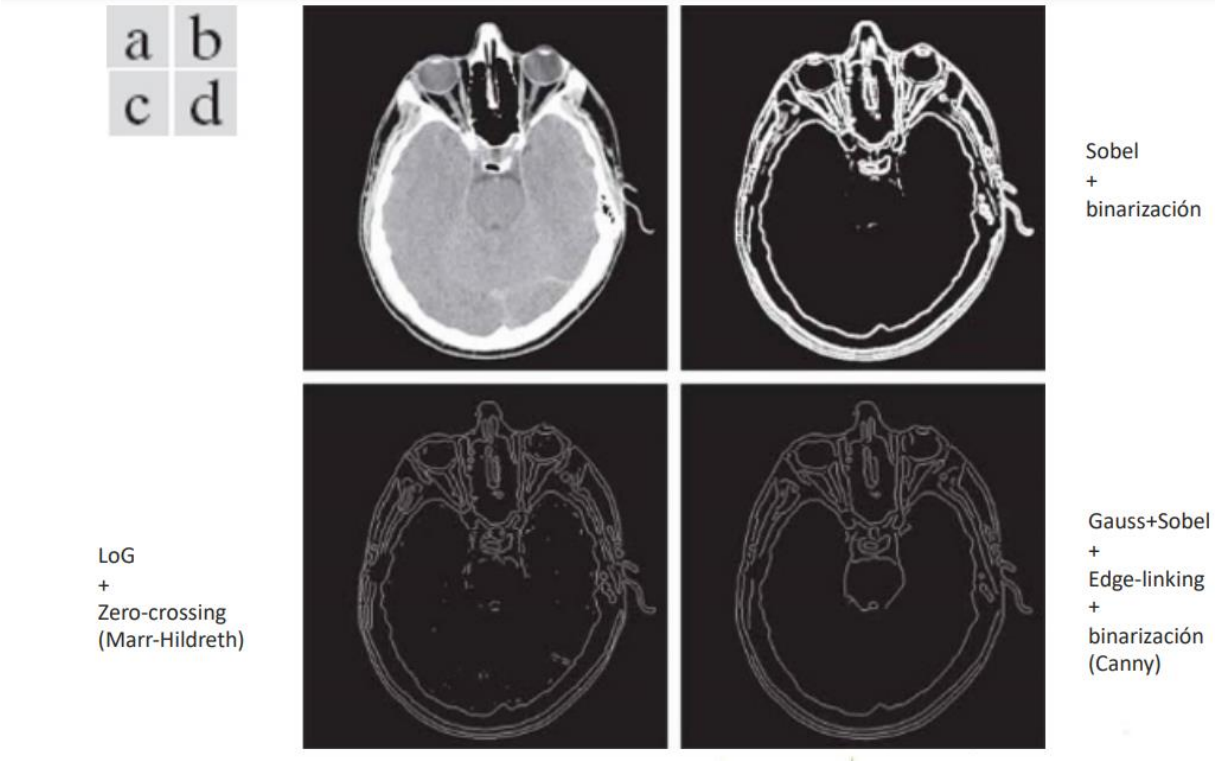
Universidad de Málaga

Materia: Imágenes biomédicas

Profesor: Ignacio Rodríguez Rodríguez y Enrique Nava Baro

Practica 3: Segmentación de imágenes



Diego De Pablo

Málaga, noviembre de 2023

índice:

Introducción:

Las imágenes biomédicas son una fuente valiosa de información para el diagnóstico y tratamiento de enfermedades. No obstante, muchas de estas imágenes resultan difíciles de reconocer para la vista de un experto, muchísimo más para una máquina. Por esta razón existe la segmentación de imágenes, es un proceso que divide una imagen digital en varias regiones o grupos de píxeles, denominados segmentos. En otras palabras, es un proceso de clasificación por píxel que asigna una categoría a cada píxel de la imagen analizada.

La segmentación de imágenes es una técnica fundamental en muchas aplicaciones de procesamiento de imágenes, como el reconocimiento de objetos, la clasificación de imágenes y la detección de defectos. En el ámbito de las imágenes biomédicas, la segmentación se utiliza en una gran variedad de aplicaciones, como el diagnóstico médico para identificar y localizar objetos de interés como tumores, lesiones o vasos sanguíneos, en investigación biomédica se puede utilizar para estudiar la evolución de una enfermedad o para identificar nuevos patrones.

La segmentación de imágenes es muy útil en distintas disciplinas, no solo esta relacionada con temas de salud. También la segmentación de imágenes es una herramienta esencial en el campo de la visión por computadora que permite a las máquinas “ver” y entender imágenes de una manera más detallada y significativa

Esta práctica tiene como objetivo introducir los conceptos básicos de la segmentación de imágenes. además, se utilizarán dos técnicas de segmentación:

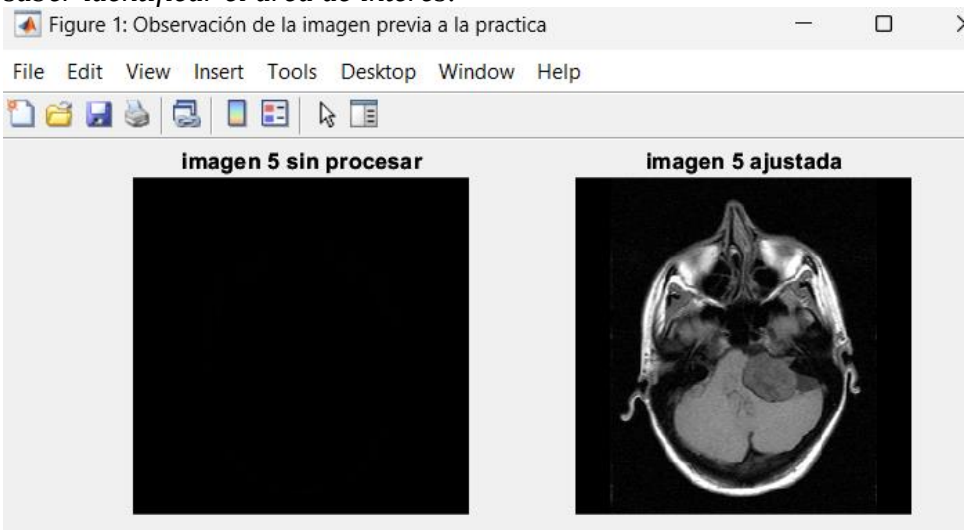
- **Umbralización**: Esta técnica se basa en asignar a cada píxel de la imagen un valor binario, dependiendo de si su nivel de gris está por encima o por debajo de un umbral determinado, esto se logrará de manera manual, automática e iterativa.
- **Superpixel**: Esta técnica divide la imagen en un conjunto de regiones de tamaño pequeño, llamadas superpíxeles, esto se lograra aplicando el utilizando el algoritmo SLIC (Simple Linear Iterative Clustering).

Cabe recordar que esta memoria trabaja con la imagen 5 correspondiente a una imagen de tumor y masa cerebral

Así mismo decir que esta memoria sigue un formato donde por temas de practicidad y continuación de la práctica se ira resolviendo los apartados dados por la guía de práctica, anexando codigo de Matlab creado por el alumno y una explicación de teoría de los métodos usados además de la comprensión dada por el alumno.

Contenido de la practica:

Antes de empezar quisiera recordar la imagen que toco y como se ve ajustada, para saber identificar el área de interés:



Cabe acotar que esta imagen a diferencia de otras. La umbralización se tendrá que hacer con más cuidado al haber tanta diferencia entre los 2 objetos de estudios. El tumor cuyo color es más opaco y la masa cerebral menos opaca, además de que el cráneo cuenta con tonalidades cercanas al objeto esto ser explicara mejor en el apartado 1.

