Práctica 4: Operadores puntuales (II)		Grupo	
		Puesto	
Apellidos, nombre	**Sánchez Garzón, Laura**	**Fecha**	
Apellidos, nombre	**Remuiñán Cid, Sara**	**19/10/23**	

El objetivo de esta práctica es presentar al alumno los fundamentos de los operadores puntuales y parte de sus aplicaciones.

Desarrolle cada ejercicio en un fichero de comandos 'ejercicio_X.m' separado. Para conocer el funcionamiento preciso de los comandos que se introducen en este guion, utilice la ayuda de MATLAB. Para evitar posibles interferencias con otras variables o ventanas recuerde incluir siempre las instrucciones `clear all` y `close all` al principio de cada fichero de comandos.

Al finalizar la práctica, comprima el documento de observaciones y los ficheros '.m' generados en un único fichero con el nombre '**FTDI_P4_II_ApellidosNombre1_ApellidosNombre2.zip**', conéctese al sistema de entrega de prácticas de *Moodle* y entréguelo.

NOTA IMPORTANTE: En el desarrollo de esta práctica y posteriores se pide realizar operaciones repetidamente, por ejemplo, modificando individualmente píxeles concretos de una imagen. En estos casos se recomienda acudir al uso de estructuras de control (bucles, condiciones, etc.). Adicionalmente, debido a la creciente complejidad de los programas que se le va a pedir desarrollar, se recomienda encarecidamente el uso del depurador o *debugger* de MatLab, cuya funcionalidad está descrita en la ayuda del programa (*User's Guide - Desktop Tools and Development Environment - Editing and Debugging MATLAB Code*).

1 Operadores puntuales básicos

Negativo de una imagen en color

El negativo o inversión de una imagen es una operación puntual, es decir que afecta a cada píxel, independientemente de su vecindad. Para una imagen en escala de grises que puede tomar L posibles valores o niveles de intensidad, es decir con rango $r_k \in [0, L-1]$. El negativo de esa imagen viene definido por la transformación:

$$s_k = T(r_k) = (L-1) - r_k$$

, con $s_k \in [0, L-1]$. Es decir, puede entenderse como una inversión del eje de intensidades.

Si la imagen es en color, por ejemplo, en el espacio RGB, el negativo de la imagen se calculará realizando el negativo de cada banda independientemente.

Es decir, si cada banda tiene un rango r_k^i:

$$\mathbf{r_k} = [r_k^R \quad r_k^G \quad r_k^B],\ 0 \le r_k^i \le (L-1)$$

, el negativo se calculará como:

$$\mathbf{s_k} = [s_k^R \quad s_k^G \quad s_k^B],\ 0 \le s_k^i \le (L-1)$$

, donde:

$$s_k^i = T\left(r_k^i\right) = (L-1) - r_k^i$$

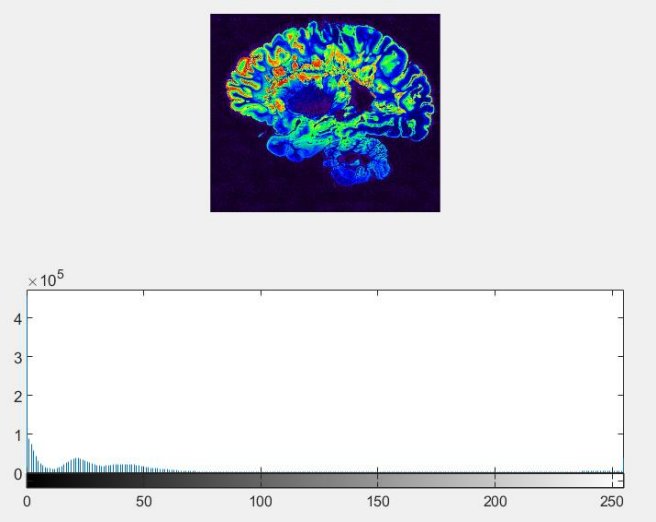
Posteriormente utilice alguno de las operaciones estudiadas en la práctica anterior para realizar la operación local. Recuerde que en la obtención de la transformación es posible que su imagen haya cambiado de tipos. Para representar tanto la imagen como su histograma con los comandos que conoce quizás debe realizar una conversión del tipo de la imagen resultante a uint8.

1.1 Ejercicio 1: obtención del negativo

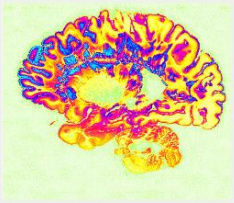
Cargue y visualice la imagen **`MRI_pseudo_colored.jpg`**[1] así como su histograma y el histograma de cada uno de sus tres canales RGB.

