UNIVERSITÀ DI PARMA

Image Filtering

Image processing reduces image information. A processed image can often seem more eye appealing, but is always poorer than original one

Summary

UNIVERSITÀ DI PARMA

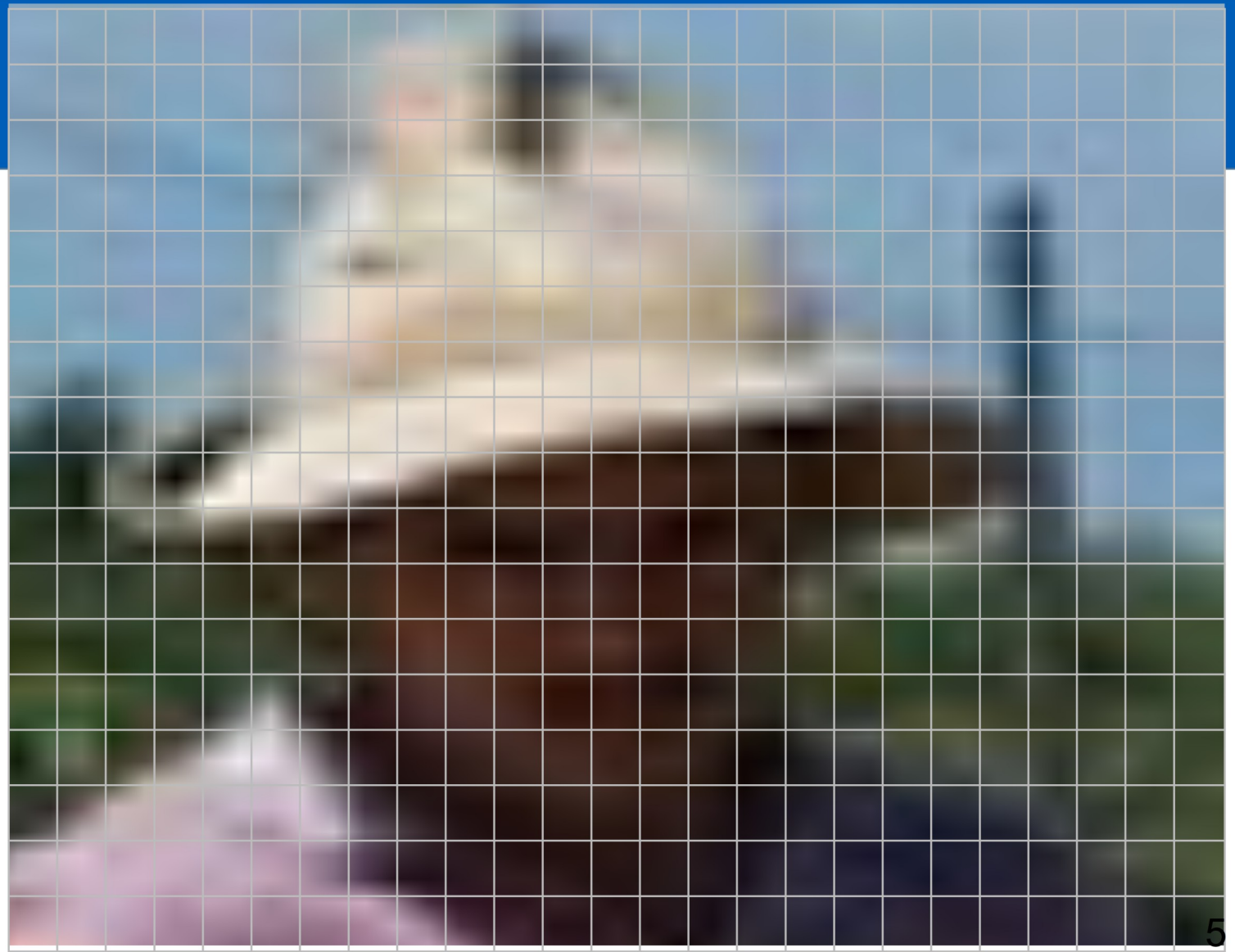
- What is an Image?
- Image Filtering
- Local Operations
  - Image Convolution
  - Linear filters
    - Smoothing
    - Gradient
  - Padding
  - Stride
  - Non linear filters
  - Bilateral filtering
  - Integral Image

What are images?

What are images?

$$P = f(x, y)$$

$$f : R^2 \Rightarrow R$$



$$P = f(x, y)$$

$$f : R^2 \Rightarrow R$$

1. Signal is **sampled** to obtain a function (r,c)
2. Result is also **quantized** given we need a discrete function
3. One or more **channels** for each pixel

R(r,c), G(r,c), B(r,c)

Luminance(r,c)

Image filtering

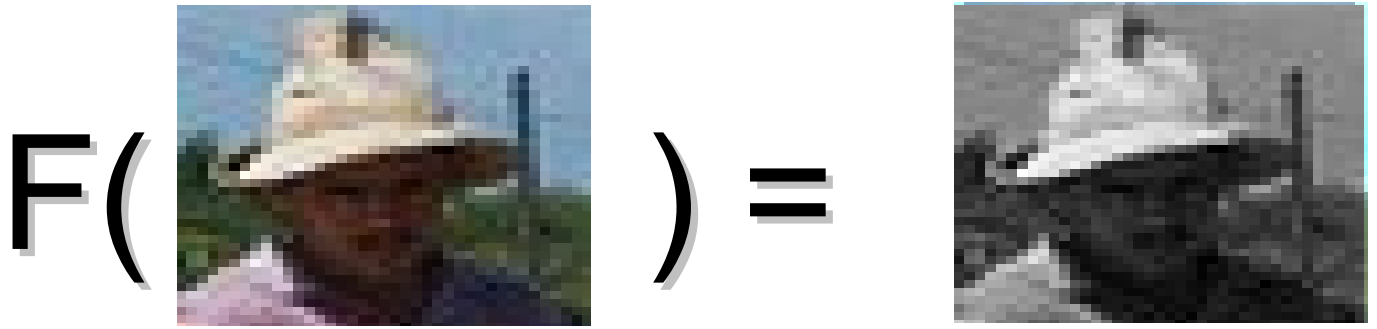


Image filtering

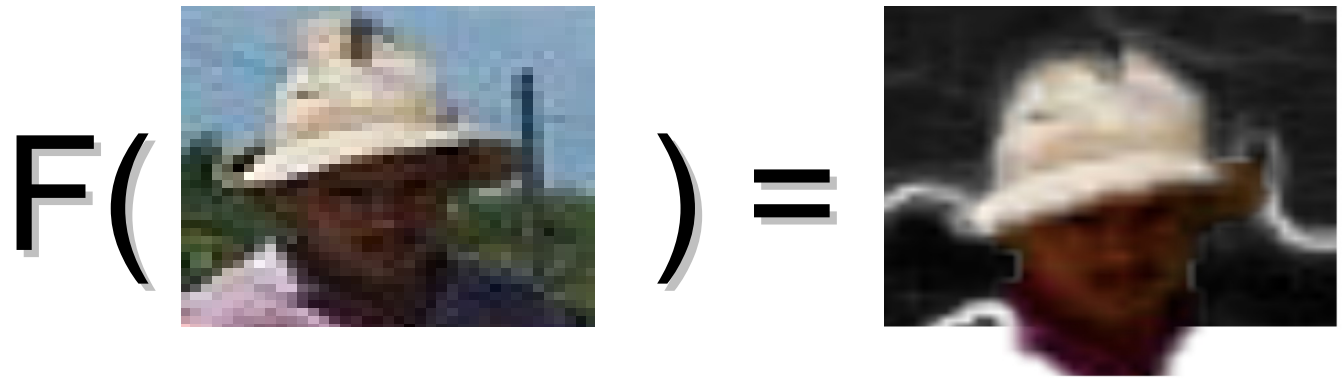


Image filtering

F() =

0.1
0
0.8
0.9
0.9
0.9
0.2
0.4
0.3
0.6
0
0
0.1
0.5
0.9
0.9
0.2
0.4
0.3
0.6
0
0
0.1
0.9
0.9
0.2
0.4
0.3

Image filtering

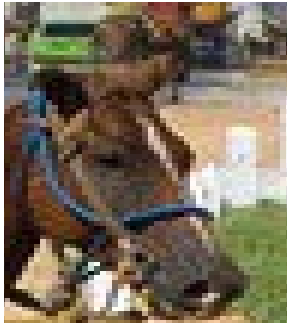
F( , ) = 

Image Filtering

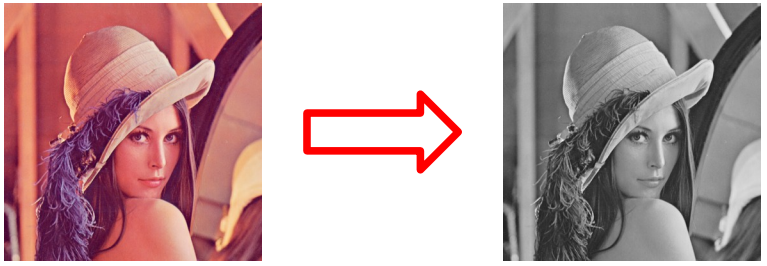
UNIVERSITÀ DI PARMA

- Filters
  - Mathematical functions that operate on the pixels
  - Often Image ⟶ Image
- Different classes
  - **Point operations:** result depends only on each pixel value
  - **Local operations:** result depends on each pixel + neighborhood values
  - **Global operations:** result depends on the values of all imaghe pixels