1. Implementar un algoritmo de umbralización para obtener la estructura anatómica de interés. Para ello, se propone la realización de las siguientes tareas:
    1. Realizar una umbralización manual con los umbrales de visualización utilizados en la práctica 1 para la visualización óptima de la región de interés.

Codigo de Matlab para calcular el umbral manual:

```
%funciones
function imagen_manual = umbral_manual(imagen, valor_min, valor_max)
if nargin < 2 %si no dan valor_min se tomara como 120 caso im5
    valor_min = 120;
end
if nargin < 3%si no dan valor_max se tomara como 250 caso im5
    valor_max = 250;
end
imagen = double(imagen);
[filas, columnas]=size(imagen);
imagen_manual = zeros(filas,columnas);
for i=1:filas
    for j=1:columnas
        if imagen(i,j)> valor_min && imagen(i,j)< valor_max
            imagen_manual(i,j) = 1; %pone a 1 los pixeles en el rango
        else
            imagen_manual(i,j) = 0; %pone en negro lo demas
        end
    end
end
imagen_manual=int16(imagen_manual);
end
```

Explicación:

Umbralización: En esta sección, se implementa un algoritmo de umbralización para obtener la estructura anatómica de interés de la imagen. Este proceso incluye la realización de una umbralización manual, una automática e iterativa, y probando la diferencia umbralizando después de filtrar la imagen original con un filtro gaussiano. El objetivo de la umbralización es convertir una imagen en escala de grises a una nueva con sólo dos niveles, de manera que los objetos queden separados del fondo. Si los valores de gris del objeto y del resto de la imagen difieren claramente, entonces el histograma mostrará una distribución bimodal, con dos máximos distintos, separados por una región vacía. Con lo cual se logrará una separación perfecta entre el objeto y el fondo, al establecer un valor umbral ubicado en esta región del histograma

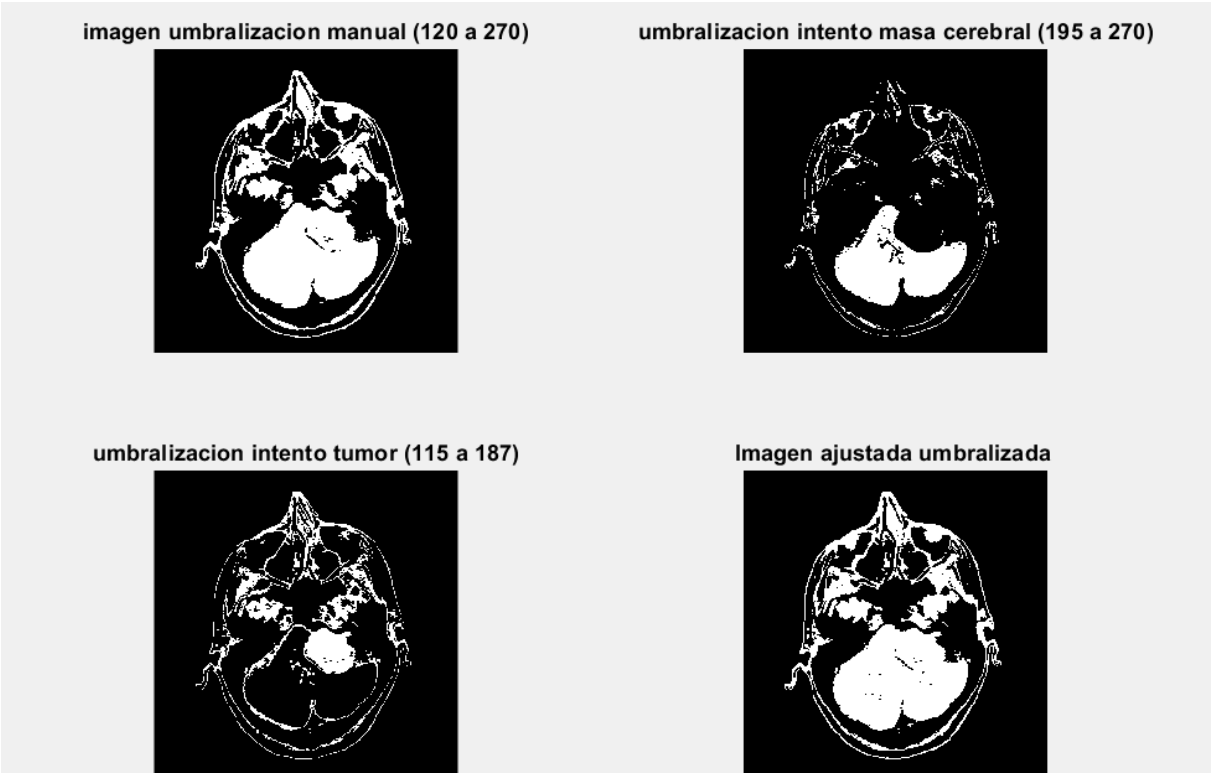
En el caso de la **imagen 5** la estructura anatómica de interés es la **masa cerebral y el tumor**, para esto podemos estudiar el histograma de la imagen para seleccionar el umbral manualmente.

En este código de MATLAB creamos una función llamada **umbral_manual**. La cual pide de parámetros una imagen a la cual umbralizaremos, en nuestro caso es la imagen 5, también como es una umbralización manual tenemos la opción de pasar los parámetros del umbral, siendo el mínimo y máximo el margen que queremos, esto es opcional ya que en caso de no darle a la función dichos valores, entrara a los if nargin<2 y nargin<3 donde se le asignara los valores de 120 y 270 respectivamente. Ya una vez introducidos los parámetros, si convierte la imagen a doble precisión con el comando double(), que permite operaciones matemáticas más sencillas, y mediant size se obtiene la cantidad de filas y columnas que tiene la matriz de la imagen, esto será útil para crear bucles con la cantidad necesaria de iteraciones para buscar todos los valores de la matriz, a su vez se crea una matriz de ceros del tamaño de la imagen original. Esta se usara para que en el bucle anidado al recorrer cada pixel de la imagen original y se evalúa si está dentro del umbral asignado, en caso positivo se establece en 1 en la matriz zero, al haber recorrido todos los pixeles se convierte la nueva matriz en enteros de 16 bits. Dándonos por resultado una matriz binaria donde se mostrarán en blanco todos los valores que nos interesan

Codigo de Matlab para comparar las umbralizaciones manuales :

```
%% 1.1 umbralizacion manual
%Al trabajar con una imagen desconocida lo mejor es ver el histograma
figure('Name', 'Histograma de la imagen:'); imhist(imagen)

%se podría usar el imshow(imagen,[120 270]) pero pienso que esto quedaba
%muy simple y no sé podía controla tan bien la salida binaria

%comparacion
figure('Name', 'Distintos valores para la misma imagen:')
subplot(2,2,1);
imshow(umbral_manual(imagen),[])
title('imagen umbralizacion manual (120 a 270)')
subplot(2,2,2);
imshow(umbral_manual(imagen, 195, 270),[])
title('umbralizacion intento masa cerebral (195 a 270)')
subplot(2,2,3);
imshow(umbral_manual(imagen, 115, 187),[])
title('umbralizacion intento tumor (115 a 187)')
subplot(2,2,4);
imshow(umbral_manual(imagen_ajustada, 17000, 38000),[])
title('Imagen ajustada umbralizada ') %Comparamos como se vería si umbralizamos
% una imagen ajustada (debería verse peor)
```

Resultado:



Explicación:

Como se explicó previamente según el umbral que se usemos se puede separar el objeto de estudio del fondo o de otras cosas que no sean de nuestro interés, en el caso de la imagen 5 es más complicado poder separar fácilmente la masa cerebral y el tumor del cráneo, esto a razón de que parte del cráneo comparte escalas de grises lo cual hace que a pesar de umbralizar siga apareciendo. También que si queremos una imagen donde aparezca la masa cerebral y el tumor, el umbral necesario es muy amplio lo cual termina mostrando gran parte del cráneo prácticamente solo se elimina el ruido de fondo y ciertas partes del cráneo.

