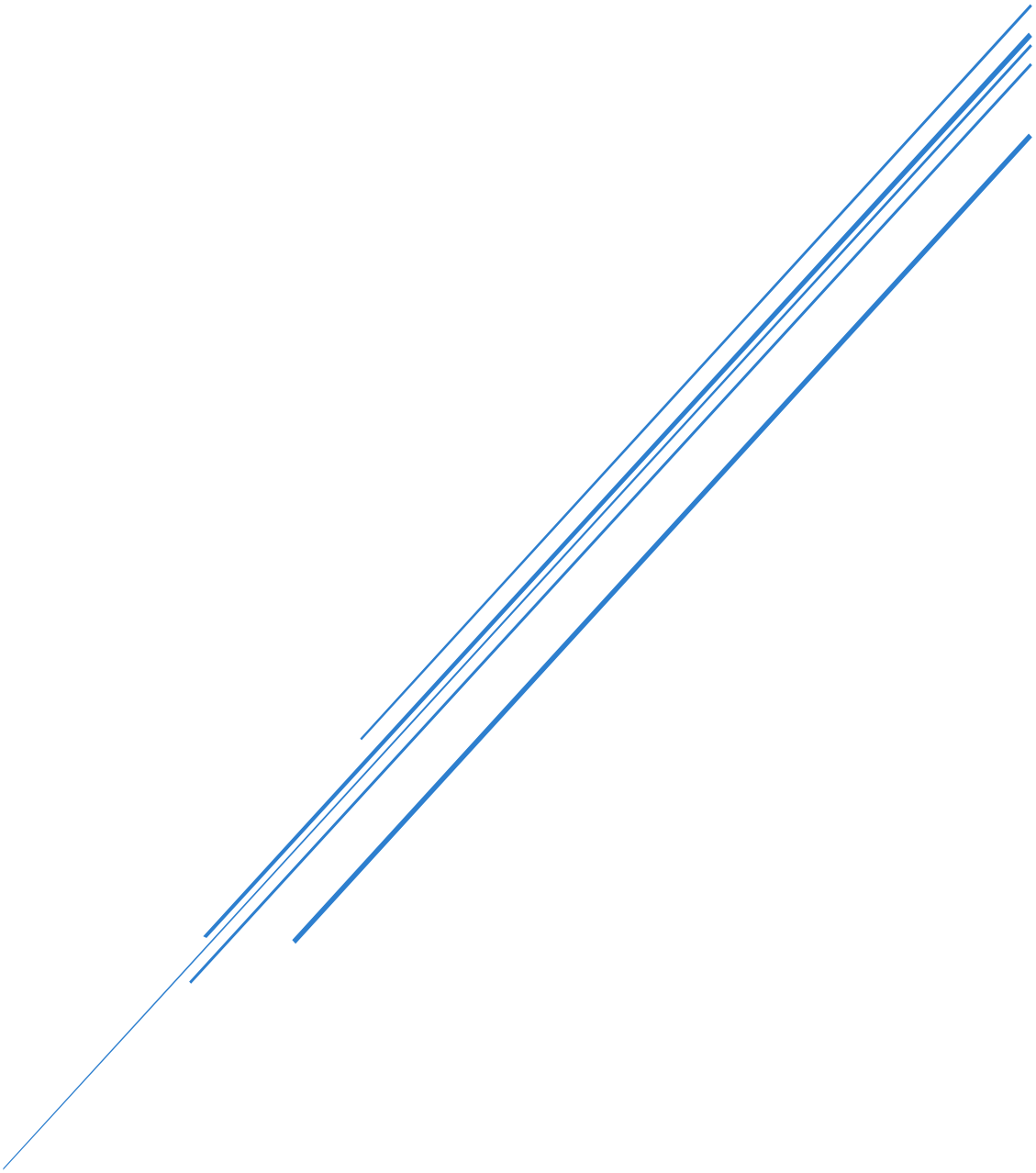
PRÁCTICA 1: TÉCNICAS BÁSICAS

Manejo de ficheros DICOM, histograma y modificación de contraste.



E.T.S.I - Universidad de Málaga
Sara Giménez Gómez

Contenido

1. INTRODUCCIÓN

En esta práctica, trabajaremos con imágenes médicas en el formato estándar DICOM (*Digital Imaging and Communication in Medicine*), un formato ampliamente utilizado en hospitales y centros médicos para almacenar imágenes de diagnóstico. Utilizaremos la *toolbox de Image Processing* de MATLAB, que ofrece herramientas para manipular y procesar imágenes. El objetivo es aplicar técnicas de procesamiento de imágenes para visualizarlas correctamente y extraer información relevante.

2. LECTURA DE LA IMAGEN Y DE LA INFORMACIÓN

Primero, procederemos a cargar la imagen correspondiente y a extraer los metadatos del archivo DICOM utilizando el siguiente código:

```
archivo = 'C:\Users\34603\OneDrive\Documentos\MATLAB\IMAGEN
BIOMEDICA\img\im2';
imagen = dicomread(archivo)
inf = dicominfo(archivo)
```

Al ejecutar este código, obtendremos una gran cantidad de datos. Sin embargo, solo nos interesa resaltar la siguiente información:

```
Número de filas: 512 Número de columnas: 512
Descripción de imagen: ONCOLOGIA
Tipo de imagen: ORIGINAL\PRIMARY\AXIAL\HELIX
Esta imagen es una CT realizada en H.CLINICO MALAGA
Número de bits por píxel: 16
Formato de la imagen: DICOM
```

En resumen, en esta práctica trabajaremos con una matriz de tamaño 512x512 y en formato uint16. Esto significa que cada píxel está representado mediante 16 bits, y cada celda puede oscilar entre 0 y 65535.

```
figure
imshow(imagen);
title('Imagen número 02');
```

Así que con el comando de *dicomread ()* obtenemos la siguiente imagen:

La imagen inicialmente es oscura. El motivo de esto lo hallamos utilizando el comando *dicominfo()* el cual nos devuelve una matriz con los siguientes valores:

```
imagen = 512×512 uint16 matrix
     43   32   30   34   34   25    9   19   36   39   33   25   33   42   42   35   33 ...
     46   45   35   33   36   37   26   13   20   31   34   29   30   33   41   33   32
     34   41   43   39   38   34   31   29   19   20   23   24   34   35   41   40   32
     27   32   38   34   41   43   33   32   26   24   27   15   23   31   37   45   34
     21   29   36   33   32   38   44   46   44   29   24   20   19   18   23   34   40
     16   25   35   36   30   27   39   49   51   40   29   30   32   21   12   16   29
     21   19   28   31   30   28   28   31   44   53   48   37   32   31   25   17   17
     28   20   23   29   30   33   31   23   25   39   54   52   37   32   32   26   20
     41   31   25   23   24   28   36   33   23   20   35   52   49   39   27   25   29
     35   42   33   21   23   27   27   33   29   26   32   33   34   43   42   32   24
      ⋮
```

Estos son los datos que estamos mapeando cuando usamos el comando *imshow()*. Estos oscilan en un rango de0 a 65535, esto lo sabemos porque la imagen era uint16. Dado que los valores de la matriz están más cerca del límite inferior que del superior, la escala de grises tiende a ser oscura.

```
max(imagen(:))
ans = uint16
2192
```

Usando el código anterior, podemos visualizar que el valor máximo en la matriz es aproximadamente 2200, lo que significa que estamos almacenando valores relativamente pequeños dentro de un rango amplio, lo que provoca que la imagen aparezca predominantemente negra, con poca diferenciación entre las partes blancas.