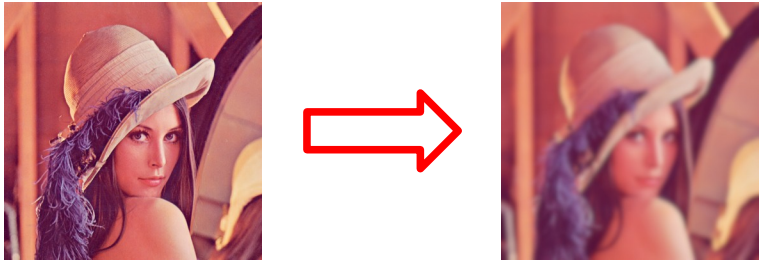
Image Filtering Examples

UNIVERSITÀ DI PARMA

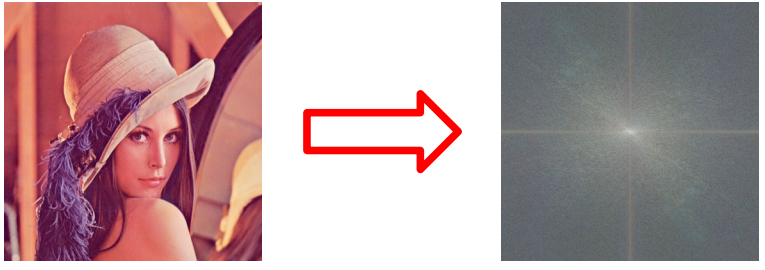
- Point operations:
  - Color to gray levels



- Local operations:
  - Running average (blur)



- Global operations:
  - Fourier transform



UNIVERSITÀ DI PARMA

Image Filtering
Local & Linear operations
Convolution

Local Operations

F() = 

- Result depends only on
  - Pixel value
  - Pixel Neighborhood

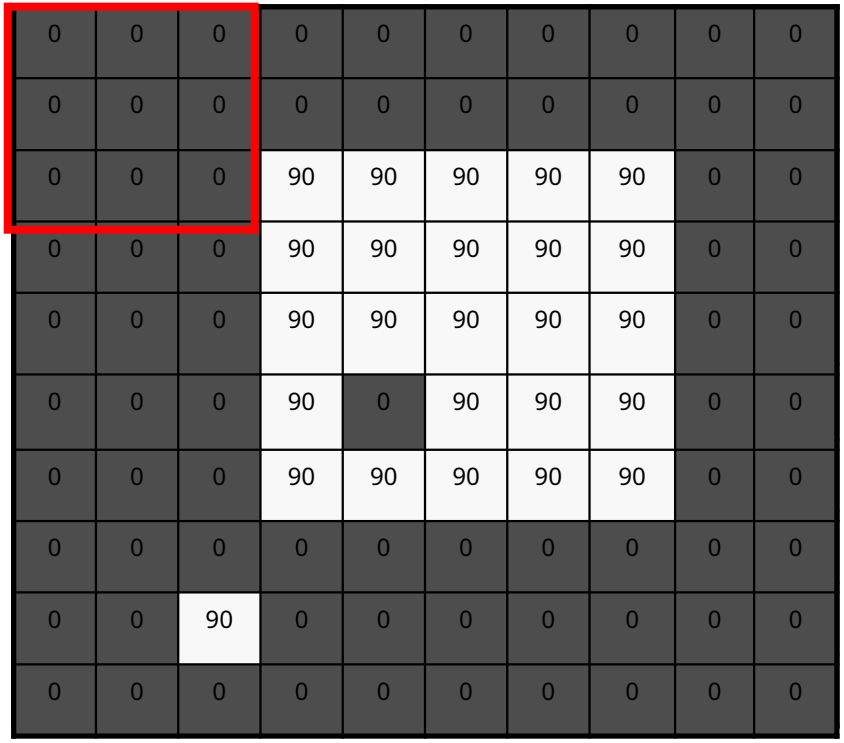
Image Convolution – Linear Filters

UNIVERSITÀ DI PARMA

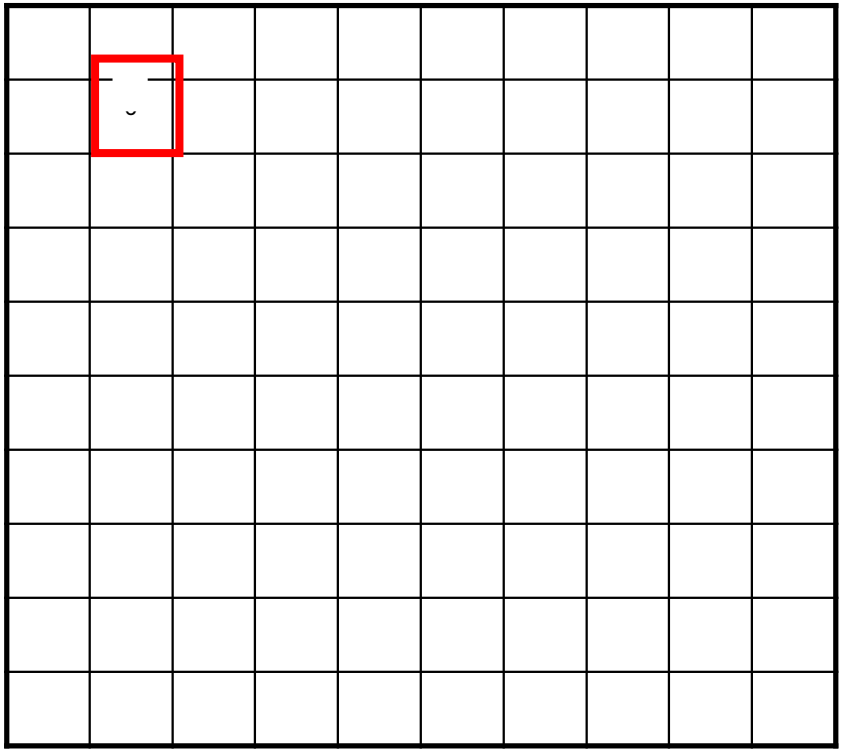
- Image convolution is the main operation used for local image filtering
  - From f[.,.] → h[.,.]
- g[k,l] is the **Kernel** or Convolution Matrix and represents the pixel neighborhood weight
  - Kernel size
- **Linear operation**
- Symbol: *
  - h=g*f

