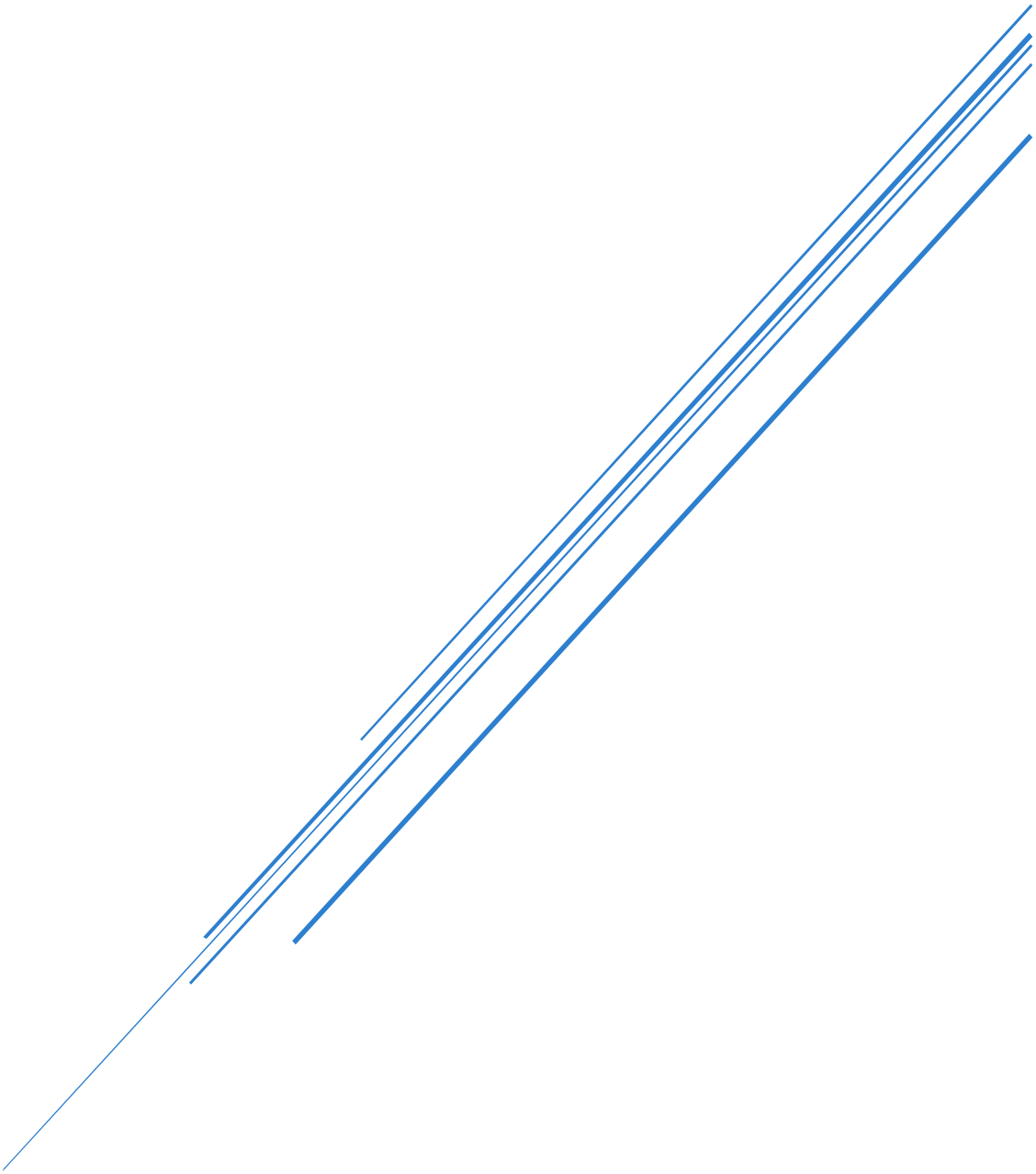
MEMORIA DE LA PRÁCTICA 2

Filtrado espacial de imágenes biomédicas



ETS Informática – Grado Ingeniería De La Salud
Sara Giménez Gómez

1. *Introducción*

En una imagen tenemos en cada celda un número almacenado el cual describe la intensidad en cada uno de ellos. Gracias a esto podemos ver si existen saltos abruptos. Es decir, la idea es que se siga la estela de que los números suban si originalmente suben o viceversa, pero con saltos menos notorios… suavizar.

Para hacer esto vamos a pasar una matriz KERNER la cual hará distintos promediados para ir suavizando o no estos valores.

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

Esta matriz que vemos es la que describíamos anteriormente y con la que vamos a trabajar a continuación para suavizarla.

Aquí se ve un poco mejor esos grandes saltos que comentaba, para encontrarlos me he movido por la matriz y seleccioné un rango de filas de la 200 a la 209 y respecto a las columnas de la 35 a la 40.

```
imagen = 512×512 uint16 matrix
          Rows 200:209 | Columns 35:40
    43    91   148   146   206
    93   169   180   169   287
   138   192   163   226   382
   157   137   136   280   431
   176   133   198   348   489
   135   126   252   403   603
   100   161   315   475   745
   118   251   384   578   874   1
   172   318   442   705   988   1
   266   392   548   842  1025   1
```

Vamos a recordar cual es la imagen con la que vamos a trabajar. Imagen a estudio ajustada al rango [35000 50000].