Vamos a ajustar el rango al nuevo mencionado para poder ver que esconde tanta oscuridad:

```
figure
imshow(imagen,[0 2200]);
```

```
title('Imagen número 02, ajustada a los ejes adecuados');
```

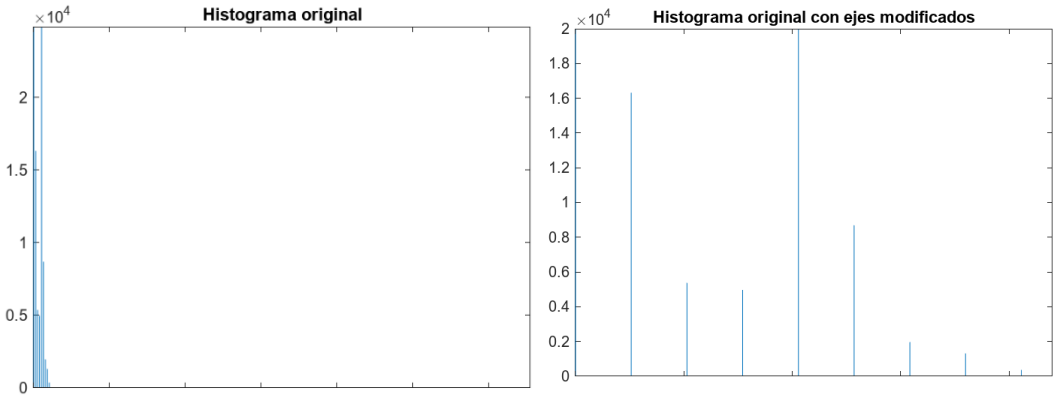
3. REPRESENTACIÓN E INTERPRETACIÓN DEL HISTOGRAMA

Se nos pide realizar el cálculo del histograma.

3.1 USANDO IMHIST ()

Para analizar la intensidad de los píxeles, calcularemos el histograma utilizando el método que MATLAB proporciona a través de imhist ():

```
figure
imhist(imagen)
title('Histograma original')

figure
imhist(imagen)
axis([0 2200 0 20000]);
title('Histograma original con ejes modificados')
```



Aquí vemos representado de manera discreta lo que se comentaba en el apartado anterior. En este histograma vemos que la mayoría se encuentran en el lado izquierdo de la gráfica. Esto significa que la imagen que visualizamos tiene poca intensidad y por ende muchas zonas oscuras.

Podríamos ajustar los ejes del histograma para poder ver áreas de interés en la imagen original, pero nos centraremos en esto más adelante.

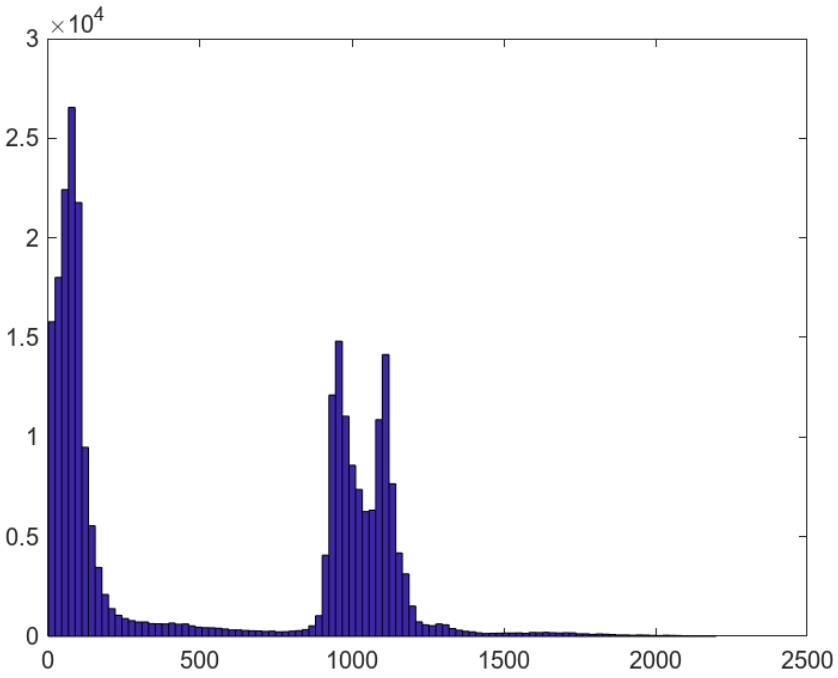
3.2 USANDO UN ALGORITMO ALGORTIMO

Crear un algoritmo personalizado para visualizar un histograma implica la capacidad de elegir cuántos intervalos queremos almacenar los datos. A continuación, presento un ejemplo de cómo se puede implementar este algoritmo:

```
function histograma = histogramacasero(array,num_itervalos,maximo_array)


grosor_intervalo = maximo_array/num_itervalos;
histograma=zeros(1,num_itervalos);


[filas,columnas] = size(array);
for i=1:filas
    for j=1:columnas

        intensidad = array(i, j);
        % calculo donde guardo el valor, en que intervalo
        bin_index = floor(intensidad / grosor_intervalo) + 1;


        % si el intervalo aproximado es mayor que el número de intervalos
        % hemos llegado al final
        if bin_index > num_itervalos
            bin_index = num_itervalos;
        end
        % si no entra al bucle es porque no hemos llegado al final

        % Incrementar el contador del bin correspondiente entonces
        % aumento el valor del contador del intervalo/bin que toque
        histograma(bin_index) = histograma(bin_index) + 1;


    end
end
bin_edges = (0:num_itervalos-1) * grosor_intervalo;% limite intervalos
figure;
bar(bin_edges, histograma, 'histc');
end
```

Con esta implementación tratamos de ajustar nosotros mismos el número de intervalos que queremos y ya de paso ajustar la escala de los ejes para tener la intensidad lo más repartida posible. Evitando así zonas muy oscuras o claras.

```
histogramacasero(imagen,100,2200)
```

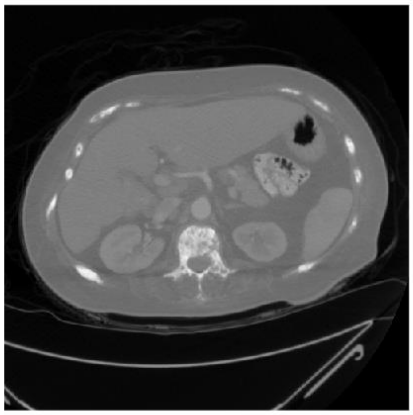


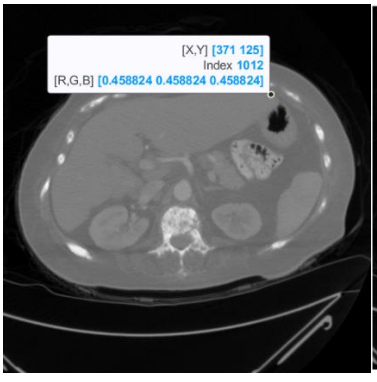
4. VISUALIZACIÓN DE ÁREA DE INTERÉS CON LA SELECCIÓN DE LA VENTANA IDÓNEA

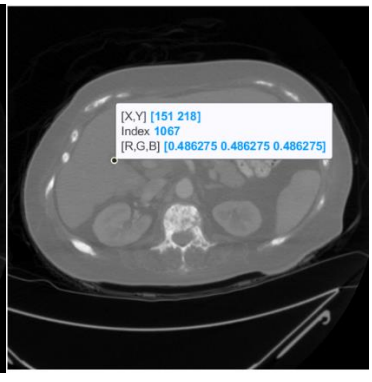
La imagen que estamos analizando corresponde a un corte axial del abdomen, donde podemos observar el hígado. Después de identificar el área de interés, utilizaremos diferentes métodos de ajuste de contraste para mejorar la visualización.