$$h[m,n]=\sum_{k,l} g[k,l]f[m+k,n+l]$$

Local Operations



$f[.,.]$



$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

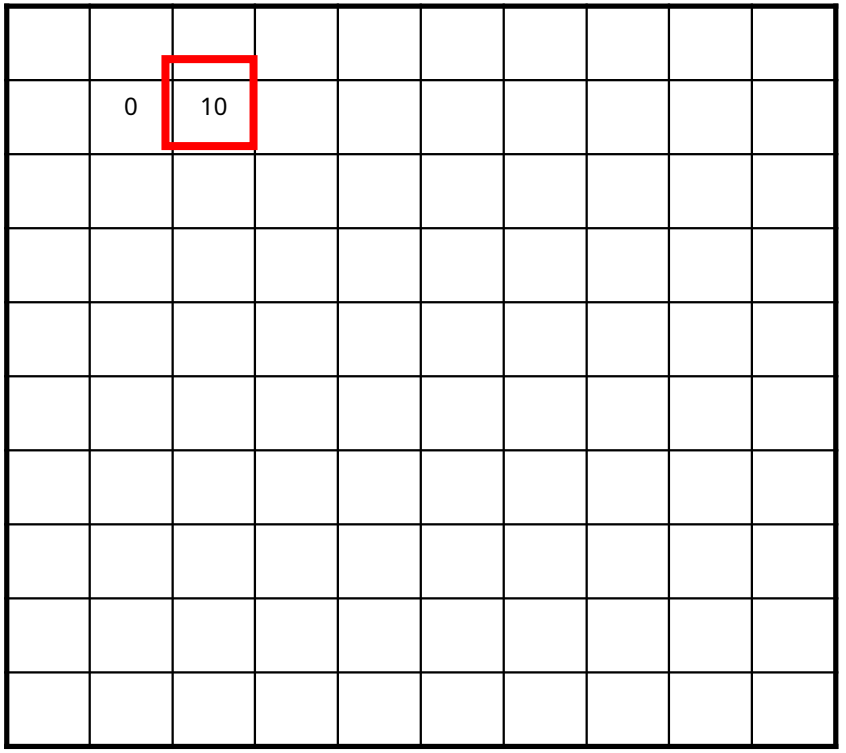
$g[\cdot,\cdot]$

Credit: S. Seitz

Local Operations

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$f[.,.]$



$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

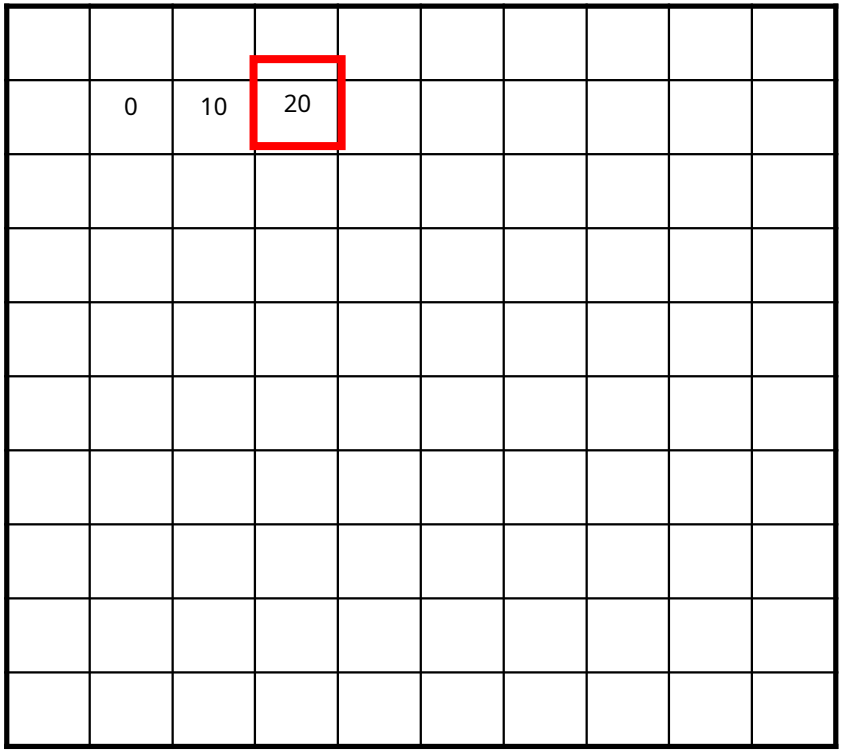
$g[\cdot,\cdot]$

Credit: S. Seitz

Local Operations

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$f[.,.]$



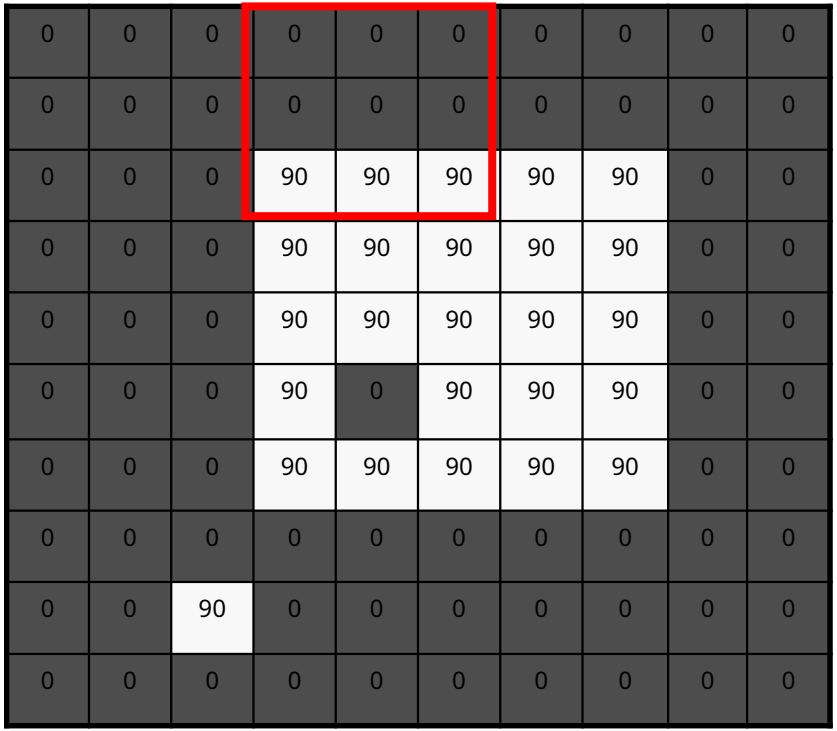
$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

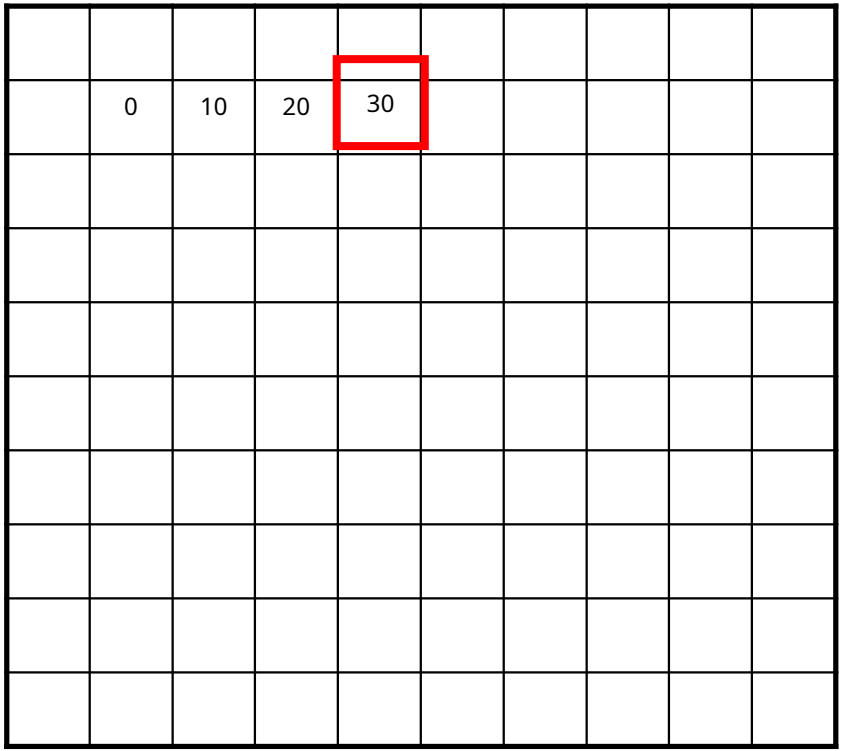
$g[\cdot,\cdot]$

Credit: S. Seitz

Local Operations



$f[.,.]$



$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$g[\cdot,\cdot]$

Credit: S. Seitz

Local Operations

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$f[.,.]$

	0	10	20	30	30				

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

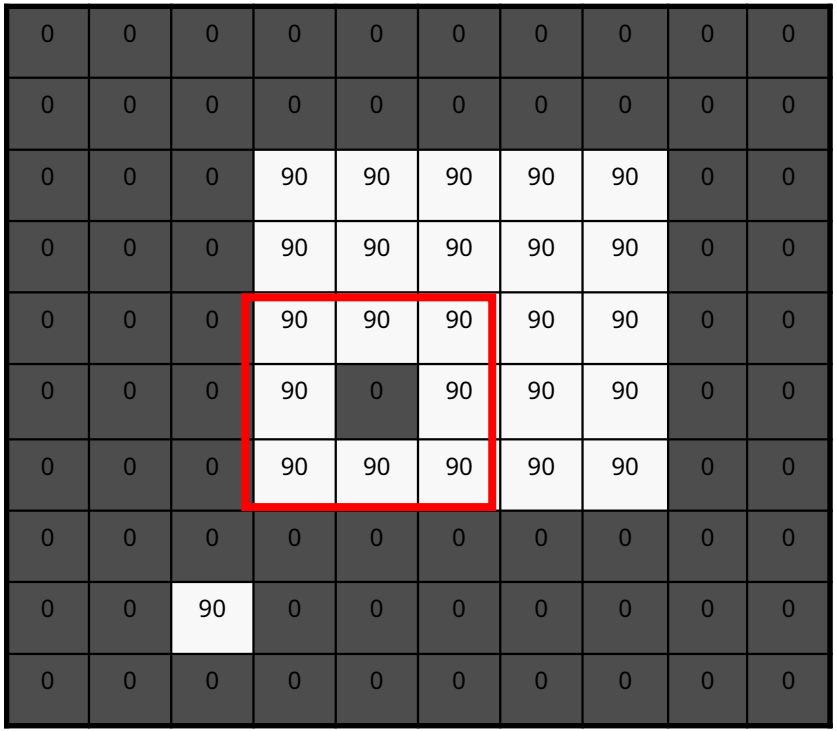
$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

