FILTRADO FRECUENCIAL

Puede comprobar el comportamiento de los distintos filtros en:

http://bigwww.epfl.ch/demo/ip/demos/FFT-filtering/

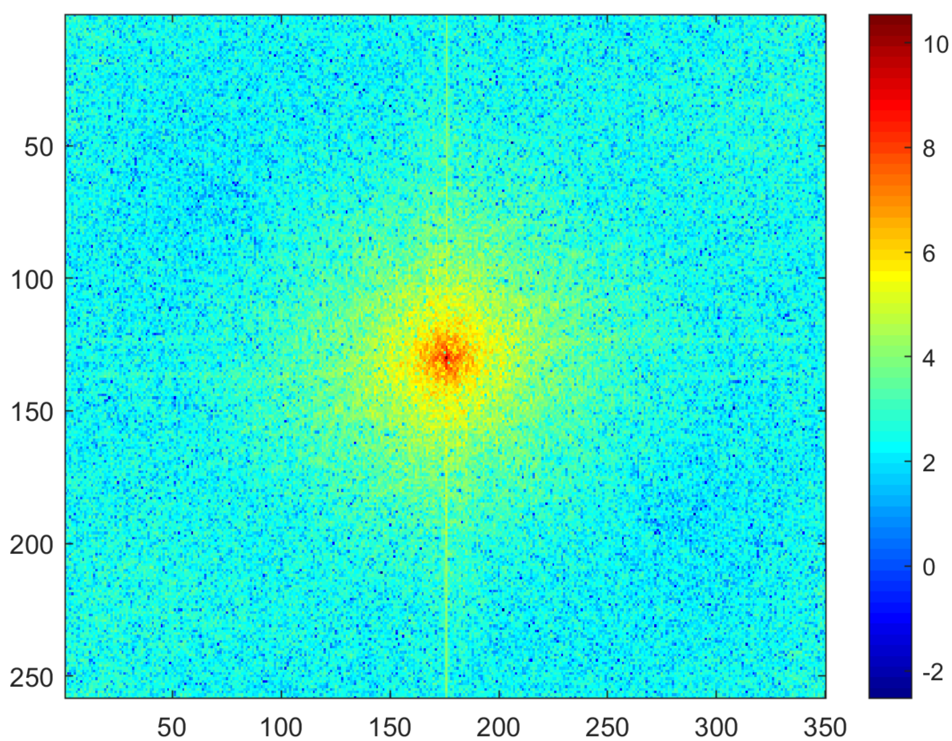
En este documento se estudiarán la transformada rápida de Fourier (FFT) y la transformada del coseno discreta proporcionadas por MATLAB. Se verá la forma de recuperar una imagen a partir de los primeros coeficientes de la transformada del coseno discreta. A continuación, se describirán las técnicas que permiten realizar operaciones de filtrado en el dominio de la frecuencia, explicando cómo se definiría un filtro en dicho dominio.

Transformada rápida de Fourier (FFT).

La función *fft2* calcula la transformada bidimensional rápida de Fourier (FFT).

Este ejemplo genera la FFT bidimensional de la imagen “trees” y visualiza la magnitud principal del resultado.

```
load trees
I=ind2gray(X,map);
F=fftshift(fft2(I)); %Se utiliza fftshift para centrar la F(0,0) de la transformada
figure,colormap(jet(64)), imagesc(log(abs(F))); colorbar
```



fft2 devuelve las componentes en frecuencia para frecuencias dentro del rango de 0 a 2π, con el origen (la componente de frecuencia cero) en la esquina superior izquierda de la matriz resultante. La función *fftshift* recoloca la salida de *fft2*, moviendo el origen al centro (-π a π). Como se puede observar, la transformada

tiene las mismas dimensiones que la imagen de entrada (258x350) pero ahora los ejes se corresponden con frecuencias. Observe que la mayor parte de las componentes significativas se encuentran cerca del origen.

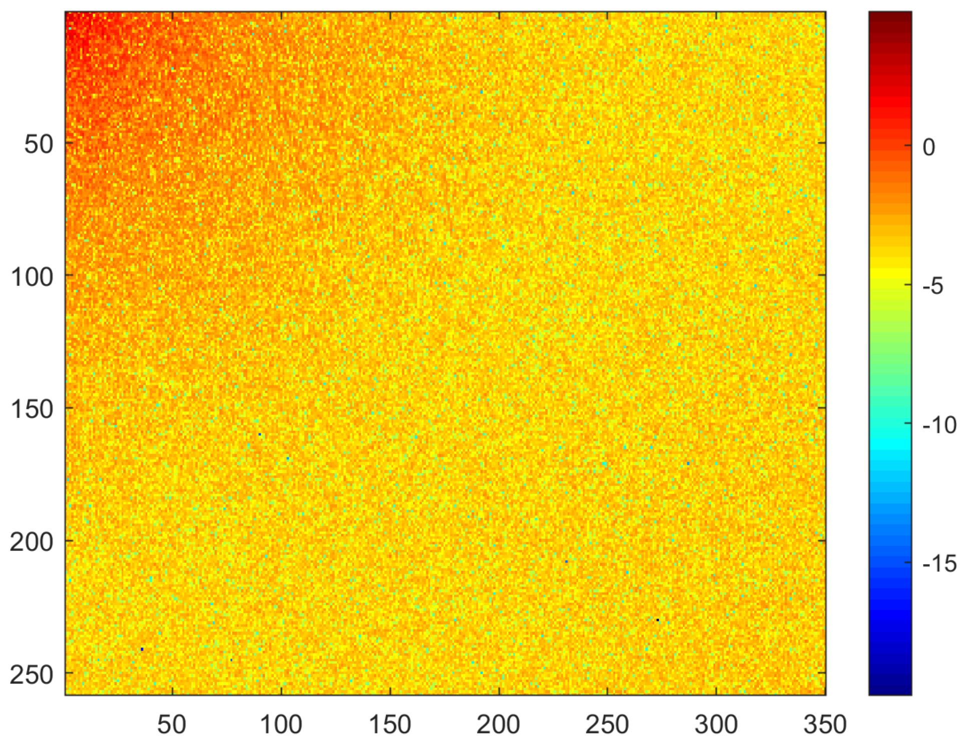
Realice la fft de diferentes imágenes y observe los resultados.

