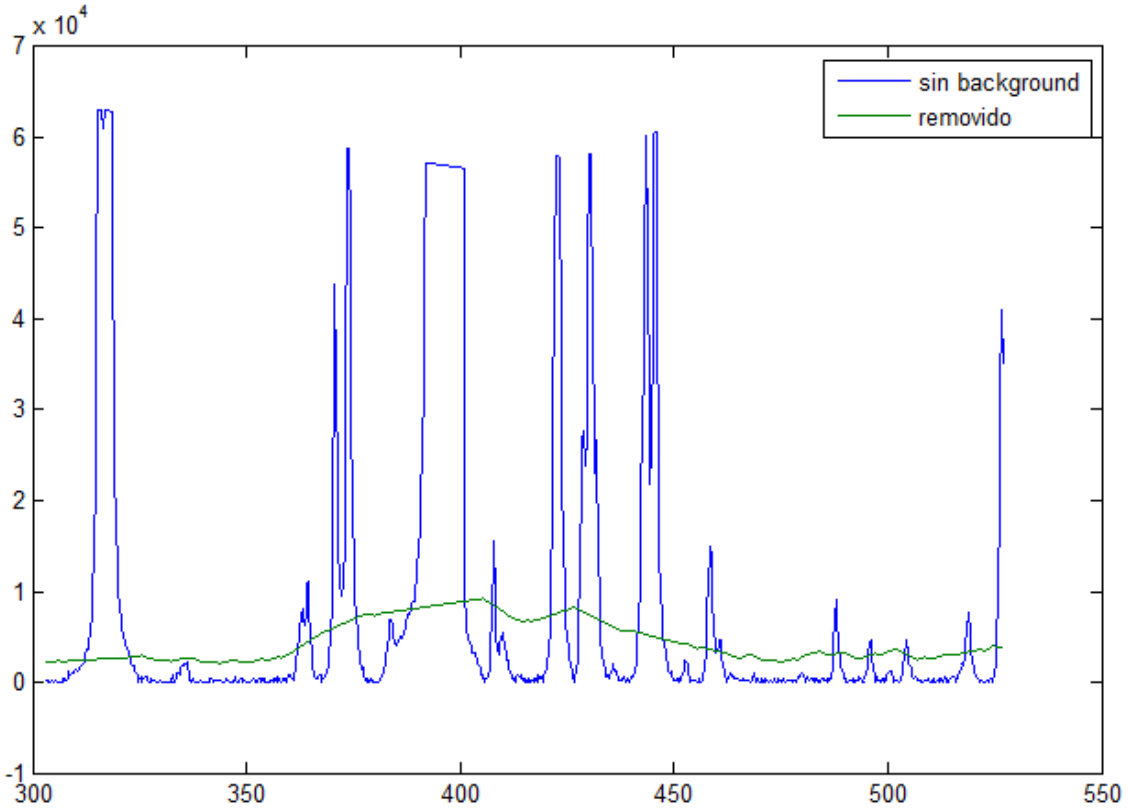


Ilustración 42 Espectro7 final

4. Conclusiones y líneas futuras

La técnica LIBS es muy útil para determinar la composición atómica de diferentes materiales. Para poder aprovechar todo el potencial que ofrece ahí que conseguir eliminar de la manera más precisa posible el ruido de fondo. Además realizar programas que realicen de manera automática el recorrido e iteraciones, así como programas que muestren de manera intuitiva los datos más característicos, agiliza mucho el trabajo.

A la hora de llevar a cabo el guiado del láser el resultado ha sido bastante bueno con excepción de los casos en los que los colores de la capa no difieren mucho del resto de la imagen, donde se encuentran problemas para detectar la capa.

La utilización del algoritmo de Sun y Yu para la eliminación del ruido de fondo ha dado muy buenos resultados, pudiendo así identificar los picos de los componentes de manera más precisa.

La creación del interfaz supone un ahorro de tiempo considerable a la hora de estudiar los resultados obtenidos, ya que permite interactuar de manera más intuitiva y rápida que si se realizara directamente.

