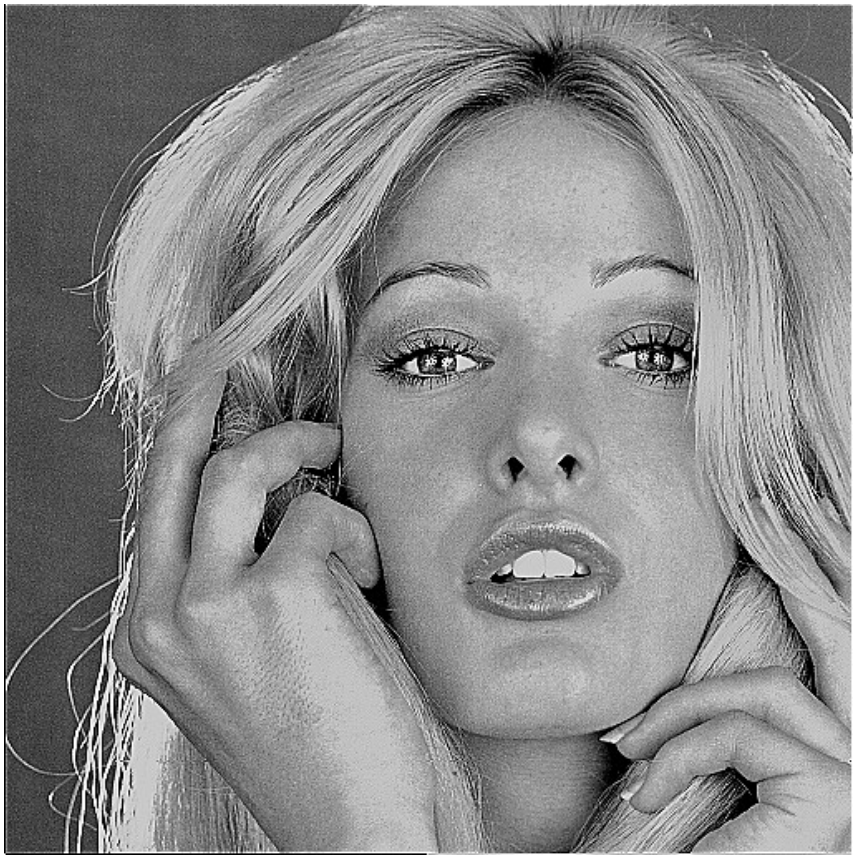
Filtros espaciales

Imagen Original

Resultado del Filtro

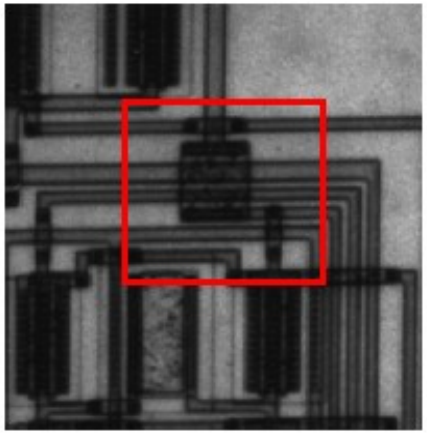


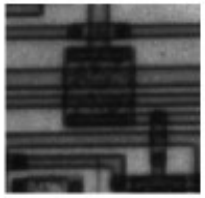
Aplicación del filtro unsharp masking

Vecindades

peraciones sobre vecindades:

n aquellas que se realizan sobre bloques definidos
ntro de la imagen, en vez de procesar toda la image





0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Operaciones basadas en vecindades

Las operaciones se realizan de la siguiente forma:

1.- Se selecciona el píxel a procesar.
2.- Se selecciona el entorno del píxel.
3.- Se aplica una función que depende del valor de los píxeles del entorno seleccionado
4.- Se altera el píxel de la imagen de salida equivalente al píxel de la imagen de entrada, por el valor devuelto por la función.
5.- Repetir de 1 a 4 por cada píxel de la imagen de entrada.

Operaciones basadas en vecindades (II)

Ejemplo: Máximo de una vecindad de 3x3

Imagen de entrada (I)

6	14	10	10	4	3
11	16	17	20	6	3
11	20	15	10	5	4
13	16	6	2	2	2
11	16	7	3	4	2
6	4	4	2	3	2

Imagen de salida (I1)

16	17	20	20	20	6
20	20	20	20	20	6
20	20	20	20	20	6
20	20	20	15	10	5
16	16	16	7	4	4
16	16	16	7	4	4

Lo cual se realizaría con el siguiente programa Matlab

```
>> f=inline('max(x(:))');  % Define función máxima
>> I1=nlfilter(I,[3, 3],f);   % Devuelve máximo
```

Operaciones basadas en vecindades (III)

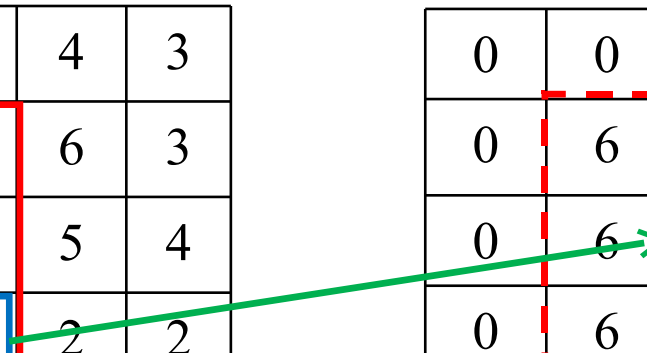
Ejemplo: Mínimo de una vecindad de 3x3

Imagen de entrada (I)

6	14	10	10	4	3
11	16	17	20	6	3
11	20	15	10	5	4
13	16	6	2	2	2
11	16	7	3	4	2
6	4	4	2	3	2

Imagen de salida (I1)

0	0	0	0	0	0
0	6	10	4	3	0
0	6	2	2	2	0
0	6	2	2	2	0
0	4	2	2	2	0
0	0	0	0	0	0



Lo cual se realizaría con el siguiente programa Matlab

```
>> f=inline('min(x(:))');
>> I1=nlfilter(I,[3, 3],f);
```

Operaciones basadas en vecindades (IV)

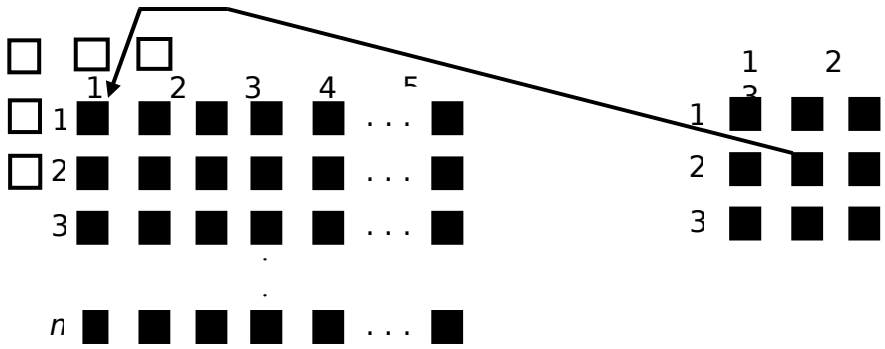
El problema del contorno

Imagen de entrada

6	14	10	10	4	3
11	16	17	20	6	3
11	20	15	10	5	4
13	16	6	2	2	2
11	16	7	3	4	2
6	4	4	2	3	2

Imagen de salida

0	0	0	0	0	0
0	6	10	4	3	0
0	6	2	2	2	0
0	6	2	2	2	0
0	4	2	2	2	0
0	0	0	0	0	0



Operaciones basadas en vecindades (V)

Solución: Relleno de píxeles

Imagen de entrada

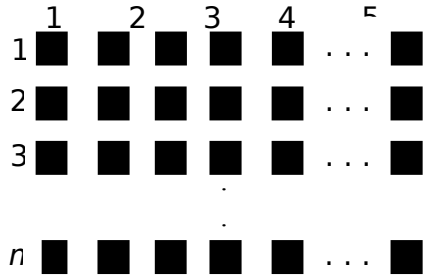
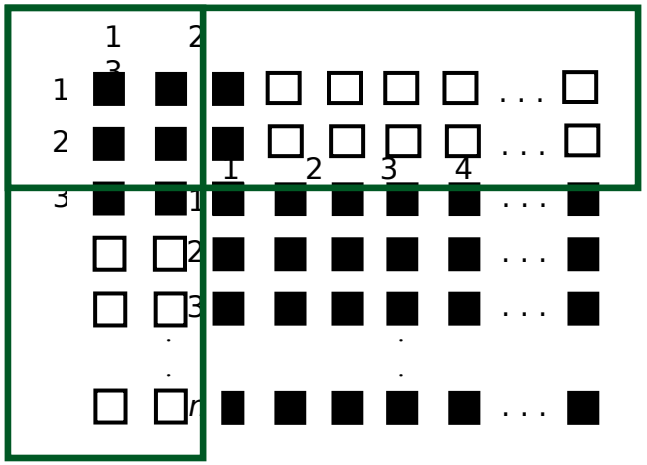


Imagen de salida



Rellenar contorno sin alterar la imagen original:

Función: *imfilter* ('La_imagen.jpg', máscara, **relleno**, salida)

'same': Mismo tamaño imagen original
'full': Tamaño resultante con relleno (mayor)

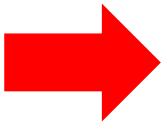
máscara=
[0 0 0 0 0
0 0 0 0 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 0]

Operaciones basadas en vecindades (VI)

Relleno de una constante

```
f =
 1   2   3   2   1
-1  -2  -3  -2  -1
 1   2   3   2   1
-1  -2  -3  -2  -1
 1   2   3   2   1
```

```
w =
[0 0 0 0 0
0 0 0 0 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 0]
```



25	25	25	25	25	25	25	25	25
25	25	25	25	25	25	25	25	25
25	25	1	2	3	2	1	25	25
25	25	-1	-2	-3	-2	-1	25	25
25	25	1	2	3	2	1	25	25
25	25	-1	-2	-3	-2	-1	25	25
25	25	1	2	3	2	1	25	25
25	25	25	25	25	25	25	25	25
25	25	25	25	25	25	25	25	25

```
>> imfilter(f,w,25,'full')
```

Operaciones basadas en vecindades (VII)

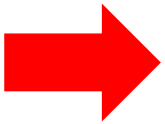
Relleno basado en réplica

f =

```
 1   2   3   2   1
-1  -2  -3  -2  -1
 1   2   3   2   1
-1  -2  -3  -2  -1
 1   2   3   2   1
```

w =

```
[0 0 0 0 0
0 0 0 0 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 0]
```



1	1	1	2	3	2	1	1	1
1	1	1	2	3	2	1	1	1
1	1	1	2	3	2	1	1	1
-1	-1	-1	-2	-3	-2	-1	-1	-1
1	1	1	2	3	2	1	1	1
-1	-1	-1	-2	-3	-2	-1	-1	-1
1	1	1	2	3	2	1	1	1
1	1	1	2	3	2	1	1	1
1	1	1	2	3	2	1	1	1

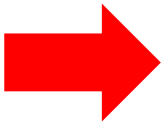
>> imfilter(f,w,'**replicate**' ,'full')

Operaciones basadas en vecindades (VIII)

Relleno basado en espejo

```
f =
 1   2   3   2   1
-1  -2  -3  -2  -1
 1   2   3   2   1
-1  -2  -3  -2  -1
 1   2   3   2   1
```

```
w =
[0 0 0 0 0
0 0 0 0 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 0]
```



-2	-1	-1	-2	-3	-2	-1	-1	-2
2	1	1	2	3	2	1	1	2
2	1	1	2	3	2	1	1	2
-2	-1	-1	-2	-3	-2	-1	-1	-2
2	1	1	2	3	2	1	1	2
-2	-1	-1	-2	-3	-2	-1	-1	-2
2	1	1	2	3	2	1	1	2
2	1	1	2	3	2	1	1	2
-2	-1	-1	-2	-3	-2	-1	-1	-2

>> imfilter(f,w,' **symmetric** ','full')

Operaciones basadas en vecindades (IX)

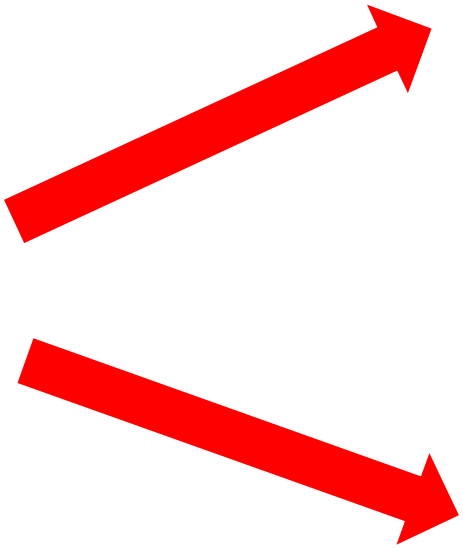
Tamaño imagen destino

f =

1	2	3	2	1
-1	-2	-3	-2	-1
1	2	3	2	1
-1	-2	-3	-2	-1
1	2	3	2	1

w =

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0



0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	1	2	3	2	1	0	0
0	0	-1	-2	-3	-2	-1	0	0
0	0	1	2	3	2	1	0	0
0	0	-1	-2	-3	-2	-1	0	0
0	0	1	2	3	2	1	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

>> imfilter(f,w,0,' **full**')

1	2	3	2	1
-1	-2	-3	-2	-1
1	2	3	2	1
-1	-2	-3	-2	-1
1	2	3	2	1

>> imfilter(f,w,0,' **same**')

Filtros espaciales

Responden a la siguiente ecuación:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

donde:

f(x+s, y+t): Valor de los píxeles del bloque seleccionado

w(s, t): Coeficientes que se aplicarán al bloque (máscara)

Siendo la matriz del bloque:

m(filas) = 2a + 1

n(columnas) = 2b + 1

Filtros espaciales (II)

Concepto:

Son las **operaciones** que se realizan **directamente sobre los píxeles**. Se define una **matriz de coeficientes** del filtro (o máscara, de tamaño *mxn*) los cuales constituirán pesos ponderados por los que se **multiplicarán** los valores originales de los píxeles.

