PRACTICA 3:

Contenido

0. INTRODUCCIÓN: 1

1. UMBRALIZACIÓN: 1

1.1 Umbralización manual 2

1.2 Umbralización automática 3

1.3 Umbralización automática con filtrado previo 4

2. ALGORITMO DE SUPERPÍXEL 6

3. CÓDIGO.....10

4. BIBLIOGRAFÍA 14

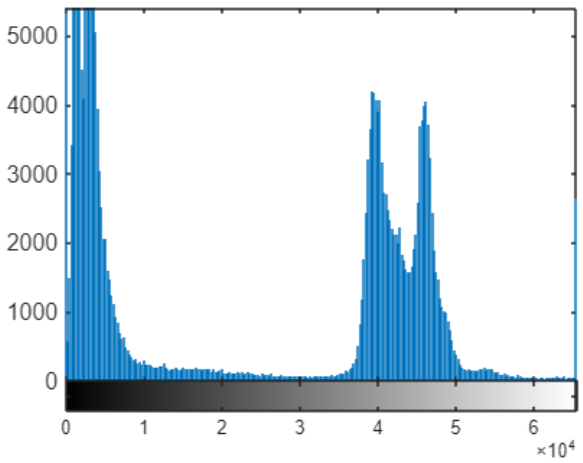
0. INTRODUCCIÓN:

La segmentación en imágenes biomédicas se refiere al proceso de identificar y separar regiones específicas.

Por lo tanto, a lo largo de esta práctica pretendo aprender y aplicar las técnicas de segmentación para limitar contornos de nuestro objeto de interés.

1. UMBRALIZACIÓN:

Antes de empezar con la Umbralización analizo el histograma para saber cuál es el pico que me interesa estudiar y, por ende, con que sección del umbral me voy a ir quedando. En este caso:



La primera división que se hará será con *T = media de los valores para dividir el histograma en dos partes iguales* y nos quedaremos con la parte derecha. Esto sigue un proceso iterativo hasta aislar el pico de interés el cual oscila entre N1 = 41000 y N2 = 48000. Uso este datos pues fueron conclusiones a las que llegué en prácticas anteriores.

1.1 Umbralización manual

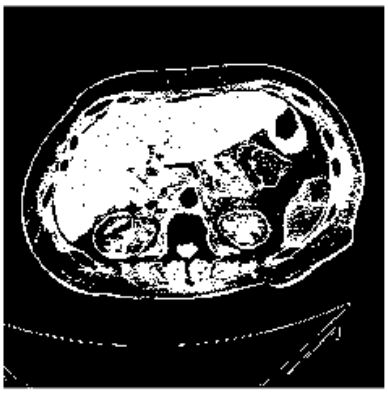
Seleccionamos manualmente dos valores N1 y N2 para definir un rango de intensidad que preservará en la imagen segmentada. Los pixeles que caen dentro de este rango serán conservados en la imagen segmentada. Con esta descripción implementamos este código para la umbralización:

```
N1 = uint16(41000);
N2 = uint16(48000);


[filas, columnas] = size(A);
imagenUmbralizada = zeros(filas, columnas, 'uint16');


for i = 1:filas
    for j = 1:columnas
        xij = A(i,j);
        if (xij > N1) && (xij < N2)
            yij = uint16(1);
        else
            yij = uint16(0);
        end
        imagenUmbralizada(i,j) = yij;
    end
end
imshow(imagenUmbralizada, []);
```

El resultado muestra la imagen umbralizada, donde se observa que la ventana de interés está más definida, pero hay pérdida de textura y suavidad. Se han generado una serie de defectos como puntos negros que indican una pérdida de información. Se muestra a continuación a la izquierda la imagen original y a la derecha la imagen umbralizada manualmente.



