Sistemas de Visión Artificial

Práctica 2. Técnicas Básicas de Procesamiento de Imágenes

Nombre: Juan Casado Ballesteros

Fecha: 1 de octubre de 2019

El objetivo principal de esta práctica es familiarizarse con las técnicas básicas de procesamiento de imágenes mediante la Image Processing Toolbox de Matlab. Para ello se proponen ejercicios sobre operaciones geométricas, ecualización del histograma, diseño e implementación de filtros y filtrado de imágenes en el dominio del espacio, mediante máscaras de convolución. La mayoría de las técnicas de mejora se aplican a imágenes de intensidad (escala de grises). Para mejorar una imagen RGB, trabajaremos, normalmente, con las matrices de las componentes rojo, verde y azul separadamente.

La memoria de la práctica se completará por parejas en este mismo fichero .mlx, añadiendo el código en cada apartado, además de los comentarios pertinentes, la explicación del código y conclusiones sobre cada resultado obtenido. Las distintas partes serán secciones independientes (deberá insertar "section break"), y se visualizarán los resultados conjuntamente con subplot cuando sea conveniente.

Se valorará positivamente la ampliación con las explicaciones que considere oportunas, pruebas adicionales (modificando parámetros, utilizando otras imágenes que resalten algún efecto del procesamiento, etc.), descripción de problemas surgidos en la ejecución de la práctica y solución proporcionada, etc. La entrega se realizará renombrando este fichero para incluir los nombres de los autores, y enviándolo por email a sira.palazuelos@uah.es (p. ej.: Estudiante1Apellido1_Estudiante2Apellido2.mlx)

Implemente el siguiente código:

1. Cargue la imagen en color (‘flowers.tif’, 'trees.tif',…) e indique el formato de la misma (RGB, mapa de colores indexado, etc.).

Para cargar una imagen utilizamos la función `imread`.
Para ver la información de una imagen utilizamos la función `imfinfo`.

```
ruta = '/ruta/a/las/imágenes';
flowers = imfinfo(strcat(ruta,'flowers.tif'));
peppers = imfinfo(strcat(ruta,'peppers.png'));
trees = imfinfo(strcat(ruta,'trees.tif'));
```

Realizamos esto para varias imágenes (flowers.tif, trees.tif y peppers.png). Para saber el formato de las imágenes debemos ver el campo `ColorType` de la estructura que `imfinfo` nos retorna.

```
display(strcat('La imagen flowers es de tipo: ',flowers.ColorType));
display(strcat('La imagen peppers es de tipo: ', peppers.ColorType))
```

Esto funciona en las imágenes flowers y peppers que son ambas de tipo truecolor, es decir, son imágenes RGB. En el formto .png podemos saber que es RGB solo con mirar el campo `ColorType` pero para la imagen tif debemos mirar también en el campo `PhotometricInterpretation` que en este caso nos lo confirma (podría haber sido CYMK…).
Trees es un caso distinto pues esta imagen es en realidad un conjunto de 5 imágenes que podrían cada una tener formatos distintos. Con un bucle podemos observar el formato de cada imagen viendo que las 5 son de tipo indexed de modo que para utilizarlas deberemos cargar tanto la imagen que es una matriz de claves como el mapa de color que es el que indica el color al que cada clave se refiere.

```
len_trees = size(trees);
for image = 1:len_trees(1)
    display(strcat('La imagen trees [', int2str(image),'] es de tipo: ',
trees(image).ColorType))
end
```

2. Visualice la imagen que acaba de leer.

Para visualizar las imágenes debemos utilizar la sentencia `imread` para cargarlas en una variable y posteriormente `imshow` para visualizar el contenido de dicha variable como una imagen. Ya que son true color en las dos primeras imágenes (flowers y peppers) solo tendremos que cargar una variable, en el caso de trees deberemos cargar dos como hemos mencionado anteriormente, una con la matriz de claves y otra que contiene el mapa de color que enlaza cada clave con su color.
De trees mostramos solo la primera de las 5 imágenes para mostrar todas deberíamos hacerlo en un bucle.
Mostramos las tres imágenes en una misma ventana.

```
flowers = imread(strcat(ruta,'flowers.tif'));
peppers = imread(strcat(ruta,'peppers.png'));
[trees, map_trees] = imread(strcat(ruta,'trees.tif'));

figure('Name', 'Imágenes EJ2')
subplot(1,3,1);imshow(flowers);title('Flowers')
subplot(1,3,2);imshow(peppers);title('Peppers')
subplot(1,3,3);imshow(trees,map_trees);title('Trees')
```

3. Convierta la imagen anterior a una imagen en escala de grises (conteniendo sólo la información de la luminancia) y almacénela en un array llamado i. Indique el rango de valores en los que se cuantifica la luminancia (nivel mínimo y nivel máximo) de la imagen resultante.

A continuación, trabajaremos solo con la imagen flowers que damos a escala de grises t damos el nombre de i tal y como se nos ha pedido en el enunciado. Posteriormente calculamos su rango de valores y lo mostramos por pantalla.

Ya que la imagen es true color debemos utilizar la función `rgb2gray`, si hubiera sido indexada deberíamos haber utilizado `ind2gray` pasando como argumentos la matriz primero y el mapa de color después.

```
i = rgb2gray(flowers);
figure('Name', 'EJ3')
subplot(1,2,1);imshow(flowers); title('Imagen original')
subplot(1,2,2);imshow(i); title('Imagen en gris')
iluminance_max = max(i,[], 'all');
iluminance_min = min(i,[], 'all');

display(strcat('La iluminación máxima es: ', int2str(iluminance_max)))
display(strcat('La iluminación mínima es: ', int2str(iluminance_min)))
```

4. Averigüe las dimensiones de nuestra imagen i (filas x columnas).

Para saber las dimensiones de nuestra imagen utilizamos la función `size` sobre la matriz de la imagen. Esta función se puede aplicar sobre cualquier matriz contenga los datos de una imagen o no. En caso de que la imagen fuera indexada `size` deberá aplicarse sobre la matriz de la imagen y no sobre el mapa de color.

```
gray_flowers_lenght = size(gray_flowers);

display(strcat('La imagen tiene: ', int2str(gray_flowers_lenght(1)), ' filas'))
display(strcat('La imagen tiene: ', int2str(gray_flowers_lenght(2)), '
columnas'))
```

5. Muestre el histograma de la imagen en escala de grises.

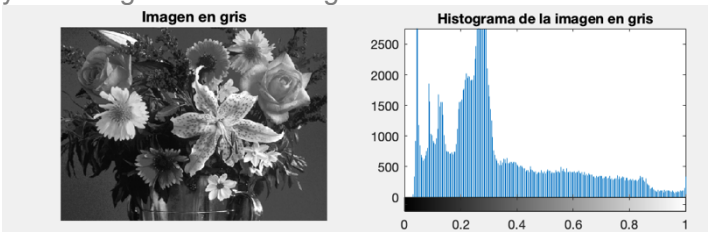
Para mostrar el histograma de una imagen debemos utilizar la función `imhist` que nos solo nos retorna el histograma si no que además nos lo muestra sobre una ventana.
Mostramos el histograma tal cual es y el histograma agrupado en 100 valores de gris.

```
figure('Name', 'EJ5')
subplot(3,3,4);imshow(gray_flowers); title('Imagen en gris')
subplot(3,3,2:3);imhist(gray_flowers); title('Histograma completo')
subplot(3,3,8:9);imhist(gray_flowers, 100); title('Histograma agrupado con
n=100')
```