Filtrado espacial de la imagen

Principalmente nos vamos a olvidar de los bordes y las esquinas, pues al poner la matriz KERNEL sobre estos no tenemos la información suficiente para calcular los valores. Esto será arreglado con los distintos padding que exista, pero no entra en la práctica. A esta sum(sum(multiplicación de matrices) se le llama CONVOLUCIÓN. Será importante normalizar la matriz kernel para evitar que existan problemas en la iluminación de la imagen.

a. Filtrado por imfilter ()

Los filtros espaciales modifican la imagen mediante operaciones de convolución entre la matriz de la imagen y una matriz pequeña llamada KERNEL. Dependiendo del filtro, podremos suavizar o resaltar características como los bordes.

Para ello usamos la función imfilter (A, filtro, tipo_padding) siendo la matriz imagen ajustada = A, de lo contrario estamos cargando la imagen ooscura y no podremos visualizar la estructura realmente. Tambien le meteremos el filtro según el apartado donde nos encontremos.

Los casos a estudio serán:

i. Filtro paso bajo 5x5

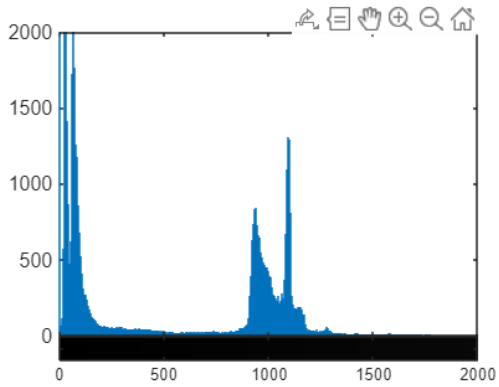
¿Qué hace este filtro?

Este filtro suaviza la imagen, reduciendo el ruido y haciendo menos notorio los cambios abruptos en los valores de los píxeles. El kernel que usaremos es un 5x5 unitario y normalizado para conservar la iluminación.

Para ello como filtro introduciré una matriz kernel normalizada.

```
dim = 5*5;
norma = 1/dim;
hPB =norma * ones (5,5)
```

```
hPB = 5×5
    0.0400    0.0400    0.0400    0.0400    0.0400
    0.0400    0.0400    0.0400    0.0400    0.0400
    0.0400    0.0400    0.0400    0.0400    0.0400
    0.0400    0.0400    0.0400    0.0400    0.0400
    0.0400    0.0400    0.0400    0.0400    0.0400
```




```
filtro_PasoBajo =
imfilter(imagen,hPB,"symmetric");
imhist(filtro_PasoBajo,(2^(16))-1)
axis([0 2000  0 2000])
```

Tras ajustar los ejes, vemos que nuestra zona de interés oscila entre 1000 y 1200:

```
imshow(filtro_PasoBajo,[1000 1200])
```

¿Qué hemos obtenido?

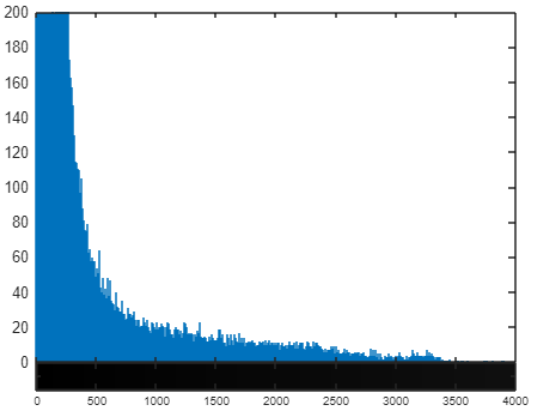
Obtenemos una buena imagen respecto a tonalidad y armonía de los tonos grises, además de ver la ventana de interés bastante bien. Sin embargo, el exceso de suvizado provoca un efecto borroso, lo que impide visualizar claramente los límites entre distintas regiones. Esto es muy típico en los filtros de paso bajo, ya que estos tienden a eliminar los detalles finos y reducir las frecuencias altas.

ii. Filtro Sobel

¿Qué hace?

Es un filtro de paso alto que destaca bordes al amplificar los camios bruscos de intensidad. Son útiles para detectar contornos y los vamos a aplicar en dos direcciones: ejes x, y.

Tras aplicar el código correspondiente para este filtro nos encontramos con esta matriz de valores y su correspondiente histograma para poder interpretar correctamente lo que ocurre:




```
gggg = 512×512 uint16 matrix
   175   175   148   137   143   149   108    87   102   116   131   122  ...
   163    34    47    27    23    68    86    25    64    71    42    11
   167    46    15    12    17    40    60    63    25    27    35    44
   139    64    33    25    23    19    51    77    87    46    24    22
   119    66    24    14    37    45    30    65    97    85    51    42
    99    66    47    18    23    47    79    61    16    94   100    73
    85    30    57    27    11     7    40    94    94    15    77    87
   104    45    22    29    22    12    22    13    84   124    61    50
   129    72    44    14    34    37     7    36    18    71   113    64
   151     9    62    42    34    26    15    31    23    32    55    58
    :
```

Lo que vemos es el histograma de la imagen una vez que la hemos devuelto a la unidad uint16. Y esto lo hacemos porque al estar trabajando con kernels direccionales estos implican hacer una serie de cálculos, aunque es verdad que luego por ser un filtro que aplica gradiente no vamos a tener valores negativos creí recomendable hacerlo de esta forma. Además, también con este histograma vemos que nuestra imagen para una buena observación debería ser [0 1000].

