Memoria

Laboratorio de Procesamiento de Imágenes

Antonio J. Galán Herrera

Este documento recoge los **ejercicios** de todas las prácticas que fueron **realizados de forma diferente** a los ejemplos incluidos en los enunciados tutorizados (parámetros, imágenes alternativas...) o que **fueron solicitados en las entregas** de sus correspondientes prácticas.

Se indicará qué ejercicios fueron solicitados y con qué variación en cada apartado del documento.

Funciones de MATLAB empleadas

`bwselect(BW,c,r,n)`	Crea una imagen binaria que contiene los objetos que se superponen al píxel (r, c), donde n especifica la conectividad. Los objetos son conjuntos conectados de píxeles, es decir, píxeles que tienen un valor de 1. Por defecto, bwselect busca objetos con 4 conexiones.
`imhist(I)`	Calcula el histograma para la imagen de intensidad I y muestra un gráfico del histograma. El número de ubicaciones en el histograma está determinado por el tipo de imagen.
`imadjust(I,[low_in high_in],[low_out high_out])`	Correlaciona los valores de intensidad en I con los nuevos valores en J de manera que los valores entre low_in y high_in se correlacionan con valores entre low_out y high_out. Puede omitir el argumento [low_out high_out], en cuyo caso, imadjust usa el valor predeterminado [0 1].
`histeq(I)`	Devuelve la transformación T de escala de grises que asigna niveles de gris en la imagen I a niveles de gris en J.
`plot(T,FIG)`	Traza el árbol T en la figura cuyo mango es FIG. Esta figura ya fue utilizada para trazar un árbol,
`imnoise(I,'gaussian',m)`	Agrega ruido blanco Gaussiano con media m y varianza de 0.01.
`medfilt2(A,[m n])`	Realiza una mediana de filtrado, donde cada píxel de salida contiene el valor mediano en la vecindad m-by-n alrededor del píxel correspondiente en la imagen de entrada.
`fspecial('average',hsize)`	Devuelve un filtro promedio h de tamaño hsize.
`filter2(H,X)`	Aplica un filtro de respuesta de impulso finito a una matriz de datos X según los coeficientes en una matriz H.
`edge(I,method,threshold)`	Devuelve todos los bordes que son más fuertes que el umbral.
`Abs`	Devuelve el valor absoluto.

`max(A)`	Una matriz de tipo de dominio DOM_HFARRAY con entradas reales
`min(A)`	Una matriz de tipo de dominio DOM_HFARRAY con entradas reales
`colorbar`	Muestra una barra de colores vertical a la derecha de los ejes o cuadros actuales. Las barras de colores muestran el mapa de colores actual e indican el mapeo de los valores de datos en el mapa de colores.
`fft2(X)`	Devuelve la transformada de Fourier bidimensional de una matriz utilizando un algoritmo rápido de transformación de Fourier, que es equivalente a calcular fft (fft (X). ').'. Si X es una matriz multidimensional, entonces fft2 toma la transformación 2-D de cada dimensión más alta que 2. La salida Y es del mismo tamaño que X.
`log(x)`	Alias para el logaritmo natural ln (x).
`freqz2(___)`	Produce un gráfico de malla de la respuesta de frecuencia de magnitud bidimensional cuando no se especifican argumentos de salida.
`fftshiftX`	Reorganiza una transformada de Fourier X desplazando el componente de frecuencia cero al centro de la matriz.
`colormap`	Mapa de color establece el mapa de color para la figura actual en uno de los mapas de color predefinidos. Si configura el mapa de colores para la figura, los ejes y cuadros en la figura usan el mismo mapa de colores. El nuevo mapa de colores tiene la misma longitud (número de colores) que el mapa de colores actual. Cuando usa esta sintaxis, no puede especificar una longitud personalizada para el mapa de colores.
`colorbar(target)`	Agrega una barra de color a los ejes o al gráfico especificado por el objetivo.
`phantom(___)`	Devuelve la matriz E utilizada para generar el fantasma
`freqspace([m n])`	Devuelve los vectores de frecuencia bidimensional f1 y f2 para una matriz m-by-n.

`meshz(X,Y,Z)`	Dibuja una cortina alrededor de la malla de alambre con un color determinado por Z, por lo que el color es proporcional a la altura de la superficie. Si X e Y son vectores, longitud (X) = n y longitud (Y) = m, donde [m, n] = tamaño (Z). En este caso, (X (j), Y (i), Z (i, j)) son las intersecciones de las líneas de la malla de alambre; X e Y corresponden a las columnas y filas de Z, respectivamente. Si X e Y son matrices, (X (i, j), Y (i, j), Z (i, j)) son las intersecciones de las líneas de la cuadrícula de alambre.
`hamming(L)`	Devuelve una ventana Hamming simétrica de L-point.
`fwind1(Hd,win)`	Usa una especificación de ventana 1-D para diseñar un filtro h FIR bidimensional basado en la respuesta de frecuencia deseada Hd. fwind1 devuelve h como una molécula computacional, que es la forma apropiada de usar con filter2. fwind1 usa la ganancia de ventana unidimensional para formar una ventana bidimensional simétrica circularmente simétrica utilizando el método de Huang. Funciona solo con ventanas 1-D;usar fwind2 para trabajar con ventanas bidimensionales.
`fsamp2(Hd)`	Diseña un filtro FIR bidimensional con respuesta de frecuencia Hd y devuelve los coeficientes de filtro en la matriz h. El filtro h tiene una respuesta de frecuencia que pasa por puntos en Hd. Diseña filtros FIR bidimensionales basados en una respuesta de frecuencia bidimensional deseada muestreada en puntos en el plano cartesiano.
`axis(limits)`	Especifica los límites para los ejes actuales. Especifique los límites como vector de cuatro, seis u ocho elementos.
`real(z)`	Devuelve la parte real de z. Si z es una matriz, los actos reales son por elementos sobre z.
`ifft2(Y)`	DWevuelve la transformada de Fourier inversa discreta bidimensional de una matriz utilizando un algoritmo rápido de transformación de Fourier. Si Y es una matriz multidimensional, ifft2 toma la transformación inversa 2-D de cada dimensión más alta que 2. La salida X tiene el mismo tamaño que Y.

`pause`	Pausa la cola del planificador de trabajos de MATLAB para que los trabajos que esperan en el estado en cola no se ejecuten. Los trabajos que ya se están ejecutando también se detienen después de completar las tareas que ya se están ejecutando. No se ejecutarán más trabajos o tareas hasta que se solicite la función de reanudación para el MJS. La función de pausa no hace nada si el MJS ya está en pausa.
`mesh(X,Y,Z)`	Dibuja una malla de alambre con un color determinado por Z, por lo que el color es proporcional a la altura de la superficie. Si X e Y son vectores, longitud (X) = n y longitud (Y) = m, donde [m, n] = tamaño (Z). En este caso, (X (j), Y (i), Z (i, j)) son las intersecciones de las líneas de la malla de alambre; X e Y corresponden a las columnas y filas de Z, respectivamente. Si X e Y son matrices, (X (i, j), Y (i, j), Z (i, j)) son las intersecciones de las líneas de la cuadrícula de alambre. Los valores en X, Y o Z pueden ser valores numéricos, de fecha y hora, de duración o categóricos.
`imfilter(___,options,...`	Realiza el filtrado multidimensional de acuerdo con las opciones especificadas.
`deconvwnr (I,psf,nsr)`	Deconvierte la imagen I, donde ncorr es la función de autocorrelación del ruido e icorr es la función de autocorrelación de la imagen original.
`sum(A)`	Devuelve la suma a lo largo de diferentes dimensiones de la matriz A. Si A es un vector, suma (A) devuelve la suma de los elementos. Si A es una matriz, la suma (A) trata las columnas de A como vectores, devolviendo un vector de fila de las sumas de cada columna. Si A es una matriz multidimensional, la suma (A) trata los valores a lo largo de la primera dimensión no única como vectores, devolviendo una matriz de vectores de fila.

`imadd (X,Y)`	Agrega cada elemento de la matriz X con el elemento correspondiente en la matriz Y y devuelve la suma en el elemento correspondiente de la matriz de salida Z. Si X es una matriz entera, los elementos en la salida que exceden el rango del tipo entero se truncarán, y los valores fraccionarios se redondearán.
`Deconvblind(I,psfi)`	Deconvierte la imagen I utilizando el algoritmo de máxima verosimilitud y una estimación inicial de la función de dispersión de puntos (PSF), psfi. La función desconfundir devuelve tanto la imagen deblurred J como una PSF restaurada, psfr. Para mejorar la restauración, desconvblind admite varios parámetros opcionales, que se describen a continuación. Utilice [] como marcador de posición si no especifica un parámetro intermedio.
`padarray(A,padsize,padval)`	Rellena la matriz A donde padval especifica un valor constante para usar para elementos rellenos o un método para replicar elementos de la matriz.
`imcomplement(I)`	Calcula el complemento de la imagen I. En el complemento de una imagen binaria, los ceros se vuelven unos y los unos se vuelven ceros; blanco y negro están invertidos. En el complemento de una imagen de intensidad o RGB, cada valor de píxel se resta del valor de píxel máximo admitido por la clase (o 1,0 para imágenes de precisión doble) y la diferencia se utiliza como el valor de píxel en la imagen de salida. En la imagen de salida, las áreas oscuras se vuelven más claras y las áreas claras se oscurecen.
`graythresh(I)`	Calcula un umbral global, nivel, que se puede usar para convertir una imagen de intensidad en una imagen binaria con imbinarize. La función graythresh usa el método de Otsu, que elige el umbral para minimizar la variación intraclase de los píxeles en blanco y negro [1].
`imclose(I,SE)`	Realiza el cierre morfológico en la escala de grises o la imagen binaria I, devolviendo la imagen cerrada, J. SE es un elemento de elemento de estructura único devuelto por las funciones strel o offsetstrel. La operación de cierre morfológico es una dilatación seguida de una erosión, utilizando el mismo elemento estructurante para ambas operaciones.

`imclose(I,nhood)`	Cierra la imagen I, donde nhood es una matriz de 0s y 1s que especifica el vecindario del elemento estructurador. Esto es equivalente a la sintaxis imclose (I, strel (nhood)). La función imclose determina el elemento central del vecindario por piso ((tamaño (nhood) +1) / 2).
`imopen(I,SE)`	Realiza la apertura morfológica en la escala de grises o imagen binaria I, devolviendo la imagen abierta, J. SE es un elemento de elemento de estructura único devuelto por las funciones strel o offsetstrel. La operación abierta morfológica es una erosión seguida de una dilatación, utilizando el mismo elemento estructurador para ambas operaciones.
`imopen(I,nhood)`	Abre la imagen I, donde nhood es una matriz de 0 y 1 que especifica el vecindario del elemento estructurador. Esto es equivalente a la sintaxis de abrir (I, strel (nhood)). La función de apertura determina el elemento central del vecindario por piso ((tamaño (nhood) +1) / 2).
`bwmorph(BW,operation,n)`	Aplica la operación n veces. n puede ser Inf, en cuyo caso la operación se repite hasta que la imagen ya no cambie.
`bwhitmiss(BW,SE1,SE2)`	Realiza la operación de error definido por los elementos estructurantes SE1 y SE2. La operación de error conserva píxeles en la imagen binaria BW cuyos vecindarios coinciden con la forma de SE1 y no coinciden con la forma de SE2. Esta sintaxis es equivalente a imerode (BW, SE1) e imerode (~ BW, SE2).
`imfill(BW,locations)`	Realiza una operación de llenado de inundación en los píxeles de fondo de la imagen binaria de entrada BW, a partir de los puntos especificados en las ubicaciones. Si ubicaciones es un vector p-by-1, contiene los índices lineales de las ubicaciones de inicio. Si ubicaciones es una matriz p-by-ndims (BW), cada fila contiene los índices de matriz de una de las ubicaciones de inicio.
`imfill(BW,'holes')`	Rellena los agujeros en la imagen binaria de entrada BW. En esta sintaxis, un agujero es un conjunto de píxeles de fondo que no se puede alcanzar rellenando el fondo desde el borde de la imagen.
`imfill(I,conn)`	Rellena los agujeros en la imagen en escala de grises I, donde conn especifica la conectividad.
`imclearborder(IM,conn)`	Especifica la conectividad deseada.