En cuanto a las líneas de trabajo futuro, en primer lugar sería probar con nuevos algoritmos para que el guiado automático del láser funcione correctamente en todo tipo de imágenes incluidas las que tienen menos contraste. Respecto a la eliminación del background un punto interesante a mejorar sería conseguir de manera automática el parámetro de estimación y asi no tener que cambiarle dependiendo del espectro.

Otro trabajo interesante es aplicar el algoritmo de eliminación de ruido a espectros de Raman para probar si el funcionamiento sería óptimo al igual que en LIBS o por el contrario no sirve para Raman.

Con respecto al guiado del láser, sería interesante realizar un solape de las distintas imágenes tomadas en los distintos puntos, para poder así tener una imagen de la totalidad de la trayectoria que a seguido el láser.

Por último sería de mucha utilidad conseguir disponer de una imagen en la que se pueda observar la composición química. Sabiendo a que punto corresponde cada espectro utilizar los valores de Mg y Ca para saber si ahí una mayor o menor concentración, obteniendo una imagen global de la composición de la muestra.

5. Bibliografía

[1] Rafael C. González, Richard E. Woods, Steven L. Eddins “Digital image procesing using Matlab”.

[2] Oge Marques “Practical image and video processing using Matlab”.

[3] Image processing toolbox Matlab MathWorks

[4] Lesther Moreira Osorio, Luís V. Ponce Cabrera, Eduardo de Posada “Descubriendo la materia mediante luz láser: La espectroscopia de plasma inducido por láser (LIBS) como método para determinar la composición elemental de la materia”.

[5] Lanxiang Sun, Haibin Yu “Automatic estimation of varying continuum background emission in laser-induced breakdown spectroscopy”.

[6] D. A. Cremers and L. J. Raziemski “Laser Plasmas for Chemical Analysis”in Laser Spectroscopy and ist Applications.