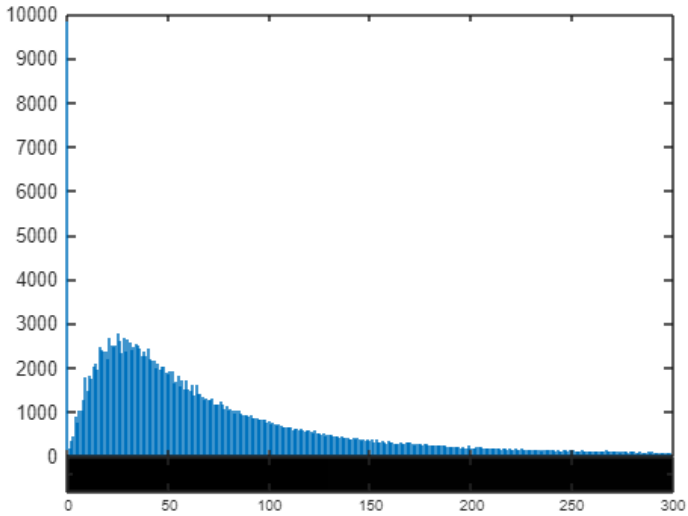
¿Qué hemos obtenido?

La salida es lo que esperábamos de un filtro de paso alto pues este implica remarcar los contornos de las imágenes, a nivel matemático serían aquellos pixeles donde existe un mayor salto con los pixeles adyacentes.

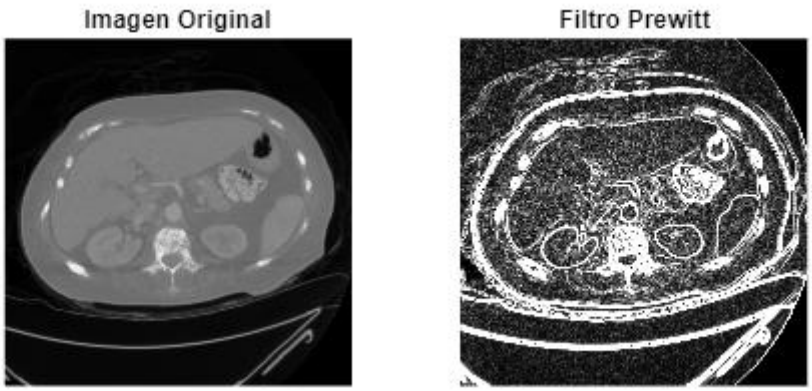
iii. Filtro Prewitt

¿Qué hace?

Es similar al filtro Sobel, pero con un cambio ligero en la forma de los kernels que se utilizarán en la convolución, es decir varía en que matrices se van a utilizar en el cálculo de la nueva imagen. Al igual que sobel también estamos ante un filtro de paso alto.



Volvemos a mostrar el histograma ajustado para poder ver donde poner el rango donde prewitt nos da mejor información y conseguimos lo siguiente:



Como conclusión, ambos filtros son útiles para la detección de bordes y esto nos es útil para localizar dónde y cómo se encuentra nuestro hígado. Sin embargo es verdad que con prewitt obtenemos un contorno más marcado pero con excesivo ruido.

iv. Filtro Gaussiano 5x5

Es un filtro de paso bajo que suaviza la imagen utilizando una distribución gaussiana a la hora de hacer los cálculos. Esto implica que se van a difuminar las intensidades de los píxeles eliminando el ruido de alta frecuencia.

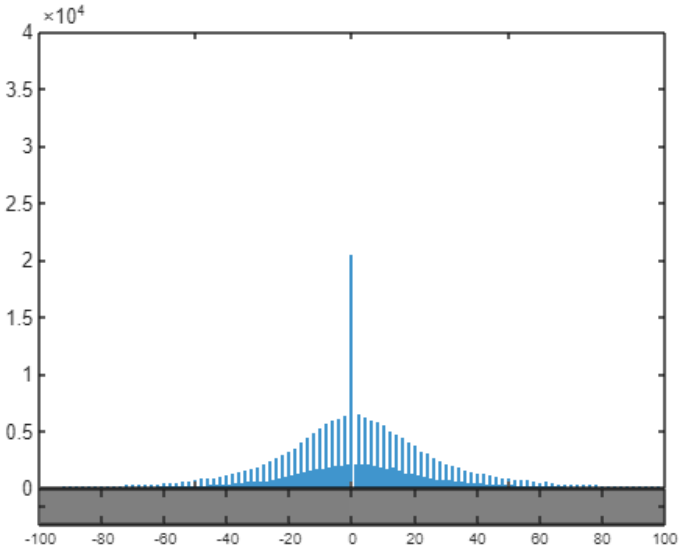
Visualizamos la imagen ajustada al rango [1000 1250] para observar cómo el filtro suaviza la imagen, manteniendo una buena estructura sin perder demasiados detalles.