`bwperim (BW)`	Devuelve una imagen binaria que contiene solo los píxeles del perímetro de los objetos en la imagen de entrada BW. Un píxel es parte del perímetro si no es cero y está conectado a al menos un píxel de valor cero. La conectividad predeterminada es 4 para dos dimensiones, 6 para tres dimensiones y conndef (ndims (BW), 'mínimo') para mayores dimensiones. Si no especifica un valor de retorno, bwperim muestra el resultado en una ventana de figura.
`blockproc(A,[m n],fun)`	Procesa la imagen A aplicando la función fun a cada bloque de tamaño distinto [m n] y concatenando los resultados en la matriz de salida, B.
`blockproc(src_filename,[m n],fun)`	Procesa la imagen con el nombre de archivo src_filename, leyendo y procesando un bloque a la vez. Esta sintaxis es útil para procesar imágenes grandes.
`blockproc(adapter,[m n],fun)`	Procesa la imagen de origen especificada por el adaptador, un objeto ImageAdapter.
`blockproc(___,Name,Value)`	Procesa la imagen de entrada, especificando argumentos y valores correspondientes que controlan varios aspectos del comportamiento del bloque. Los nombres de argumento son insensibles a las mayúsculas y minúsculas.
`imtophat(I,SE)`	Realiza el filtro morfológico de sombrero de copa en la escala de grises o imagen binaria I, devolviendo la imagen filtrada, J. El filtro de sombrero calcula la apertura morfológica de la imagen (usando imopen) y luego resta el resultado del original imagen. SE es un objeto de elemento de estructura único devuelto por las funciones strel o offsetstrel.
`imtophat(I,nhood)`	Sombrero de copa filtra la imagen I, donde nhood es una matriz de 0s y 1s que especifica el vecindario del elemento estructurador. Esto es equivalente a la sintaxis imtophat (I, strel (nhood)). La función imtophat determina el elemento central del vecindario por piso ((tamaño (nhood) +1) / 2).
`bwlabel(BW,conn)`	Devuelve una matriz de etiqueta, donde conn especifica la conectividad.

`label2rgb(L)`	Convierte una matriz de etiqueta, L, como las devueltas por labelmatrix, bwlabel, bwlabeln o watershed, en una imagen de color RGB con el fin de visualizar las regiones etiquetadas. La función label2rgb determina el color que se asignará a cada objeto en función de la cantidad de objetos en la matriz de etiquetas. La función label2rgb selecciona colores de todo el rango del mapa de colores.

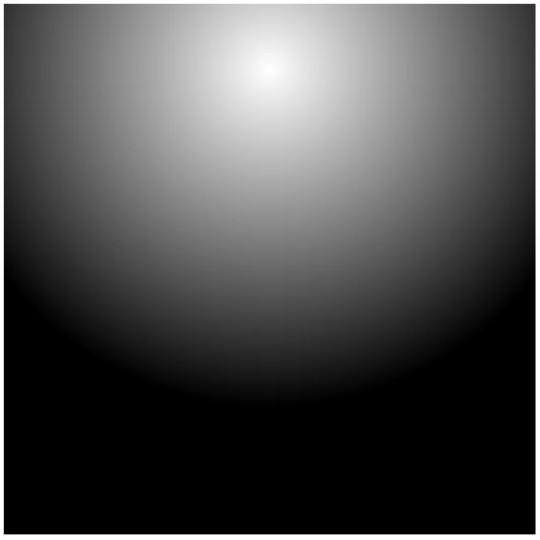
Práctica 1

- **Todos** los ejercicios.
- **Parámetros** diferentes para las funciones.
- **Imágenes** diferentes a las del enunciado.

1 - Representación de imágenes digitales monocromáticas

Apartado a)

```
% Coordenadas del foco (a,b)
a = 50;
b = 200;

% Construcción de la imagen M x N
M = 400;
N = 400;

for x = 1:M
for y = 1:N
I(x,y) = (255-sqrt((x-a)^2+(y-b)^2)) / 255;
end
end

% Mostrar la imagen
imshow(I)
```



Apartado b)

```
% Imagen de tamaño 30x30
I = ones(30,30);
I(13:17,5:24) = 0;

imshow(I, 'initialMagnification','fit')
```

Apartado c)

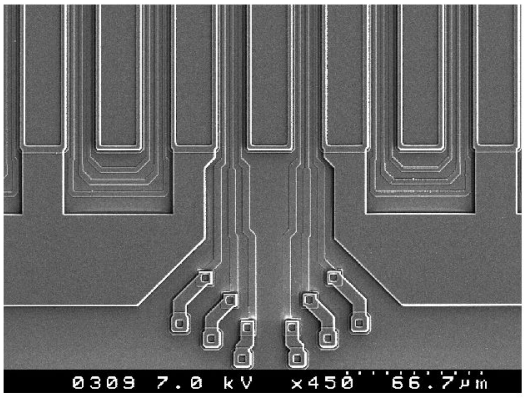
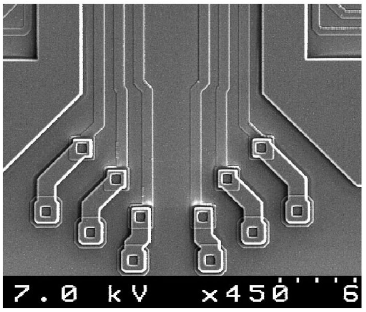


Imagen empleada

Versión 1

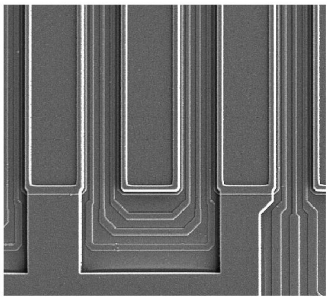
```
% Como la imagen ic.tif no está
% en la carpeta de imágenes para
% las prácticas, usé WATER.TIF.
I = imread('WAFER1.TIF');

imshow(I)

J = imcrop;

% Seleccionar con el ratón
% la región de interés
imshow(J)
```



Recorte 1: mediante el cursor (seleccionar área)

Versión 2

```
% Como la imagen ic.tif no está
% en la carpeta de imágenes para
% las prácticas, usé WATER.TIF.

% También se puede hacer fijando
% [Xmin, Ymin, ancho y alto]
I = imread('WAFER1.TIF');

I1 = imcrop(I, [0 0 500 450]);

imshow(I)

figure, imshow(I1)
```



Recorte 2: mediante vector, en `imcrop()`

2 - Operaciones aritméticas y geométricas con imágenes

Aunque la imagen se llame igual, no es la indicada en el enunciado. Esta imagen al contrario que las demás, no es monocromática.

Apartado a)

```
I = imread('Galaxia.jpg');

imshow(I)

% Multiplicar la imagen por 1,3
J = I * 1.3;

% Tamaño de la imagen (matriz)
[M,N] = size(J);

% Sumar 50 a cada píxel
for x = 1:M
      for y = 1:N
      J(x,y) = J(x,y) + 50;
      end
end

% Mostrar la imagen
imshow(J)
```

Comparada con la imagen original, esta presenta una mayor saturación y brillo.

Original

Modificada

Apartado b)

```
I = imread('Galaxia.jpg');

imshow(I);

zoom on;
```

El zoom (y el recorte) se realiza automáticamente al crear una selección con el cursor.

Apartado c)

Versión 1

```
I = imread('Galaxia.jpg');

J = imresize(I, 0.1, 'bilinear');
J = imresize(I, [100 50]); % Reduce la imagen a 100x50:
                           % 50 filas y 100 columnas

figure, imshow(J)

imshow(J)
```

Apartado d)

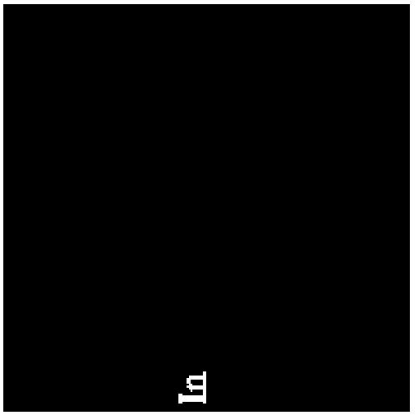
```
I = imread('Galaxia.jpg');
J = imresize(I, 0.5, 'bilinear');

IR = imrotate(J, 75);

imshow(IR);
```

3 - Conectividad de píxeles

Durante este ejercicio usé la misma imagen que la del enunciado tutorizado, debido a que ya se especifican unas coordenadas cuyo resultado muestra una conectividad entre píxeles.

```
I = imread('text.tif');

imshow(I)

% Arrays para los valores de
% las columnas y las filas
c = [90, 16];
r = [197, 67];

J = bwselect(I,c,r,8);

figure, imshow(J)
```

Original



Objetos encontrados

Práctica 2

- **Solo** el ejercicio 5.
- **Parámetros** diferentes para las funciones.
- **Imágenes** diferentes a las del enunciado.

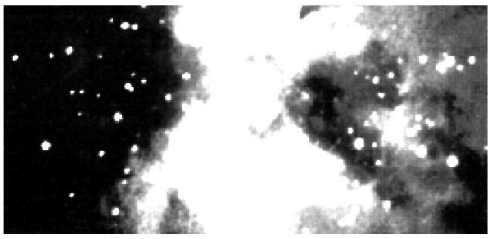
5 - Transformaciones puntuales

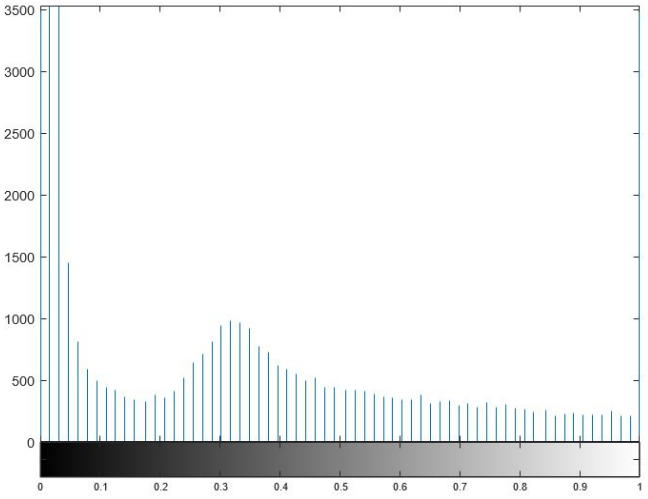
Apartado a)

En este apartado se ha usado la imagen que se usan en los demás; no obstante, el primer apartado se hacía con una imagen distinta llamada *spine.tif*.

