

Procesamiento de Imágenes

Ingeniería Biomédica

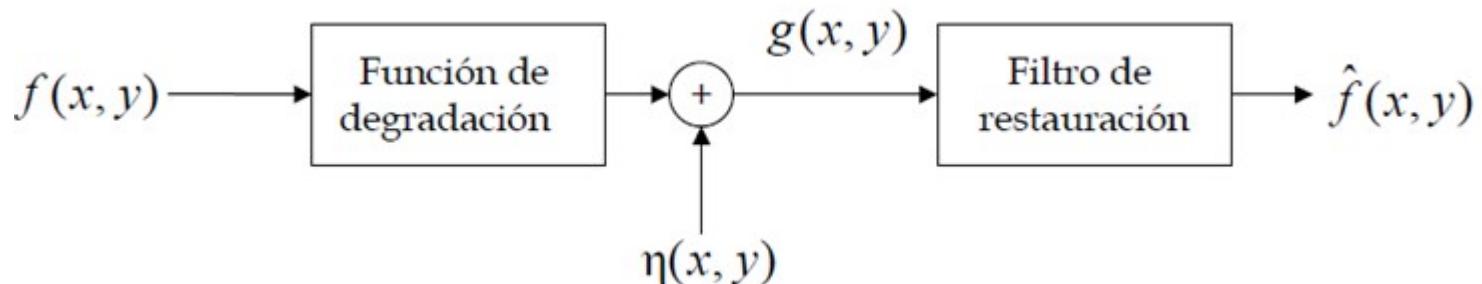
Unidad 5: Degradación y restauración de imágenes

- Modelo del proceso de degradación/restauración de una imagen.
- Optical Transfer Function (OTF) y Point Spread Function (PSF)
- Modelos de ruido:
 - a) Gaussiano b) Rayleigh c) Exponencial
 - d) Uniforme e) Impulsivo
- Restauración por filtrado espacial:
 - 1) Filtros de medias 2) Filtros de orden 3) Filtros adaptativos
- Restauración por filtrado frecuencial:
 - 1) Rechaza banda 2) Pasa Banda 3) Notch
- Filtrado inverso
- Filtrado de Wiener

Degradación

Si la imagen $f(x, y)$ es afectada por una *función de degradación* $h(x, y)$ y por *ruido aditivo* $\eta(x, y)$, la imagen resultante $g(x, y)$ podrá expresarse como:

$$g(x, y) = h[f(x, y)] + \eta(x, y)$$



Restauración

Consiste en obtener una estimación $\hat{f}(x, y)$ de la imagen original a partir de $g(x, y)$, teniendo algún conocimiento sobre $h(x, y)$ y $\eta(x, y)$.

Si la degradación es un *proceso lineal e invariante en el espacio*, entonces:

Dominio espacial $\rightarrow g(x, y) = f(x, y) \otimes h(x, y) + \eta(x, y)$

Dominio frecuencial $\rightarrow G(u, v) = F(u, v) \cdot H(u, v) + N(u, v)$

Degradación: ruido, desenfoque, distorsiones geométricas, movimiento ...

Restauración: se busca modelizar el fenómeno de degradación y aplicar *procesos inversos* para recuperar la imagen original.

- $H(u,v)$ también es conocida con el nombre de *Optical Transfer Function (OTF)*

- *Respuesta al impulso* $h(x,y)$ también conocida con el nombre *Point Spread Function (PSF)*

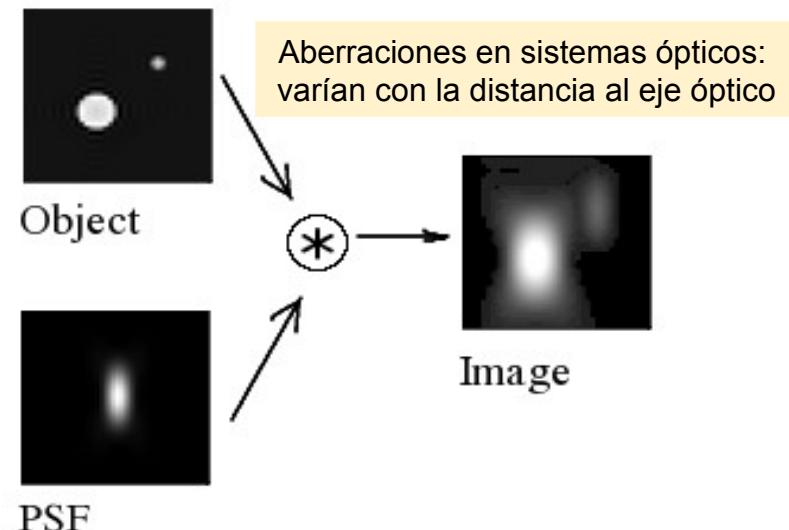
Podemos convertir de una forma a otra usando los comandos: `otf2psf` y `psf2otf`

- El módulo de OTF es conocido como *Modulation Transfer Function (MTF)*

Observación: en Astronomía las estrellas son consideradas objetos puntuales (delta de Dirac) y su imagen representa la PSF del sistema.

Abordaje del problema:

- Primero asumiremos que la degradación es el *operador identidad* (es decir, $H=1$) y estudiaremos solamente el ruido.
- Luego estudiamos la degradación H (*sin ruido, N=0*)
- Finalmente estudiaremos la restauración en presencia de ambos componentes.



Modelos de ruido

Fuentes de ruido:

- ❑ *Adquisición:* defectos en los sensores, bajos niveles de luz, CCD sensibles a la temperatura, etc.
- ❑ *Transmisión:* interferencias en el canal de transmisión (variaciones de las condiciones atmosféricas, ej. tormentas).

Asumimos que:

- ❖ El ruido es independiente de las coordenadas espaciales (**invarianza espacial**), salvo que sea ruido periódico.
- ❖ El ruido queda descripto mediante variables estadísticas, en particular por su **función de densidad de probabilidad, $p(z)$**

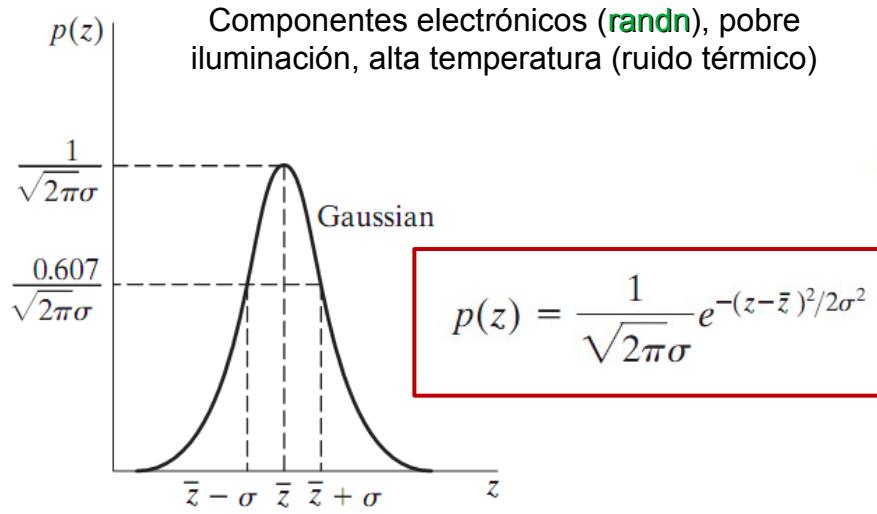
Ruido aleatorio

- ❑ Existe la necesidad de generar otros tipos de ruido, además de aquellos disponibles con el comando imnoise.
- ❑ Generaremos entonces números aleatorios caracterizados por su **$p(z)$**

Modelos de ruido más comunes

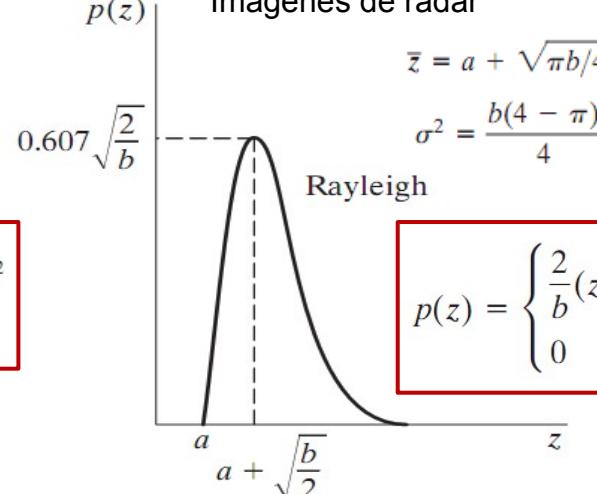
Ruido Gaussiano (normal)

Componentes electrónicos (`randn`), pobre iluminación, alta temperatura (ruido térmico)



Ruido Rayleigh

Imágenes de radar

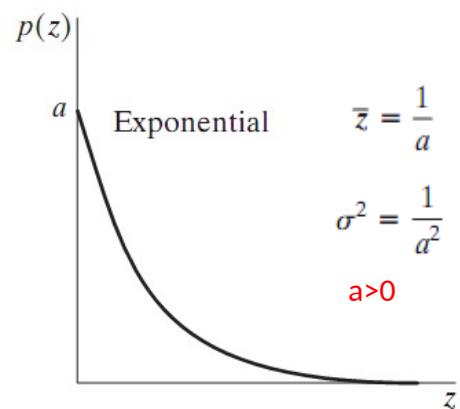


p(z) pdf
z variable aleatoria
z̄ valor medio
σ desvío estándar
σ² varianza.

$$p(z) = \begin{cases} \frac{2}{b}(z - a)e^{-(z-a)^2/b} & \text{for } z \geq a \\ 0 & \text{for } z < a \end{cases}$$

Ruido exponencial

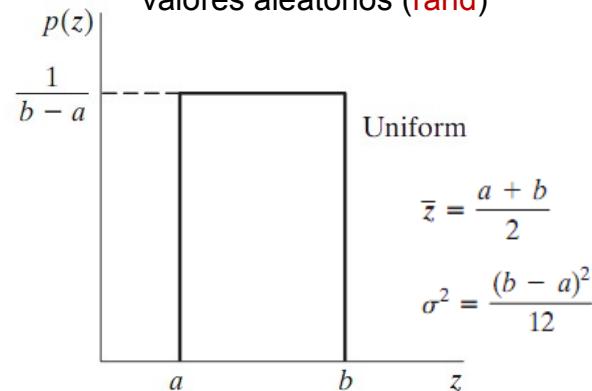
Imágenes generadas por láser.



$$p(z) = \begin{cases} ae^{-az} & \text{for } z \geq 0 \\ 0 & \text{for } z < 0 \end{cases}$$

Ruido uniforme

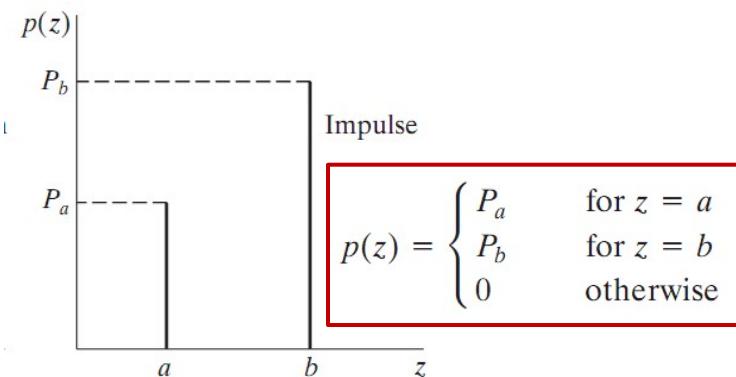
Utilizado para simulación de valores aleatorios (`rand`)



$$p(z) = \begin{cases} \frac{1}{b-a} & \text{if } a \leq z \leq b \\ 0 & \text{otherwise} \end{cases}$$

Ruido impulsivo (salt & pepper)

Defectos en CCD, errores de transmisión, ruido externo que contamina la conversión A/D, etc.