En las imágenes de arriba a la derecha y abajo a la izquierda se intentan umbralizar por separado el tumor y la masa cerebral, se puede ver que ahora hay menor cantidad de cráneo y podría intentar limpiarse mediante técnicas de post procesados para luego sumarse y que queden solamente la masa cerebral y el tumor, pero esto no compete exactamente este apartado

Por ultimo destacar y comparar la imagen de arriba a la izquierda en la cual se umbralizo la imagen sin un ajuste automatico con la imagen de abajo a la derecha en la cual si fue previamente ajustada. Podemos ver como la imagen es peor si la ajustamos, encontrando más huecos en el objeto de interés y aparición de más pixeles en los elementos que deseábamos quitar, además que la forma del objeto de interés cambia ligeramente. Es decir que al momento de umbralizar es mejor trabajar con la imagen original si nos importa resaltar correctamente un objeto.

2. Realizar una umbralización automática e iterativa, utilizando algún método que permita el cálculo automático de los umbrales a partir de la información obtenida con el histograma. (Por ejemplo, puede usar la idea de la transparencia 16 de la clase 3.1 Segmentación, el algoritmo de la transparencia 20 u otro que se adapte mejor a su problema. Nota: es muy posible que deba aplicarlo varias veces para poder separar cada región del histograma, si tiene más de dos picos).

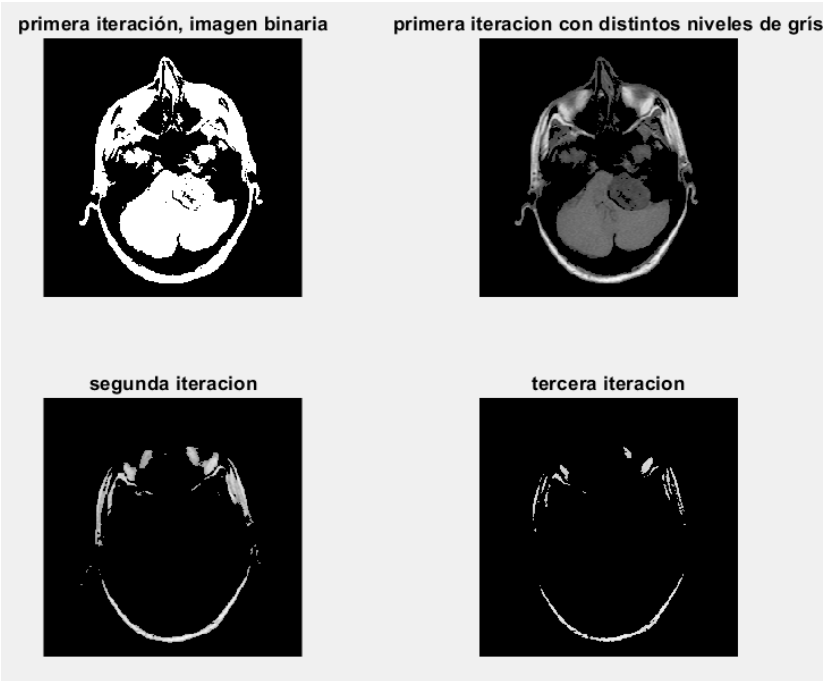
Explicación:
Para este apartado hice un programa de automatización automática para un caso más general siguiendo un algoritmo del tipo

1. De la imagen de entrada seleccionamos los pixeles que sean mayores a 0 (solo positivos)
2. Establecemos un umbral inicial llamado T que viene a ser la media de la matriz (contando que en el paso 1 trabajamos con una nueva matriz sin 0 como elemento)
3. Se hace la segmentanción diferenciando entre los niveles de gris superiores a T y los niveles de gris inferiores a T
4. Calculamos el nuevo umbral T utilizando las medias de los dos grupos entre 2
5. Se repite el bucle hasta que la diferencia de los dos sea menor que un grado de aceptación establecido
6. Se segmenta con imbinarize() finalmente

Codigo de Matlab:

```
function imagen_automatica = umbral_automatico(imagen,grado_de_aceptacion)
if nargin < 2 %si no dan un segundo valor se pone automatico
    grado_de_aceptacion = 0;5;
end
%Segmentación con umbral automático
imagen=double(imagen);
cnt=0;% Contador de Iteracioenes
entrada=imagen;
ent=entrada(entrada>0);
T = mean2(ent); % Calculamos el umbral inicial
condicion = true;
while condicion % bucle hasta que tenga un umbral decente
 cnt=cnt+1; %en caso de querer evitar bucles infinitos, podría usar el cnt
 %para poner un numero maximo de repeticiones
 imagen_automatica = ent>T;
 medI = mean(ent(~imagen_automatica)); % Media de los inferiores al umbral
 medS = mean(ent(imagen_automatica)); % Media de los superiores al umbral
 Tsig = 0.5*(medI+medS); % Umbral para la siguiente iteracion
 condicion=abs(T-Tsig)>grado_de_aceptacion;
 T=Tsig;
end
imagen_automatica = imbinarize(uint16(entrada),T/65535);
end
```

Resultado:



Explicación:
Este algoritmo sigue una metodología muy automática, y podría servir para muchas imágenes donde la moda corresponda con el objeto de que estudiaremos. Donde usualmente resalta muy bien la estructura ósea.

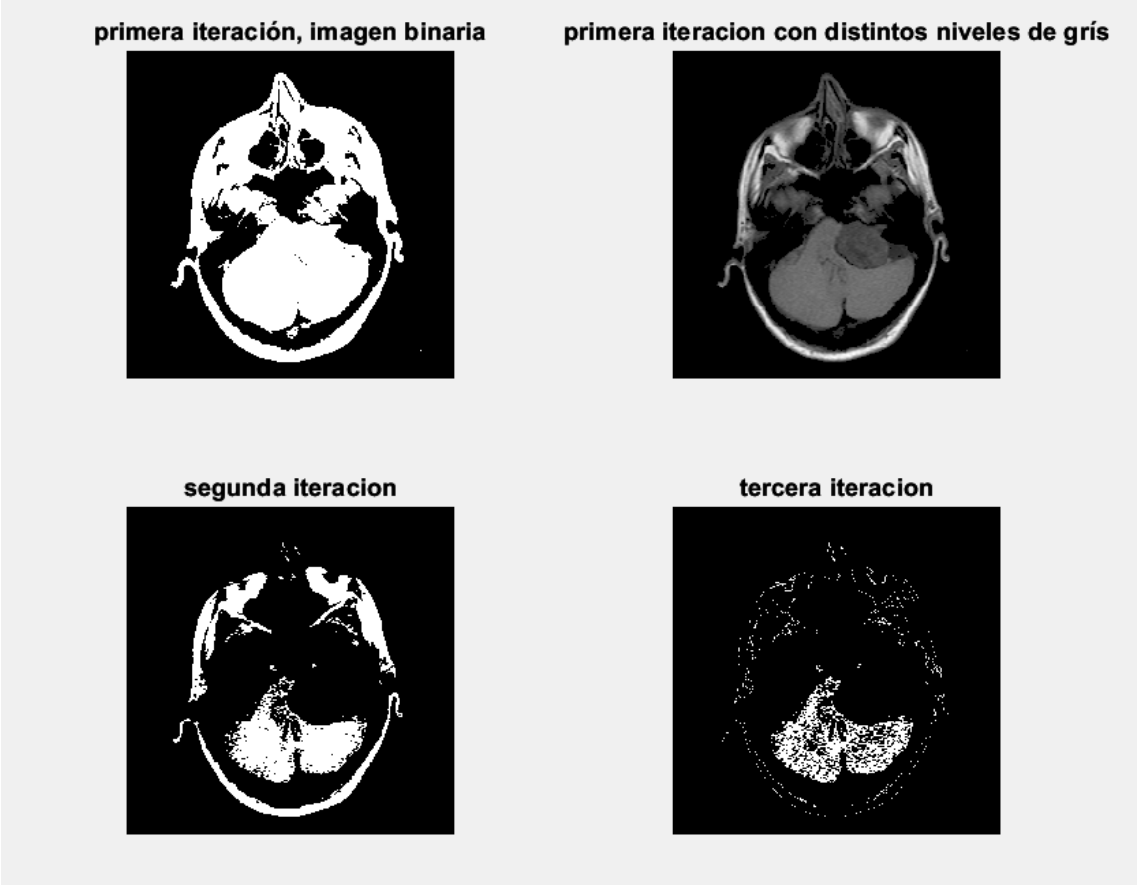
Lástima que para nuestro caso nos interesa estudiar la masa cerebral y el tumor, entonces la segunda iteración tendremos que hacer una modificación para elegir un umbral mayor, ya podemos ir viendo que una resonancia magnética del cráneo no es un tipo de imagen que sea tan facil de umbralizar automáticamente, ya que necesitara un preprocesado o en su defecto una modificación ligera al código.

