Visión artificial III

Preprocesamiento

Preprocesamiento

- Su objetivo es producir una nueva imagen con mayor calidad que facilite y simplifique etapas posteriores.
- En el dominio del espacio:
  - Operaciones sobre puntos individuales.
  - Histogramas.
  - Operaciones de vecindad.
- En el dominio de la frecuencia:
  - Transformadas de Fourier, Coseno, ...

Beneficiarios

- Observador humano:
  - el objetivo es conseguir una imagen más agradable y más apta para la interpretación y conservación.
- Ordenador:
  - el objetivo es conseguir una imagen más sencilla de procesar computacionalmente. El resultado no tiene por qué ser una imagen interpretable por un ser humano.

Operaciones sobre puntos individuales

- Implica la generación de una nueva imagen modificando el valor del un píxel en una localización concreta.
- El proceso se repite en todos los puntos de la imagen.
- El píxel es cambiado sin ninguna información del entorno.
- La función a aplicar puede ser lineal o no lineal.
- La imagen resultado tiene la misma dimensión que la original.

Operaciones sobre puntos individuales

- Operaciones aritméticas:
  - imadd (imagen,valor)
    - Equivale a:

      uint8(double(imagen)+valor)
  - imsubtract (imagen,valor)
  - immultiply (imagen,valor)
  - imdivide (imagen,valor)

Operaciones sobre puntos individuales

- Operador inverso o negativo:
  - Devuelve la inversa de la imagen de entrada.
  - 255 – P
  - MATLAB: imcomplement (imagen)

```
imagen = imread('cameraman.tif');
subplot(1,2,1);
imshow(imagen);
subplot(1,2,2);
imshow(imcomplement(imagen));
%imshow(255-imagen);
```

Operaciones sobre puntos individuales

- Operador umbral:
  - Crea una imagen binaria a partir de una imagen de niveles de gris donde el nivel de transición viene dado por el valor umbral (θ).
  - $$Q = \begin{cases} 0 & \text{si } P \leq \theta \\ 255 & \text{si } P > \theta \end{cases}$$

```
umbral = 128;
imagen = imread('cameraman.tif');
subplot(1,2,1);
imshow(imagen);
subplot(1,2,2);
imagen_binaria = zeros(size(imagen));
indices = find(imagen > umbral);
imagen_binaria(indices) = 255;
imshow(imcomplement(imagen_binaria));
```

Operaciones sobre puntos individuales

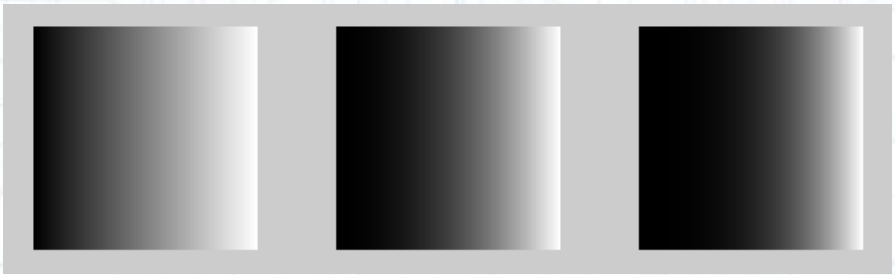
- Operador intervalo:
  - Crea una imagen donde los valores de gris entre θ_1 y θ_2 toman un valor determinado.
  - $$Q = \begin{cases} 255 & \text{para } P <= \theta_1 \text{ ó } P >= \theta_2 \\ P \text{ ó } 0 & \text{para } \theta_1 < P < \theta_2 \end{cases}$$

```
umbral01 = 40;
umbral02 = 100;
imagen = imread('cameraman.tif');
subplot(1,2,1);
imshow(imagen);
subplot(1,2,2);
imagen_salida =
repmat(255,size(imagen));
indices = find(imagen > umbral01 &
imagen < umbral02);
imagen_salida(indices) = 0;
imshow(imcomplement(imagen_salida));
```

Operaciones sobre puntos individuales

- Funciones cuadráticas ($P^2/255$) y cúbicas ($P^3/255^2$):
  - Las imágenes resultantes son más oscuras, pero el contraste es mayor en los píxeles oscuros.

```
imagen = uint8(repmat([0:255],255,1));
subplot(1,3,1);
imshow(imagen);
subplot(1,3,2);
imshow(uint8(double(imagen).^2/255));
subplot(1,3,3);
imshow(uint8(double(imagen).^3/255^2));
```



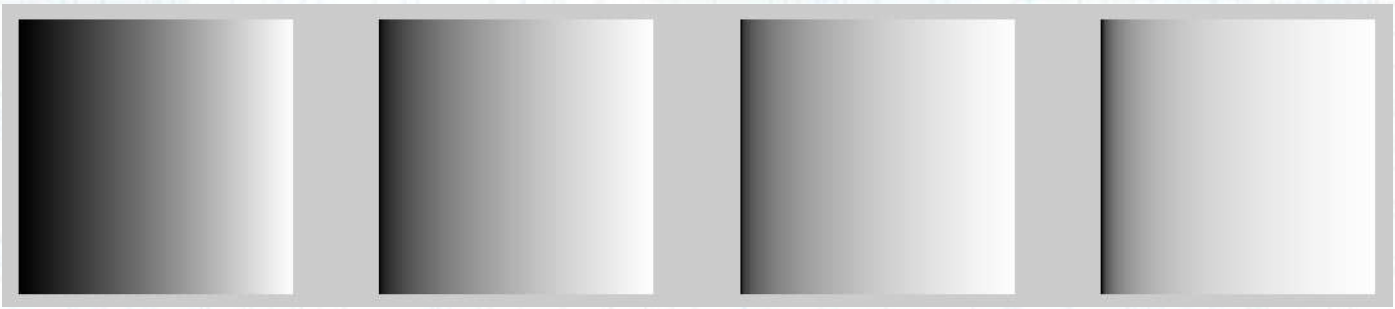
Operaciones sobre puntos individuales

- Funciones raíz cuadrada ($Q = \sqrt{255 \cdot P}$), raiz cúbica ($Q = \sqrt[3]{255^2 \cdot P}$) y función logarítmica ($Q = 255 \frac{\ln(1+P)}{\ln(1+255)}$):
  - Las imágenes resultantes son más claras, pero el contraste es mayor en los píxeles claros.

```
imagen = uint8(repmat([0:255],255,1));
subplot(1,4,1);
imshow(imagen);
subplot(1,4,2);
imshow(uint8(sqrt(255*double(imagen))));
subplot(1,4,3);
imshow(uint8(nthroot(255^2*double(imagen),3)));
subplot(1,4,4);
imshow(uint8(255*log(1+double(imagen))/log(256)));
```

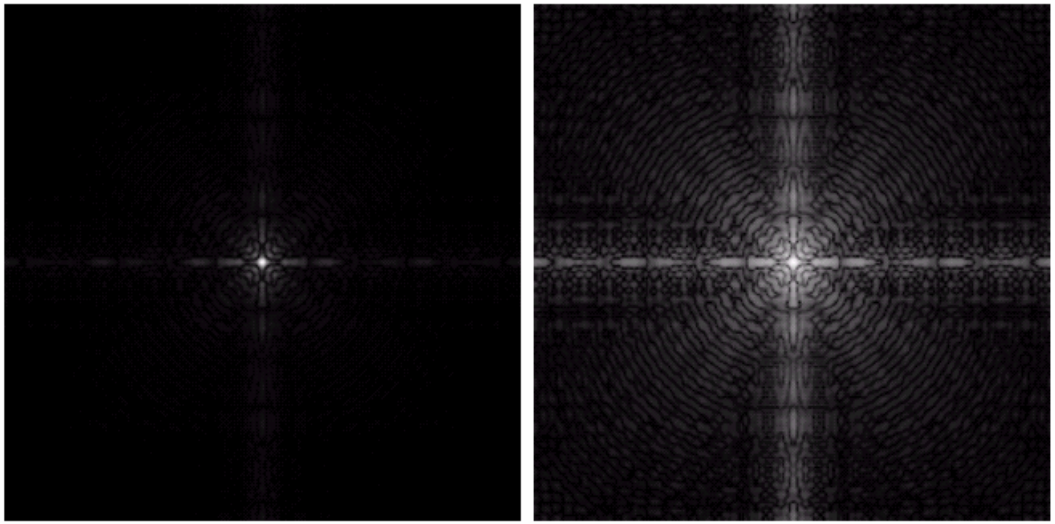
Operaciones sobre puntos individuales

- Funciones raíz cuadrada ($Q = \sqrt{255 \cdot P}$), raiz cúbica ($Q = \sqrt[3]{255^2 \cdot P}$) y función logarítmica ($Q = 255 \frac{\ln(1+P)}{\ln(1+255)}$):



Operaciones sobre puntos individuales

- Función logarítmica:
  - Expande valores oscuros para resaltar detalles de los píxeles con valores más bajos.
  - c*log(1+double(imagen)) donde *c* es una cte.



Histogramas

- Es una representación gráfica de las frecuencias de grises en una imagen.

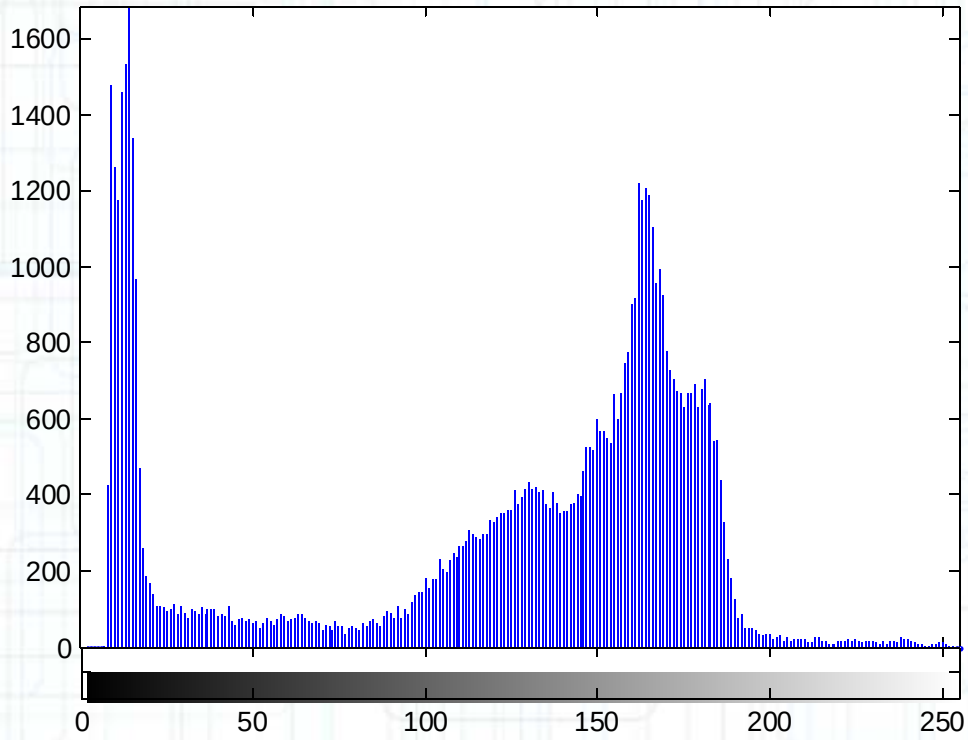
$$H(g) = \frac{N_g}{N_{total}}$$

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

Gray Value	Count	Rel. Freq.
0	2	.05
1	2	.05
2	4	.11
3	6	.17
4	7	.20
5	8	.22
6	6	.17
7	1	.03

Histogramas

```
imagen = imread('cameraman.tif');
imhist(imagen), axis tight;
```



Histogramas

- El histograma proporciona información estadística de la imagen (muy útil para conocer cómo se ha producido el proceso de formación):
  - Brillo: valor medio de la imagen (del histrograma).
  - Contraste: Dispersión de los niveles de gris (relacionado con la varianza, ya que, a mayor varianza, mayor contraste).

Histogramas

- El histograma proporciona información estadística de la imagen (muy útil para conocer cómo se ha producido el proceso de formación):
  - Energía:
    - Se calcula como:

$$\Sigma H(g)^2$$

    - Indica el grado de dispersión de los niveles de gris (valor máxima cuando sólo hay un nivel de gris).

Histogramas

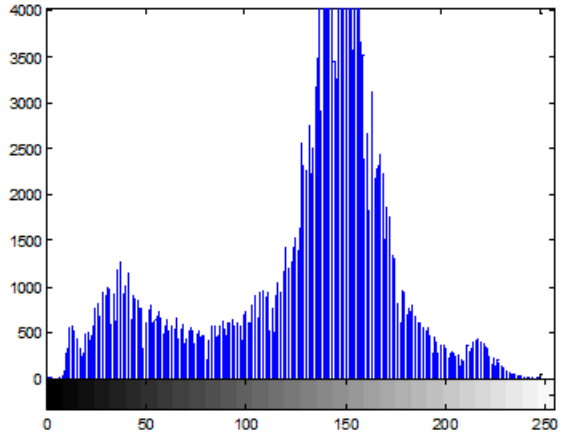
- El histograma proporciona información estadística de la imagen (muy útil para conocer cómo se ha producido el proceso de formación):
  - Entropía:
    - Se calcula como:

      $$- \Sigma H(g) * \log (H(g))$$

    - Indica el desorden en la imagen, es decir, a más entropía, más niveles de grises participan en la imagen.