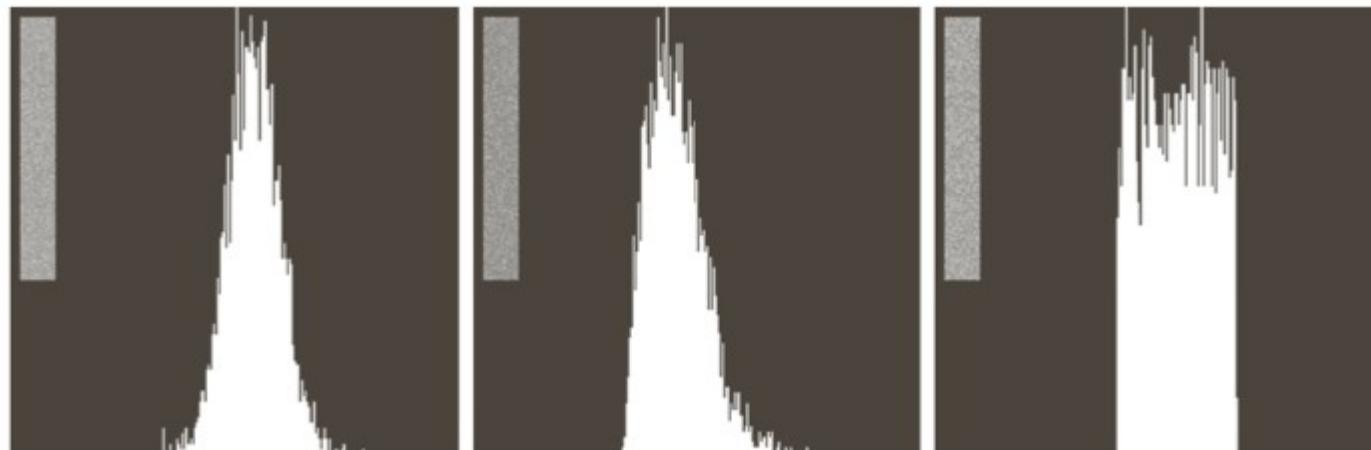


Casos particulares:

- Ruido Bipolar: $P_a \approx P_b$
- Ruido Unipolar: $P_a = 0$ o $P_b = 0$

Estimación de los parámetros del ruido

- Se estiman observando la DFT2D de la imagen degradada (ej. ruido periódico)
- Los parámetros de la PDF del ruido pueden determinarse a partir de las especificaciones del sensor.
- Tomar una imagen de una placa gris uniformemente iluminada, permite obtener la PDF del ruido.
- Si sólo se tiene imágenes la PDF se puede obtener de alguna sección de gris constante en la imagen.
- En las figuras se aprecia que las formas de los histogramas corresponden en forma muy cercana a las formas de los histogramas vistos.



Ruido Gaussiano

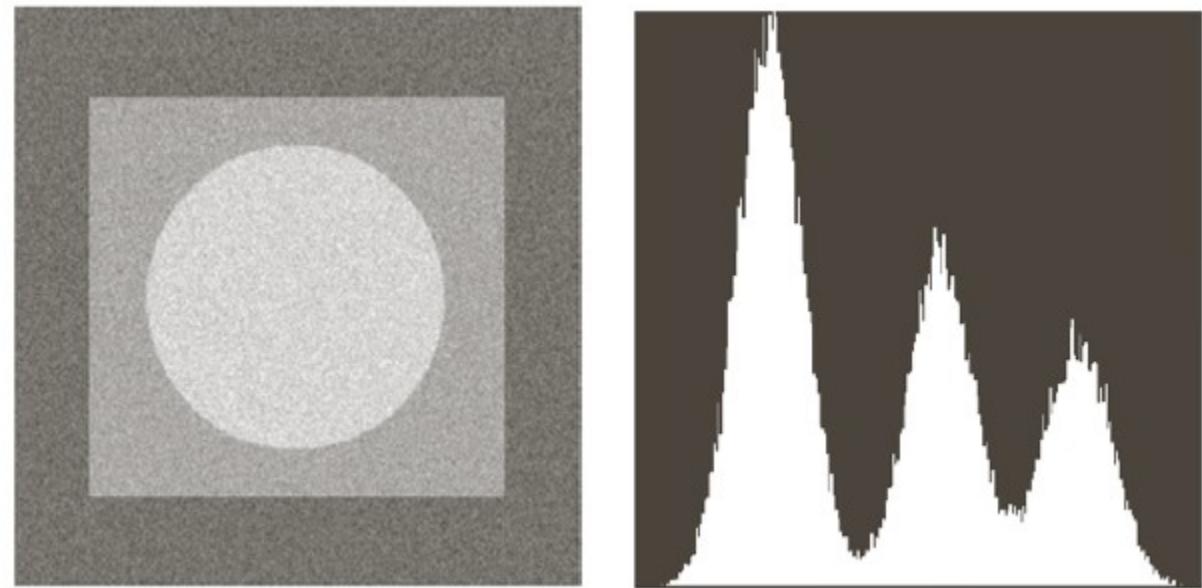
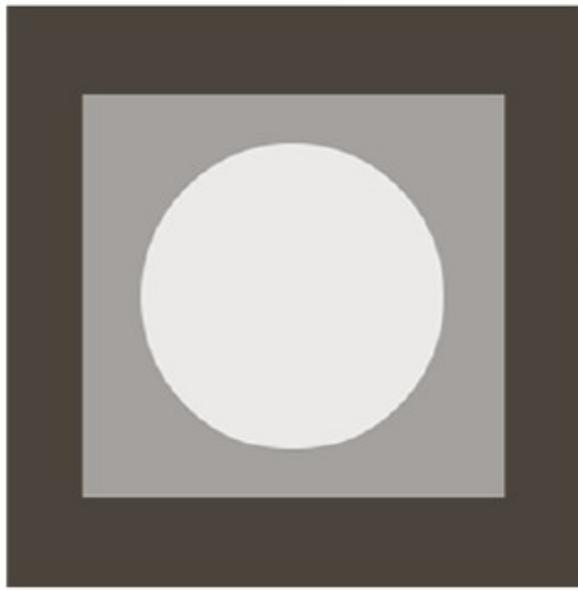
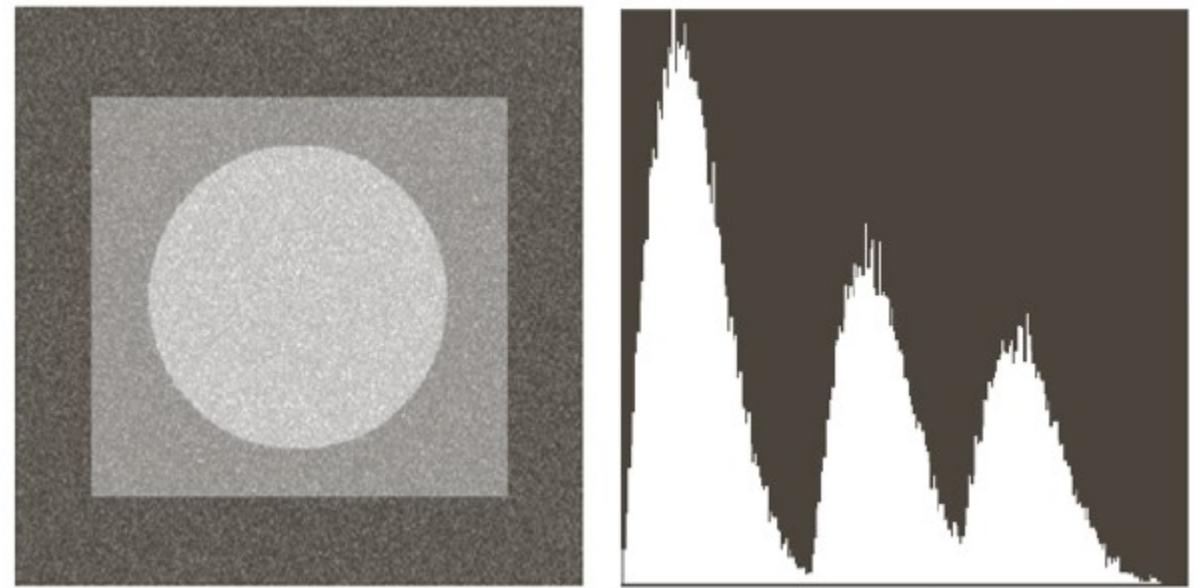


Imagen original



Ruido Rayleigh



Ruido Exponencial

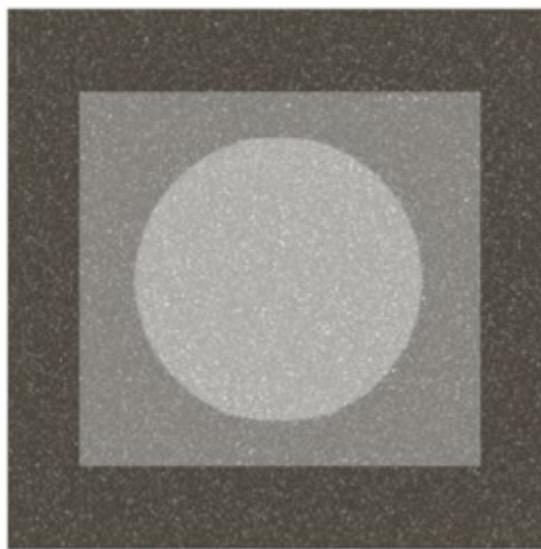
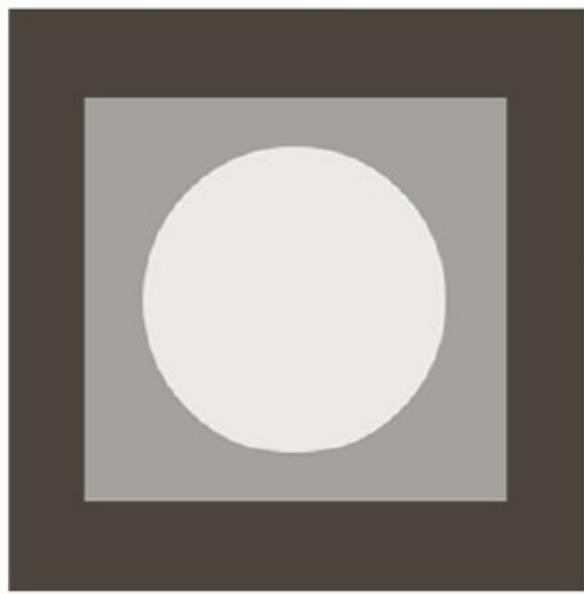
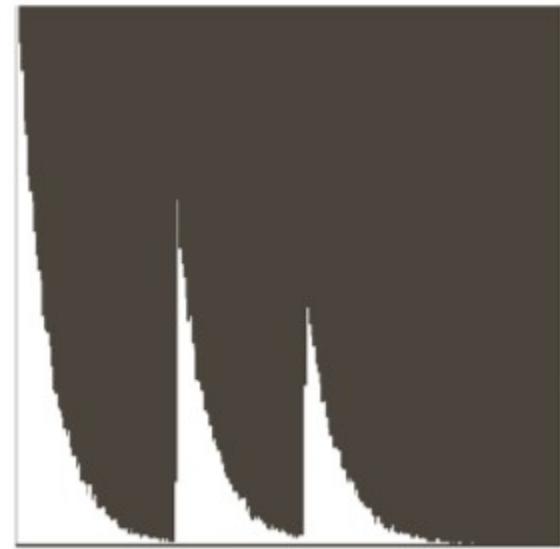
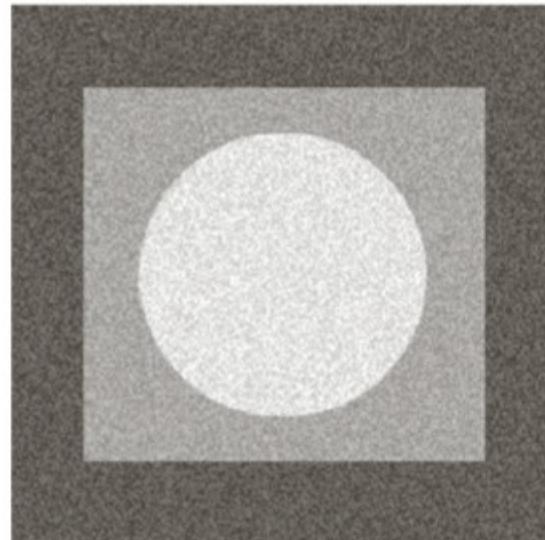