Versión 1

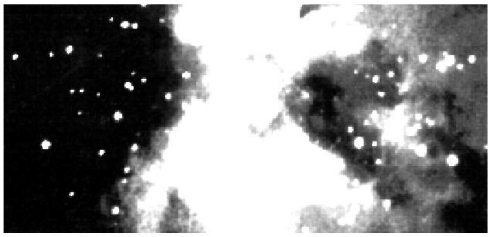
```
I = imread('ngc4024m.tif');

figure, imshow(I);

I1 = im2double(I);
I2 = I1;

for n = 1:368
    for m = 1:174
        I2(m,n) = I1(m,n) * 255/63;
        if I2(m,n) > 1
            I2(m,n) = 1;
        end
    end
end

figure, imshow(I2);
figure, imhist(I2)
```

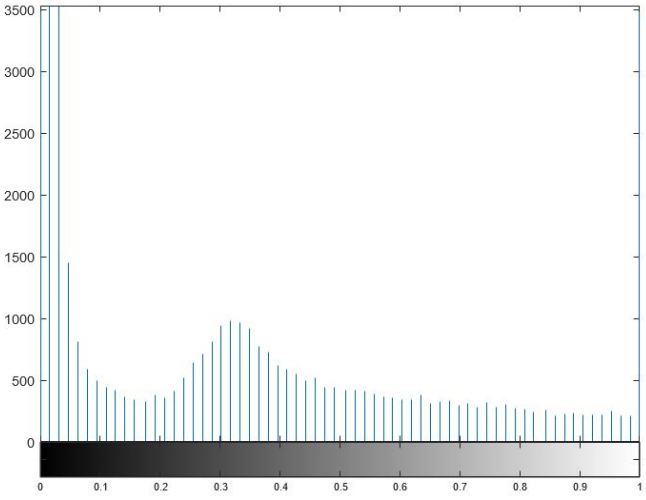




Versión 2

```
I = imread('ngc4024m.tif');

imshow(I);

J = imadjust(I,[0 0.247],[]);

figure, imshow(J)
figure, imhist(J)
```

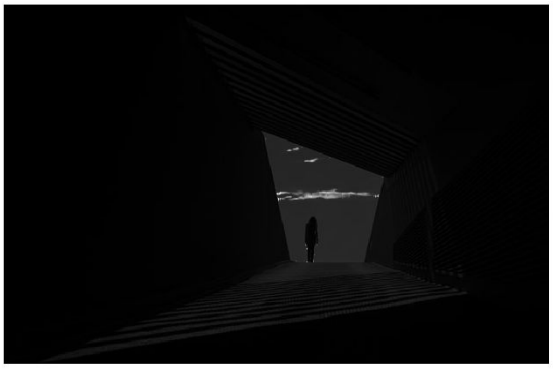


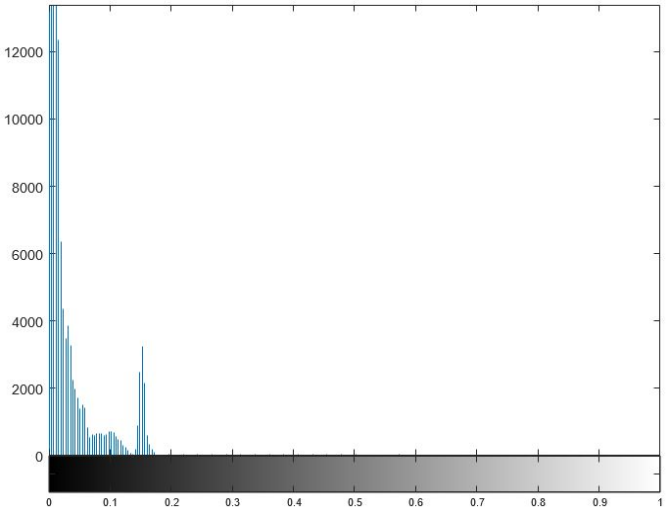


Como se puede apreciar, la segunda versión ofrece los mismos resultados en menos líneas de código -se podría decir que es más eficiente- gracias a la función `imadjust()`.

Apartado b)

```
I = imread('arc25.jpg');

figure, imshow(I);

I1 = im2double(I); % Rango de la imagen en [0,1]

a = 0.2;
c = 220/255;

for n = 1:580
for m = 1:379
if I1(m,n) < c
I2(m,n) = I1(m,n)*a;
else
I2(m,n) = a*c + (I1(m,n)-c)*(1-a*c)/(1-c);
end
end
end

figure, imshow(I2)
figure, imhist(I2)
```





Apartado c)

```
I = imread('arc25.jpg');

I1 = im2double(I); % Rango de la imagen en [0,1]
I2 = I1;

a = 0.5;

c = 0.4;
d = 0.6;

for n = 1:580
    for m = 1:379
        if (c < I1(m,n)) || (I1(m,n) < d)
            I2(m,n) = I1(m,n)*a;
        else
            I2(m,n) = a*c + (I1(m,n)-c)*(1-a*c)/(1-c);
        end
    end
end

figure, imshow(I);
figure, imshow(I2);
```

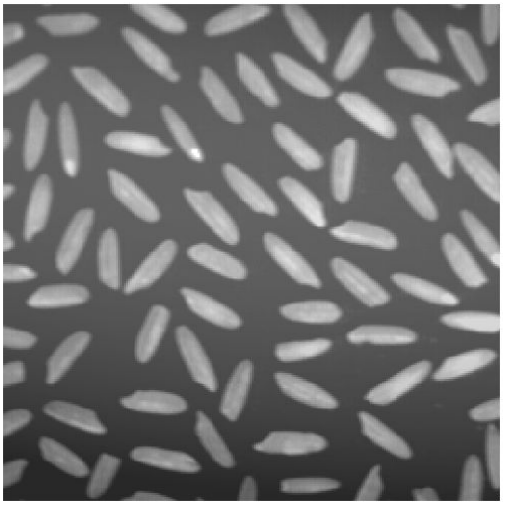
Apartado d)

```
I = imread('arc25.jpg');

I1 = im2double(I); % Rango de la imagen en [0,1]

[fil,col] = size(I);

a = 0.8;
c = 220/255;
col = col/3;

for n = 1:col
    for m = 1:fil
        if I1(m,n) < c
            I2(m,n) = I1(m,n)^2 * a;
        else
            I2(m,n) = a*c + (I1(m,n)^2-c)*(1-a*c)/(1-c);
        end
    end
end

figure, imshow(I);
figure, imshow(I2);
```

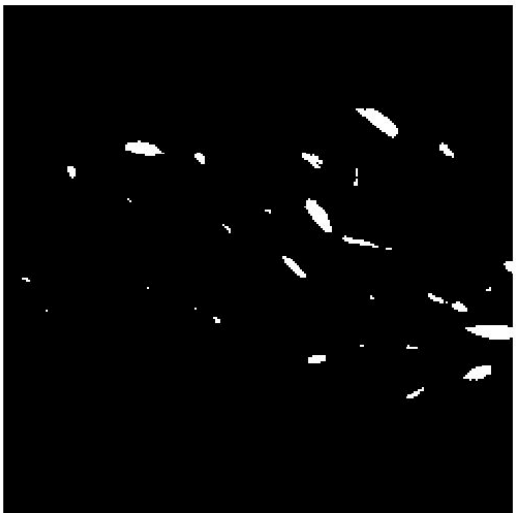
Apartado e)

En este apartado decidí cambiar de imagen respecto a la de los apartados anteriores, porque pensé que esta sería más representativa.

```
I = imread('rice.tif');

I1 = im2double(I); % Rango de la imagen en [0,1]
I2 = I1;

[fil,col] = size(I);

x = 0.8;

for n = 1:col
    for m = 1:fil
        if I1(m,n) < x
            I2(m,n) = 0;
        else
            I2(m,n) = 1;
        end
    end
end

figure, imshow(I);
figure, imshow(I2);
```



Original



Binaria

Práctica 3

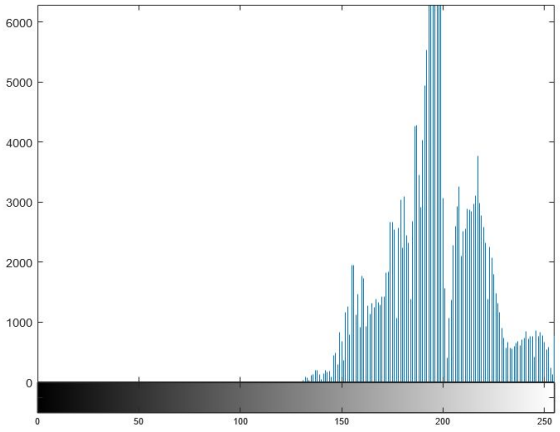
- **Todos** los ejercicios.
- **Parámetros** diferentes para las funciones.
- **Imágenes** diferentes a las del enunciado.

6 - Igualación del histograma

Apartado a)

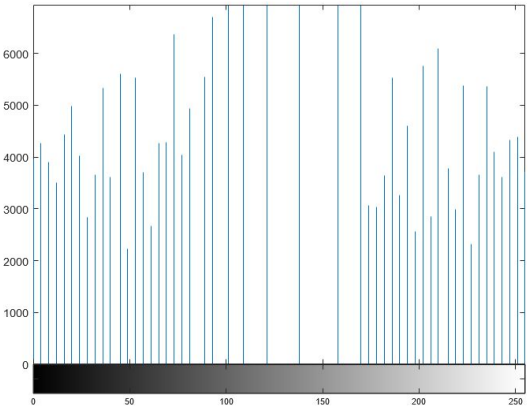
```
I = imread('motion01.512.tiff');

figure, imshow(I);

figure, imhist(I)
figure, histeq(I)
figure, imhist(histeq(I))
```

Original



Histograma de la original

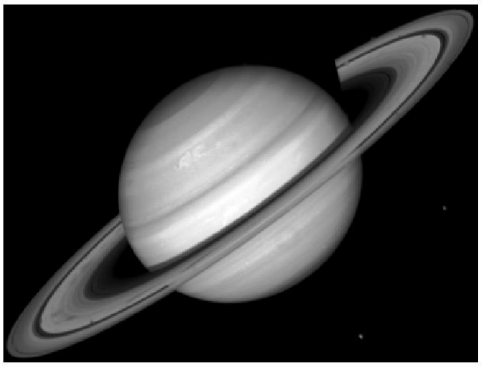
Igualación de Histograma



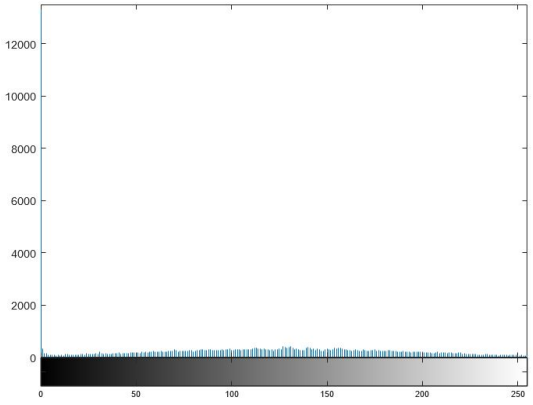
Histograma de la igualación

Apartado b)

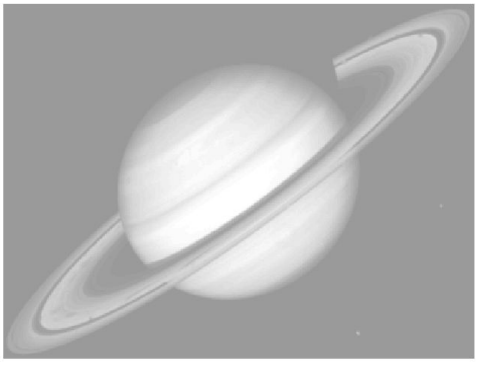
```
I = imread('saturno.tif');

imshow(I);

figure, imhist(I)
figure, histeq(I)
figure, imhist(histeq(I))

[J,T] = histeq(I);
plot((0:255)/255, T);
```



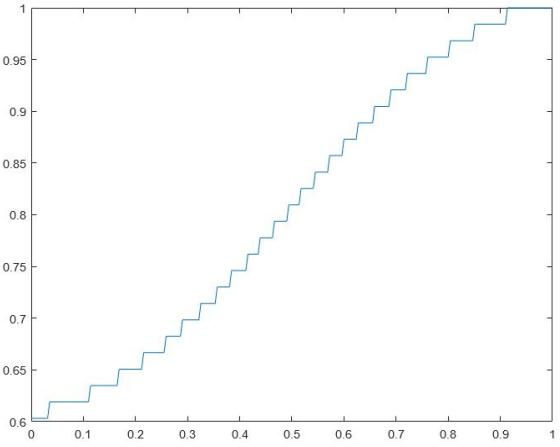
Original



Histograma de la original



Igualación del Histograma



Transformación realizada

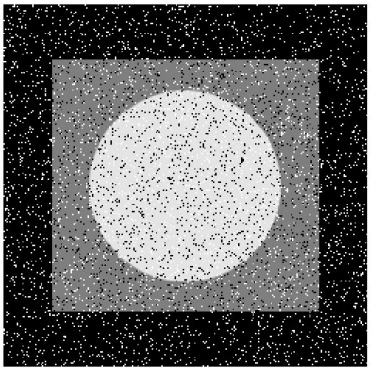
Como se puede observar, la imagen no ha perdido tanta calidad como hizo la imagen *moon.tif* en el ejemplo del enunciado. Esto puede deberse a que la imagen actual, posea un fondo más uniforme, lo que produce menos ruido en la transformación.