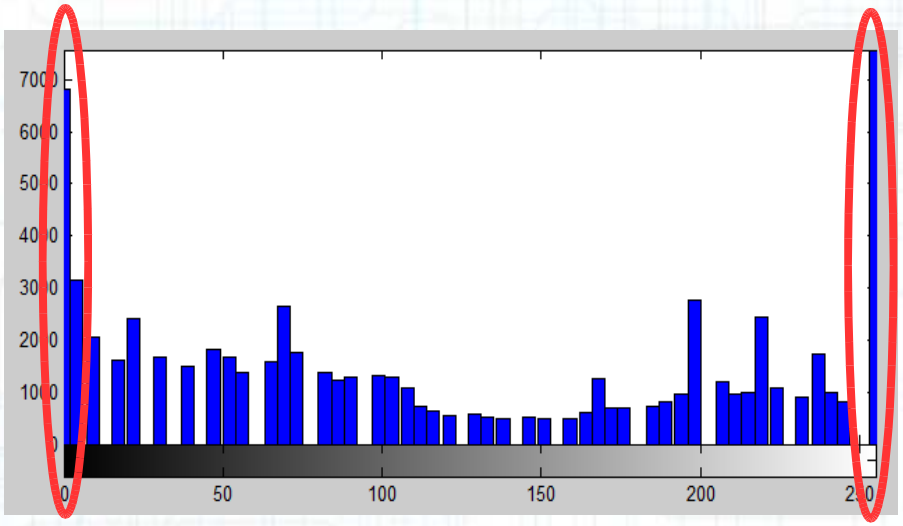
Histogramas

- Picos (peaks): Muchos pixeles concentrados en pocos niveles de gris.
- Llanos (plains): Pocos pixeles distribuidos sobre un gran rango de niveles de gris.



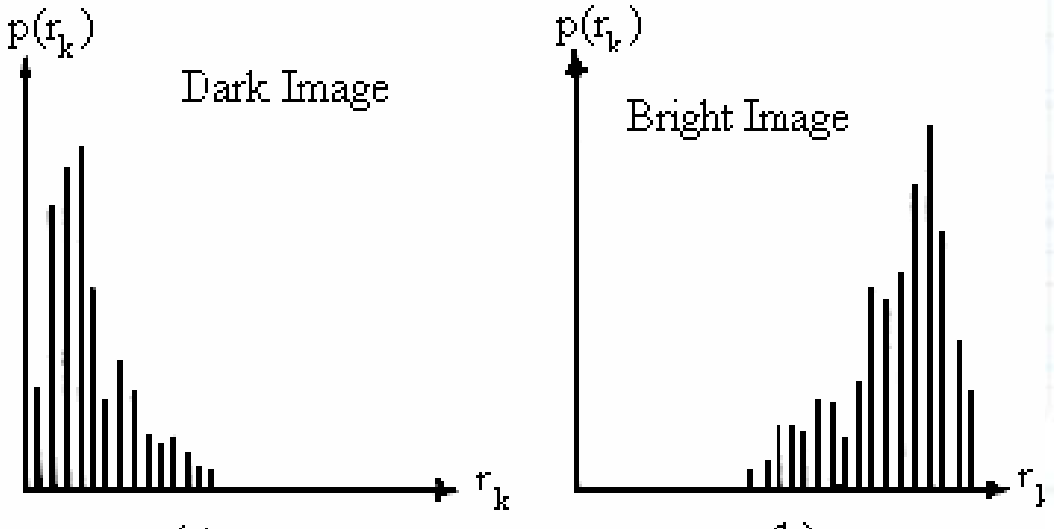
Histogramas

- Saturación: El histograma presenta valores muy altos en sus extremos del rango dinámico (forma de U).



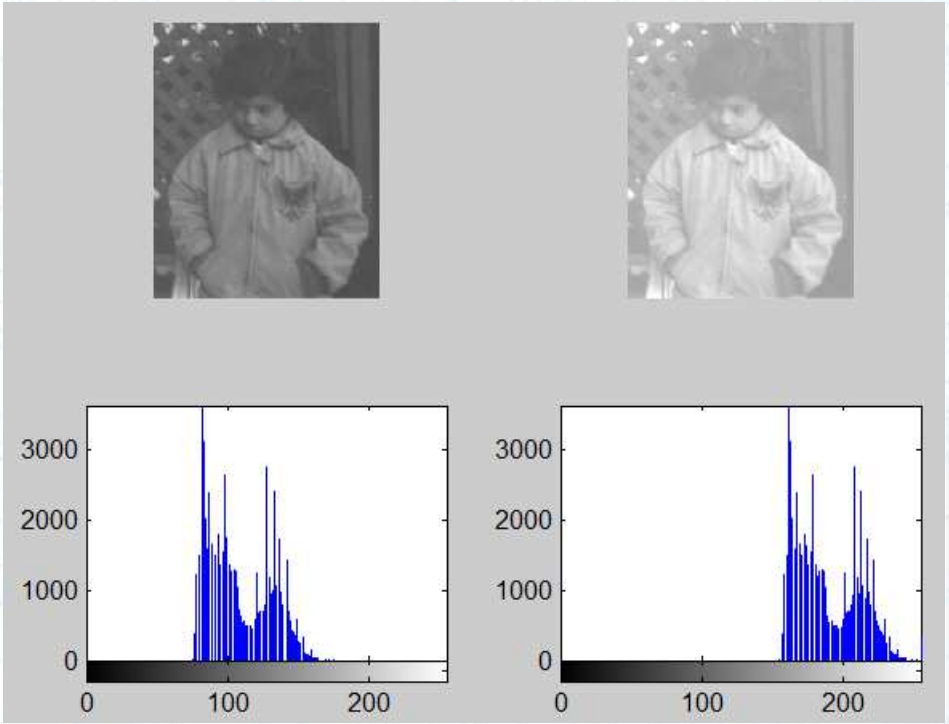
Histogramas

- Brillo:
  - Imagen oscura (niveles de gris concentrados en la zona baja del rango)
  - Imagen brillante (niveles de gris concentrados en la parte alta del rango)



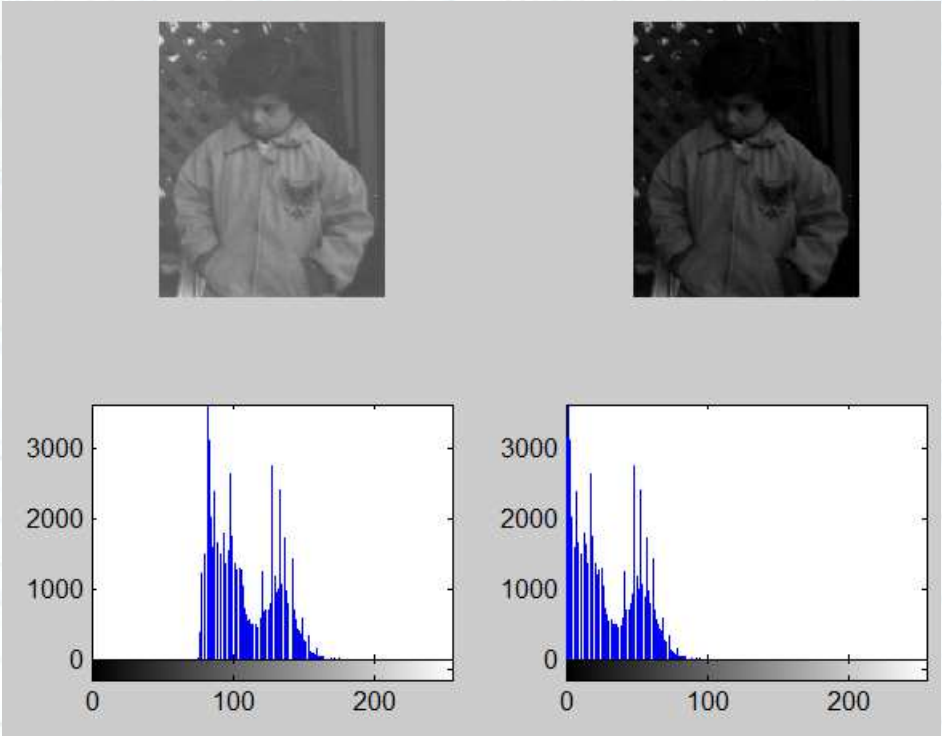
Histogramas

```
inc_brillo = 80;
imagen = imread('pout.tif');
subplot(2,2,1);
imshow(imagen);
subplot(2,2,3);
imhist(imagen), axis tight;
subplot(2,2,2);
imshow(imadd(imagen,inc_brillo));
subplot(2,2,4);
imhist(imagen + inc_brillo), axis tight;
```



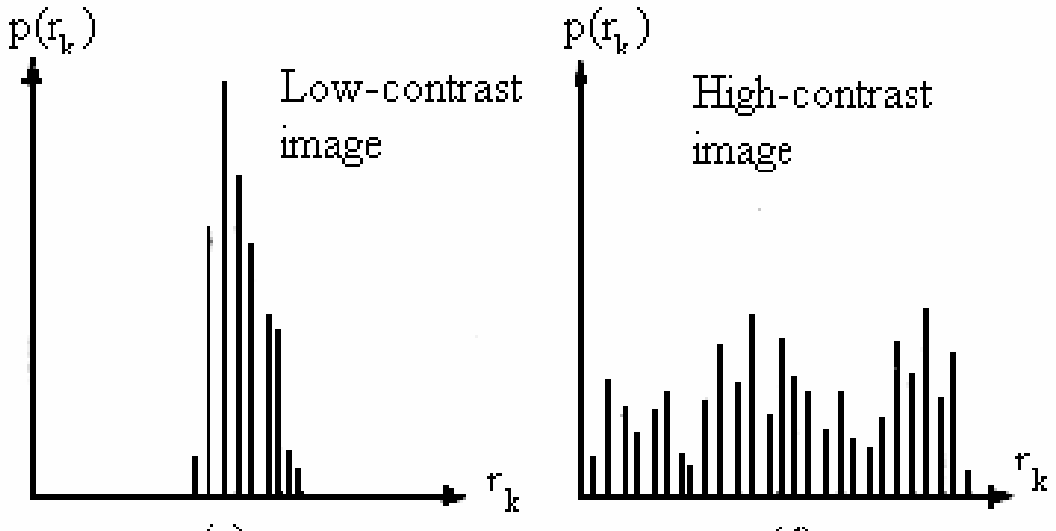
Histogramas

```
dec_brillo = 80;
imagen = imread('pout.tif');
subplot(2,2,1);
imshow(imagen);
subplot(2,2,3);
imhist(imagen), axis tight;
subplot(2,2,2);
imshow(imagen-dec_brillo);
subplot(2,2,4);
imhist(imsubtract(imagen,inc_brillo)), axis tight;
```



Histogramas

- Contraste:
  - Contraste bajo (distribución de niveles de gris concentrados en una deter-minada zona)
  - Contraste alto (amplia distribución de los niveles de gris)



Histogramas

- Modificar el contraste (Ecualizar el histograma):
  - Expande la distribución de los niveles de gris para mejorar el contraste de una imagen.
  - Comprime píxeles en las zonas planas del histograma sobre un conjunto pequeño de niveles de grises.
  - Aplana el histograma.

Histogramas

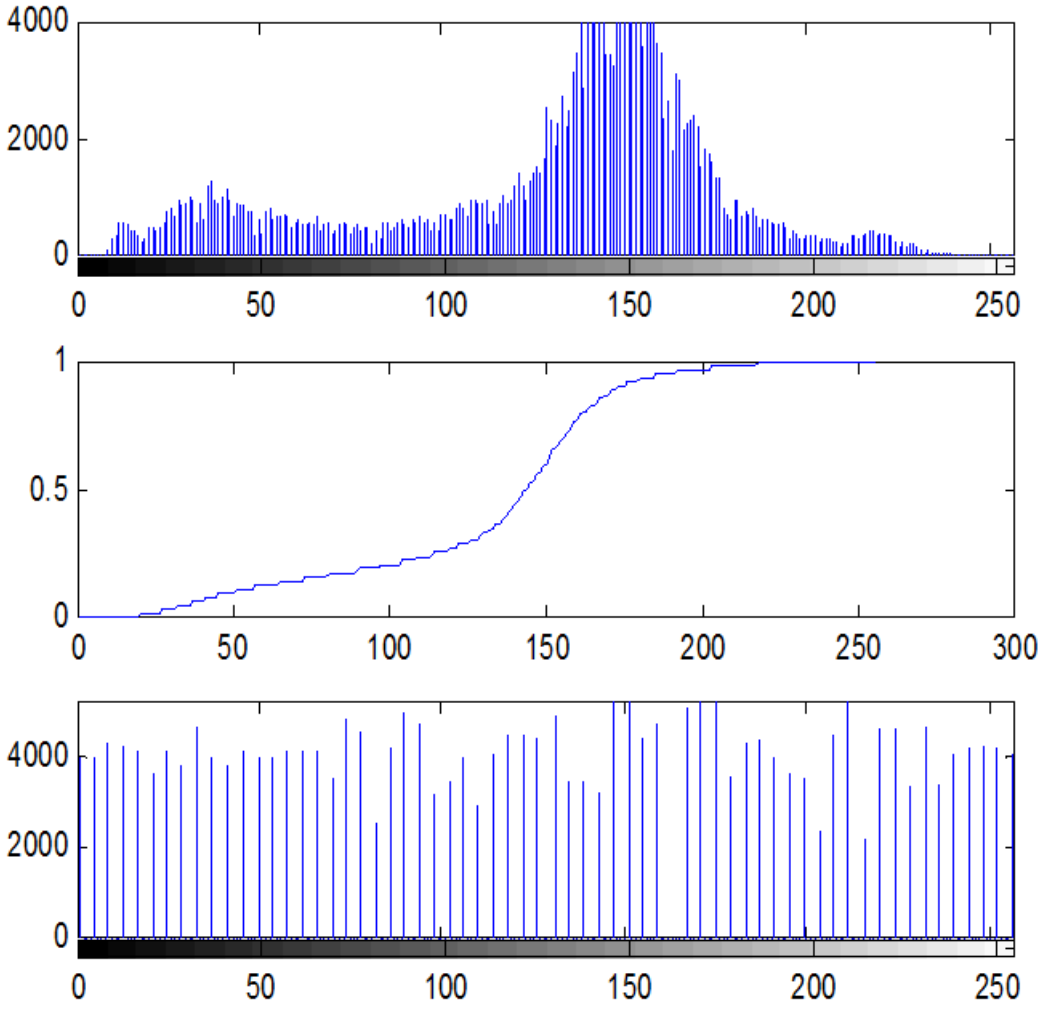
- Ejemplo de equilización:

Grey level i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
n_i	15	0	0	0	0	0	0	0	0	70	110	45	80	40	0	0

$$15/360 = 1/24:$$

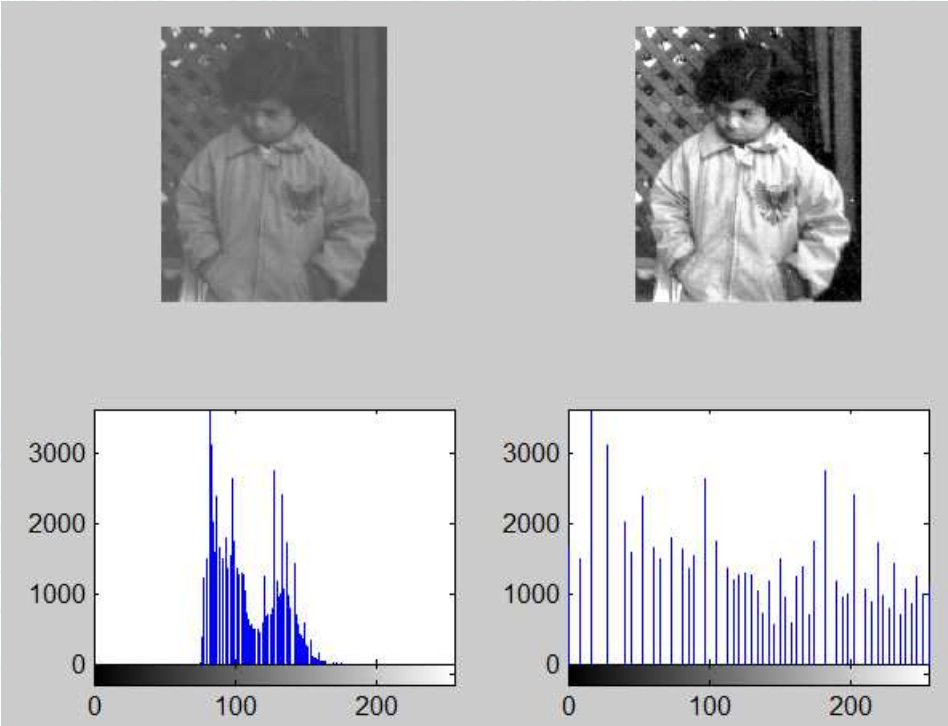
Grey level i	n_i	Σn_i	$(1/24)\Sigma n_i$	Rounded value
0	15	15	0.63	1
1	0	15	0.63	1
2	0	15	0.63	1
3	0	15	0.63	1
4	0	15	0.63	1
5	0	15	0.63	1
6	0	15	0.63	1
7	0	15	0.63	1
8	0	15	0.63	1
9	70	85	3.65	4
10	110	195	8.13	8
11	45	240	10	10
12	80	320	13.33	13
13	40	360	15	15
14	0	360	15	15
15	0	360	15	15

Histogramas



Histogramas

```
imagen = imread('pout.tif');
subplot(2,2,1);
imshow(imagen);
subplot(2,2,3);
imhist(imagen), axis tight;
subplot(2,2,2);
imagen_eq = histeq(imagen);
imshow(imagen_eq);
subplot(2,2,4);
imhist(imagen_eq), axis tight;
```



Histogramas

- Modificar el contraste (Contracción):

$$b = \left(\frac{CMAX - CMIN}{\text{Imax} - \text{Imin}} \right) \cdot (a - \text{Imin}) + CMIN$$

  - a: Nivel de gris de la imagen de entrada
  - Imax e Imin: Mayor y menor nivel de gris de la imagen de entrada.
  - CMAX y CMIN: Máximo y mínimo valor deseado en la compresión del histograma.

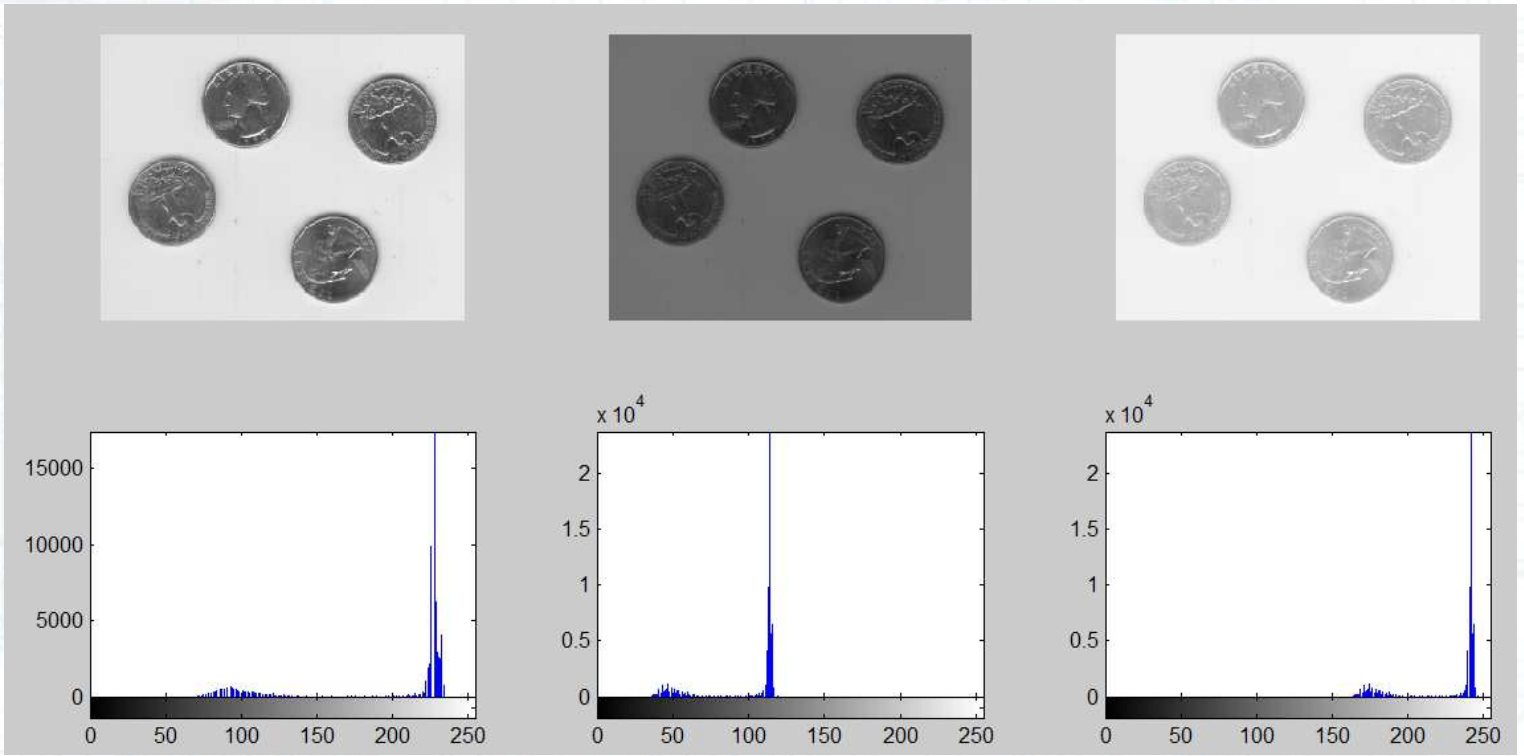
Histogramas

- Modificar el contraste (Contracción):

```
Imin = 0;
Imax = 255;
imagen = imread('eight.tif');
subplot(2,3,1);
imshow(imagen);
subplot(2,3,4);
imhist(imagen), axis tight;
subplot(2,3,2);
CMIN = 0;
CMAX = 127;
imagen_exp = uint8((CMAX-CMIN)/(Imax-Imin)*(double(imagen)-Imin)+ CMIN);
imshow(imagen_exp);
subplot(2,3,5);
imhist(imagen_exp), axis tight;
subplot(2,3,3);
CMIN = 128;
CMAX = 255;
imagen_exp = uint8((CMAX-CMIN)/(Imax-Imin)*(double(imagen)-Imin)+ CMIN);
imshow(imagen_exp);
subplot(2,3,6);
imhist(imagen_exp), axis tight;
```

Histogramas

- Modificar el contraste (Contracción):



¿Qué ha ocurrido?

Histogramas

- Modificar el contraste (Expansión):

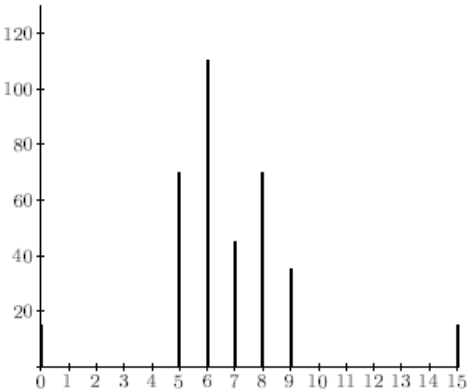
$$b = \left(\frac{a - \mathrm{Imin}}{\mathrm{Imax} - \mathrm{Imin}} \right) \cdot (CMAX - CMIN) + CMIN$$

  - a: Nivel de gris de la imagen de entrada
  - Imax e Imin: Mayor y menor nivel de gris de la imagen de entrada.
  - CMAX y CMIN: Máximo y mínimo valor deseado en la expansión del histograma.

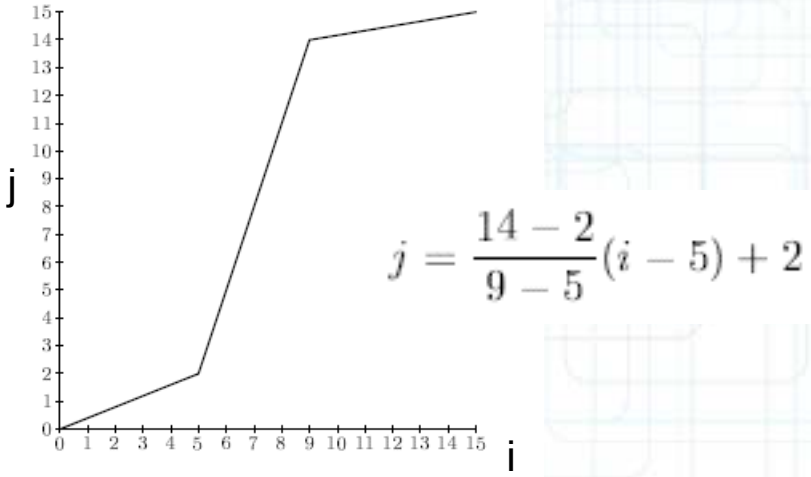
Histogramas

- Modificar el contraste (Expansión):

$$b = \left(\frac{a - \text{Imin}}{\text{Imax} - \text{Imin}} \right) \cdot \left(CMAX - CMIN \right) + CMIN$$