Filtros espaciales (III)

Valores de los píxeles del bloque

f(x-1, y-1)	f(x-1, y)	f(x-1, y+1)
f(x, y-1)	f(x, y)	f(x, y+1)
f(x+1, y-1)	f(x+1, y)	f(x+1, y+1)

Valores de los píxeles de los coeficientes (máscara)

w(-1, -1)	w(-1, 0)	w(-1, 1)
w(0, -1)	w(0, 0)	w(0, 1)
w(1, -1)	w(1, 0)	w(1, 1)

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

Filtros espaciales (IV)

Ejemplo:

Valor de los píxeles

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

Máscara (o filtro)

8	1	6
3	5	7
4	9	2

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

g(2,4)=1(8)+8(1)+15(6)+7(3)+14(5)+16(7)+13(4)+20(9)+22(2)=585

Filtros espaciales (V)

Pasos para la aplicación del filtro lineal:

1.- Definir la máscara
2.- Definir tipo de relleno
3.- Aplicar la ecuación:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

4.- Definir tamaño de la imagen de salida

Filtros espaciales (VI)

Ejemplos de filtros

Filtro promedio:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \frac{1}{mxn} f(x+s, y+t)$$

Filtro promedio ponderado:

$$g(x,y) = \frac{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)}{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)}$$

Filtros espaciales (VII)

Ejemplo (usando Matlab):

```
% Imagen original
>> I=imread('tire.tif');

% Se define una matriz de pesos de valores iguales,
% a lo que se le denomina filtro promedio
>> w=ones(5,5)/25;

% Se aplica el filtro
>> I2=imfilter(I,w);
```

Original

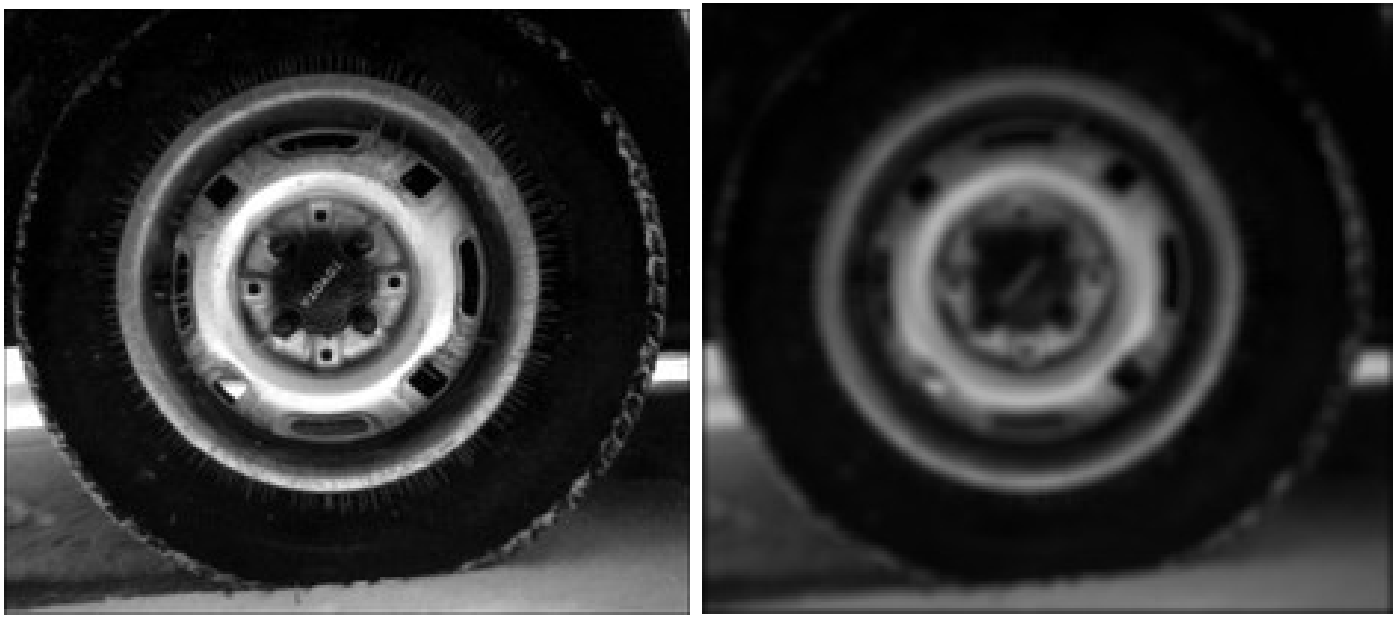
```
>> I(1:5,1:5)
ans =
   6   14   10   10    4
  11   16   17   20    6
  11   20   15   10    5
  13   16    6    2    2
  11   16    7    3    4
```

Resultado

```
>> I2(1:5,1:5)
ans =
   5    6    7    6    5
   6    8    9    7    5
   8    9   10    9    6
   7    8    9    8    5
   6    6    7    6    4
```

Filtros espaciales (VIII)

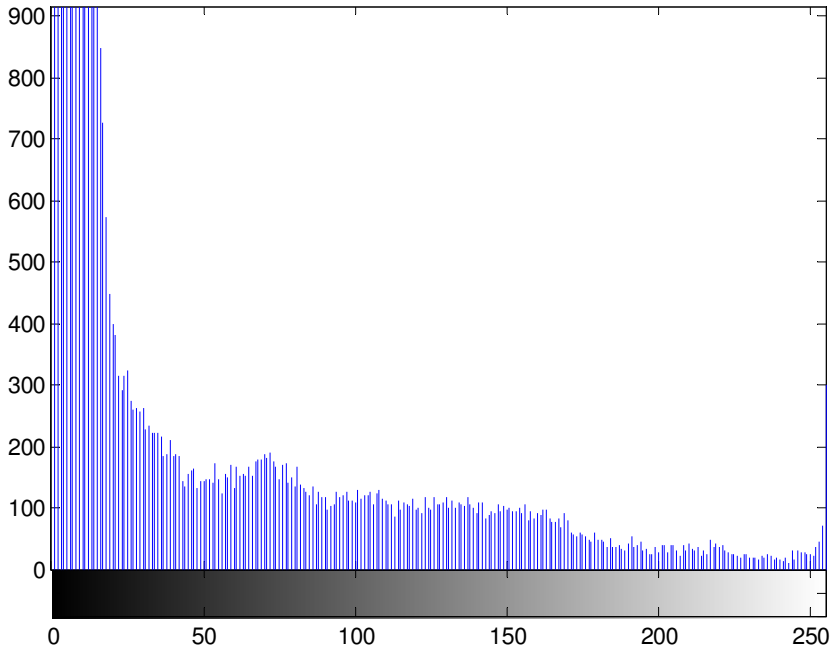
Original



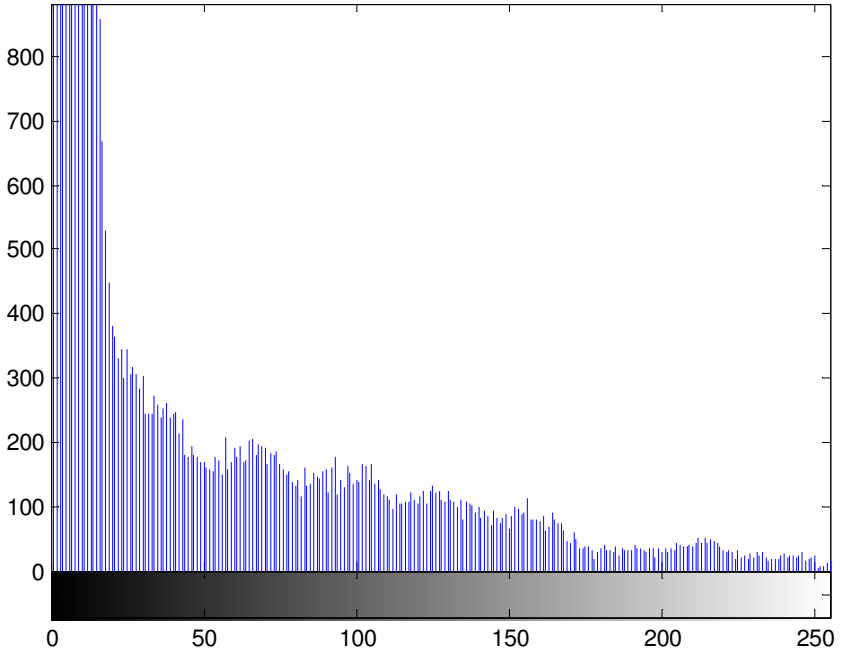
Filtros espaciales (IX)

Original

>> imhist(I)



>> imhist(I2)



Filtros espaciales (X)

Formas de aplicar la máscara

Método de correlación: Se aplica la máscara tal y como se define

Método de convolución: Se rota la máscara 180 grados alrededor del píxel central, antes de aplicar el filtro

Imagen

f =

0 0 0 0 0
0 0 0 0 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 0

Máscara

w =

1 2 3
4 5 6
7 8 9

0	0	0	0	0
0	9	8	7	0
0	6	5	4	0
0	3	2	1	0
0	0	0	0	0

>> imfilter(f,w,'corr')

0	0	0	0	0
0	1	2	3	0
0	4	5	6	0
0	7	8	9	0
0	0	0	0	0

>> imfilter(f,w,'conv')

Imágenes ruidosas

Ruido: Es un deterioro de la imagen que puede producirse debido a:

1.- Píxeles perdidos en un sensor CCD

2.- Cuando se comprime o transmite la imagen

3.- Inadecuada iluminación de la escena

4.- Escáner de documentos

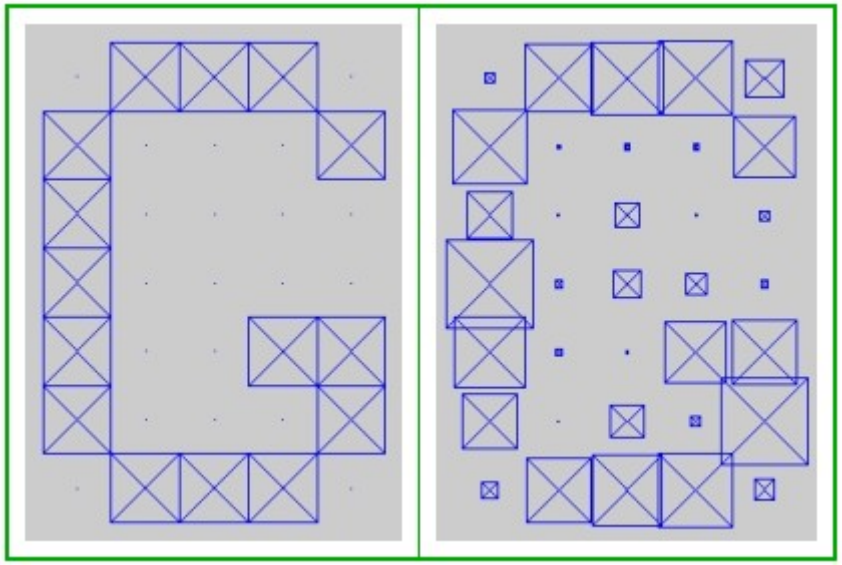
5.- Sensibilidad inadecuada de las cámaras, etc.