Primero vamos a intentar visualizar en que rango oscila nuestra área de interés:

(uso la herramienta de MATLAB llamada Data Tips)







Viendo el **index**,, más o menos podemos sacar un rango que nos permita visualizar el hígado. Si aun así no somos capaces de elegir un rango podemos ver el histograma anterior.
Analizándolo podemos identificar tres picos que representan el rango de intensidades de la imagen ajustada.
Según dónde se localicen estos picos podremos diferenciar los distintos tipos de tejidos según sus diferentes densidades. Un valor muy bajo en el histograma como el primer pico que visualizamos implica zonas muy oscuras como el aire. En mi caso concreto nos interesa ver el hígado, un tejido blando que se localizaría en valores intermedios como esos dos picos que se pueden ver en un rango de [1000 1300] Tras el uso de la herramienta mencionada anteriormente vemos que en verdad el hígado es el segundo pico, aquel que oscila entre [1000 1100] Así que ese será el rango que utilizaremos para ver nuestra ventana:

```
imshow(imagen,[1000 1100]);
```

Teniendo en cuenta que la zona en la que se encuentra el hígado oscila la mayoría de órganos entre los mismos rangos de valores por ser tejido blando no conseguiremos aislar totalmente el hígado en la visualización pero aún así considero que hemos conseguido ver el hígado bastante bien, incluso con un poco de textura.

Aún así también podemos ajustar un rango un poco más grande en la visualización de tal forma que tome los picos de interés y así no ver la imagen tan blanco o ngro como la anterior y que halla ciertos grises de intensidad:

```
figure
imshow(imagen,[800 1500]);
```

5.

MODIFICACIÓN DEL CONTRASTE

Para mejorar los detalles y facilitar la diferenciación de las distintas partes de la imagen, recurrimos a la modificación del contraste. Esto es crucial para un diagnóstico efectivo.

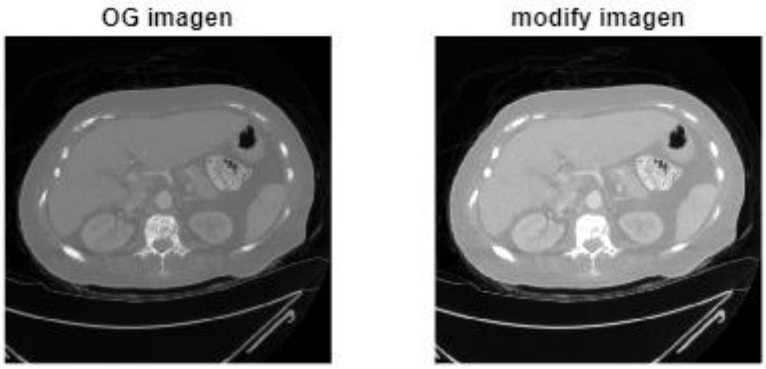
5.1 USANDO IMADJUST ()

La función imadjust() normaliza la matriz, ajustando los valores para facilitar su visualización. Esta función realiza una transformación lineal. La sintaxis es la siguiente:

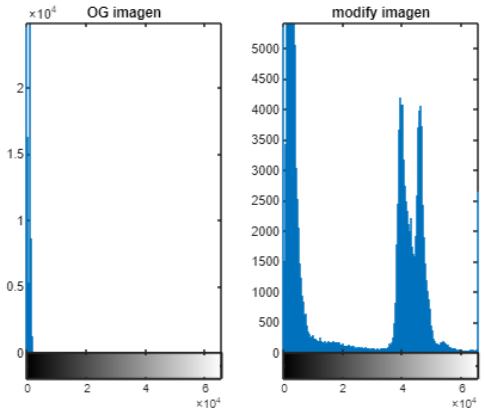
J = imadjust (Imagen, [LOW_IN HIGH_IN], [LOW_OUT HIGH_OUT], gamma)

- LOW_IN y HIGH_IN: Límites inferior y superior de los valores de intensidad de la imagen original que se desean ajustar.
- LOW_OUT y HIGH_OUT: Límites inferior y superior de los valores de intensidad de la imagen resultante.
- Gamma: Parámetro que describe la relación no lineal entre los valores de intensidad de la imagen original y los de la imagen resultante.

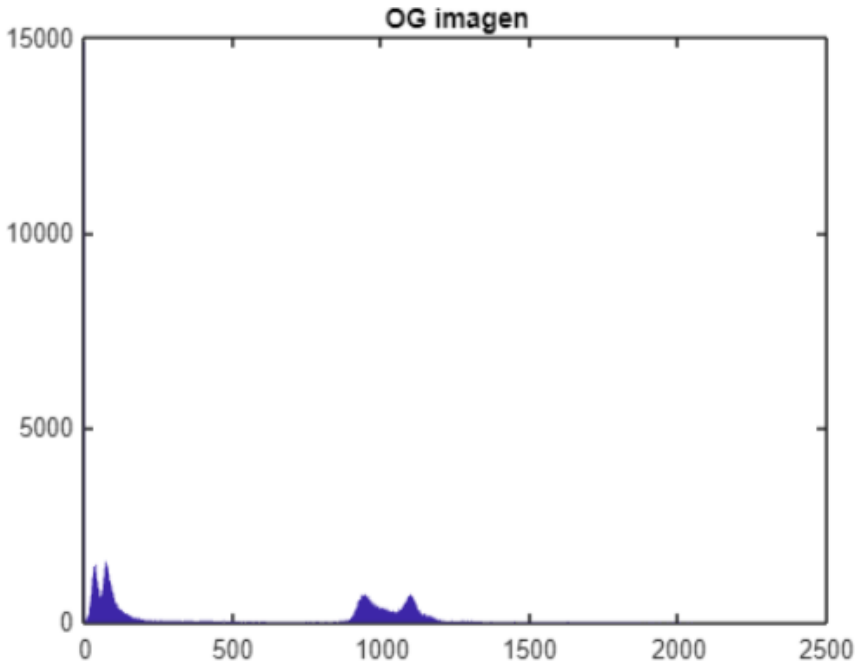
Un imadjust() de ajuste automático se ve de la siguiente forma:



A continuación muestro los histogramas de la imagen usando imhist() . Aquí vemos que el segundo histograma refleja lo estudiado en apartados anteriores.



Enseño también a continuación como se vería el histograma creado para esta segunda imagen ajustada automáticamente sin rango por ahora de visualización:



Como ya comenté anteriormente, con este histograma creado simplemente lo que buscábamos era poder elegir cuantos elementos queremos guardar en cada

intervalo y además ajustar los ejes de visualización para poder ver todos los datos adecuadamente.