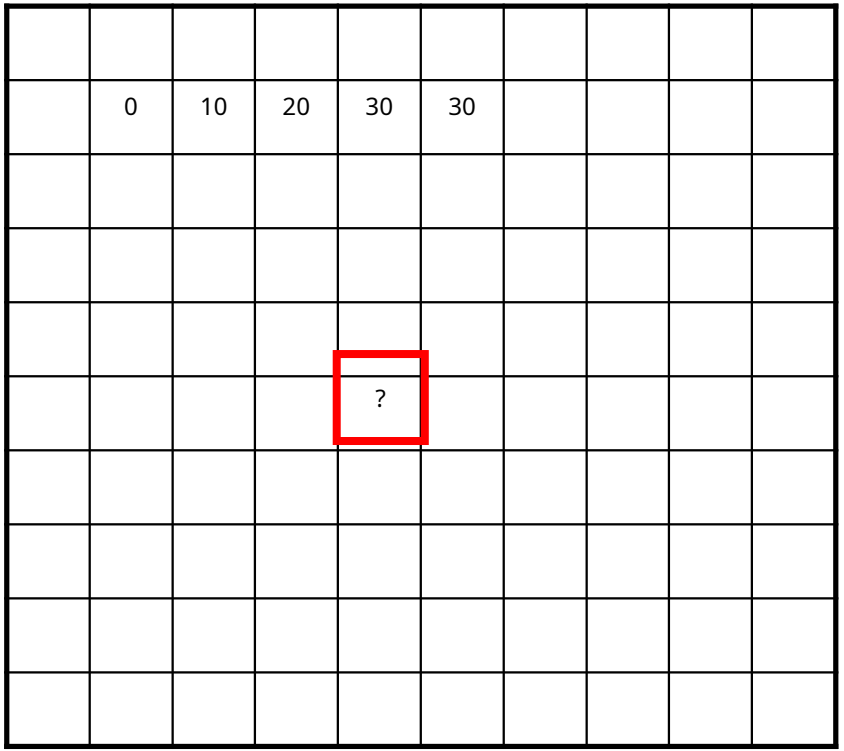
$g[\cdot,\cdot]$

Credit: S. Seitz

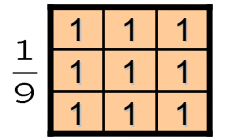
Local Operations



$f[.,.]$



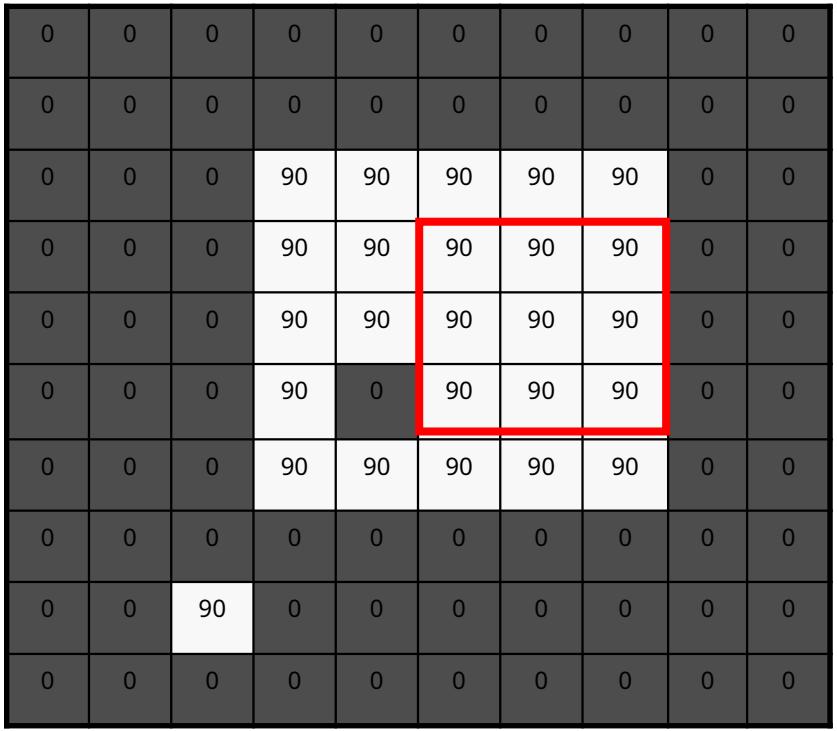
$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$



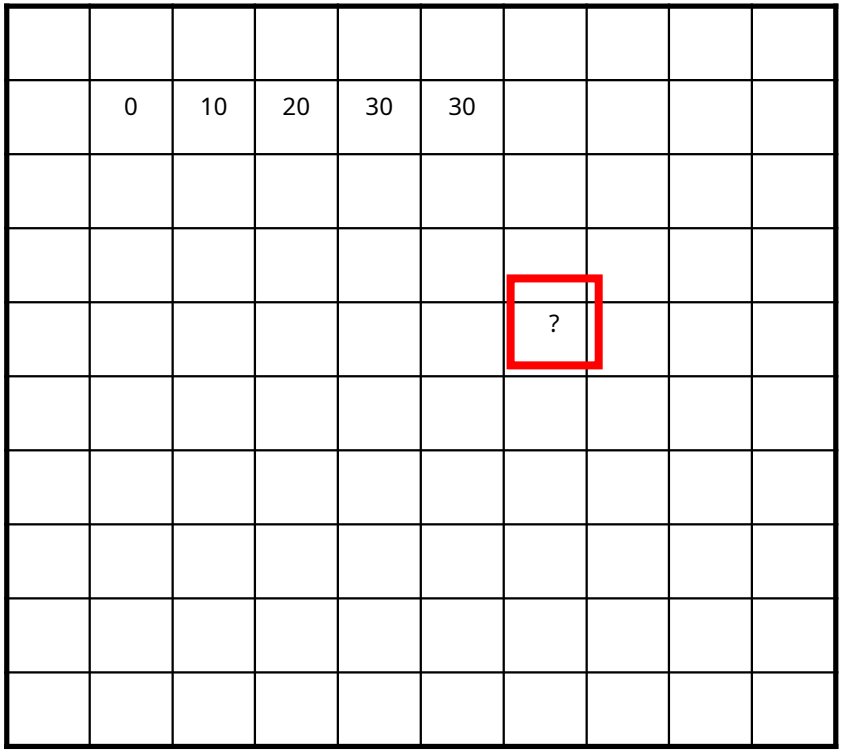
$g[\cdot,\cdot]$

Credit: S. Seitz

Local Operations



$f[.,.]$



$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$g[\cdot,\cdot]$

Credit: S. Seitz

Local Operations

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$f[.,.]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$g[\cdot,\cdot]$

Credit: S. Seitz

Local operations

UNIVERSITÀ DI PARMA

- What is the result of such filtering?
  - Neighborhood average
  - Low pass filter
- Basically sharp transitions are reduced
  - blurring
- **Mean/Box filter smoothing**