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

Ruído

Imágenes ruidosas (II)

Escáner de una letra con y sin ruído



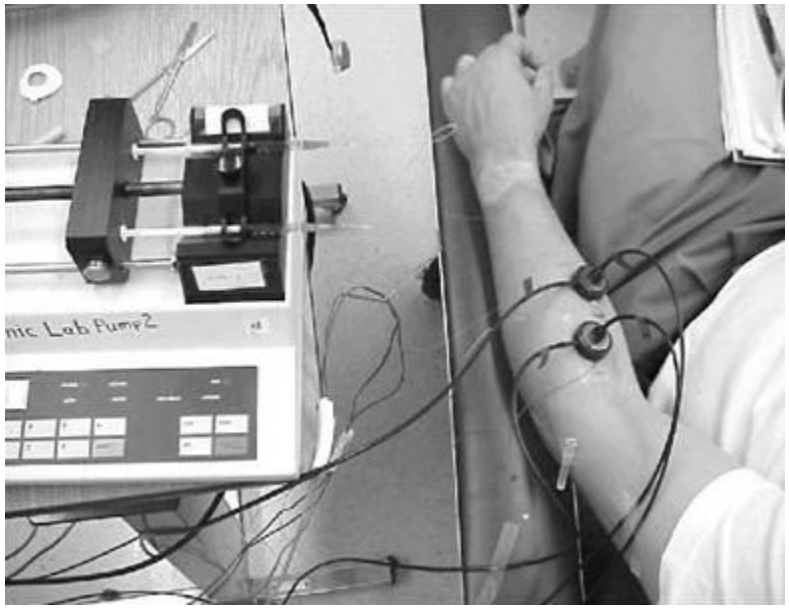
>>[R,Q] = size(G);

>>G_ruidosa = G + randn(R,Q)*0.2

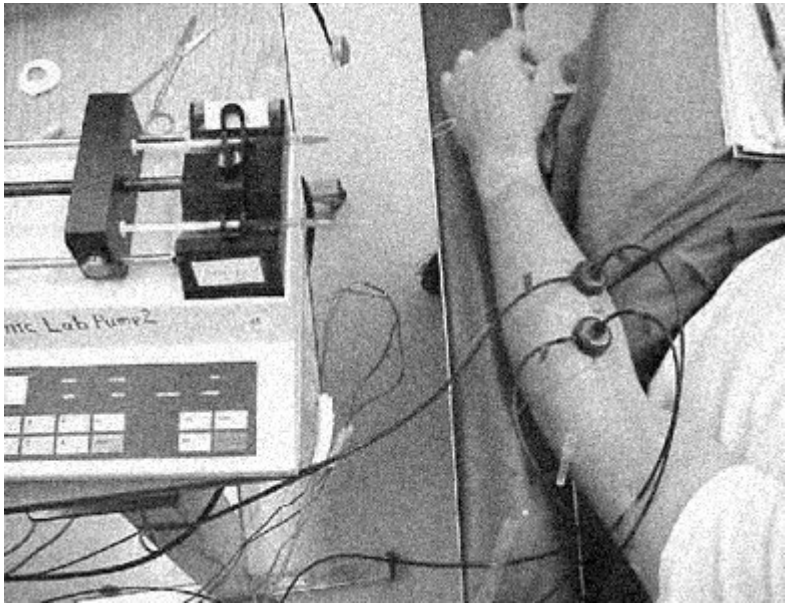
G = 0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 1 1 1 0

G_ruidosa = 0.1337 0.9859 1.0447 1.0773 -0.5392 1.0712 0.0547 0.0860
-0.0817 0.9028 0.6783 0.0299 0.3474 -0.0147 0.1335 1.2666 0.0991
0.3885 -0.2928 0.1002 1.0181 -0.0948 0.0390 0.8881 0.9455 0.8147
0.0208 0.4779 0.1484 1.2493 -0.2367 0.9407 1.0204 1.0842 -0.2981

Imágenes ruidosas (III)

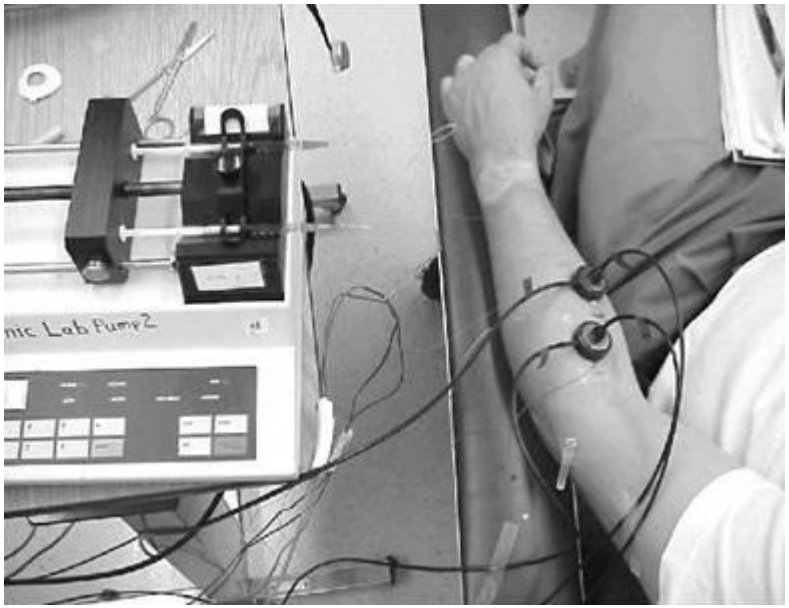


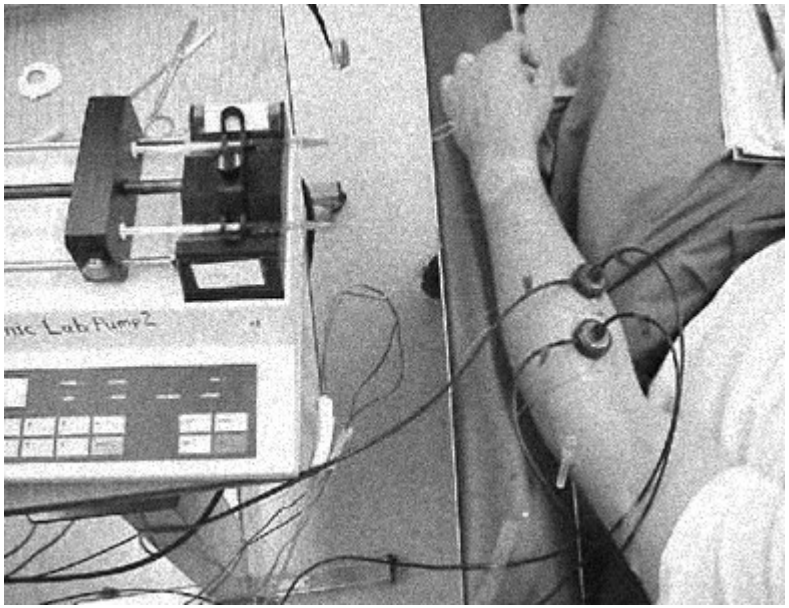
```
% Ruido gausiano
>> s = 15;          % Desviación estándar
>> s = s/255;       % Se normaliza a [0,1]
>> II = imnoise(road,'gaussian',0,s^2);
                    % media 0, varianza (15^2)
```



Imágenes ruidosas (IV)

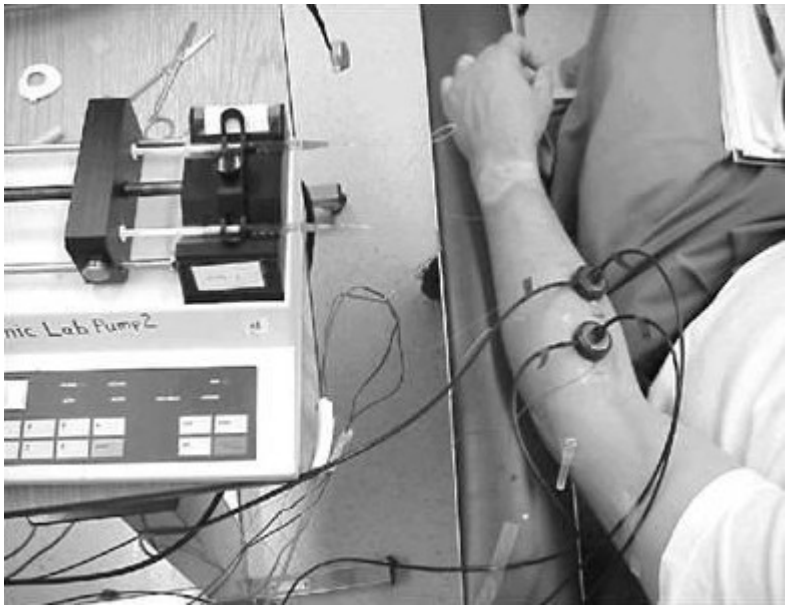
III=imnoise(I,'poisson');

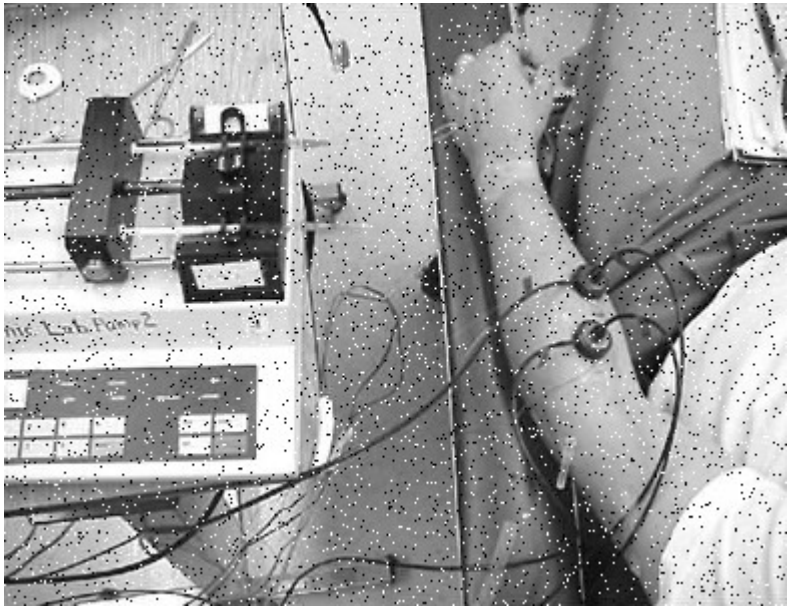




Imágenes ruidosas (V)

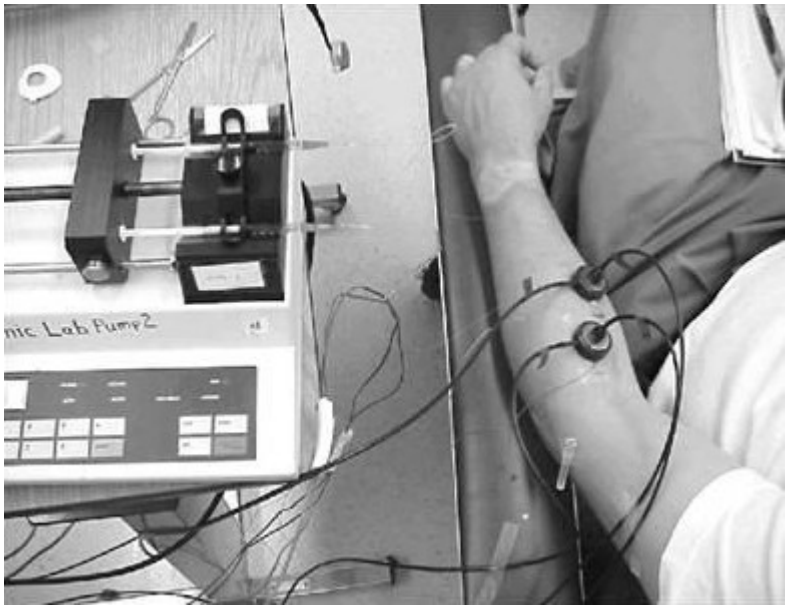
IV=imnoise(I,'salt & pepper');