Anexo I: Código de Matlab del guiado láser

A continuación se encuentra la función de matlab utilizada, las variables de entrada son la distancia entre los puntos (DELTA), la distancia entre los bordes de la capa (PROFUNDIDAD), la tolerancia y el nombre y extensión de la imagen de interés.

```
function guide (DELTA,PROFUNDIDAD,tolerancia,imagen)


%Leer imagenes para aportar info a la funcion
f=imread(imagen);

%% Datos dados por el usuario:
% DELTA=50;
% PROFUNDIDAD=50;
% tolerancia=0.5; % 0.5 ->prueba, caracol y caracol1, ->0.2 caracol2 y
juguete
%% Obtener info de la imagen

f=im2double(f); %transformacion de la imagen a double
imshow(f)
impixelinfo
[r,n,p]=size(f)
H = size(f, 1); % image height
W = size(f, 2); % image width
y_sig=1; x_sig=1;

[x_sig,y_sig]=ginput(2) %ginput(n) espera que le des n puntos, en este
caso 2 para trazar la linea
hold on


%line(x_sig,y_sig)
plot(x_sig(1),y_sig(1), 'o'); hold on
m = (diff(y_sig)/diff(x_sig));  alfa=atan(m); %calcular pendiente de
la recta trazada por el usuario
minv = -1/m; beta=atan(minv);



c_r = double(f(:, :, 1)); % Rojo
c_g = double(f(:, :, 2)); % verde
c_b = double(f(:, :, 3)); % Azul


    x1=x_sig(1)-DELTA*cos(alfa);
    y1=y_sig(1)-DELTA*sin(alfa);
    line([x1 x1+50],[y1 y1+50*minv],'Color','red')
    line([x1 x1-50],[y1 y1-50*minv],'Color','red')%axis equal
    %improfile automatico:
```

```
    xi=[ x1+150*cos(beta) x1-150*cos(beta)];
    yi=[ y1+150*sin(beta) y1-150*sin(beta)];
    plot(xi,yi,'r')


    [CX,CY,C,xi,yi] =improfile(f,xi,yi) %C-> Intensidades de la linea,
CX y CY coordenadas// recta perpendicular
    %[CX,CY,C,xi,yi] =improfile(f,x_sig,y_sig) %recta usuario

%% Bucle para obtener puntos

    %% Diferencia de colores
    xlist=double(CX(round(length(CX)/2)));
ylist=double(CY(round(length(CY)/2))); %Pixeles de referencia

    xlist=round(xlist); %redondea
    ylist=round(ylist);
    % Encontrar todos los pixeles que cumplan con la tolerancia
    color_mask = false(H, W); %array HxW de 0´s logicos

        idx=1;


        ref_r = double(f(ylist(idx), xlist(idx), 1));
        ref_g = double(f(ylist(idx), xlist(idx), 2));
        ref_b = double(f(ylist(idx), xlist(idx), 3));
        color_mask = color_mask | ... % A|B or de A y B elemento a
elemente y devuelve un vector de verdadero(1) y falso(0)
            ((c_r - ref_r) .^ 2 + (c_g - ref_g) .^ 2 + ...
            (c_b - ref_b) .^ 2) <= tolerancia ^ 2;

    [objects, count] = bwlabel(color_mask, 8); %Objetos presentes en
la imagen con 8 vecinos

    % Inicializacion mascara de salida
    bin_mask = false(H, W);

    % Indice del pixel de referencia
    pos_idxs = (xlist - 1) * H + ylist;

    for idx = 1:count
        object = (objects == idx); % posiciones de cada objeto

        % Añadir a la salida si el objeto tiene el pixel de referencia
        if any(object(pos_idxs)) %comprueba si hay algun nuemro
distinto de 0
            bin_mask = bin_mask | object;
        end
    end
    figure(3)
    subplot(1, 3, 1); imshow(f);
    subplot(1, 3, 2); imshow(bin_mask); hold on

     subplot(1, 3, 3); imshow(bin_mask); hold on


while  x_sig(end)>=150
```

```
    x1=x_sig(1)-DELTA*cos(alfa);
    y1=y_sig(1)-DELTA*sin(alfa);
%     line([x1 x1+50],[y1 y1+50*minv],'Color','red')
%     line([x1 x1-50],[y1 y1-50*minv],'Color','red')%axis equal
    %improfile automatico:


    xi=[ x1+150*cos(beta) x1-150*cos(beta)];
    yi=[ y1+150*sin(beta) y1-150*sin(beta)];
    plot(xi,yi,'r')



    %Busco limites de la capa
    [CX1,CY1,C1,xi1,yi1] =improfile(bin_mask,xi,yi)
    C1(isnan(C1))=0;
    k=find(C1) %k tengo todos los indices de los elementos de C1
distintos de 0
    a=[CX1(max(k)),CY1(max(k))]; b=[CX1(min(k)), CY1(min(k))];
    subplot(1, 3, 3);
    plot(a(1), a(2),'og')
    plot(b(1),b(2),'or')

    %Busco el siguiente punto
    d=abs([b(1) b(2)]-[a(1) a(2)])
    d=sqrt(d(1)^2+d(2)^2)

    figure(3)
    subplot(1, 3, 3);
    %line(x_sig,y_sig)
    plot(x_sig(1),y_sig(1), 'o'); hold on

    d_n=(PROFUNDIDAD/100)*d;
    alpha=atan(minv);

    x_sig1=a(1)+d_n*cos(alpha);
    y_sig1=a(2)+d_n*sin(alpha);

    if y_sig1(end)>=r && x_sig1(end)>=n
        break;
    end
    subplot(1, 3, 3);
    plot(x_sig1,y_sig1,'yo')

    m = (y_sig(1)-y_sig1)/(x_sig(1)-x_sig1);  alfa=atan(m); %calcular
pendiente de la recta trazada por el usuario
minv = -1/m; beta=atan(minv);

    y_sig= y_sig1; x_sig=x_sig1;

    pause(1)
end
end
```