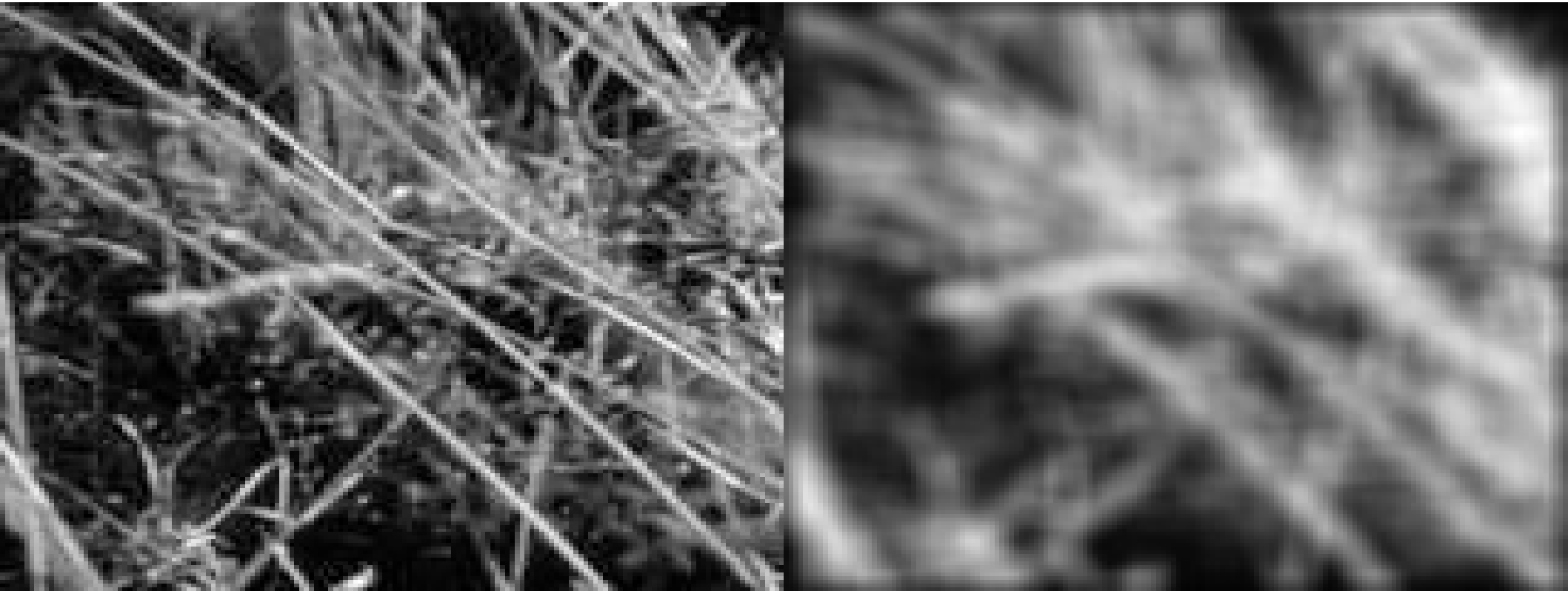
$g[\cdot, \cdot]$

$$\frac{1}{9}$$

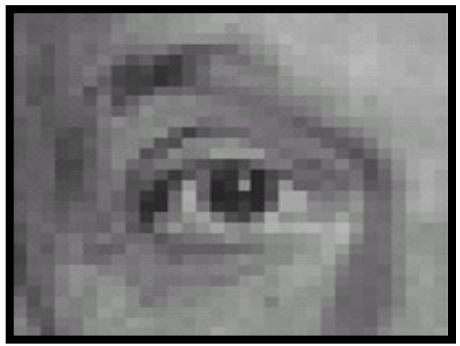
1	1	1
1	1	1
1	1	1

Box Filter Smoothing

UNIVERSITÀ DI PARMA



Other examples



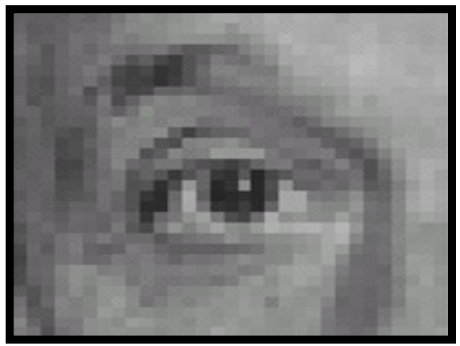
Original

0	0	0
0	1	0
0	0	0

?

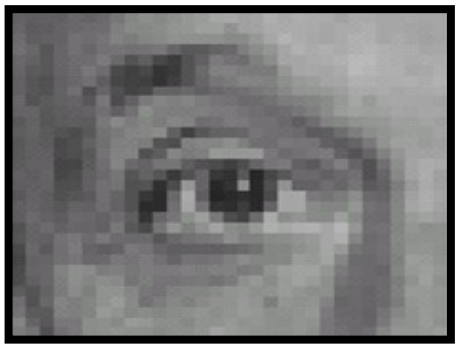
Slide credit: David Lowe (UBC)

Other examples



Original

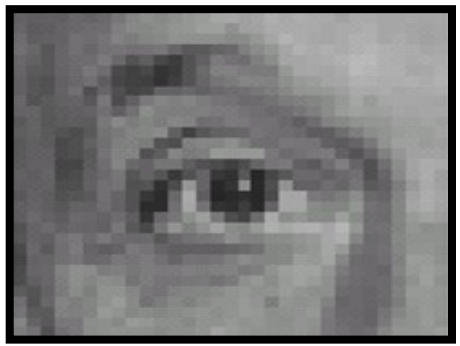
0	0	0
0	1	0
0	0	0



Filtered
(no change)

Slide credit: David Lowe (UBC)

Other examples



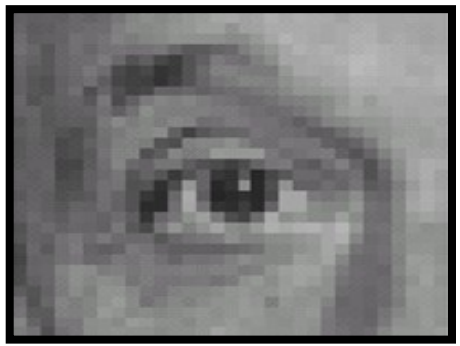
Original

0	0	0
0	0	1
0	0	0

?

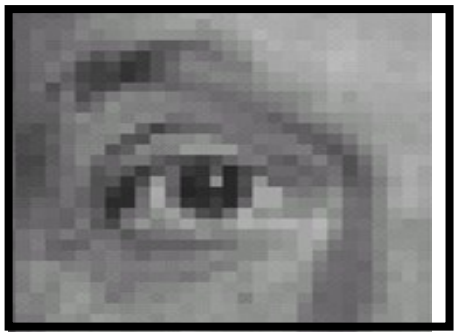
Slide credit: David Lowe (UBC)

Other examples



Original

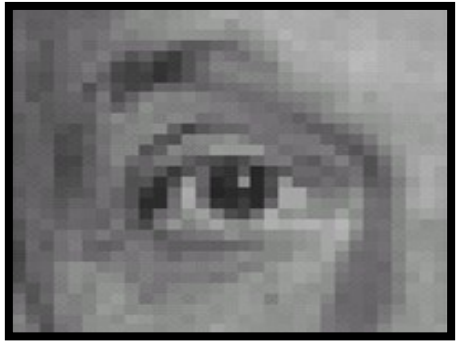
0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Slide credit: David Lowe (UBC)

Other examples

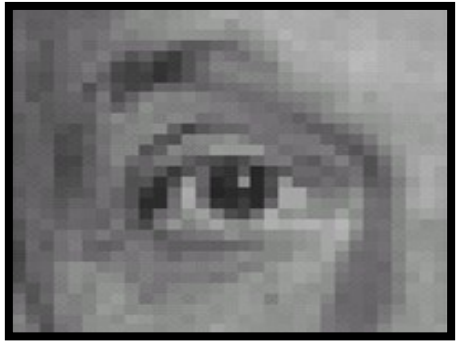


Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = ?$$

Slide credit: David Lowe (UBC)

Other examples



Original

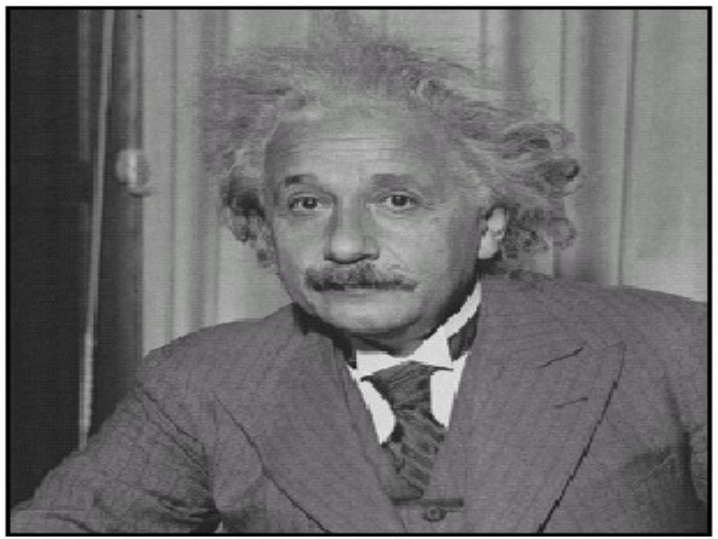
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter

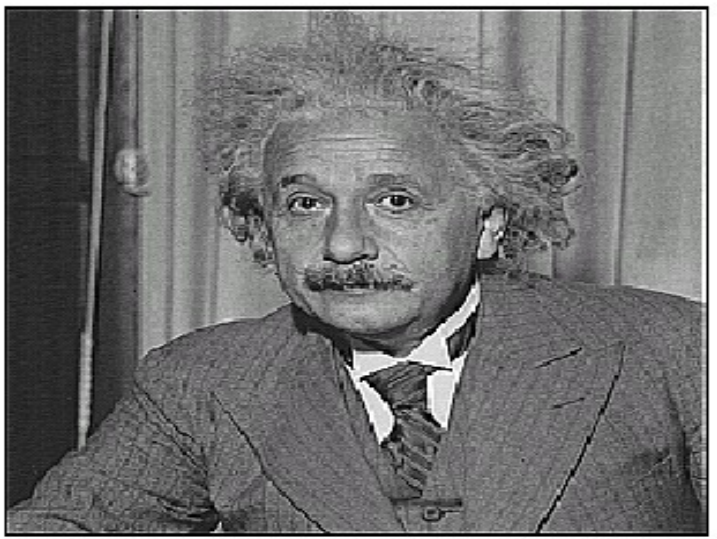
- Accentuates differences with local average
- High pass filtering

Slide credit: David Lowe (UBC)