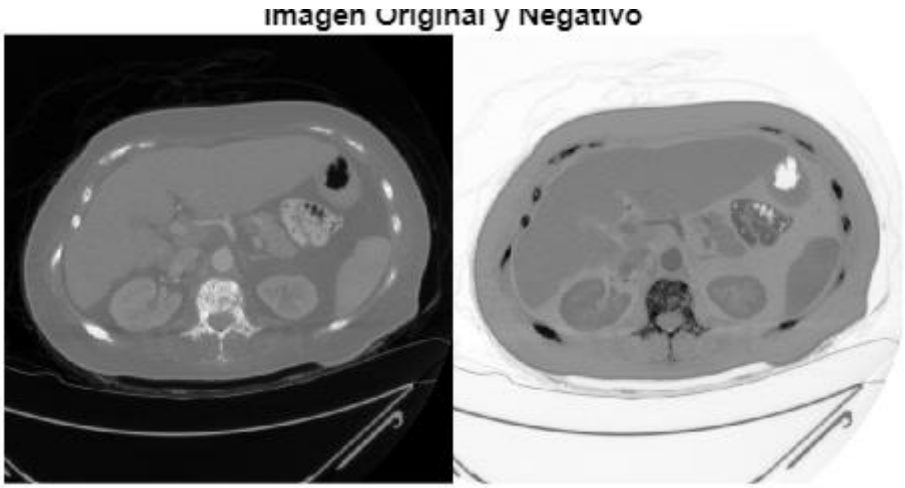
Si nos centramos en los rangos descritos anteriormente obtenemos una imagen ajustada automáticamente y con rangos definidos para la correcra visualización de la ventana de interés:

5.1.1 NEGATIVO DE LA IMAGEN

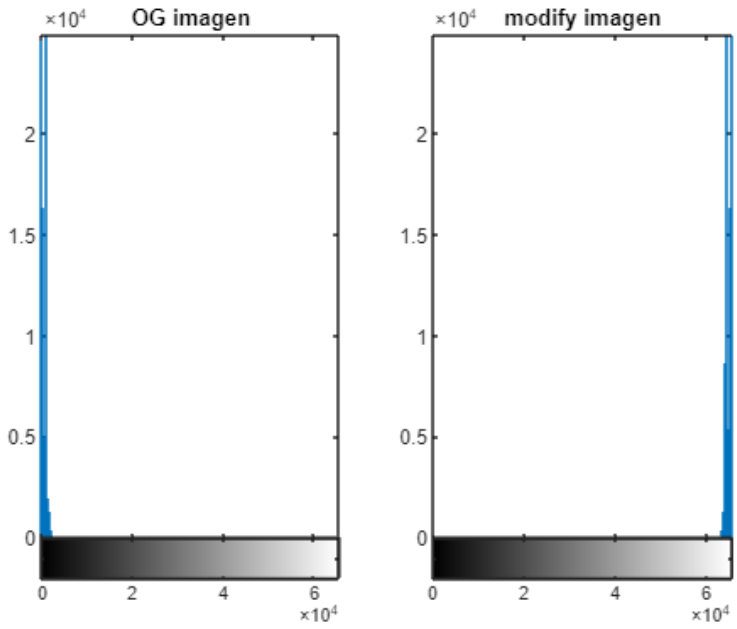
Los códigos se encuentran al final de la práctica, por intentar hacerla lo menos pesada posible procederé a omitir los que no sean estrictamente necesarios.

Ajustamos la imagen para que sea negativo, para ello invertimos los rangos de salida y entrada de la imagen.

```
negativo = imadjust(imagen, [0 1], [1 0]);
```

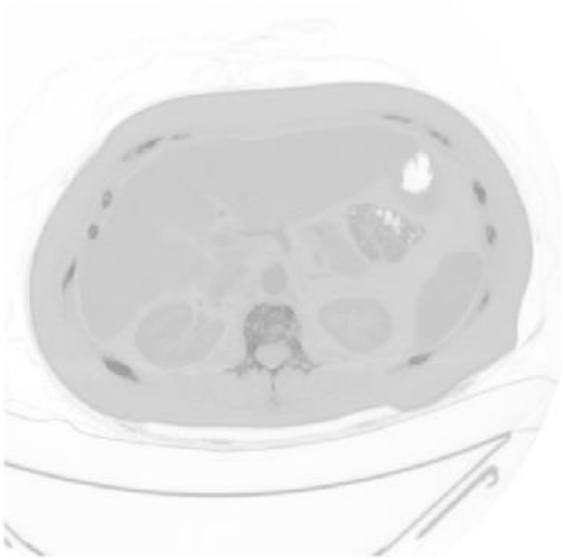


Los histogramas usando imhist () nos proporciona una información exacta de lo que está ocurriendo. Se invirtieron los valores tal y como queríamos consiguiendo así el negativo.



Vamos a ajustar la imagen al rango adecuado para poder ver información :

```
imshow(negativo,[60000 65536]);
```

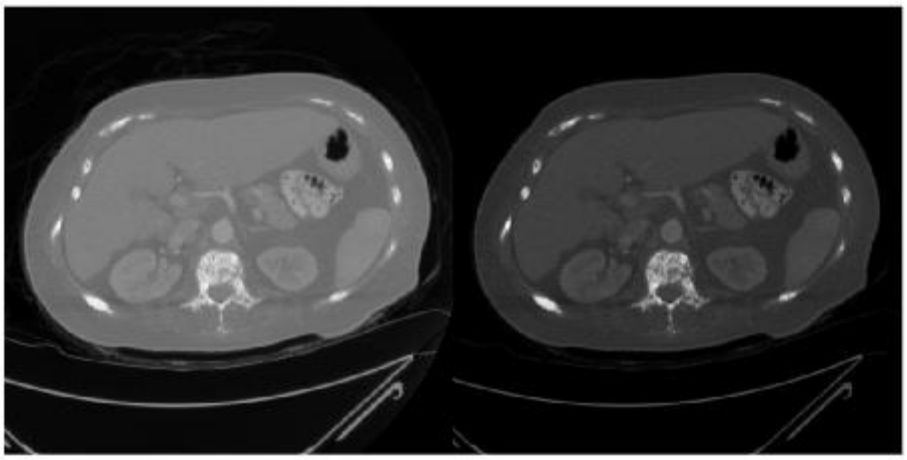


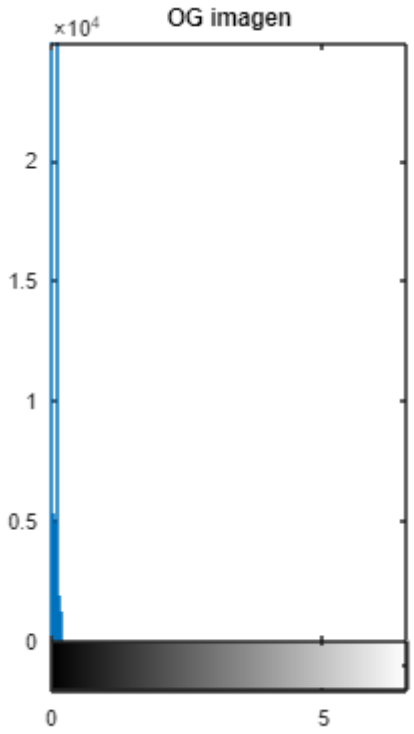
Esta imagen se ve muy clara perdiendo todo tipo de información acerca del objetivo de la práctica. Por lo tanto no creo que este método sea el mejor, pero aún así luego los compararemos.

