

PROCESAMIENTO DE IMÁGENES

Programa de teoría

1. Adquisición y representación de imágenes.
2. Procesamiento global de imágenes.
- 3. Filtros y transformaciones locales.**
4. Transformaciones geométricas.
5. Espacios de color y el dominio frecuencial.
6. Análisis de imágenes.

Tema 3. Filtros y transformaciones locales.

- 3.1. Filtros y convoluciones
- 3.2. Suavizado, perfilado y bordes.
- 3.3. Filtros no lineales.
- 3.4. Morfología matemática.
- A.3. Filtros en OpenCV.

3.1. Filtros y convoluciones.

- **Recordatorio:** en las transformaciones **globales**, cada píxel de salida depende sólo de un píxel de entrada.

90	67	75	78	Transf. global	62	68	78	81
92	87	78	82		102	89	76	85
45	83	80	130		83	109	80	111
39	69	115	154	Transf. local	69	92	115	120
Entrada					Salida			

- No se tiene en cuenta la relación de **vecindad** entre píxeles. El resultado no varía si los píxeles son *permutados* aleatoriamente y después *reordenados*.
- **Transformación local:** el valor de un píxel depende de la vecindad local de ese píxel.

3.1. Filtros y convoluciones.

- **Transformación global:**

$$R(x,y) := f(A(x,y)) \quad \text{ó} \quad R(x,y) := f(A(x,y), B(x,y))$$

- **Filtros y transformaciones locales:**

$$R(x,y) := f(A(x-k,y-k), \dots, A(x,y), \dots, A(x+k,y+k))$$

- **Ejemplo.** Filtro de la media.

$$R(x,y) := (A(x-1,y-1) + A(x,y-1) + A(x-1,y) + A(x,y)) / 4$$

92	78	82		-	-	-
45	80	130		-	74	93
39	115	154	$\Sigma / 4$	-	70	120
A				R		

3.1. Filtros y convoluciones.

- **Ejemplo.** Entrada, A



Salida, R



- **Resultado:** la imagen se *suaviza*, *difumina* o *emborrona*.
- Las transformaciones locales tienen sentido porque existe una relación de **vecindad** entre los píxeles.
- **Recordatorio:** un píxel representa una magnitud física en un punto de una escena → dos píxeles próximos corresponden a puntos cercanos de la escena → el mundo es “continuo” → los píxeles próximos tendrán valores parecidos.

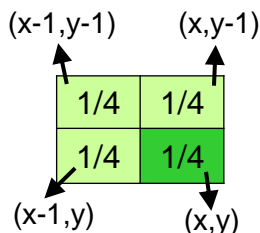
3.1. Filtros y convoluciones.

- Un tipo interesante de transformaciones locales son las convoluciones discretas.
- **Convolución discreta:** transformación en la que el valor del píxel resultante es una **combinación lineal** de los valores de los píxeles vecinos en la imagen.
- **Ejemplo.** El filtro de la media es una convolución.

$$R(x,y) := 1/4 \cdot A(x-1,y-1) + 1/4 \cdot A(x,y-1) + 1/4 \cdot A(x-1,y) + 1/4 \cdot A(x,y)$$

- **Otra forma de ver la convolución:**

Matriz de coeficientes de la combinación lineal.



3.1. Filtros y convoluciones.

- La matriz de coeficientes es conocida como la **máscara o núcleo (kernel) de convolución**.
- **Idea intuitiva:** se pasa la máscara para todo píxel de la imagen, aplicando los coeficientes según donde caigan.

Máscara de convolución

.1/4	.1/4
.1/4	.1/4

¿Cuánto valen estos píxeles?

Imagen de entrada, A

92	78	82
45	80	130
39	115	154

Imagen de salida, R

Σ

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

7



3.1. Filtros y convoluciones.

- Sea **M** una máscara de convolución. Se puede definir como **array** $[-k\dots k, -p\dots p]$ de real

En X la máscara va de -k a k, y en Y de -p a p. El punto central es (0,0)

- **Algoritmo.** Cálculo de una convolución. Denotamos la convolución como: $R := M \otimes A$

- **Entrada.** A: imagen de $\max_x \times \max_y$
M: array $[-k\dots k, -p\dots p]$ de real

- **Salida.** R: imagen de $\max_x \times \max_y$

- **Algoritmo:**
para cada píxel (x, y) de la imagen A hacer

$$R(x, y) := \sum_{i=-k..k} \sum_{j=-p..p} M(i, j) \cdot A(x+i, y+j)$$

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

8

3.1. Filtros y convoluciones.

- Ejemplos. $R := M \otimes A$

Punto central o ancla (*anchor*)

M	1/9	1/9	1/9
	1/9	1/9	1/9
	1/9	1/9	1/9

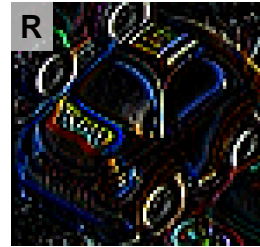
M	1	1	1
1/9.	1	1	1
	1	1	1

N	-1	1
---	----	---

El valor de un píxel es la media de los 9 píxeles circundantes

Igual que antes, pero factorizamos el múltiplo común (suma total = 1)

Restar al píxel el valor del píxel de la izquierda

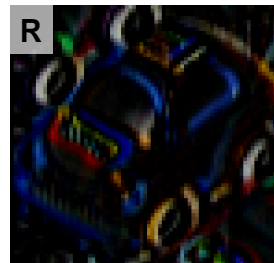


Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

9

3.1. Filtros y convoluciones.

- Sobre una imagen se pueden aplicar **sucesivas operaciones** de convolución: $\dots M_3 \otimes (M_2 \otimes (M_1 \otimes A))$



Máscara de media aplicada 4 veces

Máscara de media + máscara de resta

- Ojo:** la combinación de convoluciones es equivalente a una sola convolución:

$$M_2 \otimes (M_1 \otimes A) = M \otimes A$$

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

10

3.1. Filtros y convoluciones.

- ¿Cómo calcular el resultado de la combinación?
- **Respuesta:** comprobar el efecto sobre una imagen sólo con el píxel central a UNO (“señal impulso”).

$$\begin{array}{c}
 \begin{array}{|c|c|} \hline -1 & 1 \\ \hline \end{array} \otimes \frac{1}{9} \cdot \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \otimes \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline \end{array} = \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline \end{array} = \frac{1}{9} \cdot \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & -1 \\ \hline 0 & 1 & 0 & 0 & -1 \\ \hline 0 & 1 & 0 & 0 & -1 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}
 \end{array}$$

Máscara equivalente

3.1. Filtros y convoluciones.

- Análogamente, algunas convoluciones se pueden obtener combinando otras más simples: **núcleos separables**.
- **Ejemplo.**

$$\frac{1}{3} \cdot \begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline 1 \\ \hline \end{array} \otimes \frac{1}{3} \cdot \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline \end{array} \otimes A = \frac{1}{9} \cdot \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \otimes A$$

- **Resultado:** el filtro de la media es separable.
 - En lugar de aplicar una máscara de 3x3 se pueden aplicar dos máscaras de 1x3 y 3x1 (**máscaras unidimensionales**).
 - Puede ser útil para hacer los cálculos más **eficientes**.

3.1. Filtros y convoluciones.

- ¿Qué hacer con los píxeles de los bordes?

·1/4	·1/4
·1/4	·1/4

⊗

9	4	8
7	8	4
3	2	2

- Posibilidades:**

- Asignar un 0 en el resultado a los píxeles donde no cabe la máscara.

0	0	0
0	7	6
0	5	4

- Suponer que los píxeles que se salen tienen valor 0 (u otra constante).

2	3	3
4	7	6
2	5	4

- Modificar la operación en los píxeles que no caben (variar el multiplicador).

9	6	6
8	7	6
5	5	4

- Suponer que la imagen se extiende por los extremos (p.ej. como un espejo).

5	4	4
7	7	6
8	5	4

3.1. Filtros y convoluciones.

- Las convoluciones son una discretización de la idea de convolución usada en señales. (Repasar teoría de señales...)
- Diferencias:** las convoluciones usadas aquí son discretas y bidimensionales.
- Idea:** las máscaras de convolución son matrices de números → se pueden considerar, a su vez, como imágenes.
- Propiedades:**
 - **Asociativa:** $M_2 \otimes (M_1 \otimes A) = (M_2 \otimes M_1) \otimes A$
 - **Conmutativa:** $M_2 \otimes M_1 \otimes A = M_1 \otimes M_2 \otimes A$
 - **Ojo:** al aplicar una convolución puede ocurrir **saturación** de píxeles. Si ocurre esto, el orden sí que puede ser importante.

3.2. Suavizado, perfilado y bordes.

- Aplicando distintos operadores de convolución es posible obtener **diferentes efectos**:
 - **Suavizado**: o difuminación de la imagen, reducir contrastes abruptos en la imagen.
 - **Perfilado**: resaltar los contrastes, lo contrario al suavizado.
 - **Bordes**: detectar zonas de variación en la imagen.
 - **Detección** de cierto tipo de características, como esquinas, segmentos, etc.
- Suavizado y perfilado son más habituales en **restauración y mejora** de imágenes.
- Bordes y detección de características suelen usarse más en **análisis de imágenes**.

3.2.1. Operadores de suavizado.

- El operador de suavizado más simple es la **convolución de media** (media aritmética).
- **Parámetros** del operador:
 - Ancho y alto de la región en la que se aplica: $w \times h$.
 - Posición del ancla.
- Normalmente, w y h son impares y el ancla es el píxel central.
- La máscara es un simple array de unos de tamaño $w \times h$.

1	1	1
1	1	1
1	1	1

Máscara de media de 3x3

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Media de 5x5

3.2.1. Operadores de suavizado.

- Cuanto mayor es la máscara, mayor es el efecto de difuminación de la imagen.