Transformada coseno discreta.

La transformada coseno discreta está basada en la FFT, pero tiene mejores propiedades de compactación de energía, haciéndola útil para codificación de imágenes. La función *dct2* implementa la transformada del coseno discreta bidimensional.

El siguiente ejemplo calcula la transformada del coseno discreta para la imagen “trees”. Notar que la mayoría de la energía está en la esquina superior izquierda.

```
load trees
I=ind2gray(X,map);
J=dct2(I);
figure, colormap(jet(64)), imagesc(log(abs(J))),colorbar
```

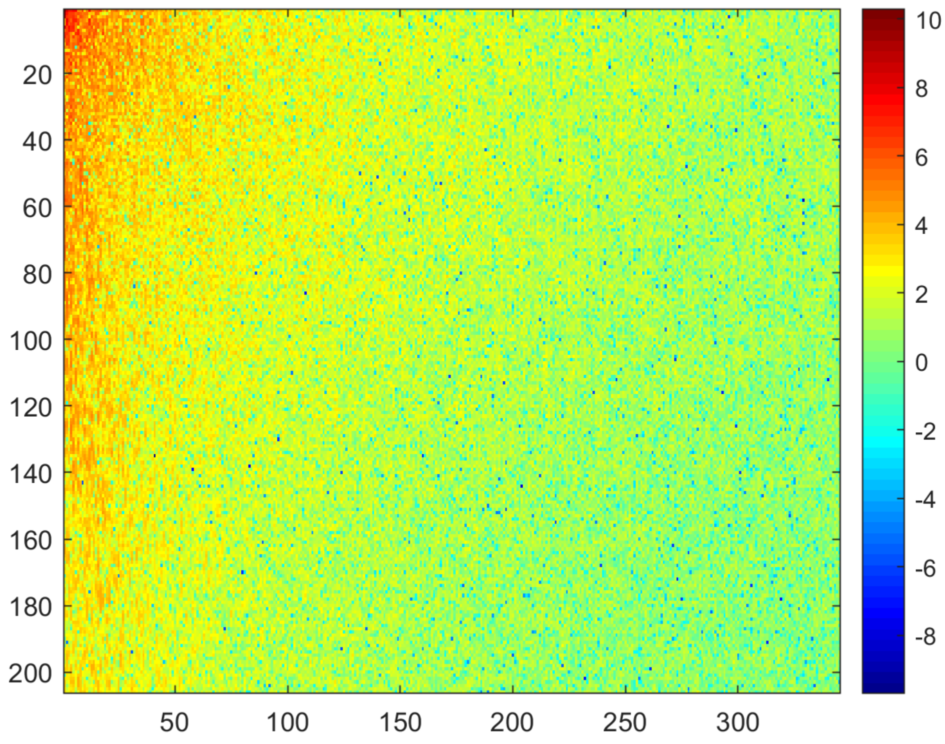


Compare los resultados de esta transformada con los de la FFT del apartado anterior.

Los siguientes comandos ponen los valores menores que 10 en la matriz DCT a cero y visualiza la imagen formada utilizando la función inversa DCT, *idct2*.

```
RGB = imread('autumn.tif');       %Los valores R,G,B están entre 0 y 255 (uint8)
I = rgb2gray(RGB);                          %Convierte la imagen a niveles de gris uint8 (0 negr
J = dct2(I);                                    %J es la transformada coseno discreta de la ima
```

```
figure, imagesc(log(abs(J))), colormap(jet(64)), colorbar
```




```
J(abs(J)<10) = 0;                              %Pone a 0 los valores de la transformada de mod
K = idct2(J);  %Halla la transformada inversa de J con componentes eliminadas. K: tipo double e
figure, subplot(1,2,1), imshow(I), title('Imagen original');
subplot(1,2,2), imshow(K,[0 255]), title('Imagen tras eliminar componentes de la DCT ');
```


```
%Uso imshow para mostrar la imagen K de tipo double pero de rango entre [0 255] y no el de por
```

En este caso, se puede comprobar como el 76'6% de los elementos de la matriz de la transformada son puestos a cero. Sin embargo, debido a las propiedades de la compactación DCT, la mayoría de la información de la imagen está todavía presente. Por este motivo la transformada del coseno discreta es ampliamente usada en aplicaciones de compresión, como, por ejemplo, en el algoritmo de compresión de imágenes JPEG.