Codigo de Matlab:

```
%probar en caso de que el objeto de estudio se va del umbral automatico
function imagen_automatica = umbral_automatico_inferior(imagen,grado_de_aceptacion)
if nargin < 2 %si no dan un segundo valor se pone automatico
    grado_de_aceptacion = 0.5;
end
%Segmentación con umbral automático
imagen=double(imagen);
cnt=0;% Contador de Iteraciones
entrada=imagen;
ent=entrada(entrada>0);
T = mean2(ent); % Calculamos el umbral inicial
condicion = true;
while condicion % bucle hasta que tenga un umbral decente
 cnt=cnt+1; %en caso de querer evitar bucles infinitos, podría usar el cnt
 %para poner un numero maximo de repeticiones
 imagen_automatica = ent<T; % Aquí cambiamos la lógica del umbral
 Tsig= mean(ent(~imagen_automatica)) % Media de los inferiores al umbral
  % Umbral para la siguiente iteracion
 condicion=abs(T-Tsig)>grado_de_aceptacion;
 T=Tsig;
end
imagen_automatica = imbinarize(uint16(entrada),T/65535);
end
```

```
imagen_automatica=umbral_automatico(imagen); %debería eliminar el fondo
f1=imagen.*uint16(imagen_automatica);
%imshow(f1,[])
imagen_automatica2=umbral_automatico(f1);  %eliminar tumor y parte del craneo
%imshow(f2,[])
imagen_automatica3=umbral_automatico_inferior(f2); % ver que tal


%comparacion
figure('Name', '1)algoritmo automatico')
subplot(2,2,1);
imshow(imagen_automatica,[])
title('primera iteración, imagen binaria')
subplot(2,2,2);
imshow(f1,[])
title('primera iteracion con distintos niveles de grís')
subplot(2,2,3);
imshow(imagen_automatica2,[])
title('segunda iteracion')
subplot(2,2,4);
imshow(imagen_automatica3,[])
title('tercera iteracion')
```

Resultado:



Explicación:

En este caso la umbralización automática fue ajustada para a partir de la segunda iteración buscar el umbral más bajo, esto se debe porque la umbralizacion automática anterior nos llevaba al umbral justo de la zona del cráneo que eran cercanos 300, ahora el T final que se usa para umbralizar está por los 200, con esta modificación pasa de borrarse la masa cerebral a borrar la parte del cráneo que teníamos anteriormente, con una tercera iteración hace un buen limpiado que nos deja pie a usar un filtro morfológico durante el postprocesado para eliminar los pixeles restantes del cráneo mediante erosión y rellenar los huecos de la masa cerebral mediante una dilatación, podría usarse una operación de cierre.

Se podría hacer lo mismo para obtener el tumor, pero termina siendo ir viendo si quedarnos con el umbral superior o inferior y a mi perspectiva me parece que para llegar a ese punto de molestias es mejor aplicar otro método para tratar las imágenes o directamente hacerlo por umbralización manual, las automáticas no son muy buenas para este tipo de imágenes.

Cabe destacar que intentando encontrar una forma automática de umbralizar la masa cerebral con un mismo método capaz de darnos la masa cerebral programe los siguientes algoritmos vistos en clases.
Codigo de Matlab:

```
function imagen_otsu = metodo_otsu(imagen)
imagen=double(imagen);
grises = mat2gray(imagen);
[level, ~]= graythresh(grises);

mejora = round(grises);
mejora2 = uint8(mejora);
imagen_otsu = imbinarize(mejora2, level/255);
end
```

Resultado:

```
%% Métodod de Otsu por MATLAB
imagen_manual=umbral_manual(imagen);
figure('Name', 'Imagen Otsu')
imshow(metodo_otsu(imagen))
title('imagen con el método otsu ')
```

Explicación:

El método de Otsu calcula el valor umbral de forma que la dispersión dentro de cada segmento sea lo más pequeña posible, pero al mismo tiempo la dispersión sea lo más alta posible entre segmentos diferentes. Este método es especialmente útil para separar el texto de un documento del fondo de la imagen.

Este método funciona al igual que mi primer algoritmo de automatización, pero sin tener que iterar para obtener esta imagen. Vemos que sigue con el mismo problema de que se borro el objeto del estudio así que para esta imagen no fue útil. Pero es otra forma de implementar la umbralización, como ya se programo y probo me pareció un buen añadido poner las variantes que hice al intentar crear un método que pudiera llegar a la masa cerebral sin ningún otro parámetro más que la imagen a umbralizar.

Codigo de Matlab:

```
function imagen_global = umbral_global_treshold(imagen)
%Segmentación con umbral automático
imagen=double(imagen);
cnt=0;% Contador de Iteracioenes
grises = mat2gray(imagen); %convierte la imagen en escala de grises que
% contiene valores en el intervalo 0 (negro) a 1 (blanco).
T=mean2(grises);
condicion = true;
while condicion % bucle hasta que tenga un umbral decente
 cnt=cnt+1; %en caso de querer evitar bucles infinitos, podría usar el cnt
 %para poner un numero maximo de repeticiones
 imagen_automatica = grises>T;
 medI = mean(grises(imagen_automatica)); % Media de los inferiores al umbral
 medS = mean(grises(~imagen_automatica)); % Media de los superiores al umbral
 Tsig = 0.5*(medI+medS); % Umbral para la siguiente iteracion
 condicion=abs(T-Tsig)~=0;
 T=Tsig;
end
mejor = round(grises); % Redondeamos la Imagen
mejo3 = uint8(mejor); % La trasformamos a enteros de 8 bits
imagen_global = imbinarize(mejo3, T/255); % Binarizamos la imagen según el threshhold
end
```

Resultado:

```
%% Método de Global Threshold Progrmamado
imagen_global_treshold=umbral_global_treshold(imagen);
figure('Name', 'método de global treshold')
imshow(imagen_global_treshold,[])
title('imagen con el método de Global Threshold Programamado')
```

Explicación:

En el método global del valor umbral, es un algoritmo más general para umbralizar de este se tomó referencia para crear los algoritmos de umbralización superiores, pero esta variación también nos deja una umbralización más directa sin necesidad de volver a llamar el método, Este algoritmo consiste en que se elige un valor umbral según a media de los elementos de la imagen. Este método es el más fácil de calcular, pero también es muy sensible a las pequeñas variaciones que puedan existir en la luminosidad de la imagen. El método global, por lo tanto, sólo se utiliza para segmentar imágenes con mucho contraste.