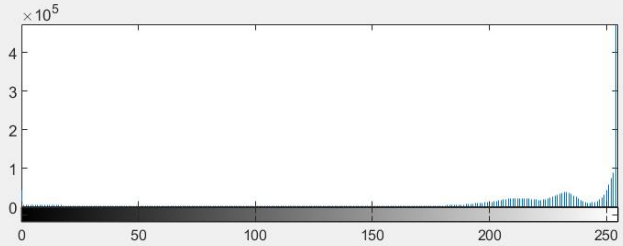
Calcule el negativo de cada canal independientemente según lo comentado en el apartado anterior, mezcle los tres negativos para crear una nueva imagen `ima_p` que contenga el negativo de la imagen original `ima`.

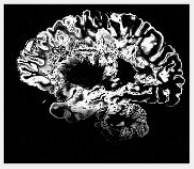
Obtenga y copie en su memoria las imágenes y los histogramas involucrados en el proceso. Observe y describa cuál es el efecto de esta transformación.

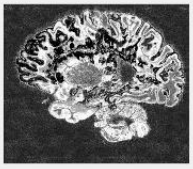


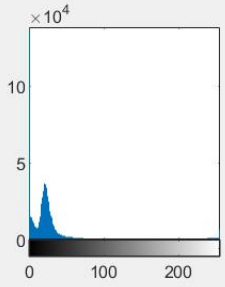
[1] Descárguese esta imagen del Moodle de la asignatura

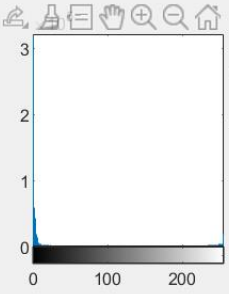


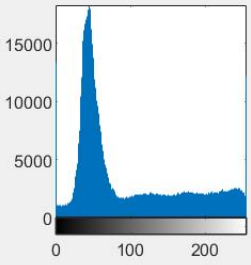

×10^5
4
3
2
1
0
0
50
100
150
200
250

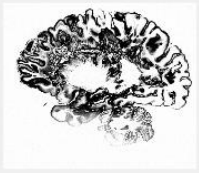


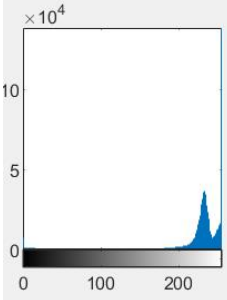


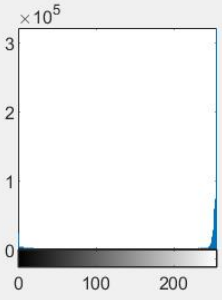

×10^4
10
5
0
0
100
200

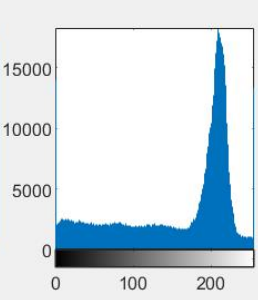

3
2
1
0
0
100
200


15000
10000
5000
0
0
100
200




×10^4
10
5
0
0
100
200


×10^5
3
2
1
0
0
100
200


15000
10000
5000
0
0
100
200

Umbralización

Como han visto en teoría, la umbralización de imágenes es un caso particular del recorte, cuyo objetivo principal es la segregación, separación o segmentación de los dos niveles o clases de una imagen bimodal a las que solemos referirnos como frente y fondo.

Para umbralizar una imagen podemos partir de conocimientos previos: por ejemplo, número y distribución de objetos o de una estimación de la probabilidad de los objetos respecto de la del fondo. Aunque, por el contrario, lo habitual es no disponer de ninguna información previa sobre la imagen ni sobre sus componentes.

En este caso, el proceso suele modelarse automáticamente partiendo de una estimación inicial del umbral y alterando el umbral en función de criterios de inter-separación entre clases o/e intra-homogeneidad de cada una de las dos clases. Este tipo de procesos pueden ser iterativos, y convergen cuando la diferencia entre el umbral anterior y el estimado es menor que un umbral de convergencia prefijado.

En este ejercicio intentaremos umbralizar una imagen en escala de grises que, en principio, no es bimodal, pero en la cual puede apreciarse una clase diferenciada.