Anexo II: Código de Matlab del algoritmo de eliminación de background

```
load 'espectro1.mat'; % he grabado varios espectros espectro1,
espectro2 ... para probar, en la variable sp

figure(1)
plot(lambdas,sp); xlabel('lambdas(nm)'); ylabel('Intensity');
%Cálculo de todos los minimos
i=2;
[mini,lamda]=min(sp);
Imin(1)=sp(lamda);
lamda(i)=lambdas(lamda);
for k=2:length(sp)-1
if sp(k)<sp(k-1) && sp(k)<sp(k+1)

lamda(i)=lambdas(k);
Imin(i)=sp(k);
i=i+1;
end
end

for j=1:length(lamda)-1
r_j(j)=abs((Imin(j+1)-Imin(j))/(lamda(j+1)-lamda(j))); %varience ratio
entre dos minimos próximos
end

%Cálculo del umbral
k=1.4;
RSD=std(r_j)/mean(r_j); %relative standard desviation
kt=k/RSD; %RSD-> desviacion de r_j y k-> background stimation factor
tetha=kt*mean(r_j);% threshold

I_L(1)=Imin(1); lamda_l(1)=lamda(1); j=1; l=1;

for j=1:length(lamda)-1
r_j(j)=(Imin(j+1)-Imin(j))/(lamda(j+1)-lamda(j));
if r_j(j)>tetha
    Imin(j+1)=Imin(j); lamda(j+1)=lamda(j);
else
    l=l+1;
    I_L(l)=Imin(j+1); lamda_l(l)=lamda(j+1);
end
j=j+1;
end

I_R(1)=I_L(l); lamda_r(1)=lamda_l(l); p=1;

for q=l:-1:2
r_j_l(q)=(I_L(q-1)-I_L(q))/(lamda_l(q-1)-lamda_l(q)); %varience ratio
entre dos minimos próximos

if r_j_l(q)<-tetha
    I_L(q-1)=I_L(q); lamda_l(q-1)=lamda_l(q);
else
```

```
    p=p+1;
    I_R(p)=I_L(q-1); lamda_r(p)=lamda_l(q-1);

end
q=q-1;
end

x=lamda_r'; % lambdas que han quedado tras procesar por la derecha
y=I_R';  % las intensidades de los mínimos
% lambdas,sp es el espectro original

x = x(1:size(x)-1); % quito el último elemento
y = y(1:size(y)-1); % quito el último elemento

%1° forma con el interp1
rem=interp1(x,y,lambdas)
%2° forma con spapi
mi_spline =spapi(2,x,y,lambdas);   %spline de segundo orden
rem = fnval(mi_spline,lambdas);

figure(2); plot(lambdas,sp,lambdas,rem);xlabel('lambdas(nm)');
ylabel('Intensity');
legend('original','removido');
removido=sp-rem;

figure(3); plot(lambdas,removido,lambdas,rem);
legend('sin background','removido');
```