Imagen original



Ruido Uniforme



Generador de ruido aleatorio

```
switch lower(type)
    case 'gaussian'
        R = a + b * randn(M,N);
    case 'rayleigh'
        R = a + sqrt(-b * log(1-rand(M,N))); %b>0
    case 'exponential'
        % a debe ser positivo
        R = (-1/a)*log(1-rand(M,N));
    case 'uniform'
        R = a +(b - a)*rand(M,N);
    case 'salt & pepper'
        R = ruido_sal_pimienta(M,N,Pa,Pb);
end
```

randn: genera números aleatorios con distribución normal $N(0,1)$ o Gaussiana, media=0, desvío=1

a = media b = desvío

rand: genera números aleatorios distribuidos uniformemente en el intervalo [0,1]

```
function R = ruido_sal_pimienta (M,N,Pa,Pb)
% Genera una matriz R (MxN) de 0 (con Pa) pimienta, 1 (con Pb) sal y 0.5 (P=1-(Pa+Pb))
% Pa = probabilidad de pixeles 0, Pb = probabilidad de pixeles 1, P = probabilidad pixeles 0.5
```

```
if (Pa + Pb) > 1 %Debo chequear que Pa+Pb<1.
    error( 'La suma de probabilidades no debe exceder 1')
end
```

```
R = ones(M,N)*0.5;
X = rand(M,N) ;
R(X <= Pa)=0; %pimienta
P = Pa + Pb;
R(X > Pa & X <= P) = 1; %sal
```

Ejemplo: Pa= 0.2, Pb=0.7, MxN=3x3

>> X=rand(3)			>> R(X<=0.2)=0			>> R(X>0.2 & X <0.9)=1		
R =	X =	R =	R =					
0.5 0.5 0.5	0.964 0.957 0.141	0.5 0.5 0	0.5 0.5 0					
0.5 0.5 0.5	0.157 0.485 0.421	0 0.5 0.5	0 1.0 1.0					
0.5 0.5 0.5	0.970 0.800 0.915	0.5 0.5 0.5	0.5 1.0 0.5					

```

%Genero Ruido Aleatorio de diferente PDF
I=imread('cameron.tif');
I=im2double(I); %convierito a clase double
[M,N]=size(I);
In = I;

%ruido sal y pimienta, con igual probabilidad Pa=Pb=0.02
R=ruido_sal_pimienta(M,N,0.02,0.02);
ix1=find(R==0); In(ix1)=0;
ix2=find(R==1); In(ix2)=1;

%ruido pimienta (no hay sal)
R=ruido_sal_pimienta(M,N,0.02,0);
ix=find(R==0); In(ix)=0;

%ruido sal (no hay pimienta)
R=ruido_sal_pimienta(M,N,0,0.02);
ix=find(R==1); In(ix)=1;

%..... para los demás ruidos, In se calcula haciendo:
In=I+R; %agrego el ruido a la imagen de entrada
In=max(min(In,1),0); %clampeo al rango [0,1]

```

Ruido sal y pimineta, $Pa=Pb=0.02$



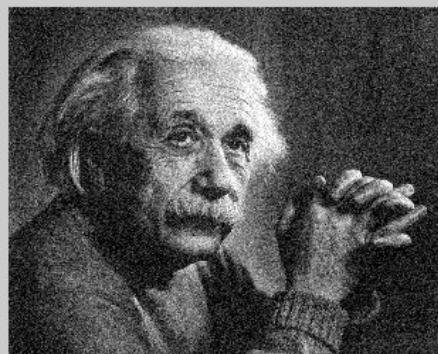
Ruido pimienta, $Pa=0.02$ y $Pb=0$



Ruido sal, $Pa=0$ y $Pb=0.02$



Ruido Gaussiano, $\mu=0$ $\sigma^2=0.01$



Histograma ruido Gaussiano

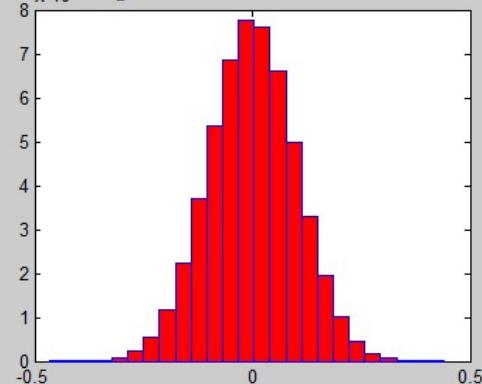
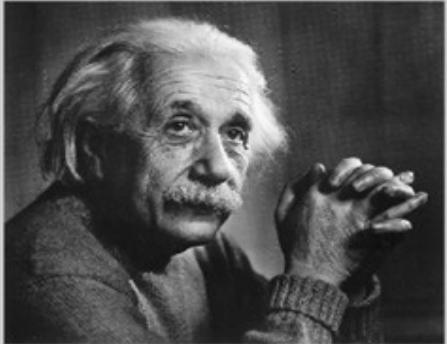


Imagen original



Ruido uniforme, $a=0.2$ y $b=0.8$



Histograma ruido uniforme

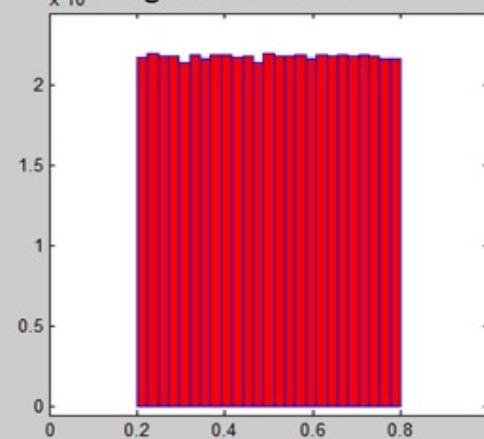
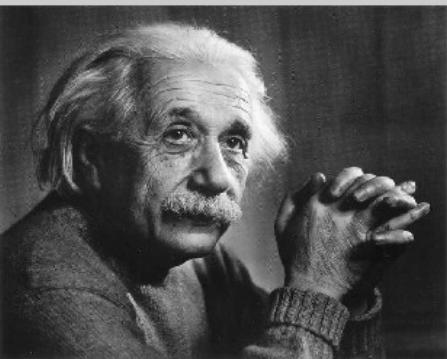


Imagen original



Ruido Rayleigh



Histograma ruido Rayleigh

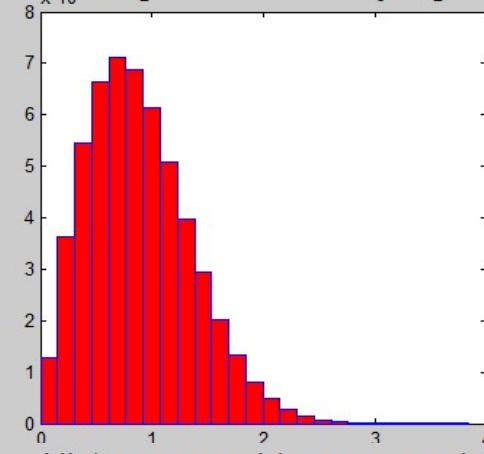
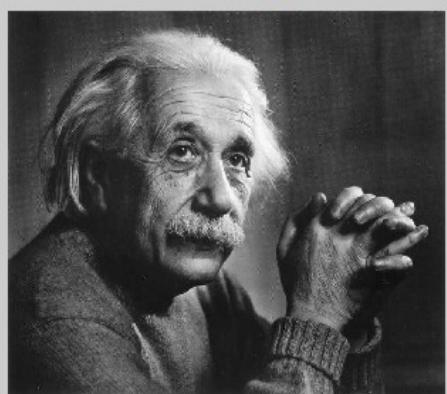


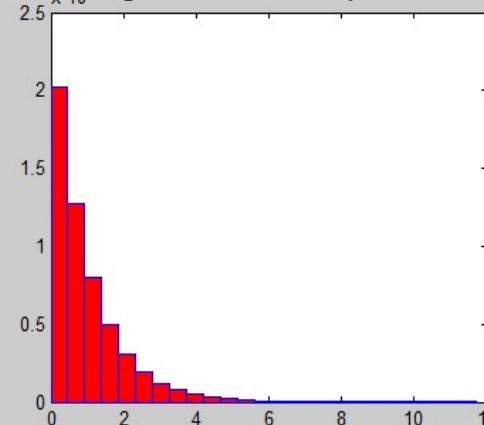
Imagen original



Ruido exponencial



Histograma ruido exponencial



Restauración mediante filtrado espacial

Cuando la degradación es sólo ruido ($H = 1$)

$$g(x, y) = f(x, y) + \eta(x, y)$$

$$G(u, v) = F(u, v) + N(u, v)$$

- Generalmente **no se conoce** la forma de N (salvo cuando el ruido es periódico), por lo que la operación $G(u, v) - N(u, v)$ no es posible.
- Para restaurar utilizaremos el **filtrado espacial**, empleando diferentes tipos de filtros (lineales y no lineales): **filtros de medias, filtros de orden, filtros adaptativos**

Filtros de medias

- **Media aritmética:** reduce las variaciones locales (borronea/difumina bordes). Se convoluciona la imagen con una máscara de coeficientes $1/mn$

```
%Filtro promediador de dimensión mxn  
H = fspecial ('average',[m n])  
O = conv2(I,H,'same');
```

$$\hat{O}(x, y) = \frac{1}{mn} \sum_{(s,t) \in S_{xy}} I(s, t)$$

$$H = \begin{matrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{matrix}$$

Imagen original

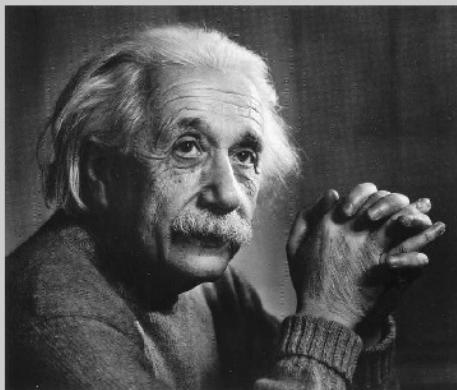


Imagen con ruido sal

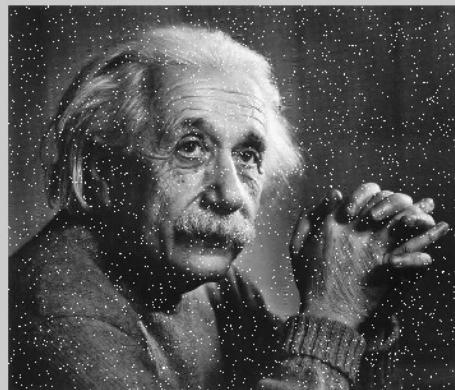
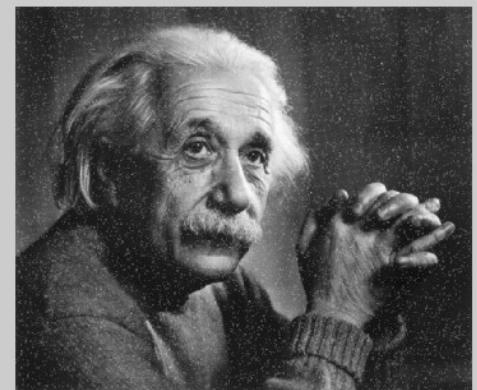


Imagen filtrada, media aritmética



Filtros no-lineales

- Se basan en operaciones no lineales involucrando píxeles en una vecindad.
- Comando **colfilt**: organiza los datos dentro de la ventana móvil en columnas.
- Dada una imagen degradada por ruido, de dimensión (FxC), y una vecindad (mxn), colfilt genera una matriz A (mnxFc). Cada columna corresponde a los píxeles abarcados por la vecindad.

```
O = colfilt (I, [m n], 'sliding', @fun, parameters)
```

'**sliding**' indica que el proceso a realizar es el **desplazamiento** de la ventana mxn sobre cada pixel de la imagen de entrada, **@fun** es un puntero a la función a emplear, **parameters** son los **parámetros** (separados por coma) requeridos por la función. Obs: colfilt efectúa **zero padding** cuando es necesario.

- **Media geométrica**: suaviza la imagen con menor pérdida de detalles que la media aritmética. Apropiado para el **ruido Gaussiano**, pero falla con el ruido impulsivo.

$$\hat{O}(x, y) = \left[\prod_{(s,t) \in S_{xy}} I(s, t) \right]^{\frac{1}{mn}}$$

```
function v = media_geometrica(A)
mn = size(A, 1);
v = prod(A) .^(1/mn);
```



```
O = colfilt (In, [m n], 'sliding', @media_geometrica);
```

- **Media armónica:** apropiado para **ruido sal**, falla para ruido pimienta. Trabaja bien con **ruido Gaussiano**, preservando detalles de la imagen.