histograma



$$j = \frac{14 - 2}{9 - 5}(i - 5) + 2$$

función expansión

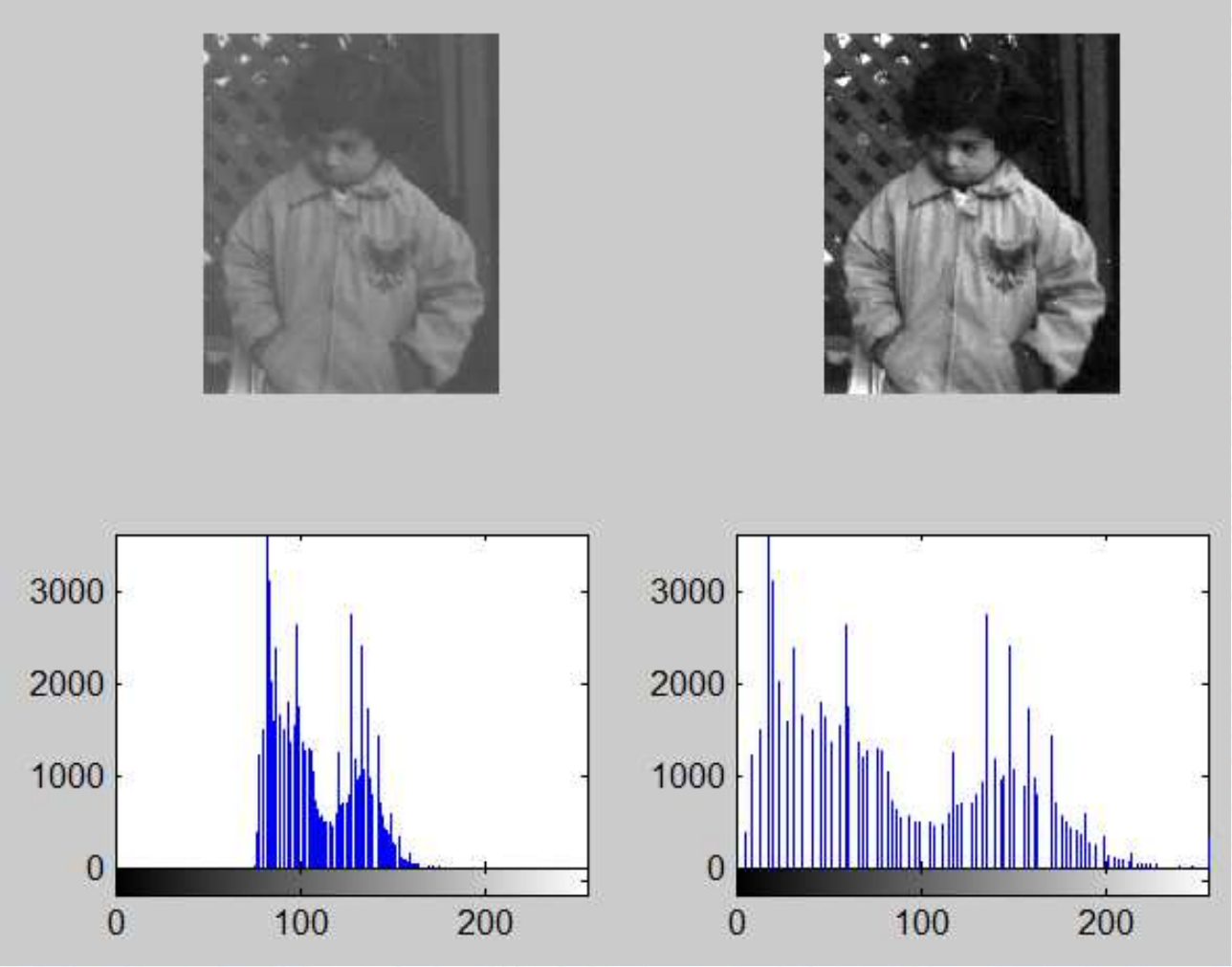
Histogramas

- Modificar el contraste (Expansión):

```
Imin = 75;
Imax = 175;
CMIN = 0;
CMAX = 255;
imagen = imread('pout.tif');
subplot(2,2,1);
imshow(imagen);
subplot(2,2,3);
imhist(imagen), axis tight;
subplot(2,2,2);
imagen_exp = uint8((double(imagen)-Imin)...
/(Imax-Imin)*(CMAX-CMIN)+ CMIN);
imshow(imagen_exp);
subplot(2,2,4);
imhist(imagen_exp), axis tight;
```

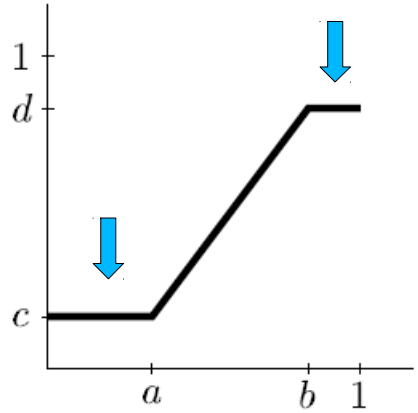
Histogramas

- Modificar el contraste (Expansión):



Histogramas

- Modificar el contraste (MATLAB):
  - imadjust (im, [a,b], [c,d])



¿Qué diferencia hay con la función vista anteriormente?

Ejemplo anterior usando *imadjust*

```
Imin = 75;
Imax = 175;
CMIN = 0;
CMAX = 255;
imagen = imread('pout.tif');
subplot(2,2,1);
imshow(imagen);
subplot(2,2,3);
imhist(imagen), axis tight;
subplot(2,2,2);
imagen_exp=imadjust(imagen,[Imin/255,Imax/255],[CMIN/255,CMAX/255]);
imshow(imagen_exp);
subplot(2,2,4);
imhist(imagen_exp), axis tight;
```

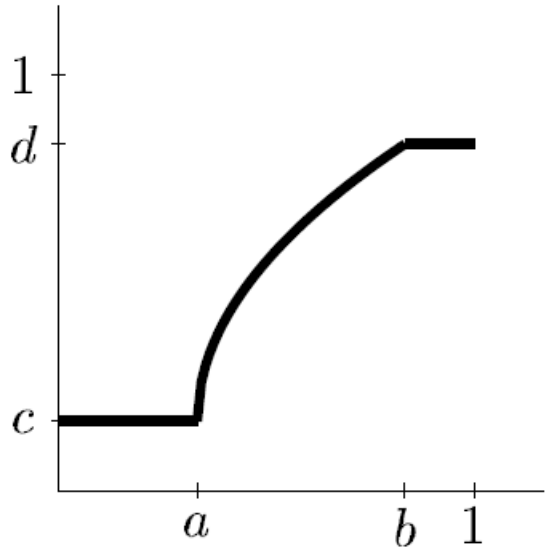
Histogramas

- Modificar el contraste (MATLAB):
  - imadjust (im, [a,b], [c,d], *gamma*)
    - *gamma* describe la forma de la función entre (a,c) y (b,d).
    - Si gamma:
      - = 1, el mapeo es linear (por defecto)
      - < 1, la función es cóncava hacia abajo.
      - > 1, la función es cóncava hacia arriba.

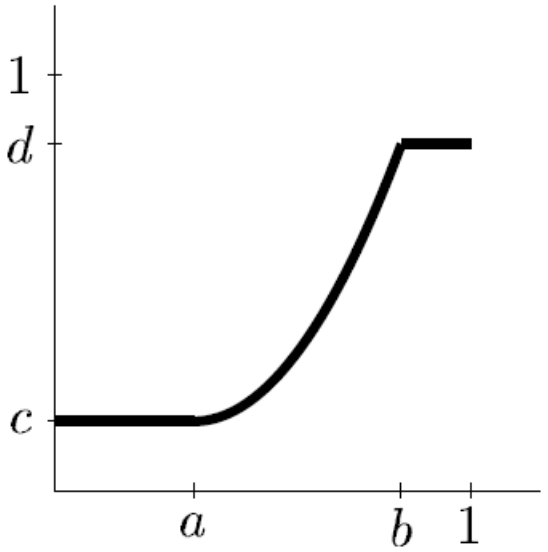
Histogramas

- Modificar el contraste (MATLAB):
  - imadjust (im, [a,b], [c,d], *gamma*)

$$y = \left(\frac{x-a}{b-a}\right)^{\gamma}(d-c)+c.$$



gamma < 1



gamma > 1

Histogramas

- Modificar el contraste (MATLAB):
  - imadjust (im, [a,b], [c,d], *gamma*)

```
Imin = 75;
Imax = 175;
CMIN = 0;
CMAX = 255;
imagen = imread('pout.tif');
subplot(2,3,1);
imshow(imagen);
xlabel('Original');
subplot(2,3,4);
imhist(imagen), axis tight;
subplot(2,3,2);
gamma = 0.5;
imagen_exp=imadjust(imagen,[Imin/255,Imax/255],[CMIN/255,CMAX/255],gamma);
imshow(imagen_exp);
xlabel(['gamma = ',num2str(gamma)]);
subplot(2,3,5);
imhist(imagen_exp), axis tight;
subplot(2,3,3);
gamma = 1.5;
imagen_exp=imadjust(imagen,[Imin/255,Imax/255],[CMIN/255,CMAX/255],gamma);
imshow(imagen_exp);
xlabel(['gamma = ',num2str(gamma)]);
subplot(2,3,6);
imhist(imagen_exp), axis tight;
```

¿Qué diferencia hay entre los histogramas obtenidos?

Operaciones de vecindad

- Operaciones de vecindad:
  - Filtrado:
    - Tipos de filtros (lineales y no lineales)
    - Objetivos:
      - Suavizar.
      - Realzar.
      - Detectar bordes.
  - De-blurring: Movimiento o desenfoque.

Operaciones de vecindad

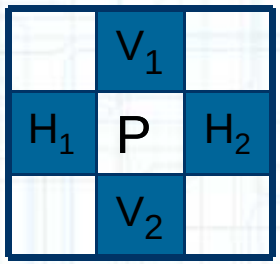
- Vecinos:
  - Cada píxel en la imagen de salida depende de una función que es combinación de los valores de los n-píxeles vecinos.

x_1	x_2	x_3
x_4	P	x_5
x_6	x_7	x_8

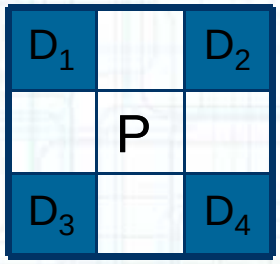
$p'=f(x_1,x_2,\ldots,x_8)$
x2,x4,x5,x7 $N_4(p)$
x1,x3,x6,x8 $N_D(p)$
Todos $N_8(p)$

Operaciones de vecindad

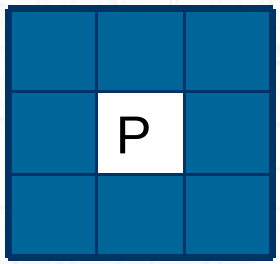
- Vecinos:



N_4: Conectividad 4 (por filas y columnas)
- Vecinos horizontales: H_i (x-1,y) (x+1,y)
- Vecinos verticales: V_i (x,y-1) (x,y+1)



N_{D4}: Conectividad Diagonal 4
- Vecinos Diagonales:
  (x-1,y-1) (x+1, y-1)
  (x-1,y+1) (x+1,y+1)



N_8: Conectividad 8
- Combina N_4 y N_{D4}

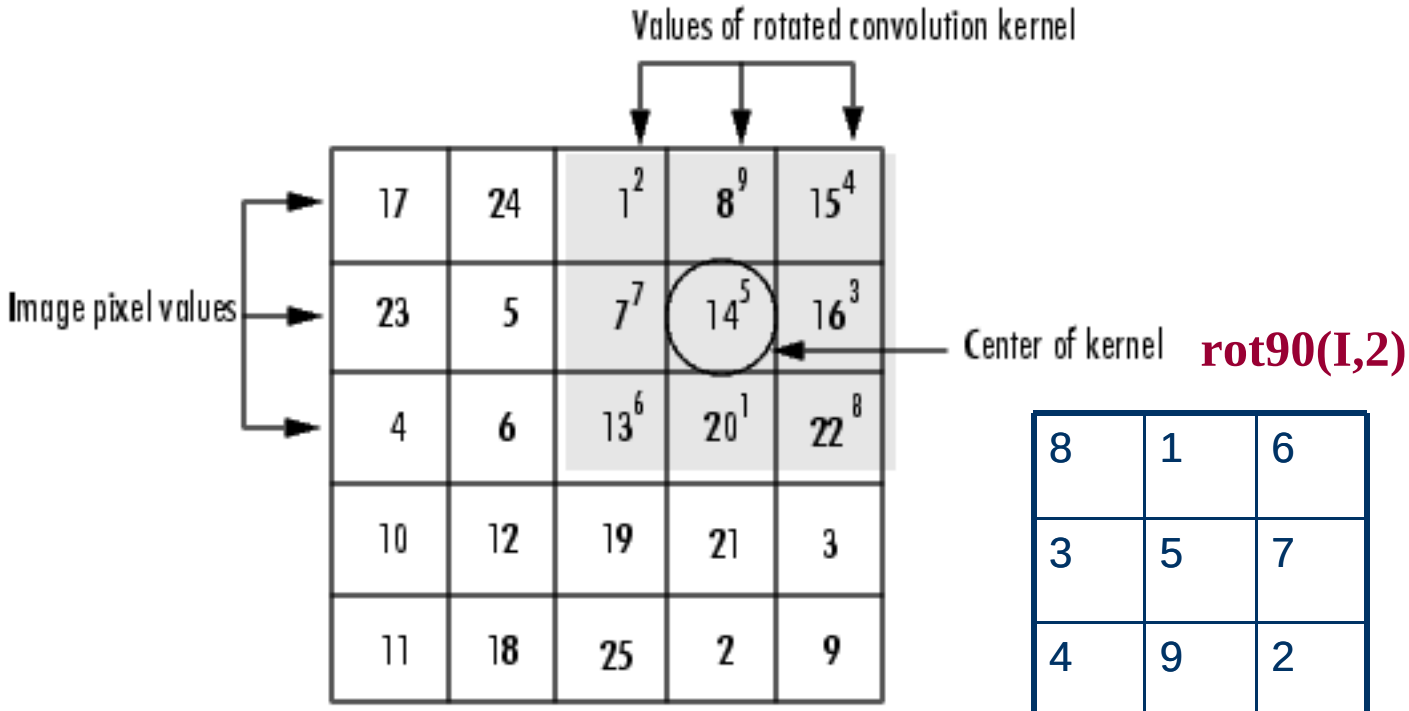
Operaciones de vecindad

- Filtrado:
  - Conceptos:
    - Alta frecuencia: cambios bruscos en los niveles de gris.
    - Baja frecuencia: cambios graduales en la variación de los niveles de gris.
  - Filtros:
    - Paso banda: Filtran las frecuencias de un determinado rango.
    - Paso bajo: suavizan la imagen (filtran las frecuencias bajas y eluden las altas)
    - Paso alto: extraen bordes (filtran las frecuencias altas y eluden las bajas).

Operaciones de vecindad

- Filtrado:
  - Convolución espacial:

$$1 \cdot 2 + 8 \cdot 9 + 15 \cdot 4 + 7 \cdot 7 + 14 \cdot 5 + 16 \cdot 3 + 13 \cdot 6 + 20 \cdot 1 + 22 \cdot 8 = 575$$



Operaciones de vecindad

- Filtrado:
  - Convolución espacial (pasos):
    1) Rotar el filtro 180º
    2) Posicionar el centro del filtro con el pixel a convolucionar.
    3) Multiplicar cada peso del filtro por su correspondiente vecino en la imagen.
    4) Sustituir el valor del pixel por la suma del paso 3.