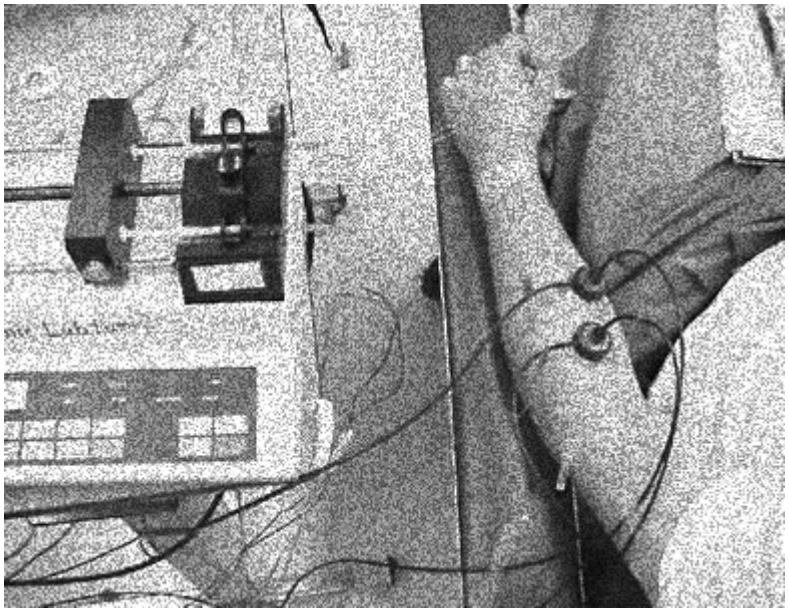




Imágenes ruidosas (VI)

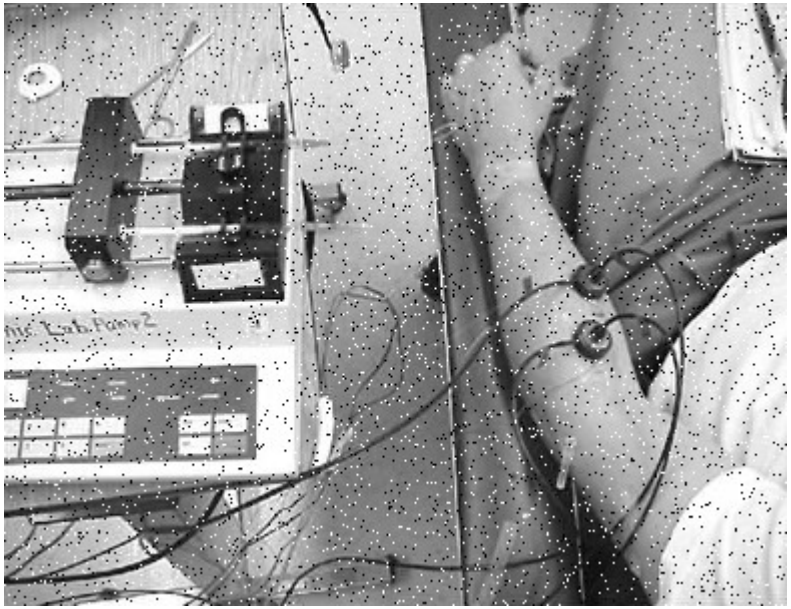
V=imnoise(I,'speckle');



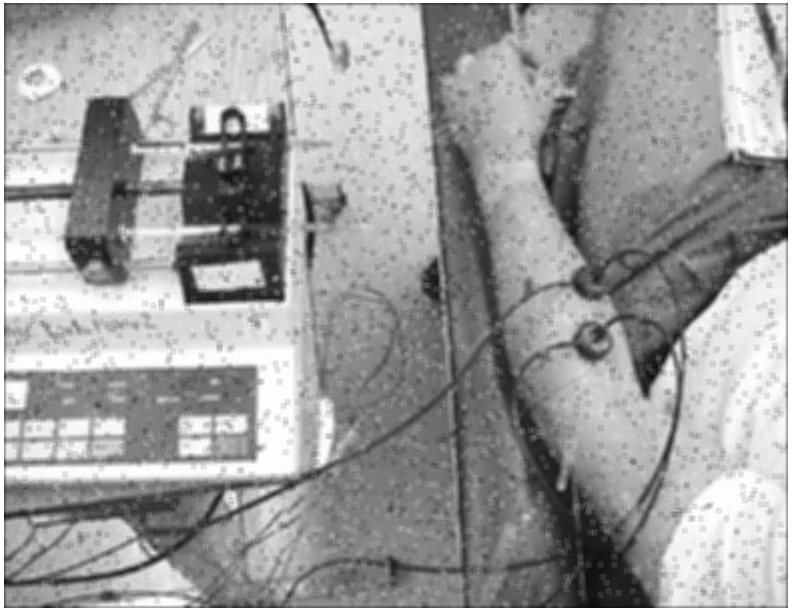


Supresión de ruido con *filtro promedio*

IV=imnoise(I,'salt & pepper');



>> w=ones(3,3)/9;
>> I2=imfilter(I,w);



Filtro Laplaciano

Este tipo de filtro se basa en un **operador derivativo**, por lo que **acentúa las zonas que tienen gran discontinuidad** en la imagen

$$\frac{\partial f(x)}{\partial x} = f(x+1) - f(x)$$

Filtro Laplaciano (II)

Si se cumple:

$$\frac{\partial f(x)}{\partial x} = f(x+1) - f(x)$$

Entonces:

$$\frac{\partial^{٢} f(x)}{\partial x^{٢}} = f(x+١) + f(x-١) - ٢ f(x)$$

Si la función depende de dos variables

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

Filtro Laplaciano (III)

Función dependiente de dos variables

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

La derivada de segundo orden con respecto a al variable x:

$$\frac{\partial^2 f(x,y)}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

De forma similar, con respecto a y:

$$\frac{\partial^2 f(x,y)}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

Finalmente:

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) - 2f(x,y) + f(x,y+1) + f(x,y-1) - 2f(x,y)$$

Filtro Laplaciano (IV)

El Laplaciano queda definido por:

$$\nabla^{2} f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

La anterior expresión es equivalente a aplicar una máscara definida por:

w =

0	1	0
1	-4	1
0	1	0

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

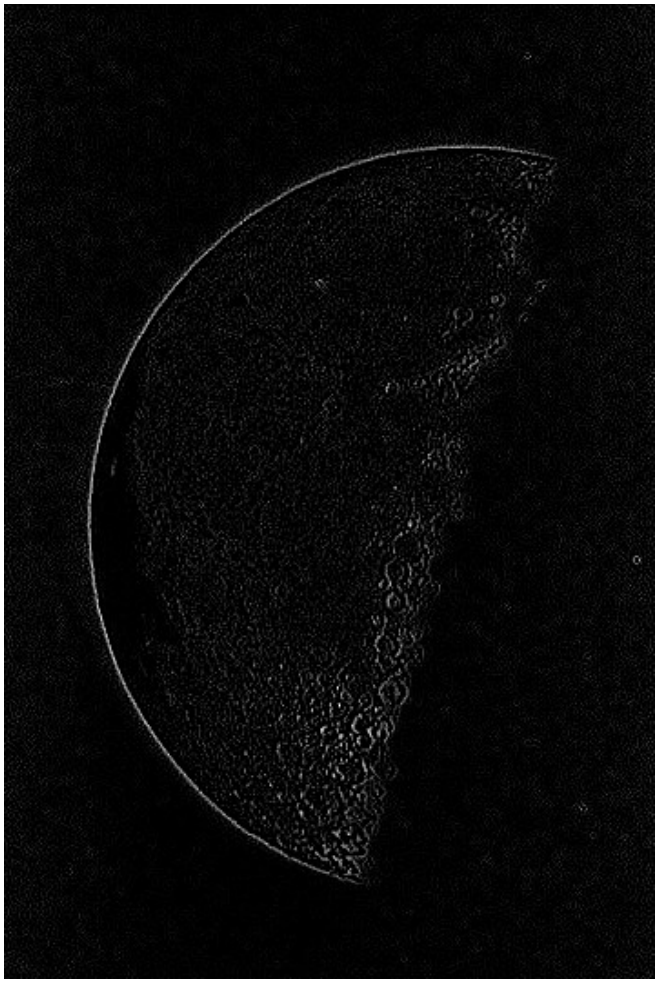
Filtro Laplaciano (V)

```
% Imagen original
>> I=imread('moon.tif' );
% Se define una matriz de pesos
>> w=[0, 1, 0; 1, -4, 1; 0, 1, 0];
% Se aplica el filtro
>> I2=imfilter(I,w);
```

Alternativa:

```
>> I=imread('moon.tif');
>> w=fspecial('laplacian',0);
>> I2=imfilter(I,w,'replicate');
```

Filtro Laplaciano (VI)



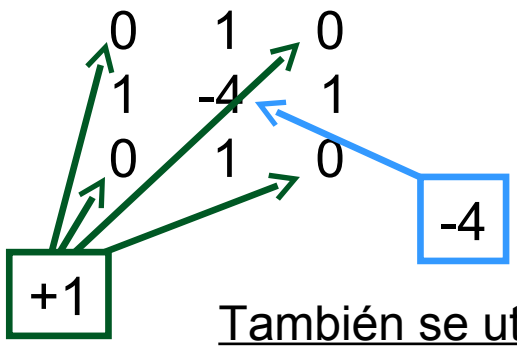
Imágenes: www.imageprecessingplace.com

Filtro Laplaciano (VII)

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

Si se desea considerar las variaciones con respecto a la diagonal

w =



w =

1	1	1
1	-8	1
1	1	1

También se utiliza el negado de las anteriores máscaras

w =

-1	-1	-1
-1	8	-1
-1	-1	-1

w =

0	-1	0
-1	4	0
0	-1	0

Filtros Laplacianos alternativos

Se emplea en pocas ocasiones en la práctica

- Excesivamente sensible a la presencia de ruido
- Da lugar a bordes dobles y no permite determinar direcciones

Se utiliza para realzar una imagen

Sumar o restar el Laplaciano de la imagen dependiendo del signo del coeficiente central de la máscara utilizada

$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y) \\ f(x,y) + \nabla^2 f(x,y) \end{cases}$$

Coeficiente central de la máscara negativo

Coeficiente central de la máscara positivo

Filtros Laplacianos alternativos (II)

Resultado de sustraer el Laplaciano (una única máscara)

$$g(x,y) = f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)]$$

$$g(x,y) = 5f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)]$$

Incluye diagonal

0	-1	0
-1	5	-1
0	-1	0

-1	-1	-1
-1	9	-1
-1	-1	-1

Filtro Gaussiano

La máscara o filtro que responde a:

$$w(x,y) = \frac{e^{-(x^2+y^2)/4\sigma^2}}{\sum_{s=-a}^{a}\sum_{s=-b}^{b} e^{-(s^2+t^2)/4\sigma^2}}$$

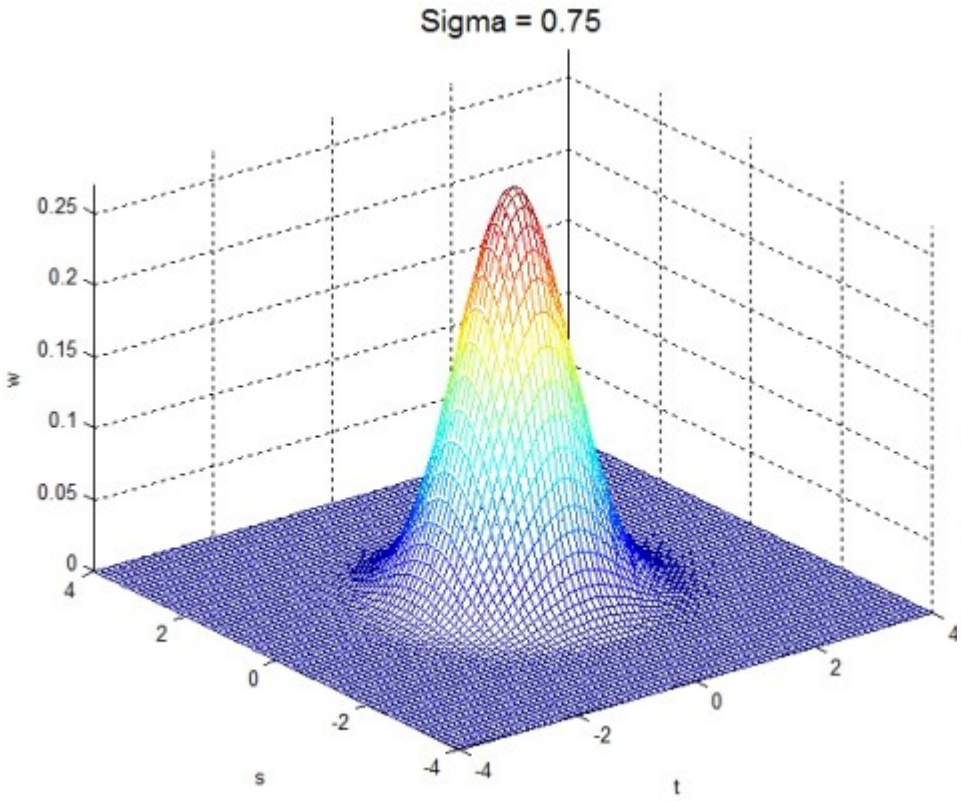
Siendo la matriz del bloque:
m(filas) = 2a + 1
n(columnas) = 2b + 1