Filtrado en el dominio frecuencial

Realice el siguiente ejemplo de filtrado en el dominio de la frecuencia:

```
load trees;
im=ind2gray(X,map); %Se trabaja con una imagen a nivel de gris double [0 1]
h=1/9*[1 1 1;1 1 1; 1 1 1]; %Definición de la máscara filtro paso bajo en el dominio espacial
imagen_tras_filtro_espacial=filter2(h,im,'full'); %Filtrado en el dominio espacial

%Filtrado equivalente en el dominio de la frecuencia
[M,N]=size(im);
[P,Q]=size(h);
tamano_filas_fft=M+P-1;
%Este es el tamaño en filas de la imagen tras la convolucion y sera el tamano en filas de su FF
tamano_columnas_fft=N+Q-1;
%Este es el tamaño en columnas de la imagen tras la convolucion y sera el tamano en col. de su
TF_imagen=fft2(im,tamano_filas_fft,tamano_columnas_fft); %FFT de la imagen
TF_mascara=fft2(h,tamano_filas_fft,tamano_columnas_fft); %FFT de la mascara
```

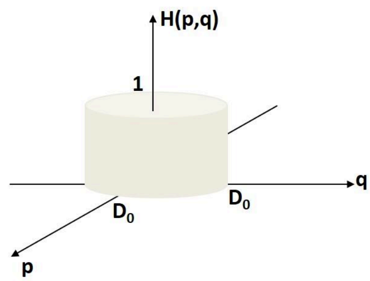
```
TF_imagen_filtrada=TF_imagen.*TF_mascara; %Multiplico transformadas en el dominio de la frec.
imagen_tras_filtro_frecuencial=real(ifft2(TF_imagen_filtrada)); %Con FFT inversa -> imagen filt
figure, subplot(1,2,1), imshow(imagen_tras_filtro_espacial), title('Imagen tras filtro espacial
subplot(1,2,2), imshow(imagen_tras_filtro_frecuencial), title('Imagen tras filtro dominio frecu
```

En el ejemplo anterior, partimos de un filtro (máscara) definido en el dominio del espacio, del cual se obtuvo la FFT para hallar la respuesta del filtro en el dominio de la frecuencia. También se puede definir el filtro directamente en el dominio de la frecuencia, como se verá en el apartado siguiente.

Filtro paso bajo ideal (en el dominio de la frecuencia):

En las frecuencias (p,q) situadas en un radio alrededor de la frecuencia más baja (0,0) el filtro da salida 1, en el resto da salida 0.

$$H(p,q)=\begin{cases}1 & \sqrt{p^2+q^2}\le Ho \\ 0 & resto\end{cases}$$



En la ecuación anterior se supone el píxel de frecuencia (0,0) en la posición (0,0) de la imagen, de modo que el término $\sqrt{(\mathrm{p2}+\mathrm{q2})}=\sqrt{[(p-0)^2+(q-0)^2]}$ representa la distancia del píxel (p,q) al píxel (0,0). En el programa se supone el punto de frecuencia (0,0) en el centro de la imagen y por eso se halla la distancia de un píxel al central en la posición [floor((M/2)+1),floor((N/2)+1)] donde estaría la frecuencia (0,0) de la imagen.

Ejecute el siguiente código, poniendo título a las gráficas mostradas, utilizando diferentes frecuencias de corte, además de la puesta en el ejemplo de 0.2*M:

```
load trees; im=ind2gray(X,map);figure,imshow(im), title('Imagen original');
```

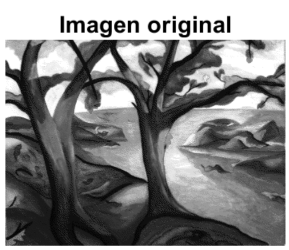
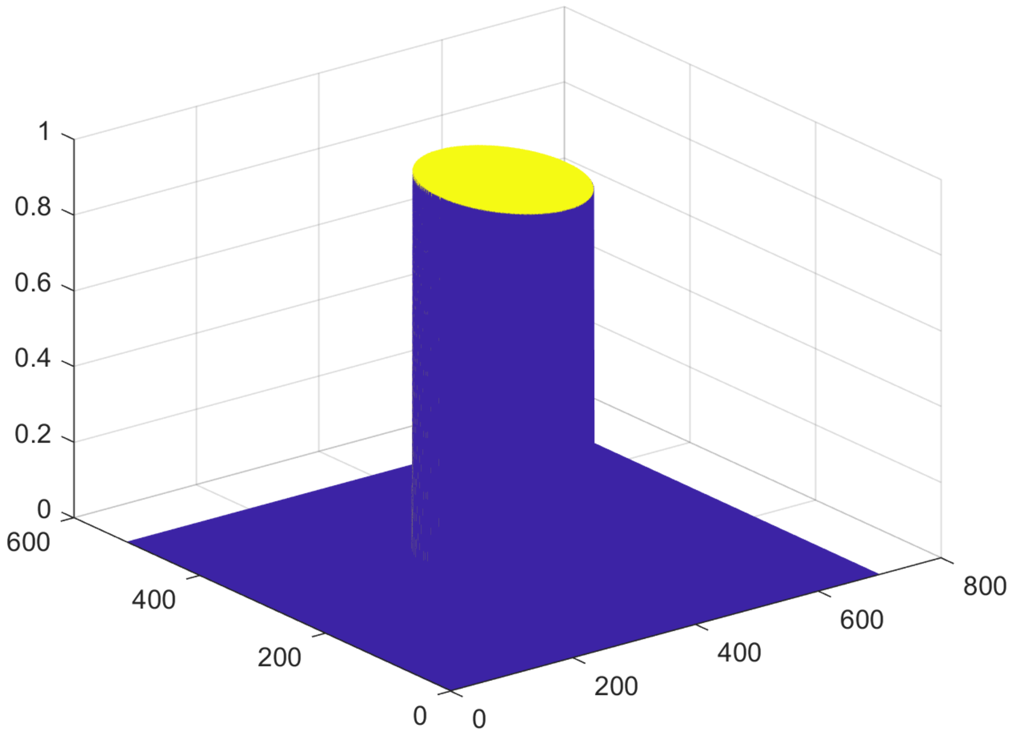