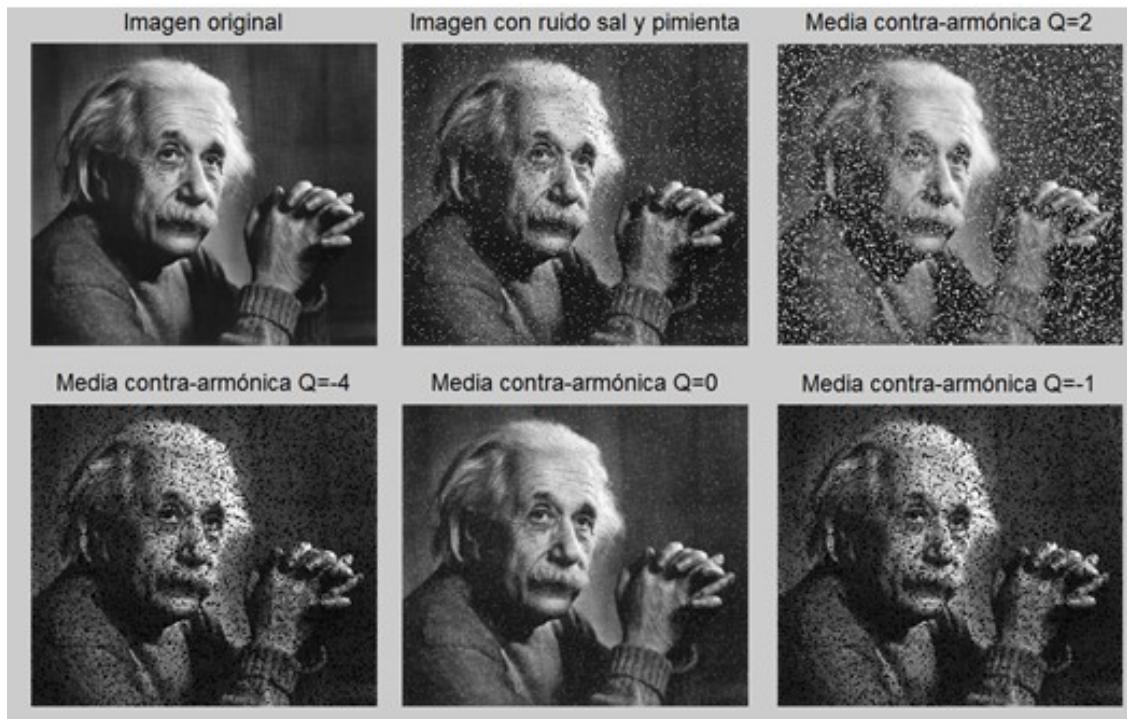
$$\hat{O}(x, y) = \frac{mn}{\sum_{(s,t) \in S_{xy}} \frac{1}{I(s, t)}}$$

```
function v = media_armonica(A)
mn = size(A, 1);
v = mn ./ sum(1 ./ (A+eps));
```



O = colfilt(In, [m n], 'sliding', @media_armonica);

- **Media contra-armónica:** apropiado para **ruido sal y pimienta**. **Q** es el **orden** del filtro, si **Q>0** elimina pimienta, si **Q<0** elimina la sal, si **Q=0** media aritmética, **Q=-1** media armónica.



$$\hat{O}(x, y) = \frac{\sum_{(s,t) \in S_{xy}} I(s, t)^{Q+1}}{\sum_{(s,t) \in S_{xy}} I(s, t)^Q}$$

```
function v=media_contra_armonica(A, Q)
v=sum(A.^ (Q+1)) ./ sum(A.^ Q);
```

O=colfilt(In, [m n], 'sliding', @media_contra_armonica, Q);

Filtros de orden

- ❑ **Filtro mediana:** selecciona el valor medio de los valores en la vecindad. Buena reducción del **ruido impulsivo** sin el desenfoque de un filtro lineal de la misma dimensión.

$$\hat{O}(x, y) = \text{mediana}_{(s,t) \in S_{xy}}\{I(s, t)\}$$

O = **medfilt2 (I, [m n])** donde [m n] es el tamaño de la ventana de análisis, por defecto es 3x3



- ❑ **Filtro de máxima:** selecciona el valor máximo de la vecindad. Útil para **ruido pimienta**.

$$\hat{O}(x, y) = \text{máximo}_{(s,t) \in S_{xy}}\{I(s, t)\}$$

```
function v = filtro_maximo(A)
v = max(A);
```

O=colfilt(In, [m n], 'sliding', @filtro_maximo);



- ❑ **Filtro de mínima:** selecciona el valor mínimo de la vecindad. Útil para **ruido sal**.

$$\hat{O}(x, y) = \min_{(s,t) \in S_{xy}} \{I(s, t)\}$$

```
function v = filtro_minimo(A)
v = min(A);
```

```
O=colfilt(In, [m n], 'sliding', @filtro_minimo);
```



- ❑ **Filtro de punto medio:** selecciona el promedio entre el máximo y el mínimo de la vecindad. Útil para **ruido Gaussiano o uniforme**.

$$\hat{O}(x, y) = \frac{1}{2} \left[\min_{(s,t) \in S_{xy}} \{I(s, t)\} + \max_{(s,t) \in S_{xy}} \{I(s, t)\} \right]$$

```
O=colfilt(In, [m n], 'sliding', @filtro_punto_medio);
```



```
function v=filtro_punto_medio(A)
v1 = max(A);
v2 = min(A);
v = (v1+v2)/2;
```

- ❑ **Filtro de la media alfa-recortado:** Los valores $d/2$ más bajos y $d/2$ más altos son eliminados de la vecindad, I_r denota los píxeles remanentes ($mn-d$) en la vecindad. Se calcula el promedio en esa vecindad recortada de valores.

$$\hat{O}(x, y) = \frac{1}{mn - d} \sum_{(s,t) \in S_{xy}} I_r(s, t)$$

Comportamiento entre la media aritmética y la mediana, dependiendo del valor de d . Útil para combinaciones de **ruido Gaussiano y sal y pimienta**.



Filtros adaptativos

- ❑ Cambian su comportamiento según ciertos **parámetros estadísticos** de los valores dentro de la ventana de análisis.
 - ❑ Ventaja: **desempeño** superior al de los filtros probados previamente.
 - ❑ Desventaja: se incrementa la **complejidad** del filtro.
- ❖ **Filtro adaptativo de reducción local de ruido:**

$$\hat{O}(x, y) = I(x, y) - \frac{\sigma_\eta^2}{\sigma_L^2} [I(x, y) - \mu_L]$$

Media: intensidad promedio en una región
Varianza: medida del contraste en la región

- Se calcula la media y la varianza local (es decir, de los valores dentro de la ventana de análisis), y la varianza del ruido.
- Si el ruido es nulo, $\sigma_\eta^2 = 0$, la salida será la **imagen original**.
- Si $\sigma_L^2 \approx \sigma_\eta^2$ el filtro devuelve la **media local** μ_L

Obs: La ventana tiene las mismas propiedades que toda la imagen, el ruido local se reducirá simplemente por promediación.

- Si $\sigma_L^2 \gg \sigma_\eta^2$ la salida es aprox. la imagen original (varianza local alta, asociada a bordes y detalles)
- Se asume que $\sigma_\eta^2 \leq \sigma_L^2$ Si no se conoce la varianza del ruido, debe estimarse.

Imagen original

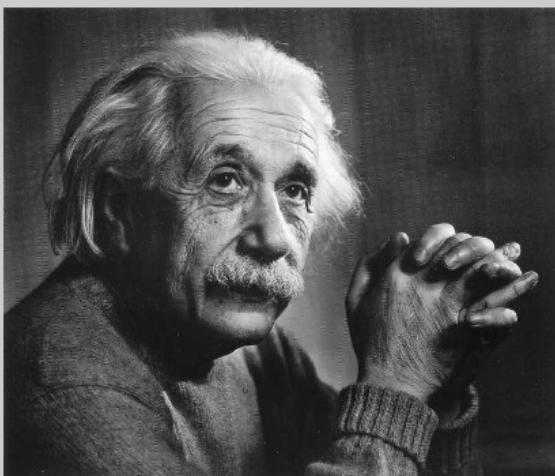
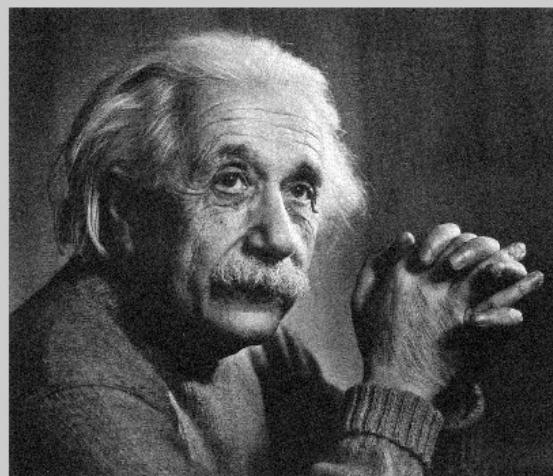
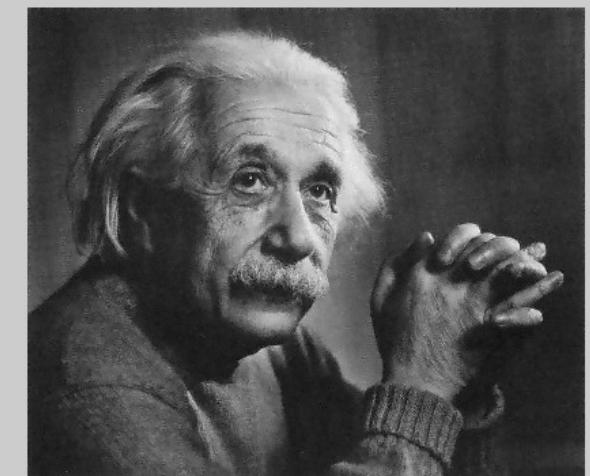


Imagen con ruido Gaussiano, $\mu=0$ $\sigma=0.05$



Filtro adaptativo de reducción local de ruido



- ❖ **Filtro adaptativo de mediana:** puede manejar **ruido impulsivo** con densidad de puntos muy grandes (P_a y $P_b > 0.2$). Busca preservar el detalle mientras que suaviza el ruido que no es impulsivo.

S_{xy} vecindad entorno a un pixel

Z_{min} mínima intensidad en la ventana S_{xy}

Obs: no usamos colfilt, usamos ordfilt2

Z_{max} máxima intensidad en la ventana S_{xy}

Z_{med} mediana de los valores en S_{xy}

Z_{xy} valor de intensidad en las coordenadas (x,y)

S_{max} tamaño máximo permitido de la ventana de filtrado adaptativo

El algoritmo de filtrado trabaja en dos niveles (A y B):

Nivel A: Si $Z_{min} < Z_{med} < Z_{max}$, ir al nivel B, sino, incrementar el tamaño de la ventana. Si el tamaño de la ventana es $\leq S_{max}$, repetir esta etapa, sino la salida será Z_{med}

Nivel B: Si $Z_{min} < Z_{xy} < Z_{max}$, la salida es Z_{xy} , sino, la salida es Z_{med}

Imagen original

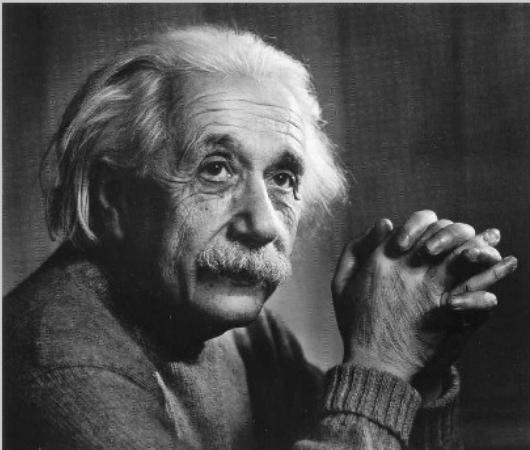
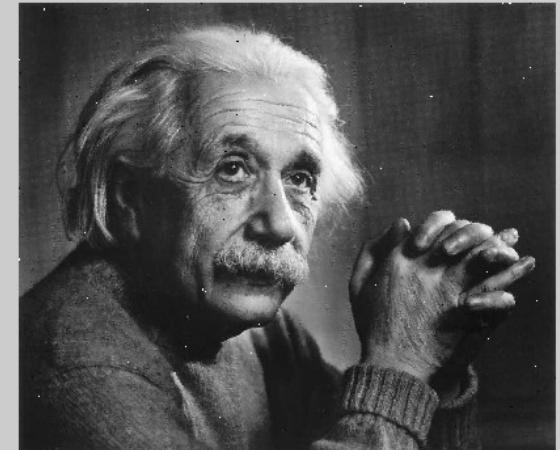