ó

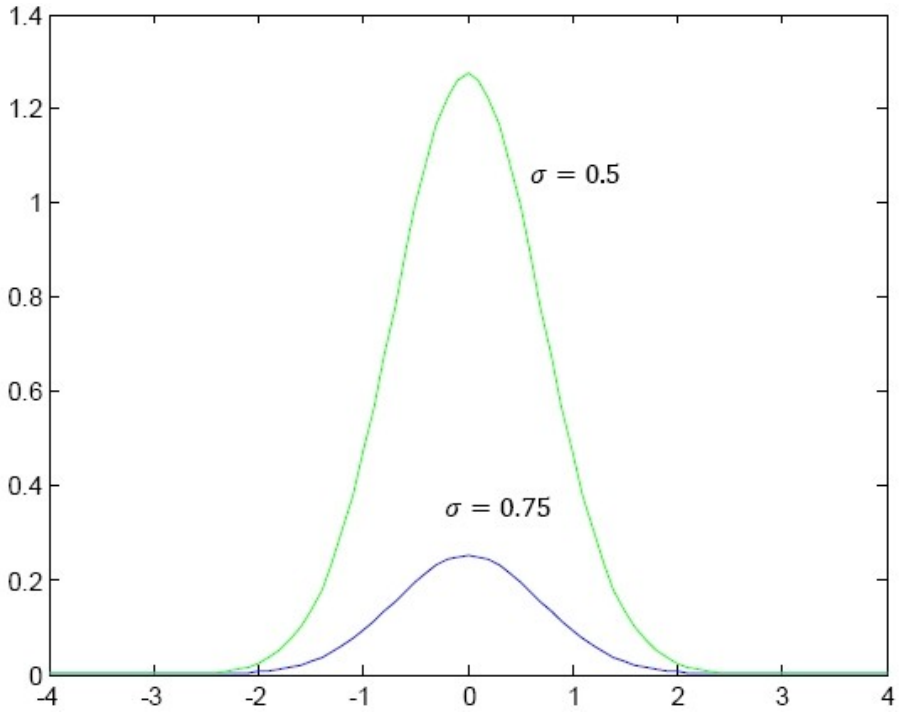
$$w(s,t) = \frac{e^{-\frac{s^2+t^2}{\sigma^2}}}{2\pi\sigma^2}$$

Filtro Gaussiano (II)

$$w(s,t) = \frac{e^{-\frac{s^2+t^2}{\sigma^2}}}{2\pi\sigma^2}$$

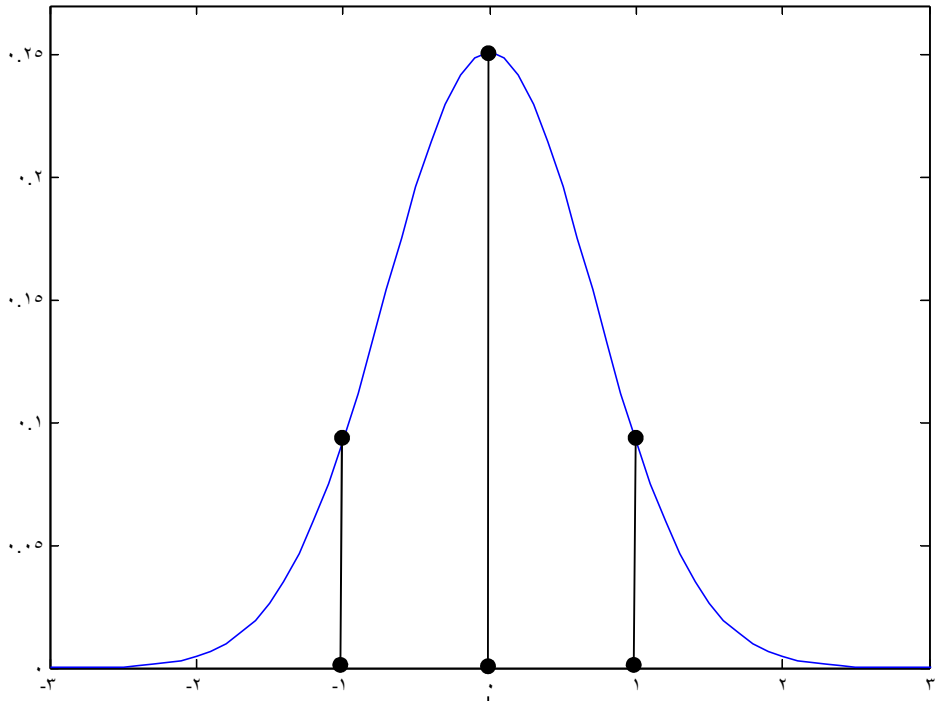


Influencia del parámetro σ



Filtro Gaussiano (III)

- Máscara a partir de función no lineal
- Filtro que se aplica es lineal



0.047	0.1163	0.047
0.1163	0.2829	0.1163
0.047	0.1163	0.047

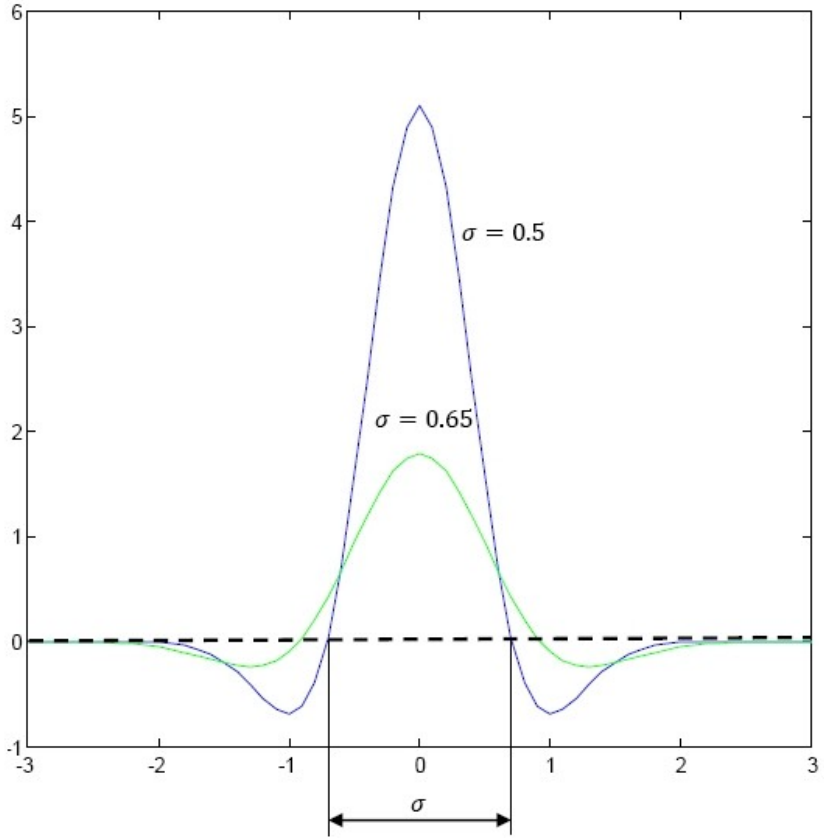
```
%Define máscara
>> fspecial('gaussian',3,0.5)
ans =
0.0113 0.0838 0.0113
0.0838 0.6193 0.0838
0.0113 0.0838 0.0113
```

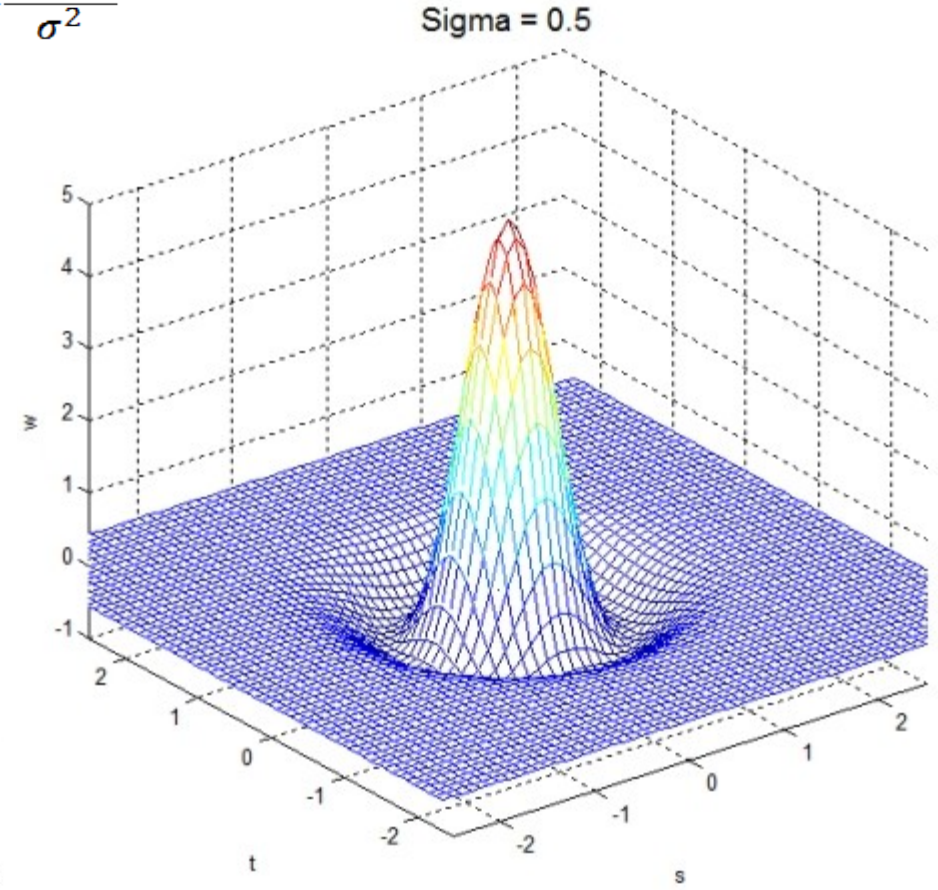
```
% Máscara y filtro
>> w=fspecial('gaussian',3,0.5);
>> I2=imfilter(I,w,'replicate');
```

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

Laplaciana de la gausiana

$$w(s,t) = \frac{1}{2\pi\sigma^4}\left[1 - \frac{s^2+t^2}{\sigma^2}\right]e^{-\frac{s^2+t^2}{\sigma^2}}$$