Anexo III: Código de Matlab de la GUI

Código de matlab de la función *procesa*, utilizada por el interfaz para realizar cálculos recurrentes.

```
function
[lambdas,meanspec,eje,ratio_MgCa_del_promedio_de_ratios,ratio_MgCa_del
_espectro_promedio]=procesa(PN,Ncleaningshots, Npulses, LAMBDA_CALCIO,
LAMBDA_MAGNESIO)
load([ PN, '\', 'matlabData.mat'] , '-regexp', '^(?!PN$)\w');

 pixelCa =
buscaPixel(lambdas,LAMBDA_CALCIO,mean(mean(spectra,2),3),1); %
devuelve el píxel más cercano a ese valor, luego comprueba que sea un
máximo
pixelMg =
buscaPixel(lambdas,LAMBDA_MAGNESIO,mean(mean(spectra,2),3),1);
mininum_peak_value = 2000;
 maximum_peak_value = 65000;
 meanspec=zeros(Npixeles,Nparam1,Nparam2); % mean spectra of each
spatial spot or parameter point
 tmp_spectra = zeros(Npixeles,1);
 num_valid_spectra = zeros(Nparam1,Nparam2);
 lampara_FWHM = 0.8;
 ancho = lambdas(1024)-lambdas(1);
nm2pixels=1024/ancho;
 pixex=ceil(lampara_FWHM*nm2pixels/2);
 for idx2= 1:Nparam2
  for idx1= 1:Nparam1
      howmanyspec=0;
      for npulse=Ncleaningshots:Npulses
        peak_max = max(spectra(:,npulse,idx1,idx2));
        peak_mean = mean(spectra(:,npulse,idx1,idx2));
        meanspec(:,idx1,idx2) = 0;
        if (spectra_is_valid(npulse,idx1,idx2)) % están ya marcados en
la medida
          tmp_spectra = spectra(:,npulse,idx1,idx2);
          intensidad_todos(npulse,idx1,idx2) = sum(tmp_spectra);
          meanspec(:,idx1,idx2) = meanspec(:,idx1,idx2) + tmp_spectra;
          howmanyspec = howmanyspec +1 ;
          %ratio_MgCa_todos(npulse,idx1,idx2) = NaN; %
        end
      end
      if (howmanyspec==0)
      else
          meanspec(:,idx1,idx2) = meanspec(:,idx1,idx2) / howmanyspec;
% mean value
      end

    % ahora hay que calcular otra vez el espectro promedio pero
quitando
    % los NUEVOS NO VÁLIDOS
      howmanyspec=0;
      meanspec(:,idx1,idx2) = 0;
      for npulse=1:Npulses
```

```
        if (spectra_is_valid(npulse,idx1,idx2)) % solo los válidos
actuales
          tmp_spectra = spectra(:,npulse,idx1,idx2);
          intensidad_todos(npulse,idx1,idx2) = sum(tmp_spectra);
          meanspec(:,idx1,idx2) = meanspec(:,idx1,idx2) + tmp_spectra;
          ratio_MgCa_todos(npulse,idx1,idx2) =
integraPico(tmp_spectra,pixelMg,pixex,0)/integraPico(tmp_spectra,pixel
Ca,pixex,0);

          howmanyspec = howmanyspec +1 ;
          else % no es un espectro válido
            ratio_MgCa_todos(npulse,idx1,idx2) = NaN; % con esto el
cálculo de std no lo tiene en cuenta

        end
      end
      if (howmanyspec==0)
          disp(['Warning: spectrum #' num2str(idx1) ' without VALID
peaks.']);
      else
          meanspec(:,idx1,idx2) = meanspec(:,idx1,idx2) / howmanyspec;
% mean value
          disp(['Point (',num2str(idx1),',',num2str(idx2),') with
',num2str(howmanyspec),' valid spectra.']);
      end
      num_valid_spectra(idx1,idx2) = howmanyspec;
      tmp_spectra = meanspec(:,idx1,idx2);

     ratio_MgCa_del_espectro_promedio(idx1,idx2) =
integraPico(tmp_spectra,pixelMg,pixex,0)/integraPico(tmp_spectra,pixel
Ca,pixex,0);
     ratio_MgCa_del_promedio_de_ratios(idx1,idx2) =
nanmean(ratio_MgCa_todos(:,idx1,idx2),1);
   end % param1
end % param2
lambdas=lambdas
eje = param1From:param1Step:param1To;

end
```