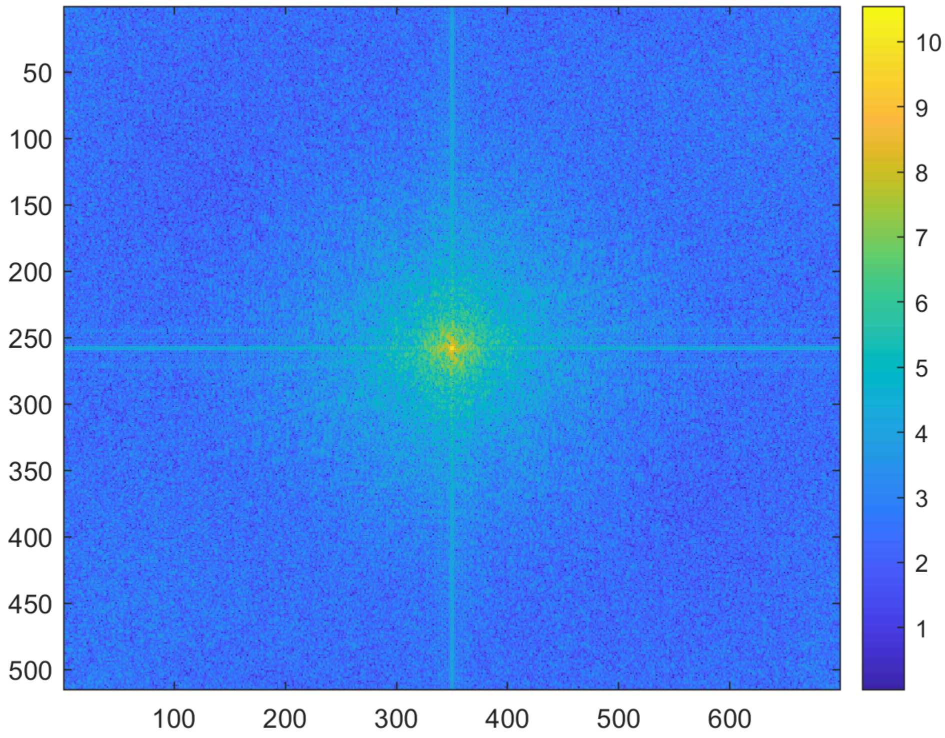
Imagen tras filtro espacial

Imagen tras filtro dominio frecuencial

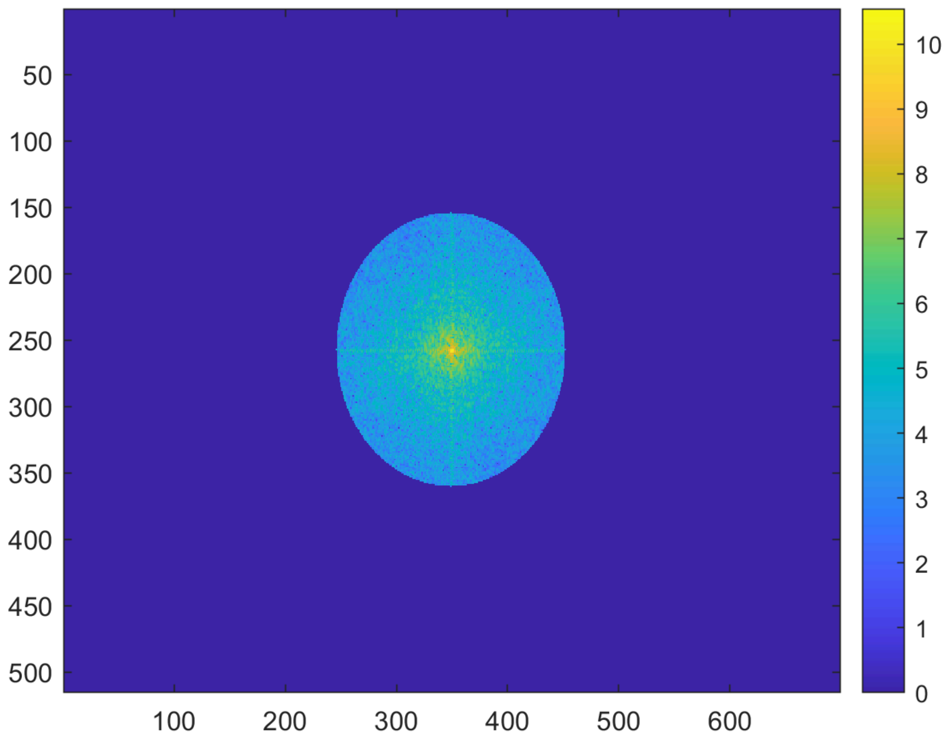
```
[filas_im,columnas_im]=size(im);
N=(2*columnas_im)-1; %Tamano columnas de la transformada
M=(2*filas_im)-1;    %Tamano filas de la transformada
pp=1:N; qq=1:M;[p,q]=meshgrid(pp,qq);
%p,q forman todas las posibles combinaciones de pp y qq, definiendo las MxN posibles frecuencia
%La frecuencia 0,0 la supondremos centrada en la imagen, en la columna floor((N/2)+1) y fila fl
D=sqrt((p-floor((N/2)+1)).^2+(q-floor((M/2)+1)).^2)<=(0.2*M);%Radio filtrado: 0.2*M. Fija frecu
%Cojo frecuencias centrales alrededor del píxel central de la imagen situado en: floor((N/2)+1)
% Para ello hallo la distancia entre el pixel en la posición central de la imagen (de frecuenci
% pixeles, quedandonos con los pixeles cuyas posiciones estan dentro de un radio alrededor del
H=zeros(M,N);
H(D)=1; %Pongo a 1 las frecuencias centrales del filtro, dentro del radio de filtrado alrededor
figure,mesh(p,q,H);
```