Imagen con ruido sal y pimienta ($P_a=P_b=0.2$)



Imagen filtrada, adaptativo de mediana



Restauración mediante filtros frecuenciales

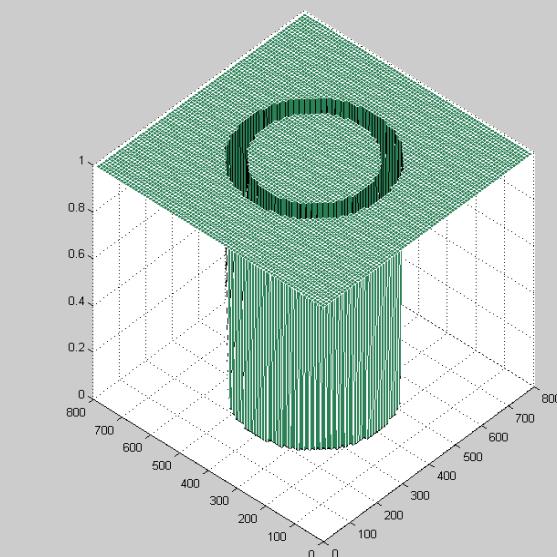
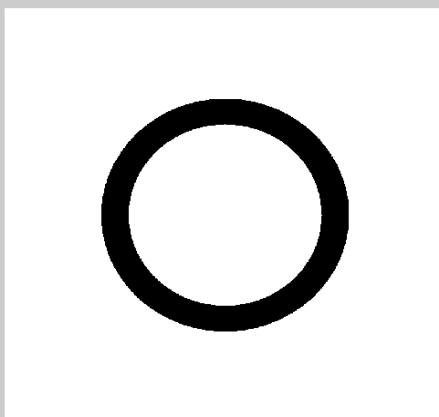
- ❑ **Filtro rechaza banda:** elimina o atenúa una banda de frecuencias alrededor del origen del espectro. Hay de distintos tipos:

Rechaza banda ideal:

$$H(u, v) = \begin{cases} 0 & \text{si } D_0 - \frac{w}{2} \leq D(u, v) \leq D_0 + \frac{w}{2} \\ 1 & \text{otro caso} \end{cases}$$

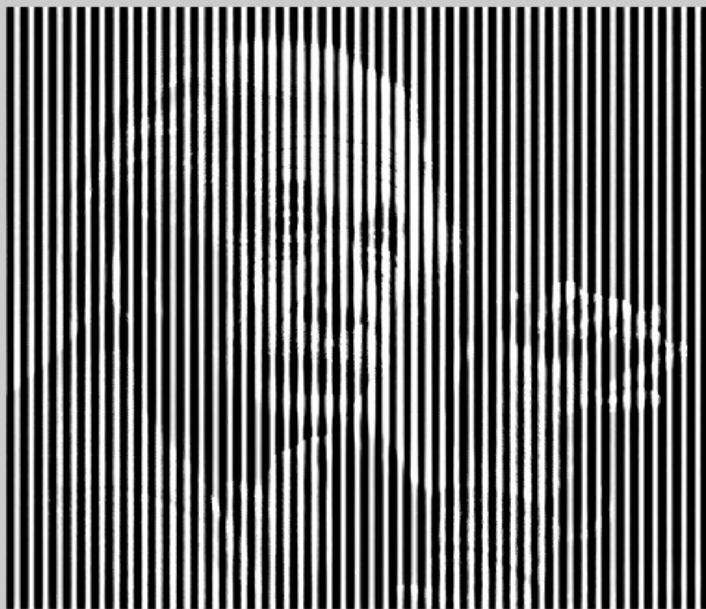
w = **ancho** del filtro (para filtros ideales), D_0 = **diámetro** medio

Filtro rechaza banda ideal 800x800, D0=200, W=50

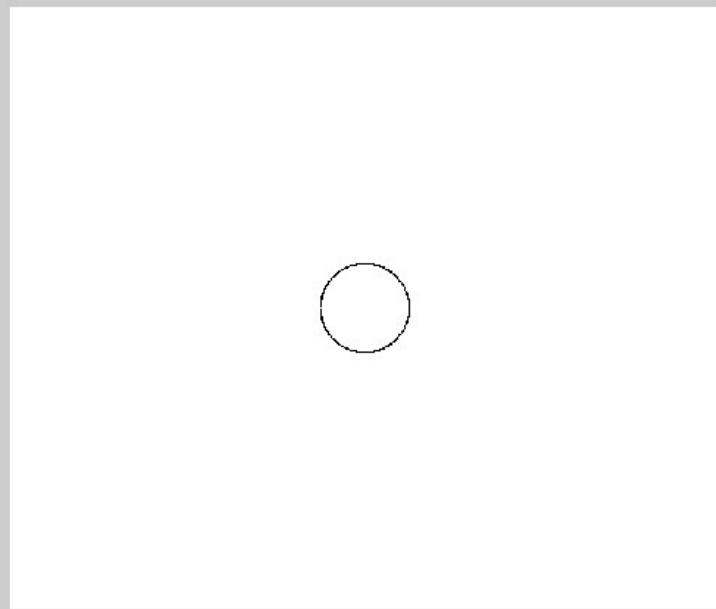


```
function H = idealReject(D,D0,w)
H= not((D>=D0-w/2) & (D<=D0+w/2));
%filtro rechaza banda ideal
```

Imagen original con ruido periódico



Filtro rechaza banda ideal, D0=50, W=2



Espectro de magnitud imagen con ruido



Espectro de magnitud de imagen filtrada

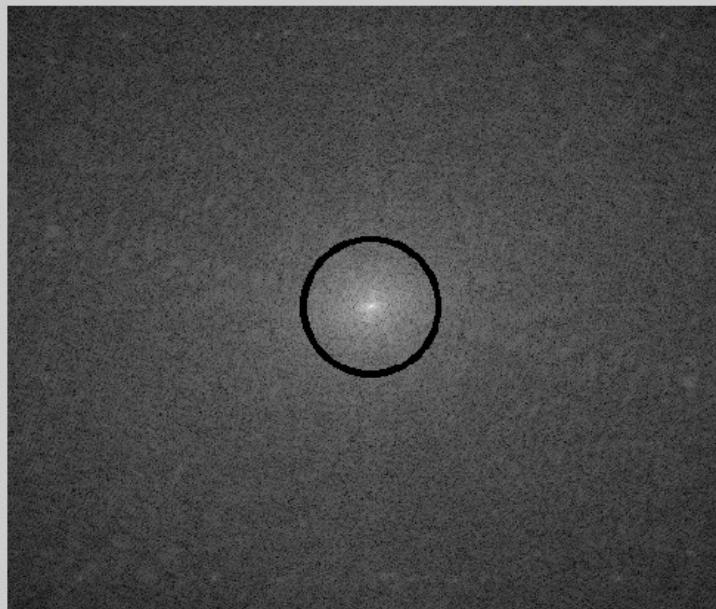


Imagen con ruido

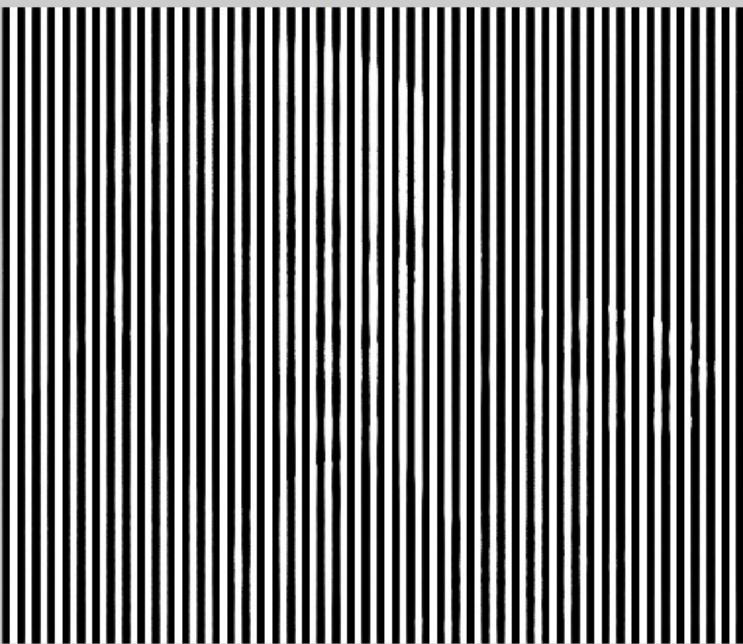
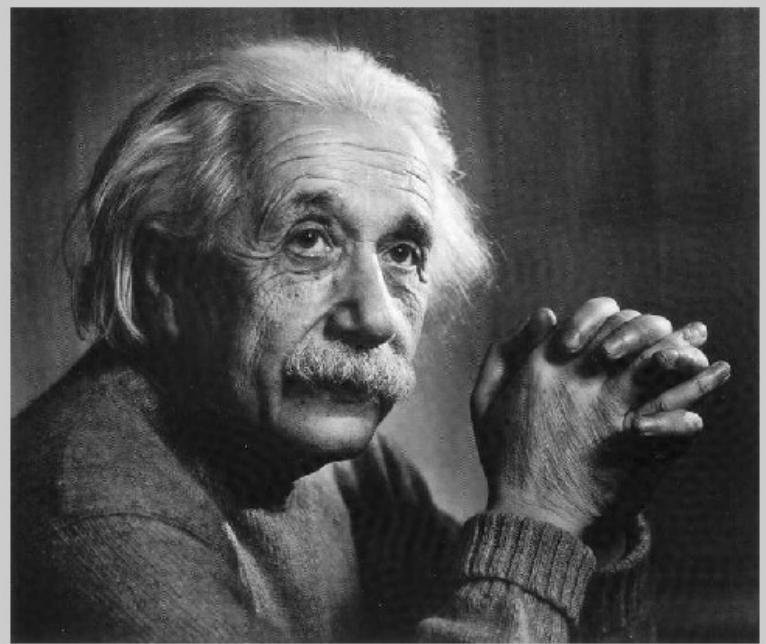


Imagen filtrada



Se eliminó el ruido periódico aunque aparecieron anillos (**ringing**) ocasionados por el fenómeno de Gibbs.

$$R(u, v) = j \frac{AMN}{2} [e^{-j2\pi(u_0B_x/M + v_0B_y/N)} \delta(u + u_0, v + v_0) - e^{j2\pi(u_0B_x/M + v_0B_y/N)} \delta(u - u_0, v - v_0)]$$

$u = 0, 1, 2, \dots M - 1$ y $v = 0, 1, 2, \dots N - 1$,
par de impulsos unitarios ubicados en $(u + u_0, v + v_0)$ y $(u - u_0, v - v_0)$
(ver **script imnoise3.m**)

Rechaza banda Butterworth de orden n

$$H(u, v) = \frac{1}{1 + \left[\frac{D(u, v) \cdot w}{D^2(u, v) - D_0^2} \right]^{2n}}$$

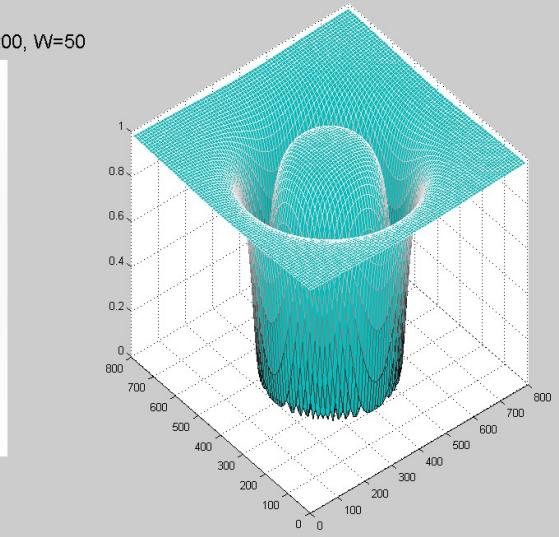
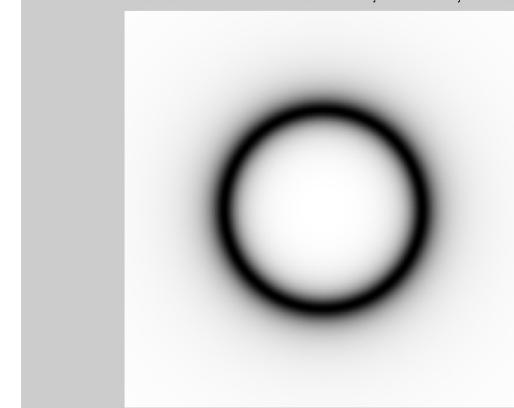
```
function H = btwReject(D, DO, w, n)
H=1./ (1+((D*w)./(D.^2-DO.^2)).^2*n));
%filtro rechaza banda BTW
```