3. Realizar la umbralización automática e iterativa, con un algoritmo similar al del apartado anterior, pero después de filtrar previamente la imagen original con un filtro gaussiano de 5x5, con sigma=1.

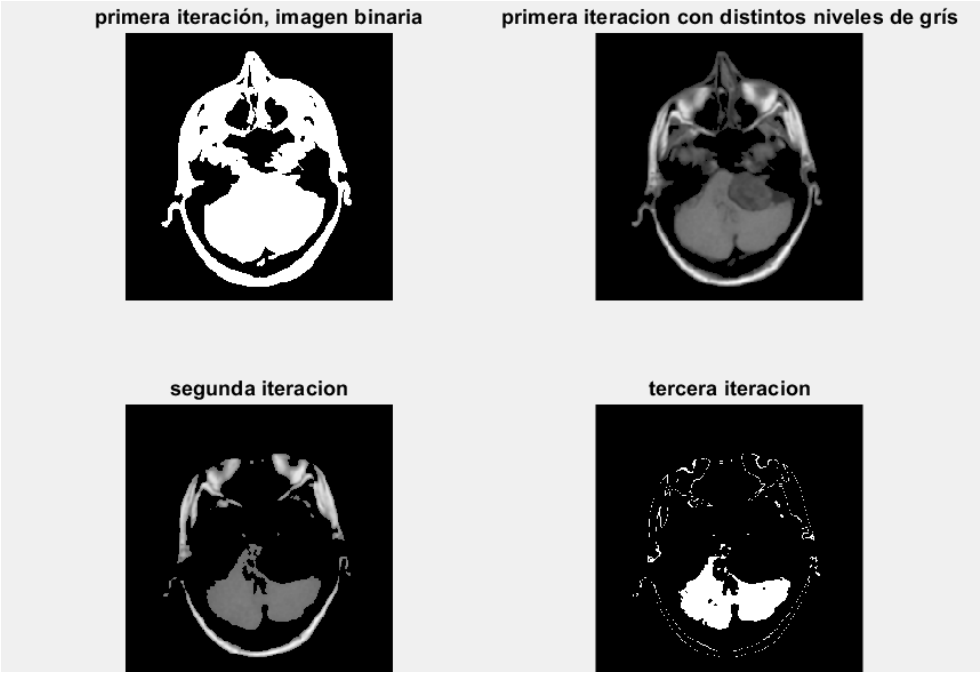
Explicación:

Recordando de la practica 2 **un filtro gaussiano** es un tipo de filtro que se utiliza en el procesamiento de imágenes y señales. Este filtro tiene una respuesta al impulso que es una función gaussiana (o una aproximación a ella). En términos más sencillos, el filtro gaussiano se utiliza para suavizar una imagen. Este filtro produce un suavizado más uniforme que el filtro de media. Es especialmente útil para reducir el ruido y los detalles de una imagen, lo que puede ayudar a resaltar las características importantes de la imagen

Codigo de Matlab:

```
%Filtro gaussiano de 5x5, con sigma=1.
filtro_gaussiano = fspecial('gaussian', [5 5], 1);
imagen_filtrada = imfilter(imagen, filtro_gaussiano);

%title('filtrada y umbralizada')
imagen_automatica=umbral_automatico(imagen_filtrada);
f1=imagen_filtrada.*uint16(imagen_automatica);

imagen_automatica2=umbral_automatico(f1);
f2=f1.*uint16(imagen_automatica2);

imagen_automatica3=umbral_automatico_inferior(f2);
f3=f2.*uint16(imagen_automatica3);

%comparacion
figure('Name', '1)algoritmo automatico')
subplot(2,2,1);
imshow(imagen_automatica,[])
title('primera iteración, imagen binaria')
subplot(2,2,2);
imshow(f1,[])
title('primera iteracion con distintos niveles de grís')
subplot(2,2,3);
imshow(f2,[])
title('segunda iteracion')
subplot(2,2,4);
imshow(f3,[])
title('tercera iteracion')
```

Resultado:



Explicación:
Como era de esperarse vemos una gran mejora en los mismos métodos usados en el apartado 2, esto es debido a que el filtro gaussiano es muy útil en la reducción de ruidos, que en métodos donde usamos umbrales automáticos, nos ayuda a elegir mejores umbrales. Además, que gran cantidad de pixeles tendrán valores más relacionados con los pixeles de su alrededor, dejando de resultado que la masa cerebral ya no cuente con tantos aguajeros por pixeles que se encuentren fuera del umbral.

Podemos concluir que el filtro gaussiano es un buen filtro para hacer preprocesamiento que ayuda a obtener mejores resultados.

2. Utilizar el algoritmo SLIC (simple linear iterative clustering) para generar superpíxeles sobre la imagen utilizada y evaluar el efecto que su número tiene sobre el resultado. Puede utilizar la función de Matlab llamada superpixels(im,N). Esta función da un buen resultado cuando los superpíxeles delimitan con nitidez la zona de interés, ya que puede aplicarse posteriormente un análisis cuantitativo de cada región delimitada. El objetivo es encontrar un valor de N adecuado para ello.

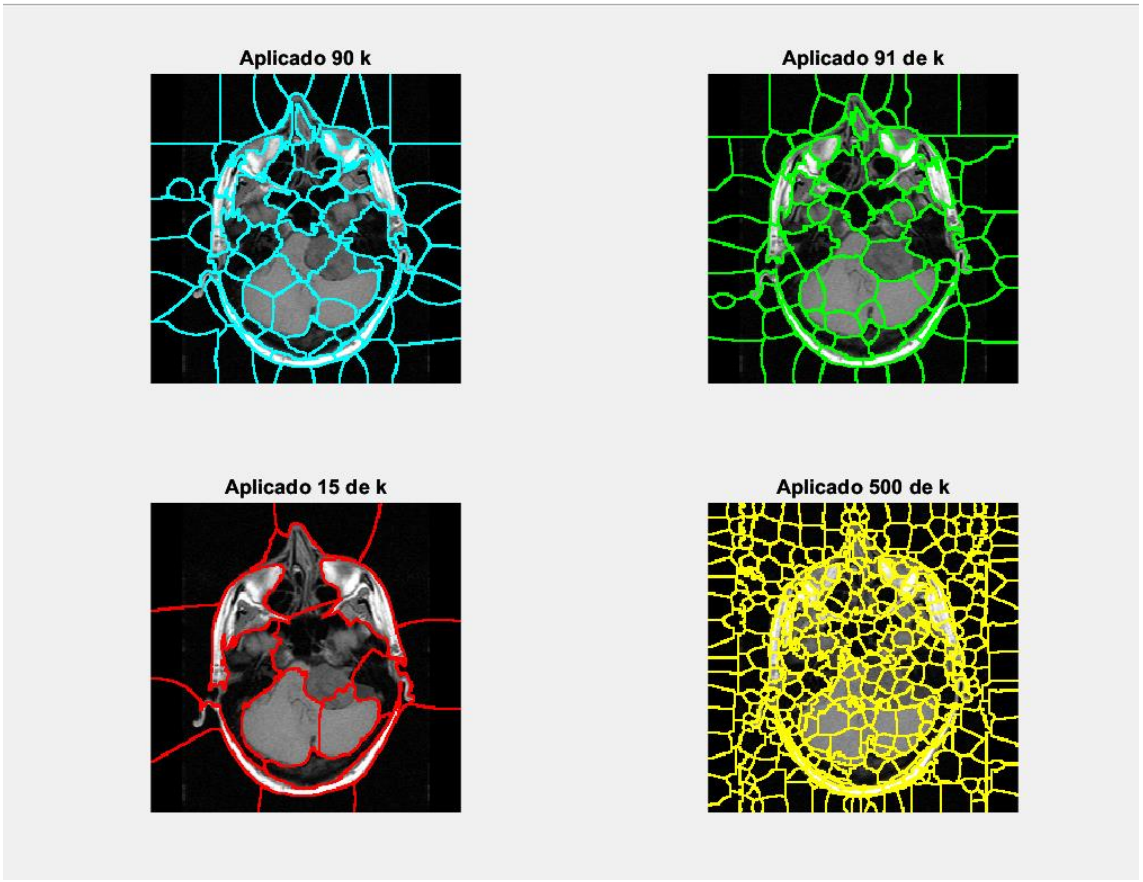
Explicación:
Algoritmo **SLIC** (Simple Linear Iterative Clustering): En esta parte, se utiliza el algoritmo SLIC para generar superpíxeles en la imagen y evaluar el efecto que su número tiene sobre el resultado. El objetivo es encontrar un valor de N (número de superpíxeles) que permita una definición sencilla de la región de interés.