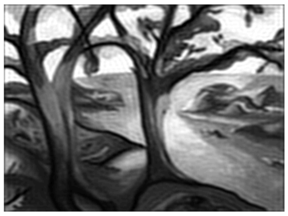

```
NIM=fft2(im,M,N).*fftshift(H);
%Multiplico la transformada de Fourier de la imagen (con F(0,0) en esquina superior izquierda .
%...con el filtro (desplazado para que frecuencia 0,0 del filtro este en esquina superior izqui
figure,imagesc(log(1+abs(fftshift(fft2(im,M,N))))),colorbar  %Visualizo transformada de la imag
```




```
figure,imagesc(log(1+abs(fftshift(NIM)))),colorbar   %Visualizo transformada de la imagen filtr
```



```
nim=real(ifft2(NIM)); %Hago transformada inversa para recuperar la imagen filtrada en el domini
nim=nim(1:filas_im,1:1:columnas_im);figure,imshow(nim) %parte imagen filtrada con tamaño = imag
```



En el proceso anterior cabe destacar como el píxel central de una imagen de tamaño MxN se sitúa en la celda [fila,columna] = [floor((M/2)+1),floor((N/2)+1)].

La línea:

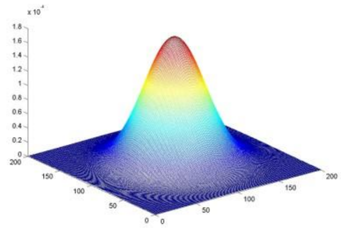
```
D=sqrt((p-floor((N/2)+1)).^2+(q-floor((M/2)+1)).^2)<=(0.2*M);%Radio filtrado: 0.2*M. Fija frecu
```

pone a 1 los píxeles cuya distancia al píxel central de frecuencia (0,0), es menor de un cierto valor (frecuencia de corte: en este caso 0,2*M). Esto define un círculo de selección alrededor del píxel central.

Por último, cabe destacar como, en el código implementado, hay funciones (como *meshgrid*) que trabajan con coordenadas cartesianas (primera componente x aumenta hacia la derecha e y hacia abajo) y otras en coordenadas (filas,columnas) (como fft2) donde la primera componente (filas) aumenta hacia abajo y la segunda (columnas) hacia la derecha.

Filtro paso bajo gaussiano

En este caso, la ecuación que define la respuesta en frecuencia del filtro sería: $H(p,q) = e^{-k\left(p^2+q^2\right)}$

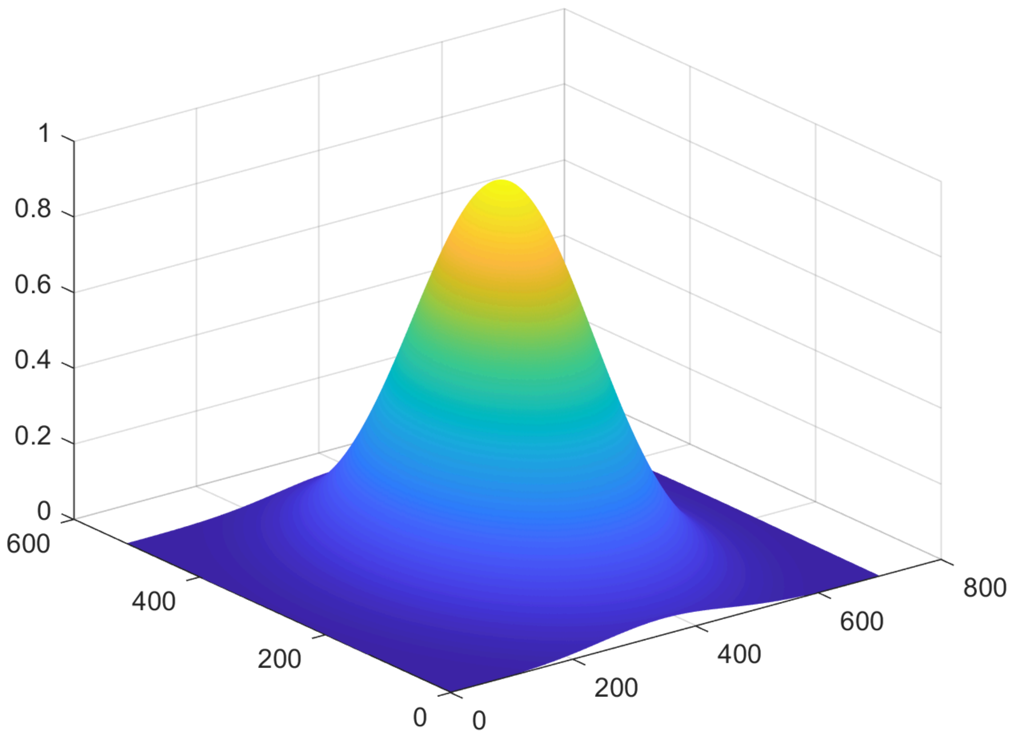


Como se observa en la figura, el filtro da salida 1 para la frecuencia (p,q) = (0,0) y disminuye hasta alcanzar el valor de amplitud 0, de forma gradual, con forma de campana de gauss.