Operaciones de vecindad

- Filtrado:
  - Convolución espacial (los bordes de la imagen):
    - ¿Qué sucede en los bordes de la imagen donde parte de la máscara está fuera?
    - Soluciones:
      - Se ignoran los bordes:
        - Sólo se aplica dentro de la imagen.
        - La imagen resultado es más pequeña.
        - Pérdida significativa de información para máscaras grandes).
      - Se rellenan con ceros los valores externos de la imagen:
        - La imagen tiene el mismo tamaño que la original.
        - Puede introducir artefactos no deseados alrededor de la imagen.

Operaciones de vecindad

- Filtrado:
  - Convolución espacial (recordatorio):
    - Una convolución en el dominio espacial es lo mismo que una multiplicación en el dominio de la frecuencia.
    - Una multiplicación en el dominio espacial es lo mismo que una convolución en el dominio de la frecuencia.

Operaciones de vecindad

- Filtrado:
  - Convolución espacial (MATLAB):
    - c = conv2(A, h)
    - c = conv2(h1, h2, A) por filas con h1 y por columnas con h2.
    - c = conv2(..., shape) :
      - *'full'* : (por defecto) rellena con 0's los extremos. Aplica la convolución a todos los lugares donde la imagen y máscara intersectan.
      - *'same'*: rellena con 0's los extremos devuelve la parte central de la convolución (del tamaño de A)
      - *'valid'*: [ma-mh+1,na-nh+1] sin rellenar con 0's los extremos.

Operaciones de vecindad

- Filtrado:
  - Convolución espacial (MATLAB):

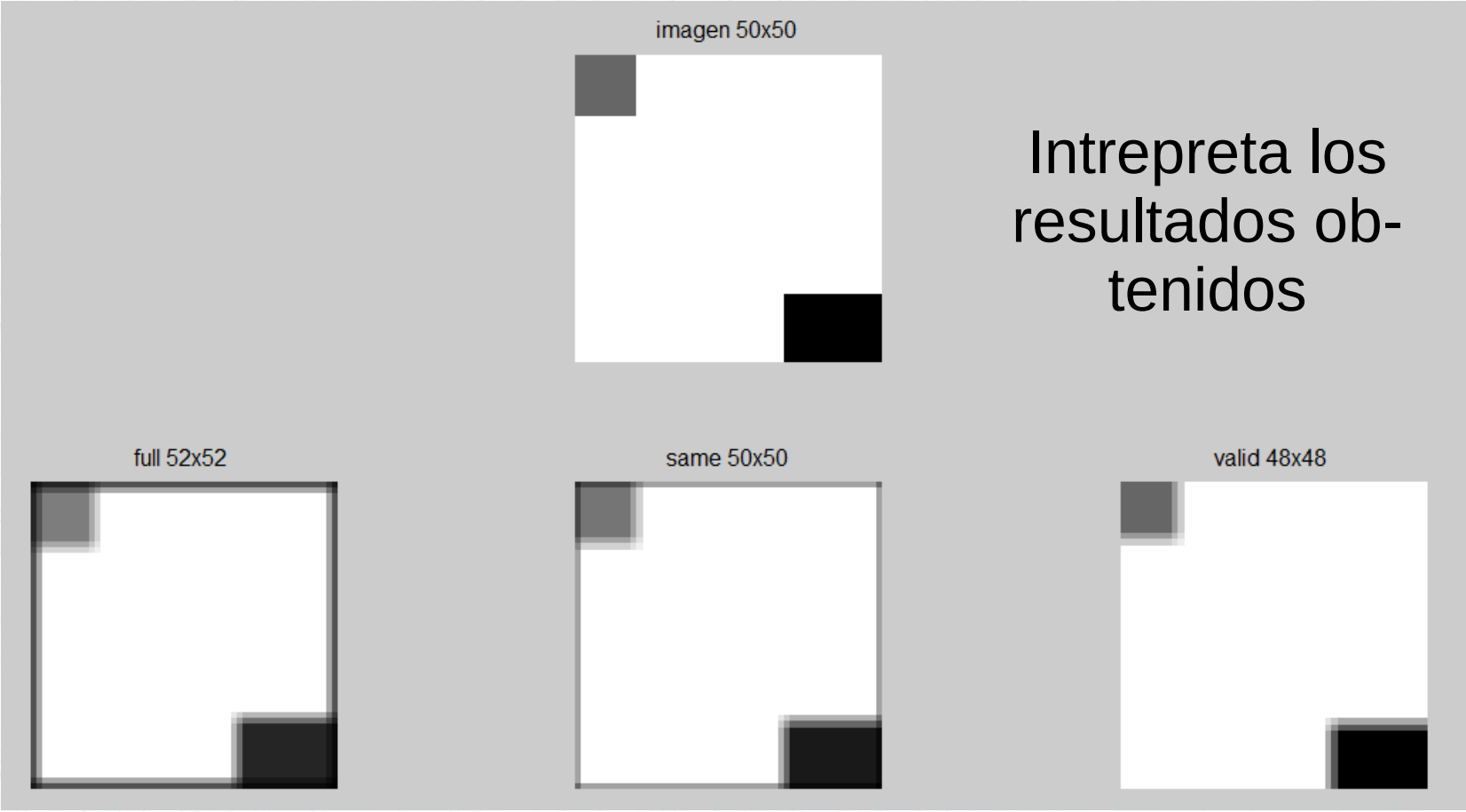
```
A=ones(50,50).*150;
A(1:10,1:10)=75;
A(end-10:end,end-15:end)=25;
subplot(2,3,2);
imshow(A,[]);
title(['imagen ',num2str(size(A,1)),'x',num2str(size(A,2))]);

% Opciones del Filtro Convolución
h=zeros(3,3) + 5./9;

subplot(2,3,4);
b=conv2(A,h,'full');
imshow(b,[]);
title(['full ',num2str(size(b,1)),'x',num2str(size(b,2))]);
subplot(2,3,5);
b=conv2(A,h,'same');
imshow(b,[]);
title(['same ',num2str(size(b,1)),'x',num2str(size(b,2))]);
subplot(2,3,6);
b=conv2(A,h,'valid');
imshow(b,[]);
title(['valid ',num2str(size(b,1)),'x',num2str(size(b,2))]);
```

Operaciones de vecindad

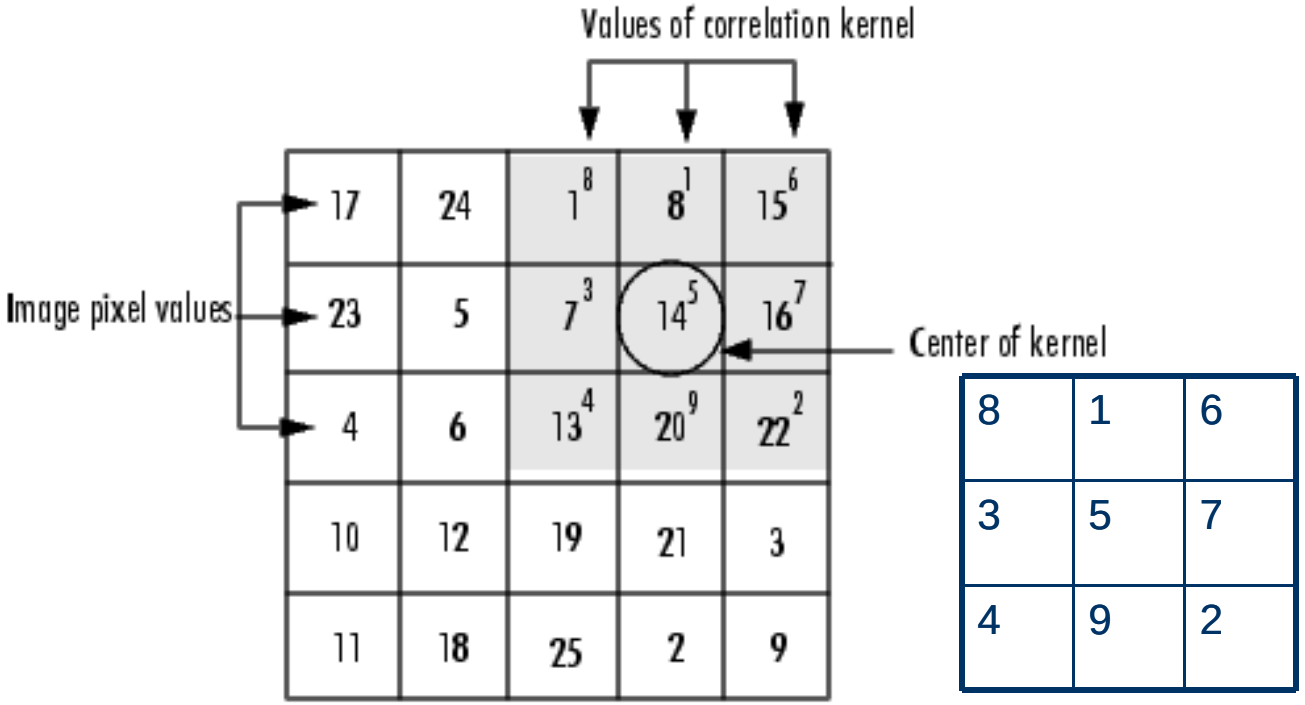
- Filtrado:
  - Convolución espacial (MATLAB):



Operaciones de vecindad

- Filtrado:
  - Correlación:

$$1 \cdot 8 + 8 \cdot 1 + 15 \cdot 6 + 7 \cdot 3 + 14 \cdot 5 + 16 \cdot 7 + 13 \cdot 4 + 20 \cdot 9 + 22 \cdot 2 = 585$$

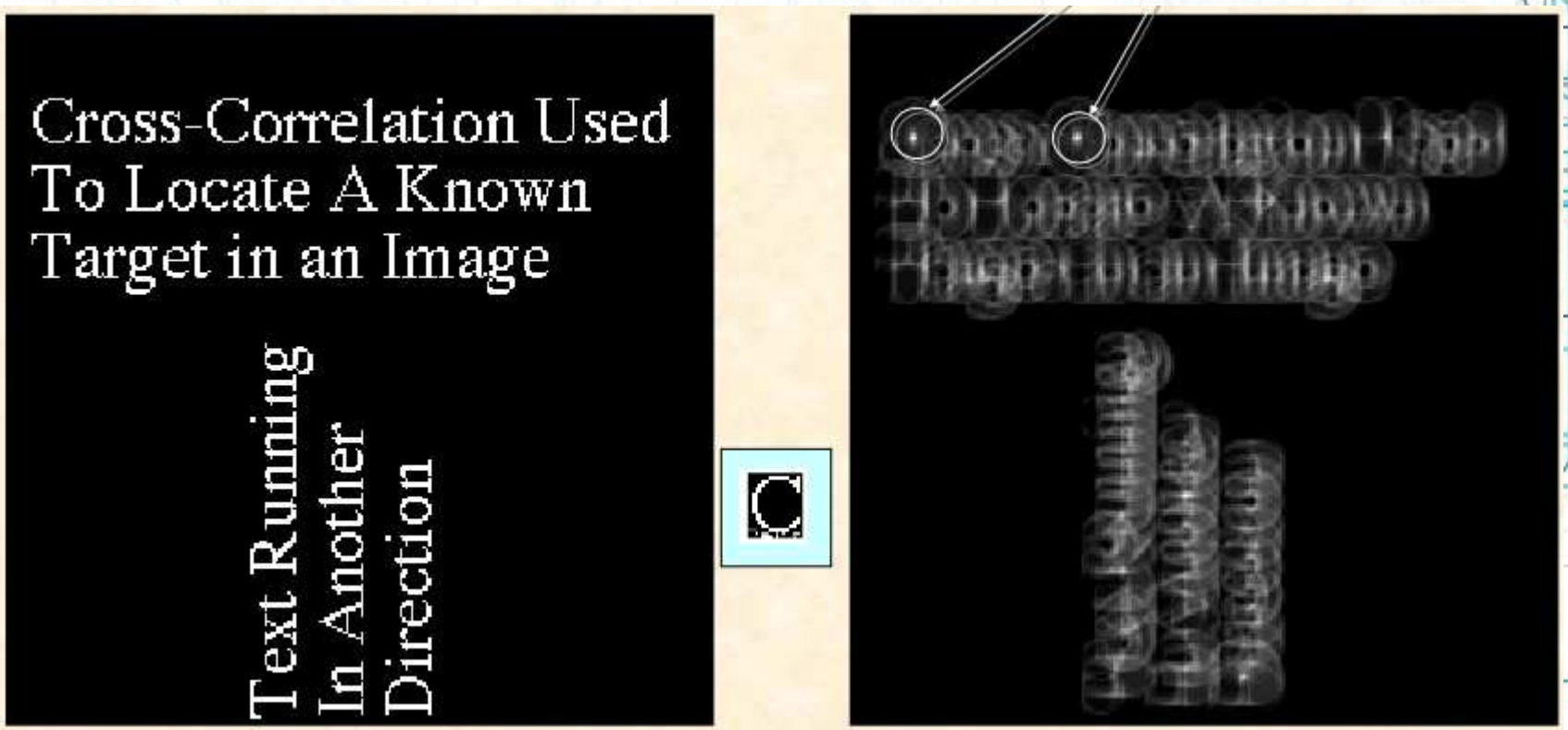


Operaciones de vecindad

- Filtrado:
  - Correlación (pasos):
    1) Posicionar el centro del filtro con el pixel a convolucionar.
    2) Multiplicar cada peso del filtro por su correspondiente vecino en la imagen.
    3) Sustituir el valor del pixel por la suma del paso 3.