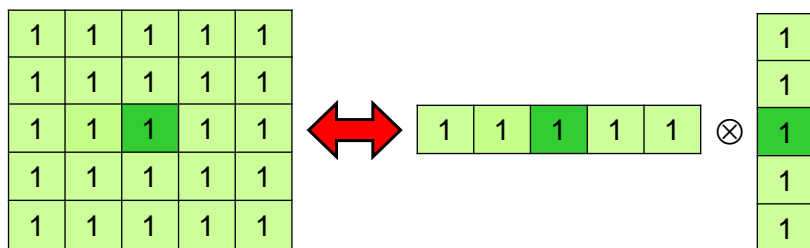
Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

17



3.2.1. Operadores de suavizado.

- **Ventajas** (respecto a otros suavizados):
 - Sencillo y rápido de aplicar.
 - Fácil definir un comportamiento para los **píxeles de los bordes**: tomar la media de los píxeles que quepan.
 - Recordatorio: el operador de media es **separable**.



Media de 5x5
Total: 25 sumas $\rightarrow o(n^2)$

Media de 5x1 y de 1x5
Total: 10 sumas $\rightarrow o(2n)$

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

18

3.2.1. Operadores de suavizado.

- En algunos casos puede ser interesante aplicar **suavizados direccionales**: horizontales, verticales o en cualquier dirección.

1	1	1	1	1
---	---	---	---	---

Media horizontal 5 píxeles

1
1
1

Media vertical 3p

0	0	1
0	1	0
1	0	0

Media diagonal 3p



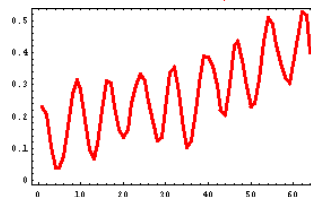
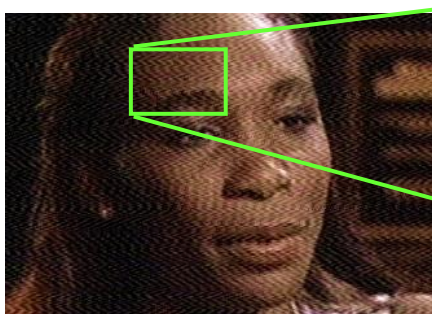
Media horiz. 31p



Media vert. 31p

3.2.1. Operadores de suavizado.

- Ejemplo 1.** En una aplicación trabajamos con imágenes capturadas de TV. El canal tiene muchas interferencias, que provocan una oscilación cada 7 píxeles horizontales. ¿Cómo reducir el efecto de las interferencias?

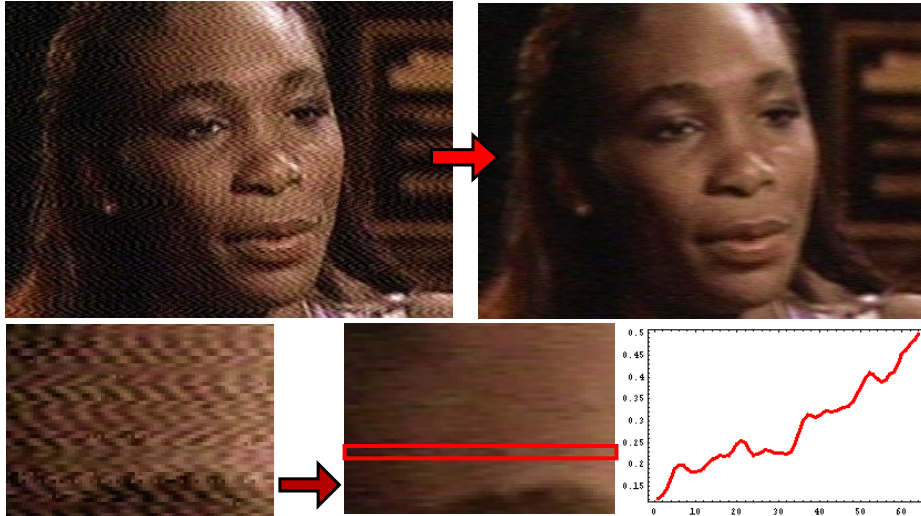


- Idea:** Probar con una media horizontal de 7 píxeles...

3.2.1. Operadores de suavizado.

- Aplicación de media horizontal de 7 píxeles.

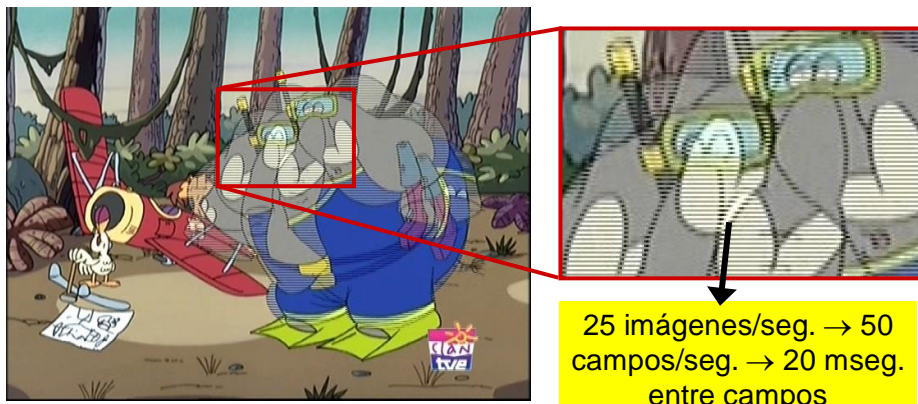
1	1	1	1	1	1	1
---	---	---	---	---	---	---



Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

3.2.1. Operadores de suavizado.

- **Ejemplo 2. Entrelazado de vídeo:** para aumentar la frecuencia de refresco del vídeo se separan las líneas pares y las impares (1 **campo** (*field*)=1/2 imagen). Al capturar una imagen, se mezclan los campos produciendo efectos raros.



Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

3.2.1. Operadores de suavizado.

- Duplicar las filas pares (o las impares) y luego aplicar una media vertical de 2 píxeles (para interpolar).

1
1



Imagen entrelazada

Duplicadas filas pares

Suavizado vertical (interp.)



Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

23

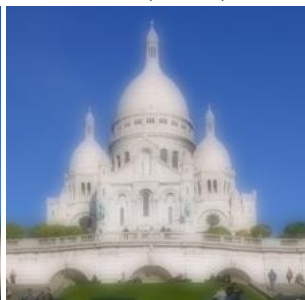
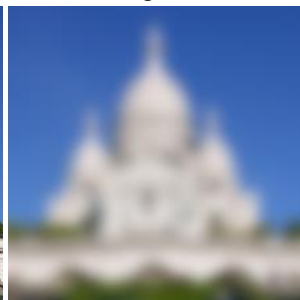
3.2.1. Operadores de suavizado.

- Ejemplo 3. Efecto de niebla.** Dada una imagen bien definida, queremos simular una niebla (objetivo *empañado*).
- Idea:** calcular una media ponderada entre la imagen original y un suavizado gaussiano de la imagen.

A. Imagen original

B. Suaviz. gauss. 40x40

Suma: $0,3A+0,7B$



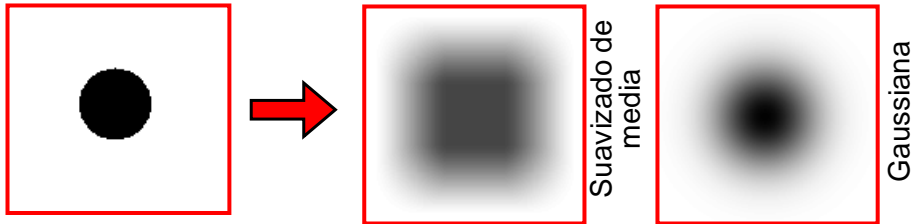
- Se puede conseguir el mismo resultado con una sola convolución. ¿Cuál sería la máscara equivalente?

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

24

3.2.1. Operadores de suavizado.

- Cuando se aplica la media con tamaños grandes se obtienen resultados **artificiosos** (a menudo **indeseados**).



- **Motivo:** la media se calcula en una región cuadrada.
- Sería mejor aplicarla a una **región "redonda"**.
- O, mejor, usar suavizado gaussiano...

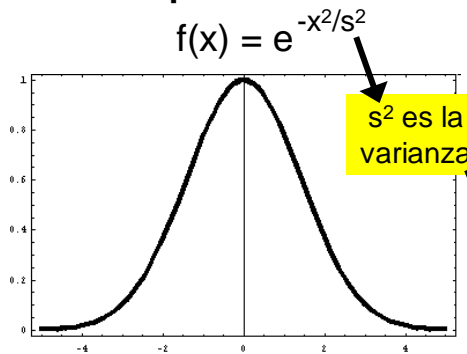
0	0	1	1	1	0	0
0	1	1	1	1	1	0
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
0	1	1	1	1	1	0
0	0	1	1	1	0	0

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

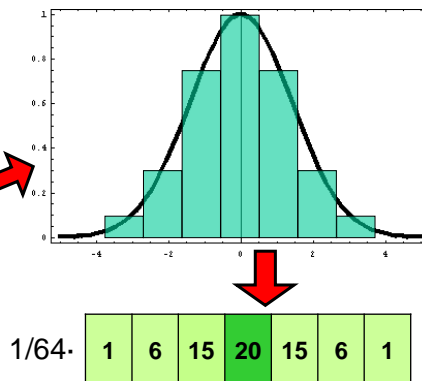
3.2.1. Operadores de suavizado.

- **Suavizado gaussiano:** media ponderada, donde los pesos toman la forma de una campana de Gauss.
- **Ejemplo.** Suavizado gaussiano horizontal.