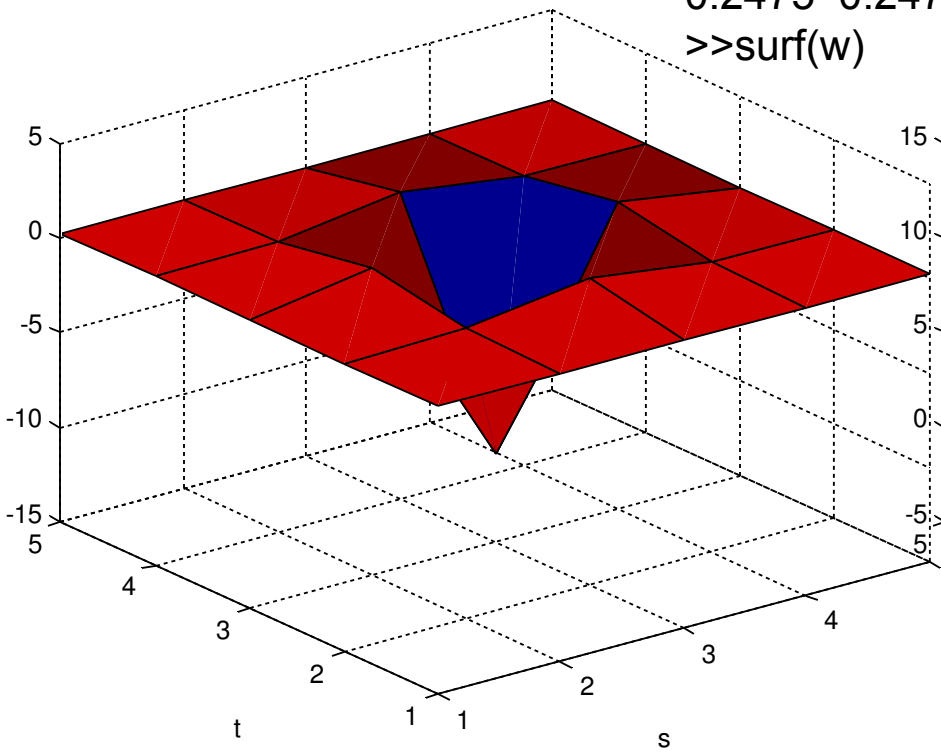




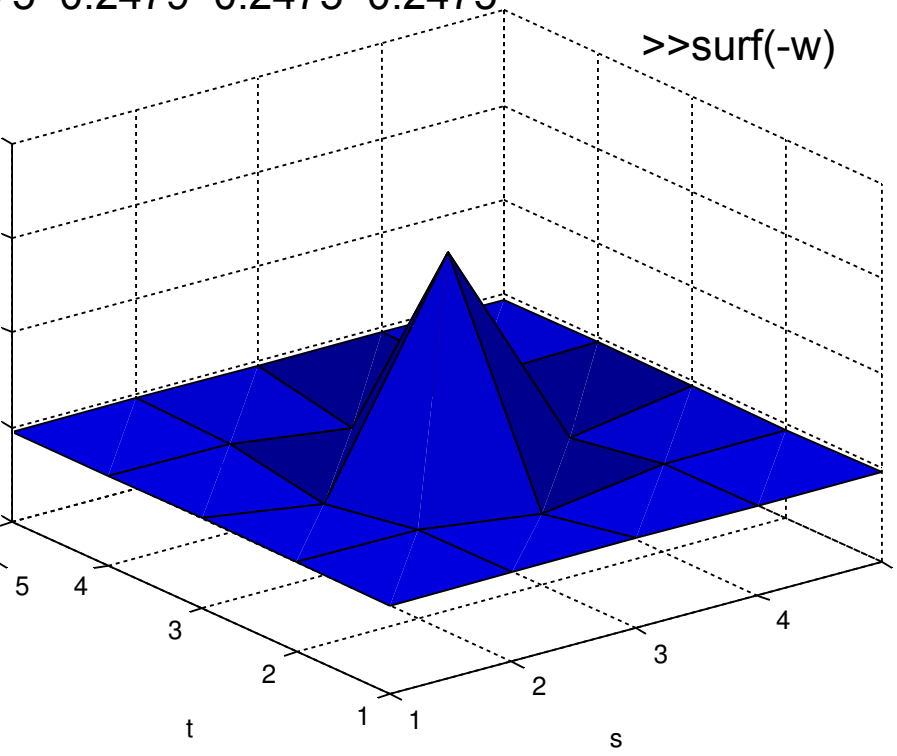
Laplaciana de la gausiana (II)

```
>> w=fspecial('log',5,0.4)
ans =
0.2475  0.2475  0.2479  0.2475  0.2475
0.2475  0.3545  1.2336  0.3545  0.2475
0.2479  1.2336 -10.31   1.2336  0.2479
0.2475  0.3545  1.2336  0.3545  0.2475
0.2475  0.2475  0.2479  0.2475  0.2475
```

>>surf(w)



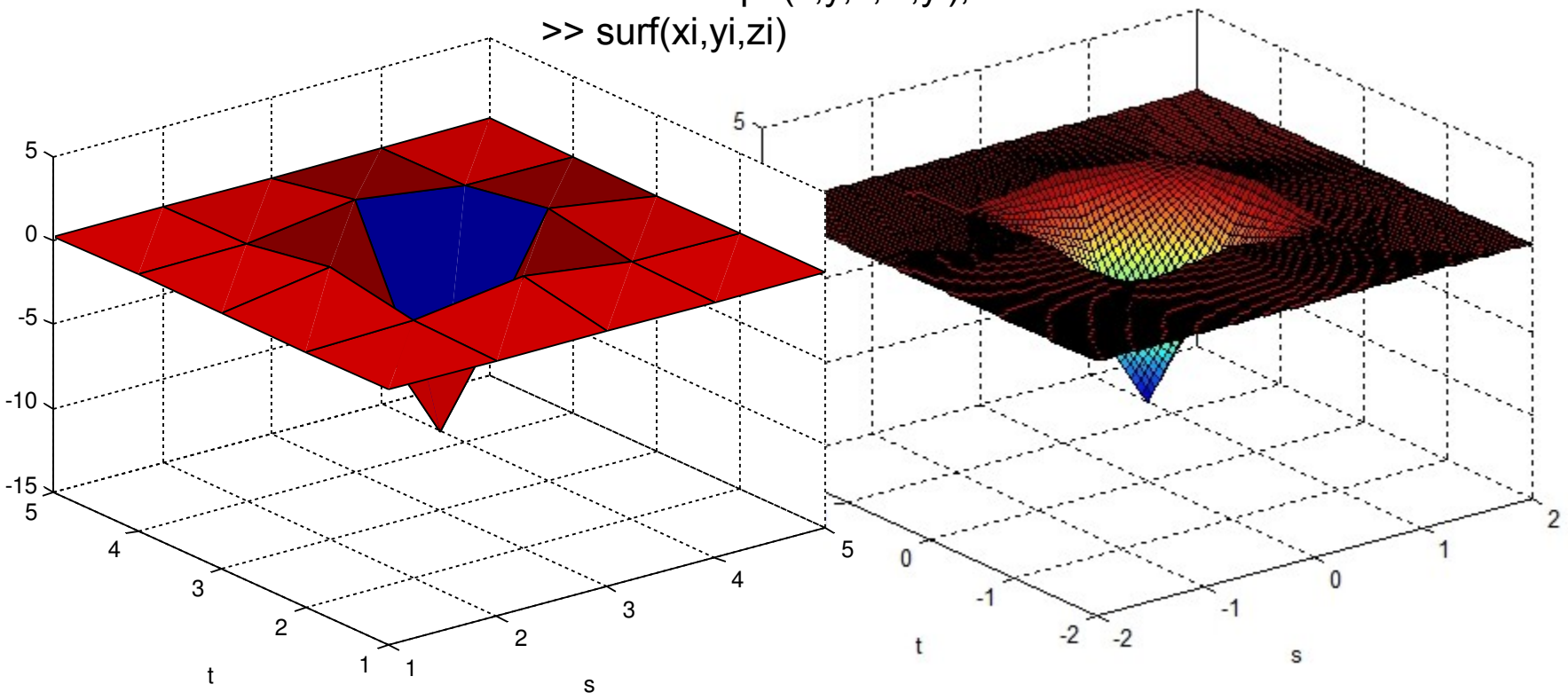
>>surf(-w)



Laplaciana de la gausiana (III)

Mejorando detalles del tipo de filtros

```
>> w=fspecial('log',5,0.4)
>> [x, y]=meshgrid(-2:1:2)
>> [xi, yi]=meshgrid(-2:.05:2);
>> zi = interp2(x,y,z,xi,yi);
>> surf(xi,yi,zi)
```



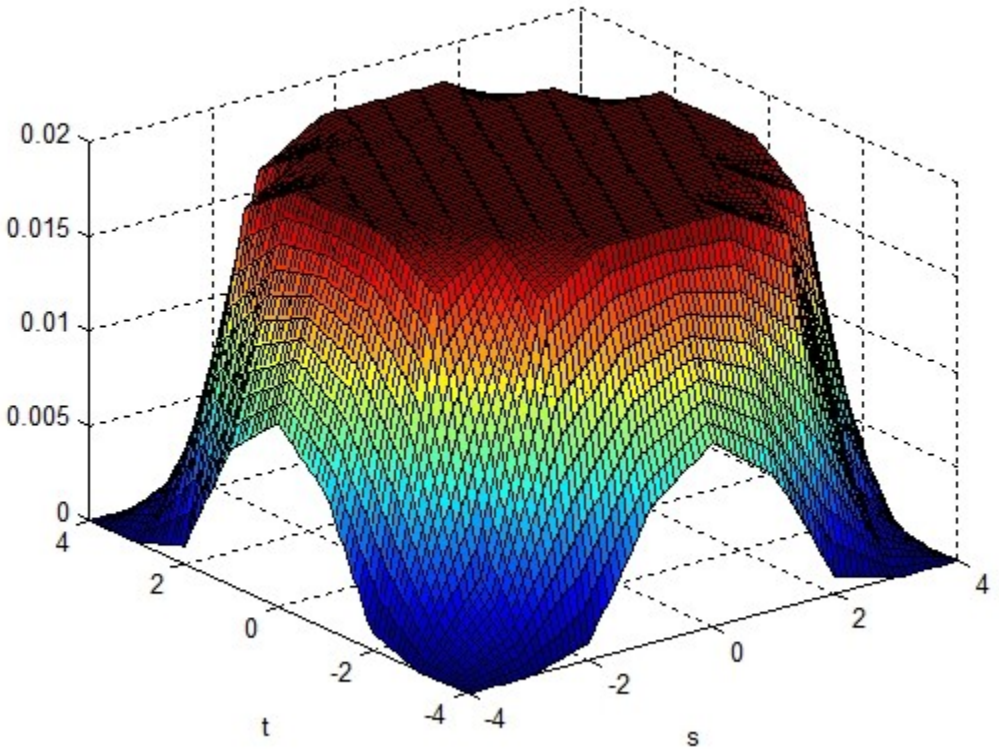
Máscaras con *fspecial*

Promedio

```
>> w = fspecial('average',5)
w =
    0.0400    0.0400    0.0400    0.0400    0.0400
    0.0400    0.0400    0.0400    0.0400    0.0400
    0.0400    0.0400    0.0400    0.0400    0.0400
    0.0400    0.0400    0.0400    0.0400    0.0400
    0.0400    0.0400    0.0400    0.0400    0.0400
>> 1/25
ans =
    0.0400
```

Disco

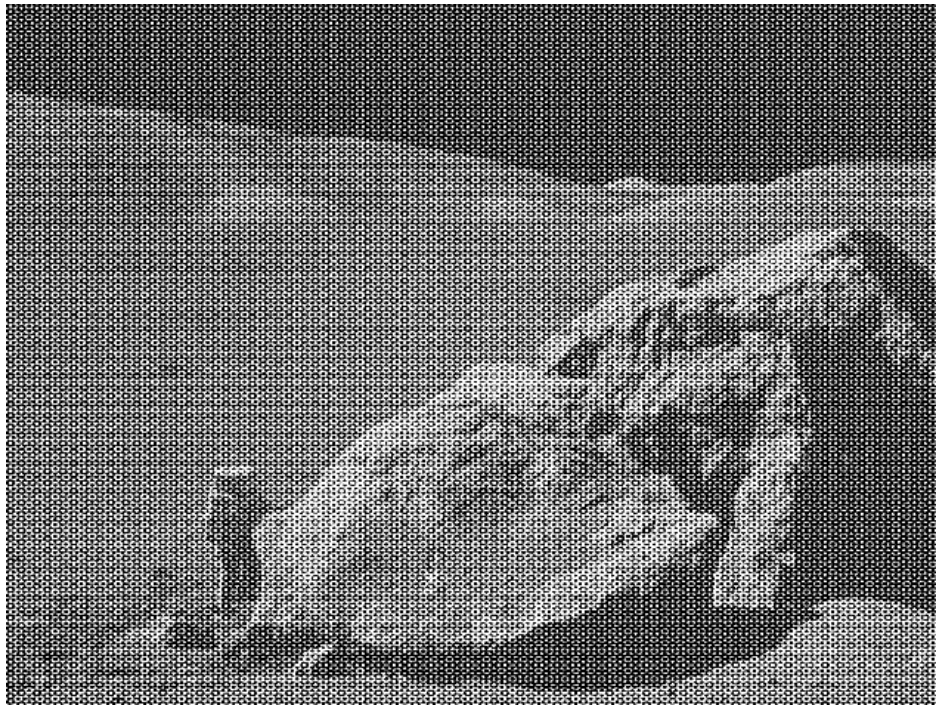
```
>> w = fspecial('disk',4)
```



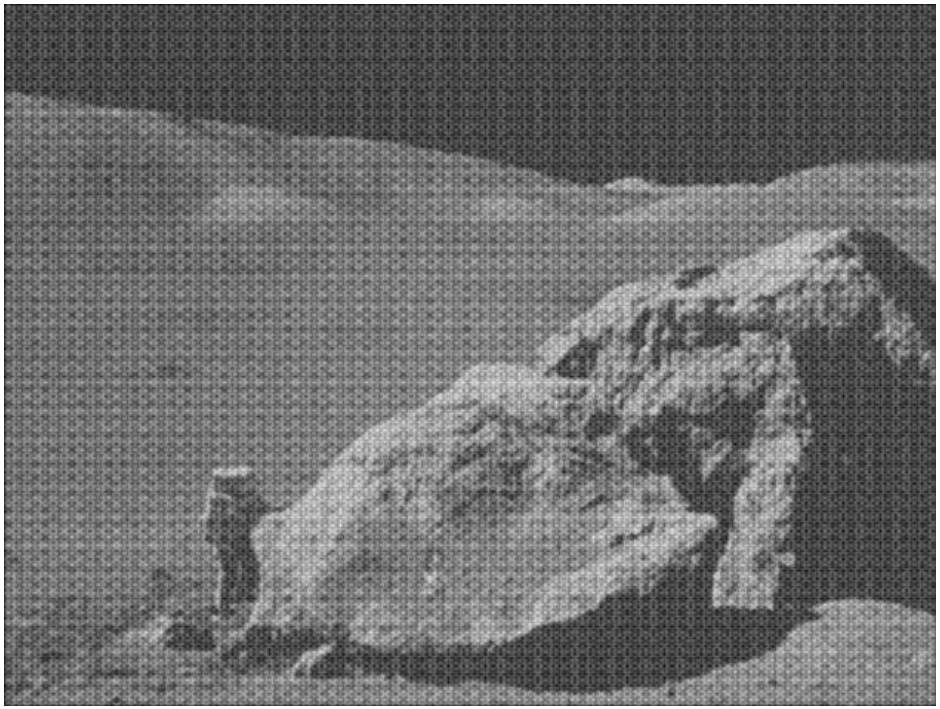
Ejemplo: Promedio

```
>> w=fspecial('average');
```