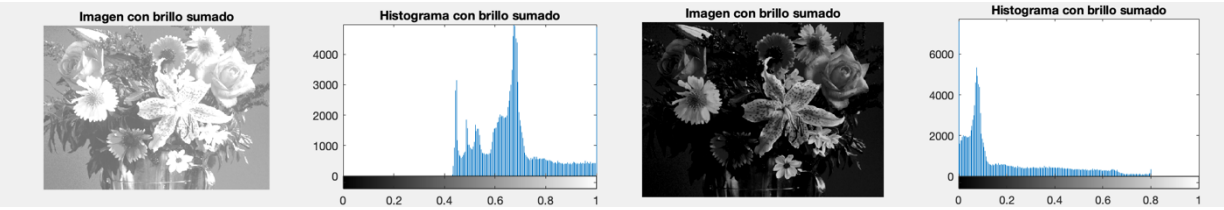
Podemos ver en el histograma como los valores de baja iluminación son predominantes, aunque los valores de alta iluminación llegan hasta el máximo de la imagen, el contraste de la imagen será por tanto elevado, ocupando casi todo el espectro de iluminación, pero la media de esta será baja.
Contraste 0.9686 (sobre 1)
Media: 0.3316 (sobre 1)

6. Modifique el brillo (sumando una constante) y el contraste (multiplicando por una constante) de la imagen en escala de grises y visualice las imágenes resultantes y sus histogramas.

Abordaremos este problema en dos partes distintas, primero modificaremos el brillo a partir de sumarle una constante y luego modificaremos el contraste mediante un factor. Ambos resultados los compararemos con la imagen de partida y especialmente con su histograma.
La imagen de partida y su histograma son los siguientes:



Primero modificaremos el brillo. Los resultados que debemos obtener deben de afectar a la integral del histograma de modo que este debe desplazarse a derecha y a izquierda sin que su forma se vea afectada.

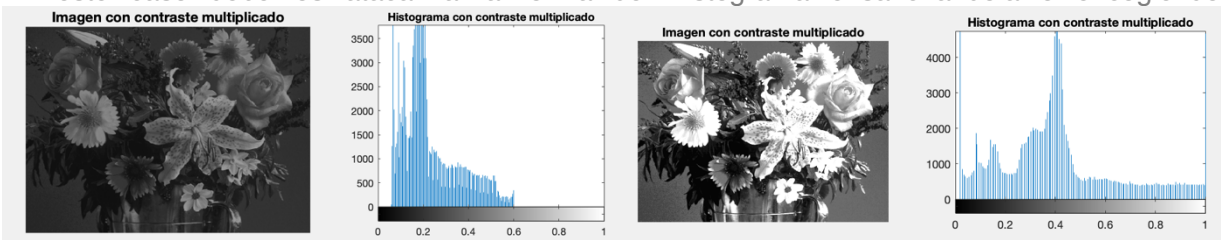


Aumento del brillo Reducción del brillo

Logramos esto ajustando linealmente los brillos máximos de entrada y salida tratando el caso de un factor de ajuste positivo y un negativo por separado.

```
brillo_a_sumar = -0.2; %Puede ser positivo o negativo menor que 1

if brillo_a_sumar > 0
    % Para aumentar el brillo debemos lograr desplazar el histograma de la
    % imagen hacia la derecha.
    brillo_sumado = imadjust(gray_flowers,...
        [0 1-brillo_a_sumar],[brillo_a_sumar 1]);
else
    % Para aumentar el brillo debemos lograr desplazar el histograma de la
    % imagen hacia la izquierda.
    brillo_sumado = imadjust(gray_flowers,...
        [abs(brillo_a_sumar) 1],[0 1-abs(brillo_a_sumar)]);
end
```

En el caso de ajustar el contraste con un factor multiplicativo solo debemos multiplicar la entrada y la salida de brillos por el factor y asegurarnos de que nos mantenemos dentro de valores menores o iguales que 1 para el brillo máximo y mayores o iguales que 0 para el mínimo.
En este caso debemos atacar a la forma del histograma ensanchándola o encogiéndola.



Aumento del contraste Reducción del contraste

```
contraste_a_multiplicar = 0.7; %Siempre positivo si <0 reduce, si >0 amuenta

contraste_multiplicado = imadjust(gray_flowers,...
        [max(0, min(gray_flowers, [], "all")*contraste_a_multiplicar),...
         min(1, max(gray_flowers, [], "all")*1/contraste_a_multiplicar)],...
        [max(0, min(gray_flowers, [], "all")*1/contraste_a_multiplicar),...
         min(1, max(gray_flowers, [], "all")*contraste_a_multiplicar)]);
```

Podemos ver como en los casos de cambiar el brillo y aumentar el contraste se producen saturaciones de la imagen en el 0 y el 1 del histograma, loa valores se agolpan ahí.
Mostramos las imágenes y los histogramas en una sola.

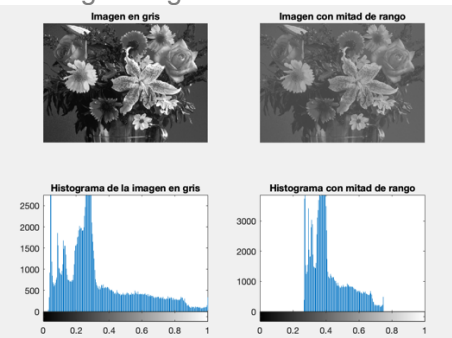
```
figure('Name', 'EJ6')
subplot(2,3,1);imshow(gray_flowers); title('Imagen en gris')
subplot(2,3,4);imhist(gray_flowers); title('Histograma de la imagen en gris')
subplot(2,3,2);imshow(brillo_sumado); title('Imagen con brillo sumado')
subplot(2,3,5);imhist(brillo_sumado); title('Histograma con brillo sumado')
subplot(2,3,3);imshow(contraste_multiplicado); title('Imagen con contraste
multiplicado')
subplot(2,3,6);imhist(contraste_multiplicado); title('Histograma con contraste
multiplicado')
```

7. Realice una binarización de la imagen i con un umbral cuyo valor sea aproximadamente la mitad de su rango de grises y visualice el resultado.