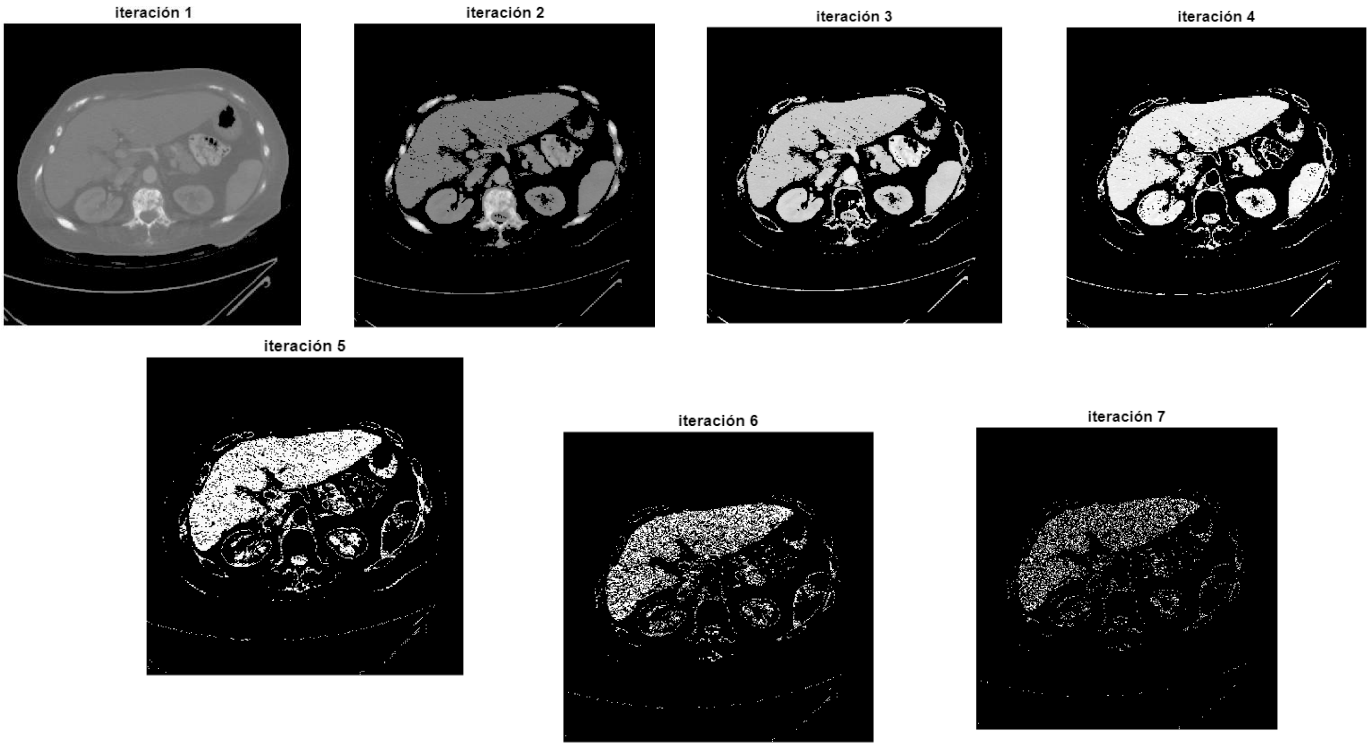
Tras la comparación se muestra en grande:

1.2 Umbralización automática

Tras el histograma mostrado en el apartado 1, vemos que tenemos varios picos. Esto implica que tendremos que iterar varias veces hasta quedarnos con el pico de interés. Para ello cree dos cálculos: **AUmbral** y **AUmbralinv**. Que se pueden en el apartado 6 de códigos.

La principal diferencia entre estas dos funciones reside en que la primera conserva aquellos valores que supera el umbral determinado y la segunda es la inversa por lo que se queda con aquellos valores que no superan dicho umbral.

Por lo tanto, fui ejecutando iteración por iteración para poder ir viendo cuando aplicar una función u otra. Finalmente recogí lo siguiente:

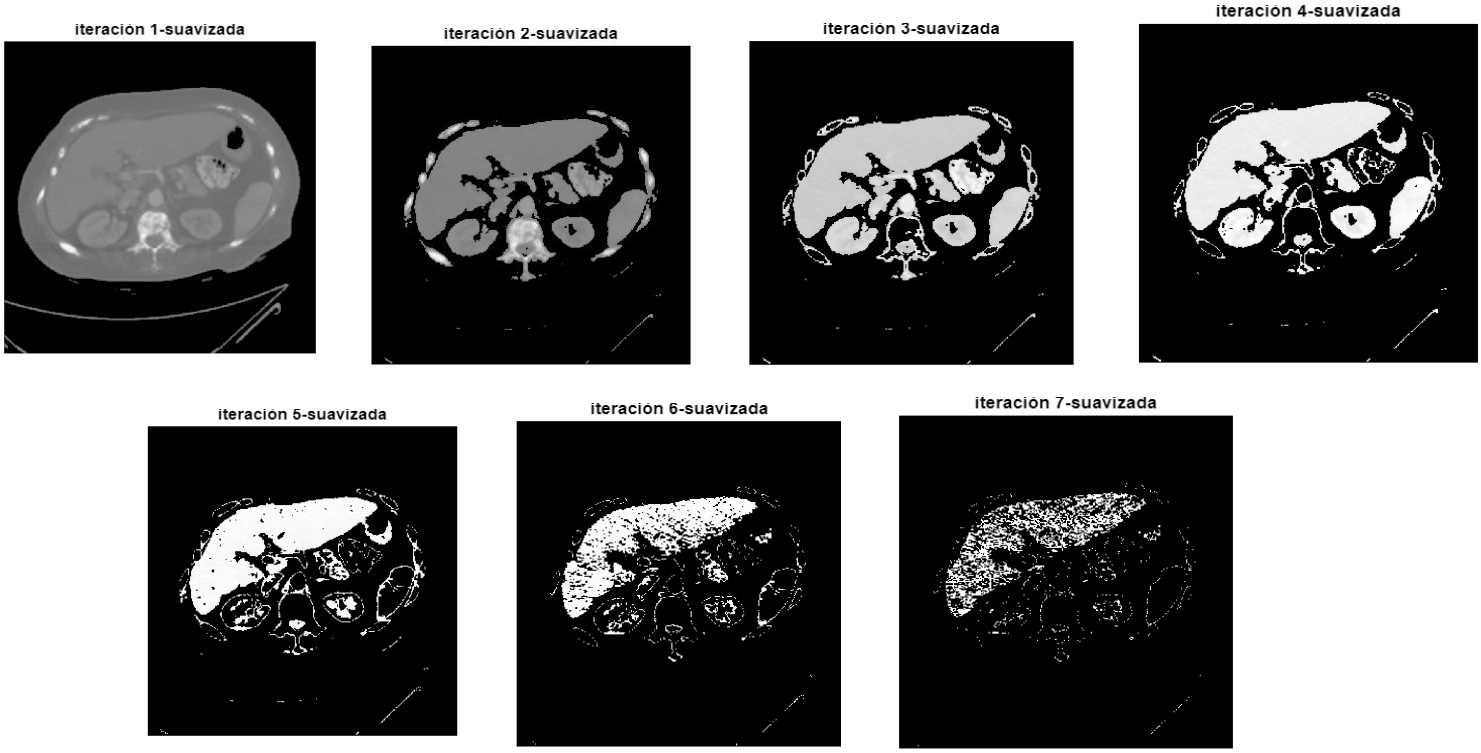


De todas las iteraciones, considero que a partir de la quinta se empieza a perder información de manera excesiva. Además, la cantidad de ruido es bastante elevada y por ello la perdida de información también. Esto lo podríamos arreglar aplicando algún filtro previo y suavizando la imagen. Esto lo vimos en la práctica anterior y llegamos a la conclusión de que usar un filtro gaussiano podría ser de bastante utilidad.

1.3 Umbralización automática con filtrado previo

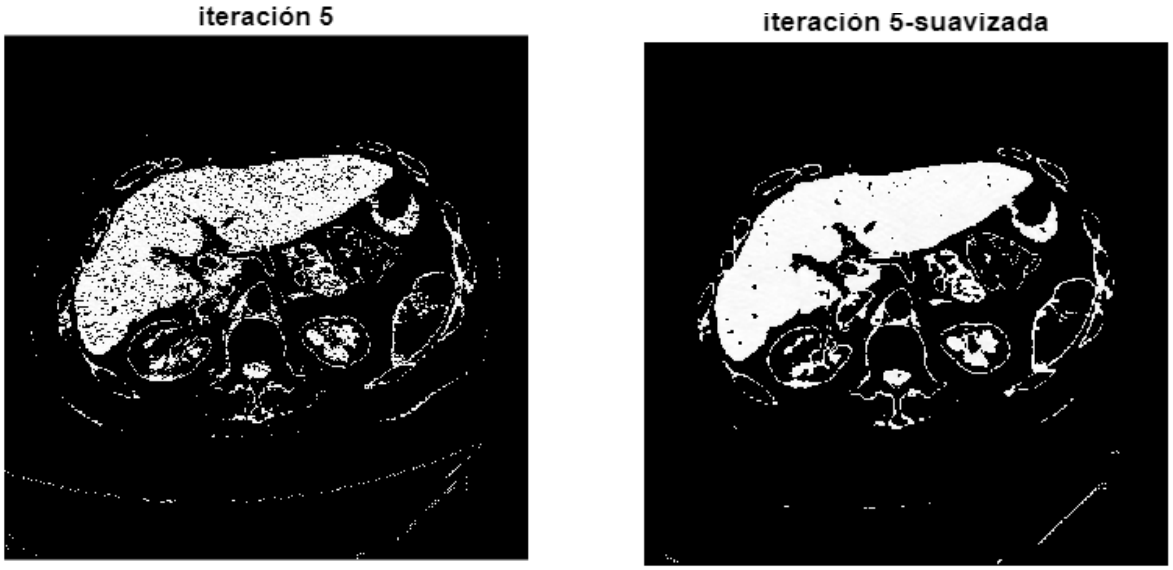
Una vez estudiado que ocurre con la umbralización automática vamos a intentar arreglar aquello que nos perturbaba anteriormente. La pérdida de información.