Comparando las gráficas de la transformación entre *moon.tif* y *saturno.tif* se puede observar que: la segunda presenta un crecimiento escalonado más uniforme que la primera, que crece muchísimo para los valores iniciales y lentamente para los siguientes.

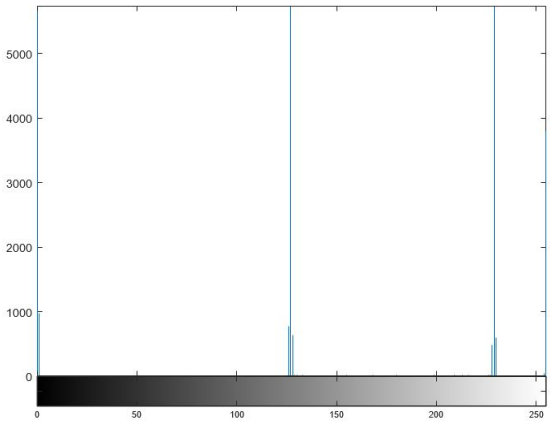
7 - Restauración mediante filtrado en el dominio espacial

Apartado a)

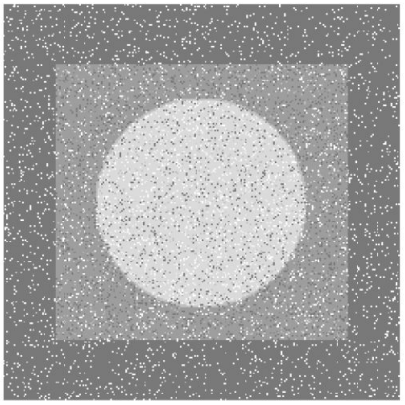
```
I = imread('Fig5.04(i).jpg');


% Añadir ruido "Sal y Pimienta"
I = imnoise(I, 'salt & pepper', 0.02);

figure, imshow(I); % Mucho ruido
figure, imhist(I);


% Igualar el histograma
[J,T] = histeq(I);
plot((0:255)/255, T); % Transformación (igualación)

figure, imshow(J); % Un poco de ruido
figure, imhist(J);


% Eliminar el ruido: filtro mediana (3x3)
F3 = medfilt2(J,[3 3]);

figure, imshow(F3); % Ruido filtrado
figure, imhist(F3);


% Eliminar el ruido: filtro mediana
g = fspecial('average', [5 5]);

M = filter2(g,J)/255;

figure, imshow(M);
figure, imhist(M);
```



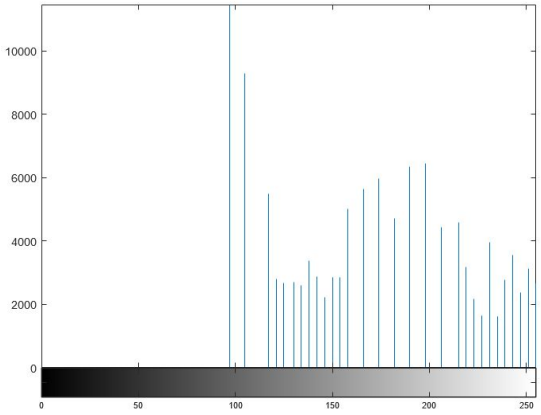
Original + Sal & Pimienta



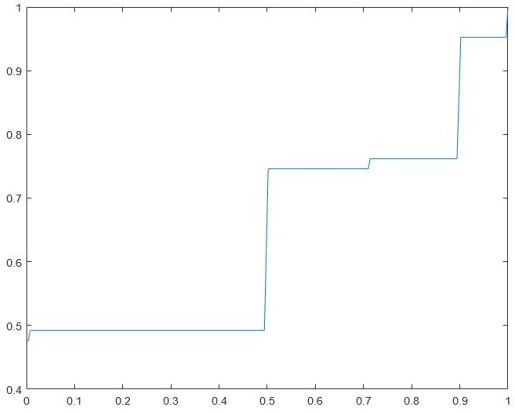
Histograma (S&P)



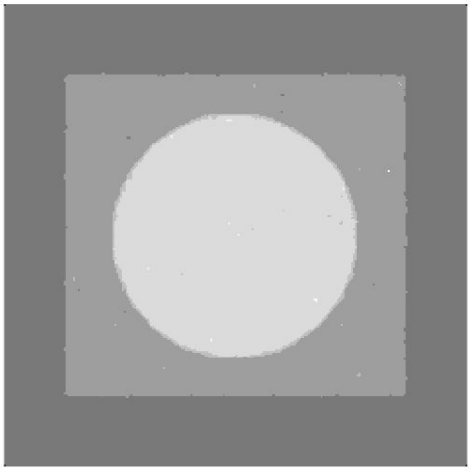
Igualación del Histograma (S&P)



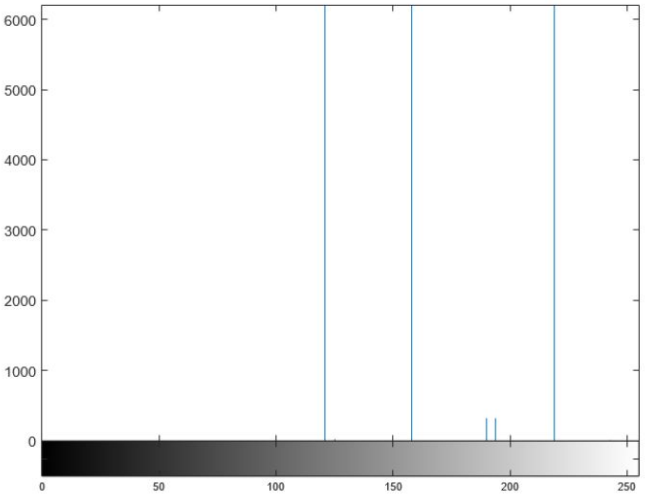
Histograma de la igualación (S&P)



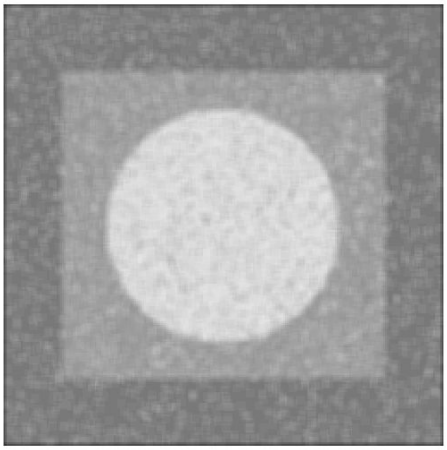
Transformación con el ruido (S&P)



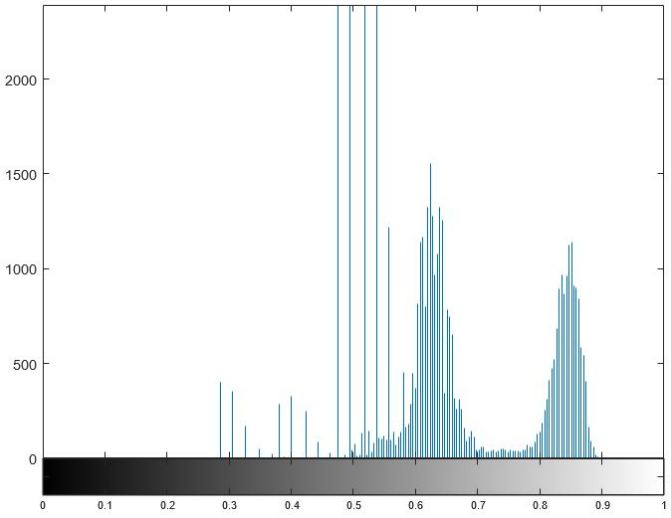
Filtro mediana (3 x 3)



Histograma del filtrado (3 x 3)



Filtro mediana (5 x 5)



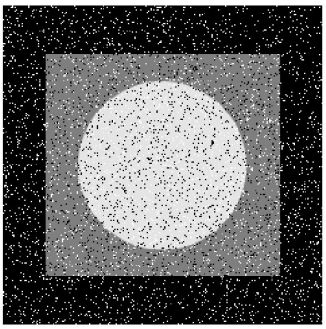
Histograma del filtrado (5 x 5)

Puede apreciarse que la figura corrige en gran medida el ruido "Sal y Pimienta" usando el filtro mediana 3x3, mientras que con el filtro mediana normal no funciona tan bien.

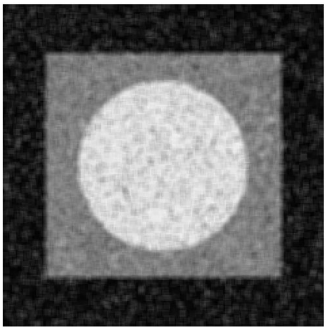
Esto puede deberse a que como la imagen presenta varias figuras de colores sólidos, funciona mejor con un filtro que actúa en zonas pequeñas (3x3) que usando el valor genérico de la mediana de toda la imagen.

Apartado b)

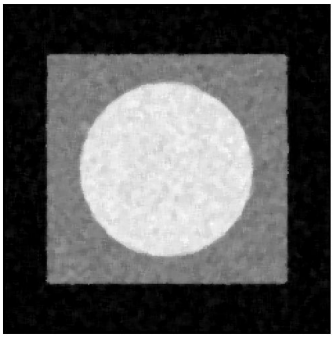
```
I = imread('Fig5.04(i).jpg');

% Añadir ruido "Sal y Pimienta" (figura 21.a)
I = imnoise(I, 'salt & pepper', 0.02);

figure, imshow(I);


% Añadir ruido gaussiano
J = imnoise(I, 'gaussian', 0, 0.01);
g = fspecial('average', [5 5]);

M1 = filter2(g,J)/255;
figure, imshow(M1); % Figura 25.a

M2 = medfilt2(J,[5 5]);
figure, imshow(M2); % Figura 25.b
```



Original + Sal & Pimienta



Filtro media



Filtro mediana (5 x 5)

El filtro de media no ha eliminado del todo el ruido, pero ha logrado disminuirlo lo suficiente para ver claramente las figuras de la imagen, aunque aún conserva los desperfectos a un nivel bastante notable.

En cambio, el filtro de mediana pese a tampoco haber suprimido el ruido por completo, presenta una mayor calidad que el filtro anterior y el ruido no es tan notable, más bien lo contrario; sin embargo, al haber corregido el ruido de esa forma los lados de las figuras se han visto más afectados y se notan más los desperfectos en ellos.

Práctica 4

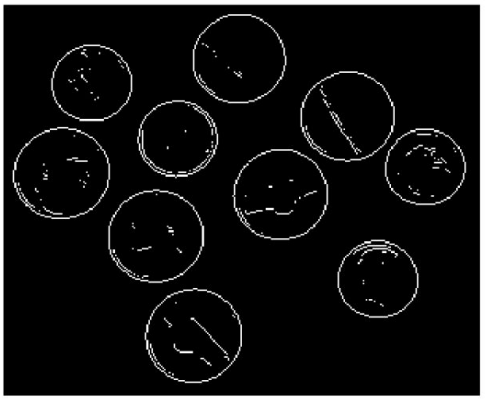
- **Todos** los ejercicios.
- **Parámetros** diferentes para las funciones.
- **Imágenes** diferentes a las del enunciado.