Para hacerlo tomaremos la imagen en double de flowers y redeciremos su contraste a la mitad de modo que el rango de grises de la imagen resultado sea la mitad del rango de grises de la imagen original.

```
mitad_de_rango = imadjust(gray_flowers, [0,1], [0.25, 0.75]);

figure('Name', 'EJ7')
subplot(2,2,1);imshow(gray_flowers); title('Imagen en gris')
subplot(2,2,3);imhist(gray_flowers); title('Histograma de la imagen en gris')
subplot(2,2,2);imshow(mitad_de_rango); title('Imagen con mitad de rango')
subplot(2,2,4);imhist(mitad_de_rango); title('Histograma con mitad de rango')
```

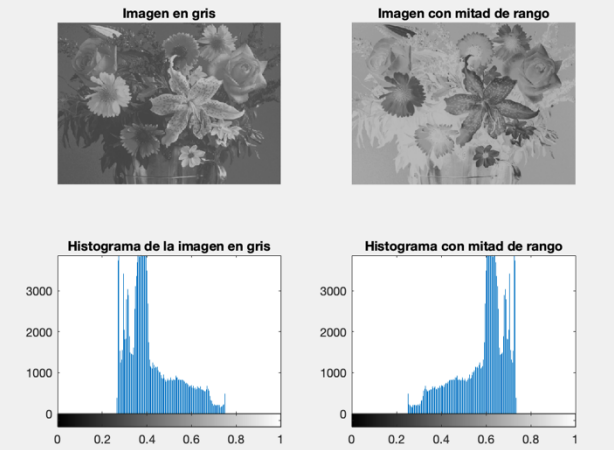
Como resultado obtenemos una imagen cuyo histograma ha sido comprimido de modo que tenga la mitad de los valores de gris que la imagen original.



8. Obtenga el negativo de la imagen en escala de grises obtenida en el punto anterior.

A partir de la imagen obtenida en el punto anterior invertimos sus valores de gris, es decir, le daremos obtendremos el inverso del histograma con respecto al eje y.

```
invertida = imadjust(mitad_de_rango, [0, 1], [1, 0]);

figure('Name', 'EJ8')
subplot(2,2,1);imshow(mitad_de_rango); title('Imagen en gris')
subplot(2,2,3);imhist(mitad_de_rango); title('Histograma de la imagen en gris')
subplot(2,2,2);imshow(invertida); title('Imagen con mitad de rango')
subplot(2,2,4);imhist(invertida); title('Histograma con mitad de rango')
```



9. Obtenga tres versiones ruidosas de la imagen i:

Aplicaremos ahora con la función `imnoise` tres distintos tipos de ruido a la imagen.

(a) con ruido de distribución uniforme ‘speckle’,

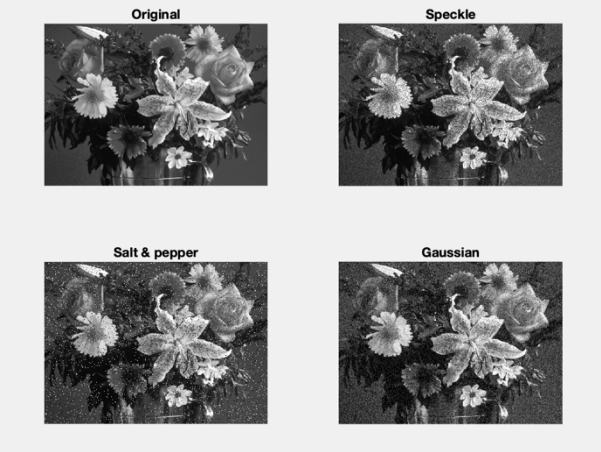
```
gray_flowers_speckle = imnoise(gray_flowers,'speckle');
```

(b) con ruido de tipo ‘sal y pimienta’ de densidad de ruido 5%,

```
gray_flowers_salt_and_pepper = imnoise(gray_flowers,'salt & pepper',0.05);
```

(c) con ruido de tipo gaussiano de media cero y varianza 0.01.

```
gray_flowers_gaussian = imnoise(gray_flowers,'gaussian',0,0.01);
```



El ruido satl&pepper consiste en añadir puntos blancos y negros a la imagen, el ruido speakle por el contrario es un ruido multiplicativo. El ruido gaussiano distribuye con cierta varianza y media ruido a lo largo de la imagen.

10. Partiendo de la imagen contaminada con ruido gaussiano anterior, fíltrela utilizando la media del entorno de vecindad, primero con un entorno de vecindad de 3x3 y luego con uno de 5x5, comparando los resultados obtenidos.

Para completar este ejercicio se han aplicado los filtros de reducción de ruido sobre todas las imágenes contaminadas anteriormente, no solo para la contaminada con ruido gaussiano de modo que se puede ver como cada tipo de ruido el corregido con cada filtro. Para corregir las imágenes utilizaremos las funciones `filter2` y `fspecial`.
Kernell 3x3:

```
 gray_flowers_3=filter2(fspecial('average',3),gray_flowers);
 gray_flowers_speckle_3=filter2(fspecial('average',3),gray_flowers_speckle);
 gray_flowers_salt_and_pepper_3=filter2(fspecial('average',3),gray_flowers_salt_
and_pepper);
 gray_flowers_gaussian_3=filter2(fspecial('average',3),gray_flowers_gaussian);
```

Kernell 5x5:

```
 gray_flowers_5=filter2(fspecial('average',5),gray_flowers);
 gray_flowers_speckle_5=filter2(fspecial('average',5),gray_flowers_speckle);
```

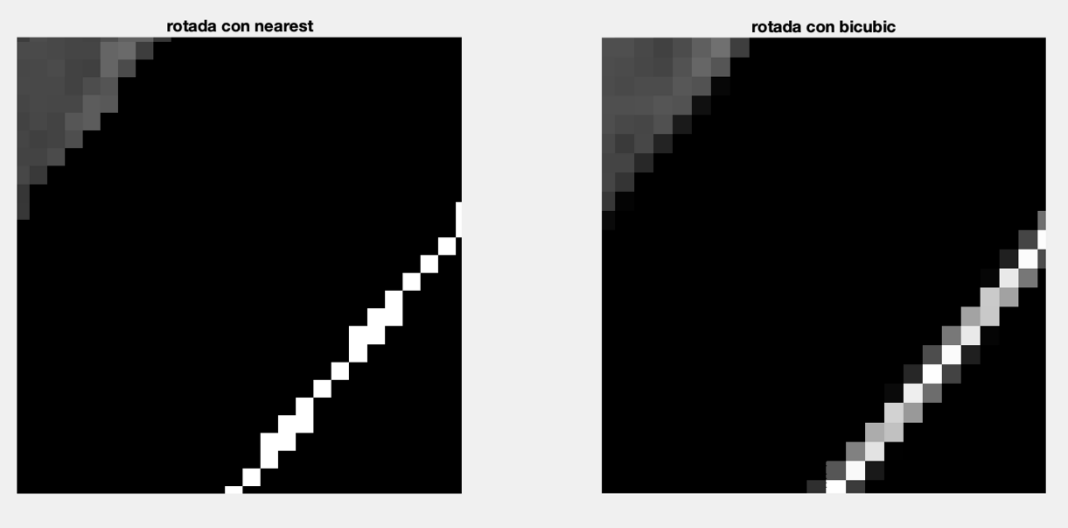
```
 gray_flowers_salt_and_pepper_5=filter2(fspecial('average',5),gray_flowers_salt_
and_pepper);
 gray_flowers_gaussian_5=filter2(fspecial('average',5),gray_flowers_gaussian);
```

Podemos ver como los ruidos que mejor son corregidos con el filtro de media son el gaussiano y el speakle, en el caso del ruido salt&pepper este simplemente ha sido difuminado por la imagen produciendo malos resultados tras haber sido filtrada. En el caso del ruido salt&pepper hubiera sido mucho mejor haber aplicado un filtro pasa bajos como la mediana en vez de la media.
Cuanto menor es el filtro menor lo es también la corrección, si este fuera demasiado grande también se obtiene pobres resultados, es fundamental utilizar el tamaño del kernell adecuado para cada imagen y ruido.