Código del interfaz:

```
function varargout = interfaz(varargin)
% INTERFAZ M-file for interfaz.fig
%      INTERFAZ, by itself, creates a new INTERFAZ or raises the
existing
%      singleton*.
%
%      H = INTERFAZ returns the handle to a new INTERFAZ or the handle
to
%      the existing singleton*.
%
%      INTERFAZ('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in INTERFAZ.M with the given input
arguments.
%
%      INTERFAZ('Property','Value',...) creates a new INTERFAZ or
raises the
```

```
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before interfaz_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to interfaz_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help interfaz

% Last Modified by GUIDE v2.5 09-Mar-2015 16:49:38

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @interfaz_OpeningFcn, ...
                   'gui_OutputFcn',  @interfaz_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before interfaz is made visible.
function interfaz_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to interfaz (see VARARGIN)

PN='C:\Datos\Dropbox\GifLibs\Arqueo\LAN3\19112014-175923_interior'


caracteristico = '_layers'

%PN = strcat(path,experimentPath,datePath);

handles.edit4=70;
handles.edit3=100;
Ncleaningshots =handles.edit4;
 Npulses =handles.edit3;
%ventana 150lpp 416nm original
LAMBDA_CALCIO = 300.67 % calcium line that is monitorized for RSD
```

```
LAMBDA_MAGNESIO = 285.21; % magnesium line that is monitorized for RSD
[lambdas,meanspec,eje,ratio_MgCa_del_promedio_de_ratios,ratio_MgCa_del
_espectro_promedio]=procesa(PN,Ncleaningshots, Npulses, LAMBDA_CALCIO,
LAMBDA_MAGNESIO)

%     axes(handles.axes1)
%     plot(lambdas,meanspec(:,1,1))
%     zoom on
save ('datos')
x=[LAMBDA_CALCIO,LAMBDA_CALCIO];
y=[1,1.5e4];plot(x,y,'g')

z=[LAMBDA_MAGNESIO,LAMBDA_MAGNESIO];
q=[1,1.5e4];plot(x,y,'r')

axes(handles.axes1)
plot(lambdas,meanspec(:,1,1),x,y,z,q)
zoom on
axes(handles.axes2)
plot(eje,ratio_MgCa_del_promedio_de_ratios(:,1),'LineWidth',2,'Color',
'r');
grid on;
ratio=findobj(gcbf,'tag','edit5')
set(ratio,'String',ratio_MgCa_del_espectro_promedio (1,1))
       handles.edit5=ratio_MgCa_del_espectro_promedio (1,1);
handles.pushbutton3=0;
handles.pushbutton4=0;


% Choose default command line output for interfaz
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes interfaz wait for user response (see UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
function varargout = interfaz_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double
Punto=get(hObject,'String'); %almacenar el valor ingresado
```

```
Punto = str2double(Punto) %pasar de string a double
handles.edit1=Punto %lo almacenamos en el identificador
guidata(hObject, handles);

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

punto=handles.edit1
 if punto==0
    punto=1
    Punto=findobj(gcbf,'tag','edit1')
      set(Punto,'String',punto)
    disp('El 0 no es un valor disponible')
    end
 load('datos.mat');

axes(handles.axes1)
x=[LAMBDA_CALCIO,LAMBDA_CALCIO];
y=[1,1.5e4];plot(x,y,'g')

z=[LAMBDA_MAGNESIO,LAMBDA_MAGNESIO];
q=[1,1.5e4];plot(x,y,'r')

handles.edit1=punto;
plot(lambdas,meanspec(:,punto,1),x,y,z,q)
zoom on
     ratio=findobj(gcbf,'tag','edit5')
set(ratio,'String',ratio_MgCa_del_espectro_promedio (punto,1))
       handles.edit5=ratio_MgCa_del_espectro_promedio (punto,1);

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in togglebutton1.
function togglebutton1_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton1


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if (handles.pushbutton3==0 && handles.pushbutton4==0 )
punto=1
else
punto=handles.edit1
end
      load('datos.mat');
      axes(handles.axes1)