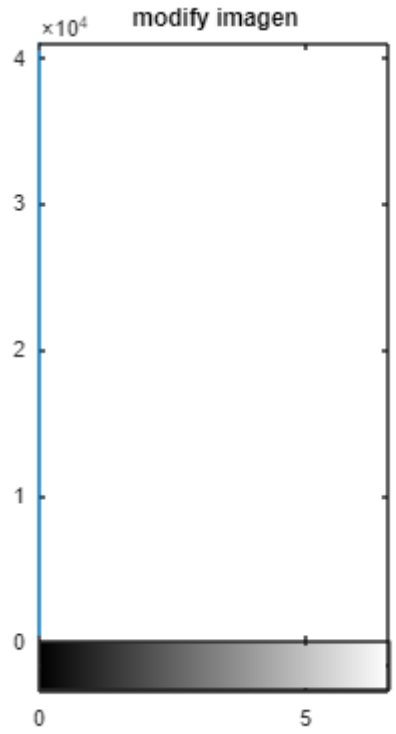
5.1.2 CORECCIÓN GAMMA

La corrección gamma es una técnica no lineal. Al aplicar un gamma de 2, la asignación se moverá hacia valores de salida más oscuros:

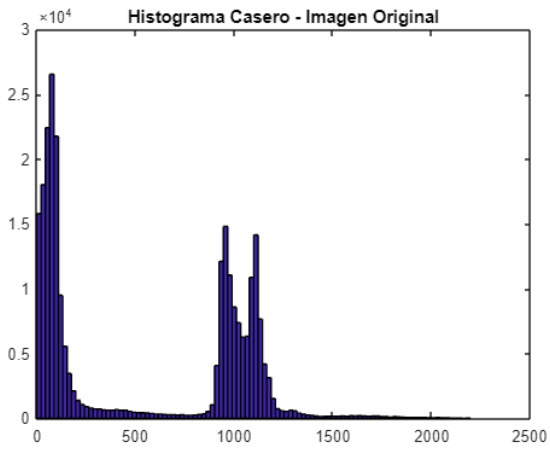
```
gammaimg = imadjust(imagen, [], [], 2);
```

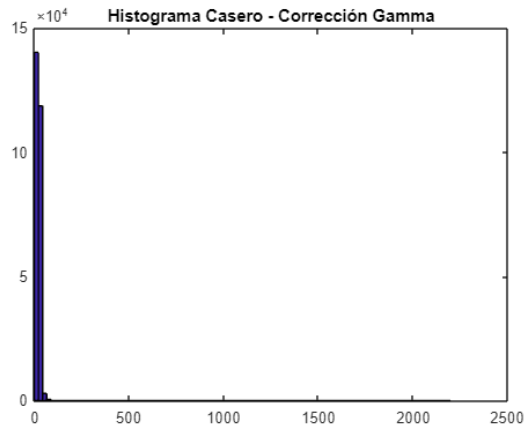






Aquí opté por utilizar también el histograma creado por el algoritmo ya mencionado, consideré que proporcionaba más información:





Si abro la imagen en el rango de [0 25] obtendríamos la siguiente imagen para el ajuste con gamma=2 :

Tal y como se mencionó anteriormente la imagen se oscurece aún más.

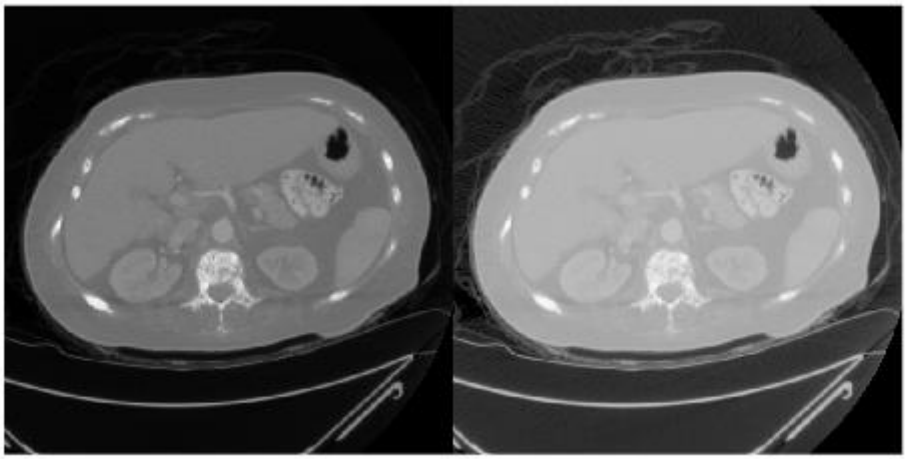
```
imshow(gammaimg,[0 25]);
```

La verdad que aquí si que se ve bien el hígado, tenemos una clara visualización de este y gran contraste entre los demás tejidos blandos y óseos de la imagen.

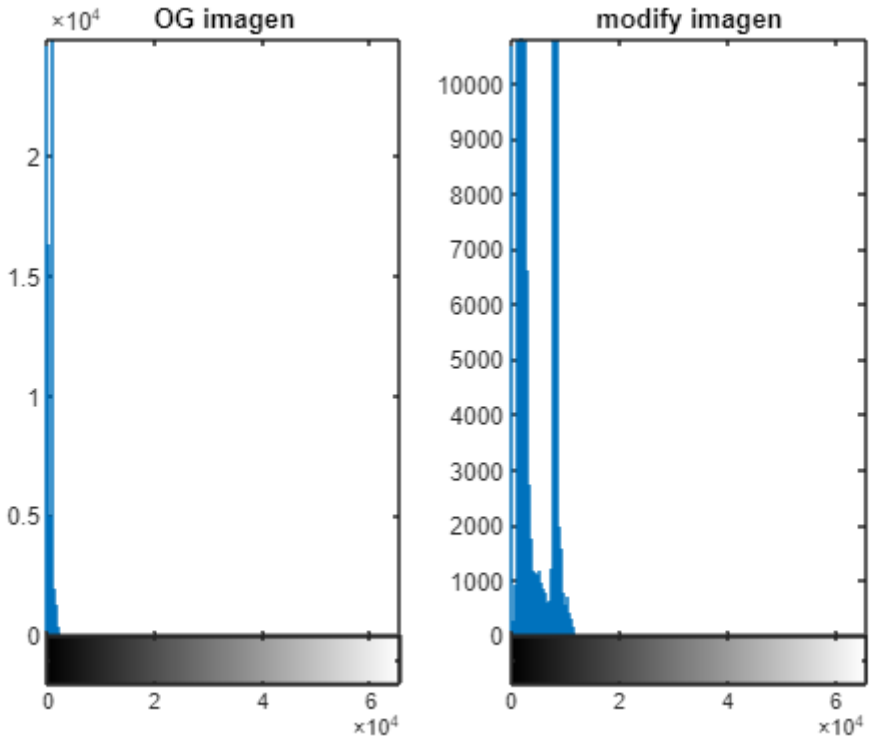
5.1.3 AJUSTE RAÍZ CUADRADA

Aunque se nos pide un ajuste de raíz cuadrada, esto es lo mismo que decir que usemos un valor para gamma =1/2. Es decir si al aumentar gamma obteníamos imágenes más oscuras aquí se espera aclarar más la imagen original:

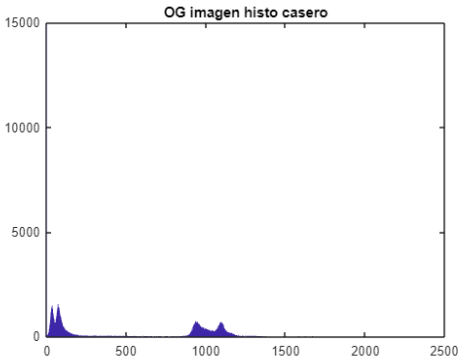
```
sqrt_img = imadjust(imagen,[],[],0.5);
```

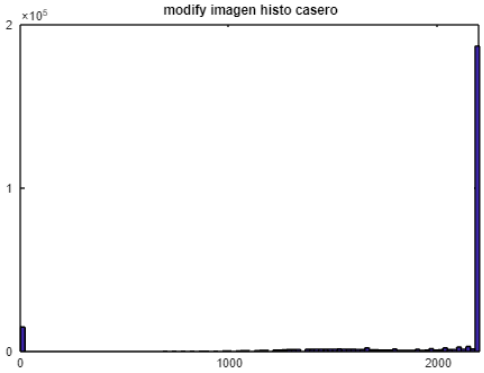


Los histogramas muestran lo dicho anteriormente. Vemos como el histograma de la derecha se desplazó bastante hacia la derecha, provocando este aumento de bits con tonalidades más claras.