Campana de Gauss



Campana discreta



Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

26



3.2.1. Operadores de suavizado.

- La **varianza**, s^2 , indica el nivel de suavizado.
 - **Varianza grande:** campana más ancha, más suavizado.
 - **Varianza pequeña:** campana más estrecha, menos suavizado.
 - Se mide en píxeles.
- **Cálculo de la máscara gaussiana (1D):** calcular la función, discretizar en el rango, discretizar en el valor y calcular el multiplicador...

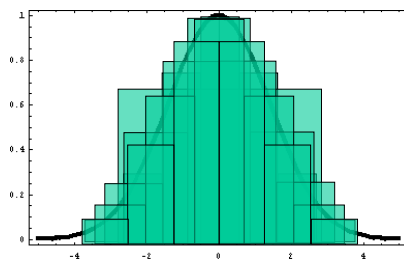
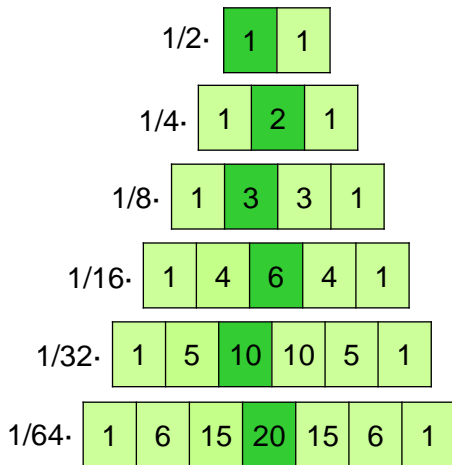
```

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
    
```

- ¿No existe una forma más rápida?
- **Idea:** el triángulo de Pascal.

3.2.1. Operadores de suavizado.

- ¡**Magia!** Las filas del triángulo de Pascal forman discretizaciones de la campana de Gauss.



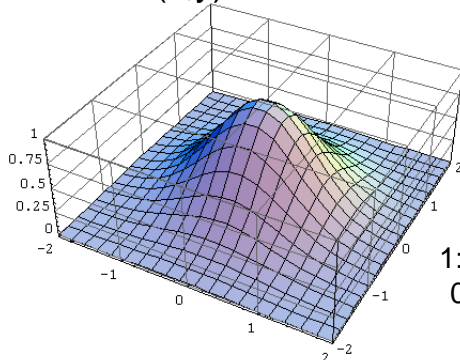
¿Por qué ocurre así?
Recordar el **teorema central del límite...**

3.2.1. Operadores de suavizado.

- Normalmente, el suavizado gaussiano se aplica en dos dimensiones. Los pesos de la máscara dependen de la distancia al píxel central.

Campana de Gauss 2D

$$f(x,y) = e^{-(x^2+y^2)/s^2}$$

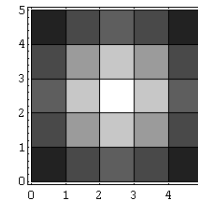
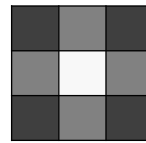


Máscara gaussiana de 3x3

1	2	1
2	4	2
1	2	1

1/16 ·

1: blanco
0: negro



Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

29

3.2.1. Operadores de suavizado.

- **Propiedad interesante:** el filtro gaussiano es separable.
- **Resultado:** se puede obtener un suavizado 2D aplicando dos máscaras gaussianas bidimensionales, una horizontal y otra vertical.

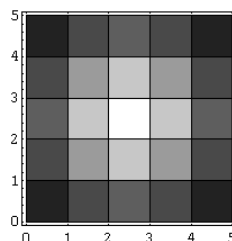
1	2	1
2	4	2
1	2	1



1	2	1
---	---	---



1
2
1



Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

30

3.2.1. Operadores de suavizado.

- Comparación: media y suavizado gaussiano, 2D.



Media de 11x11



Media de 21x21



Gaussiana 21x21



Gaussiana 41x41

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

31



3.2.1. Operadores de suavizado.

- Comparación: media y suavizado gaussiano, 1D.



Media horiz. 31p



Media vert. 31p



Gaussiana 61x1



Gaussiana 1x61

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

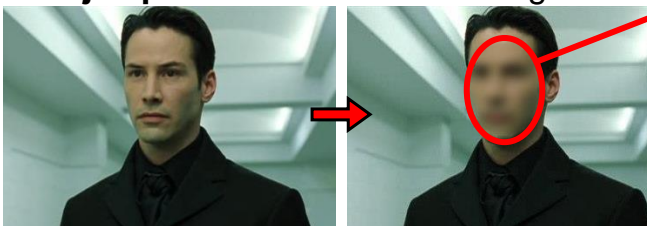
32

3.2.1. Operadores de suavizado.

- **Resultados** de la comparación:
 - Para conseguir un mismo “**grado de suavizado**” la máscara gaussiana debe ser de mayor tamaño.
 - Se puede tomar como medida la **varianza** de la máscara correspondiente.
 - El efecto del suavizado gaussiano es más **natural** (más similar a un desenfoque) que la media.
 - Suele ser más habitual en procesamiento y análisis de imágenes.
 - Ambos filtros son **separables**.
 - Si la máscara es de $n \times n$, pasamos de $o(n^2)$ a $o(2n)$.

3.2.1. Operadores de suavizado.

- **Ejemplo 1.** Protección de testigos.



Se aplica un suavizado pero sólo en cierta región de interés (**ROI**), en este caso elíptica.

¿Cómo encontrar la posición de la cara automáticamente?

- **Ejemplo 2.** Resaltar objetos de interés.



Se suaviza el fondo para destacar al personaje, simulando un desenfoque.

3.2.1. Operadores de suavizado.

- **Ejemplo 2b.** Simulación de efecto *tilt-shift*.



La imagen parece enfocada en una zona pequeña, simulando un efecto de **miniatura**.

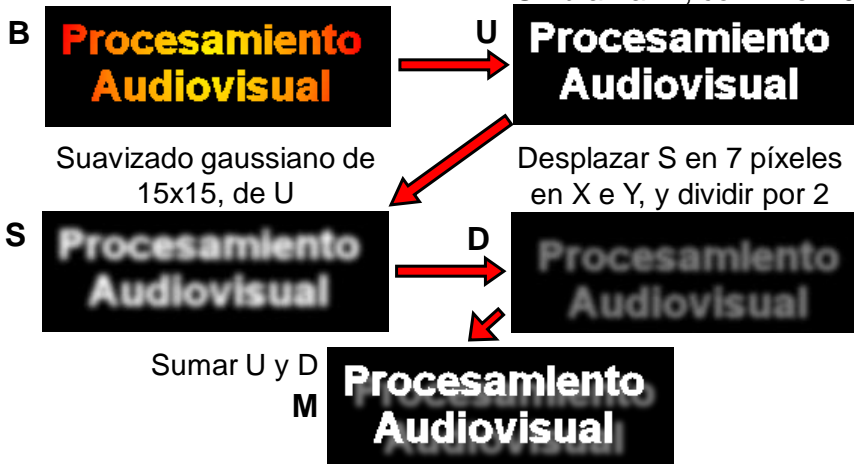


Proces
Tema 3. Filtros y transformaciones locales.

3.2.1. Operadores de suavizado.

- **Ejemplo 3.** Sombra difusa.

Añadir a una imagen **A** una etiqueta de texto **B**, con un efecto de sombra difuminada. Umbralizar B, con nivel 10



Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

3.2.1. Operadores de suavizado.

A



Multiplicar A por M, en posición (x_0, y_0)

B

Procesamiento Audiovisual

M

Procesamiento Audiovisual

Sumar T y B, en posición (x_0, y_0)

T



R



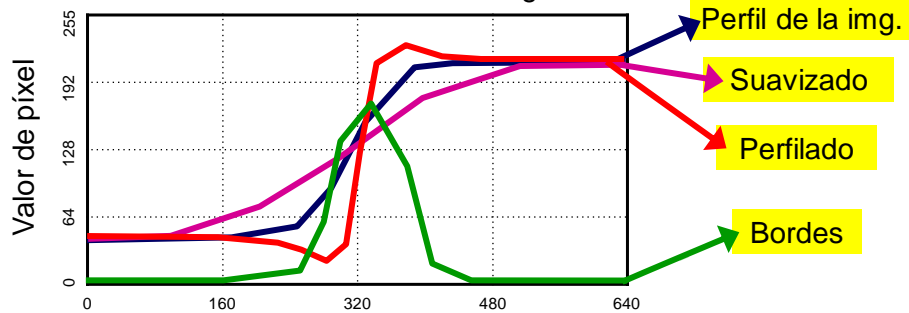
Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

37

3.2.2. Operadores de bordes.

- **Perfilado y detección de bordes** están relacionados con el suavizado:
 - **Suavizado:** reducir las variaciones en la imagen.
 - **Perfilado:** aumentar las variaciones en la imagen.
 - **Bordes:** encontrar las zonas de variación.