v. Filtro Laplaciano 3x3

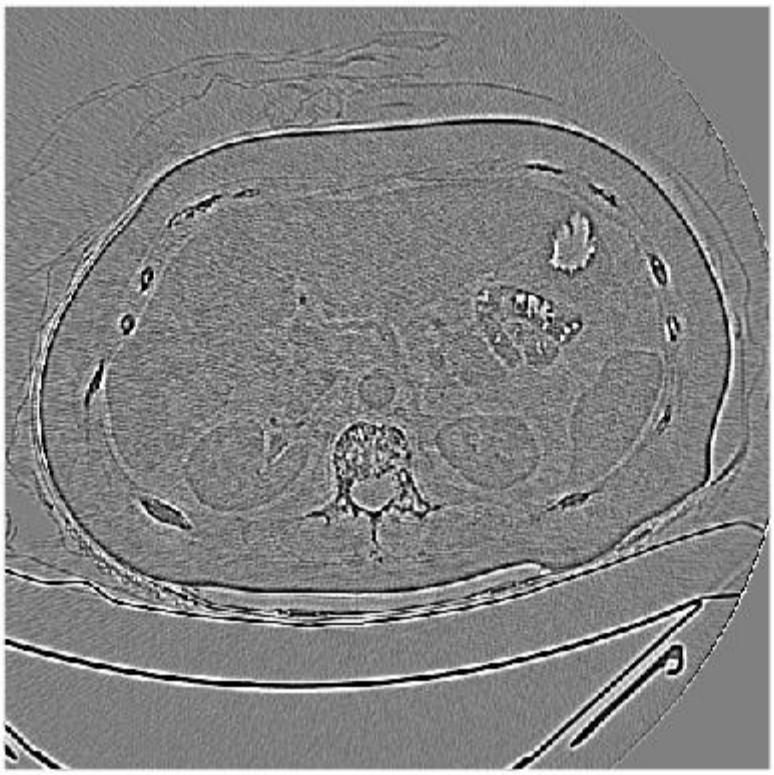
Este filtro detecta bordes al destacar los cambios abruptos en la intensidad de la imagen.

```
h_laplacian = fspecial('laplacian',1);
filtro_laplaciano = imfilter(int16(imagen), h_laplacian, 'replicate')

figure;
imhist(filtro_laplaciano,2^16)
axis([-100 100 0 40000])

figure;
imshow(filtro_laplaciano, [ -100 100])
```

En esta parte de la práctica si que veo útil compartir el código de manera aislada pues el laplaciano matemáticamente es la segunda derivada, esto implica que si acepta rango de valores negativos, por ende, para trabajar con ellos es de utilidad pasar la imagen original a int16 y de esta forma ajustar los ejes del histograma dando como solución lo siguiente:



Finalmente, la imagen resultante será:



Considero que la imagen resultante cumple con lo estudiado, remarca los contornos. Pero no soy capaz de encontrar un rango válido que me permita la correcta visualización del hígado. Achaco esto a que el hígado en comparación

con el resto de los componentes no tiene tanto salto entre los píxeles y el algoritmo no le da la misma importancia que por ejemplo un hueso, el cual tiene bastante diferencia con sus pixeles adyacentes.

vi. Filtro Laplace Gaussiano 5x5 (LoG)

Este filtro combina la suavización del filtro gaussiano con la detección de bordes del filtro laplaciano, proporcionando así una técnica más avanzada para resaltar los bordes evitando o minimizando el ruido.

A nivel de código se tuvo que hacer un ajuste automático especificando un rango de visión bastante pequeño:

Esta imagen muestra como quedaría la imagen original aplicando un filtro LoG con un rango de visión de [0 40] Vemos bastante ruido que podríamos identificarlo como el conocido como ruido de “sal y pimienta”.

b. Filtrado por algoritmo propio

```
filtro(A,hgauss);
```

```
% en mi codigo para que funcione bien hay que meterle la imagen ya
% preajustada con el rango de la zona de interés marcada
```


```
% en mi codigo para que funcione bien hay que meterle la imagen ya
% preajustada con el rango de la zona de interés marcada
```


He usado la hgaussiana pues considero que de todos los filtros e el que mejor filtrado hace por eso la use para mi implementación.

Además, si vemos el código he puesto un rango usando DataTips para que se vea mejor el hígado que es lo que me interesa en esa imagen.

2. *Detección de bordes*

a. *Usando función Edge ()*

La función edge() en MATLAB detecta bordes en una imagen buscando en su matriz de valores saltos bruscos en la intensidad de los píxeles, estos son en realidad los limites entre diferentes objetos o regiones.

La sintáxis que vamos a seguir es la siguiente:

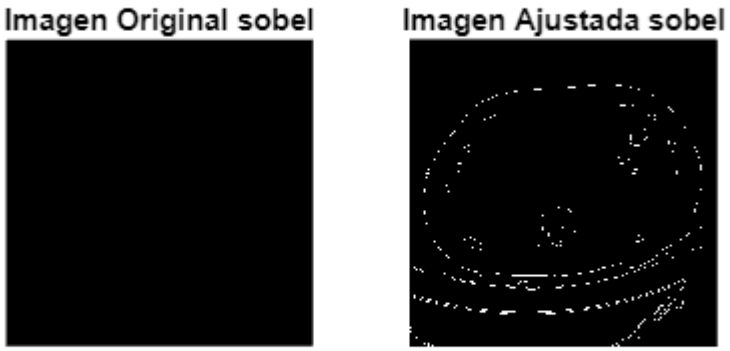
```
% vble = edge(imagen, metodo,umbral)
```

¿Qué imagen hay que meterle a esta sintaxis?

Cuando comparo la imagen original con la ajustada, visualmente vemos que la primera saldrá oscura por el formato .dicom y la segunda saldrá con la estructura correspondiente visible, a mayor o peor calidad pero algo saldrá.

SI yo aplico Edge sobre la imagen original que no tiene cambios bruscos debido a que viendo su histograma no abarca todo el rango útil que tiene el uint16 no va a dibujar bordes pues originalmente tampoco los hay.

Así que a diferencia de cuando hacíamos los filtros en el apartado anterior ahora es necesario ajustar la imagen antes de introducirla en la función de MATLAB. Así obtendremos una expansión en el rango de intensidades donde se distribuirán mejor los valores de grises.