Para ello vamos a aplicar un filtro de paso bajo como por ejemplo el filtro gaussiano pues este nos aporta una suavidad y obtendríamos una ventana de interés más clara.



Efectivamente, aquí encontramos una ventana de interés más sólida y si tuviera que elegir una sería la de la iteración 5 pues a pesar de que tiene un poco de ruido tiene el hígado super diferenciado del resto.

Como conclusión de este primer apartado saco que, comparando ambas iteraciones 5 con ambas umbralizaciones con y sin filtro:



Ambas imágenes presentan una ventana de interés diferenciada del resto de componentes del TC. Sin embargo, al no aplicar ningún tipo de filtro conseguimos una imagen con mucha textura y por ende esta pérdida de información que nos preocupa. Como solución se propone aplicar un filtro gaussiano el cual nos reduce ese ruido y sigue marcando bastante bien el hígado.

2. ALGORITMO DE SUPERPÍXEL

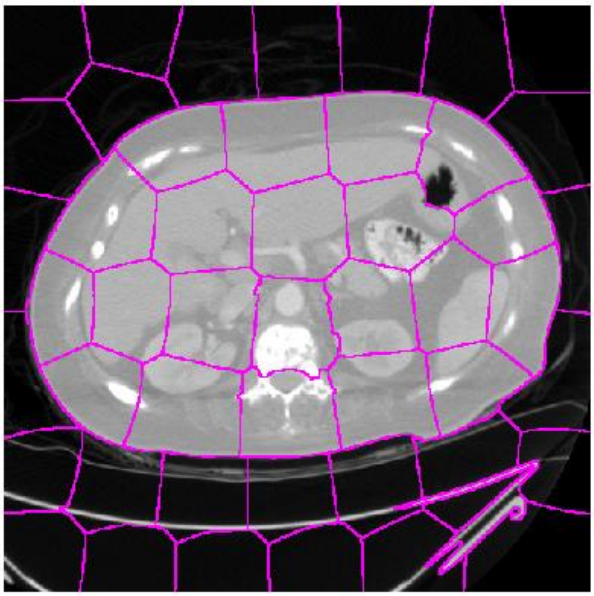
La complejidad aquí reside en la cantidad de píxeles que poner en el superpíxel.

Primero me gustaría comentar qué es esto y que me proporciona en la práctica.

La función superpíxel divide una imagen en regiones homogéneas que tienen en común intensidades muy similares. Estas regiones se les llama superpíxeles.

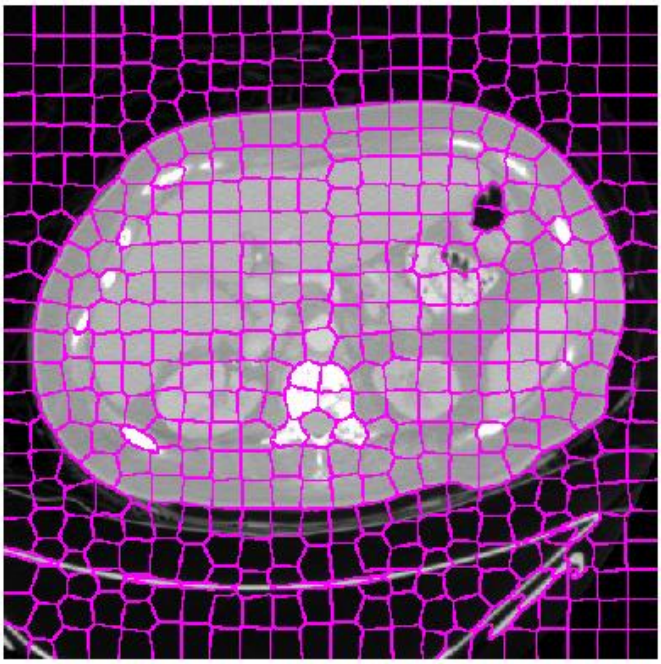
No queda más que probar con distintos valores de N.