Perfil de una fila de una imagen

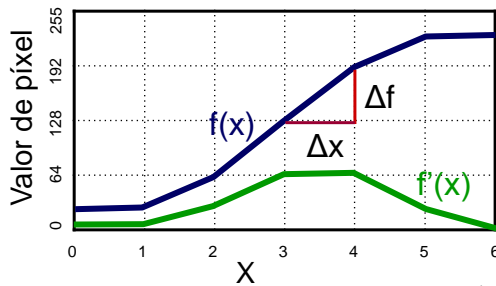


Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

38

3.2.2. Operadores de bordes.

- Matemáticamente, la variación de una función $f(x)$ cualquiera viene dada por la **derivada** de esa función:
 - $f'(x) > 0$: función creciente en X
 - $f'(x) < 0$: función decreciente en X
 - $f'(x) = 0$: función uniforme en X
- En nuestro caso, tenemos **funciones discretas**. La “**derivada discreta**” se obtiene calculando diferencias.



$$f'(x) = \Delta f / \Delta x$$

$$\Delta f = f(x) - f(x-1) \quad \Delta x = 1$$

$$f'(x) = f(x) - f(x-1)$$

- **Conclusión:** la derivada se calculará con máscaras del tipo:

-1	1
----	---

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

39

3.2.2. Operadores de bordes.

Máscara de derivada en X (M):

-1	1
----	---

Derivada en Y:

-1
1

Derivadas en diagonales:

-1	0	0	-1
0	1	1	0

- **Ejemplo. Derivada en X.** $R := M \otimes A$



Imagen de entrada



Derivada en X (x2)

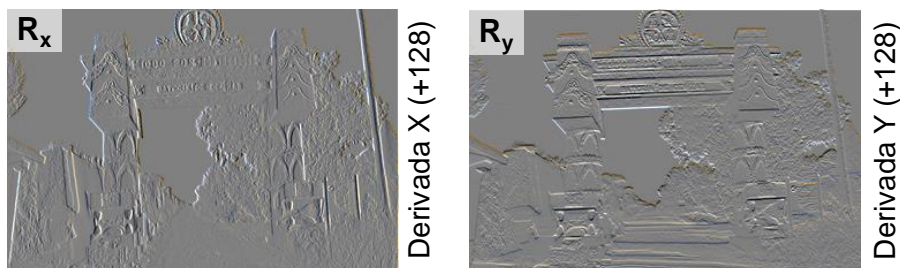
$$[0..255] - [0..255] = [-255..255]$$

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

40

3.2.2. Operadores de bordes.

- Los bordes decrecientes se saturan a 0...
- Podemos sumar 128 para apreciar mejor el resultado:
 - Gris (128): diferencia 0
 - Negro: decreciente
 - Blanco: creciente

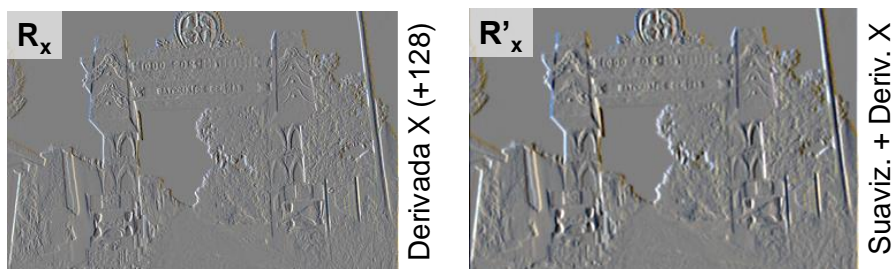


- Se produce una especie de “bajorrelieve” (*emboss*), que puede usarse en efectos especiales.

3.2.2. Operadores de bordes.

- Los operadores de bordes son muy **sensibles al ruido**.
- Es posible (y adecuado) **combinar** los operadores de **bordes** con **suavizados**.

$$\begin{array}{|c|c|} \hline -1 & 1 \\ \hline \end{array} \otimes \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 1 & 1 & -1 & -1 \\ \hline 2 & 2 & -2 & -2 \\ \hline 1 & 1 & -1 & -1 \\ \hline \end{array}$$



3.2.2. Operadores de bordes.

- Existen algunos **operadores** de bordes **estándar**.
- **Filtros de Prewitt:**

Filtro de Prewitt 3x3, derivada en X

-1	0	1
-1	0	1
-1	0	1

Filtro de Prewitt 3x3, derivada en Y

-1	-1	-1
0	0	0
1	1	1

- **Filtros de Scharr:**

Filtro de Scharr 3x3, derivada en X

-3	0	3
-10	0	10
-3	0	3

Filtro de Scharr 3x3, derivada en Y

-3	-10	-3
0	0	0
3	10	3

3.2.2. Operadores de bordes.

- **Filtros de Sobel:** se construyen usando la derivada de la gaussiana.

Filtro de Sobel 3x3, derivada en X

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

Filtro de Sobel 3x3, derivada en Y

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

- Además, el filtro de Sobel permite calcular derivadas conjuntas en X e Y, derivadas segundas, terceras, etc.
- **Ejemplo.** Derivada segunda en X.

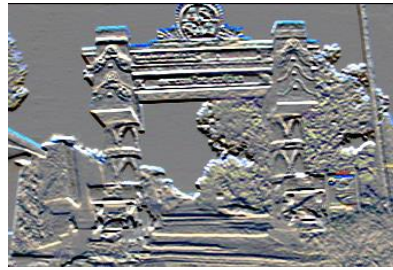
$$\begin{bmatrix} -1 & 1 \end{bmatrix} \otimes \begin{bmatrix} -1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 2 & -1 \end{bmatrix}$$

3.2.2. Operadores de bordes.

- Ejemplos.



Imagen de entrada



Prewitt Y (3x3)



Sobel Y (3x3)



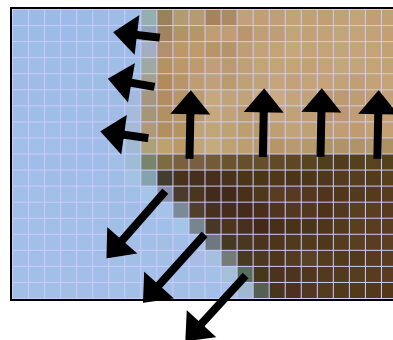
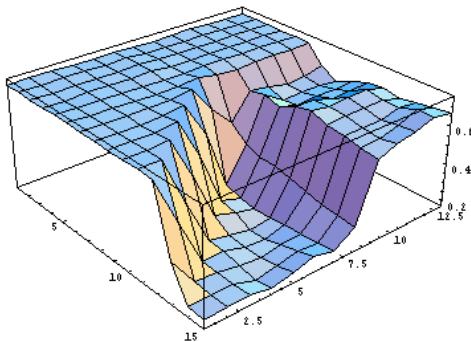
Sobel 2ª deriv. Y

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

45

3.2.2. Operadores de bordes.

- Realmente, en dos o más dimensiones, en lugar de la derivada tiene más sentido el concepto de **gradiente**.
- ¿Qué es el gradiente? → Repasar cálculo...
- **El gradiente** indica la dirección de máxima variación de una función (en 2D, la máxima pendiente).

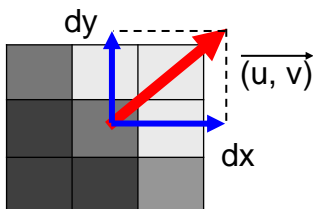


Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

46

3.2.2. Operadores de bordes.

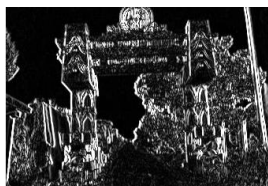
- El **gradiente** en un punto es un vector (u, v) :
 - **Ángulo**: dirección de máxima variación.
 - **Magnitud**: intensidad de la variación.



- El **gradiente** está relacionado con las **derivadas**:
 - u = Derivada en X del punto
 - v = Derivada en Y del punto
 - Teniendo dy y dx , ¿cuánto vale el ángulo y la magnitud?

3.2.2. Operadores de bordes.

- **Cálculo del gradiente**:
 - Calcular **derivada en X**: D_x (por ejemplo, con un filtro de Sobel, Prewitt,...)
 - Calcular **derivada en Y**: D_y
 - **Magnitud** del gradiente: $\sqrt{D_x^2 + D_y^2}$
 - **Ángulo** del gradiente: $\text{atan2}(D_y, D_x)$



Valor absoluto de derivada en X (Sobel de 3x3)



Valor absoluto de derivada en Y (Sobel de 3x3)



Magnitud del gradiente

3.2.2. Operadores de bordes.

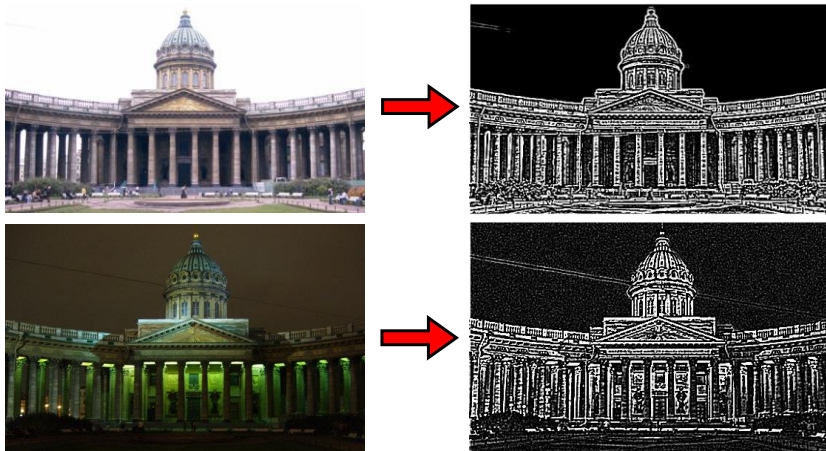
- El gradiente da lugar al concepto de **borde**.
- Un **borde** en una imagen es una curva a lo largo de la cual el gradiente es máximo.



El borde es perpendicular a la dirección del gradiente.

3.2.2. Operadores de bordes.

- Los bordes de una escena son **invariantes a cambios de luminosidad, color de la fuente de luz, etc.** → En **análisis de imágenes** usar los bordes (en lugar de las originales).



3.2.2. Operadores de bordes.

- **Otras formas de calcular los bordes:**

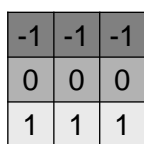
1. Calcular la derivada en diferentes direcciones: D_1, D_2, D_3, D_4 .