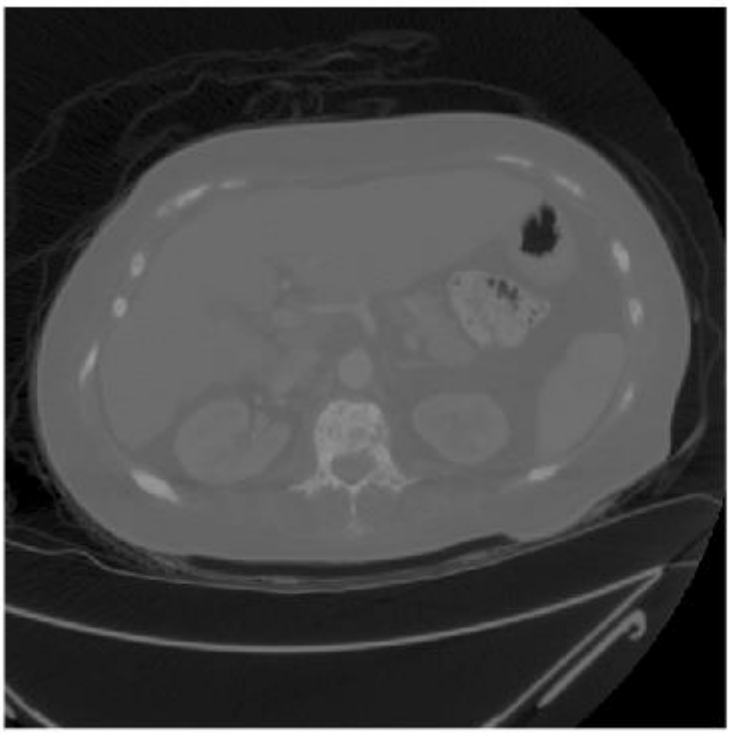
Usé el algoritmo creado para que se vea que si que fue grande el desplazamiento del histograma hacia la derecha pues con los ejes de los histogramas superiores quizás no se aprecia:





Si ajustamos el rango de la imagen con el ajuste de la raíz cuadrada para mejorar un poco la visualización obtenemos lo siguiente:

```
imshow(sqrt_img,[0 20000]);
```



Aquí vemos que la imagen con el rango marcado se oscurece demasiado. Esto es debido a que no existen un promedio o media en el histograma visto. Es decir, al no existir un reparto equitativo de datos no podemos obtener una imagen con las intensidades repartidas. Por lo tanto, este método también deja mucho que desear.

5.2 CREANDO UN ALGORITMO

```
function matrizAjustada=ajustePrc(matriz,gamma,prcE , prcS)
    % limites del eje de entrada (eje x)
    limXinf = prctile(double(matriz(:)), prcE);
    limXsup= prctile(double(matriz(:)), prcS);
    % limites de la salida (eje y) como trabahamos con uint16 son
positivos
    minY = 0;
    maxY = 65535;
    % recorro las filas y columnas para poder obtener las X necesatias a
la
    % hora de calcular sus respectivos Yij o salidas
    [filas, columnas] = size(matriz);


    %creo la matriz que vamos a devolver!!!!
    matrizAjustada = zeros(filas, columnas);
    % ahora recorro por cada entrada del array, lo que metamos luego en
la
    % de zeros inicializada será la salida
    for i = 1:filas
        for j = 1: columnas
            xij = double(matriz(i,j)); % double que si no no podemos
operar
            % compruebo que está dentro de los limites de los perceptiles
            if (xij > limXinf) && (xij < limXsup)
                % si cumple la condicion entonces aplicamos los
triangulos
                % semejantes
               yij = ((maxY - minY) * (xij - limXinf)) / (limXsup -
limXinf) + minY;
               %para aplicar gamma comento que era elevando el valor asi
               %que:
               yij = yij .^gamma;
            elseif xij <= limXinf
                yij = minY;
            else
                yij = maxY;
            end
            matrizAjustada(i,j) = yij;
        end
    end
    matrizAjustada = uint16(matrizAjustada);
    imshow(matrizAjustada);


end
```

Usando el algoritmo dado intentamos hacer un mejor ajuste de la imagen de manera manual:

```
imagen = dicomread(archivo);
matrizAjustada = ajustePrc(imagen,5, 99, 1);
```

La mejor imagen que conseguimos con el algoritmo es haciendo un ajuste en los intervalos de tal forma que me quedo con el intervalo del percentil 1 hasta el percentil 95. Con un valor para la constante gamma = 1.

5.3 COMPARACIÓN

IMAGEN ORIGINAL Y NEGATIVO

Con este método obtenemos una imagen cuyo objeto de interés se oscurece bastante. Provocando que no sea un método útil para este caso exacto de visualizar un tejido blando.

Muestro a continuación:

Imagen original, Imagen en negativo, Imagen ajustada para ver el hígado en negativo.

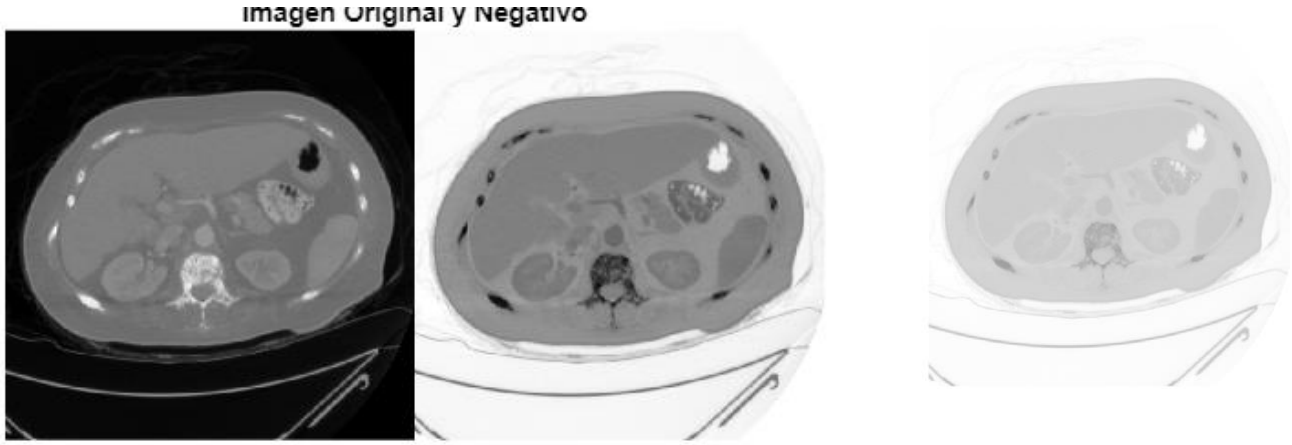
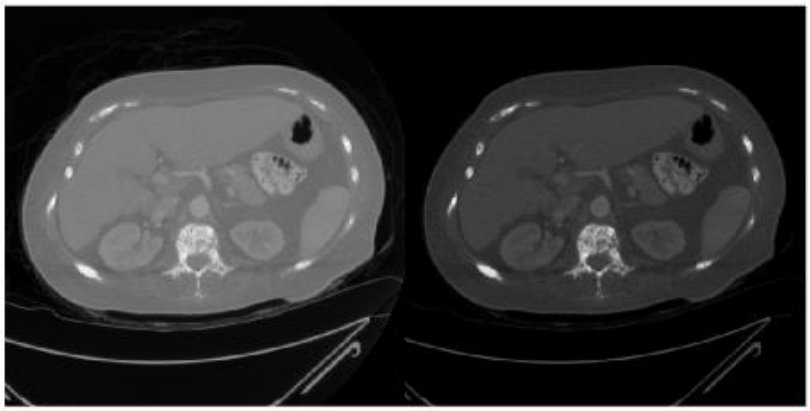