11. Lea la imagen venas.tif y rótela un ángulo de 50 grados (por ejemplo), utilizando los métodos de interpolación nearest y bicubic (parámetros de la función imrotate). Aumente las imágenes lo suficiente para ver las diferencias entre ambos resultados.

Rotamos la imágen con la función `imrotate`.

```
venas_nearest_rotated=imrotate(venas,50,'nearest');
venas_bicubic_rotated=imrotate(venas,50,'bicubic');
```



Podemos observar como la rotación con interpolación de vecino más cercano no se inventa valores nuevos, esta genera líneas afiladas en la imagen resultado, esto puede resultar desagradable en algunos casos y hacer que la imagen rotada se vea “fea”. Con la interpolación bicúbica si se crean nuevos tonos de gris haciendo que los bordes sean más suaves, puede resultar más agradable, pero en cierto modo nos estamos inventando más información.

12. A partir de la imagen original blood1.tif, obtenga 16 imágenes en diferentes ensayos con ruido gaussiano de media cero y varianza 0.05. Guárdelas en ficheros con sus respectivos nombres (ruidosa1.bmp, …, ruidosa16.bmp).

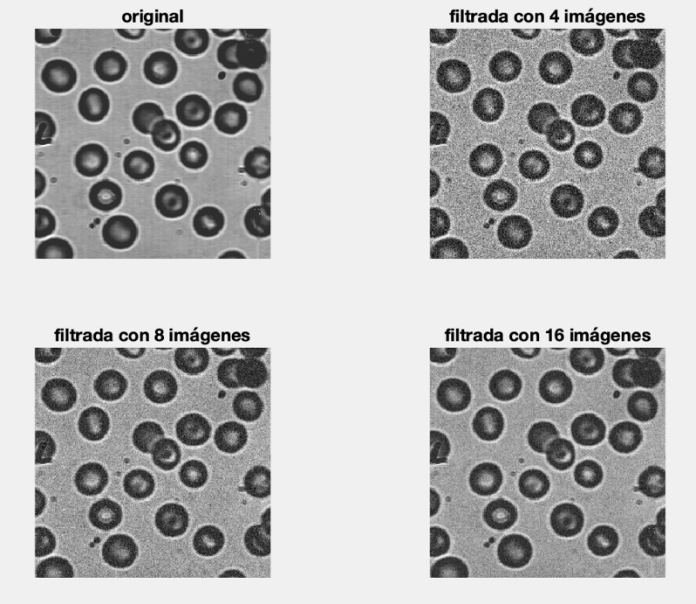
Aplicamos el ruido con `imnoise` y guardamos las imágenes obtenidas con `imwrite`.

```
blood = imread(strcat(ruta,'blood1.tif'));

%Aplicar ruido
for i = 1:16
    noise = imnoise(blood,'gaussian',0,0.05);
    imwrite(noise, strcat(ruta,'blood_gaussian',num2str(i),'.bmp'));
end
```

13. Realice pruebas de filtrado de ruido usando la técnica de promediado de imágenes sobre las imágenes creadas en el punto anterior. Para ello, promedie en primer lugar con 4 imágenes ruidosas, luego con 8 y, finalmente, con las 16. Compare visualmente los resultados.

El ruido que la función `imnoise` aplicó a cada una de las imágenes creadas fue distinto, tiene aleatoriedad. Si este hubiera sido el mismo hacer filtrado con la media de varias imágenes no tendría sentido.

```
len = size(blood);
% Filtro con 4 imágenes
blood_filtered_4 = uint8(zeros(len(1), len(2)));
for i = 1:4
    blood_filtered_4 = blood_filtered_4 +
imread(strcat(ruta,'blood_gaussian',num2str(i),'.bmp'))*1/4;
end
% Filtro con 8 imágenes
blood_filtered_8 = uint8(zeros(len(1), len(2)));
for i = 1:8
    blood_filtered_8 = blood_filtered_8 +
imread(strcat(ruta,'blood_gaussian',num2str(i),'.bmp'))*1/8;
end
% Filtro con 16 imágenes
blood_filtered_16 = uint8(zeros(len(1), len(2)));
for i = 1:16
    blood_filtered_16 =blood_filtered_16 +
imread(strcat(ruta,'blood_gaussian',num2str(i),'.bmp'))*1/16;
end
Podemos comprobar
```



Podemos que cuantas más imágenes ha sido utilizadas para hacer la media, mejores han sido los resultados obtenidos.

14. Utilizando la imagen P2gris.jpg, compuesta únicamente por un fondo grisáceo de luminancia intermedia, complete los siguientes pasos:

- cargue la imagen, conviértala a escala de grises y muestre su histograma,

```
P2gris = im2double(rgb2gray(imread(strcat(ruta,'P2gris.jpg'))));
subplot(4,2,1);imshow(P2gris);title('original')
subplot(4,2,2);imhist(P2gris);title('histograma original')
```

- contamine con ruido speckle la imagen de grises y almacene el resultado en otro fichero de imagen,