2. Para cada punto, la magnitud del gradiente es la derivada de máximo valor absoluto:

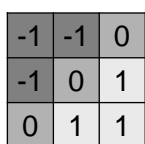
$$G(x,y) := \max \{ |D_1(x,y)|, |D_2(x,y)|, |D_3(x,y)|, |D_4(x,y)| \}$$

3. La dirección del gradiente viene dada por el ángulo que ha producido el máximo:

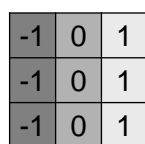
$$A(x,y) := \operatorname{argmax} \{ |D_1(x,y)|, |D_2(x,y)|, |D_3(x,y)|, |D_4(x,y)| \}$$



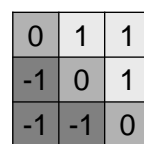
D_1 : N-S



D_2 : NE-SO



D_3 : E-O

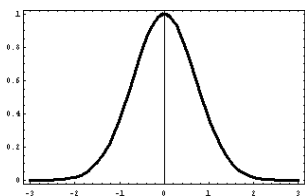


D_4 : SE-NO

3.2.2. Operadores de bordes.

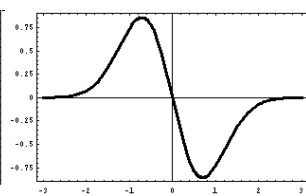
- Otra forma más sencilla (aproximada) es usar máscaras de convolución adecuadas, por ejemplo de **Laplace**.

- La **función de Laplace** es la segunda derivada de la gaussiana.



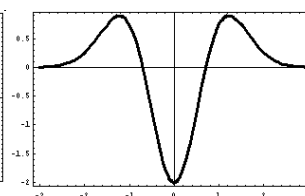
$$f(x) = e^{-x^2/s^2}$$

Másc. Gaussiana
Operador de suavizado



$$df(x)/dx$$

Másc. Sobel
Operador de derivación



$$d^2f(x)/dx^2$$

Másc. Laplaciana
Operador de gradiente

3.2.2. Operadores de bordes.

- La máscara **laplaciana** se define usando la función de Laplace.
- Ejemplos de **máscaras de Laplace**.

0	1	0
1	-4	1
0	1	0

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

“Diferencia entre el píxel central y la media de sus vecinos...”

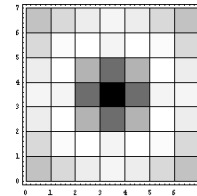


Imagen de entrada



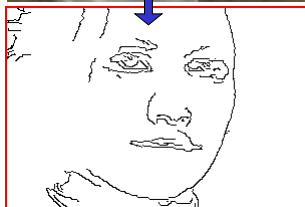
Laplaciana 2 (3x3)

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

53

3.2.2. Operadores de bordes.

- **Detector de bordes de Canny:**
 - No sólo usa convoluciones (operadores de gradiente), sino que busca el **máximo gradiente** a lo largo de un borde.
 - El resultado es una **imagen binaria** (borde/no borde), ajustable mediante un umbral.



Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

54

3.2.3. Operadores de perfilado.

- **Perfilado:** destacar y hacer más visibles las variaciones y bordes de la imagen. Es lo contrario al suavizado.
- Permite eliminar la apariencia borrosa de las imágenes, debida a imperfecciones en las lentes.
- ... aunque tampoco se pueden hacer milagros...



← Suavizado Original Perfilado →

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

55

3.2.3. Operadores de perfilado.

- El perfilado se puede conseguir sumando a la imagen **original**, la **laplaciana** ponderada por cierto factor.
- Lo cual equivale a usar una máscara de convolución adecuada:

$$\begin{array}{c} \text{Laplaciana} \\ 1 \cdot \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \end{array} + \begin{array}{c} \text{Identidad} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{array} = \begin{array}{c} \text{Perfilado} \\ \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \end{array}$$

- Más o menos perfilado dando distintos pesos, **a**.

$$\begin{array}{c} a \cdot \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \end{array} + \begin{array}{c} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{array} = \begin{array}{c} \begin{bmatrix} 0 & -a & 0 \\ -a & 4a+1 & -a \\ 0 & -a & 0 \end{bmatrix} \end{array}$$

Ojo: la función Laplacian usa máscaras "invertidas", luego **a** debe ser < 0

56

3.2.3. Operadores de perfilado.

- **Ejemplos.** Variando pesos y tamaño de la laplaciana.



Imagen de entrada



Perfilado 33%, 3x3



Perfilado 60%, 1x1



Perfilado 15%, 7x7

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

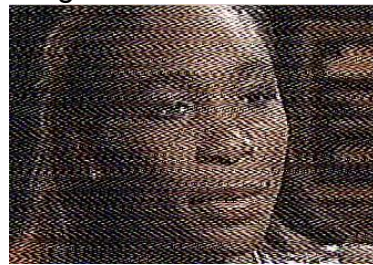
57

3.2.3. Operadores de perfilado.

- **Cuidado con el perfilado.** La operación de perfilado aumenta el nivel de ruido de la imagen.



Imagen con ruido
por interferencias
TV



Perfilado 33%, 3x3



Imagen con ruido
por compresión
JPEG



Perfilado 60%, 3x3

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

58

3.2. Suavizado, perfilado y bordes.

Conclusiones:

- Las **convoluciones** son una herramienta fundamental en procesamiento de imágenes.
 - **Una misma base común**: combinaciones lineales de una vecindad local de los píxeles (de cierto tamaño).
 - **Diversos usos**: según los valores de los coeficientes: suavizado, eliminación de ruido, bordes, perfilado, etc.
- Se pueden definir **operaciones similares** sobre **vídeo** (usando la dimensión temporal, por ejemplo, suavizado a lo largo del tiempo), y sobre **audio digital** (por ejemplo, suavizado de la señal o introducción de eco).
- Es importante conocer el **significado matemático** de los procesos aplicados (derivadas, gradientes, integrales,...).

3.3. Filtros no lineales.

- **Recordatorio**: las transformaciones locales son funciones del tipo:

$$R(x,y) := f(A(x-k,y-k), \dots, A(x,y), \dots, A(x+k,y+k))$$

- En las convoluciones, **f** es una **combinación lineal** cualquiera. Pero...
- También puede ser interesante usar otras **funciones no lineales**.
- **Ejemplo**, media geométrica.

$$R(x,y) := \sqrt[4]{A(x-1,y-1) \cdot A(x,y-1) \cdot A(x-1,y) \cdot A(x,y)}$$

3.3. Filtros no lineales.

- **Ejemplo.** Media geométrica de 5x5. ... muy parecido a la media aritmética...



- Aunque existen muchas (en teoría infinitas) posibles transformaciones no lineales, en la práctica no todas son útiles e interesantes.
- Las que más se usan son: **máximo**, **mínimo** y **mediana**.

3.3. Filtros no lineales.

- **Filtro de Máximo:**

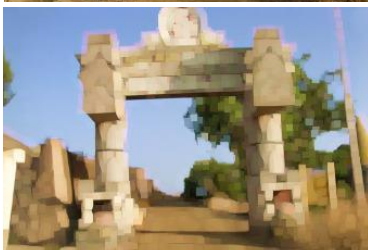
$$R(x,y) := \max \{A(x-k,y-k), \dots, A(x,y), \dots, A(x+k,y+k)\}$$
donde **k** es el radio, el tamaño (o *apertura*) es **2k+1**



Imagen de entrada



Máximo, tamaño 3



Máx., tamaño 6



Máx., tamaño 12

3.3. Filtros no lineales.

- El resultado es un cierto efecto de **difuminación** y **aclaramiento** de la imagen. Desaparecen los detalles más oscuros.
- Si el **tamaño es grande**, pueden ocurrir dos efectos:

1. Efecto de cuadrículado.

Como el máximo se aplica en una zona cuadrada, los píxeles muy claros *generan* un cuadrado uniforme alrededor.



2. Aparición de colores falsos.

Al aplicarlo en los tres canales (R,G,B) independientemente, el máximo en los 3 puede no corresponder a un color presente en la imagen original.



3.3. Filtros no lineales.

• **Filtro de Mínimo:**

$R(x,y) := \min \{A(x-k,y-k), \dots, A(x,y), \dots, A(x+k,y+k)\}$
donde **k** es el radio, el tamaño (o *apertura*) es **2k+1**



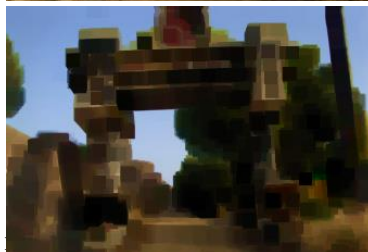
Imagen de entrada



Mínimo, tamaño 3



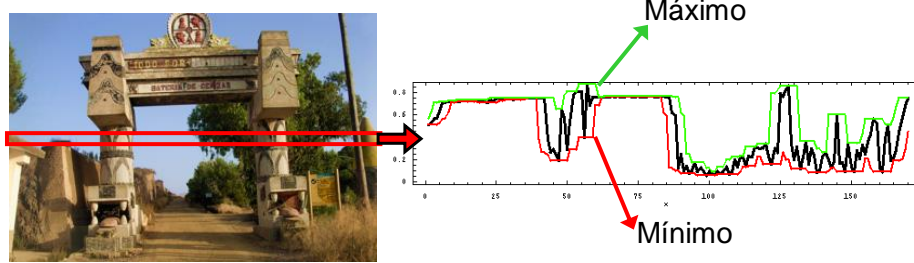
Mín., tamaño 6



Mín., tamaño 12

3.3. Filtros no lineales.

- El efecto es **parecido** al máximo, pero tomando los valores menores (los más oscuros).



- **Ideas:**

- Para evitar el **efecto de cuadrículado** se podría aplicar el máximo/mínimo a una zona circular.
- Para evitar la aparición de **colores falsos** se podría tomar el máximo de las sumas de R+G+B.