¿que vemos en esta primera comparación?

La imagen ajustada produce bordes, aunque difícilmente vemos la parte de interés y difícilmente diferenciamos otras partes así que el siguiente paso será averiguar un umbral que nos dé más información sobre los bordes de la imagen.

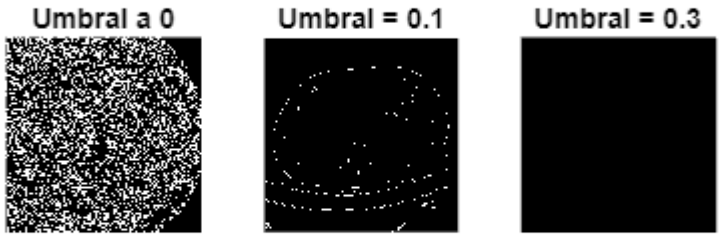
Análisis del umbral mediante fallo y error:

El umbral es un valor clave en la detección de bordes. Este por definición es "el límite que determinará si un cambio en la intensidad es lo suficientemente grande como para ser considerado un borde".

Por lo tanto, primero probaremos con:

- Umbra = 0 -> para ver de qué partimos
- Umbral = 0.1 ->un valor bajo
- Umbral = 0.3 -> un valor más bien intermedio

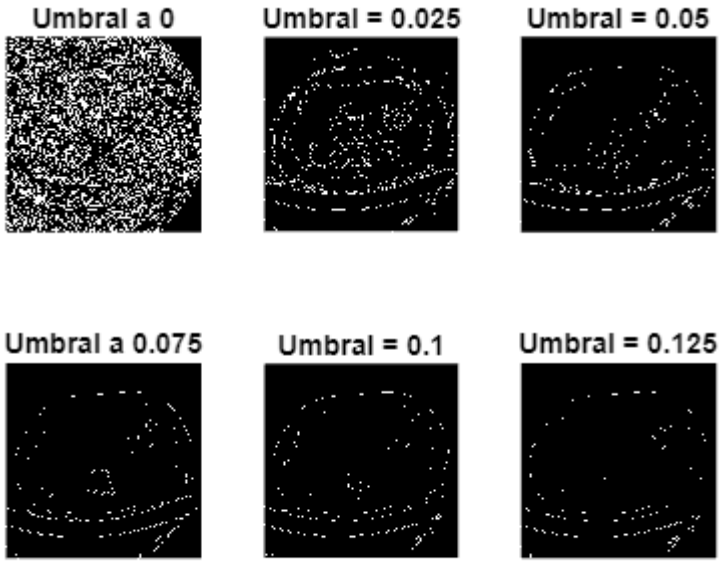
Hacemos la comparación de estos tres:



Vemos que a partir del umbral bajo predefinido como 0.1 empezamos a perder información por lo tanto, vamos a buscar otro umbral que se encuentre entre [0 0.1]

Refinar el umbral:

Lo hecho anteriormente me da una pista de que un valor óptimo para mi ejercicio se debe encontrar entre [0 0.1] así que daremos pequeños psaos de 0.025, ya que considero que dentro del rango no es ni muy pequeño ni muy grande para trabajar con este.



Tras esta comparación visual, identifico la imagen correspondiente al Umbral = 0.025 como la mejor para poder ver bien los límites de la imagen. A continuación muestro la imagen a mayor escala para que se pueda apreciar.

```
figure;
imshow(edge(A, 'sobel', 0.025)); title('Umbral = 0.025');
```

¿Porque hemos tenido que bajar tanto el umbral? ¿Qué relación guarda el umbral en la imagen?

El filtro sobel necesita cambios bruscos para poder dar buenos resultados. Como la zona donde se localiza el hígado no tenía grandes contrastes o cambios de intensidades este filtro no era capaz de encontrar los bordes y por ello tuvimos que bajar tanto el umbral.

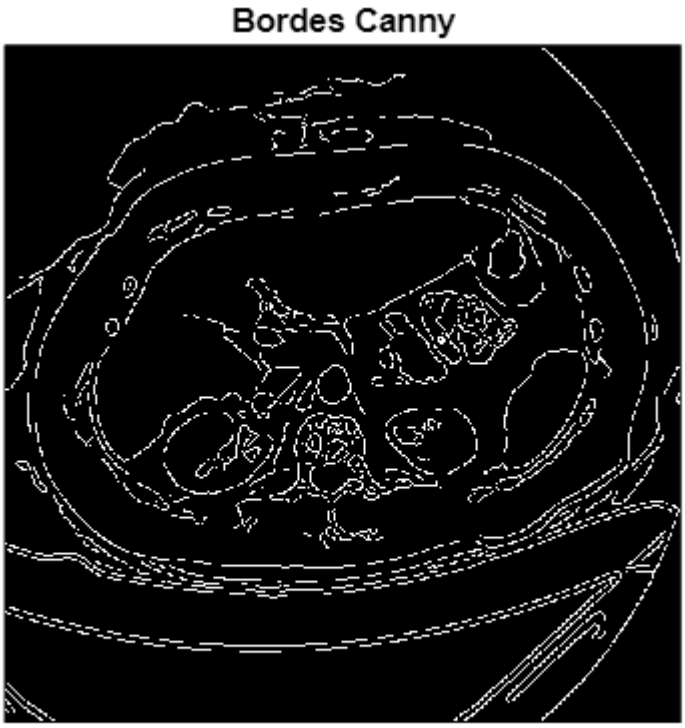
El umbral es el que va a controlar como de brusco va a ser el cambio de intensidad por ende para zonas muy claras de estudio necesitaremos umbrales pequeños de trabajo. Y así poder detectar esos pequeños cambios.