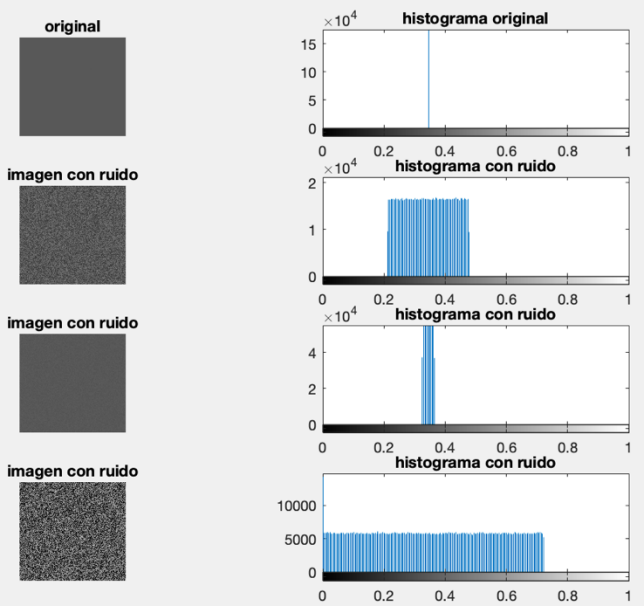
```
P2gris_speckle_004 = imnoise(P2gris,'speckle');
desviacion_de_la_media = mean(mean(P2gris))-mean(mean(P2gris_speckle_004));
display(strcat('La media ha cambiado en: ',num2str(desviacion_de_la_media),' al
añadir el ruido, V=0.04'))
subplot(4,2,4);imhist(P2gris_speckle_004);title('histograma con ruido')
subplot(4,2,5);imshow(P2gris_speckle_0001);title('imagen con ruido')
P2gris_speckle_0001 = imnoise(P2gris,'speckle', 0.001);
desviacion_de_la_media = mean(mean(P2gris))-mean(mean(P2gris_speckle_0001));
display(strcat('La media ha cambiado en: ',num2str(desviacion_de_la_media),' al
añadir el ruido, V=0.001'))
P2gris_speckle_04 = imnoise(P2gris,'speckle',0.4);
desviacion_de_la_media = mean(mean(P2gris))-mean(mean(P2gris_speckle_04));
display(strcat('La media ha cambiado en: ',num2str(desviacion_de_la_media),' al
añadir el ruido, V=0.4'))
```

- observe el histograma de la imagen ruidosa y confirme que el ruido tiene una distribución uniforme,

```
La media ha cambiado en:-0.00011263 al añadir el ruido, V=0.04
La media ha cambiado en:1.8194e-05 al añadir el ruido, V=0.001
La media ha cambiado en:-0.00051705 al añadir el ruido, V=0.4
```

- consulte la ayuda del comando imnoise, para ver cómo se puede cambiar la varianza del ruido, contamine la imagen original con ruido speckle con diferentes varianzas, y observe qué ocurre con los histogramas de las distintas imágenes ruidosas generadas.

```
help imnoise
```



La imagen original solo tiene un único color de gris. Al aplicar ruido hacemos que aparezcan más o menos colores de gris nuevos distribuidos de forma uniforme en el histograma. Esto lo vemos en el histograma pues aumenta el rango de colores de la imagen teniendo cada color nuevo aproximadamente la misma cantidad de valores que el resto de los colores.
Cuanto mayor es la varianza más colores nuevos aparecen.

15. Busque una imagen oscura y poco contrastada, o créela a partir de una fotografía (multiplicando por 0.1 o 0.2 sus niveles de gris). A continuación, visualice dicha imagen y su histograma. Finalmente, obtenga el logaritmo de dicha imagen, calculando previamente el valor óptimo de K, para asegurar que el nivel máximo de gris a la salida sea el máximo posible. Visualice la imagen resultante y observe la mejora obtenida. Pruebe utilizando valores de K diferentes al calculado.

Creamos nuestra imagen a partir de bajar el brillo a la imagen flowers.

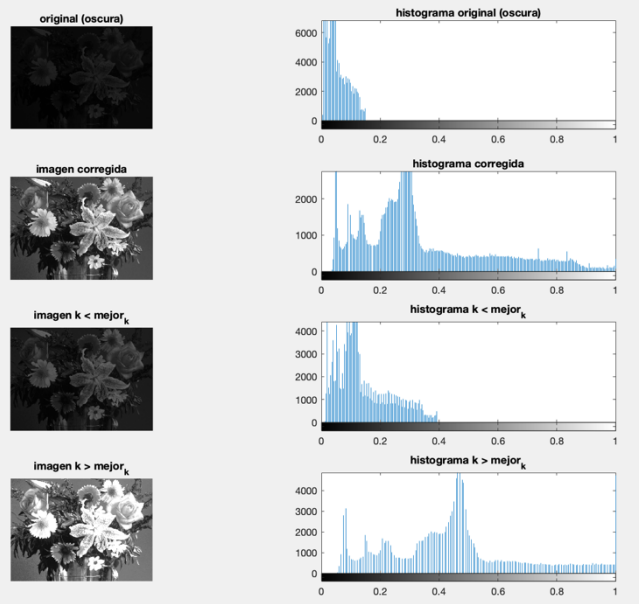
```
dark_flowers = im2double(rgb2gray(imread(strcat(ruta,'flowers.tif'))))*0.15;
```

Calculamos el factor k ideal que hará que nuestra imagen recupere su brillo originar expandiendo con un logaritmo su histograma sin llegar a saturarlo.

```
k=1/log10(1+max(dark_flowers,[],'all'));
bright_flowers=k*log10(1+dark_flowers);
```

Valores de k superiores e inferiores hará que nuestra imagen se siga viendo oscura o que se vea demasiado clara. En el caso de la demasiado clara el histograma se satura al llegar al brillo máximo.

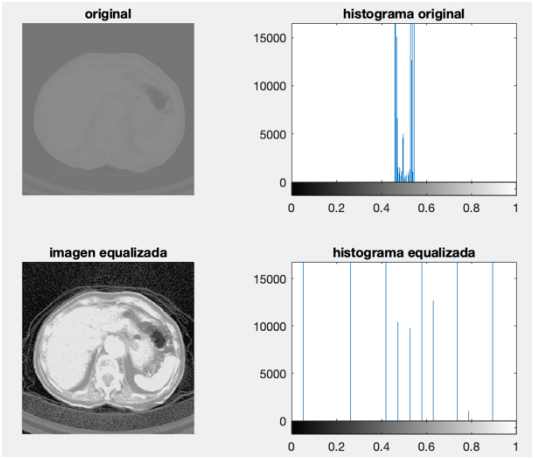
```
still_dark_flowers=(k-10)*log10(1+dark_flowers);
to_bright_flowers=(k+10)*log10(1+dark_flowers);
```



16. Cargue la imagen bazo.bmp y muestre su histograma. Compruebe que se trata de una imagen poco contrastada. Convierta a escala de grises la imagen y luego ecualice su histograma uniformemente. Visualice la imagen resultante y su histograma, contrastando las diferencias con la imagen original.

```
[image,map]=imread(strcat(ruta,'bazo.bmp'));
bazo = im2double(ind2gray(image,map));

contrast = max(bazo,[],'all')-min(bazo,[],'all');
display(strcat('El contraste de la imagen es: ', num2str(contrast),' (contraste máximo = 1)'))

bazo_eq=histeq(bazo,20);
contrast_eq = max(bazo_eq,[],'all')-min(bazo_eq,[],'all');
display(strcat('El contraste de la imagen equalizada es: ', num2str(contrast_eq),' (contraste mV°ximo = 1)'))
```

```
El contraste de la imagen es:0.086275 (contraste máximo = 1)
El contraste de la imagen equalizada es:0.84211 (contraste máximo = 1)
```



Al reducir el número de colores posibles en el histograma y repartir los existentes entre ellos vemos como se logra aumentar el contraste de las imágenes cuando este es muy bajo. Si la imagen ya tuviera un elevado contraste esto solo reduciría la calidad de la imagen simplificando su cantidad de colores.