Imagen Original



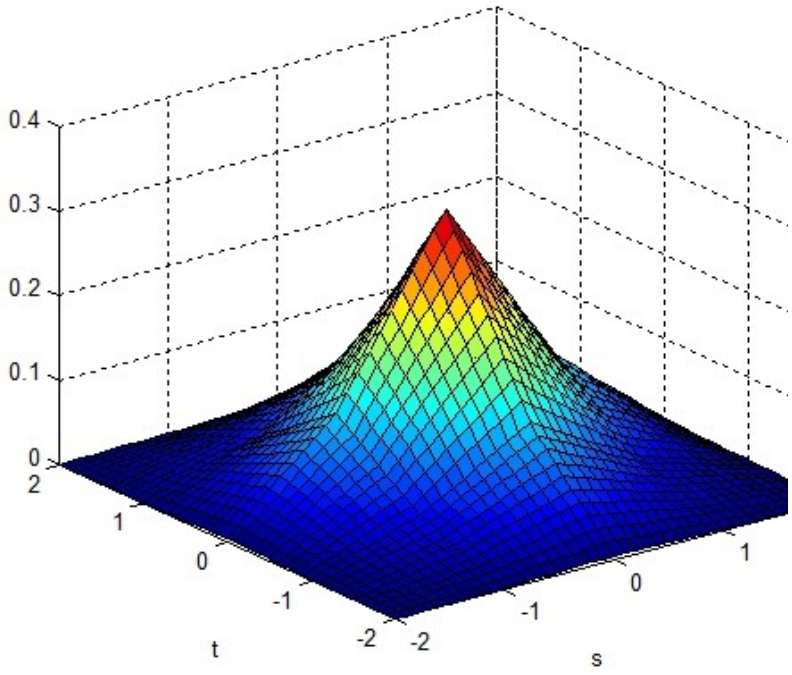
Resultado del Filtro



Máscaras con *fspecial* (II)

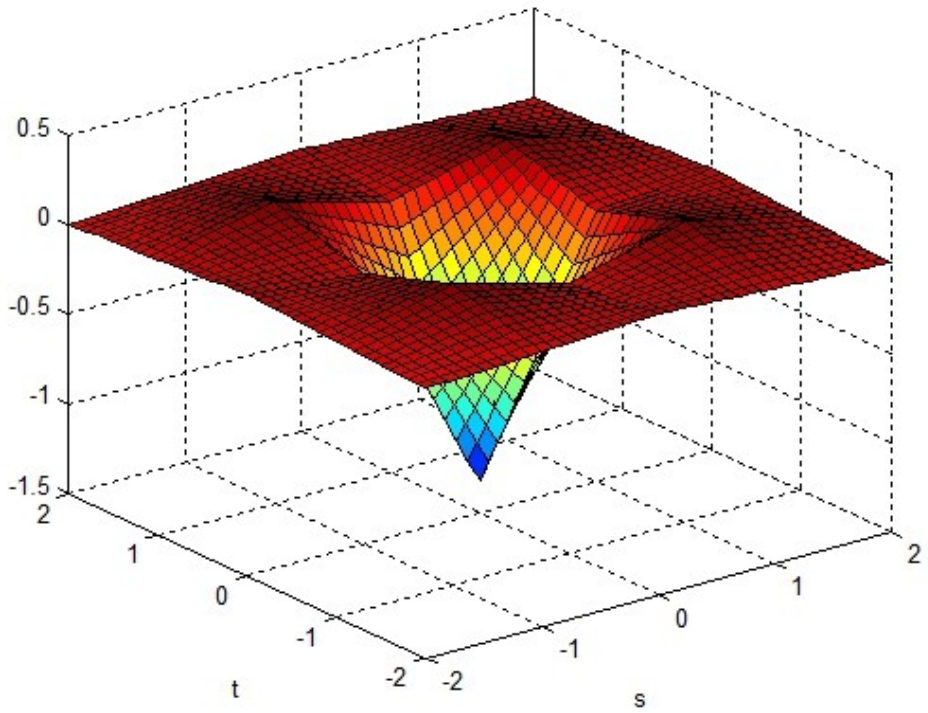
Gausiana

>>z = fspecial('gaussian',5,0.7)



Laplaciana de la gausiana

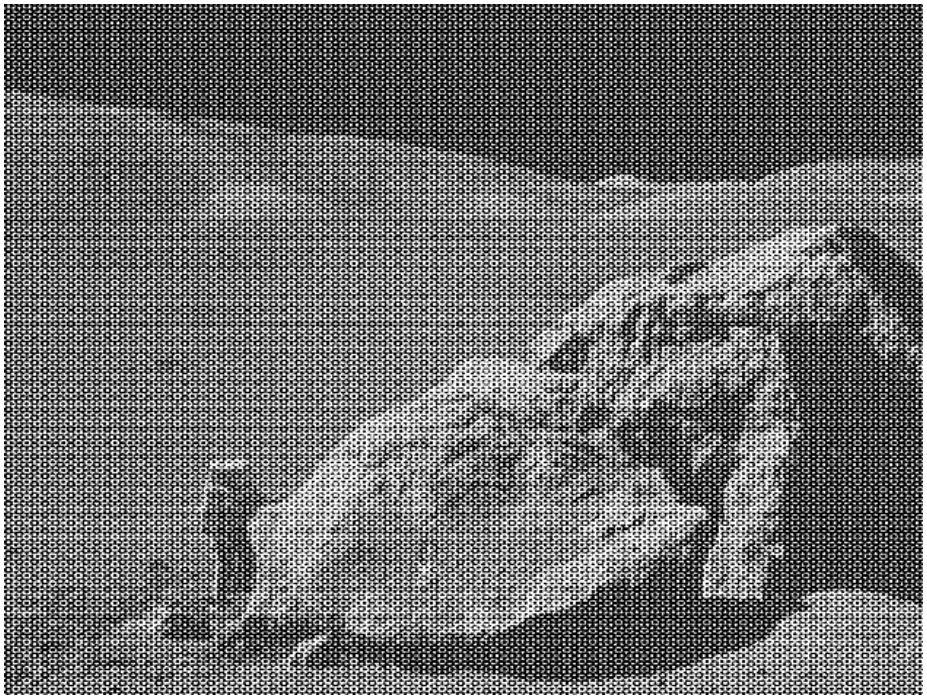
>> z = fspecial('log',5,0.7)



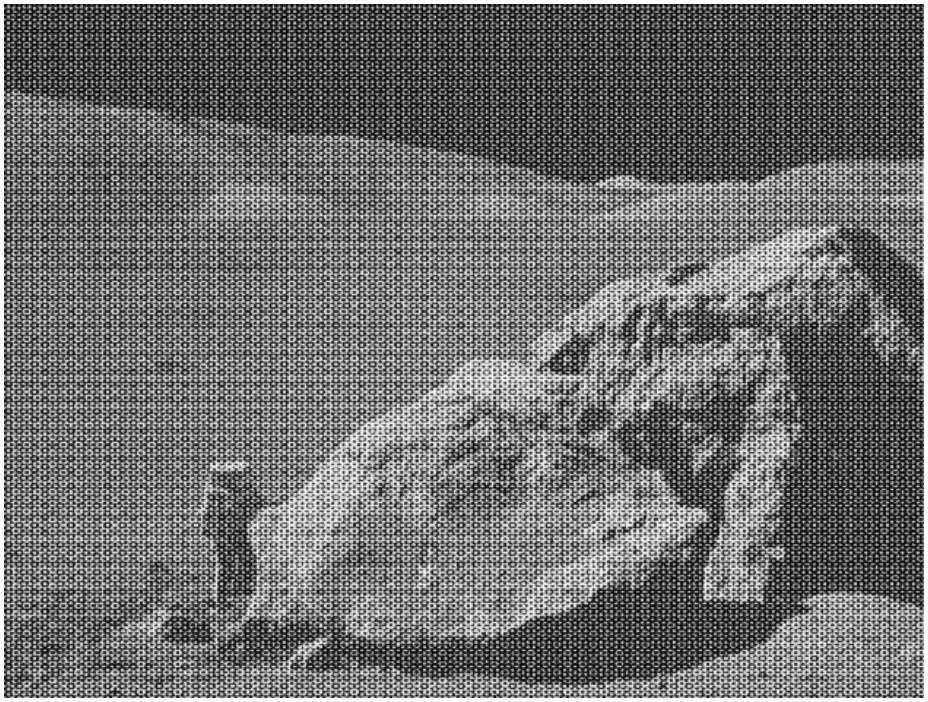
Ejemplo: Filtro Gaussiano

```
>> w=fspecial('gaussian');
```

Imagen Original



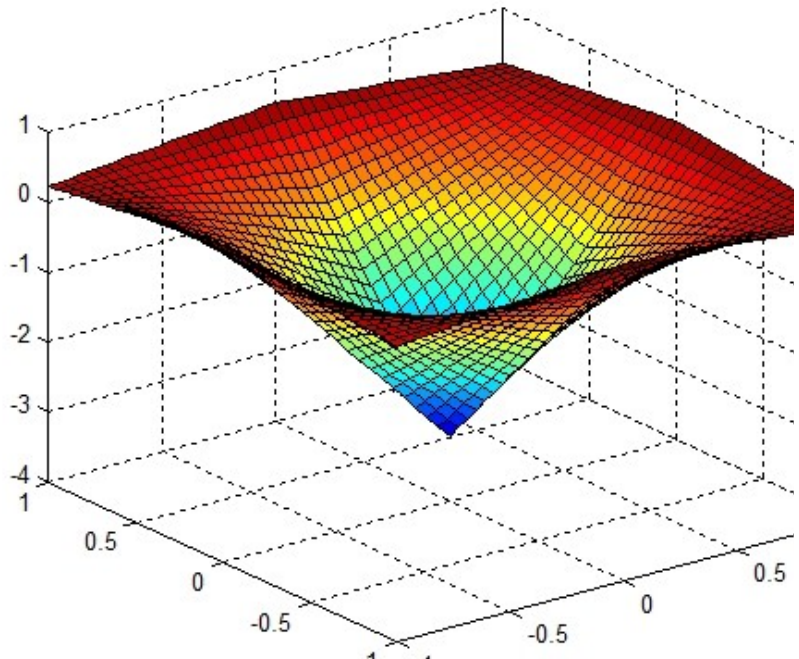
Resultado del Filtro



Máscaras con *fspecial* (III)