IMAGEN ORIGINAL Y GAMMA

Aunque anteriormente comenté que el negativo nos oscurecía la imagen, con e contraste gamma esto empeora pues prácticamente estamos visualizando la imagen muy oscura.

Muestro a continuación:

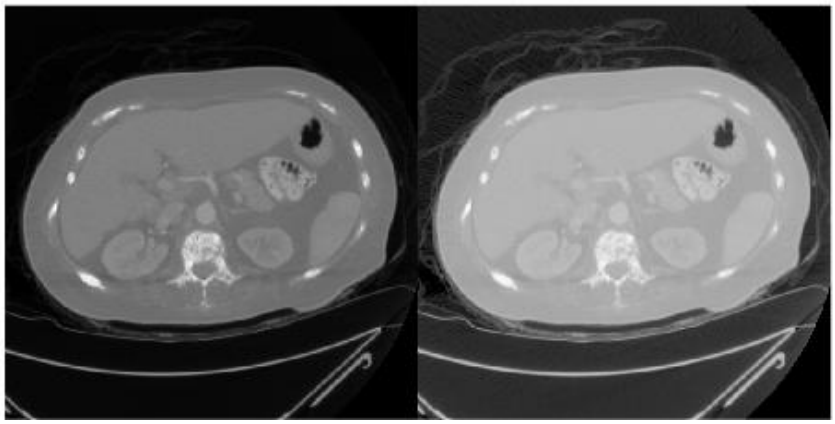
Imagen original, Imagen en gamma=2, Imagen ajustada para ver el hígado en gamma=2.

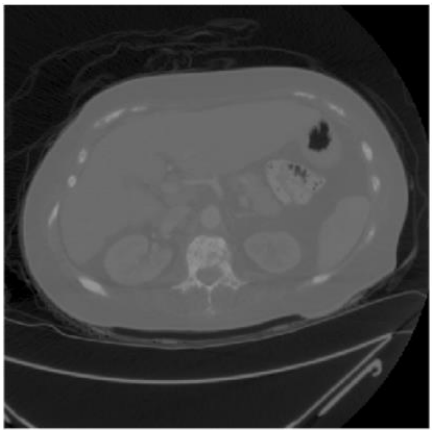


A pesar de que la imagen original ajustada con el parámetro gamma=2 no daba los resultados esperados, a la hora de ajustar la imagen en un rango válido si que conseguimos una gran visualización.

IMAGEN ORIGINAL Y RAÍZ CUADRADA

En este método realmente usabamos un gamma = 0,5. Y al hacer esto vemos que la imagen se aclara mucho, haciendo dificil la visualización.





Como con gamma=2 conseguimos una imagen más clara era de esperar que el efecto de usar un gamma menor como en este caso obtuviésemos lo contrario, siendo así una imagen más oscura donde no podemos ver bien la información deseada.

IMAGEN CON EL ALGORITMO CREADO:

Con este método, simplemente quisimos poder ajustar nosotros mismos el rango de visualización, el parámetro gamma y unos percentiles de visualización para poder recortar los datos que aparezcan en la matriz y por ende en la imagen:

```
matrizAjustada = ajustePrc(imagen,1, 99, 1);
```

IMAGEN AJUSTADA IMADJUST Y CON RANGOS DEFINIDOS:

Finalmente, tras el estudio individual y en comparación entre ellos de cada uno de los métodos. Me quedo con dos buenas imágenes que me dan la información que se esperaba:

- Imagen con el ajuste gamma = 2:

- Imagen ajustada automáticamente con imadjust() pero con rangos definidos:

Sin embargo si me tuviera que quedar con solo una, concluiría la práctica diciendo que la mejor imagen para la visualización del área de interés es la *Imagen ajustada automáticamente con imadjust() pero con rangos definidos.* El motivo de mi elección es por la gran cantidad de contraste que existe entre los distintos tejidos y componentes óseos que visualizamos.