17. Disminuya el brillo de la imagen bazo.bmp hasta que el nivel de gris más bajo sea 0 y sobre esa imagen realice una ecualización gamma. Pruebe con distintos valores de gamma y analice el efecto de cada uno sobre la imagen de salida.

Primero desplazamos el histograma de la imagen hasta que su menor valor esté en el 0 pero sin modificar el contraste.

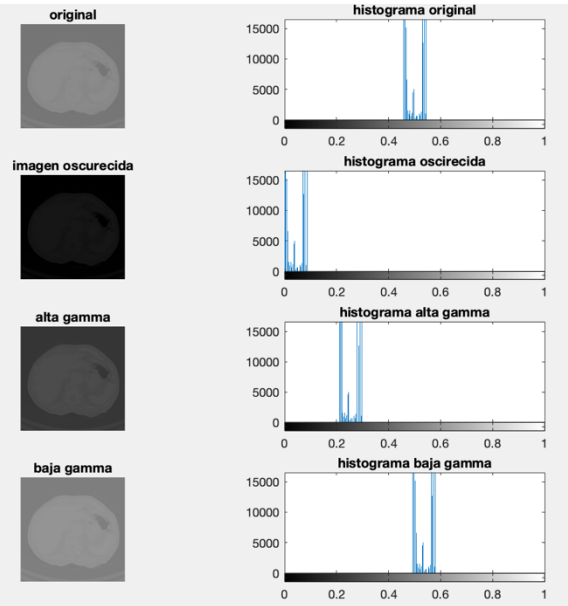
```
 [image,map]=imread(strcat(ruta,'bazo.bmp'));
 bazo = im2double(ind2gray(image,map));
 contrast = max(bazo,[],'all')-min(bazo,[],'all');
 display(strcat('El contraste de la imagen original es: ', num2str(contrast),'
(contraste máximo = 1)'))

 bazo_dark = imadjust(bazo,[min(bazo,[],'all') max(bazo,[],'all')], [0
(max(bazo,[],'all')-min(bazo,[],'all'))]);
 contrast = max(bazo_dark,[],'all')-min(bazo_dark,[],'all');
 display(strcat('El contraste de la imagen oscurecida es: ', num2str(contrast),'
(contraste máximo = 1)'))
```

Ahora aplicamos distintos valores de gamma observando que cuanto mayores sean más aumentarán el brillo de la imagen sin en ningún caso afectar al contraste de ésta.

```
 bazo_high_gamma = imadjust(bazo,[0 1], [0,1], 2);
 contrast = max(bazo_high_gamma,[],'all')-min(bazo_high_gamma,[],'all');
 display(strcat('El contraste de la imagen con gamma alto es: ',
num2str(contrast),' (contraste máximo = 1)'))
 % Una gamma baja genera una imagen clara, si fuera 0 la imagen es blanca
 bazo_low_gamma = imadjust(bazo,[0 1], [0,1], 0.9);
 contrast = max(bazo_low_gamma,[],'all')-min(bazo_low_gamma,[],'all');
```

```
 display(strcat('El contraste de la imagen con gamma bajo es: ',
num2str(contrast),' (contraste máximo = 1)'))
```

```
El contraste de la imagen original es:0.086275 (contraste máximo = 1)
El contraste de la imagen oscurecida es:0.086275 (contraste máximo = 1)
El contraste de la imagen con gamma alto es:0.086613 (contraste máximo = 1)
El contraste de la imagen con gamma bajo es:0.083199 (contraste máximo = 1)
```



18. Efectúe las siguientes operaciones, relacionadas con la mejora de imágenes:

18.1. Cargue y visualice la imagen ‘pout.tif’.

```
pout = imread(strcat(ruta,'pout.tif'));
pout = im2double(pout);
figure('Name', 'EJ18.1')
subplot(4,2,1);imshow(pout);title('original')
```

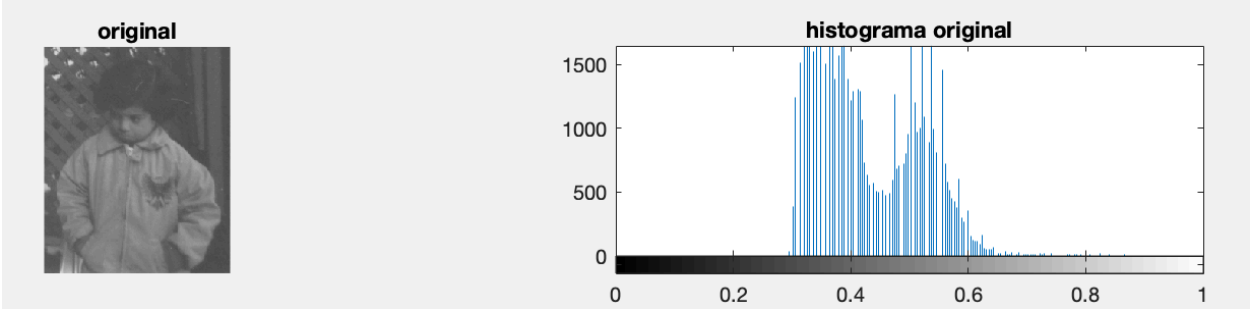
18.2. Compruebe los principales parámetros de la imagen con el comando imfinfo.

```
display(imfinfo(strcat(ruta,'pout.tif')))
```

Vemos que la imagen está en escala de grises con valores de 0 a 255.

18.3. Obtenga el histograma de la misma.

```
subplot(4,2,2);imhist(pout);title('histograma original')
```

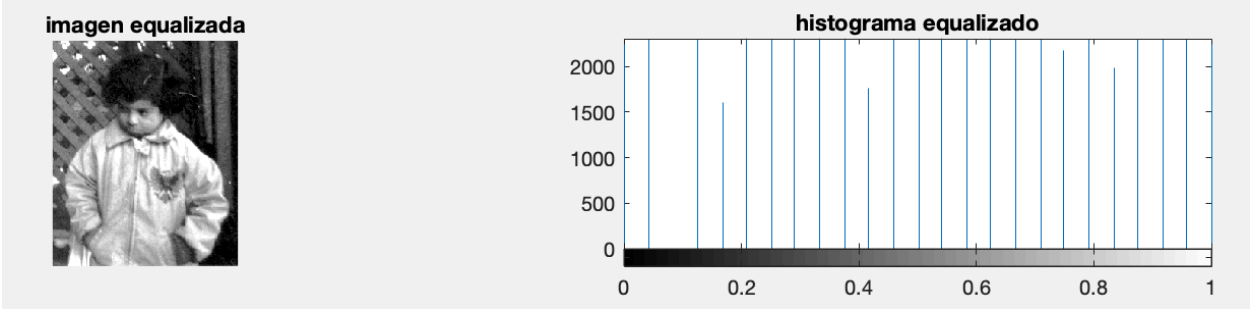


Vemos que la imagen tiene bajo contrate y que es ligeramente oscura.