Rechaza banda Gaussiano

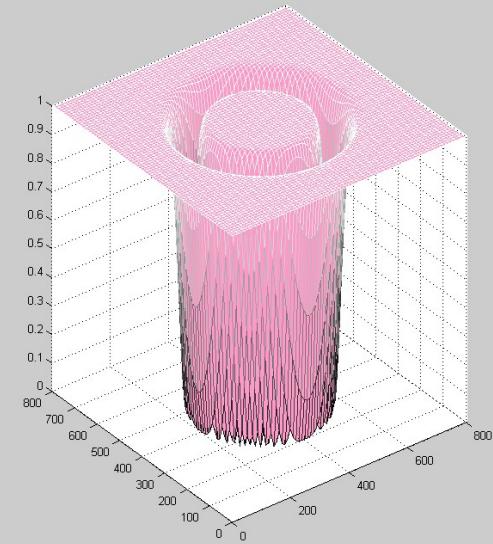
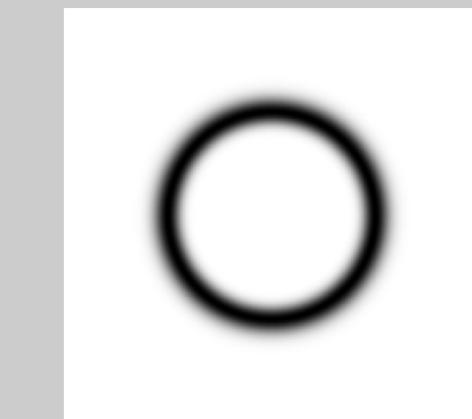
$$H(u, v) = 1 - e^{-\left[\frac{D^2(u, v) - D_0^2}{D(u, v) \cdot w}\right]^2}$$

```
function H = gaussReject(D, DO, w)
H=1-exp(-((D.^2-DO.^2)./(D.*w+eps)).^2);
%filtro rechaza banda Gaussiano
```

Filtro rechaza banda Butterworth n=2, 800x800, D0=200, W=50



Filtro rechaza banda Gaussiano, 800x800, D0=200, W=50



□ **Filtro pasa banda:** su acción es opuesta al del *rechaza banda* H_{RB} .

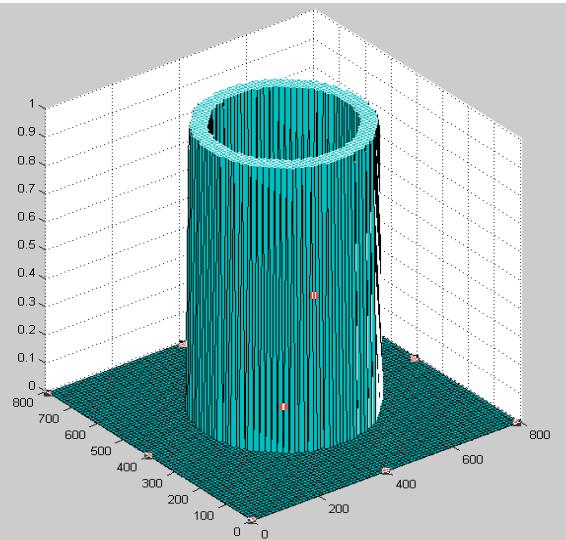
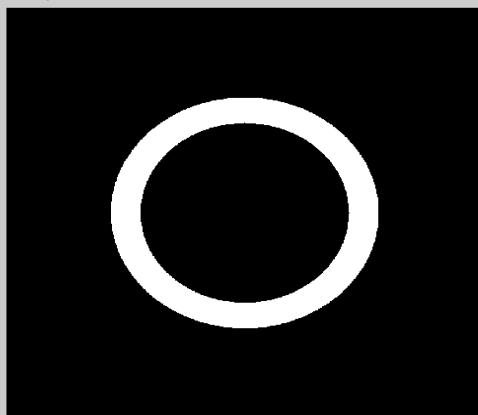
Es útil para aislar componentes del ruido y estudiarlo. Se obtiene a partir de la función de transferencia del filtro rechaza banda:

$$H_{PB}(u, v) = 1 - H_{RB}(u, v)$$

Pasa banda ideal:

$$H(u, v) = \begin{cases} 1 & \text{si } Do - \frac{w}{2} \leq D(u, v) \leq Do + \frac{w}{2} \\ 0 & \text{otro caso} \end{cases}$$

Filtro pasa banda ideal, 800x800, D0=200, W=50

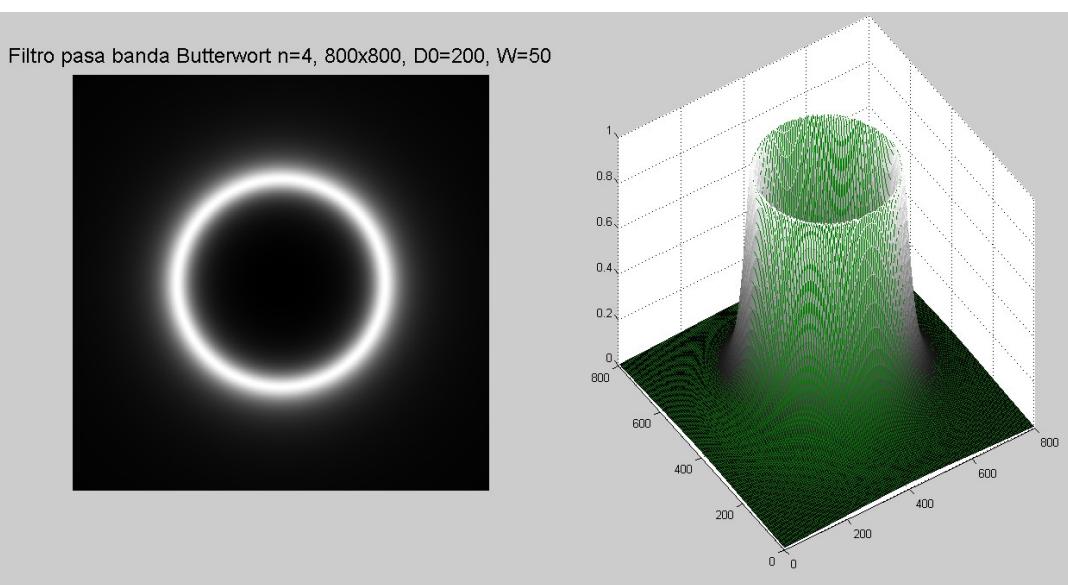


```
function H = idealPass(D,D0,w)
H= (D >= D0-w/2) & (D<=D0+w/2);
%filtro pasa banda ideal
```

Pasa banda Butterworth orden n:

$$H(u, v) = \frac{1}{1 + \left[\frac{D(u, v)^2 - D_0^2}{D(u, v).w} \right]^{2n}}$$

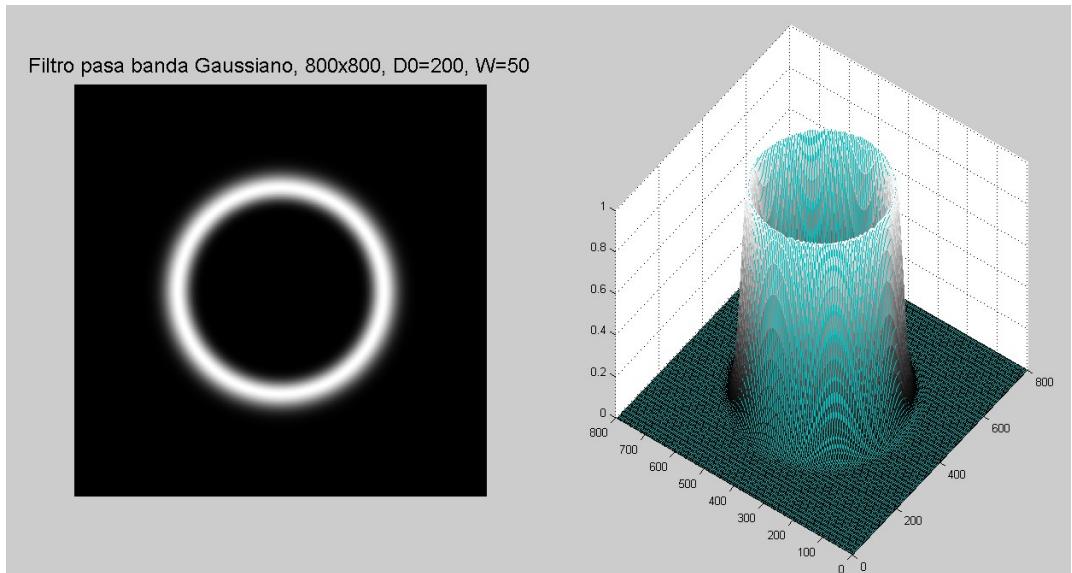
```
function H = btwPass(D, DO, w, n)
H = 1 ./ (1 + ((D.^2 - DO.^2) ./ (D.*w)).^2*n));
%filtro pasa banda BTW
```



Pasa banda Gaussiano:

$$H(u, v) = e^{-\left[\frac{D^2(u,v)-D_0^2}{D(u,v).w}\right]^2}$$

```
function H = gaussPass(D, DO, w)
H = exp(-(D.^2 - DO.^2) ./ (D.*w+eps)).^2);
%filtro pasa banda Gaussiano
```



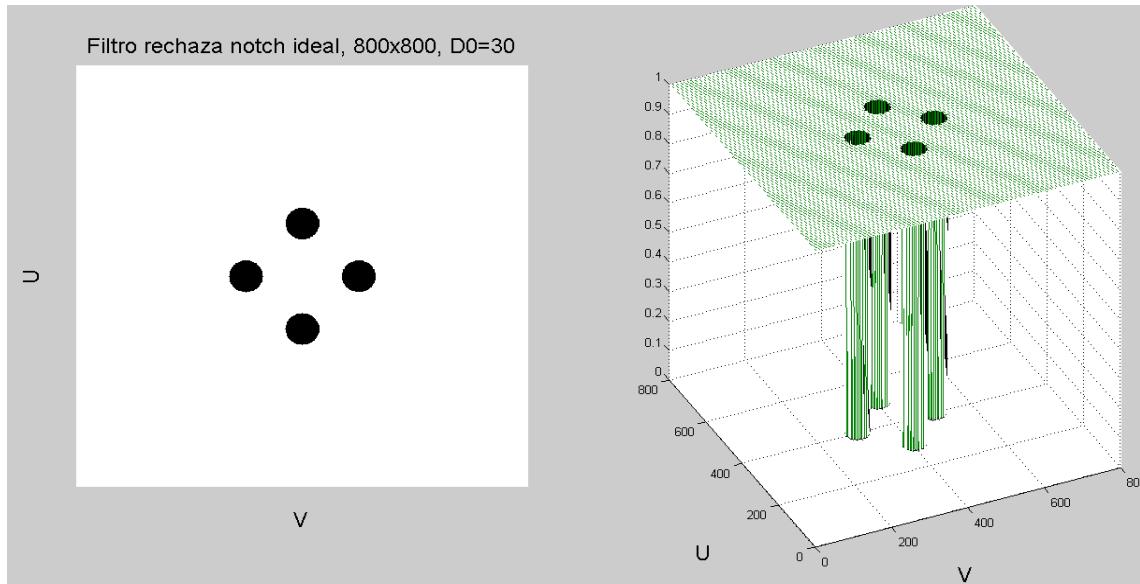
❖ **Filtro Notch:** rechaza frecuencias alrededor de una frecuencia central. Los **picos aparecen de a pares**.