Si que es verdad que asumo el riesgo del ruido algo que si el umbral fuese más alto no tendría.

Pero para la correcta visualización de los bordes del hígado había que asumir la consecuencia.

b. Usando filtro Canny

En principio este promete detectar los bordes de tal forma que reduce el ruido. Algo que es un gran problema para mi imagen por ahora.



Para hallar los parámetros de la función me base en el estudio del apartado anterior. Mi imagen como máximo tolera un rango de [x 0.1]

Por lo tanto, para hallar la x fui probando distintos valores hasta encontrar alguna imagen la cual se pudieran diferencias los contornos y obtener esa información y no obtener

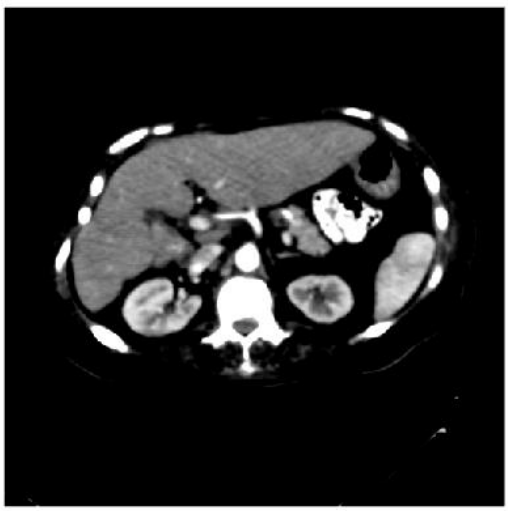
mucho ruido. (Recordatorio de que bajar el umbral está intrínsicamente ligado a poder incluir ruido)

La sigma fue puesta a 1.5 pues si aumento este valor suaviza mucho la imagen y se perdían algunos contornos importantes y si lo disminuía los contornos estaban muy detallados, pero con ruido.

3. Visualización de los resultados con modificación de contraste

Los filtros óptimos de la primera práctica fueron los de paso bajo Sobel o Prewitt y el Gaussiano. Por lo tanto:

FILTRO PASO-BAJO CON MODIFICACIÓN DE IMADJUST AUTOMÁTICO

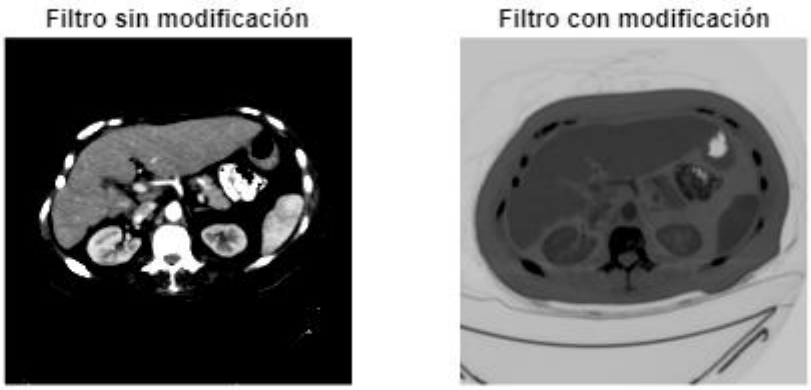


La imagen mostrada es la imagen original con un filtro paso bajo, es decir una imagen muy suavizada.

Por lo tanto, haciendo un imadjust() de forma automática vamos a ver que diferencias obtenemos:

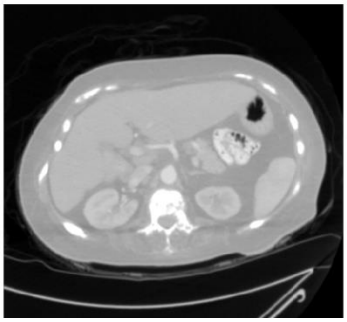
A modo de comparación vemos que la imagen con modificación resalta aún mas el hígado y sus contornos. Sin embargo, como pusimos un filtro de paso bajo vamos a obtener cierta borrosidad provocando que se pierda la textura que se observó al inicio de la práctica.

FILTRO PASO-BAJO CON MODIFICACION DE CONTRASTE NEGATIVO

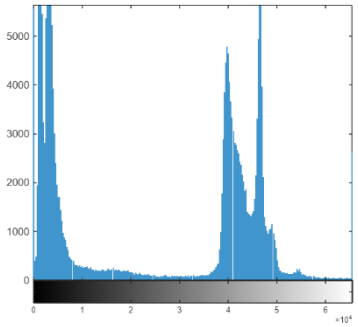


En esta comparativa vemos que tras el suavizado de la imagen por el filtro utilizado, la visibilidad del contraste usando el negativo se nos dificulta un poco, seguimos viendo el hígado pero seguro que existiría otro mejor.

FILTRO GAUSSIANO CON MODIFICACIÓN DE IMADJUST AUTOMÁTICO



En principio tendríamos la siguiente imagen para el filtro Gaussiano, pero tenemos que usar un rango de visión bastante bajito. En concreto usaré [0, 1500]

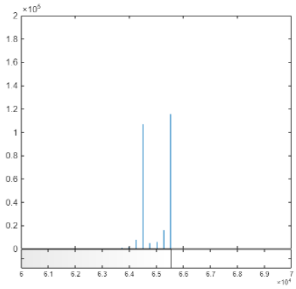


Después de esto solo nos quedaría ajustar el contraste de forma automática y conseguiremos el histograma siguiente para poder usar un rango aceptable para la correcta visualización.