18.4. Ecualice el histograma y muestre la imagen ecualizada.

```
El contraste de la imagen original es:0.58824 (contraste máximo = 1)
El contraste de la imagen equalizada es:1 (contraste máximo = 1)
```

```
pout_eq=histeq(pout,25);
subplot(4,2,3);imshow(pout_eq);title('imagen equalizada')
subplot(4,2,4);imhist(pout_eq);title('histograma equalizado')
contrast = max(pout,[],'all')-min(pout,[],'all');
display(strcat('El contraste de la imagen original es: ', num2str(contrast),'
(contraste máximo = 1)'))
contrast_eq = max(pout_eq,[],'all')-min(pout_eq,[],'all');
display(strcat('El contraste de la imagen equalizada es: ',
num2str(contrast_eq),' (contraste máximo = 1)'))
```



Reduciendo los posibles colores de la imagen aumentamos su contraste hasta el máximo posible, aunque perdemos detalles pues zonas con colores originalmente distintos ahora tienen el mismo. Una ventaja es que el brillo de la imagen se ha hecho más uniforme en el histograma.

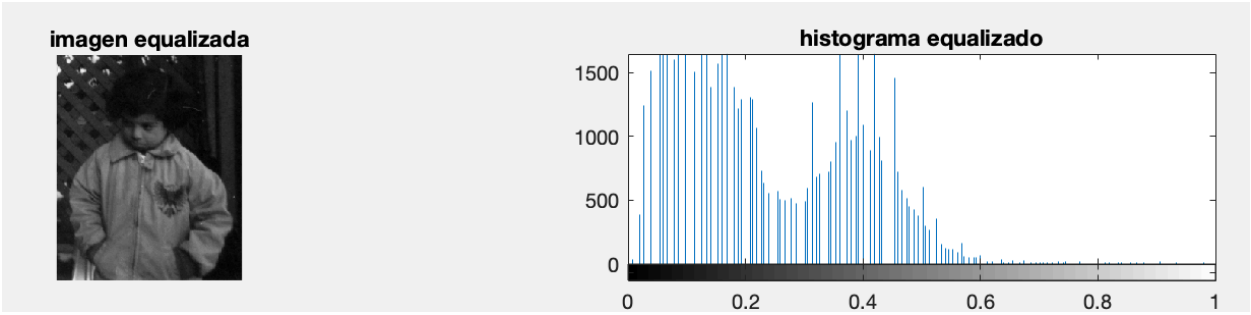
18.5. Almacene la imagen ecualizada en un fichero jpeg utilizando el comando imwrite.

```
imwrite(pout_eq, strcat(ruta,'pout_eq.tif'))
```

18.6. Sobre la imagen original: Optimice el contraste al máximo, y muestre el histograma de la imagen de salida y su FFT.

```
El contraste optimizado es:1 (contraste máximo = 1)
```

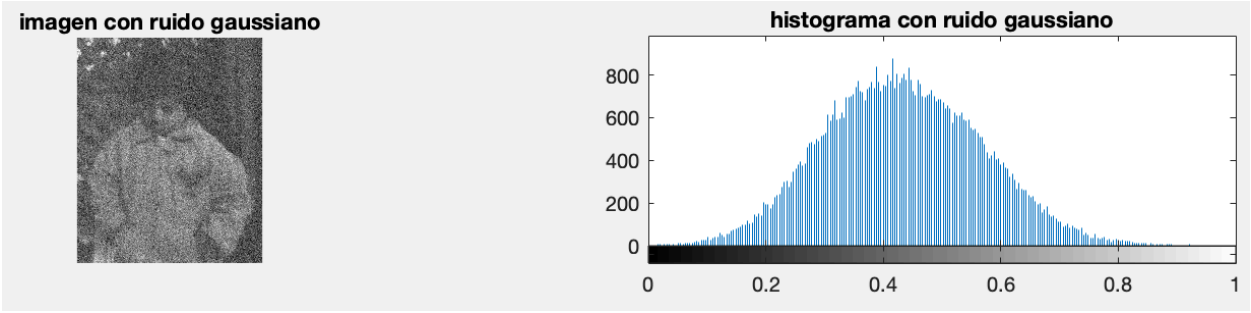
```
 pout_contrast = imadjust(pout,[min(pout, [], 'all') max(pout, [], 'all')],
[0,1]);
 subplot(4,2,5);imshow(pout_contrast);title('imagen equalizada')
 subplot(4,2,6);imhist(pout_contrast);title('histograma equalizado')
 contrast_c = max(pout_contrast,[],'all')-min(pout_contrast,[],'all');
 display(strcat('El contraste optimizado es: ', num2str(contrast_c),' (contraste
máximo = 1)'))
```



Hemos logrado que el contrate sea máximo igual que con la ecualización. No obstante, el resultado es una imagen oscurecida pues hay mucha variedad de grises claros con poca cantidad de ellos.

18.7. Añada ruido gaussiano, muestre el histograma de la imagen con ruido y obtenga su FFT.

```
 pout_gaussian = imnoise(pout,'gaussian');
 subplot(4,2,7);imshow(pout_gaussian);title('imagen con ruido gaussiano')
 subplot(4,2,8);imhist(pout_gaussian);title('histograma con ruido gaussiano')
```



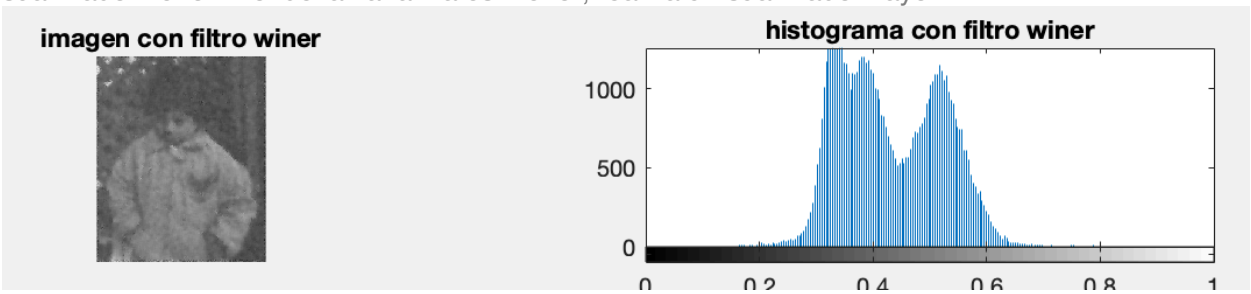
El histograma de la imagen ha adoptado la forma de una campana de gauss.