Other examples



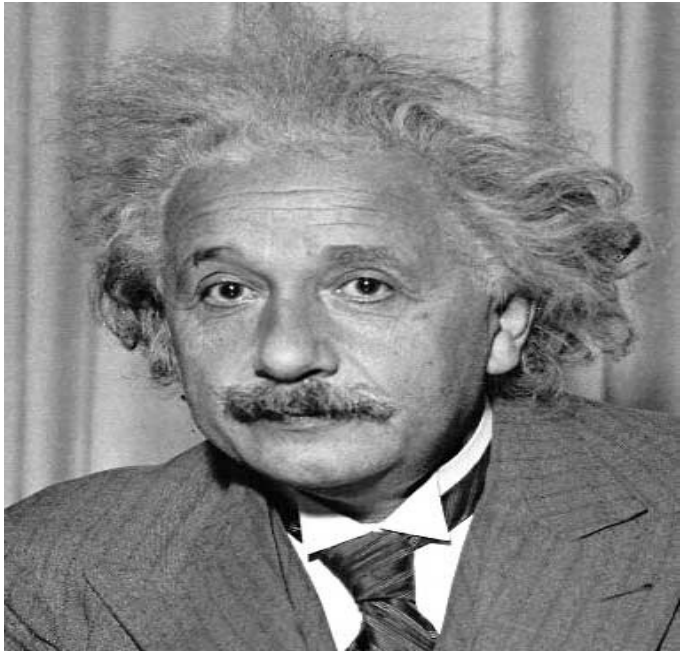
before



after

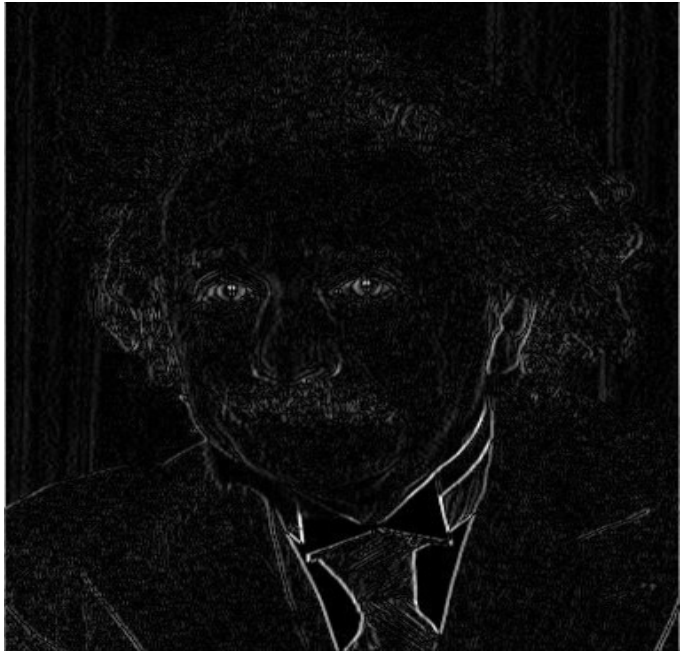
Slide credit: David Lowe (UBC)

Sobel



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

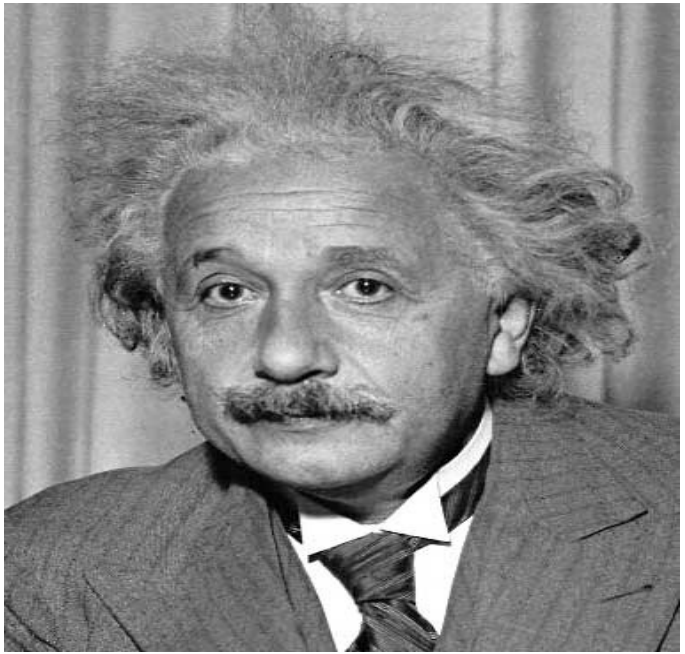
Vertical Sobel



Vertical Edge (absolute value)

Slide credit: David Lowe (UBC)

Sobel



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

Horizontal Sobel

Horizontal Edge (absolute value)

Slide credit: David Lowe (UBC)

Sobel

Horizontal Gradient

0	0	0
-1	0	1
0	0	0

or

-1	0	1

Vertical Gradient

0	1	0
0	0	0
0	-1	0

or

-1
0
1

Slide credit: David Lowe (UBC)

Negative values

UNIVERSITÀ DI PARMA

- We can have negative values
- Generally pixel channels are unsigned
- How to handle them?
  - Clamping: put negative values as 0
  - Abs: consider absolute values only
  - Shift: add a value (may reduce interval)
  - Use them: adapt internal representation to handle them

Optimization trick

UNIVERSITÀ DI PARMA

- **Some** kernel filters are **separable**
- n × n kernel in a single convolution
  - O(M×M×n×n)
- 1 × n and n × 1 convolutions
  - O(M×M×(n+n))
- Mean filtering example:

$$\frac{1}{3}\begin{bmatrix}1\\1\\1\end{bmatrix} * \frac{1}{3}\begin{bmatrix}1 & 1 & 1\end{bmatrix} = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$$

UNIVERSITÀ DI PARMA

Image Filtering
main local operations

SMOOTHING

UNIVERSITÀ DI PARMA

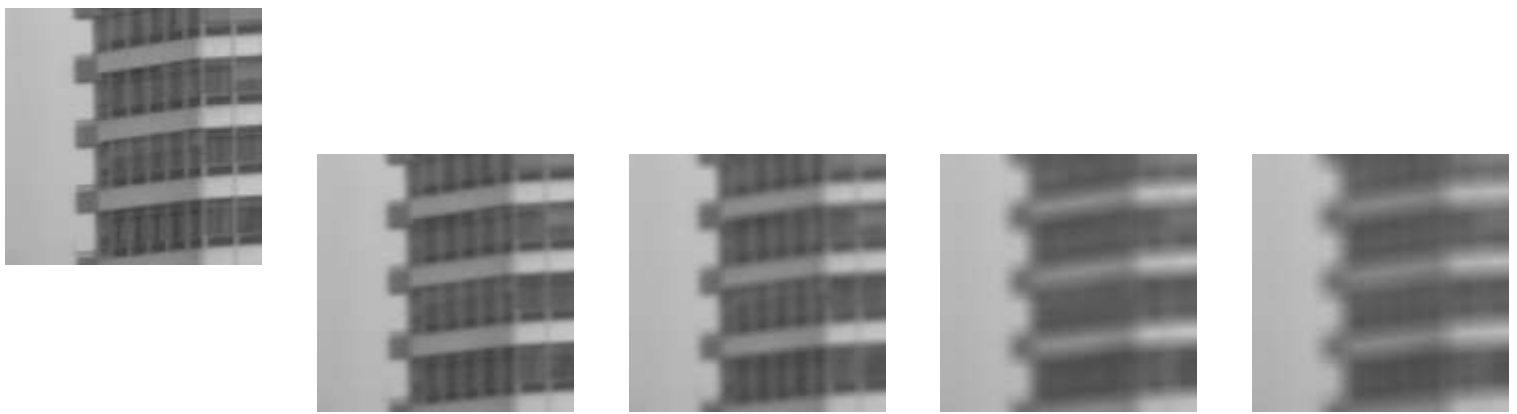
- Image can be affected by noise
  - Sensor
  - Transmission
  - Lossy compressions
  - Filtering can enhance noise (derivative operations)
- Different kind of noises
  - White noise
  - Salt&Pepper
  - …
- Smoothing is often used to “reduce” it

SMOOTHING: Mean Filtering

UNIVERSITÀ DI PARMA

- Averaging neighborhood can smooth images
- The bigger the kernel size, the augmented the smooth effect
  - Balancing act
  - Examples with 3x3, 5x5, 9x9, and 11x11 kernels

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



SMOOTHING: Mean Filtering

UNIVERSITÀ DI PARMA

- Pros
  - Easy implementation
- Cons:
  - Weight of each pixel is the same
  - Not good for specific noise effects
  - Introduce high frequency artifacts

SMOOTHING: Salt & Pepper noise

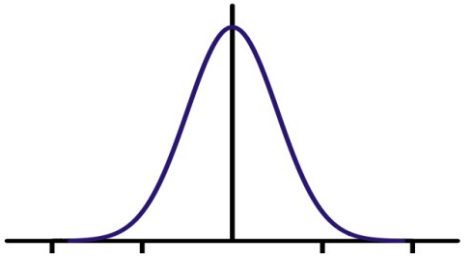
UNIVERSITÀ DI PARMA

SMOOTHING: Gaussian filtering

UNIVERSITÀ DI PARMA

- Mean filtering kernel use the same weight for each pixel
  - But the distance is different
- Gaussian filtering gives more weight to the central pixel and:
  - The farther away the neighbors, the smaller their contribution
  - A gaussian is used