En este caso nos centraremos en: [40000 50000]

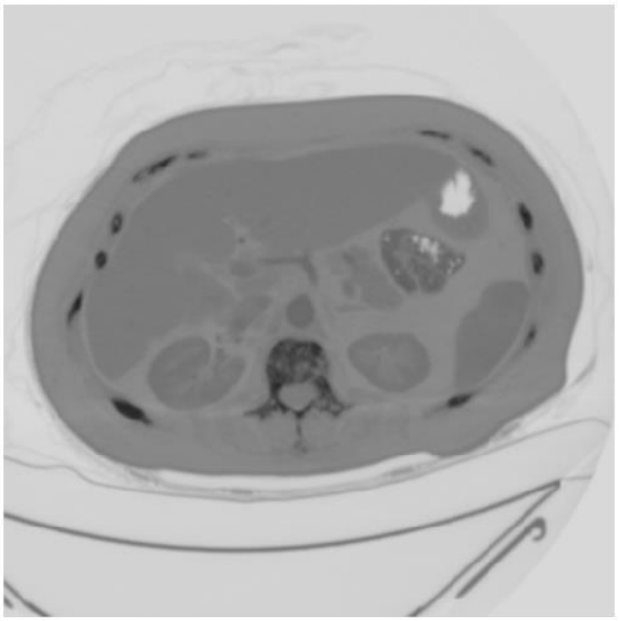
En conclusión, nuestra imagen con filtro gaussiano pero con modificación de contraste de ajuste automático sería la siguiente:

FILTRO GAUSSIANO CON MODIFICACIÓN DE CONTRASTE NEGATIVO



Agregamos un contraste negativo a la imagen original y ajustamos el rango de la imagen siguiendo el histograma correspondiente:

Por lo tanto, la imagen con el filtro gaussiano y un contraste negativo sería la siguiente:



Era de esperarse que obtuviéramos una imagen suavizada dado a que el filtro gaussiano es un filtro de paso bajo. En este caso no hemos obtenido un gran resultado, a pesar de que se diferencia el hígado de los otros componentes no considero que sea el mejor resultado posible.

4. *Valoración*

A modo de conclusión o valoración final considero que el mejor filtro con el que podríamos trabajar sería el gaussiano. Pues aporta suavizado, pero sin dejar de perder los bordes, a las pruebas de toda la práctica me remito. De hecho para poder mejorar este filtro considero que lo mejor sería aplicar un contraste automático usando imadjust() ya que este nos da unos increíbles resultados.

Sin embargo, si queremos usar el método de contraste como el del negativo si que es verdad que usaría un filtro de paso bajo asi como Sobel o Prewitt, cualquiera de los dos me sería útil pues la gran diferencia que tienen es en como calculan la solución y esta a simple vista es despreciable.