Operaciones de vecindad

- Filtrado:
  - Correlación:




```
z = xcorr2(c,imagen);
imshow (z,[]);
```

Operaciones de vecindad

- Filtrado (suavizado):
  - Intentan eliminar el nivel de ruido presente en una imagen.
  - El ruido es información no deseada que contamina la imagen.
  - El ruido se representa como una variable aleatoria que sigue una función de densidad determinada.

Operaciones de vecindad

- Filtrado (suavizado):
  - Tipos de ruido:
    - Gaussiano: La degradación de la señal durante el proceso de captación o transmisión.
    - Impulsional (sal y pimienta): Saturación de la carga que recibe un pixel. Toma el valor máximo (sal) o el mínimo (pimienta).
    - Multiplicativo: Falta de iluminación uniforme sobre la escena capturada. La señal obtenida es fruto de la multiplicación de dos imágenes (el efecto de la iluminación y la función de reflactancia de la escena).

Operaciones de vecindad

- Filtrado (suavizado):
  - Función para modelar el ruido en MATLAB:
    - imnoise(imagen,tipo_ruido,parámetros)
    - tipo_ruido:
      - 'gaussian'
      - 'salt & pepper'
      - 'speckle'

Operaciones de vecindad

- Filtrado (suavizado):
  - Función para modelar el ruido en MATLAB:

```
subplot(2,3,2);
A=imread('woodstatue.jpg');
imshow(A,[]);
title('Original');
subplot(2,3,4);
B=imnoise(A,'gaussian',0,0.01);
imshow(B,[]);
title('Gaussiano')
subplot(2,3,5);
B=imnoise(A,'salt & pepper',0.01);
imshow(B,[]);
title('Sal y Pimienta')
subplot(2,3,6);
B=imnoise(A,'speckle',0.01);
imshow(B,[]);
title('Multiplicativo');
```

Operaciones de vecindad

- Filtrado (suavizado):
  - Reducción del ruido:
    - Filtros lineales (convolución):
      - Media
      - Gausiano
    - Filtros no lineales:
      - Mediana
      - Máximo y mínimo
    - Filtros adaptativos:
      - Wiener
    - Filtros temporales:
      - Promedio de varias imágenes de la misma escena tomadas en instantes diferentes de tiempo.

Operaciones de vecindad

- Filtrado (suavizado):
  - Filtro de la media:
    - Genera una nueva imagen cuya intensidad para cada píxel se obtiene promediando los valores de intensidad de los píxeles vecinos.
    - Un mayor tamaño de la máscara supone una mayor reducción del ruido, pero también mayor difuminado de los bordes.

Operaciones de vecindad

- Filtrado (suavizado):
  - Filtro de la media:

```
A=imread('cell.tif');
subplot(1,3,1);
C=imnoise(A);
imshow(C,[]);
title ('imagen con ruido');

subplot(1,3,2);
h=fspecial('average'); % h = ones(3,3).*1/9;
% I1=conv2(double(C), h,'full');
I1 = imfilter(C,h);
imshow(I1,[]);
title('filtro media 3x3');

subplot(1,3,3);
h =fspecial('average',[7 7]); % h=ones(7,7).*1/49;
% I1=conv2(double(C), h,'full');
I1 = imfilter(C,h);
imshow(I1,[]);
title('filtro media 7x7');
```

Operaciones de vecindad

- Filtrado (suavizado):
  - Filtro de la media:
    - Problemas:
      - Presenta lóbulos en su respuesta en frecuencia.
      - No es de fase lineal.

Operaciones de vecindad

- Filtrado (suavizado):
  - Filtro basado en la función de densidad binomial discreta:
    - Soluciona los problemas del filtro de la media.
    - Sus coeficientes son obtenidos mediante el triángulo de Pascal:

n	f	máscara	σ^2
0	1	1	0
1	1/2	1 1	1/4
2	1/4	1 2 1	1/2
3	1/8	1 3 3 1	3/4
4	1/16	1 4 6 4 1	1

n (grado del filtro), f (factor de escala) y σ^2 la varianza efectiva de la máscara

Operaciones de vecindad

- Filtrado (suavizado):
  - Filtro basado en la función de densidad binomial discreta:

```
A=imread('cell.tif');
subplot(1,3,1);
C=imnoise(A);
imshow(C,[]);
title ('imagen con ruido');

subplot(1,3,2);
h=fspecial('average');
I1 = imfilter(C,h);
imshow(I1,[]);
title('filtro media 3x3');

subplot(1,3,3);
h =conv2([1 2 1],[1 2 1]')/16;
I1 = imfilter(C,h);
imshow(I1,[]);
title('filtro binomial 3x3');
```

Se construyen mediante la convolución discreta de un filtro binomial unidimensional y su traspuesta.
Se pondera la influencia de cada vecino en función de su distancia.

Operaciones de vecindad

- Filtrado (suavizado):
  - Filtro Gaussiano:
    - Devuelve un filtro gaussiano con media cero y desviación estándar σ.
    - La desv. estándar controla el grado de suavizado (mayor varianza, se tienen en cuenta vecinos más alejados).
    - Disminuye la nitidez.
    - Aumenta la borrosidad.
    - Se producen pérdida de detalles.

Operaciones de vecindad

- Filtrado (suavizado):
  - Filtro Gaussiano:

```
subplot(2,2,1);
A = imread('cameraman.tif');
imshow(A,[]);
title('Original');
subplot(2,2,2);
B = imnoise(A);
imshow(B,[]);
title('Con ruido');
subplot(2,2,3);
h1=fspecial('gaussian');
C = conv2(double(B),h1);
imshow(C,[]);
title('Filtro gaussiano');
subplot(2,2,4);
h2=fspecial('average');
D = conv2(double(B),h2);
imshow(D,[]);
title('Filtro media');
```

Operaciones de vecindad

- Filtrado (suavizado):
  - Filtro de la mediana:
    - Genera una nueva imagen cuya intensidad para cada píxel se obtiene como la mediana de los valores de intensidad de los píxeles vecinos.
    - Elimina los efectos espurios y preserva los bordes.
    - Pierde detalles finos como líneas aisladas, puntos aislados y redondea esquinas.
    - Elevado coste computacional.

Operaciones de vecindad

- Filtrado (suavizado):
  - Filtro de la mediana:

```
A = imread('cameraman.tif');
subplot(2,3,1);
B = imnoise(A,'gaussian');
imshow(B,[]);
title('Con ruido gaussiano');
subplot(2,3,4);
C = imnoise(A,'salt & pepper',0.1);
imshow(C,[]);
title('Con ruido impulsional');
h1=fspecial('average');
subplot(2,3,2);
D = conv2(double(B),h1);
imshow(D,[]);
title('Filtro media');
subplot(2,3,5);
D = conv2(double(C),h1);
imshow(D,[]);
title('Filtro media');
subplot(2,3,3);
D = nlfilter(B,[3 3],'median(x(:))'); %D=medfilt2(B);
imshow(D,[]);
title('Filtro mediana');
subplot(2,3,6);
D = nlfilter(C,[3 3],'median(x(:))'); %D=medfilt2(C);
imshow(D,[]);
title('Filtro mediana');
```

Operaciones de vecindad

- Filtrado (suavizado):
  - Filtro del máximo:
    - Sólo elimina ruido pimienta.
    - Tiende a aclarar la imagen.
    - nlfilter(imagen,dimension,'max(x(:))'
    - ordfilt2(imagen,dimensión,ones(tam. ventana))
  - Filtro del mínimo:
    - Sólo elimina el ruido tipo sal.
    - Tiende a oscurecer la imagen.
    - nlfilter(imagen,dimension,'min(x(:))'
    - ordfilt2(imagen,1,ones(tam. ventana))

Operaciones de vecindad

- Filtrado (suavizado):
  - Filtros del máximo y del mínimo:

```
C=imread('cell.tif');
subplot(1,3,1);
imshow(C,[]);
title ('imagen con ruido impulsional');

subplot(1,3,2);
I1 = ordfilt2(C,9,ones(3,3));
imshow(I1,[]);
title('filtro máximo 3x3');

subplot(1,3,3);
I1 = ordfilt2(C,1,ones(3,3));
imshow(I1,[]);
title('filtro mínimo 3x3');
```

Operaciones de vecindad

- Filtrado (suavizado):
  - Filtro de Wiener:
    - Es un tipo de filtro lineal que se adapta a la varianza local de la imagen.
    - Produce mejores resultados que el filtrado lineal (preserva zonas de alta frecuencia), pero requiere más tiempo de computación.
    - No hay que diseñar el filtro antes de aplicarlo.

Operaciones de vecindad

- Filtrado (suavizado):
  - Filtro de Wiener:

```
subplot(2,2,1);
A = imread('cameraman.tif');
imshow(A,[]);
title('Original');
subplot(2,2,2);
B = imnoise(A,'gaussian');
imshow(B,[]);
title('Con ruido gaussiano');
subplot(2,2,3);
h1=fspecial('gaussian',[5 5]);
C = conv2(double(B),h1);
imshow(C,[]);
title('Filtro gaussiano');
subplot(2,2,4);
D = wiener2(double(B),[5 5]);
imshow(D,[]);
title('Filtro de Wiener');
```

Operaciones de vecindad

- Filtrado (Realzado):
  - El uso de los anteriores filtros de eliminación de ruido (paso bajo) puede afectar a la nitidez.
  - El realce de bordes enfatiza o resalta aquellos píxeles que tienen un valor de gris diferente al de sus vecinos.
  - El realce aumenta la ganancia de altas frecuencias (filtro paso alto).
  - Si la imagen contiene ruido, su efecto se multiplica, por lo que es conveniente previamente eliminar el ruido.

Operaciones de vecindad

- Filtrado (realzado):
  - De forma general, la máscara utilizada para realzar los bordes es:

-1	-1	-1
-1	A	-1
-1	-1	-1

A = 9 * Ganancia – 1 y todo multi-plicado por 1/9

Operaciones de vecindad

- Filtrado (realzado):

```
ganancia = 2.2;
A = imread('lena256.jpg');
A = rgb2gray(A);
h = fspecial('gaussian',[3 3],10);
A = uint8(conv2(double(A),h));
subplot(1,2,1);
imshow(A,[]);
title('Original');
subplot(1,2,2);
h = ones(3,3)*(-1);
h(2,2) = 9 * ganancia - 1;
h = 1/9 * h;
B = uint8(conv2(double(A),h));
imshow(B,[]);
title('Realzada');
```

Operaciones de vecindad

- Filtrado (realzado):

```
A = imread('lena256.jpg');
A = rgb2gray(A);
h = fspecial('gaussian',[3 3],10);
A = uint8(conv2(double(A),h));
subplot(1,2,1);
imshow(A,[]);
title('Original');
subplot(1,2,2);
h = fspecial('unsharp');
B = uint8(conv2(double(A),h));
imshow(B);
title('Realzada');
```

Operaciones de vecindad

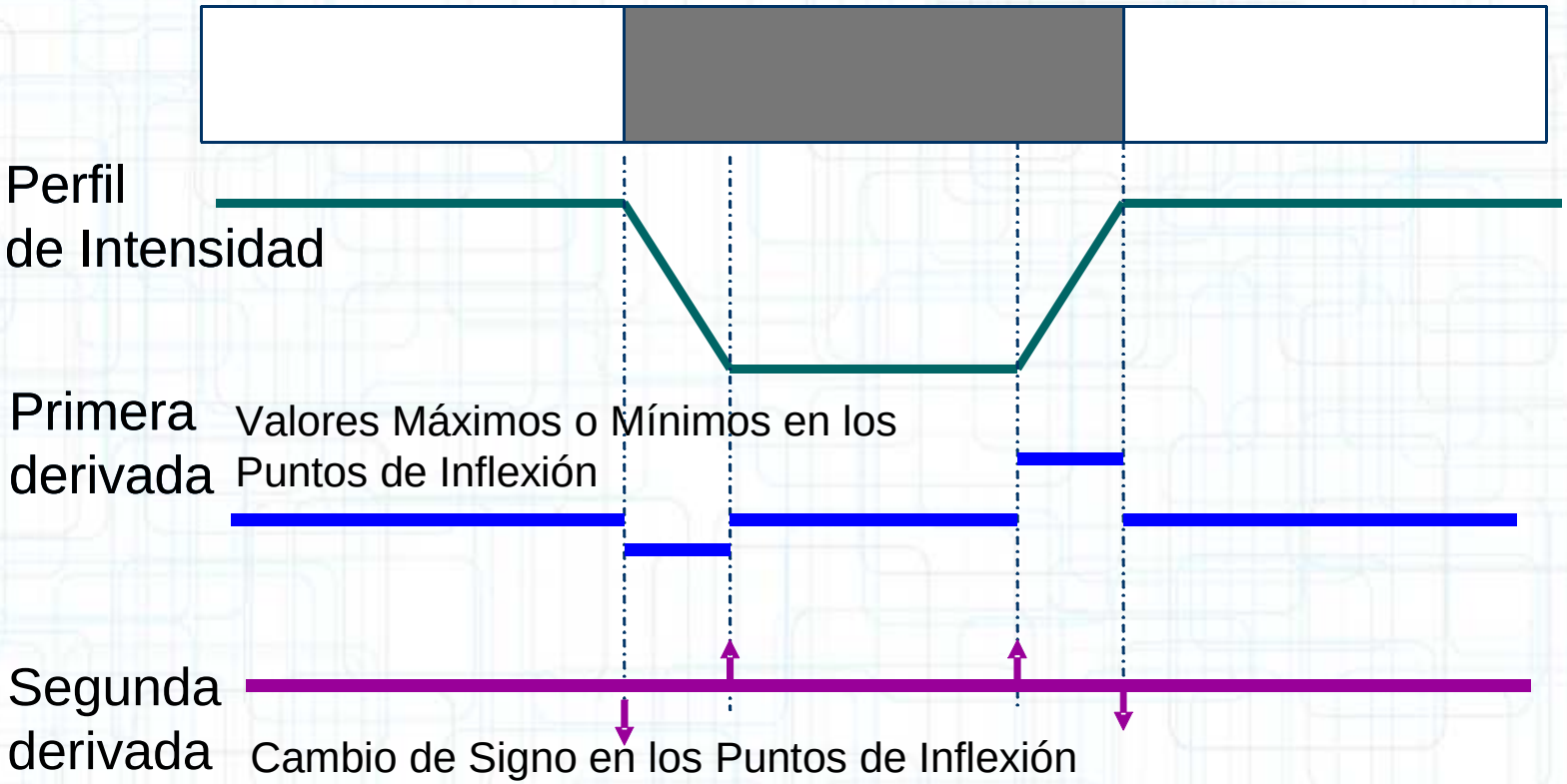
- Filtrado (detección de bordes):
  - Bordes son puntos en la imagen alrededor de los cuales la imagen presenta una brusca variación en los niveles de grises (contornos de los objetos sólidos, marcas en las superficies, sombras, ...)
  - Objetivo: localizar los bordes en la imagen evitando el posible ruido subyacente.
  - Métodos:
    - Operadores basados en la primera derivada
    - Operadores basados en la segunda derivada

Operaciones de vecindad

- Filtrado (detección de bordes):
  - Bordes: transición de oscuro a claro o viceversa
  - Se modelan como una rampa en vez de como un cambio brusco de intensidad
  - En la imagen original suelen estar desdibujados debido al muestreo sufrido.