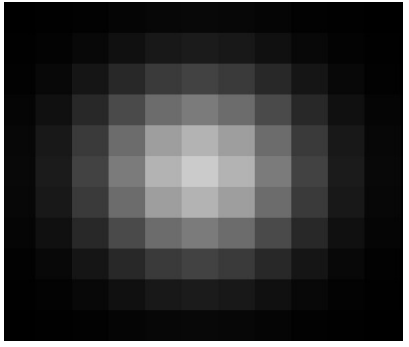
SMOOTHING: Gaussian filtering



$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

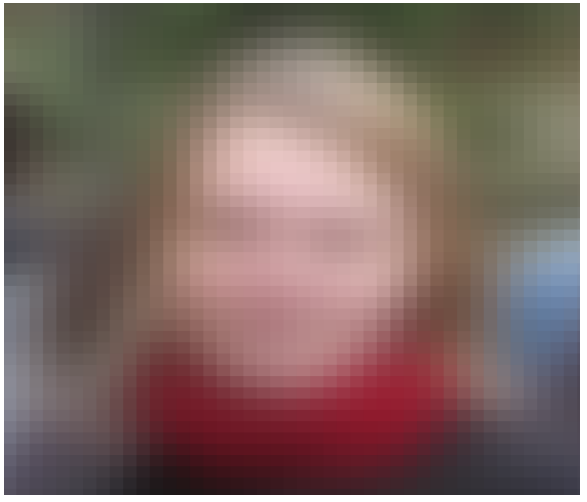
Input image f

*



Filter h

=

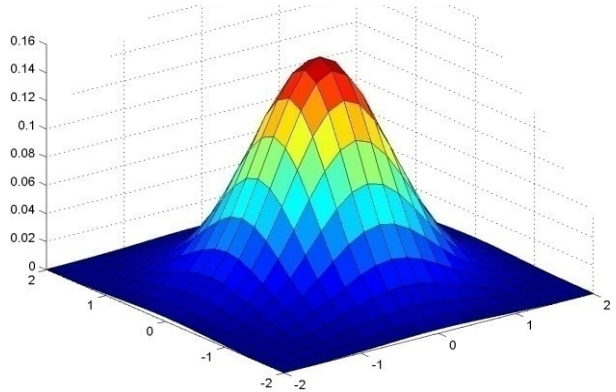


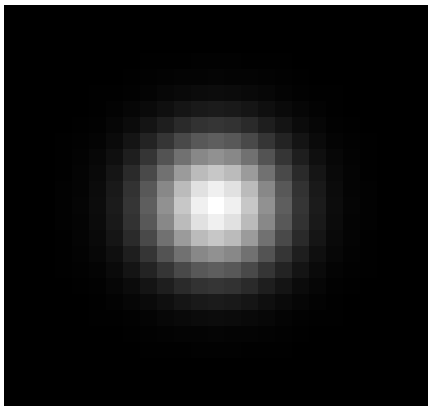
Output image g

SMOOTHING: Gaussian filtering

UNIVERSITÀ DI PARMA

Spatially-weighted average





0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

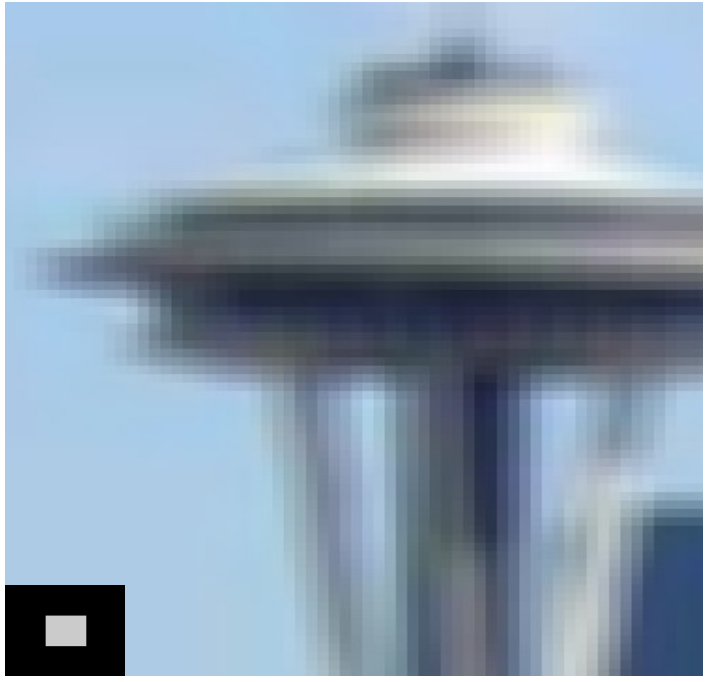
5 x 5, 📅 = 1

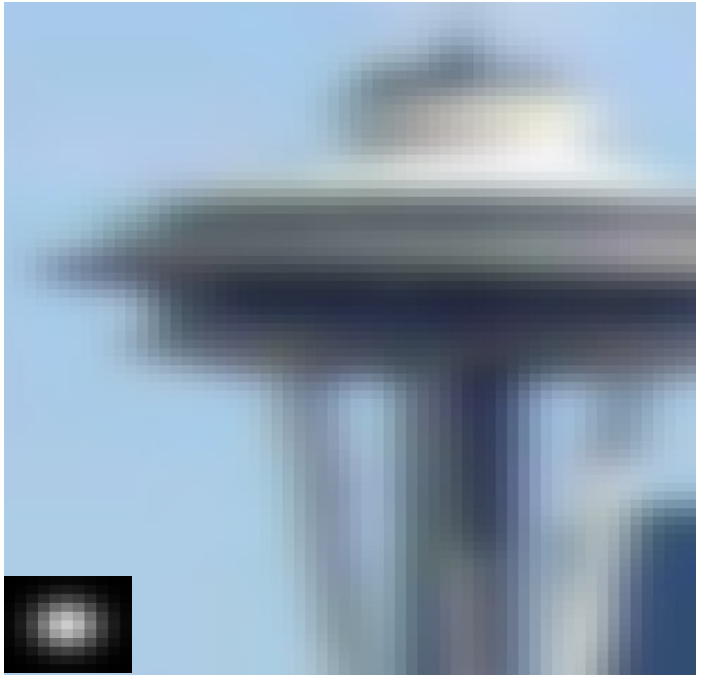
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Slide credit: Christopher Rasmussen