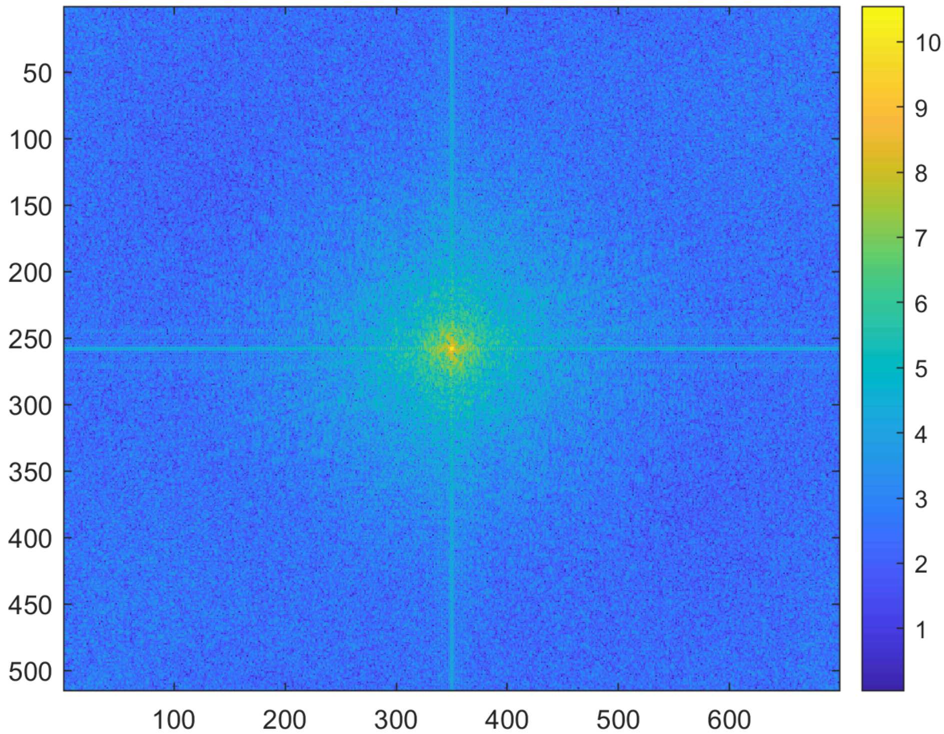
Ejecute, ponga título a las gráficas mostradas y comente el siguiente código utilizando diferentes frecuencias de corte relacionadas con la anchura de la campana gausiana (mostrada en negrita en el ejemplo con un valor de 0.2*M):

```
load trees; im=ind2gray(X,map);figure,imshow(im)
```

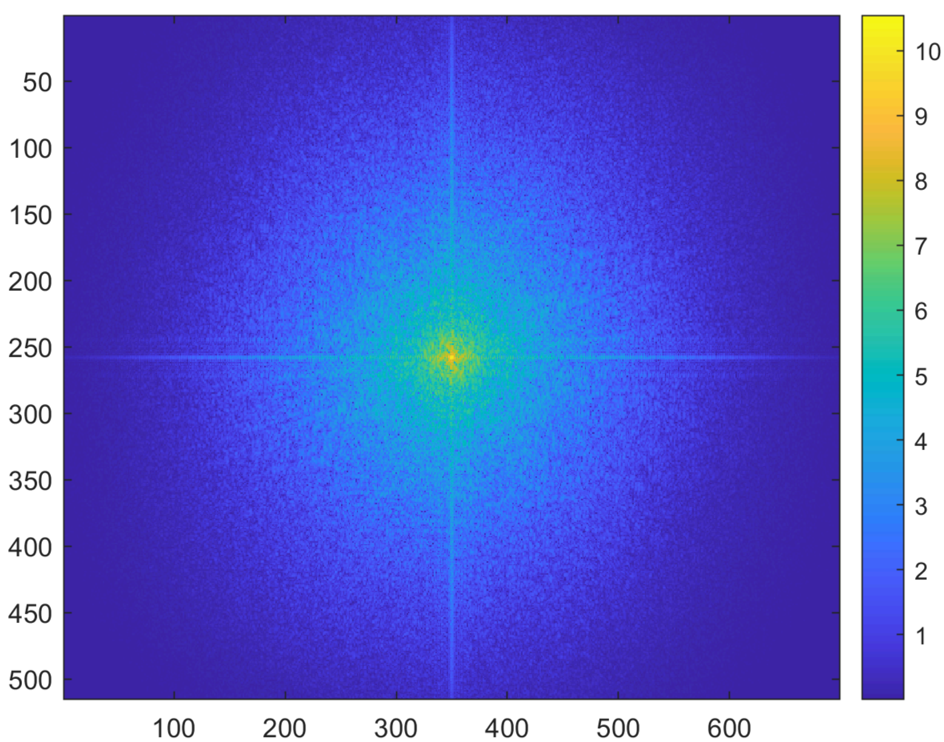
```
[filas_im,columnas_im]=size(im);
N=(2*columnas_im)-1;
M=(2*filas_im)-1;
pp=1:N; qq=1:M;[p,q]=meshgrid(pp,qq);
D=sqrt((p-floor((N/2)+1)).^2+(q-floor((M/2)+1)).^2); %Distancia de cada pixel al central de fre
k=1/(2*((0.2*M)^2)); %Parametro relacionado con la anchura de la campana de gauss
H=exp(-k.*(D.^2)); %Implementa la función gaussiana con relacion al pixel central de la imagen
figure,mesh(p,q,H);
```




```
NIM=fft2(im,M,N).*fftshift(H);
figure,imagesc(log(1+abs(fftshift(fft2(im,M,N))))),colorbar
```




```
figure,imagesc(log(1+abs(fftshift(NIM)))),colorbar
```



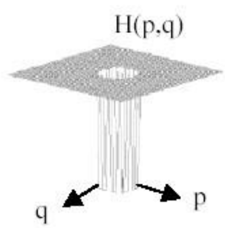
```
nim=real(ifft2(NIM));
nim=nim(1:filas_im,1:1:columnas_im);figure,imshow(nim)
```

Filtro paso alto ideal

En las frecuencias (p,q) situadas en un radio alrededor de la frecuencia más baja (0,0) el filtro da salida 0, en el resto da salida 1.