El algoritmo SLIC utiliza el método de agrupamiento de k-medias para generar superpíxeles de manera eficiente1. Aunque es simple, SLIC puede obtener límites mejor que los algoritmos anteriores. Al mismo tiempo, tiene una velocidad más rápida, mayor eficiencia de memoria y puede mejorar el rendimiento de la segmentación

Codigo de Matlab:

```
%% Utilizar el algoritmo SLIC
[L, ~] = superpixels(imagen_ajustada, 90);
mask = boundarymask(L);

[L2, ~] = superpixels(imagen_ajustada, 91);
mask2 = boundarymask(L2);

[L3, ~] = superpixels(imagen_ajustada, 15);
mask3 = boundarymask(L3);

[L4, NumLabels] = superpixels(imagen_ajustada, 500);
mask4 = boundarymask(L4);

%comparacion
figure('Name', 'Comparación de distintos N')
subplot(2,2,1);
imshow(imoverlay(imagen_ajustada,mask,'cyan'),'InitialMagnification',50)
title('Aplicado 90 k')
subplot(2,2,2);
imshow(imoverlay(imagen_ajustada,mask2,'green'),'InitialMagnification',50)
title('Aplicado 91 de k')
subplot(2,2,3);
imshow(imoverlay(imagen_ajustada,mask3,'red'),'InitialMagnification',50)
title('Aplicado 15 de k')
subplot(2,2,4);
imshow(imoverlay(imagen_ajustada,mask4,'yellow'),'InitialMagnification',50)
title('Aplicado 500 de k')
```

Resultado:



Explicación:
Esto es un ejemplo del uso de superpixel y el manejo de máscaras, en este caso elegimos el k de manera manual, podemos ver en los casos inferiores que sucede al escoger erróneamente un valor de k, abajo a la izquierda el caso de escoger un k muy baja, que la imagen empieza a ser delimitada pero se agrupan bastantes zonas que no deberían relacionarse como es el caso del tumor con parte del cráneo y del fondo, siendo recomendable aumentar el número para que se puedan distribuir de una mejor manera, en el caso de abajo a la derecha es el caso contrario, que pasa si aplicamos un k elevado como puede ser 500, 100 o más, pues en esta imagen lo vemos, que si es cierto que ya cada zona esta dividida y que no sé mezclan zonas que no deberían como antes, pero si hay una cantidad absurda de divisiones, que pierden la esencialidad del objeto, con tantas divisiones regiones que deberían ser un mismo superpixel al ser el mismo objeto con misma tonalidad y cercanos, serán catalogados como superpixeles diferentes. Al aumentar mucho el número puede pasar como la imagen que pasamos de no tener zona para el tumor a tener 12 subzonas de este. También hay que considerar que no está “mal” tener una cantidad elevada de superpixeles no obstante que al aumentar la cantidad cada vez es un mayor gasto computacional y se parece cada vez más a la imagen original, dependiendo del objetivo se puede recurrir a aumentar el número de superpixeles,

Por otra parte, en la parte superior tenemos el inicio de un margen de números adecuados para el valor de k, según mi criterio, podemos ver que el cambio de 90 a 91 es que el tumor se clasifica perfectamente, la masa cerebral tiene varias divisiones pero es de esperarse al tener distintos valores en sus pixeles, etc. Este valor capaz cuenta con muchas divisiones, pero hace que el objeto de estudio se vea muy bien delimitado
Codigo de Matlab:

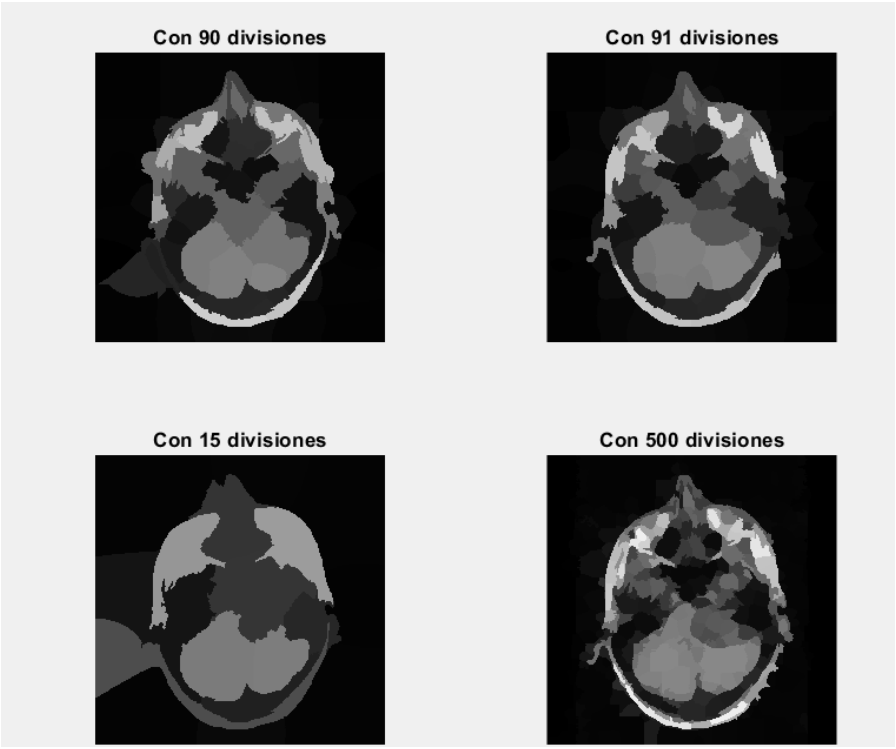
Codigo de Matlab:

```
function  salida = SLIC(imagen_ajustada,N)
[L, NumLabels] = superpixels(imagen_ajustada, N);
%mejorando la imagen
salida=zeros(size(imagen_ajustada),'like',imagen_ajustada);
idx=label2idx(L);
%[filas, columnas]=size(imagen_ajustada);
for i=1:NumLabels
    redidx=idx{i};
    salida(redidx)=mean(imagen_ajustada(redidx));
end
end
```

Llamamos la función:

```
SLIC1=SLIC(imagen_ajustada,90);
SLIC2=SLIC(imagen_ajustada,91);
SLIC3=SLIC(imagen_ajustada,15);
SLIC4=SLIC(imagen_ajustada,500);
%comparacion
figure('Name', 'Comparación al calcular el valor medio de cada superpixel')
subplot(2,2,1);
imshow(SLIC1,'InitialMagnification',100)
title('Con 90 divisiones')
subplot(2,2,2);
imshow(SLIC2,'InitialMagnification',100)
title('Con 91 divisiones')
subplot(2,2,3);
imshow(SLIC3,'InitialMagnification',100)
title('Con 15 divisiones')
subplot(2,2,4);
imshow(SLIC4,'InitialMagnification',100)
title('Con 500 divisiones')
```

Resultado:



Explicación:

Se implemento el algoritmo SLIC visto previamente para la segmentación de superpixeles, donde tomando una imageny el número de superpixeles que se quieran tener, la función se encarga de generar los superpixeles en la imagen, y posteriormente crear una imagen del mismo tamaño en la cual se dividirá por los mismos superpixeles, en la imagen original se obtendrá cada valor dentro de un superpixel para calcular la media y este valor remplazarlo en el superpixel de la nueva imagen. Dando por resultado pintar el superpixel dependiendo de la media de sus pixeles interiores.