Operaciones de vecindad

- Filtrado (detección de bordes):



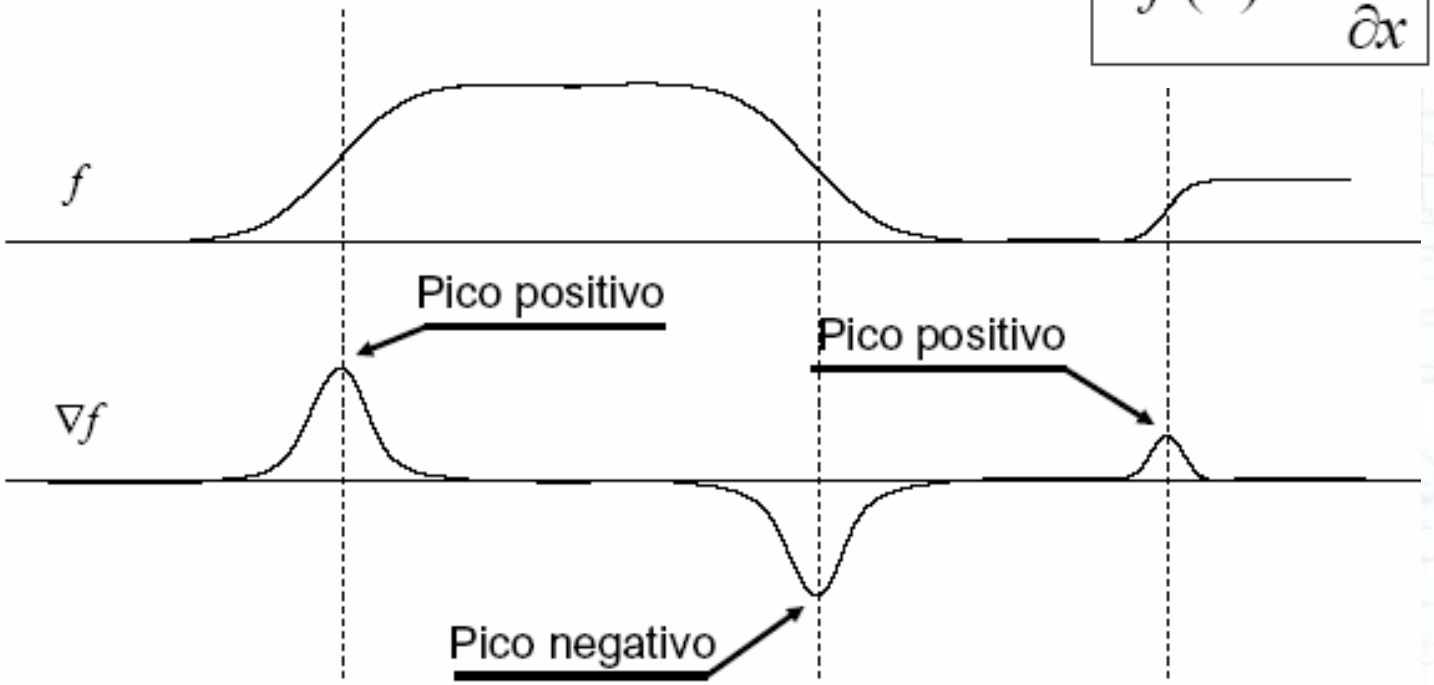
Operaciones de vecindad

- Filtrado (detección de bordes):
  - Basados en la Primera Derivada: Operadores Gradiente:
    - Sobel, Prewitt, Roberts, etc.
    - Operador DrogG
  - Basados en la Segunda Derivada:
    - Operador Laplaciana
    - Operador LoG
  - Método de Canny

Operadores de vecindad

- Filtrado (detección de bordes):
  - El gradiente mide un cambio local.

$$\nabla f(x) = \frac{\partial f}{\partial x}$$

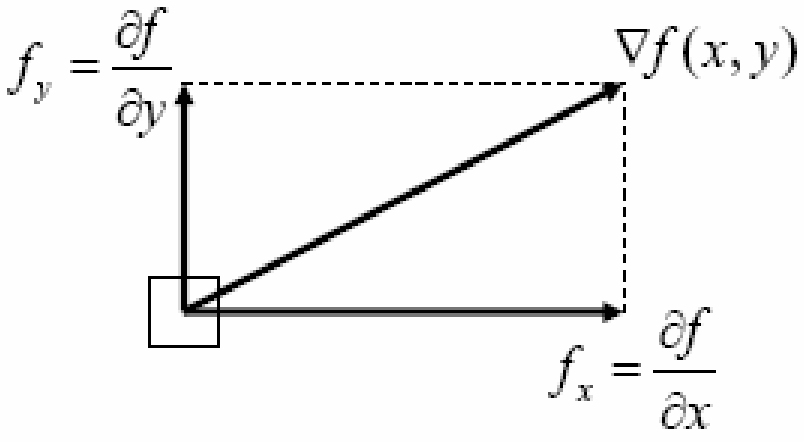


Operaciones de vecindad

- Filtrado (detección de bordes):
  - El gradiente es la primera derivada de *f*.

$$\nabla f(x,y) = \frac{\partial f}{\partial x}\vec{i} + \frac{\partial f}{\partial y}\vec{j}$$

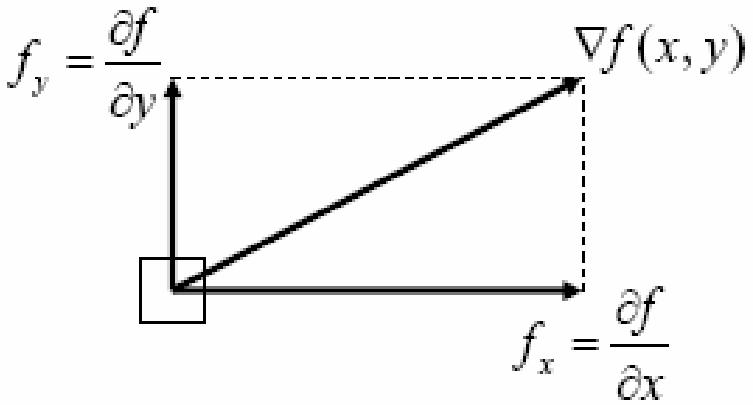
  - Es un vector de dos componentes: (*fx*,*fy*)



Operaciones de vecindad

- Filtrado (detección de bordes):
  - La derivada de una función bidimensional f(x,y) es un vector que apunta a la máxima variación de f(x,y) y cuyo módulo es proporcional a la variación.

$$\nabla f(x,y) = \begin{bmatrix} \dfrac{\partial f(x,y)}{x} \\ \dfrac{\partial f(x,y)}{y} \end{bmatrix} = \begin{bmatrix} f_x(x,y) \\ f_y(x,y) \end{bmatrix}$$



Módulo del Gradiente

Dirección del Gradiente

$$G = |\nabla f(x,y)| = \sqrt{f_x^2 + f_y^2}$$

$$\alpha(x,y) = \operatorname{atan}\left(\frac{f_y}{f_x}\right)$$

Operaciones de vecindad

- Filtrado (detección de bordes):
  - Para el caso bidimensional discreto las aproximaciones del Operador Gradiente se basan en diferencias entre los niveles de gris de la imagen.

En la dirección x

$$f_x(x,y) = G_x(i,j) = \frac{f(i,j) - f(i,j-1)}{1}$$

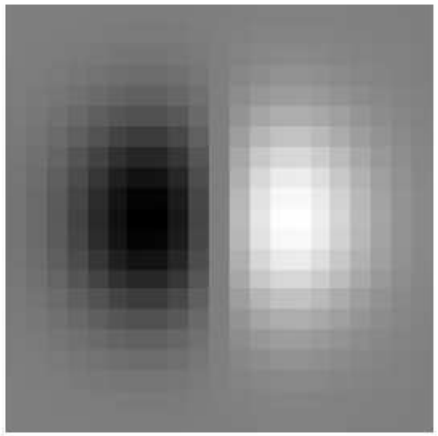
$$f_x(x,y) = G_x(i,j) = \frac{f(i,j+1) - f(i,j-1)}{2}$$

Operaciones de vecindad

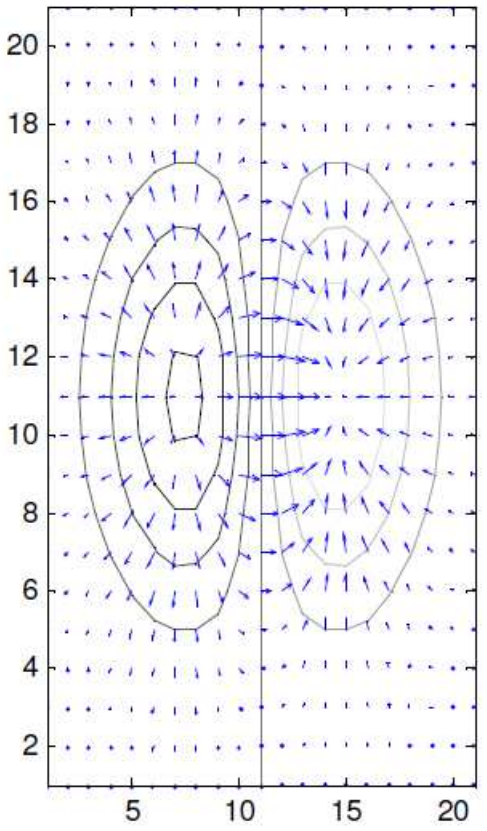
- Filtrado (detección de bordes):
  - Se considera que existe un borde si la magnitud del gradiente supera cierto umbral.

$$g(x,y) = \begin{cases} 255 \; si \; G > umbral \\ 0 \quad si \; G \leq umbral \end{cases}$$

Imagen sintética



Operador gradiente en cada píxel



El umbral suele ser el 70%-80% del máximo valor detectado en la imagen.

Operaciones de vecindad

- Filtrado (detección de bordes):
  - El gradiente fila y columna se puede obtener mediante una convolución de la imagen con las respuestas impulsionales del gradiente fila o columna.

$$G_F(i,j) = f(i,j) \otimes H_F(i,j)$$

$$G_C(i,j) = f(i,j) \otimes H_C(i,j)$$

$$|G(i.j)| = \sqrt{G_F^2 + G_C^2}$$

Operaciones de vecindad

- Filtrado (detección de bordes): Operadores de Prewitt y Sobel.

Realzado		Vertical				Horizontal		
Prewitt	1/3	-1	0	1	1/3	1	1	1
		-1	0	1		0	0	0
		-1	0	1		-1	-1	-1
Sobel	1/4	1	0	-1	1/4	-1	-2	-1
		2	0	-2		0	0	0
		1	0	-1		1	2	1
		Columna				Fila		

Es muy sensible al ruido.
Detecta mejor los bordes verticales

Convolución del operador derivada con un filtro binomial (menos sensible al ruido).
Detecta mejor los bordes diagonales.