N=50 superpíxeles:



Con un valor de N=50 superpíxeles vemos que se generan polígonos bastantes grandes que no capturan mucho detalle de la imagen médica. Probaremos con un N=500.

N=500 superpíxeles:

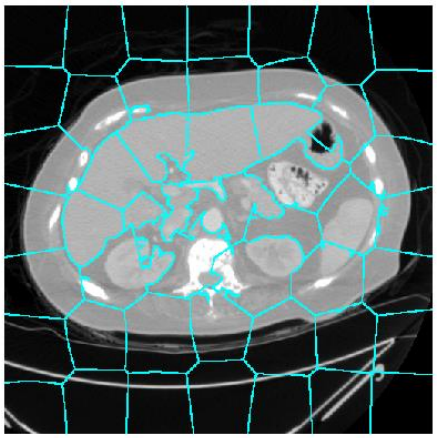


Vemos que conforme aumenta el valor de N, aumenta la cantidad de regiones. Es por esto por lo que parece que se ajusta mejor a la ventana de interés. Sin embargo, debido a que la imagen utilizada no presentaba un hígado que destacase en intensidad no conseguimos ajustar un valor preciso de N tal que el hígado quede dividido en regiones diferenciadas del resto.

Es por eso que vamos a aplicar la técnica vista en el apartado anterior de Umbralización para ver si así la función superpíxel pilla mejor la zona de interés.

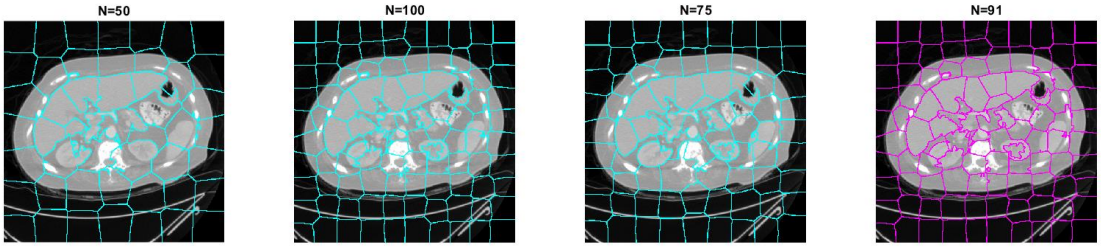
SUPERPIXEL CON N=50 EN IMAGEN UMBRALIZADA.

La imagen que he decidido utilizar es la de la iteración 5 suavizada con el filtro gaussiano.