18.8. Aplique los siguientes filtros a la imagen con ruido y compare las imágenes de salida:

- un filtro Wiener en el dominio del espacio,

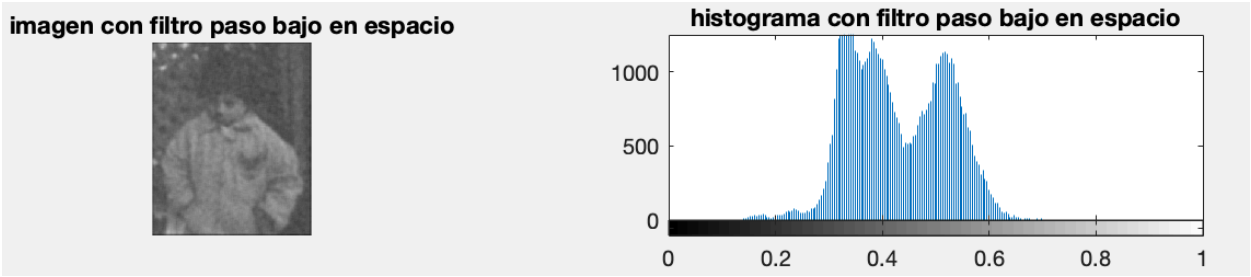
```
 winer_filter=wiener2(pout_gaussian,[6 6]);
 subplot(4,2,1);imshow(winer_filter);title('imagen con filtro winer')
 subplot(4,2,2);imhist(winer_filter);title('histograma con filtro winer')
```

Es un filtro adaptativo por lo que mantiene bordes y otras partes de alta frecuencia de una imagen. Se adapta a la varianza local de la imagen. Donde la varianza es mayor, el filtro Wiener realiza un suavizado menor. Donde la varianza es menor, realiza un suavizado mayor.

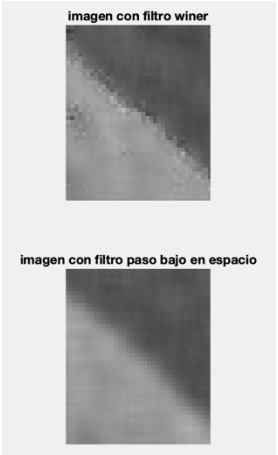


- un filtro paso bajo en el dominio del espacio,

```
 low_s_filter=filter2(fspecial('average', [6 6]),pout_gaussian);
 subplot(4,2,3);imshow(low_s_filter);title('imagen con filtro paso bajo en
espacio')
 subplot(4,2,4);imhist(low_s_filter);title('histograma con filtro paso bajo en
espacio')
```



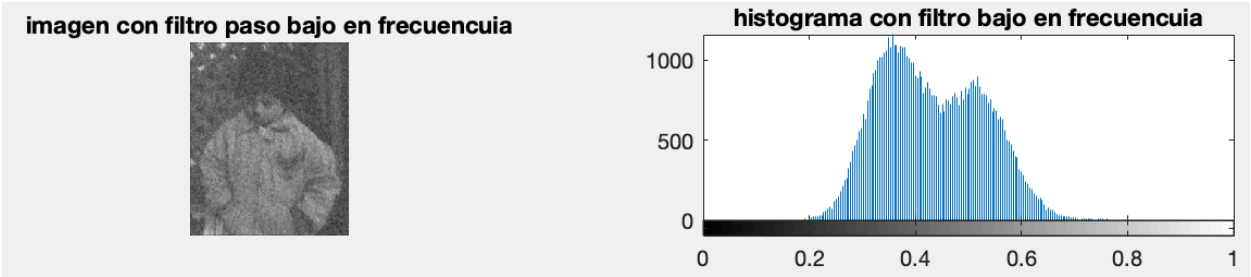
Hacemos un filtro no adaptativo obteniendo resultados similares al caso anterior a nivel de histograma.



No obstante, si hacemos zoom sobre las imágenes vemos como cada filtro ha actuado de forma muy distinta.
En una zona de alta de alta varianza como un borde vemos como el filtro winner ha producido bordes más afilados por su carácter adaptativo mientras que el filtro pasa bajo trata por igual todas las partes de la imagen.

- un filtro paso bajo en el dominio de la frecuencia, y

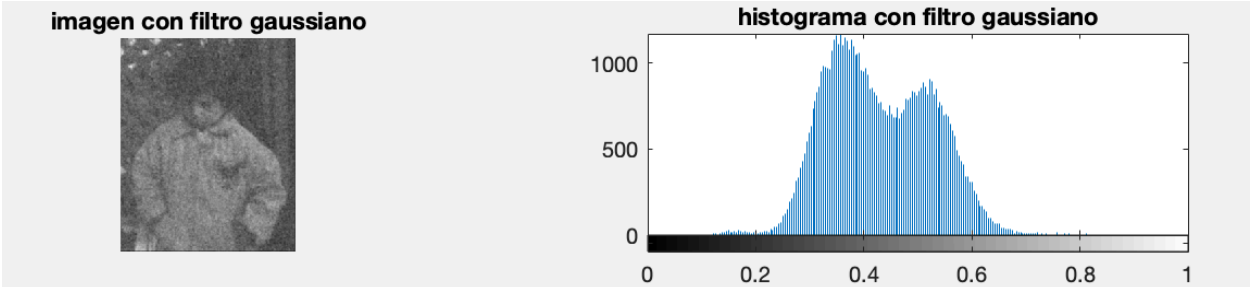
```
 [rows,cols]=size(pout_gaussian);
 N=(2*cols)-1;
 M=(2*rows)-1;
 [p,q]=meshgrid(1:N,1:M);
 D=sqrt((p-floor((N/2)+1)).^2+(q-floor((M/2)+1)).^2);
 k=1/(2*((0.2*M)^2));
 H=exp(-k.*(D.^2));
 NIM=fft2(pout_gaussian,M,N).*fftshift(H);
 low_f_filter=real(ifft2(NIM));
 low_f_filter=low_f_filter(1:rows,1:1:cols);
 subplot(4,2,5);imshow(low_f_filter);title('imagen con filtro paso bajo en
frecuencuia')
 subplot(4,2,6);imhist(low_f_filter);title('histograma con filtro bajo en
frecuencuia')
```



Vemos como este filtro deja pasar los bordes donde la frecuencia es mayor afectando solo a aquellas zonas donde el color era uniforme. Debido a la naturaleza del ruido originalmente aplicado obtenemos un efecto relativamente similar al ruido salt&pepper sobre la imagen resultado.
Cuanto menor sea el valor de k la imagen será corregida en menor medida, con un valor de k 10 veces más altos los resultados obtenidos son muy similares al filtro pasa bajos en el espacio.

- un filtro gaussiano. Genérelo con fspecial y pruebe con diferentes valores de los parámetros.

```
gaussian_filter=filter2(fspecial('gaussian', [6 6],0.8),pout_gaussian);
subplot(4,2,7);imshow(gaussian_filter);title('imagen con filtro gaussiano')
subplot(4,2,8);imhist(gaussian_filter);title('histograma con filtro gaussiano')
```



De modo similar cuanto mayor es el kernell de filtro gaussiano o mayor es la desviación estándar más se parecerá el resultado a los obtenidos con los otros filtros y menos a la imagen con ruido.
Si estos son demasiado grandes la imagen se verá demasiado difuminada.

Se observa que el filtro que mejores resultados proporciona con menor preocupación es el de wiener por el hecho de ser adaptativo.