El uso de 15 divisiones hace perder toda la información, la distribución de 91 superpixeles deja una imagen muy simplificada, pero puede ser útil para denotar de mejor manera los limites entre cada objeto reconocible en la imagen, en el caso de las 500 divisiones tenemos una imagen muy parecida a la original, esto debido a que tantas divisiones hacen que los superpixeles cuenten con pocos pixeles, lo cual le da este efecto de parecer nuestra imagen original pero desenfocada.

Conclusión

La práctica 3 de imágenes biomédicas es una práctica fundamental para la comprensión de los conceptos básicos de la segmentación de imágenes. La segmentación es una técnica esencial en muchas aplicaciones de imágenes biomédicas, como el diagnóstico médico, la investigación biomédica y la visualización de imágenes.

La **segmentación de imágenes** es una técnica crucial en el procesamiento de imágenes, especialmente en el ámbito biomédico. Permite delimitar los contornos de los objetos de interés en las imágenes, facilitando así su análisis y estudio. En la práctica, se ha aplicado técnicas de segmentación para identificar y extraer la masa cerebral y un tumor de dicha imagen.

La **umbralización** es una técnica de segmentación que se utiliza para convertir una imagen en escala de grises en una imagen binaria. En la práctica, se ha implementado un algoritmo de umbralización para obtener la estructura anatómica de interés. Mediante la experimentación de la umbralización manual, la umbralización automática e iterativa, y la umbralización automática e iterativa después de filtrar la imagen original con un filtro gaussiano. Se pudo aprender que a veces la umbralización puede llegar a complicarse según el tipo de imagen con la que se trabaje, siendo aquellas donde existe mucho ruido, poca diferencia de tonalidades de grises y mala separación entre los objetos de estudios, más complicadas de umbralizar correctamente.

El algoritmo **SLIC** (Simple Linear Iterative Clustering) es una técnica de segmentación que genera superpíxeles. Los superpíxeles son agrupaciones de píxeles que comparten características similares. En tu práctica, has utilizado el algoritmo SLIC para generar superpíxeles en la imagen y has evaluado el efecto que el número de superpíxeles tiene sobre el resultado. Este algoritmo es especialmente útil cuando los superpíxeles delimitan con nitidez la zona de interés, ya que permite realizar un análisis cuantitativo de cada región delimitada.

En resumen, la segmentación de imágenes, la umbralización y el algoritmo SLIC son técnicas poderosas que permiten extraer información valiosa de las imágenes biomédicas. Su correcta aplicación puede facilitar el diagnóstico y el estudio de diversas condiciones médicas. A través de esta práctica, has adquirido una comprensión más profunda de estas técnicas y de cómo se pueden utilizar en el campo de las imágenes biomédicas, además de entender que no todos los métodos son perfectos, son herramientas que dependiendo el tipo de imagen que estemos trabajando será más o menos útil.

Bibliografía recomendada

- Gonzalez, R.C., Woods, R.E. (2007). Digital Image Processing, 3ed. Prentice-Hall.
- Gonzalez, R.C., Woods, R.E., Eddins, S.L. (2009). Digital Image Processing using MATLAB, 2ed. Prentice-Hall.
- Pratt, W.K. (2007). Digital Image Processing, 4ed. Wiley.

- Chan, T.F., Vese, L. (2001). Active Contours Without Edges, IEEE Transactions on Image Processing, vol 10, no 2, pp. 266-276.
- Guion de laboratorio para la practica 3 de ingeniería de la salud

Y gracias a las clases de Enrique Nava Baro profesor de imágenes biomédicas en la universidad de Málaga