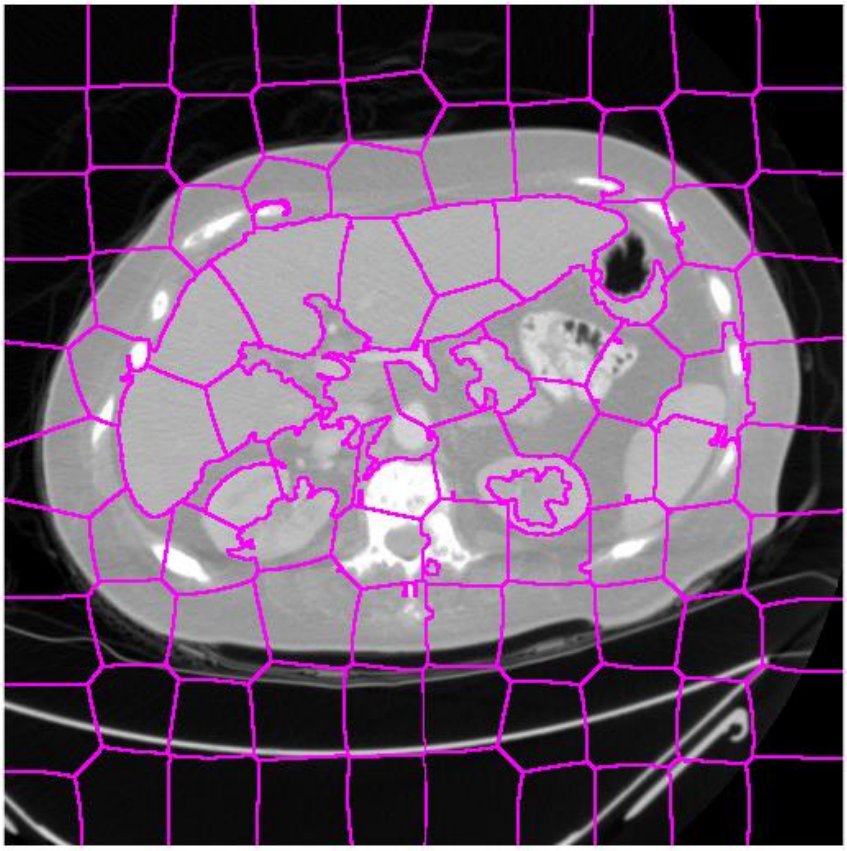


Lo que está ocurriendo aquí es que al principio nuestro hígado no conseguía diferenciarse del resto de elementos del TC por lo tanto le resulta más difícil al algoritmo capturar lo que queríamos. Por ello aplicamos una Umbralización, la cual nos marca la ventana de interés. Al hacer esto el hígado ya sí que se destaca de los elementos adyacentes provocando que la región se defina mejor.

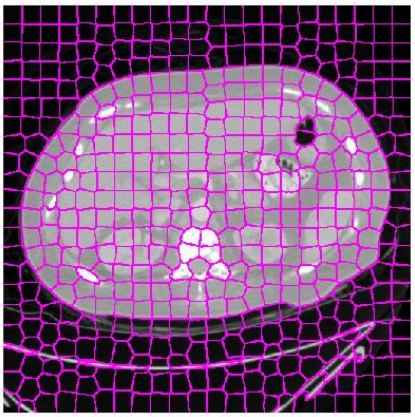
Por lo tanto, tras poner un número N de superpíxel aleatorio comienzo a hacer unas aproximaciones. Lo primero que hice fue probar con un valor bastante grande como N=100. Vemos que el hígado queda bastante bien dividido por las regiones. Por lo tanto, probé con un valor intermedio como podría ser N=75 viendo que aún no conseguía cerrar bien el hígado fui probando poco a poco hasta llegar a un valor mínimo de N=91. A continuación muestro las tres aproximaciones en azul:



Por lo tanto, en mi caso el valor mínimo de superpíxel para encerrar el hígado en regiones es de n=91. Lo muestro a continuación con mayor detalle:



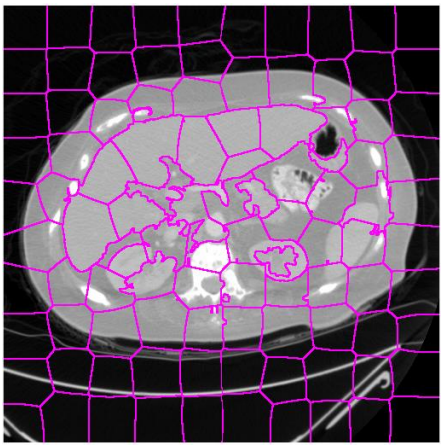
Como conclusión de este apartado me quedo con las siguientes imágenes:



En esta primera imagen usamos la imagen original con un superpíxel de N=500. Vemos que esta N no es suficiente para hallar las regiones a pesar de que es una gran cantidad.

Es por ello que se planteó usar la umbralización anterior pues en pocas palabras esta hace “brillar” nuestro hígado y gracias a esto el algoritmo de superpíxel será capaz de identificar mejor la zona de interés.

Gracias a esto conseguimos la siguiente imagen y ya de paso hicimos un estudio iterativo de cuál sería el mejor umbral mediante ensayo y error que está anteriormente descrito. Finalmente conseguí minimizarlo a N=91.