3.3. Filtros no lineales.

- Otro filtro relacionado es el de la **mediana**.
 - **La mediana** de m números es un número p tal que $\lceil m/2 \rceil$ de esos números son $\leq p$, y otros $\lfloor m/2 \rfloor$ son $\geq p$.
- $$R(x,y) := \text{mediana} \{A(x-k,y-k), \dots, A(x,y), \dots, A(x+k,y+k)\}$$



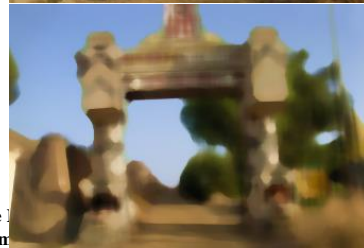
Imagen de entrada



Mediana 3x3



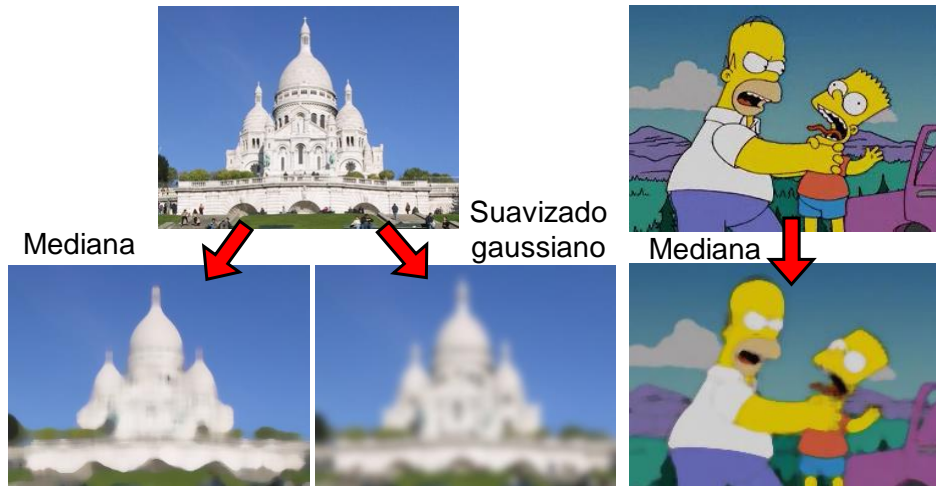
Mediana 6x6



Mediana 12x12

3.3. Filtros no lineales.

- La mediana produce un efecto de **suavizado**, aunque más “abrupto” en los bordes que la media y el suavizado gaussiano.



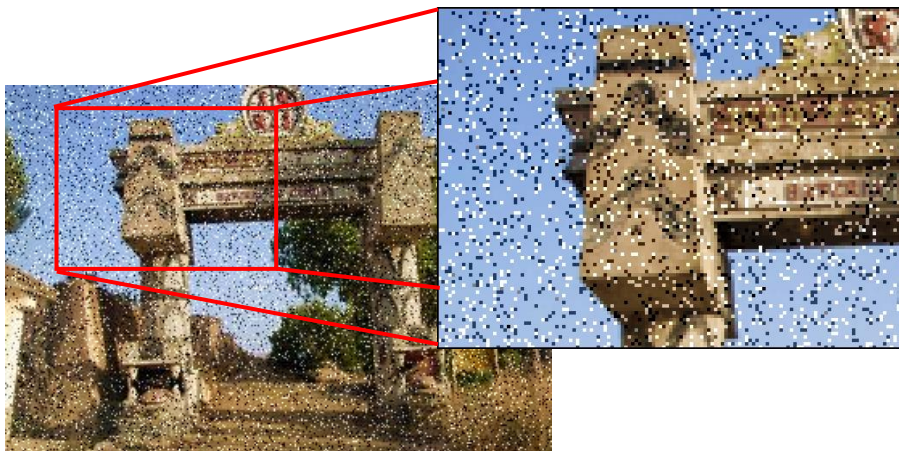
- Pero el verdadero interés es la **eliminación de ruido puntual**.

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

67

3.3. Filtros no lineales.

- **Ejemplo.** El ruido denominado “sal y pimienta” es producido por picos de perturbación, positivos o negativos. Puede deberse a un canal ruidoso.



Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

68

3.3. Filtros no lineales.

- Se puede intentar eliminar (o reducir) el ruido con un filtro gaussiano o con una mediana.



Mediana 3x3



Filtro gaussiano

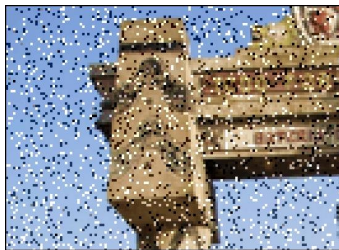


Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

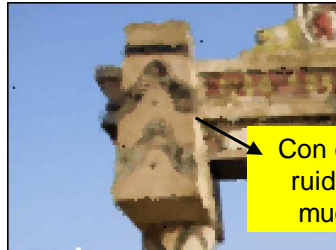
69

3.3. Filtros no lineales.

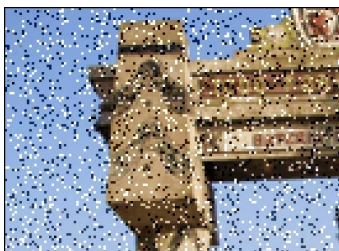
- Se puede intentar eliminar (o reducir) el ruido con un filtro gaussiano o con una mediana.



Mediana 3x3



Con este tipo de ruido funciona mucho mejor



Filtro gaussiano



El ruido se difumina, pero no llega a desaparecer

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

70

3.3. Filtros no lineales.

- Otros ejemplos de eliminación de ruido.



Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

71

3.3. Filtros no lineales.

- **Más filtros no lineales:** recordar la ecualización local del histograma.
 - Considerar una **operación global** como el estiramiento, la ecualización del histograma o la umbralización.
 - **Globalmente** se calculan los parámetros y se aplican a **toda la imagen**: estiramiento (máximo y mínimo del histograma), ecualización (función de ecualización) y umbralización (umbral a aplicar).
 - En lugar de aplicarlos globalmente, calcular los **parámetros para cada punto**, usando una vecindad local.
 - Aplicar la transformación a cada punto, usando sus parámetros **específicos**.

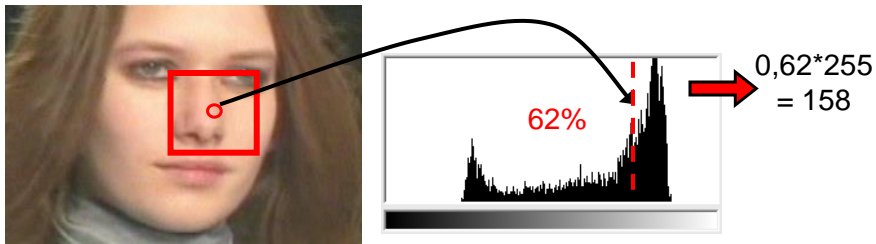
Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

72

3.3. Filtros no lineales.

- **Algoritmo. Ecuación local de tamaño $a \times b$:**

1. Para cada punto (x,y) de la imagen A , calcular el histograma de una región rectangular desde $(x-a, y-b)$ hasta $(x+a, y+b) \rightarrow H(v)$
2. Calcular el percentil del valor $A(x,y)$, es decir:
$$p := (H(0) + H(1) + \dots + H(A(x,y))) / ((2a+1)(2b+1))$$
3. Hacer $R(x,y) := 255 \cdot p$



Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

73

3.3. Filtros no lineales.

- **Ejemplo. Ecuación local del histograma.**



Imagen de entrada

Resolución: 299x202



Tamaño: 25x25

Tamaño: 50x50

Tamaño: 120x120

- La misma idea se podría aplicar a umbralización y estiramiento.

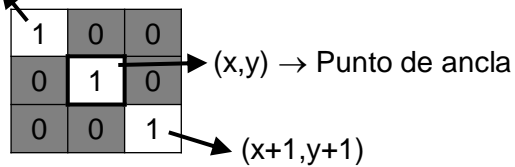
Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

74

3.4. Morfología matemática.

- Los **operadores de morfología matemática** son un conjunto de filtros locales sencillos, que se pueden combinar para obtener resultados más complejos.
- Originalmente, están definidos sobre **imágenes binarias**.
- La idea es muy parecida a una convolución, pero utilizando las **operaciones booleanas** AND y OR.
- **Ejemplo.** $R(x,y) := A(x-1,y-1) \text{ AND } A(x,y) \text{ AND } A(x+1,y+1)$

**Elemento
estructurante**
(= máscara de
convolución)



Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

75

3.4. Morfología matemática.

- El **elemento estructurante** define los píxeles que se usan en la operación y los que no.
- Dado un elemento estructurante, **E**, de cierta forma y tamaño, y una imagen binaria **B**, se definen dos **operaciones**:
 - **Dilatación** $B \oplus E$. Combinar con OR los valores correspondientes a los píxeles 1 del elemento estructurante.
 - **Erosión** $B \otimes E$. Combinar con AND los valores correspondientes a los píxeles 1 del elemento estructurante.
- La idea se puede generalizar a **imágenes no binarias**:
 - **Dilatación**. Combinar con Máximo.
 - **Erosión**. Combinar con Mínimo.

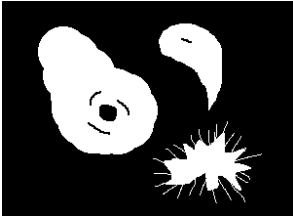
Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

76

3.4. Morfología matemática.

- El efecto de la **dilatación** es **extender o ampliar** las regiones de la imagen con valor 1 (color blanco), mientras que la **erosión** las **reduce**.
- La cantidad depende del **tamaño** y **forma** del elemento estructurante y del **número de veces** que se aplican.
- **Ejemplo.**

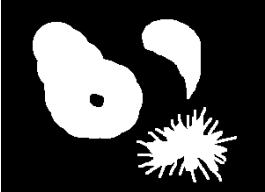
Imagen de entrada



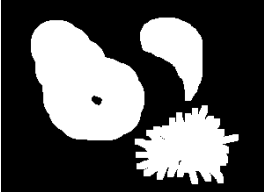
Elemento estructurante

1	1	1
1	1	1
1	1	1


Dilatación 1




Dilatación 3



Erosión 1



Erosión 3



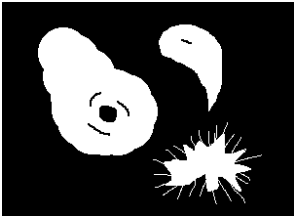
Procesamiento de Imágenes 77

Tema 3. Filtros y transformaciones locales.