8 - Detección de bordes con filtrado en el dominio espacial

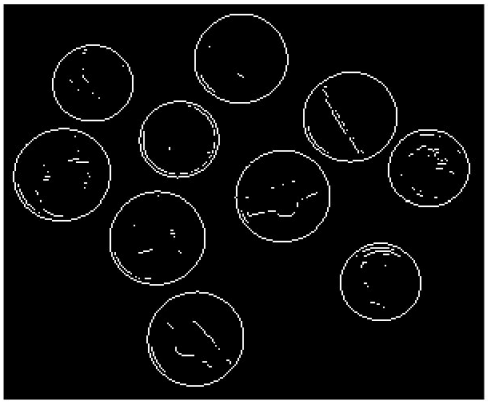
Apartado a)

```
I = imread('coins.png');

imshow(I)

% Filtrado de Sobel
I1 = edge(I, 'sobel'); % Detección de contornos

figure, imshow(I1);


% Filtrado de Prewitt
I2 = edge(I, 'prewitt'); % Detección de contornos

figure, imshow(I2);


% Filtrado de Canny
I3 = edge(I, 'canny'); % Detección de contornos

figure, imshow(I3);
```

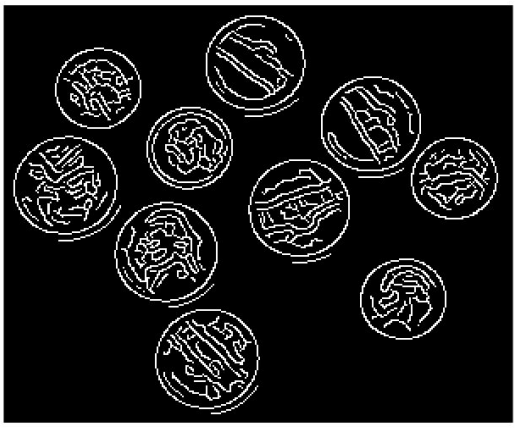
Original



Filtrado de Sobel



Filtrado de Prewitt

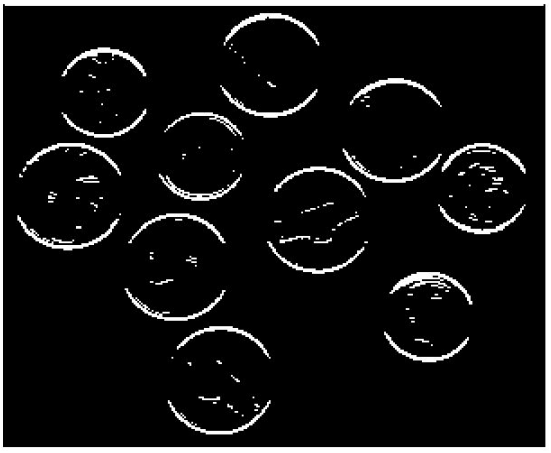


Filtrado de Canny

Apartado b)

```
I = imread('coins.png');

g = [1 1 1; 0 0 0; -1 -1 -1];
J = abs(filter2(g, I));
B = J > 0.27 * (max(J(:)) - min(J(:)));

figure, imshow(B);
```

Original



Bordes horizontales

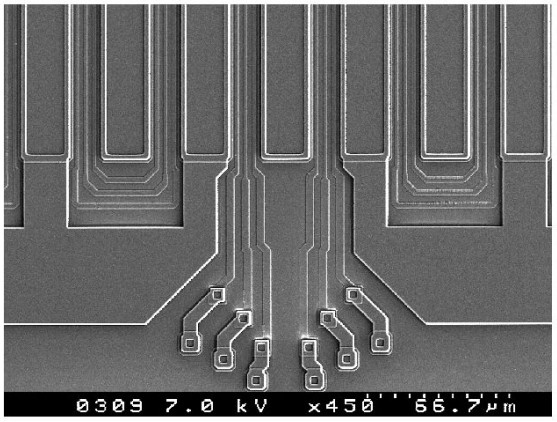
Tras definir la matriz g como el filtro de detección de bordes horizontales, se genera una nueva imagen J como el valor absoluto de cada valor de la matriz resultante de aplicarle a la imagen I la matriz g.

Posteriormente se genera una nueva imagen binaria B (0, negro; 1, blanco) mediante valores booleanos. Estos valores se obtienen multiplicando un factor a la diferencia del valor máximo y mínimo de la imagen J; y comprobando si el valor original de J es mayor que el resultado. Si el valor original supera al obtenido en la operación, se detecta un valor perteneciente al borde (blanco); en caso contrario, no (negro).

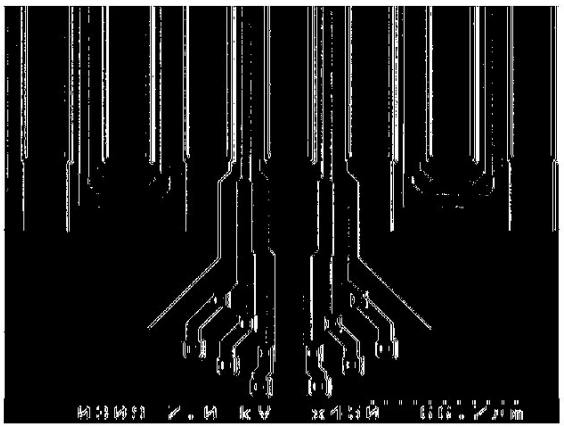
Apartado c)

Este ejercicio debe realizarse con la misma imagen que la del enunciado tutorizado.

```
I = imread('WAFER1.TIF');

figure, imshow(I);

g = [1 0 -1; 1 0 -1; 1 0 -1]; % Matriz traspuesta
(vertical)
J = abs(filter2(g, I));
B = J > 0.27 * (max(J(:)) - min(J(:)));

figure, imshow(B);
```



Original

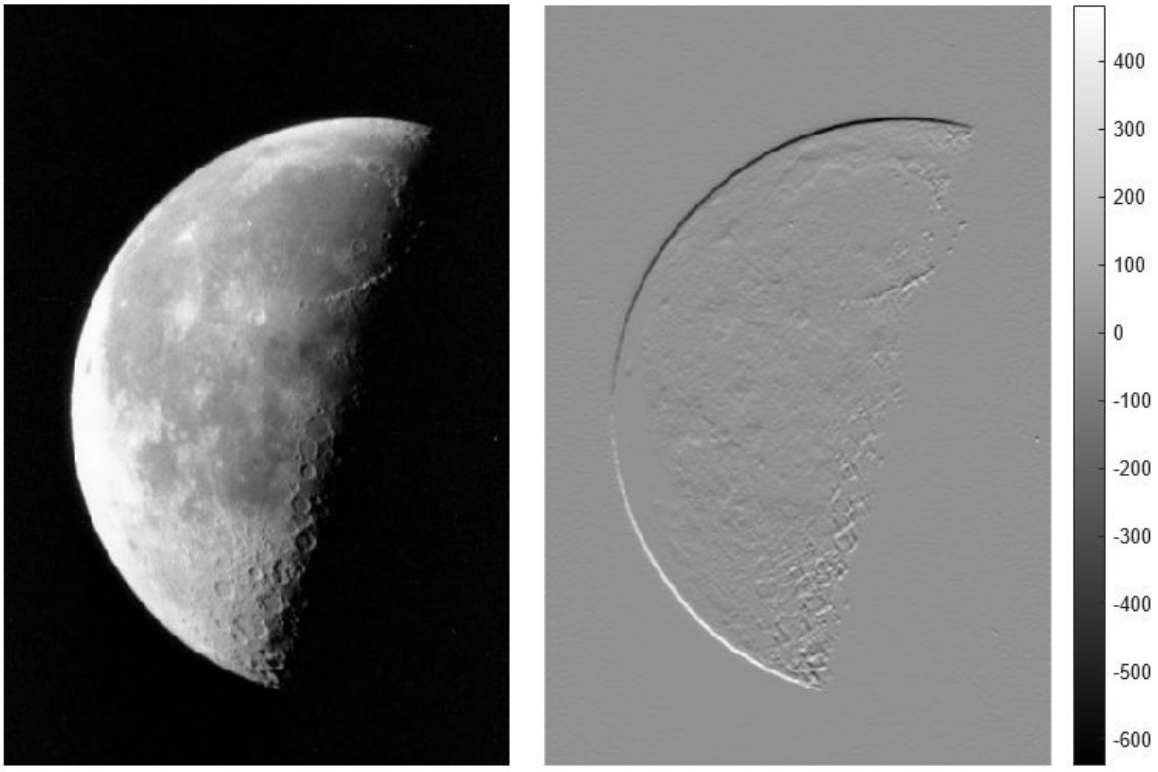


Bordes verticales

Se produce el mismo efecto que en el ejercicio anterior, solo que la matriz g es la traspuesta, de forma que los bordes detectados no serán horizontales, sino verticales.

Apartado d)

```
I = imread('moon.tif');

h = [1 2 1; 0 0 0; -1 -2 -1];
J = filter2(h, I);

imshow(J, []), colorbar
```



Original Filtrada

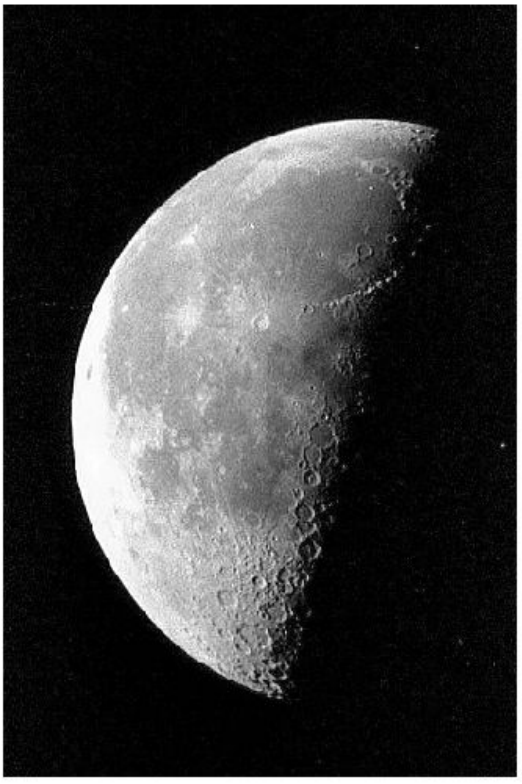
Definiendo la matriz h -un filtro de pasa alta- esta se aplica como filtro a la imagen I con la función `filter2`, generando la imagen J.

9 - Realzado de imágenes mediante filtrado

Apartado a)

```
I = imread('moon.tif');

figure, imshow(I);

h = fspecial('unsharp');
J = abs(filter2(h, I));

J1 = J/255; % J1 = J/max(J(:));

figure, imshow(J1)
```

Original



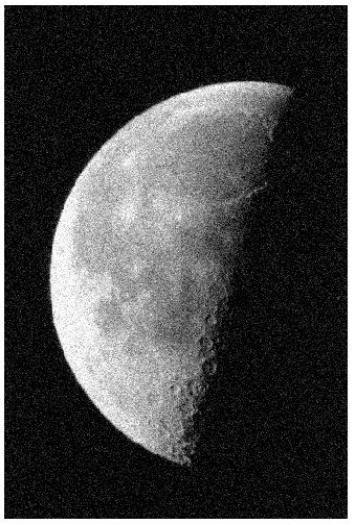
Perfilada

Usando la función `fspecial` se puede crear un filtro 2D predefinido; en este caso se usa para generar el filtro "unsharp", de modo que como se ha visto anteriormente, se genera una imagen J aplicando el filtro a la imagen I.

Además, en esta ocasión se genera una imagen J1 resultado de dividir los valores de J entre 255 (o entre el máximo valor de J) para realzarla.