Ejecute, poniendo título a las gráficas mostradas, el siguiente código utilizando diferentes frecuencias de corte, además de la puesta en el ejemplo de 0.02*M.

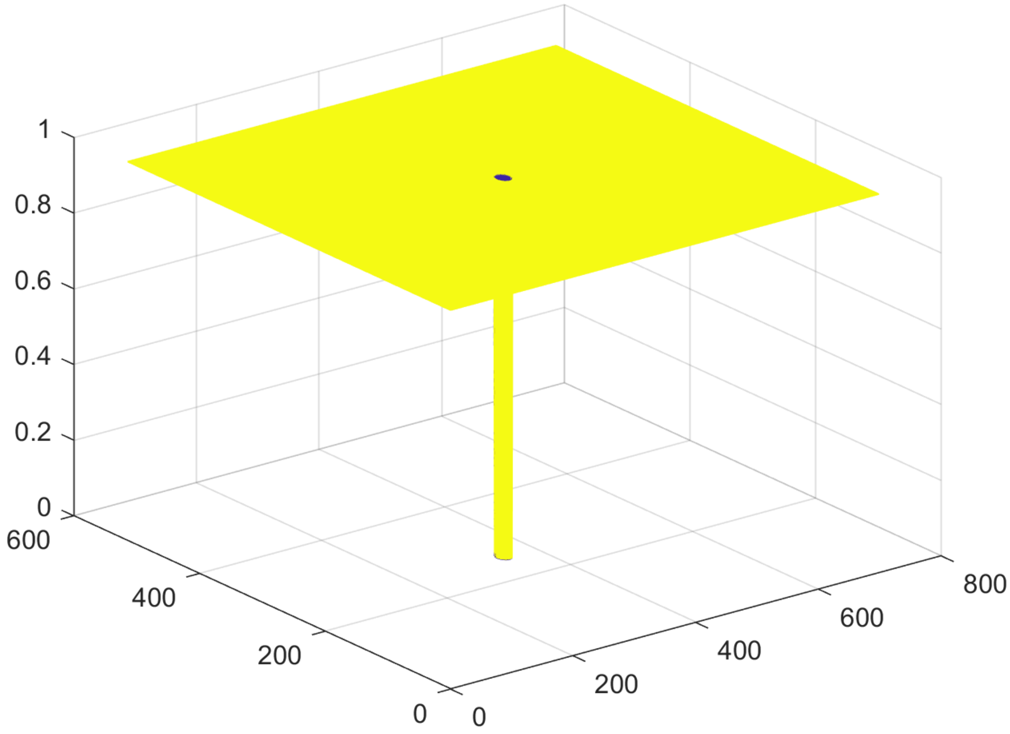
$$H(p,q)=\begin{cases}1 & \sqrt{p^2+q^2}\geq Ho\\ 0 & resto\end{cases}$$



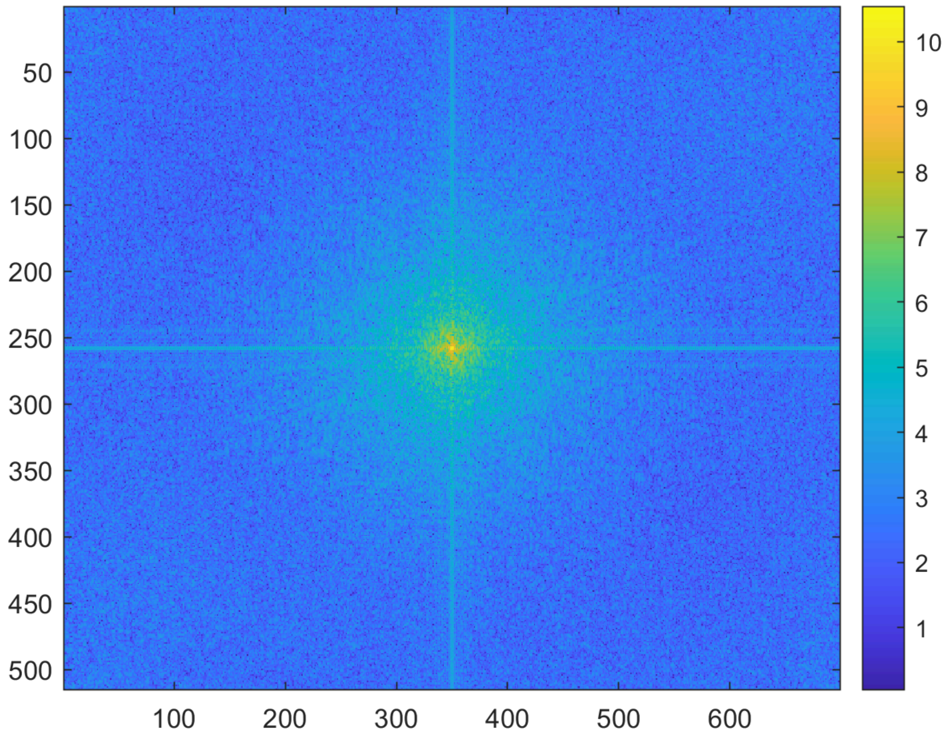

```
close all
load trees; im=ind2gray(X,map);figure,imshow(im)
```

```
[filas_im,columnas_im]=size(im);
N=(2*columnas_im)-1; %Tamano columnas de la transformada
M=(2*filas_im)-1;    %Tamano filas de la transformada
pp=1:N; qq=1:M;[p,q]=meshgrid(pp,qq);
```