3.4. Morfología matemática.

- Existen otras dos **operaciones frecuentes** basadas en erosión y dilatación:
 - **Abrir.** Aplicar erosión y después dilatación: $(B \otimes E) \oplus E$
 - **Cerrar.** Aplicar dilatación y después erosión: $(B \oplus E) \otimes E$


Imagen de entrada



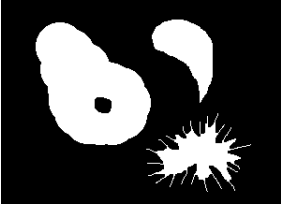
Elemento estructurante

1	1	1
1	1	1
1	1	1

Abrir:
desaparecen los puntos sueltos o estructuras finas



Cerrar: se rellenan los huecos negros de cierto tamaño



Procesamiento de Imágenes 78

Tema 3. Filtros y transformaciones locales.

3.4. Morfología matemática.

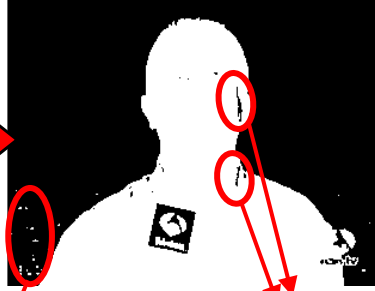
- **Ejemplo. Segmentación de objetos.**

Para segmentar un objeto del fondo usamos una simple umbralización. Funciona más o menos bien, pero aparecen algunos puntos mal clasificados.

Imagen de entrada



Umbralizada ($u=130$)



- Usar morfología para arreglar los falsos.

Falsos positivos

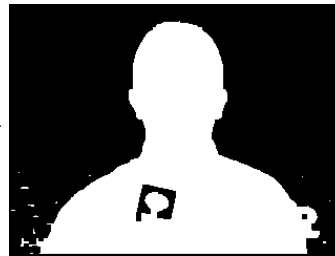
Falsos negativos

3.4. Morfología matemática.

Imagen umbralizada



Cerrar 2 ($B \oplus E \oplus E$) $\otimes E \otimes E$



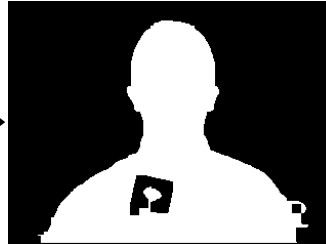
Eliminar falsos negativos

Abrir 1 ($B \otimes E$) $\oplus E$



Eliminar falsos positivos

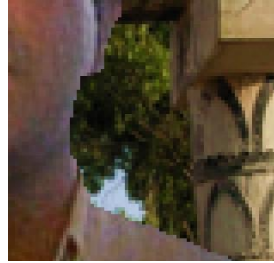
Erosión 2 ($B \otimes E$) $\otimes E$



Eliminar píxeles de los bordes

3.4. Morfología matemática.

- El resultado es la **máscara** para segmentar el objeto.



- ¿Para qué se hacen las dos últimas erosiones?



de Imágenes

81

Tema 3. Filtros y transformaciones locales.

3.4. Morfología matemática.

- En **imágenes no binarias**, el resultado de dilatación y erosión es parecido a las operaciones de máximo y mínimo.
- De hecho, es igual si el elemento estructurante es todo 1.



Imagen entrada



Erosión, 1



Dilatación, 3



Cierre, 2

Procesamiento de Imágenes

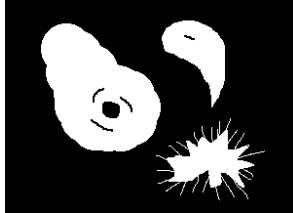
82

Tema 3. Filtros y transformaciones locales.

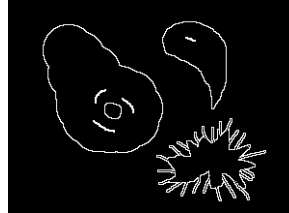
3.4. Morfología matemática.

- Existen otras operaciones de morfología, basadas en las elementales, que son útiles en análisis de imágenes.
- **Ejemplo 1. Borde morfológico: $(B \oplus E) - B$**

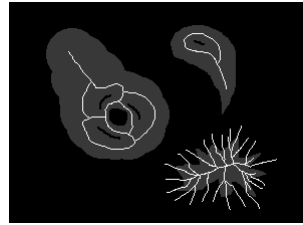
Imagen de entrada



Borde morfológico



- **Ejemplo 2. Adelgazamiento (*thinning*).** Aplicar una erosión, pero no eliminar el punto (no poner a 0) si se separa una región conexa en varias o si sólo queda un punto.



Procesamiento de Imágenes

Tema 3. Filtros y transformaciones locales.

83

3. Filtros y transformaciones locales.

Conclusiones:

- Las operaciones de **procesamiento local** son **esenciales** en mejora de imágenes, restauración, análisis, etc.
- Dos **categorías** básicas:
 - **Filtros lineales o convoluciones:** la salida es una combinación lineal de los píxeles en una vecindad → **Suavizado, bordes, perfilado**, etc.
 - **Filtros no lineales:** se usan funciones no lineales → **Máximo, mínimo**, operaciones de **morfología**, etc.
- Es posible **combinarlas** con operaciones de procesamiento global.
- La idea de “localidad” se puede extender a vídeo y a sonido, considerando la **dimensión temporal**.

Procesamiento de Imágenes

Tema 3. Filtros y transformaciones locales.

84

Anexo A.3. Filtros en OpenCV.

- Filtros lineales predefinidos.
- Filtros lineales arbitrarios.
- Filtros de máximo, mínimo y mediana.
- Operaciones de morfología matemática.
- Ejemplos.

A.3. Filtros en OpenCV.

Operaciones de procesamiento local:

- De modo práctico, podemos clasificar las operaciones de filtrado existentes en los siguientes grupos:
 - Filtros **lineales predefinidos** de suavizado y detección de bordes.
 - Filtros **lineales arbitrarios**, definidos por el usuario en tiempo de ejecución.
 - Filtros de **máximo**, **mínimo** y **mediana**.
 - Operaciones de **morfología** matemática.
- **Ojo** con las **restricciones**. Algunas operaciones requieren imágenes de 1 canal o profundidad float (habrá que usar: split, merge y convertTo).

A.3. Filtros en OpenCV.

- Filtros **lineales predefinidos**:
 - blur, GaussianBlur, Sobel, Scharr, Laplacian, Canny (este último es un filtro estándar, pero no se puede considerar como lineal).
- Filtros **lineales arbitrarios**:
 - filter2D.
- Filtros de **máximo, mínimo y mediana**:
 - dilate (máximo), erode (mínimo), medianBlur (mediana).
- Filtros de **morfología matemática**:
 - erode, dilate, morphologyEx.

A.3. Filtros en OpenCV.

- **Filtros de suavizado de una imagen**:
void **blur** (Mat src, Mat dst, Size ksize,
 [Point ancla=Point(-1,-1), int tipoBorde=DEFAULT]);
 - **src**: imagen de entrada. Puede ser de distintos tipos y número de canales.
 - **dst**: imagen de salida. Debe ser del mismo tamaño y tipo que src.
 - **ksize**: tamaño del suavizado (ancho por alto).
 - **ancla**: punto de ancla de la convolución. El valor por defecto es en el centro de la máscara.
 - **tipoBorde**: indica el modo de operar en los píxeles de los bordes (BORDER_REPLICATE, BORDER_REFLECT, BORDER_WRAP, etc.). Lo mejor es dejar el valor por defecto.

A.3. Filtros en OpenCV.

- **Filtros de suavizado de una imagen:**

```
void GaussianBlur (Mat src, Mat dst, Size ksize,  
    double sigmaX, [double sigmaY=0, int tipoBorde=DEFAULT] );
```

- **src**: imagen de entrada. Puede ser de distintos tipos y número de canales.
- **dst**: imagen de salida. Debe ser del mismo tamaño y tipo que src.
- **ksize**: tamaño de la máscara suavizado (ancho por alto). Pueden ser distintas, pero deben ser positivas e impares.
- **sigmaX, sigmaY**: Indican la desviación típica de la máscara. Se puede poner 0 (para que sea según **ksize**).
- **tipoBorde**: indica el modo de operar en los píxeles de los bordes (BORDER_REPLICATE, BORDER_REFLECT, BORDER_WRAP, etc.). Lo mejor es dejar el valor por defecto.

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

89

A.3. Filtros en OpenCV.

- **Filtros de Sobel de una imagen:**

```
void Sobel (Mat src, Mat dst, int ddepth, int dx, int dy, int ksize=3,  
    double scale=1, double delta=0, int tipoBorde=DEFAULT ] );
```

- **src**: imagen de origen (puede tener distintas profundidades y número de canales). **dst**: imagen de destino (de igual tamaño y profundidad que **src**).
- **ddepth**: profundidad de la imagen de salida. Puede ser -1 (igual que la de entrada) o una profundidad mayor que la entrada (CV_16S, CV_32F...).
- **dx, dy**: orden de la derivada en X y en Y. Normalmt. usaremos (1,0) o (0,1).
- **ksize**: tamaño de la máscara de convolución: -1 (filtro de Scharr), 1 (resta simple), 3, 5 ó 7. Ver también la función **Scharr**.
- **scale, delta**: valor que se multiplica (**scale**) y que se suma (**delta**).
- **Ejemplo**. Aplicar un efecto de bajorrelieve a una imagen **img**.
Mat emboss; Sobel(img, emboss, -1, 1, 0, 3, 1, 128);
- **Ejemplo**. Calcular la magnitud del gradiente de la imagen **img**.

```
Mat dx, dy, gradiente;  
Sobel(img, dx, CV_32F, 1, 0, -1);  
Sobel(img, dy, CV_32F, 0, 1, -1);  
pow(dx, 2.0, dx);  
pow(dy, 2.0, dy);  
gradiente= dx+dy;  
sqrt(gradiente, gradiente);  
Mat imgrad;  
gradiente.convertTo(imgrad, CV_8U);
```

Procesamiento de Imágenes
Tema 3. Filtros y transformaciones locales.