Para comprobar la bondad de los métodos descritos a continuación cargue y represente la imagen **`MRI_gray.jpg`**[2] y su histograma.

Para cada método desarrollado copie y pegue en su memoria: la imagen original y su histograma, la imagen umbralizada, su energía y su histograma. Reflexione y comente los resultados observados. Compare visualmente el resultado de las tres aproximaciones de umbralización.

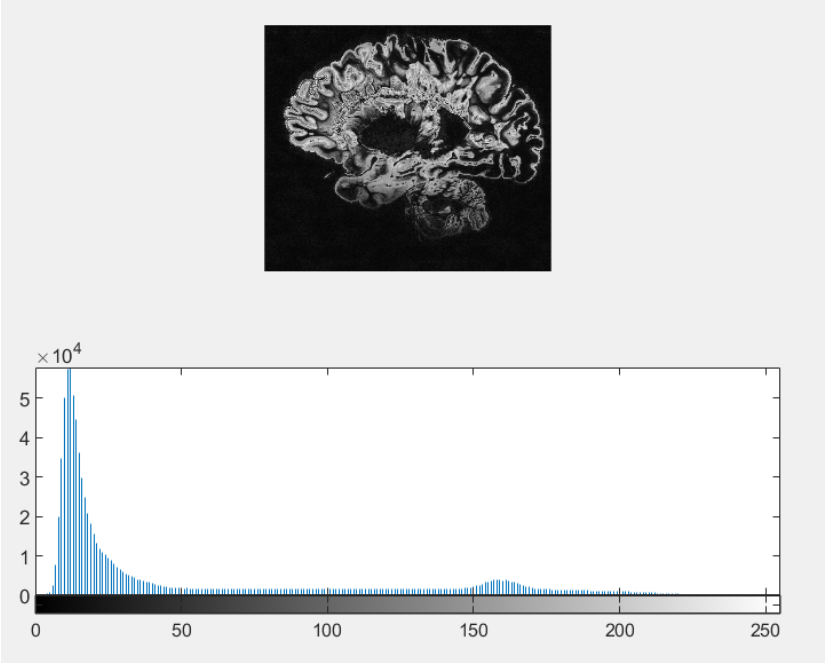
1.2 Ejercicio 2: umbralización global

Genere una función MatLab que implemente un esquema de búsqueda iterativa del umbral que equidiste de las medias superior e inferior. Le recomendamos seguir el esquema descrito en la transparencia de clase.

Esta función recibe una imagen en escala de grises (tipo `uint8`) y su mapa (si está disponible) y devuelve una máscara binaria (lógica), es decir, con los píxeles cuyo valor de gris esté por encima del umbral detectado a '1' y el resto a '0'. Ambas imágenes deberán tener un único canal.

```
function ima_th = UmbralizaGlobal(ima)
```

[2] Descárguese esta imagen del Moodle de la asignatura



1.3 Ejercicio 3: umbralización global por el método de Otsu

Genere una función MatLab que implemente un esquema de búsqueda exhaustiva del umbral que minimiza la varianza intra-clase, es decir, que fije automáticamente el umbral por el método de Otsu.

Le recomendamos sustituir en el esquema anterior el proceso iterativo por uno exhaustivo y cambiar el proceso de cálculo del umbral según lo visto en clase.

Esta función recibe una imagen en escala de grises (tipo `uint8`) y su mapa y devuelve una máscara binaria (logica), es decir, con los píxeles cuyo valor de gris esté por encima del umbral detectado a '1' y el resto a '0'. Ambas imágenes deberán tener un único canal.

```
function ima_th = UmbralizaGlobalOtsu(ima)
```

Nota: Puede comparar sus resultados con la implementación de Otsu de Matlab: graythresh(). Observe que esta función devuelve el umbral en el intervalo [0,1], por lo que tendrá que multiplicar el valor obtenido por 255 para poder comparar los resultados.

1.4 Ejercicio 4: umbralización adaptativa con ventanas no solapadas

Genere una función MatLab que implemente el ejercicio anterior sobre cada una de las divisiones –ventanas no solapadas- de tamaño *MxN* de la imagen de entrada. Experimente con distintos valores de M y *N*. Para realizar este proceso, puede hacer uso de la función `blockproc` de Matlab.

Nota: Experimente con valores de M y N que resulten en el procesado de entre 4 y 16 ventanas rectangulares (no necesariamente cuadradas). Un número superior de ventanas incrementará mucho el tiempo de proceso y es propenso a segmentar el ruido de la imagen.