3. CÓDIGO

```
archivo = "C:\Users\34603\OneDrive\Documentos\MATLAB\IMAGEN
BIOMEDICA\img\PR3\im2";
imagen = dicomread(archivo);
A = imadjust(imagen);


imhist(imagen,(2^(16))-1)
axis([0 2000  0 2000])
imhist(A)
imshow(A, [41000 48000]);


imagenUmbralizada = uint16(imagen) .* imagenUmbralizada;

% Muestra la imagen umbralizada



imshow(imagenUmbralizada, []);

ite1 = AUmbral(imagen);title("iteración 1");

ite2=AUmbral(ite1);title("iteración 2");

ite3=AUmbral(ite2);title("iteración 3, cambio de lado en la umbral.");

ite3inv = AUmbralinv(ite2);title("iteración 3");

ite4inv=AUmbralinv(ite3inv);title("iteración 4");

ite5inv = AUmbralinv(ite4inv);title("iteración 5");

ite6inv = AUmbralinv(ite5inv);title("iteración 6,cambio lado de umbra.");

ite6 = AUmbral(ite5inv);title("iteración 6");

ite7 = AUmbral(ite6);title("iteración 7");

ite7inv= AUmbralinv(ite6);title("iteración 7");

hGauss = fspecial("gaussian", [5 5], 1);% jugar cin ek 99 o 11 a mayor
numero mayor suavizado en el interior y prot el contorno

filtro_gauss = imfilter (imagen, hGauss);

ig1 = AUmbral(filtro_gauss);title("iteración 1-suavizada");

ig2 = AUmbral(ig1);title("iteración 2-suavizada");

ig3 = AUmbralinv(ig2);title("iteración 3-suavizada");

ig4 = AUmbralinv(ig3);title("iteración 4-suavizada");
```

```
ig5 = AUmbralinv(ig4);title("iteración 5-suavizada");

ig6 = AUmbral(ig5);title("iteración 6-suavizada");

ig7 = AUmbralinv(ig6);title("iteración 7-suavizada");

super = imadjust(imagen); %mejoro el contraste;

[X, Y] = superpixels(imagen,50);

BW = boundarymask(X);

imshow(imoverlay(super , BW, "m"))

[X, Y] = superpixels(imagen,500);

BW = boundarymask(X);

imshow(imoverlay(super , BW, "m"))

[X, Y] = superpixels(ig5,50);

BW = boundarymask(X);

imshow(imoverlay(super , BW, "c"))

% hemos probado:

% superpixel con imagen -> regu no consigo dividir en regiones exactas


%superpixel con umbralizacion -> muy buenos resultados


[X, Y] = superpixels(ig5,2);

BW = boundarymask(X);

imshow(imoverlay(super , BW, "c"))

function [output] = AUmbral(input)

    % Preprocesamiento: Eliminar valores no positivos

    A1 = input(input > 0);


    % Inicialización

    m = mean(A1);

    fin = false;
```

```
    % Iteración para encontrar el umbral óptimo
    while ~fin
        outputLogical = A1 > m; % Binarizar según el umbral actual
        m2 = 0.5 * (mean(A1(outputLogical)) + mean(A1(~outputLogical))); 
% Nuevo umbral
        fin = abs(m - m2) < 0.5; % Condición de parada
        m = m2; % Actualizar el umbral
    end

    % Convertir la imagen original a uint16 y binarizar
    binaryMask = uint16(input > m); % Crear una máscara binaria

    % Aplicar la máscara binaria a la imagen original
    output = input .* binaryMask; % Aplicar el umbral a la imagen 
original

    % Visualización de la imagen
    figure
    imshow(output, []); % Mostrar con escala completa
     disp(m);
end



function [output] = AUmbralinv(input)
    % Preprocesamiento: Eliminar valores no positivos
    A1 = input(input > 0);

    % Inicialización
```

```
    m = mean(A1);

    fin = false;



    % Iteración para encontrar el umbral óptimo

    while ~fin

        outputLogical = A1 <= m; % Cambiar a "menor o igual" para la
binarización

        m2 = 0.5 * (mean(A1(outputLogical)) + mean(A1(~outputLogical)));
% Nuevo umbral

        fin = abs(m - m2) < 0.5; % Condición de parada

        m = m2; % Actualizar el umbral

    end



    % Crear una máscara binaria donde los valores son menores o iguales a
m

    binaryMask = uint16(input <= m); % Crear una máscara binaria



    % Aplicar la máscara binaria a la imagen original

    output = input .* binaryMask; % Aplicar el umbral a la imagen
original



    % Visualización de la imagen

    figure;

    imshow(output, []); % Mostrar con escala completa

    disp(m);

end
```

4. BIBLIOGRAFÍA