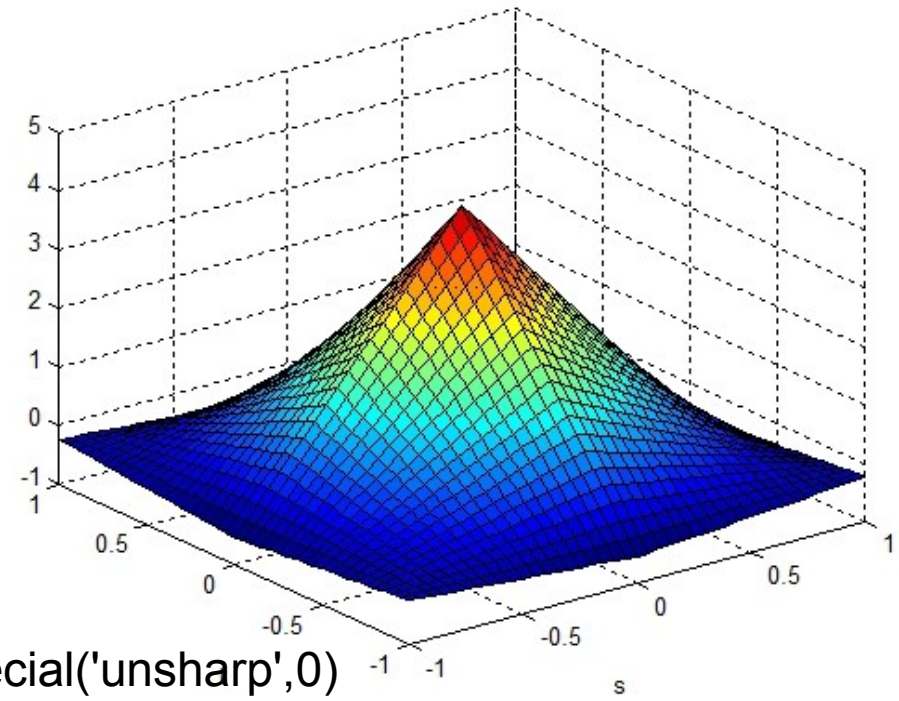
Laplaciana

>>z = z = fspecial('laplacian',0.3)



Mejora de contraste

>> z = fspecial('unsharp',0.3)



>> fspecial('laplacian',0)

0	1	0
1	-4	1
0	1	0

>> fspecial('unsharp',0)

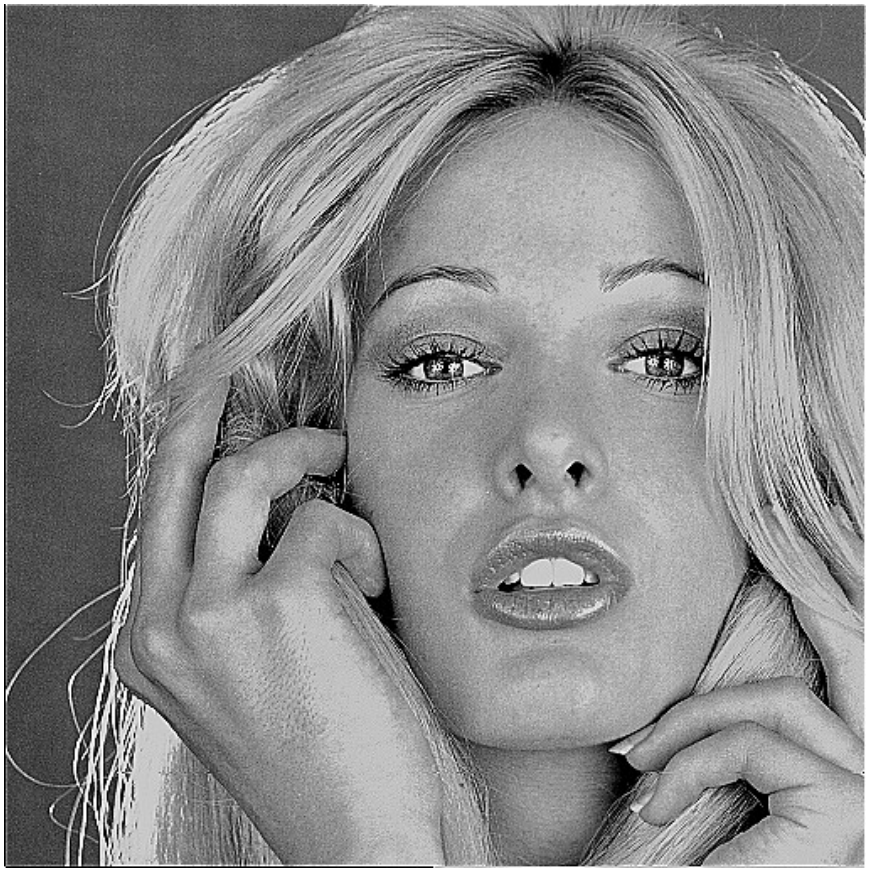
0	-1	0
-1	5	-1
0	-1	0

Filtro para acentuar contraste

```
>> w=fspecial('unsharp');
```

Imagen Original

Resultado del Filtro



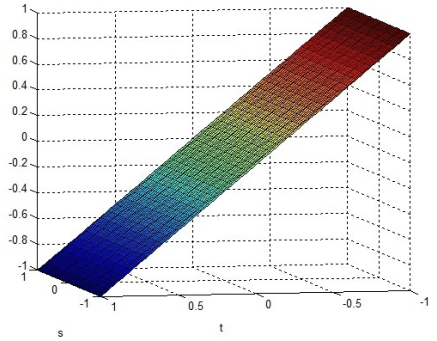
Máscaras con *fspecial* (IV)

Acentuar transiciones horizontales y verticales

Prewit: Acentuar transiciones horizontales

Máscara: w =

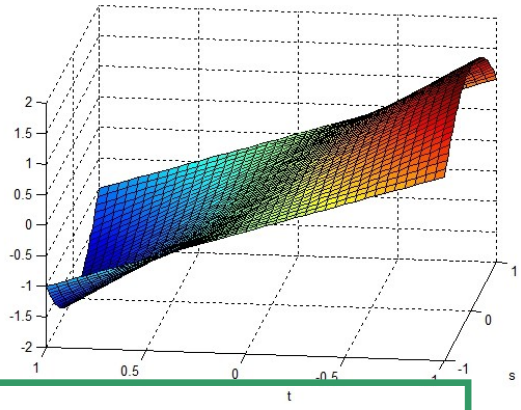
$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$



Sobel: Acentuar transiciones horizontales

Máscara: w =

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

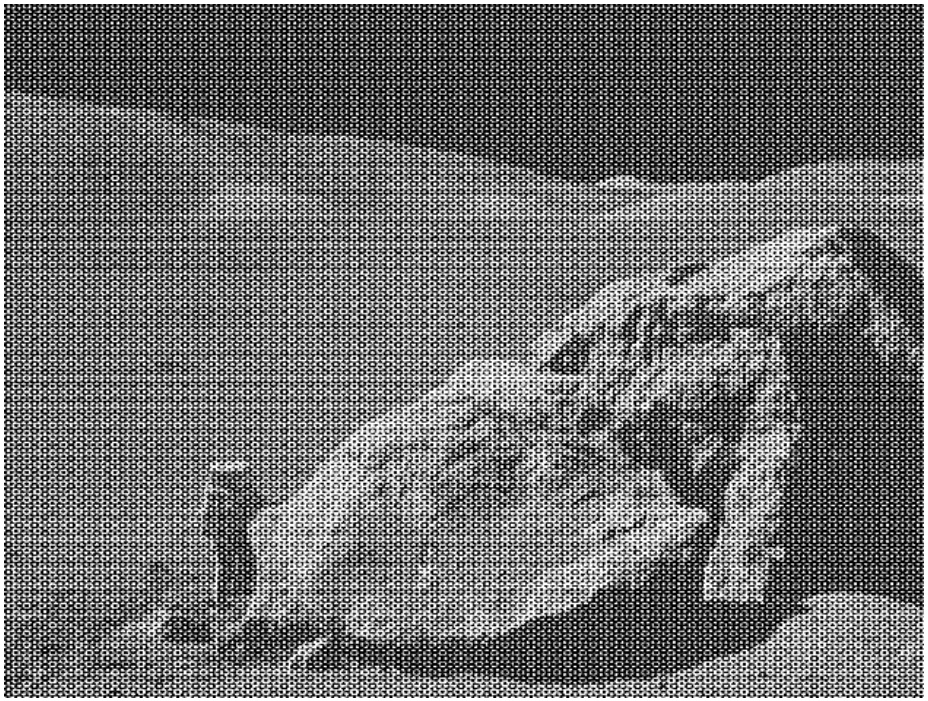


Acentuar transiciones verticales: Transpuesta de la matriz

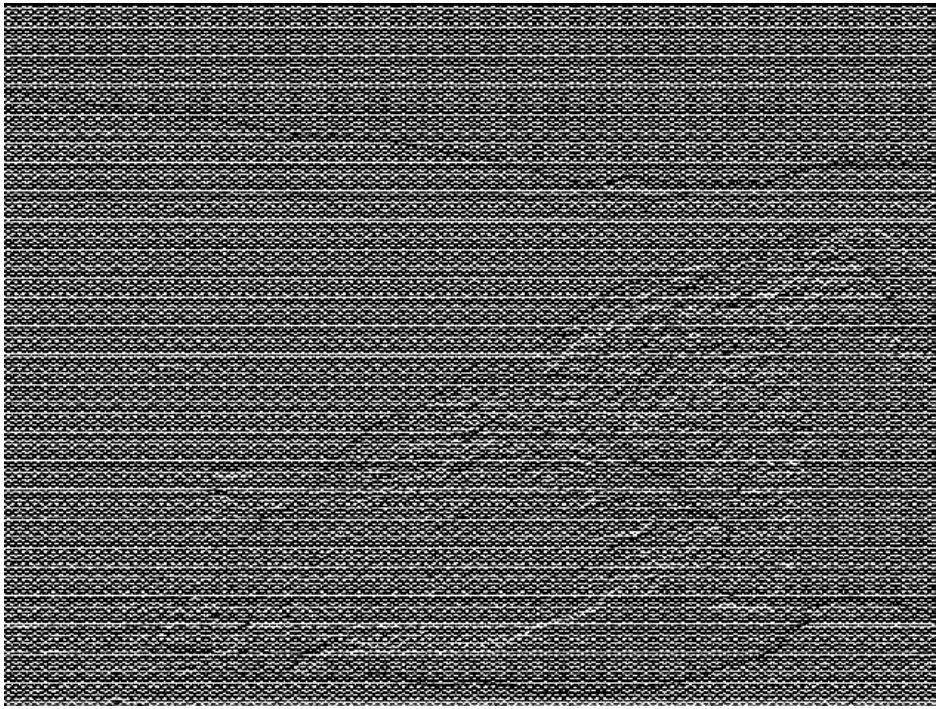
Ejemplo: Filtro Prewit

```
>> w=fspecial('prewitt');
```

Imagen Original



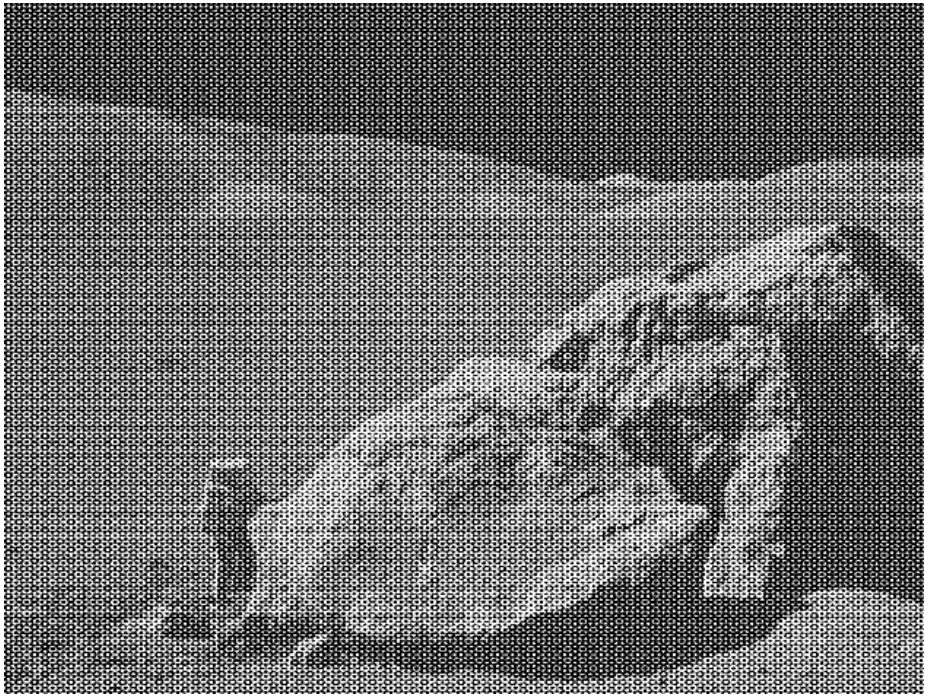
Resultado del Filtro



Ejemplo: Filtro Sobel

```
>> w=fspecial('sobel');
>> hp=transp(w); % Acentúa vertical
```

Imagen Original



Resultado del Filtro