Código: (ver mejor el propio archivo de Matlab en la carpeta zip)

```
%Practica3 imagenes biomedicas
%Borramos las variables cargadas y cargamos la imagen
clear variables
clc
close all

%Cargamos la imagen y la ajustamos automaticamente
imagen=dicomread('im5'); %imagen asignada (sin procesar)
imagen_ajustada=imadjust(imagen); %ajuste automatico
%comparacion
figure('Name', 'Observación de la imagen previa a la practica')
subplot(1,2,1);
imshow(imagen)
title('imagen 5 sin procesar')
subplot(1,2,2);
imshow(imagen_ajustada)
title('imagen 5 ajustada')
%% 1.1 umbralizacion manual
%Al trabajar con una imagen desconocida lo mejor es ver el histograma
figure('Name', 'Histograma de la imagen:'); imhist(imagen)

%se podría usar el imshow(imagen,[120 270]) pero pienso que esto quedaba
%muy simple y no sé podía controla tan bien la salida binaria

%comparacion
figure('Name', 'Distintos valores para la misma imagen:')
subplot(2,2,1);
imshow(umbral_manual(imagen),[])
title('imagen umbralizacion manual (120 a 270)')
subplot(2,2,2);
imshow(umbral_manual(imagen, 195, 270),[])
title('umbralizacion intento masa cerebral (195 a 270)')
subplot(2,2,3);
imshow(umbral_manual(imagen, 115, 187),[])
title('umbralizacion intento tumor (115 a 187)')
subplot(2,2,4);
imshow(umbral_manual(imagen_ajustada, 17000, 38000),[])
title('Imagen ajustada umbralizada ') %Comparamos como se vería si
umbralizamos
% una imagen ajustada (debería verse peor)

%% 1.2 Umbralizacion automatica
imagen_automatica=umbral_automatico(imagen); %debería eliminar el fondo
f1=imagen.*uint16(imagen_automatica);
%imshow(f1,[])
imagen_automatica2=umbral_automatico(f1);  %eliminar tumor y parte del
craneo
```

```
f2=f1.*uint16(imagen_automatica2);
%imshow(f2,[])
imagen_automatica3=umbral_automatico(f2); %
f3=f2.*uint16(imagen_automatica3);
%imshow(f3,[])
%Se podría decir que las imagenes_automaticas son imagenes umbralizadas
%binarias y las F1, F2, F3 son esas iteraciones multiplicadas por la
imagen
% para obtener los valores reales

%comparacion
figure('Name', '1)algoritmo automatico')
subplot(2,2,1);
imshow(imagen_automatica,[])
title('primera iteración, imagen binaria')
subplot(2,2,2);
imshow(f1,[])
title('primera iteracion con distintos niveles de grís')
subplot(2,2,3);
imshow(f2,[])
title('segunda iteracion')
subplot(2,2,4);
imshow(f3,[])
title('tercera iteracion')

%% Métodod de Otsu por MATLAB
imagen_manual=umbral_manual(imagen);
figure('Name', 'Imagen Otsu')
imshow(metodo_otsu(imagen))
title('imagen con el método otsu ')


%% Método de Global Threshold Progrmamado
imagen_global_treshold=umbral_global_treshold(imagen);
figure('Name', 'método de global treshold')
imshow(imagen_global_treshold,[])
title('imagen con el método de Global Threshold Programamado')
%% apartado 3: Umbralización automática ietrativa de una imagen filtrada
%Filtro gaussiano de 5x5, con sigma=1.
filtro_gaussiano = fspecial('gaussian', [5 5], 1);
imagen_filtrada = imfilter(imagen, filtro_gaussiano);

%title('filtrada y umbralizada')
imagen_automatica=umbral_automatico(imagen_filtrada);
f1=imagen_filtrada.*uint16(imagen_automatica);

imagen_automatica2=umbral_automatico(f1);
f2=f1.*uint16(imagen_automatica2);
```

```
imagen_automatica3=umbral_automatico_inferior(f2);
f3=f2.*uint16(imagen_automatica3);

%comparacion
figure('Name', '3)algoritmo automatico con un filtro gaussiano')
subplot(2,2,1);
imshow(imagen_automatica,[])
title('primera iteración, imagen binaria')
subplot(2,2,2);
imshow(f1,[])
title('primera iteracion con distintos niveles de grís')
subplot(2,2,3);
imshow(f2,[])
title('segunda iteracion')
subplot(2,2,4);
imshow(f3,[])
title('tercera iteracion')


%% Utilizar el algoritmo SLIC
[L, NumLabels] = superpixels(imagen_ajustada, 90)
mask = boundarymask(L);

[L2, NumLabels2] = superpixels(imagen_ajustada, 91);
mask2 = boundarymask(L2);

[L3, NumLabels3] = superpixels(imagen_ajustada, 15);
mask3 = boundarymask(L3);

[L4, NumLabels4] = superpixels(imagen_ajustada, 500);
mask4 = boundarymask(L4);

%comparacion
figure('Name', 'Comparación de distintos N')
subplot(2,2,1);
imshow(imoverlay(imagen_ajustada,mask,'cyan'),'InitialMagnification',50)
title('Aplicado 90 k')
subplot(2,2,2);
imshow(imoverlay(imagen_ajustada,mask2,'green'),'InitialMagnification',50
)
title('Aplicado 91 de k')
subplot(2,2,3);
imshow(imoverlay(imagen_ajustada,mask3,'red'),'InitialMagnification',50)
title('Aplicado 15 de k')
subplot(2,2,4);
imshow(imoverlay(imagen_ajustada,mask4,'yellow'),'InitialMagnification',5
0)
title('Aplicado 500 de k')
```

```
SLIC1=SLIC(imagen_ajustada,90);
SLIC2=SLIC(imagen_ajustada,91);
SLIC3=SLIC(imagen_ajustada,15);
SLIC4=SLIC(imagen_ajustada,500);
%comparacion
figure('Name', 'Comparación al calcular el valor medio de cada
superpixel')
subplot(2,2,1);
imshow(SLIC1,'InitialMagnification',100)
title('Con 90 divisiones')
subplot(2,2,2);
imshow(SLIC2,'InitialMagnification',100)
title('Con 91 divisiones')
subplot(2,2,3);
imshow(SLIC3,'InitialMagnification',100)
title('Con 15 divisiones')
subplot(2,2,4);
imshow(SLIC4,'InitialMagnification',100)
title('Con 500 divisiones')



%% funciones
function imagen_manual = umbral_manual(imagen, valor_min, valor_max)
if nargin < 2 %si no dan valor_min se tomara como 100 caso im5
    valor_min = 120;
end
if nargin < 3%si no dan valor_max se tomara como 270 caso im5
    valor_max = 270;
end
imagen = double(imagen);
[filas, columnas]=size(imagen);
imagen_manual = zeros(filas,columnas);
for i=1:filas
    for j=1:columnas
        if imagen(i,j)> valor_min && imagen(i,j)< valor_max
            imagen_manual(i,j) = 1; %pone a 1 los pixeles en el rango
        else
            imagen_manual(i,j) = 0; %pone en negro lo demas
        end
    end
end
imagen_manual=int16(imagen_manual);
end
```

```
function imagen_automatica =
umbral_automatico(imagen,grado_de_aceptacion)
if nargin < 2 %si no dan un segundo valor se pone automatico
    grado_de_aceptacion = 0;5;
end
%Segmentación con umbral automático
imagen=double(imagen);
cnt=0;% Contador de Iteracioenes
entrada=imagen;
ent=entrada(entrada>0);
T = mean2(ent); % Calculamos el umbral inicial
condicion = true;
while condicion % bucle hasta que tenga un umbral decente
 cnt=cnt+1; %en caso de querer evitar bucles infinitos, podría usar el
cnt
 %para poner un numero maximo de repeticiones
 imagen_automatica = ent>T;
 medI = mean(ent(~imagen_automatica)); % Media de los inferiores al
umbral
 medS = mean(ent(imagen_automatica)); % Media de los superiores al umbral
 Tsig = 0.5*(medI+medS); % Umbral para la siguiente iteracion
 condicion=abs(T-Tsig)>grado_de_aceptacion;
 T=Tsig;
end
imagen_automatica = imbinarize(uint16(entrada),T/65535);
end

%probar en caso de que el objeto de estudio se va del umbral automatico
function imagen_automatica =
umbral_automatico_inferior(imagen,grado_de_aceptacion)
if nargin < 2 %si no dan un segundo valor se pone automatico
    grado_de_aceptacion = 0.5;
end
%Segmentación con umbral automático
imagen=double(imagen);
cnt=0;% Contador de Iteraciones
entrada=imagen;
ent=entrada(entrada>0);
T = mean2(ent); % Calculamos el umbral inicial
condicion = true;
while condicion % bucle hasta que tenga un umbral decente
 cnt=cnt+1; %en caso de querer evitar bucles infinitos, podría usar el
cnt
 %para poner un numero maximo de repeticiones
 imagen_automatica = ent<T; % Aquí cambiamos la lógica del umbral
 Tsig= mean(ent(~imagen_automatica)); % Media de los inferiores al umbral
  % Umbral para la siguiente iteracion
 condicion=abs(T-Tsig)>grado_de_aceptacion;
 T=Tsig;
```

```
end
imagen_automatica = imbinarize(uint16(entrada),T/65535);
end


function imagen_otsu = metodo_otsu(imagen)
imagen=double(imagen);
grises = mat2gray(imagen);
[level, ~]= graythresh(grises);

mejora = round(grises);
mejora2 = uint8(mejora);
imagen_otsu = imbinarize(mejora2, level/255);
end

function imagen_global = umbral_global_treshold(imagen)
%Segmentación con umbral automático
imagen=double(imagen);
cnt=0;% Contador de Iteracioenes
grises = mat2gray(imagen); %convierte la imagen en escala de grises que
% contiene valores en el intervalo 0 (negro) a 1 (blanco).
T=mean2(grises);
condicion = true;
while condicion % bucle hasta que tenga un umbral decente
 cnt=cnt+1; %en caso de querer evitar bucles infinitos, podría usar el
cnt
 %para poner un numero maximo de repeticiones
 imagen_automatica = grises>T;
 medI = mean(grises(imagen_automatica)); % Media de los inferiores al
umbral
 medS = mean(grises(~imagen_automatica)); % Media de los superiores al
umbral
 Tsig = 0.5*(medI+medS); % Umbral para la siguiente iteracion
 condicion=abs(T-Tsig)~=0;
 T=Tsig;
end
mejor = round(grises); % Redondeamos la Imagen
mejo3 = uint8(mejor); % La trasformamos a enteros de 8 bits
imagen_global = imbinarize(mejo3, T/255); % Binarizamos la imagen según
el threshhold
end

function  salida = SLIC(imagen_ajustada,N)
[L, NumLabels] = superpixels(imagen_ajustada, N);
%mejorando la imagen
salida=zeros(size(imagen_ajustada),'like',imagen_ajustada);
idx=label2idx(L);
%[filas, columnas]=size(imagen_ajustada);
for i=1:NumLabels
```

```
    redidx=idx{i};
    salida(redidx)=mean(imagen_ajustada(redidx));
end
end
```