90

A.3. Filtros en OpenCV.

- **Filtros de Laplace de una imagen:**

```
void Laplacian (Mat src, Mat dst, int ddepth, [ int ksize=1,  
              double scale=1, double delta=0, int tipoBorde=DEFAULT ] );
```

- **src**: imagen de origen. **dst**: imagen de destino.
- **ddepth**: profundidad deseada de la imagen de salida.
- **ksize**: tamaño de la máscara de convolución, debe ser positivo e impar.
- **scale, delta**: valor que se multiplica (**scale**) y que se suma (**delta**).
- Calcula la laplaciana de una imagen (suma de las 2ª derivadas en X e Y).
- **¡¡Mucho cuidado!!** La laplaciana puede tomar valores negativos: no convertir el resultado a 8U (los negativos se saturan a 0).
- Si se va a usar para un **perfilado**, el coeficiente que multiplica al resultado debe ser negativo, ya que usa máscaras “inversas” a las que hemos visto.

- **Ejemplo**: uso de la laplaciana para calcular un perfilado de **img**.

```
Mat lap, suma;  
Laplacian(img, lap, CV_16S);  
img.convertTo(img, CV_16S);  
suma= img - 0.4*lap;  
suma.convertTo(img, CV_8U);
```

A.3. Filtros en OpenCV.

- **Filtro de bordes de Canny:**

```
void Canny (Mat image, Mat edges, double threshold1,  
            double threshold2, [ int apertureSize=3, bool L2gradient=false ] );
```

- Ojo, es un filtro de bordes más avanzado que los otros. Usa filtros de Sobel y luego un algoritmo voraz para extraer los bordes más relevantes. Requiere imágenes de 1 solo canal y 8 bits, igual que **edges**.
- **threshold1, threshold2**: umbrales del algoritmo. Se refieren al valor mínimo de la magnitud del gradiente para ser considerado como un borde relevante.
- **apertureSize**: tamaño de la máscara de convolución (igual que Sobel).
- **Ejemplo**. Aplicación del operador de bordes de Canny, sobre **img**.

```
Mat gris, bordes;  
cvtColor(img, gris, CV_BGR2GRAY);  
Canny(gris, bordes, 100, 60);
```

A.3. Filtros en OpenCV.

- **Filtros lineales arbitrarios:** son los más flexibles. Nos definimos la máscara de convolución que queramos y la aplicamos sobre las imágenes.
- Para los filtros que están predefinidos (suavizados, bordes, etc.) no hace falta utilizar estas funciones (que, además, serán menos eficientes).
- Las **máscaras de convolución** se definen como matrices de tipo **Mat**, con 1 canal y profundidad 32F.
- La máscara de convolución se puede inicializar al crearla de la siguiente forma:
 - **Ejemplo 1.** Máscara de perfilado de la página 56.
Mat kernelPerfilado= (Mat_<float>(3, 3) << -1, -1, -1,
-1, 9, -1,
-1, -1, -1);
 - **Ejemplo 2.** Máscara derivativa en diagonal.
Mat kernelDerivada= (Mat_<float>(3, 3) << -2, -1, 0, -1, 0, 1, 0, 1, 2);

A.3. Filtros en OpenCV.

- **Aplicar una máscara de convolución arbitraria en OpenCV:**
void **filter2D** (Mat src, Mat dst, int ddepth, Mat kernel,
[Point ancla=Point(-1,-1), double delta=0, int borde=DEFAULT]);
 - **kernel** es una matriz de 1 canal y 32F, indica los coeficientes de la máscara de convolución.
 - **ancla** es el punto de ancla (por defecto, el centro de la máscara).
 - **delta** es el valor que se suma después de aplicar la convolución.
 - **Ejemplo.** Aplicar a una imagen img el perfilado y la derivada.
Mat salida1, salida2;
Mat kernelPerfilado= (Mat_<float>(3, 3) << -1, -1, -1,
-1, 9, -1,
-1, -1, -1);
filter2D(img, salida1, CV_8U, kernelPerfilado);
Mat kernelDerivada= (Mat_<float>(3, 3) << -2, -1, 0, -1, 0, 1, 0, 1, 2);
filter2D(img, salida2, CV_8U, kernelDerivada, Point(-1,-1), 128);

A.3. Filtros en OpenCV.

- **Filtros no lineales de máximo, mínimo:** no existen en OpenCV, sino que deben hacerse utilizando las operaciones morfológicas **dilate** (máximo) y **erode** (mínimo).
- Ambas operaciones reciben como parámetro un elemento estructurante, que es una matriz de tipo **Mat**.
- **Implementación** de las operaciones de máximo y mínimo local:

```
void MinLocal (Mat entrada, Mat &salida, int ancho, int alto)
{
    erode(entrada, salida, Mat::ones(alto, ancho, CV_8U),
        Point(-1,-1), 1, BORDER_REPLICATE);
}
```

```
void MaxLocal (Mat entrada, Mat &salida, int ancho, int alto)
{
    dilate(entrada, salida, Mat::ones(alto, ancho, CV_8U),
        Point(-1,-1), 1, BORDER_REPLICATE);
}
```

A.3. Filtros en OpenCV.

- **Filtro de mediana** de tamaño ksize:
void **medianBlur** (Mat src, Mat dst, int ksize);
 - El tamaño sobre el que se aplica es siempre el mismo en horizontal y en vertical (**ksize**).
- **Filtros de morfología matemática:** el manejo es parecido a las convoluciones arbitrarias. 1º: definir un **elemento estructurante**. 2º: aplicarlo sobre las imágenes con operaciones de erosión, dilatación, apertura o cierre.
- Normalmente las imágenes será de 1 canal y tipo 8U. No obstante, pueden tener otras profundidades y número de canales (se usa MAX y MIN, en lugar de OR y AND).
- El elemento estructurante es de tipo **Mat**. Si se pone la constante **Mat()**, entonces se usa el elemento por defecto, un rectángulo de 3x3.

A.3. Filtros en OpenCV.

- **Aplicar erosión morfológica a una imagen:**

```
void erode (Mat src, Mat dst, Mat kernel, [ Point ancla=Point(-1,-1),  
int iteraciones=1, int borde=BORDER_CONSTANT,  
Scalar& valorBorde=morphologyDefaultBorderValue() ] );
```

- Aplica uno o varios pasos de erosión, según el parámetro **iteraciones**.
- Si el elemento **kernel** es Mat() se usa un rectángulo de 3x3.
- La forma de tratar el borde se define en **borde** y **valorBorde**.

- **Aplicar dilatación morfológica a una imagen:**

```
void dilate (Mat src, Mat dst, Mat kernel, [ Point ancla=Point(-1,-1),  
int iteraciones=1, int borde=BORDER_CONSTANT,  
Scalar& valorBorde=morphologyDefaultBorderValue() ] );
```

- Aplica uno o varios pasos de dilatación, según **iteraciones**.
- Si el elemento **kernel** es Mat() se usa un rectángulo de 3x3.
- La forma de tratar el borde se define en **borde** y **valorBorde**.

- **Ejemplo.** Aplicar una dilatación de 5x5, con elemento en diagonal.

```
dilate(img, salida, Mat::eye(5,5,CV_8U));
```

A.3. Filtros en OpenCV.

- **Aplicar operaciones morfológicas compuestas:**

```
void morphologyEx (Mat src, Mat dst, int op, Mat kernel,  
[ Point ancla=Point(-1,-1), int iterac=1, int borde=BORDER_CONSTANT,  
Scalar& valorBorde=morphologyDefaultBorderValue() ] );
```

- Permite aplicar una operación morfológica compuesta por otras elementales, erosiones, dilataciones y diferencias.
- **kernel** es el elem. estructurante, **iterac** es el número de iteraciones.
- El parámetro **op** indica el tipo de operación: MORPH_OPEN, MORPH_CLOSE, MORPH_GRADIENT, MORPH_TOPHAT, MORPH_BLACKHAT.

- **Ejemplo.** Los dos siguientes códigos deberían dar la misma salida:

```
a) morphologyEx(img, res, MORPH_OPEN, Mat());
```

```
b) erode(img, img, Mat());  
dilate(img, res, Mat());
```

A.3. Filtros en OpenCV.

- **Ejemplo 1.** Aplicar un ajuste (o estiramiento) local del histograma a la imagen **img**, con **ancho** dado. El resultado se almacena en **res**.

```
Mat mini, maxi, res;
int tam= 2*ancho+1; // tamaño de vecindad local
MinLocal(img, mini, tam, tam);
MaxLocal(img, maxi, tam, tam);
res= img-mini;
maxi= maxi-mini;
divide(res, maxi, res, 255.0);
```

A.3. Filtros en OpenCV.

- **Ejemplo 2.** Efecto de transición entre dos imágenes (que deben ser de igual tamaño), a través de un suavizado intermedio:

```
QString nombre1= QFileDialog::getOpenFileName();
QString nombre2= QFileDialog::getOpenFileName();
Mat img1= imread(nombre1.toStdString());
Mat img2= imread(nombre2.toStdString());
if (img1.empty() || img2.empty()) return;
namedWindow("Transicion", 0);
Mat int1, int2, res;
for (int i= 0; i<100; i++) {
    blur(img1, int1, Size(i*4+1, 1));
    blur(img2, int2, Size((99-i)*4+1, 1));
    addWeighted(int1, 1-i/100.0, int2, i/100.0, 0, res);
    imshow("Transicion", res);
    waitKey(1);
}
```