      punto=punto-1
      if punto==0
    punto=1
    disp('El 0 no es un valor disponible')
      end
x=[LAMBDA_CALCIO,LAMBDA_CALCIO];
y=[1,1.5e4];plot(x,y,'g')

z=[LAMBDA_MAGNESIO,LAMBDA_MAGNESIO];
q=[1,1.5e4];plot(x,y,'r')


plot(lambdas,meanspec(:,punto,1),x,y,z,q)
      Punto=findobj(gcbf,'tag','edit1')
      set(Punto,'String',punto)
      zoom on


%       set(handles.pushbutton3,'enable','off');
%       set(handles.pushbutton3,'enable','on');
      handles.edit1=punto
      handles.pushbutton3=1
           ratio=findobj(gcbf,'tag','edit5')
set(ratio,'String',ratio_MgCa_del_espectro_promedio (punto,1))
       handles.edit5=ratio_MgCa_del_espectro_promedio (punto,1);
guidata(hObject, handles);
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if (handles.pushbutton3==0 && handles.pushbutton4==0 )
punto=1
else
punto=handles.edit1
end
      load('datos.mat');
      axes(handles.axes1)
      punto=punto+1
  x=[LAMBDA_CALCIO,LAMBDA_CALCIO];
y=[1,1.5e4];plot(x,y,'g')

z=[LAMBDA_MAGNESIO,LAMBDA_MAGNESIO];
q=[1,1.5e4];plot(x,y,'r')


plot(lambdas,meanspec(:,punto,1),x,y,z,q)
```

```
axes(handles.axes2)

plot(eje,ratio_MgCa_del_promedio_de_ratios(:,1),'LineWidth',2,'Color',
'r');
grid on;
      Punto=findobj(gcbf,'tag','edit1')
      set(Punto,'String',punto)
       handles.edit1=punto;
      handles.pushbutton4=1
      zoom on
      ratio=findobj(gcbf,'tag','edit5')
set(ratio,'String',ratio_MgCa_del_espectro_promedio (punto,1))
       handles.edit5=ratio_MgCa_del_espectro_promedio (punto,1);
guidata(hObject, handles);


function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double
PN=get(hObject,'String'); %almacenar el valor ingresado


handles.edit2=PN %lo almacenamos en el identificador
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
PN=handles.edit2
caracteristico = '_layers'

%PN = strcat(path,experimentPath,datePath);

Ncleaningshots = 70;
Npulses=100;
 %ventana 150lpp 416nm original
LAMBDA_CALCIO = 300.67 % calcium line that is monitorized for RSD
```

```
LAMBDA_MAGNESIO = 285.21; % magnesium line that is monitorized for RSD
[lambdas,meanspec,eje,ratio_MgCa_del_promedio_de_ratios,ratio_MgCa_del
_espectro_promedio]=procesa(PN,Ncleaningshots, Npulses, LAMBDA_CALCIO,
LAMBDA_MAGNESIO)

save('datos');
Punto=findobj(gcbf,'tag','edit1')
set(Punto,'String',1)
       handles.edit1=1;

x=[LAMBDA_CALCIO,LAMBDA_CALCIO];
y=[1,1.5e4];plot(x,y,'g')

z=[LAMBDA_MAGNESIO,LAMBDA_MAGNESIO];
q=[1,1.5e4];plot(x,y,'r')

axes(handles.axes1)
plot(lambdas,meanspec(:,1,1),x,y,z,q)
zoom on
axes(handles.axes2)
plot(eje,ratio_MgCa_del_promedio_de_ratios(:,1),'LineWidth',2,'Color',
'r');
grid on;

ratio=findobj(gcbf,'tag','edit5')
set(ratio,'String',ratio_MgCa_del_espectro_promedio (1,1))
       handles.edit5=ratio_MgCa_del_espectro_promedio (1,1);
handles.pushbutton5=10;
guidata(hObject, handles);


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu1
PN=get(hObject,'String')
a=get(hObject,'Value')
PN1=PN(a)


caracteristico = '_layers'

%PN = strcat(path,experimentPath,datePath);
Ncleaningshots = 70;
Npulses=100;
LAMBDA_MAGNESIO = 285.21; % magnesium line that is monitorized for RSD
LAMBDA_CALCIO = 300.67 % calcium line that is monitorized for RSD

[lambdas,meanspec,eje,ratio_MgCa_del_promedio_de_ratios,ratio_MgCa_del
_espectro_promedio]=procesa(PN,Ncleaningshots, Npulses, LAMBDA_CALCIO,
LAMBDA_MAGNESIO)
```

```
save('datos');
       handles.edit1=1;
     x=[LAMBDA_CALCIO,LAMBDA_CALCIO];
y=[1,1.5e4];plot(x,y,'g')

z=[LAMBDA_MAGNESIO,LAMBDA_MAGNESIO];
q=[1,1.5e4];plot(x,y,'r')

axes(handles.axes1)
plot(lambdas,meanspec(:,1,1),x,y,z,q)
zoom on
axes(handles.axes2)
plot(eje,ratio_MgCa_del_promedio_de_ratios(:,1),'LineWidth',2,'Color',