- Hay que definir el número de pares (**Q**) y el tamaño del área notch (**D0**).
- Se forman haciendo el producto de filtros pasa altos cuyos centros han sido trasladados a los centros notch. La formula general para **Q pares notch** es:

$$H_{NR}(u, v) = \prod_{k=1}^Q H_k(u, v)H_{-k}(u, v)$$

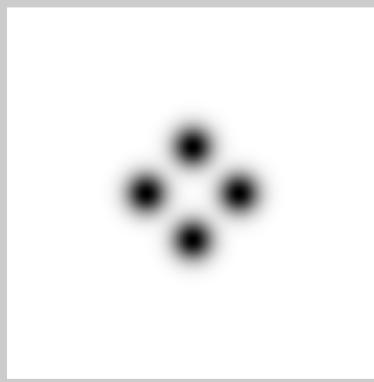
Donde $H_k(u, v)$ y $H_{-k}(u, v)$ son filtros pasa altos con centro en (u_k, v_k) y $(-u_k, -v_k)$, respectivamente.

Notch rechaza banda Ideal:

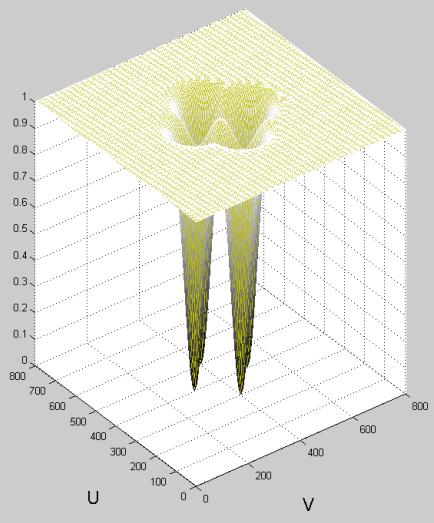


Notch rechaza banda Gausiano:

Filtro rechaza notch Gaussiano, 800x800, D0=30



V

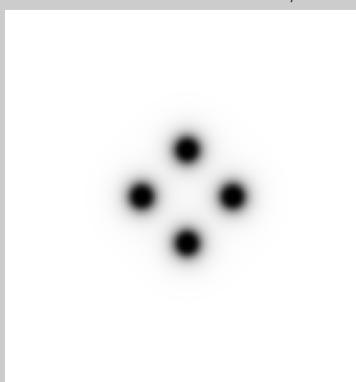


Obs: las dimensiones MxN deben ser impares

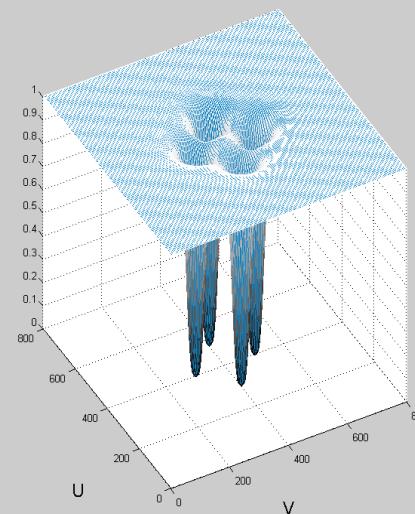
▷

Notch rechaza banda Butterworth orden n:

Filtro rechaza notch Butterworth orden n=2, 800x800, D0=30



V



```
%genera filtro Notch
function aux =
filtrado_notch_simplificado(forma, tipo, M, N, C, D0, n)
%forma= ideal, BTW, Gauss
%tipo: rechaza o pasa banda notch

f=size(C,1);
[V, U]=meshgrid(1:N, 1:M);
aux= ones(M, N);
u0 = floor(M/2)+1;
v0 = floor(N/2)+1;

for i=1:f
    u1 = u0 - C(i,1);
    v1 = v0 + C(i,2);
    D = (sqrt((U-u1).^2+(V-v1).^2));

    switch forma
        case 'ideal'
            H=D>=D0;
        case 'btw'
            H= 1-1./ (1+(D/D0).^(2*n));
        case 'gaussian'
            H=1-exp(-(D.^2)/(2*D0.^2));
    end

    Hr=rot90(H, 2);
    aux=aux.*H .* Hr;
end

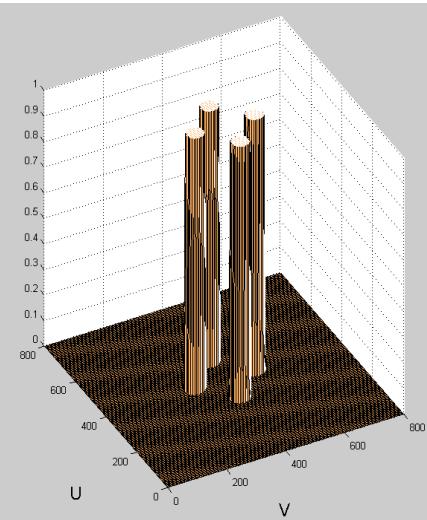
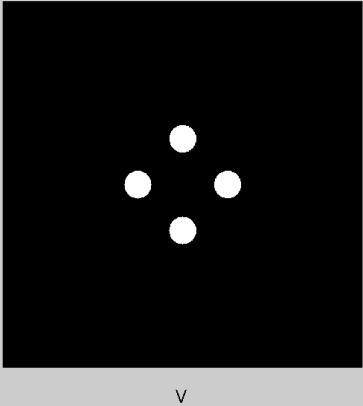
if strcmp (tipo, 'pass')
    aux=1- aux;
end
```

☐ NOTCH PASA BANDA:

$$H_{PN}(u, v) = 1 - H_{RN}(u, v)$$

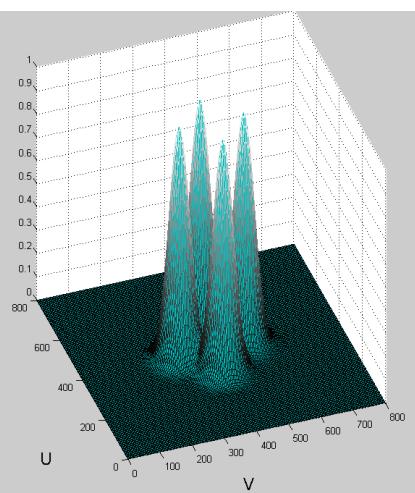
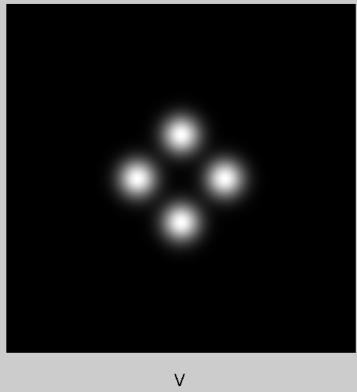
Notch pasa banda ideal:

Filtro notch pasa banda ideal, 800x800, D0=30



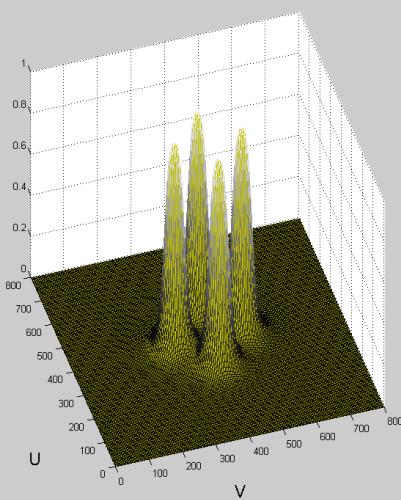
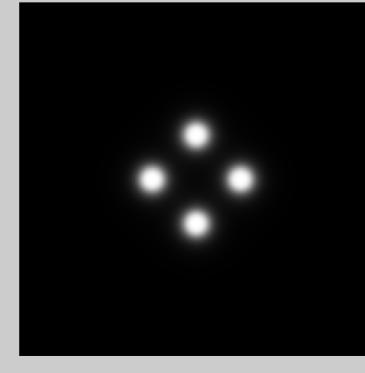
Notch pasa banda Gaussiano:

Filtro notch pasa banda Gaussiano, 800x800, D0=30

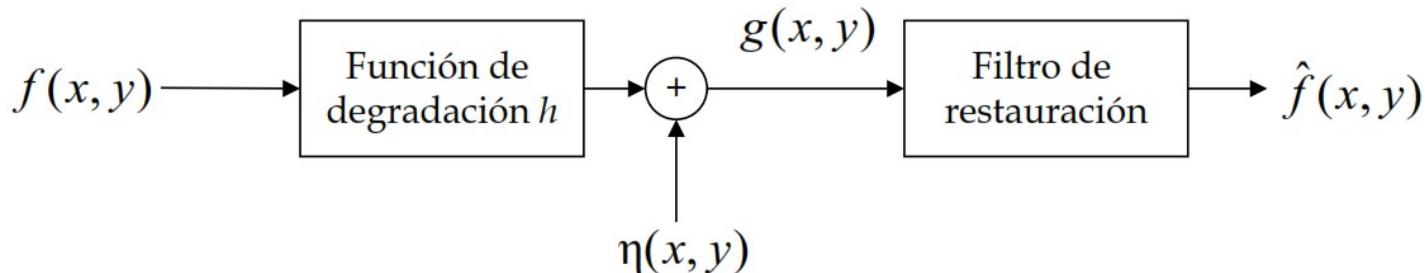


Notch pasa banda Butterworth orden n:

Filtro notch pasa banda Butterworth orden n=2, 800x800, D0=30



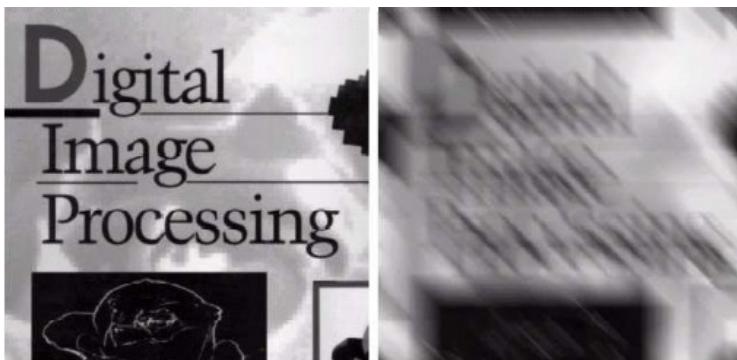
Modelo de degradación y restauración



- Si el ruido es nulo:
$$g(x, y) = h(x, y) \otimes f(x, y)$$

$$G(u, v) = H(u, v) \cdot F(u, v)$$
- Estudiaremos degradaciones que pueden ser modeladas mediante procesos lineales e invariantes a la posición.
- Los filtros utilizados reciben el nombre de *filtros de deconvolución* y el proceso de restauración se denomina *filtrado inverso*.
- ❖ Estimación de H:

En caso de conocer el origen de la degradación, se puede plantear y ajustar un modelo matemático.
Ej: desenfoque debido a movimiento uniforme en la imagen.



$$H(u, v) = \frac{T}{\pi(ua + vb)} \sin[\pi(ua + vb)] e^{-i\pi(ua + vb)}$$

Filtrado inverso

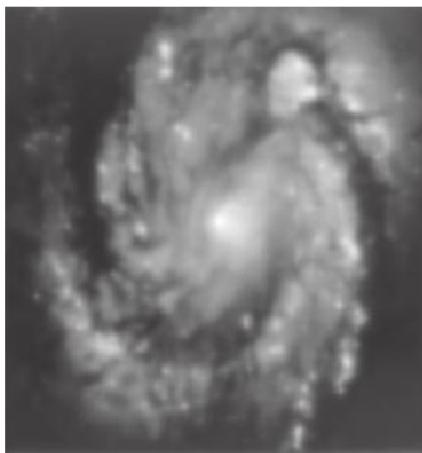
- ❑ Ausencia de ruido y conocida (o estimada) H : la manera más simple de restaurar la imagen es

$$\hat{F}(u, v) = \frac{G(u, v)}{H(u, v)} = R(u, v)G(u, v)$$

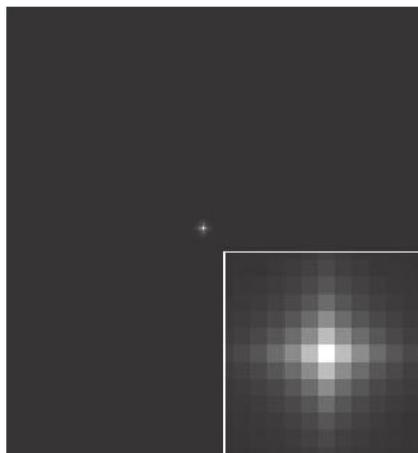
donde R es el **filtro inverso**

▪ **Problema Importante:**

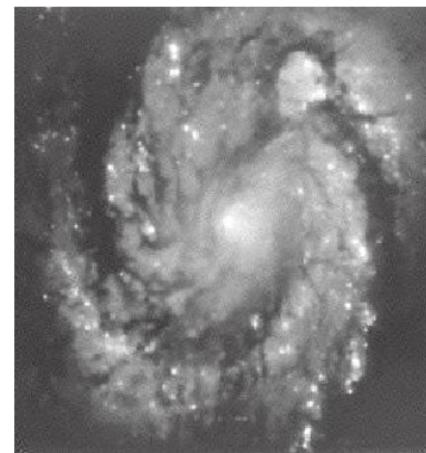
Dado que en la deconvolución se realiza un cociente, si $H(u, v)$ posee ceros (o valores pequeños) el proceso puede provocar overflow durante su cómputo



(a)



(b)



(c)

- a) Imagen original tomada por el telescopio espacial Hubble, (b) función de dispersión de puntos (PSF) medida, y (c) resultado de la **deconvolución** (division por OTF)

- ❑ En presencia de ruido tenemos que:

$$G(u, v) = H(u, v) \cdot F(u, v) + N(u, v)$$

$$\hat{F}(u, v) = F(u, v) + \frac{N(u, v)}{H(u, v)}$$

(divido todo por $H(u, v)$)

- Aún conociendo exactamente H no podremos recobrar la imagen original, ya que el ruido es una función aleatoria cuya transformada de Fourier $N(u,v)$ desconocemos.
- Para valores pequeños de H (algunos pueden ser 0), el ruido se amplifica desmesuradamente debido a que la relación N/H dominará en la estimación.