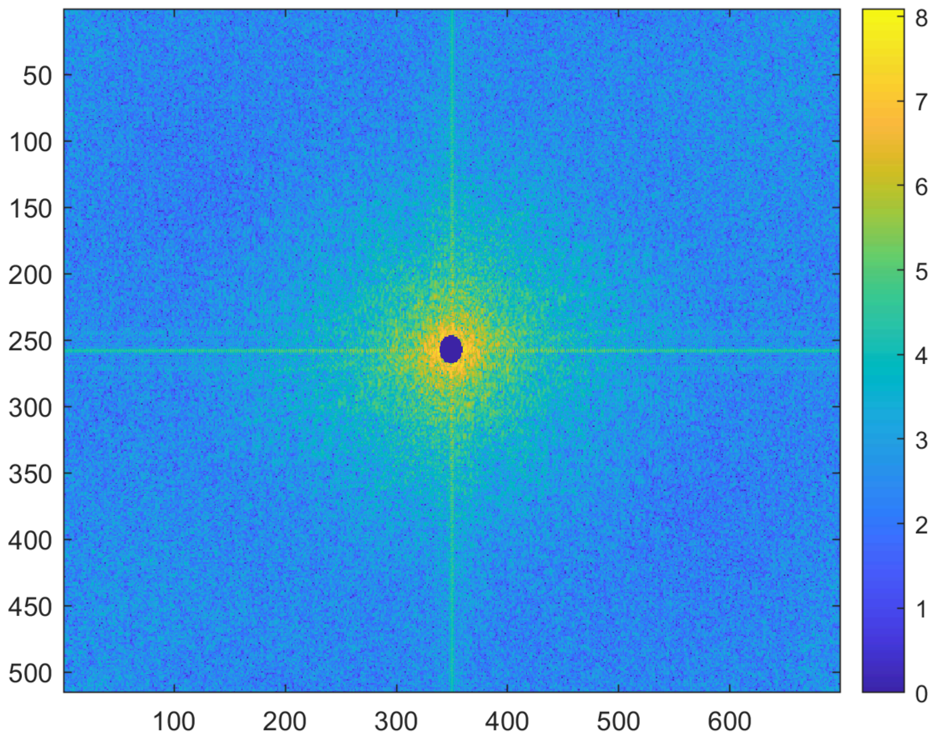
```
%p,q definen las MxN posibles frecuencias del filtro
D=sqrt((p-floor((N/2)+1)).^2+(q-floor((M/2)+1)).^2)<=(0.02*M);%Radio filtrado. Fija frecuencia
%Cojo frecuencias centrales alrededor del píxel central de la imagen situado en: floor((N/2)+1)
H=ones(M,N);
H(D)=0; %Pongo a 0 las frecuencias centrales del filtro. El resto a 1
figure,mesh(p,q,H);
```




```
NIM=fft2(im,M,N).*fftshift(H);
%Multiplico la transformada de Fourier de la imagen (con F(0,0) en esquina superior izquierda .
%...con el filtro (desplazado para que frecuencia 0,0 del filtro este en esquina superior izqui
figure,imagesc(log(1+abs(fftshift(fft2(im,M,N))))),colorbar  %Visualizo transformada de la imag
```




```
figure,imagesc(log(1+abs(fftshift(NIM)))),colorbar   %Visualizo transformada de la imagen filtr
```



```
nim=real(ifft2(NIM)); %Hago transformada inversa para recuperar la imagen filtrada en el domini
nim=nim(1:filas_im,1:1:columnas_im);figure,imshow(nim)
```

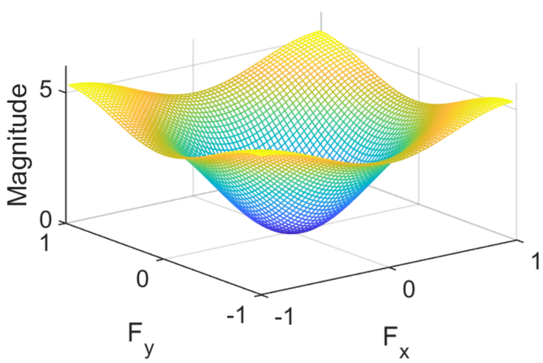
Respuesta en frecuencia.

La función freqz2 calcula la respuesta en frecuencia para un filtro bidimensional. Sin argumentos de salida, freqz2 crea un trazado en red de la respuesta en frecuencia. Por ejemplo, considerar el filtro FIR,

```
h = [0.1667   0.6667   0.1667
     0.6667  -3.3333   0.6667
     0.1667   0.6667   0.1667];
```

Para calcular y visualizar la respuesta en frecuencia 64x64 de h se utilizará:

```
freqz2(h)
```



Para obtener la matriz de respuesta en frecuencia H y los vectores de los puntos de frecuencia w1, w2, se usan los argumentos de salida:

```
[H,w1,w2]=freqz2(h);
```

freqz2 normaliza las frecuencias w1 y w2, donde la frecuencia de Nyquist es ±0.5 para el caso bidimensional.

Para una respuesta simple mxn, como se muestra arriba, *freqz2* utiliza la función bidimensional de la transformada rápida de Fourier *fft2*. También se puede especificar vectores de puntos de frecuencias arbitrarias, pero en este caso *freqz2* utiliza un algoritmo más lento.