6. CÓDIGO

```
archivo = 'C:\Users\34603\OneDrive\Documentos\MATLAB\IMAGEN
BIOMEDICA\img\im2';
imagen = dicomread(archivo)
inf = dicominfo(archivo)

[nrows,ncols] = size(imagen);
disp(['Número de filas: ', num2str(nrows),' Número de columnas: ',
num2str(ncols)]);
disp(['Descripción de imagen: ', inf.StudyDescription]);
disp(['Tipo de imagen: ', inf.ImageType]);
disp(['Esta imagen es una ',inf.Modality, ' realizada en ',
inf.InstitutionName]);
disp(['Número de bits por píxel: ', num2str(inf.BitsAllocated)]);
disp(['Formato de la imagen: ',inf.Format])

figure
imshow(imagen);
title('Imagen número 02');
% los valores varían entre 0 y 65535
% encontrar el maximo de mi array imagen :
max(imagen(:))
% por lo tanto uso los ejes 0-2200 aprox para ver la imagen con
contraste:


figure
imshow(imagen,[0 2200]);
title('Imagen número 02, ajustada a los ejes adecuados');

[counts,binLocations] = imhist(imagen);
figure
imhist(imagen)
title('Histograma original')
figure
imhist(imagen)
axis([0 2200 0 20000]);
title('Histograma original con ejes modificados')
% matriz oscila hasta el valor 2192
```

```
% ir comparando entre extremos del primer bin si el primer valor del
array
% esta en el primer intervalo del eje x que quiero en el hsitograma por
% ejemplo agruparlos en grupos de 0 a 10. Mi valor maximo es 2192. por lo
% tanto con mi funcion histograma pretendo que no me oscurezca demasiado.
% En mi caso voy a querer una funcion que recorra mi array bidimensionnal
% con ejes x e y y que vaya comparando y finalmennte cree una grafica
% histpgrama.


%%% si cumole que esta en el intervalo suma uno al histograma; si no pasa
%%% al siguiente


num_bins = 2200;
% para hallar el numero de intervalos que quiero, en principio por tener
% una imagen de uint16 podría poner hasta 65536 bins pero agrupandolos
como
% si fuera uint8 tamb serviria es decir 256. Por lo tanto; si quiero
% detallee y visualizacion usare 2200 bins que es mi maximo de
intensidad.
% Pero si quiero simplificar el histograma y que ea más manejable usre
256
% bins.
maximo = 2200;
% array, numero de intervalos en eje x, valor maximo del eje x
[count,binedges] = imhist(imagen,num_bins);
histogramacasero(imagen,100,2200)
histogramacasero(imagen,50,2200)
histogramacasero(imagen,200,2200)

% que ocurre de 800 a 1250
figure;
% esto es para ver la imagen en su rango entero:
imshow(imagen,[0 2200]);
% busco en datatip los indez maximos y minimos del higado para ver solo
esa
% ventana:
%imshow(imagen, []);
imshow(imagen,[1000 1100]);
% he usado la segunda montaña en el histograma anterior para coger el
% intervalo que me viene bn.
J = imadjust(imagen);
figure
subplot(1,2,1)
imshow(imagen,[]);
title(['OG imagen']);
subplot(1,2,2)
imshow(J,[])
title(['modify imagen']);
```

```
figure
subplot(1,2,1)
imhist(imagen);
title(['OG imagen']);
subplot(1,2,2)
imhist(J);
title(['modify imagen']);
figure;
histogramacasero(imagen,100,2200);
title(['OG imagen']);
histogramacasero(J,100,2200);
title(['modify imagen']);
imshow(J,[35000 50000]); % uso el histograma de imhist no el creado para
coger los maximos y minimoss

negativo = imadjust(imagen, [0 1], [1 0]);
figure;
imshowpair(imagen,negativo,'montage')
title('Imagen Original y Negativo');
figure
subplot(1,2,1)
imhist(imagen);
title(['OG imagen']);
subplot(1,2,2)
imhist(negativo);
title(['modify imagen']);


figure;
histogramacasero(imagen,num_bins,maximo);
title(['OG imagen']);
histogramacasero(negativo,100,2200);
title(['modify imagen']);
gammaimg = imadjust(imagen,[],[],2);
imshowpair(imagen,gammaimg,'montage')
figure
subplot(1,2,1)
imhist(imagen);
title(['OG imagen']);
subplot(1,2,2)
imhist(gammaimg);
title(['modify imagen']);
figure;
histogramacasero(imagen, 100, 2200);
title('Histograma Casero - Imagen Original');
histogramacasero(gammaimg, 100,2200);
title('Histograma Casero - Corrección Gamma');
imshow(gammaimg,[0 60]);

sqrt_img = imadjust(imagen,[],[],0.5);
figure;
imshowpair(imagen,sqrt_img,'montage');
```

```
figure;
subplot(1,2,1)
imhist(imagen);
title(['OG imagen']);
subplot(1,2,2)
imhist(sqrt_img);
title(['modify imagen']);
histogramacasero(imagen,num_bins,maximo);
title(['OG imagen histo casero']);
histogramacasero(sqrt_img,100,2200);
title(['modify imagen histo casero']);
imshow(sqrt_img,[5000 10000]);
% intento ajustar los índices para ver si con este método somos capaces
de
% ver el higado correctamente

function histograma = histogramacasero(array,num_itervalos,maximo_array)


grosor_intervalo = maximo_array/num_itervalos;
histograma=zeros(1,num_itervalos);


[filas,columnas] = size(array);
for i=1:filas
    for j=1:columnas

        intensidad = array(i, j);
        % calculo donde guardo el valor, en que intervalo
        bin_index = floor(intensidad / grosor_intervalo) + 1;


        % si el intervalo aproximado es mayor que el numero de intervalos
        % hemos llegafo al fianl
        if bin_index > num_itervalos
            bin_index = num_itervalos;
        end
        % si no entra al bucle es pq no hemos llegafo al final

        % Incrementar el contador del bin correspondiente entonces
aumento
        % el valor del contador del intervalo/bin correspondiente
        histograma(bin_index) = histograma(bin_index) + 1;


    end
end
bin_edges = (0:num_itervalos-1) * grosor_intervalo;   % limite intervalos
figure;
bar(bin_edges, histograma, 'histc'); % bar es usado para graficar
historamas
end
```

```
% los valores de salida estan tamb ekevados a ggamma, hacer triangulos
% semejantes



function matrizAjustada=ajustePrc(matriz,gamma,prcE , prcS)
    % limites del eje de entrada (eje x)
    limXinf = prctile(double(matriz(:)), prcE);
    limXsup= prctile(double(matriz(:)), prcS);
    % limites de la salida (eje y) como trabahamos con uint16 son
positivos
    minY = 0;
    maxY = 65535;
    % recorro las filas y columnas para poder obtener las X necesatias a
la
    % hora de calcular sus respectivos Yij o salidas
    [filas, columnas] = size(matriz);


    %creo la matriz que vamos a devolver!!!!
    matrizAjustada = zeros(filas, columnas);
    % ahora recorro por cada entrada del array, lo que metamos luego en
la
    % de zeros inicializada será la salida
    for i = 1:filas
        for j = 1: columnas
            xij = double(matriz(i,j)); % double que si no no podemos
operar
            % compruebo que está dentro de los limites de los perceptiles
            if (xij > limXinf) && (xij < limXsup)
                % si cumple la condicion entonces aplicamos los
triangulos
                % semejantes
               yij = ((maxY - minY) * (xij - limXinf)) / (limXsup -
limXinf) + minY;
               %para aplicar gamma comento que era elevando el valor asi
               %que:
               yij = yij .^gamma;
            elseif xij <= limXinf
                yij = minY;
            else
                yij = maxY;
            end
            matrizAjustada(i,j) = yij;
        end
    end
    matrizAjustada = uint16(matrizAjustada);
    imshow(matrizAjustada);


end
```

```
% Guardar en PNG (usando uint16)
imwrite(imagen, 'original.png');
imwrite(J, 'ajustada.png');
imwrite(negativo, 'negativo.png');
imwrite(gammaimg, 'gamma.png');
imwrite(sqrt_img, 'sqrt.png');


% Reescalar a uint8 (0-255) y guardar en JPG (para JPG es uint8)
imagen8 = uint8(255 * mat2gray(imagen)); % Debe ser uint8 para JPG
imwrite(imagen8, 'original.jpg');


J8 = uint8(255 * mat2gray(J));
imwrite(J8, 'ajustada.jpg');


negativo8 = uint8(255 * mat2gray(negativo));
imwrite(negativo8, 'negativo.jpg');


gamma8 = uint8(255 * mat2gray(gammaimg));
imwrite(gamma8, 'gamma.jpg');


sqrt8 = uint8(255 * mat2gray(sqrt_img));
imwrite(sqrt8, 'sqrt.jpg');


% Lista de archivos a comprimir
archivos = {'original.png', 'ajustada.png', 'negativo.png', 'gamma.png',
'sqrt.png', ...
            'original.jpg', 'ajustada.jpg', 'negativo.jpg', 'gamma.jpg',
'sqrt.jpg'};


% Comprimir en un archivo ZIP
nombre_zip = 'imagenes_procesadas.zip';
zip(nombre_zip, archivos);


disp(['Archivo comprimido guardado como: ', nombre_zip]);
```

7. BIBLIOGRAFÍA

- Web para el estudio de imadjust()

https://es.mathworks.com/support/search.html?q=imadjust&page=1

- Web para el estudio del histograma:

https://es.mathworks.com/help/matlab/ref/matlab.graphics.chart.primitive.histogram.html

- Libro para complementar:

https://jabega.uma.es/discovery/fulldisplay?vid=34CBUA_UMA:VU1&id=991003695449704986&inst=34CBUA_UMA&context=L