'r');
grid on;

ratio=findobj(gcbf,'tag','edit5')
set(ratio,'String',ratio_MgCa_del_espectro_promedio (1,1))
       handles.edit5=ratio_MgCa_del_espectro_promedio (1,1);

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double
Npulses=get(hObject,'String'); %almacenar el valor ingresado
Npulses=str2double(Npulses);

handles.edit3=Npulses %lo almacenamos en el identificador
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4
as a double
Ncleaningshots=get(hObject,'String'); %almacenar el valor ingresado
Ncleaningshots=str2double(Ncleaningshots);

handles.edit4=Ncleaningshots %lo almacenamos en el identificador
guidata(hObject, handles);



% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)7

caracteristico = '_layers'
if handles.pushbutton5==10
    PN=handles.edit2;
else
    PN='C:\Datos\Dropbox\GifLibs\Arqueo\LAN3\19112014-175923_interior'
end

 Ncleaningshots=handles.edit4
Npulses=handles.edit3
LAMBDA_CALCIO = 300.67 % calcium line that is monitorized for RSD
LAMBDA_MAGNESIO = 285.21; % magnesium line that
[lambdas,meanspec,eje,ratio_MgCa_del_promedio_de_ratios,ratio_MgCa_del
_espectro_promedio]=procesa(PN,Ncleaningshots, Npulses, LAMBDA_CALCIO,
LAMBDA_MAGNESIO)


save('datos');
Punto=findobj(gcbf,'tag','edit1')
set(Punto,'String',1)
       handles.edit1=1;
       x=[LAMBDA_CALCIO,LAMBDA_CALCIO];
y=[1,1.5e4];plot(x,y,'g')

z=[LAMBDA_MAGNESIO,LAMBDA_MAGNESIO];
q=[1,1.5e4];plot(x,y,'r')
```

```
axes(handles.axes1)
plot(lambdas,meanspec(:,1,1),x,y,z,q)
zoom on
axes(handles.axes2)

plot(eje,ratio_MgCa_del_promedio_de_ratios(:,1),'LineWidth',2,'Color',
'r');
grid on;
ratio=findobj(gcbf,'tag','edit5')
set(ratio,'String',ratio_MgCa_del_espectro_promedio (1,1))
       handles.edit5=ratio_MgCa_del_espectro_promedio (1,1);
handles.edit1=1;
handles.pushbutton3=0; handles.pushbutton4=0;

function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5
as a double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6
as a double
LAMBDA_CALCIO=get(hObject,'String'); %almacenar el valor ingresado
LAMBDA_CALCIO=str2double(LAMBDA_CALCIO);

handles.edit6=LAMBDA_CALCIO %lo almacenamos en el identificador
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7
as a double
LAMBDA_MAGNESIO=get(hObject,'String'); %almacenar el valor ingresado
LAMBDA_MAGNESIO=str2double(LAMBDA_MAGNESIO);

handles.edit7=LAMBDA_MAGNESIO %lo almacenamos en el identificador
guidata(hObject, handles);
% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

caracteristico = '_layers'
if handles.pushbutton5==10
    PN=handles.edit2;
else
    PN='C:\Datos\Dropbox\GifLibs\Arqueo\LAN3\19112014-175923_interior'
end
% if handles.
Ncleaningshots = 70;
%PN = strcat(path,experimentPath,datePath);

LAMBDA_CALCIO = handles.edit6 % calcium line that is monitorized for
RSD
```

```
LAMBDA_MAGNESIO= handles.edit7 % magnesium line that is monitorized
for RSD

 Ncleaningshots =handles.edit4;
 Npulses=handles.edit3;
[lambdas,meanspec,eje,ratio_MgCa_del_promedio_de_ratios,ratio_MgCa_del
_espectro_promedio]=procesa(PN,Ncleaningshots, Npulses, LAMBDA_CALCIO,
LAMBDA_MAGNESIO)

%     axes(handles.axes1)
%     plot(lambdas,meanspec(:,1,1))
%     zoom on
save ('datos')
x=[LAMBDA_CALCIO,LAMBDA_CALCIO];
y=[1,1.5e4];

z=[LAMBDA_MAGNESIO,LAMBDA_MAGNESIO];
q=[1,1.5e4];

axes(handles.axes1)
plot(lambdas,meanspec(:,1,1),x,y,z,q)
zoom on
axes(handles.axes2)

plot(eje,ratio_MgCa_del_promedio_de_ratios(:,1),'LineWidth',2,'Color',
'r');
grid on;

ratio=findobj(gcbf,'tag','edit5')
set(ratio,'String',ratio_MgCa_del_espectro_promedio (1,1))
       handles.edit5=ratio_MgCa_del_espectro_promedio (1,1);
handles.pushbutton3=0;
handles.pushbutton4=0;


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```