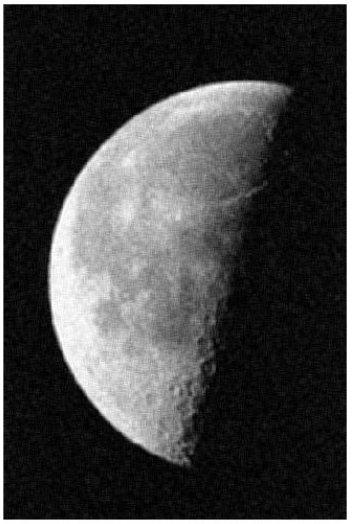
Apartado b)

```
I = imread ('moon.tif');

% Aplicación de ruido gaussiano
J = imnoise(I, 'gaussian' , 0, 0.01);

figure, imshow(J);


% Restauración con filtro media (5x5)
g = fspecial('average', [5 5]);
M1 = filter2(g, J)/255;

figure, imshow(M1);


% Realzado con el filtro "unsharp"
h = fspecial('unsharp');
M2 = abs(filter2(h, M1));

figure, imshow(M2);
```



Original + Ruido Gaussiano

Restauración con filtro media (5 x 5)

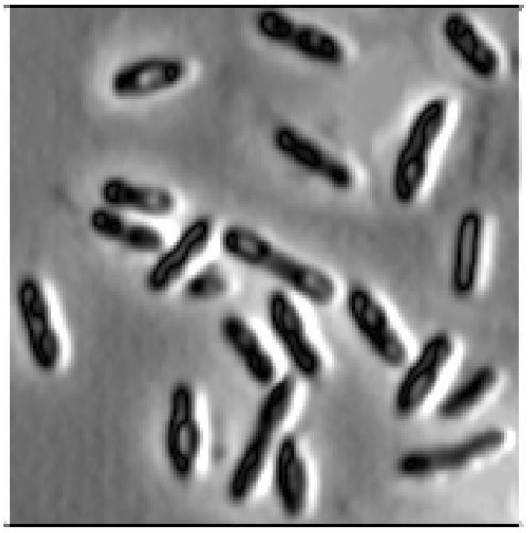


Realzado con filtro “unsharp”

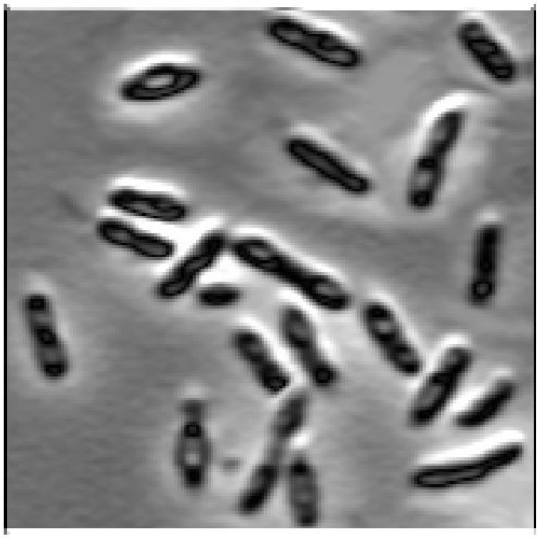
Como se mencionó antes, la función `fspecial` genera un filtro 2D predefinido, como en este caso son los filtros: *gaussiano*, de color 0 (negro) y factor de dispersión 0.01; *average*, usando matrices de 5x5; y *unsharp*, de realzado.

Apartado c)

```
I = imread('bacteria.tif');

% Realzado vertical
gv = [-1 0 1; -1 1 1; -1 0 1];
Jv = abs(filter2(gv, I))/255; % Ya se ha visto

figure, imshow(Jv);


% Realzado horizontal
gh = [1 1 1; 0 1 0; -1 -1 -1]; % Matriz anterior
traspuesta
Jh = abs(filter2(gh, I))/255; % Ya se ha visto

figure, imshow(Jh);
```



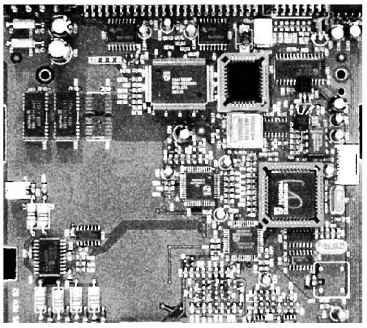
Realzado vertical



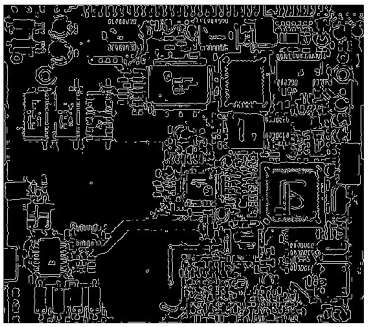
Realzado horizontal

Apartado d)

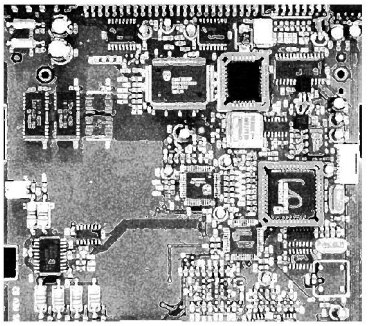
```
I = imread('Fig1.14(a).jpg');

% Realzar imagen
I = histeq(I); % Transformación en escala de
grises

figure, imshow(I);


% Generación de bordes
B = edge(I, 'canny'); % Detección de bordes
"canny"

figure, imshow(B)


% Realzar usando I y B (bordes resaltados)
R = double(I)/255 + 0.9 * double(B);

figure, imshow(R);
```



Original



Bordes

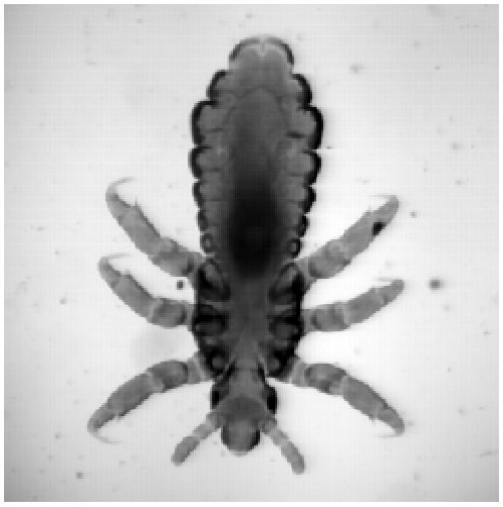


Realzado con Bordes

Apartado e)

Este ejercicio debe realizarse con la misma imagen que la del enunciado tutorizado.

```
I = imread('BUG.TIF');

figure, imshow(I);

h = [1 1 1; 0 1 0; -1 -1 -1];
J = abs(filter2(h, I))/255;

figure, imshow(J);
```



Original

Realzado de bordes horizontales

Práctica 5

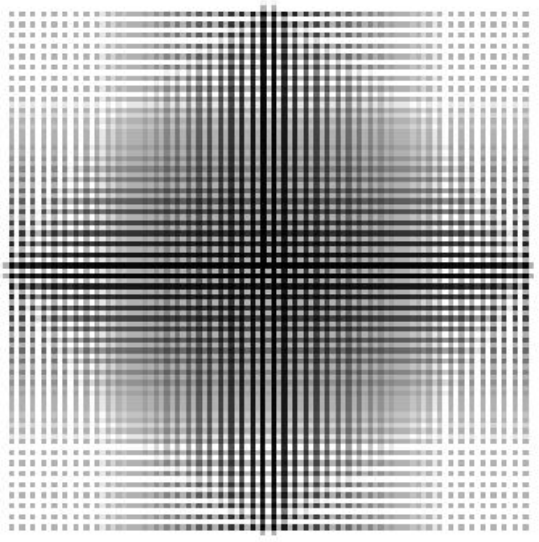
- **Todos** los ejercicios.
    - **Ejercicio 11**: aplicar cada filtro a una imagen diferente.
- **Imágenes** diferentes a las del enunciado.

10 - Transformada de Fourier discreta

Apartado a)

```
% Creación de la figura (un cuadradado)
I = ones(100, 100);
I(25:75, 25:75) = 0;

figure, imshow(I, 'initialMagnification', 'fit');


% Transformada de Fourier discreta de la figura
F = fft2(I);
F1 = log(1 + abs(F));

figure, imshow(F1, 'initialMagnification', 'fit');


% Gráfica de frecuencia
% freqz2(F1); % Interfiere con la muestra de la figura anterior
```

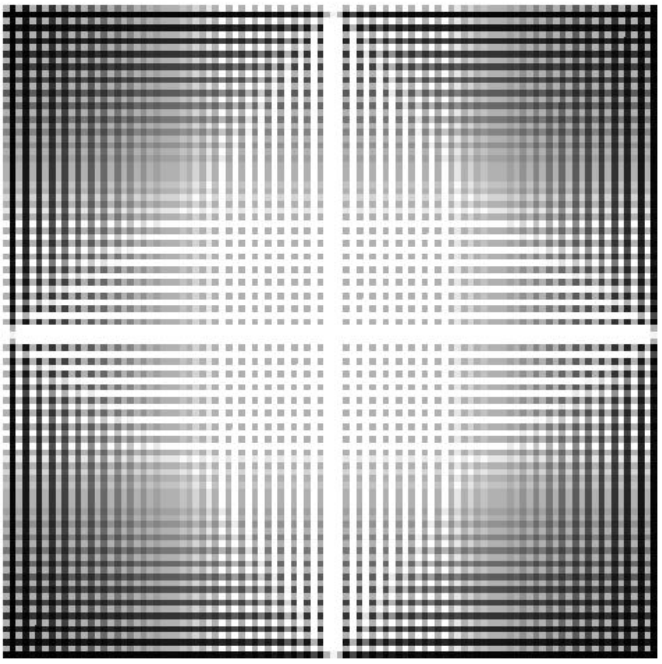
Original (cuadrado negro sobre blanco)



Transformada de Fourier discreta

Apartado b)

```
% Creación de la figura (un cuadradado)
I = ones(100, 100);
I(25:75, 25:75) = 0;

figure, imshow(I, 'initialMagnification', 'fit');


% Transformada de Fourier discreta de la figura
F = fft2(I);
F1 = fftshift(F);
F2 = log(1 + abs(F1));

figure, imshow(F2, 'YDATA', [-100,100], 'XDATA', [-100,100]);


% Gráfica de frecuencia
% freqz2(F1);             % Interfiere con la muestra de la figura
anterior
```



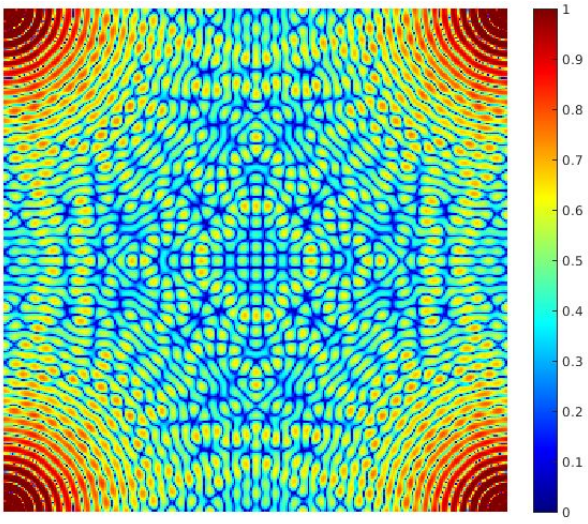
Transformada de Fourier de
la imagen original anterior con
el valor |F(0,0)| en el centro.

Durante los siguientes ejercicios se vuelve a usar la función `imshow()` con los parámetros `YDATA []` y `XDATA []`, esto para centrar el valor indicado en la imagen. Estos parámetros actúan como valor mínimo y máximo en el eje indicado usando el primer y último valor de cada vector asociado a dichos parámetros.