Operaciones de vecindad

- Filtrado (detección de bordes): Operadores de Prewitt y Sobel.

```
imagen= imread('cameraman.tif');
figure,imshow(imagen),title('Original');
imagen1 = imfilter(imagen,fspecial('prewitt'),'conv','circular');
figure,imshow(imagen1),title('Prewitt');;
imagen2 = imfilter (imagen, fspecial('sobel'),'conv','circular');
figure,imshow(imagen2),title('Sobel');
```

Prewitt

Sobel

Operaciones de vecindad

- Filtrado (detección de bordes): Operador de Roberts:
  - Obtener respuesta ante bordes diagonales a partir de la diferencia entre pares diagonales de píxeles.
  - Las máscaras representan las derivadas en las 2 direcciones diagonales perpendiculares entre sí.
  - Muy sensible al ruido

0	0	0
0	0	1
0	-1	0

-1	0	0
0	1	0
0	0	0

Operadores basados en el gradiente

- Filtrado (detección de bordes): Operador de Roberts:

```
imagen= imread('cameraman.tif');
% Aplicar un filtro de suavizado

% Realzado con Roberts
hf=[0 0 0; 0 0 1; 0 -1 0];
hc=[-1 0 0; 0 1 0; 0 0 0];
Gx= imfilter(double(imagen),hf,'conv');   % Gradiente fila
Gy= imfilter(double(imagen),hc,'conv');   % Gradiente columna
G= sqrt(Gx.*Gx+Gy.*Gy);                   % magnitud

% Fijar un valor de umbral
I=G>umbral;
figure, imshow(I,[])
```

Operaciones de vecindad

- Filtrado (detección de bordes):
  - Problema: Sensibilidad al ruido $\Rightarrow$ Es necesario hacer primero un suavizado.

$$G_S(i,j) = G(i,j) \otimes H(i,j)$$

Respuesta impulso filtro paso-bajo

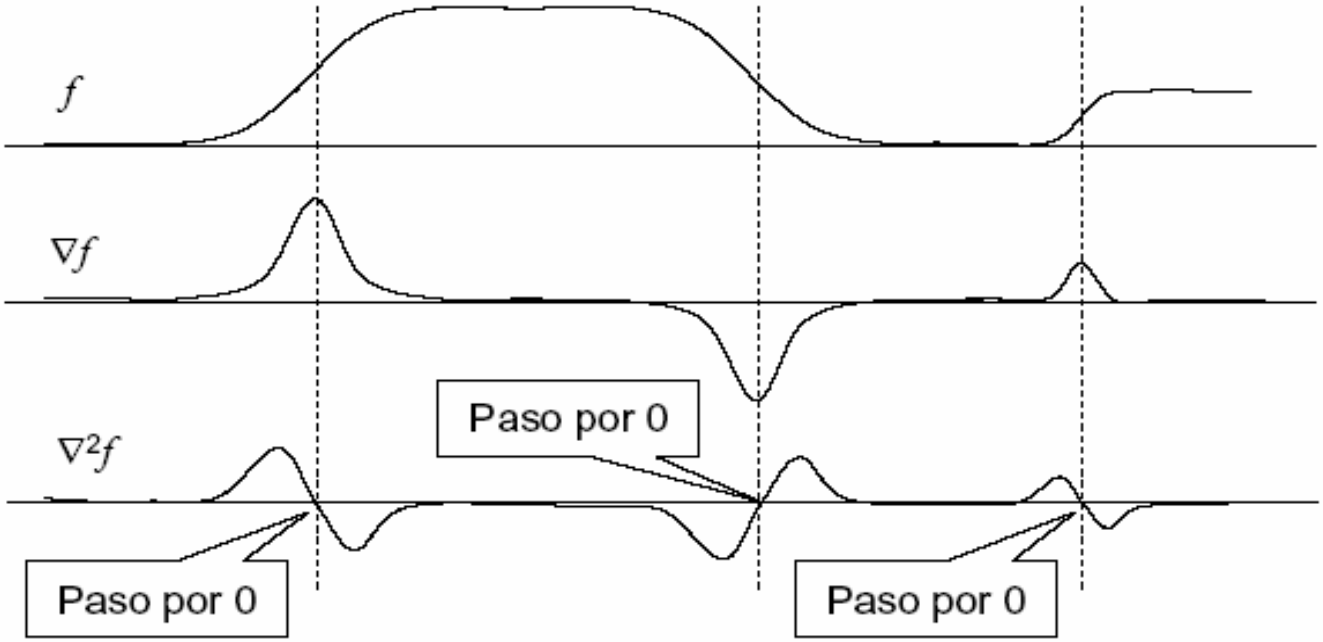
$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & -1 & -1 \\ 2 & 2 & 0 & -2 & -2 \\ 3 & 3 & 0 & -3 & -3 \\ 2 & 2 & 0 & -2 & -2 \\ 1 & 1 & 0 & -1 & -1 \end{bmatrix}$$

Un operador de este tipo muy eficaz es el DroG o Derivada de la gaussiana, que permite mediante el parámetro de la desviación típica controlar el suavizado

Operaciones de vecindad

- Filtrado (detección de bordes):
  - Laplaciana: mide cambios en el gradiente
    - ≅ Diferencia de gradiente

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



Operaciones de vecindad

- Filtrado (detección de bordes):
  - Si la imagen presenta un cambio de intensidades en una dirección, existirá un máximo en la primera derivada y un paso por cero en la segunda.
  - En un dominio continuo, la Laplaciana del borde bidimensional se define como:

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

Operaciones de vecindad

- Filtrado (detección de bordes):
  - La aproximación discreta (Laplaciana):

$$f_{xx}(x,y) = G_F(i+1,j) - G_F(i,j) = f(i+1,j) - 2f(i,j) + f(i-1,j)$$
$$f_{yy}(x,y) = G_C(i,j+1) - G_C(i,j) = f(i,j+1) - 2f(i,j) + f(i,j-1)$$

$$L[f(x,y)] = f(x,y) \otimes H(i,j)$$

$$H(i,j) = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & -1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Laplaciana de 4 vecinos. Normalizamos multiplicando por 1/4

Operaciones de vecindad

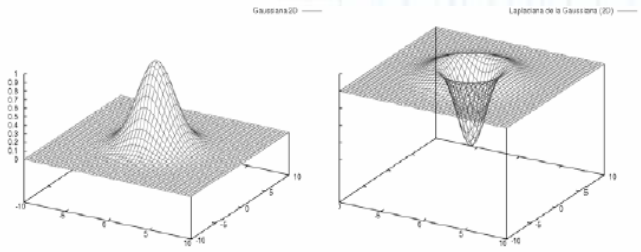
- Filtrado (detección de bordes):
  - Propiedades de la Laplaciana
    - Capacidad de localizar bordes mediante la determinación del Cruce por 0
    - Muy sensible al ruido por lo que puede realizar una pobre detección de bordes (Suavizar la imagen antes)
    - Proporciona bordes de espesor la unidad

Operaciones de vecindad

- Filtrado (detección de bordes):
  - MÁS EFICIENTE: Si aplicamos una Laplaciana suavizada con un filtro gausiano, obtenemos el operador LoG
  - LoG: Laplaciana de una gausiana

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\nabla^2 G = \left(\frac{x^2+y^2-2\sigma^2}{\sigma^4} \right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$



Gaussiana 2D LoG 2D

Operaciones de vecindad

- Filtrado (detección de bordes):

```
imagen= imread('cameraman.tif');
figure,imshow(imagen),title('Original');pause;
imagen1 = imfilter(imagen,fspecial('prewitt'),'conv','circular');
figure,imshow(imagen1),title('Prewitt');pause;
imagen2 = imfilter (imagen, fspecial('sobel'),'conv','circular');
figure,imshow(imagen2),title('Sobel');pause;
imagen3 = imfilter (imagen, fspecial('laplacian'),'conv','circular');
figure,imshow(imagen3),title('Laplaciana');pause;
imagen4 = imfilter (imagen, fspecial('log'),'conv','circular');
figure,imshow(imagen4),title('LoG');
```

Prewitt Sobel Laplaciana LoG

Operaciones de vecindad

- Filtrado (detección de bordes):
  - Método de Canny:
    - Muy utilizado en la localización de contornos.
    - Evita la ruptura de los bordes de los objetos.

Operaciones de vecindad

- Filtrado (detección de bordes):
  - Método de Canny:
    - Pasos:
      - Obtención del módulo y fase del gradiente de la imagen suavizada mediante DroG.
      - En la dirección del gradiente, eliminar puntos que no sean máximos locales, lo que equivale a encontrar el paso por 0 en el operador LoG.
      - Se emplean dos umbrales para la detección de los bordes. Uno para localizar las semillas de los bordes y otro (más bajo) para seleccionar los píxeles a añadir a las semillas.

Operaciones de vecindad

- Filtrado (detección de bordes):
  - Para buscar bordes en un mapa de intensidades:
    - edge(imagen,método [,parámetros])
    - Métodos:
      - ‘sobel’
      - ‘prewitt’
      - ‘roberts’
      - ‘zerocross’
      - ‘canny’
      - ‘log’

Detección de bordes con MATLAB

- Filtrado (detección de bordes):

```
imagen= imread('cameraman.tif');
im1=edge(imagen, 'prewitt');
im2=edge(imagen, 'sobel');
im3=edge(imagen, 'log');
im4=edge(imagen, 'canny');
subplot(2,2,1);
imshow(im1);title('Prewitt');
subplot(2,2,2);
imshow(im2);title('Sobel');
subplot(2,2,3);
imshow(im3);title('LoG');
subplot(2,2,4);
imshow(im4);title('Canny');
```

¿Cómo son los contornos obtenidos con cada operador?

Operaciones de vecindad

- Deblurring:
  - Al capturar una imagen, puede producirse una degradación de la misma debido a varios factores:
    - Movimiento de la cámara
    - Enfoque erróneo, turbulencias atmosféricas, poca exposición, …
  - Una imagen degradada (*blurred*) puede ser descrita por la ecuación:
    - Imgb = operador de distorsión ⊗ imagen + ruido

Operaciones de vecindad

- Deblurring:
  - La tarea fundamental del filtro de *deblurring* será la deconvolución de una imagen degradada por el operador de distorsión.
  - La calidad de la imagen recuperada dependerá del conocimiento del operador de distorsión.

Operaciones de vecindad

- Deblurring:

```
subplot(1,3,1);
imagen = imread('cameraman.tif');
imshow(imagen,[]);
title('Original');
subplot(1,3,2);
% Simular el una imagen degradada por una cámara en movimiento
longitud = 21; % en píxeles
angulo = 11;   % en grados
% Proceso de distorcionamiento lineal
PSF = fspecial('motion',longitud,angulo);
im_degradada = conv2(double(imagen),PSF);
%im_degradada = imfilter(double(imagen),PSF,'conv','circular');
imshow(uint8(im_degradada),[]);
title('Imagen degradada');
% Restaurar la imagen degradada
subplot(1,3,3);
im_restaurada = deconvwnr(im_degradada,PSF,0);
imshow(uint8(im_restaurada),[]);
title('Imagen restaurada');
```

http://www.mathworks.es/es/help/images/examples/deblurring-images-using-a-wiener-filter.html

Operaciones de vecindad

- Deblurring:
  - ¿Qué dice el siguiente cartel?

El ruido es 0

  - ¿Qué aparece en la parte inferior de la matrícula?

Transformada de Fourier

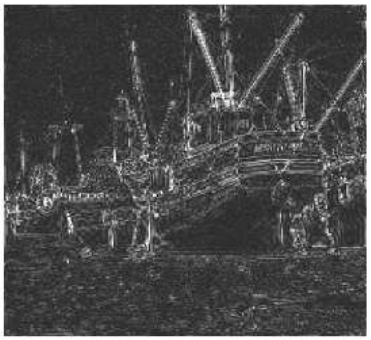
- La transformada de Fourier da información del espectro de altas y bajas frecuencias que forma una imagen:
  - Altas frecuencias espaciales corresponden a bordes abruptos en la imagen.
  - Bajas frecuencias espaciales corresponden a la ausencia de bordes (regiones de nivel de intensidad aproximadamente uniforme).

Transformada de Fourier

- Filtro paso alto ideal:

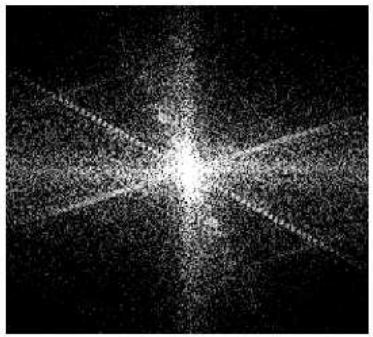
$$H(u,v)=\begin{cases}0 \ si \ D(u,v)\le D_0\\ 1 \ si \ D(u,v)>D_0\end{cases}$$

$$D(u,v)=\sqrt{(u^2+v^2)}$$

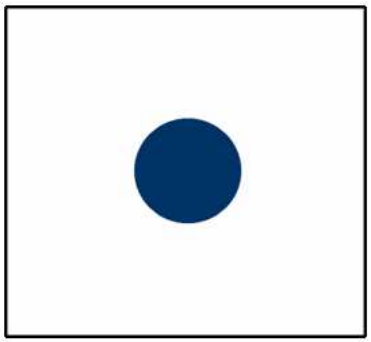


FFT

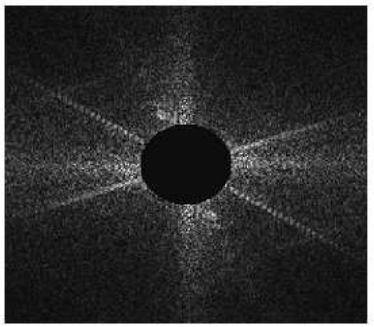
IFFT



×



=



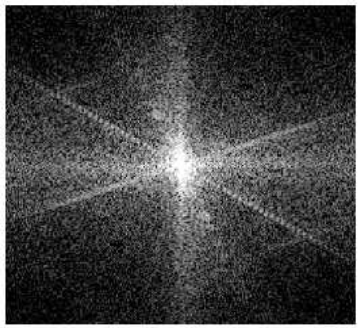
Transformada de Fourier

- Filtro paso bajo ideal:

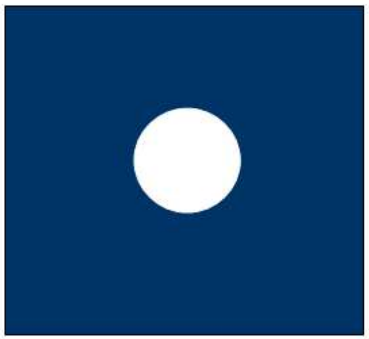
$$H(u,v)=\begin{cases}1 \ si \ D(u,v)\le D_0\\ 0 \ si \ D(u,v)>D_0\end{cases}$$

$$D(u,v)=\sqrt{(u^2+v^2)}$$

FFT



×



=

IFFT