5. *Guardado de imágenes*

La

6. *Código*

```
archivo = "C:\Users\34603\OneDrive\Documentos\MATLAB\IMAGEN
BIOMEDICA\img\PR2\im2";
imagen = dicomread(archivo);
imagen
```

```
A = imadjust(imagen)

imhist(A)

imshow(A, [35000 50000]);

dim = 5*5;

norma = 1/dim;

hPB =norma * ones (5,5)

filtro_PasoBajo = imfilter(imagen,hPB,"symmetric");% imshow(iMAGEN,[])
VISUALIZO 7 NO CAMBIO

imhist(filtro_PasoBajo,(2^(16))-1)

axis([0 2000  0 2000])


imshow(filtro_PasoBajo,[1000 1200])


matrix_X = [-1,0,1; -2,0,2; -1,0,1];

matrix_Y = matrix_X';

G = sqrt(matrix_Y .^2 + matrix_Y.^2);

% uso la imagen para el filtro y ya luego ajustaré si es necesario!!!

filtro_Sobel = imfilter(imagen,G,"replicate")

imhist(filtro_Sobel,(2^(16))-1)

axis([6000 16000  0 500])

imshow(filtro_Sobel,[12000 13000])

Px = [-1,-1,-1; 0,0,0; 1,1,1];

Py = Px';

P = sqrt(Px .^2 + Py.^2);

filtro_Prewitt = imfilter(imagen,P,"replicate" )

imhist(filtro_Prewitt)

axis([0 20000  0 20000])

imshow(filtro_Prewitt,[10000 12000])

hgauss = fspecial('gaussian',[5 5],1); % que sería sigma

filtro_Gauss = imfilter (imagen,hgauss,"symmetric",1)

imhist(filtro_Gauss)

axis([0 3000 0 150000])

imshow(filtro_Gauss, [1000 1250])

% defino la h de dos formas distintas:
```

```
hlaplaciano1 = [0,1,0;1,-4,1;0,1,0];

hlaplaciano2 = [0,-1,0;-1,4,-1;0,-1,0];

fff1 = imfilter(imagen,hlaplaciano1,'replicate');

fff2 = imfilter(imagen,hlaplaciano2,'replicate');

imshow(fff1, [0 250])

imshow(fff2, [0 250])

h_laplacian = fspecial('laplacian',1);

filtro_laplaciano = imfilter(imagen, h_laplacian, 'replicate')%CONVERTIR
A VALORES NEGATIVOS PARA VISUALIZAR FATOS QUE PIERDO


imhist(filtro_laplaciano)

axis([0 1000 0 40000])

imshow(filtro_laplaciano, [0 250])

h_log = fspecial('log', [5 5], 1);

filtro_laplaceGaussiano = imfilter(imagen, h_log, 'replicate')

imhist(filtro_laplaceGaussiano);

axis([0 5 0 40000])

imshow(filtro_laplaceGaussiano,[0 40]);

filtro(A,hgauss);

% en mi codigo para que funcione bien hay que meterle la imagen ya

% preajustada con el rango de la zona de interés marcada

% vble = edge(imagen, metodo,umbral)

umbral0 = 0;

umbral1 = 0.1; % Umbral bajo

umbral2 = 0.3; % Umbral medio

bordes_sobel_cero = edge(A, 'sobel', umbral0);

bordes_sobel_bajo1 = edge(imagen, 'sobel', umbral1);

bordes_sobel_bajo2 = edge(A, 'sobel', umbral1);

bordes_sobel_medio = edge(A, 'sobel', umbral2);


figure;

subplot(1,2,1);imshow(bordes_sobel_bajo1); title('Imagen Original
sobel');
```

```
subplot(1,2,2);imshow(bordes_sobel_bajo2); title('Imagen Ajustada
sobel');

figure;

subplot(1,3,1);imshow(bordes_sobel_cero); title('Umbral a 0');

subplot(1,3,2);imshow(bordes_sobel_bajo2); title('Umbral = 0.1');

subplot(1,3,3);imshow(bordes_sobel_medio); title('Umbral = 0.3');

figure;

subplot(2,3,1);imshow(edge(A, 'sobel', 0)); title('Umbral a 0');

subplot(2,3,2);imshow(edge(A, 'sobel', 0.025)); title('Umbral = 0.025');

subplot(2,3,3);imshow(edge(A, 'sobel', 0.05)); title('Umbral = 0.05');

subplot(2,3,4);imshow(edge(A, 'sobel', 0.075)); title('Umbral a 0.075');

subplot(2,3,5);imshow(edge(A, 'sobel', 0.1)); title('Umbral = 0.1');

subplot(2,3,6);imshow(edge(A, 'sobel', 0.125)); title('Umbral = 0.125');

figure;

imshow(edge(A, 'sobel', 0.025)); title('Umbral = 0.025');

sigma = 1;

bordes_canny3param = edge(imagen, 'canny', [0.015 0.1],1.5);


figure;

imshow(bordes_canny3param); title('Bordes Canny');

% filtro prewitt ajustado automa (pr1) sin modificacion de contraste

imshow(filtro_Prewitt,[10000 12000])

Prewitt_ajustada = imadjust(filtro_Prewitt);

imhist(Prewitt_ajustada)



imshow(Prewitt_ajustada,[40000 50000]) % modificado con contraste
automatico imafjust y ya


% comparo prewitt con/sin modificacion.

figure;

subplot(1,2,1);imshow(filtro_Prewitt,[10000 12000]); title('Filtro sin
modificación');

subplot(1,2,2);imshow(Prewitt_ajustada,[40000 50000]); title('Filtro con
modificación');
```

```
% filtro prewitt ajustado negativo (pr1) sin modificacion de contraste
imshow(filtro_Prewitt,[10000 12000])
Prewitt_ajustada = imcomplement(filtro_Prewitt);
imhist(Prewitt_ajustada)
axis([50000 70000 0 100000])

figure;
imshow(Prewitt_ajustada,[52500 57500])
figure;
subplot(1,2,1);imshow(filtro_Prewitt,[10000 12000]); title('Filtro sin
modificación');
subplot(1,2,2);imshow(Prewitt_ajustada,[52500 57500]); title('Filtro con
modificación');



% filtro prewitt ajustado automa (pr1) sin modificacion de contraste
figure;
imshow(filtro_Gauss,[0 1500])
imhist(filtro_Gauss)
axis([0 10000, 0 20000])
gaussAdjust = imadjust(filtro_Gauss);
imhist(gaussAdjust)

figure;
imshow(gaussAdjust,[40000 50000]) % modificado con contraste automatico
imafjust y ya

gaussnegativo = imcomplement(filtro_Gauss);
imhist(gaussnegativo)
axis([60000 70000 0 200000])
figure;
imshow(gaussnegativo,[63000 66000])
```

```
function filtro(imagen,kernel)
    [filas_imagen, cols_imagen] = size(imagen);
    [filas_kernel, cols_kernel] = size(kernel);

    % Calcular los offsets para centrado del kernel
    offset_filas = floor(filas_kernel / 2);
    offset_cols = floor(cols_kernel / 2);

    imagen_filtrada = zeros(filas_imagen, cols_imagen);

    for i = (1 + offset_filas):(filas_imagen - offset_filas)
        for j = (1 + offset_cols):(cols_imagen - offset_cols)

            % Extraer la vecindad alrededor del píxel actual
            region = imagen(i - offset_filas:i + offset_filas, j -
offset_cols:j + offset_cols);

            % Aplicar la convolución (multiplicar y sumar)
            valor = sum(sum(double(region) .* kernel));

            % Asignar el valor al píxel de la imagen filtrada
            imagen_filtrada(i, j) = valor;
        end
    end
     imagen_filtrada = uint16(imagen_filtrada);
     imshow(imagen_filtrada, [39000 47000])% Asegurar el tipo de datos
compatible con imágenes DICOM
end
```

7. *Bibliografía*

- Libro de la asignatura para entender los conceptos de bordes