Apartado c)

```
% Creación de la figura (un círculo)
I = ones(300, 300);

for i = 1:100
    for j = 1:200
        if (i-50)^2 + (j-50)^2 < 700
            I(i,j) = 0;
        end
    end
end

figure, imshow(I, 'initialMagnification', 'fit');


% Transformada de Fourier discreta de la figura
F = fft2(I); % F = fft2(I, 256, 256);
F1 = log(1 + abs(F));

figure, imshow(F1/4, 'initialMagnification', 'fit');


% Mapa de color del espectro de Fourier
colormap('jet');
colorbar;
```

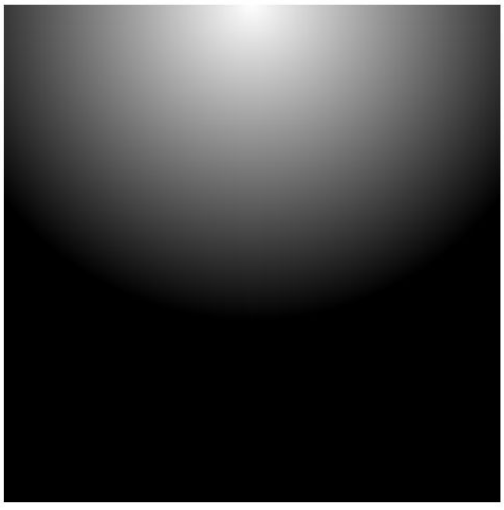
Figura



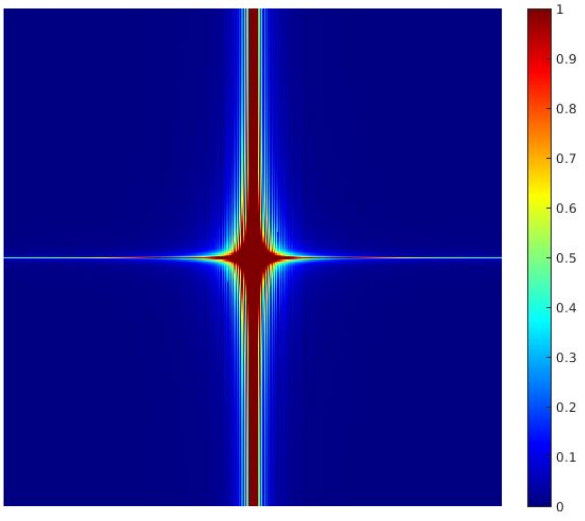
Espectro de Fourier

Apartado d)

```
% Coordenadas del foco (a,b)
a = 0;
b = 200;

% Construcción de la imagen M x N (400 x 400)
for x = 1:400
    for y = 1:400
        I(x,y) = (255 - sqrt((x-a)^2 + (y-b)^2)) / 255;
    end
end

% Mostrar la imagen
figure, imshow(I);


% Transformada de Fourier discreta de la figura
F = fft2(I); % F = fft2(I, 256, 256);
F1 = fftshift(F);
F2 = log(1 + abs(F1));

figure, imshow(F2, 'YDATA', [-100,100], 'XDATA', [-100,100]);


% Mapa de color del espectro de Fourier
colormap('jet');
colorbar;
```



Punto luminoso

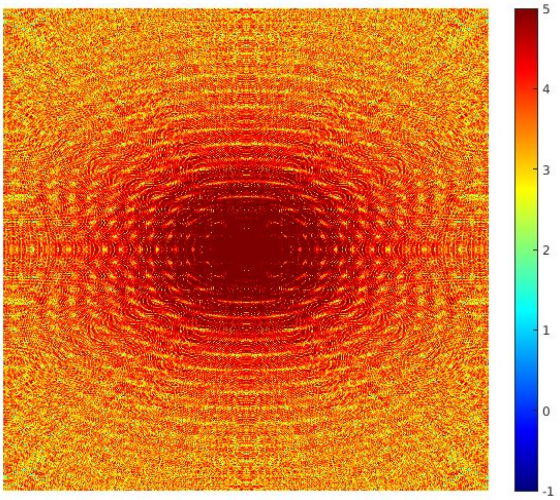


Espectro de Fourier

Apartado e)

```
% Crear una figura con la función phantom()
I = phantom(1984, 'Shepp-Logan');

figure, imshow(I);


% Transformada de Fourier discreta de la figura
F = fft2(I);
F1 = fftshift(F);
F2 = log(abs(F1));

% Figura centrada en (0,0) con una barra desde -1 hasta 5
figure, imshow(F2, [-1 5], 'YDATA', [-100,100], 'XDATA', [-100,100]);


% Mapa de color del espectro de Fourier
colormap('jet');
colorbar;
```

Tomografía



Espectro de Fourier

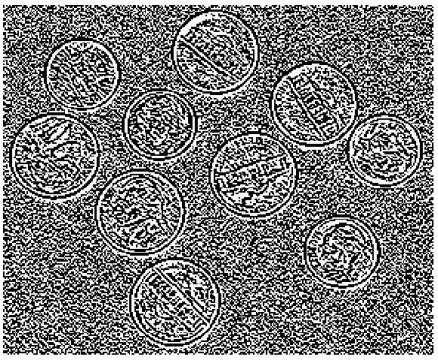
11 - Diseño de filtros en el dominio de las frecuencias

Para este ejercicio se pide aplicar todos los filtros obtenidos en los apartados a una imagen distinta cada uno, por lo que elaboré un solo script que realiza eso.

```
% Imágenes (una por apartado)
C = imread('cameraman.tif');
M = imread('coins.png');
F = imread('Fig5.26(a).jpg');
S = imread('saturno.tif');
L = imread('moon.tif');



% Ejercicio 11 a (ERROR)
% [u,v] = freqspace(25, 'meshgrid');
%
% H = sqrt(u.^2 + v.^2) < 0.5;
%
% figure, imshow(filter2(H,C)); % Aplicar el filtro H a la imagen C



% Ejercicio 11 b
H = [0 -1 0; -1 4 -1; 0 -1 0];

figure, imshow(filter2(H,M)); % Aplicar el filtro H a la imagen M



% Ejercicio 11 c (ERROR)
% H = fspecial('gaussian');
%
% figure, imshow(filter2(H,F)); % Aplicar el filtro H a la imagen F



% Ejercicio 11 d
H = zeros(15,15);
H(5:10, 5:10) = 1;

% Función de ventana
h = fwind1(H, hamming(15));

figure, imshow(filter2(h,S)); % Aplicar el filtro H a la imagen S



% Ejercicio 11 e
[f1,f2] = freqspace(15, 'meshgrid');

Hd = zeros(15,15);
Hd(4:11, 4:11) = 1;

% Muestreo de frecuencias
h = fsamp2(Hd);

figure, imshow(filter2(h,L)); % Aplicar el filtro H a la imagen L
```

Apartado b)

Imagen original



Filtro de Laplace (3 x 3)

Apartado d)

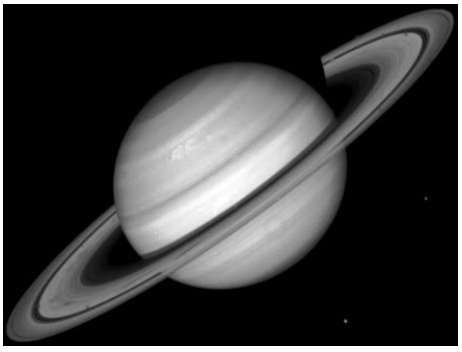
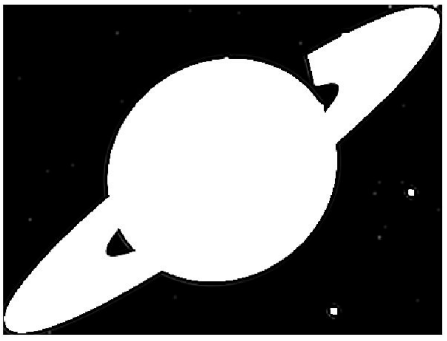


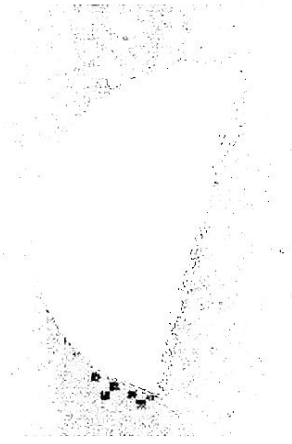
Imagen original



Filtro por ventanas de Hamming

Apartado e)

Imagen original



Filtro por muestreo de frecuencias

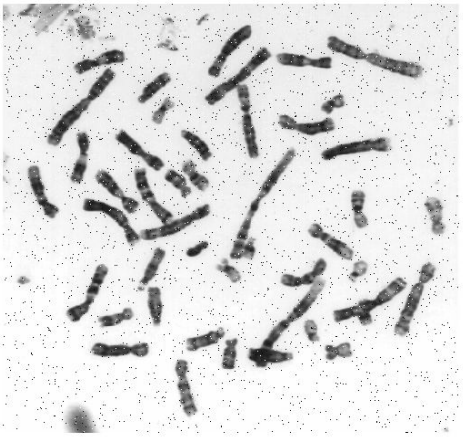
Práctica 6

- **Todos** los ejercicios.
    - **Ejercicio 12**: e, f, g, h, i.
    - **Ejercicio 13**: d.
- **Parámetros** diferentes a los del enunciado.
- **Imágenes** diferentes a las del enunciado.

12 - Aplicación de filtros en el dominio de las frecuencias

Apartado e)

```
I = imnoise(imread('CHROMOS2.TIF'), 'salt & pepper', 0.02);

figure, imshow(I);

I = im2double(I);
[M,N] = size(I);

F = fft2(I);
H = lpfilter('btw', M, N, 25);
G = H.*F;

I1 = real(ifft2(G));

figure, imshow(I1);
```



Original (+ Sal & Pimienta)

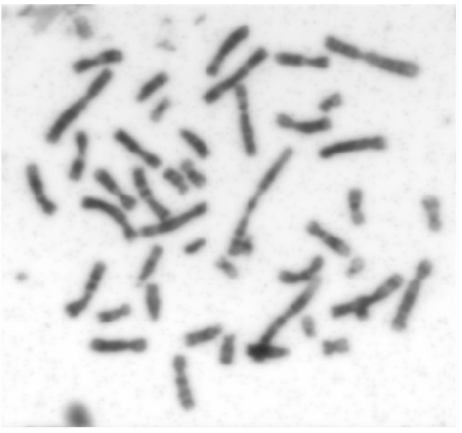


Imagen filtrada

Apartado f)

```
I = imread('cameraman.tif');

I = imnoise(I, 'salt & pepper', 0.02);
F = fft2(I);

[f1,f2] = freqspace(25, 'meshgrid');

H = sqrt(f1.^2+f2.^2) < 0.5;
H1 = ifft2(H(13:25, 13:25));

B = filter2(real(H1), I);

figure, imshow(I);
figure, imshow(B, [min(min(B)) max(max(B))]);
```

Original (+ Sal & Pimienta)

Imagen filtrada

Apartado g)

```
I = imread('cameraman.tif');
figure, imshow(I);

F = fft2(I);

[f1,f2] = freqspace(25, 'meshgrid');

H = sqrt(f1.^2+f2.^2) > 1.1;
H1 = ifft2(H(13:25,13:25));

B = filter2(real(H1), I);

figure, imshow(B, [min(min(B)) max(max(B))]);
```

Imagen original

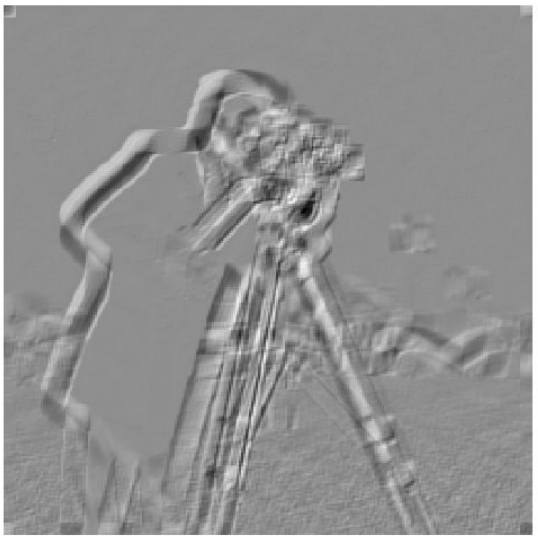


Imagen filtrada

Apartado h)

```
[f1,f2] = freqspace(15, 'meshgrid');

H0 = zeros(15,15);
H0(12:15, 12:15) = 1;

H = fwind1(H0, hamming(15));

colormap(jet(64));
freqz2(H,[32 32]);

pause(3); % Se usa para ver el cambio del histograma
mesh(f1, f2, abs(H)); % Cambio del histograma
pause(3); % Se usa para ver el cambio del histograma

h = ifft2(H(7:15, 7:15));

I = imread('cameraman.tif');

B = filter2(real(h), I);

figure, imshow(B, [min(min(B)) max(max(B))]);
```