Solución: restringir el cociente sólo a aquellos pixeles que no causen overflow (limitar el rango de frecuencias para la obtención de la inversa, a las frecuencias próximas al origen). Los valores pequeños o nulos de H son menos probables cerca del origen debido a la magnitud de la OTF.

Filtro Pseudo-inverso

$$\text{Filtro } (u,v) = \begin{cases} \frac{1}{H(u,v)} & \text{si } |H(u,v)| > \delta \\ 0 & \text{si } |H(u,v)| \leq \delta \end{cases}$$

$\delta = \text{umbral pequeño}$

Filtro Inverso Limitado Radialmente

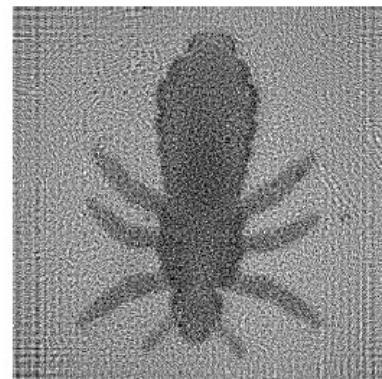
$$\text{Filtro } (u,v) = \begin{cases} \frac{1}{H(u,v)} & \text{si } \sqrt{u^2 + v^2} \leq R \\ 0 & \text{si } \sqrt{u^2 + v^2} > R \end{cases}$$

$R = \text{radio límite}$

La energía de las imágenes normalmente se concentra a bajas frecuencias.
La energía del ruido se distribuye sobre todas las frecuencias (ruido blanco)



(a)



(b)



(c)

(a) Imagen degradada (blurred + ruido aleatorio), (b) aplicación de filtro inverso ideal (se amplifica el ruido de modo que domina el resultado, se pierden los detalles), (c) aplicación de la deconvolución con apodización (elimino aquellos términos frecuenciales que producen overflow). Mejor resultado pero con ringing y se observan algunos restos de desenfoque.

- ❖ **Filtro de Wiener:** incorpora tanto la función de degradación como las características del ruido en el proceso de reconstrucción.

Considera a la imagen y al ruido procesos estocásticos. Busca estimar \hat{f} tal que minimice el **error cuadrático medio** (EMS):

$$EMS = E\{(f - \hat{f})^2\}$$

E es el *valor esperado* y **f** la *imagen no degradada*. La solución en el dominio frecuencial es:

$$\hat{F}(u, v) = \left[\frac{1}{H(u, v)} * \frac{|H(u, v)|^2}{|H(u, v)|^2 + S_\eta(u, v)/S_f(u, v)} \right] G(u, v)$$

$H(u, v)$: función degradación

$$|H(u, v)|^2 = H^*(u, v)H(u, v)$$

$H^*(u, v)$: complejo conjugado de $H(u, v)$

$S_\eta(u, v) = |N(u, v)|^2$: espectro de potencia del ruido

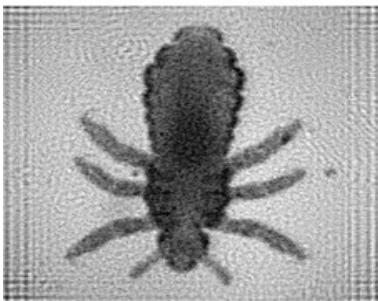
$S_f(u, v) = |F(u, v)|^2$: espectro de potencia de la **imagen original**

$S_\eta(u, v)/S_f(u, v)$: relación de **potencia ruido vs señal**

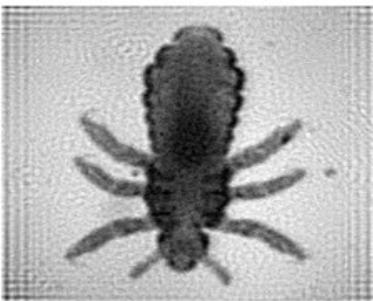
- La implementación requiere conocer los espectros de potencia de la imagen original y del ruido. Cuando no se conocen (o no se pueden estimar), se emplea la siguiente aproximación:

$$\hat{F}(u, v) = \left[\frac{1}{H(u, v)} \frac{|H(u, v)|^2}{|H(u, v)|^2 + K} \right] G(u, v)$$

K es un parámetro ajustable que controla el equilibrio entre nitidez (**sharpening**) y ruido



(a)



(b)



(c)



(d)

Deconvolución Wiener incrementando K (a-d). Se reduce el ruido a expensas de perder nitidez (sharpness).

- ❖ K pequeños: mejor remoción de la degradación y pobre filtrado del ruido.
- ❖ K grandes: pobre restauración de la degradación y mejor filtrado del ruido.

En Matlab se usa el comando **deconvwnr** para restaurar una imagen degradada utilizando el filtro Wiener.

La deconvolución Wiener puede emplearse efectivamente cuando las características frecuenciales de la imagen y del ruido son conocidas, aunque sea en un cierto grado.

En ausencia de ruido, el filtro Wiener se reduce a un filtro inverso ideal

$$J = \text{deconvwnr}(I, PSF, NSR)$$

Se **deconvoluciona** la imagen degradada **I** usando el filtro Wiener, y retorna la imagen restaurada **J** (deblurred).

PSF es la función de dispersión de puntos con la cual la imagen original **I** fue convolucionada.

NSR es la **relación de potencia ruido/señal**, puede ser un escalar o un array en el dominio frecuencial del mismo tamaño que **I**.

```
%Simulo el movimiento
LEN = 20; THETA = 45;
PSF = fspecial('motion', LEN, THETA); %Point-spread function
Imov= imfilter(I, PSF, 'conv', 'circular');

%Restauro asumiendo que no hay ruido
estimated_nsr = 0; %relación de potencial ruido/señal (ruido aditivo)

%como NSR=0 tenemos filtro inverso ideal
wnr = deconvwnr(Imov,PSF,estimated_nsr);
```



```
%Sumo ruido gaussiano a la imagen movida
noise_mean=0;
noise_var=0.0001;
Imov_n=imnoise(Imov, 'gaussian', noise_mean, noise_var);

%Restauro asumiendo que no hay ruido
estimated_nsr = 0;
wnr2 = deconvwnr(Imov_n,PSF,estimated_nsr);
```

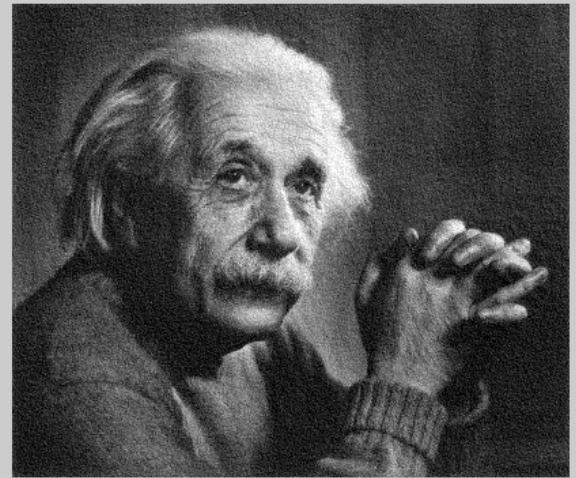


```
%Restauro usando un mejor estimador de la NSR  
estimated_nsr = noise_var / var(I(:));  
wnr3 = deconvwnr(Imov_n, PSF, estimated_nsr);
```

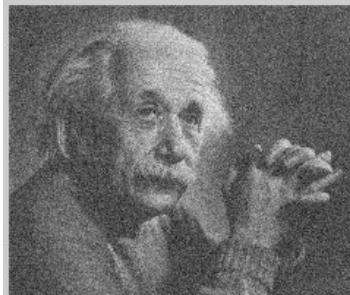
```
%itero buscando el mejor parámetro K  
K=0.001:0.001:0.01;
```

```
figure  
for i=1:10  
    wnr4 = deconvwnr(Imov_n, PSF, K(i));  
    subplot(2,5,i);imshow(wnr4,[]);  
    title(['K=',num2str(K(i))]);  
end
```

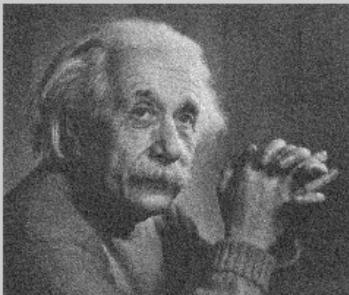
Restauración imagen estimando NSR



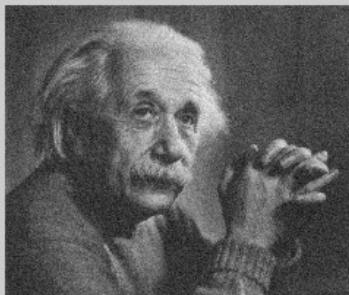
K=0.001



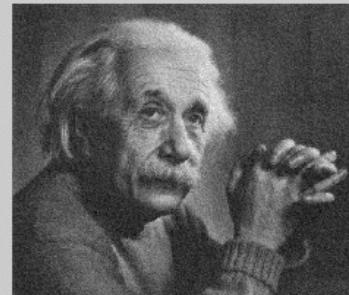
K=0.002



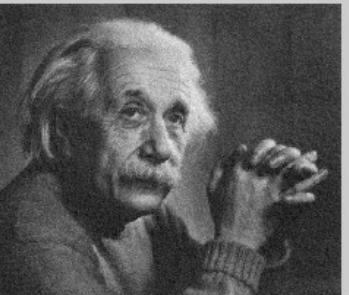
K=0.003



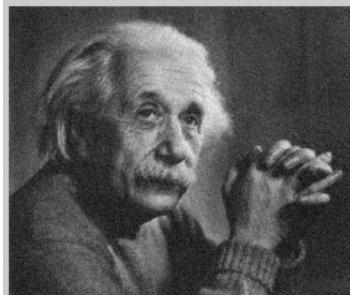
K=0.004



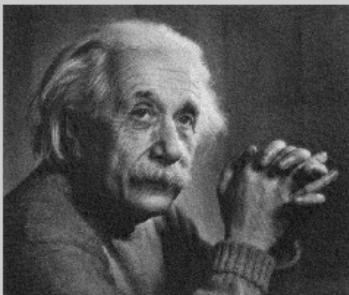
K=0.005



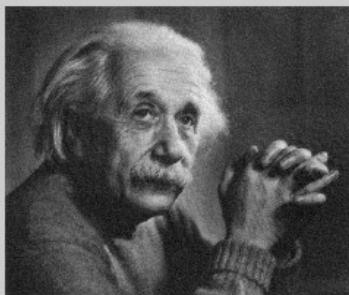
K=0.006



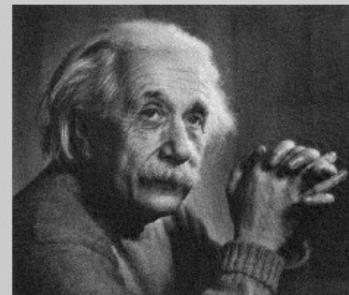
K=0.007



K=0.008



K=0.009



K=0.01