Imagen original

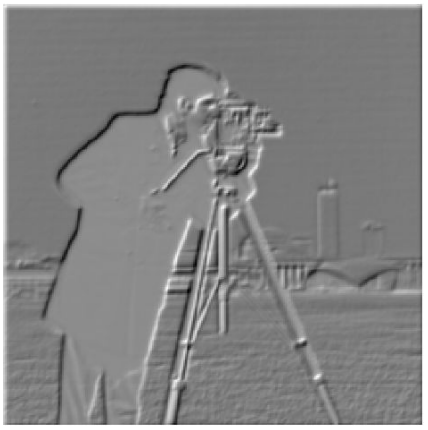
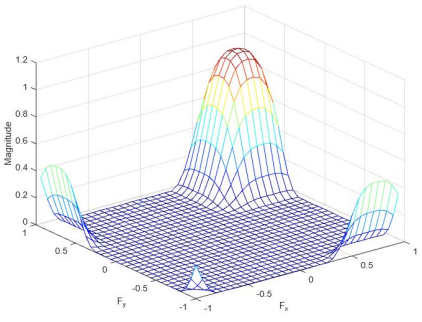
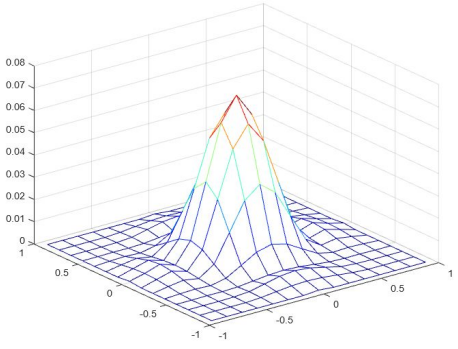


Imagen filtrada

Cambio del histograma:



Antes



Después

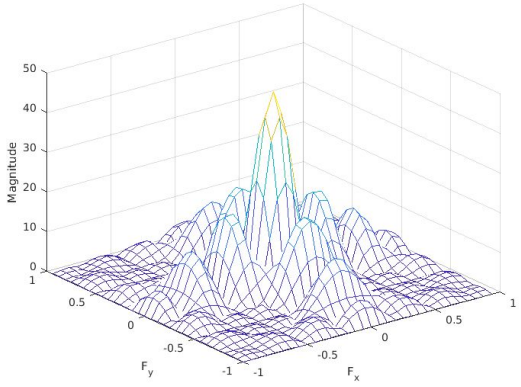
Apartado i)

```
[f1,f2] = freqspace(16, 'meshgrid');

H = zeros(16,16);
H(5:12, 5:12) = 1;
H(7:10, 7:10) = 0;

freqz2(H, [32 32]);

pause(3); % Se usa para ver el cambio del histograma
mesh(f1, f2, H); % Cambio del histograma
pause(3); % Se usa para ver el cambio del histograma

h = ifft2(H(9:16, 9:16));

I = imread('cameraman.tif');

B = filter2(real(h), I);

figure, imshow(B, [min(min(B)) max(max(B))]);
```

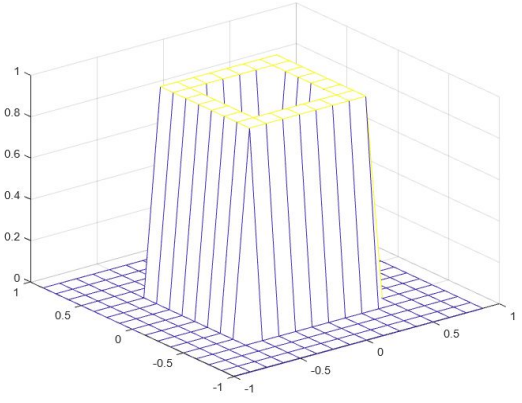
Imagen original

Imagen filtrada

Cambio del histograma:



Antes



Después

13 - Restauración de imágenes degradadas o distorsionadas

Apartado d)

```
I = imread('5.1.12.tiff');

figure, imshow(I);

% Crear el filtro para la imagen movida
h = fspecial('gaussian', 7, 10); % Crear el filtro de movimiento
B = imfilter(I, h, 'symmetric', 'conv'); % Aplicar el filtro

figure, imshow(B);


% Restauracion 1
h1 = ones(size(h) - 4); % Crea una matriz de unos
[J1, P1] = deconvblind(B, h1); % Des-borrosear la imagen a ciegas

figure, imshow(J1);


% Restauracion 2
h2 = padarray(h1, [4:4], 'replicate', 'both');
[J2, P2] = deconvblind(B, h2); % Des-borrosear la imagen a ciegas

figure, imshow(J2);


% Restauracion 3
h3 = padarray(h1, [2 2], 'replicate', 'both');
[J3, P3] = deconvblind(B, h3); % Des-borrosear la imagen a ciegas

figure, imshow(J3);


% Restauracion 4
% Creacion del filtro
peso = edge(I, 'sobel', .3); % Bordes usando el metodo Sobel (factor .3)
ee = strel('disk', 2); % Crea una estructura de disco (factor 2)
peso = 1 - double(imdilate(peso, ee)); % Dilata la imagen y la transforma

peso(:, [1:3 end-[0:2]]) = 0; % Imagen resultante, con bordes marcados

figure, imshow(peso);

% Restauracion
[J, P] = deconvblind(B, h3, 30, [], peso); % Des-borrosear la imagen a ciegas

figure, imshow(J);
```

Imagen a restaurar

Restauración 1

Restauración 2

Restauración 3

Creación del filtro (método Sobel)

Restauración 4

Práctica 7

- **Ejercicio 14**: e, f, g.

14 - Análisis morfológico

Apartado e)

```
I = imread('cell.tif');

figure, imshow(I);


B = edge(I, 'sobel', (graythresh(I)*.1));

figure, imshow(B);


h90 = strel('line', 3, 90);
h0 = strel('line', 3, 0);
Bd = imdilate(B, [h90 h0]);

figure, imshow(Bd)


Br = imfill(Bd, 'holes');
Bs = imclearborder(Br, 4);

figure, imshow(Br);
figure, imshow(Bs);


hd = strel('diamond',1);
Bfinal = imerode(Bs,hd);

figure, imshow(Bfinal);


BL = bwperim(Bfinal);
Segout = I;
Segout(BL) = 255;

figure, imshow(Segout);
```

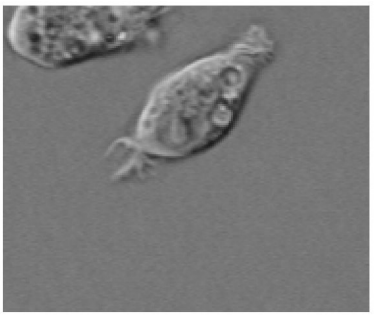
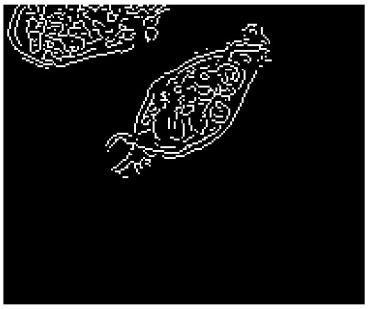


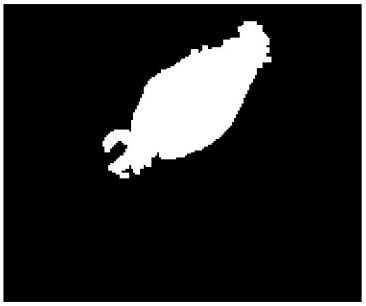
Imagen original



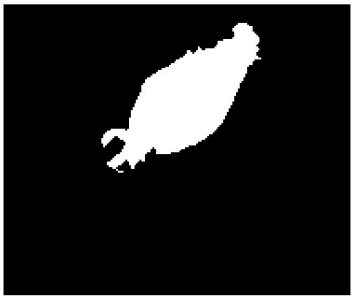
Filtro de Sobel

Dilatación de bordes

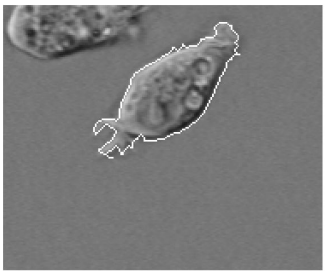
Relleno de huecos



Limpieza de bordes



Erosionado de bordes



Detección de bordes

Apartado f)

```
% Otra opcion: Aplicacion directa del operador TopHat
I = imread('rice.tif');

se = strel('disk', 10);
U = imtophat(I, se);

figure, imshow(U);
```

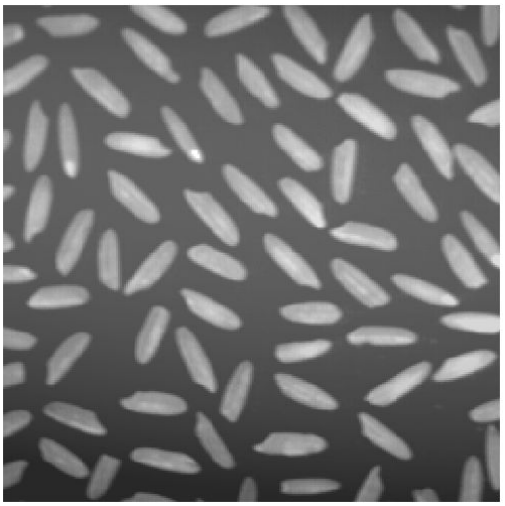


Imagen original

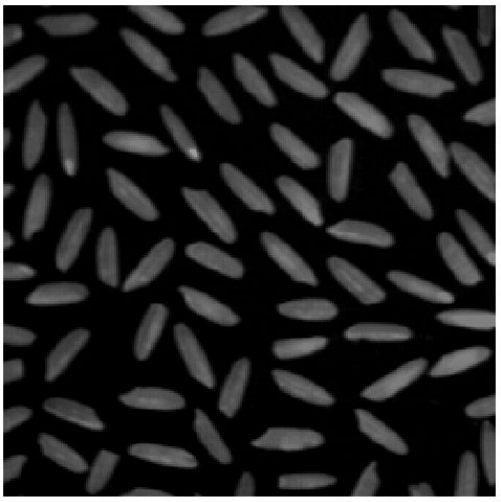


Imagen corregida

Apartado g)

```
I = imread('ngc4024m.tif');

figure, imshow(I);

se = strel('disk', 3);
E = imerode(I, se);
D = imdilate(E, se); % D = imtophot(I);

F = I - D;

figure, imshow(F);
```

Imagen original

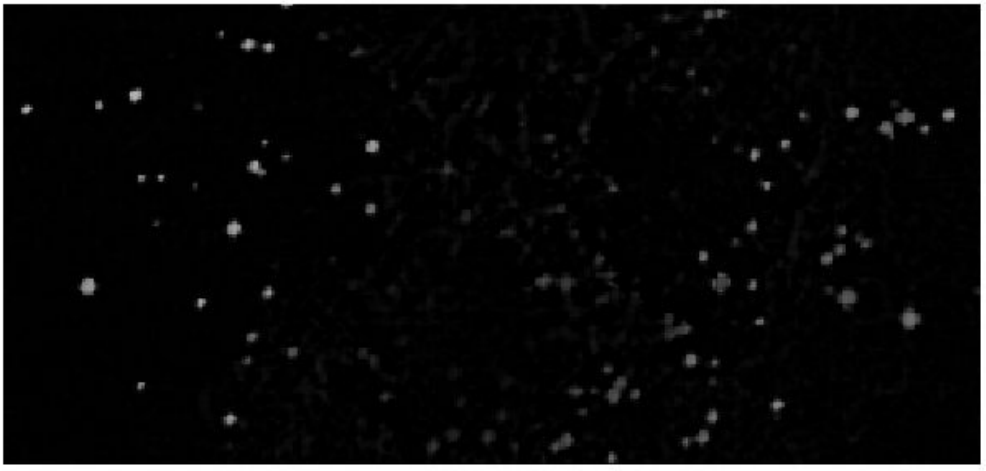


Imagen con los objetos más pequeños