SMOOTHING: Gaussian vs Mean

UNIVERSITÀ DI PARMA





SMOOTHING: Gaussian vs Mean

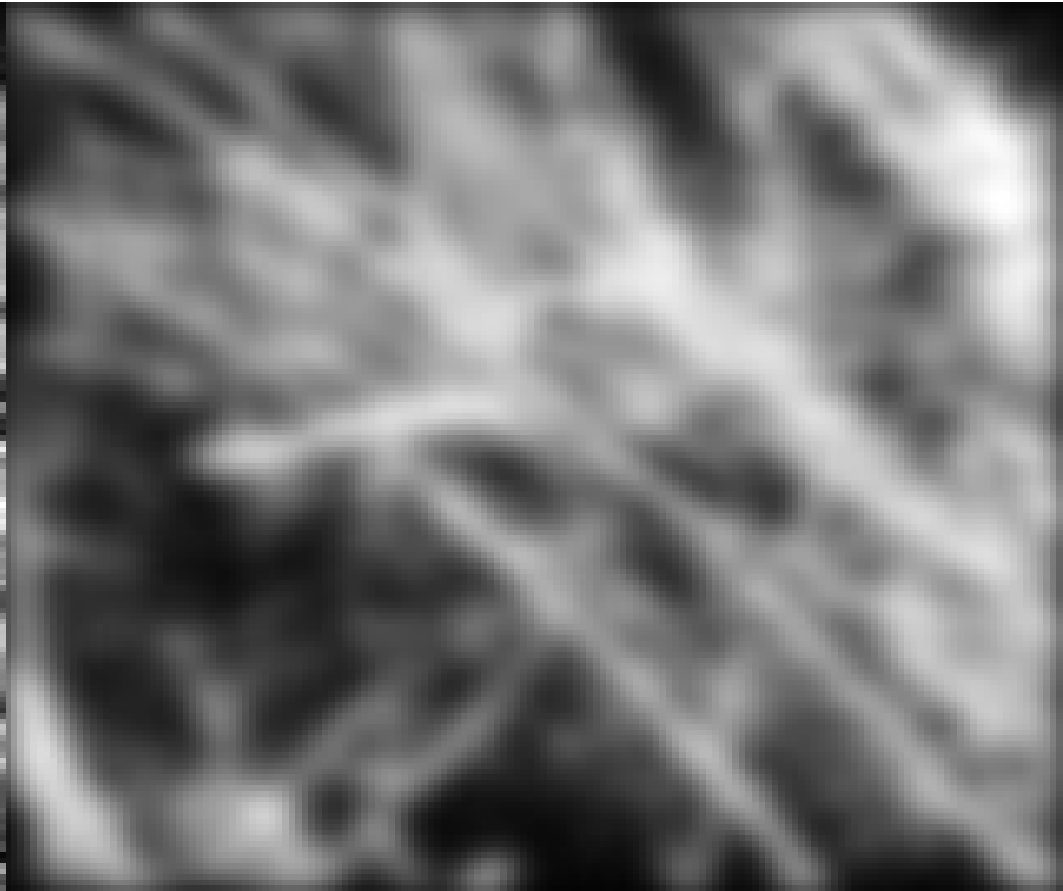
Slide credit: Robert Collins

SMOOTHING: mean filtering artifacts

UNIVERSITÀ DI PARMA

SMOOTHING: Gaussian

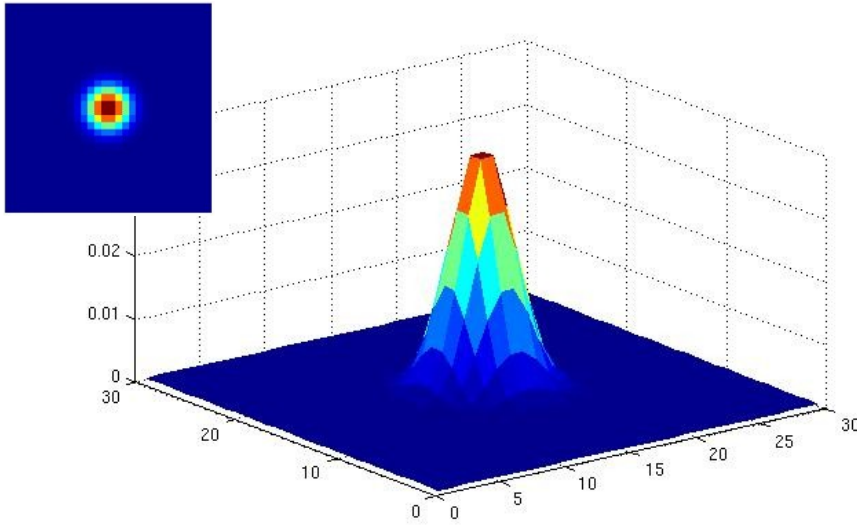
UNIVERSITÀ
DI PARMA

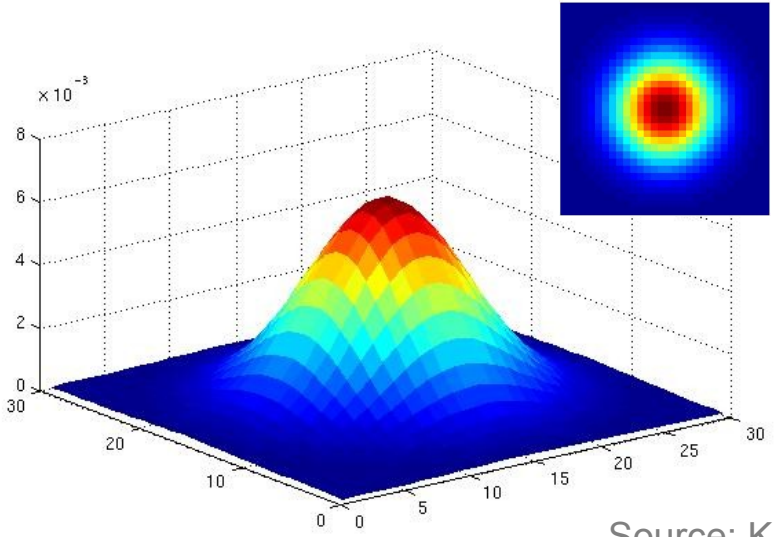


SMOOTHING: Gaussian

UNIVERSITÀ DI PARMA

What parameters matter here?
Variance of Gaussian: determines extent of smoothing



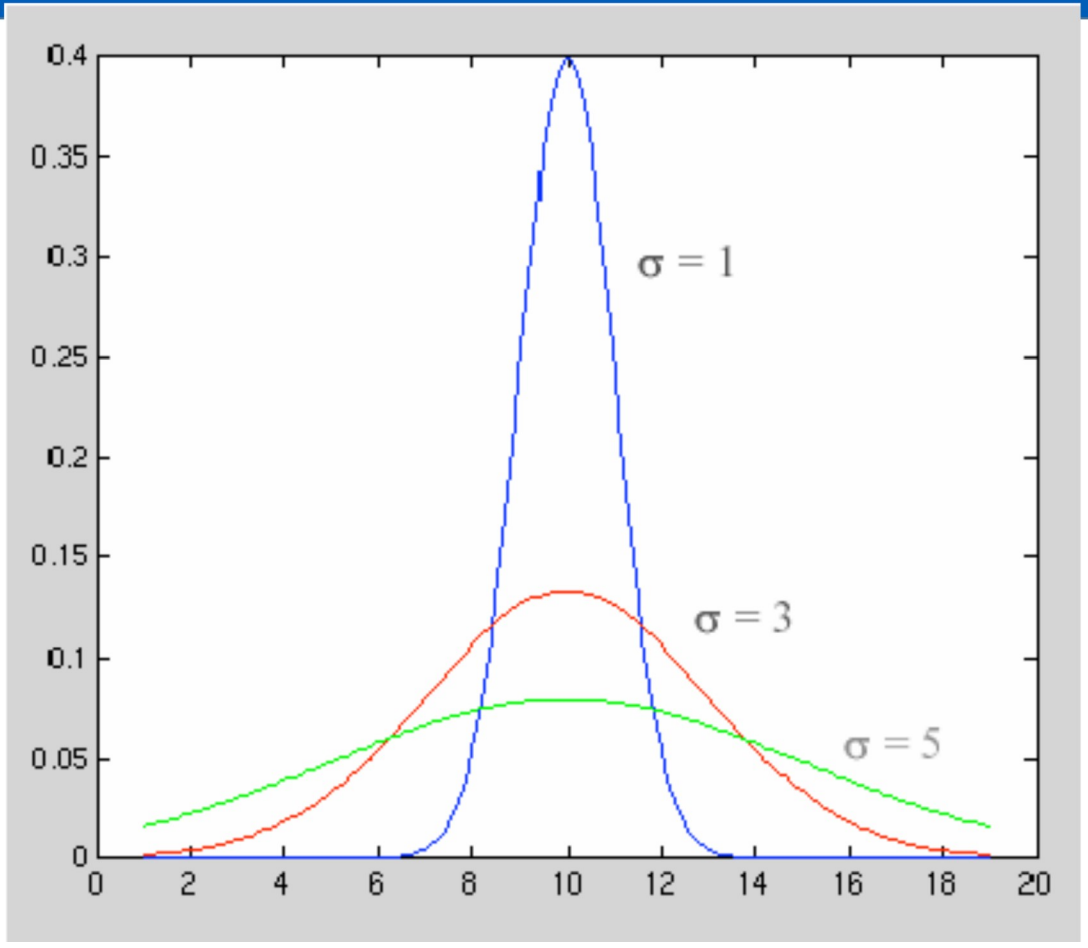


Source: K. Grauman

SMOOTHING: Gaussian

What parameters matter here?

Variance of Gaussian: determines extent of smoothing



SMOOTHING: Gaussian

UNIVERSITÀ DI PARMA

Parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.

...

Source: K. Grauman

SMOOTHING: efficient Gaussian

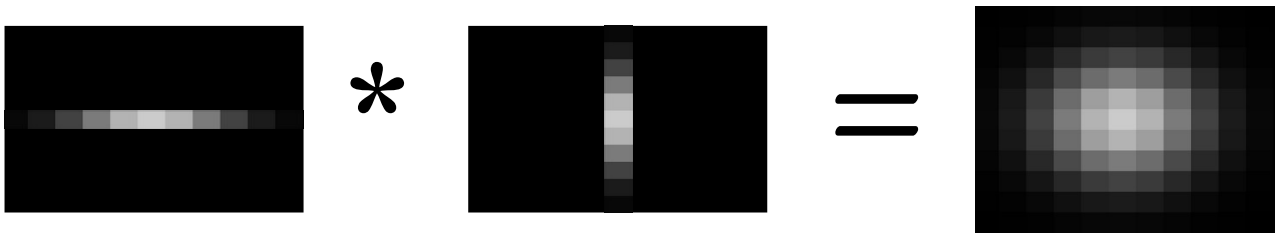
UNIVERSITÀ DI PARMA

$$\mathcal{G}_\sigma = \mathcal{G}_\sigma^x * \mathcal{G}_\sigma^y$$

$$\mathcal{G}_\sigma^x(x,y) = \frac{1}{Z} e^{\frac{-(x^2)}{2\sigma^2}}$$

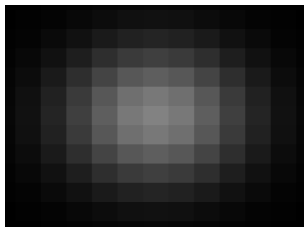
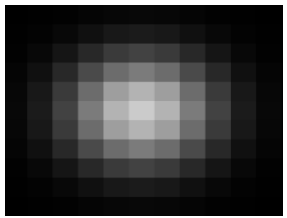
$$\mathcal{G}_\sigma^y(x,y) = \frac{1}{Z} e^{\frac{-(y^2)}{2\sigma^2}}$$

Gaussian filter is **separable**: compute in horizontal direction, followed by the vertical direction.



Faster!

SMOOTHING: efficient Gaussian

$$* \quad = \quad ?$$

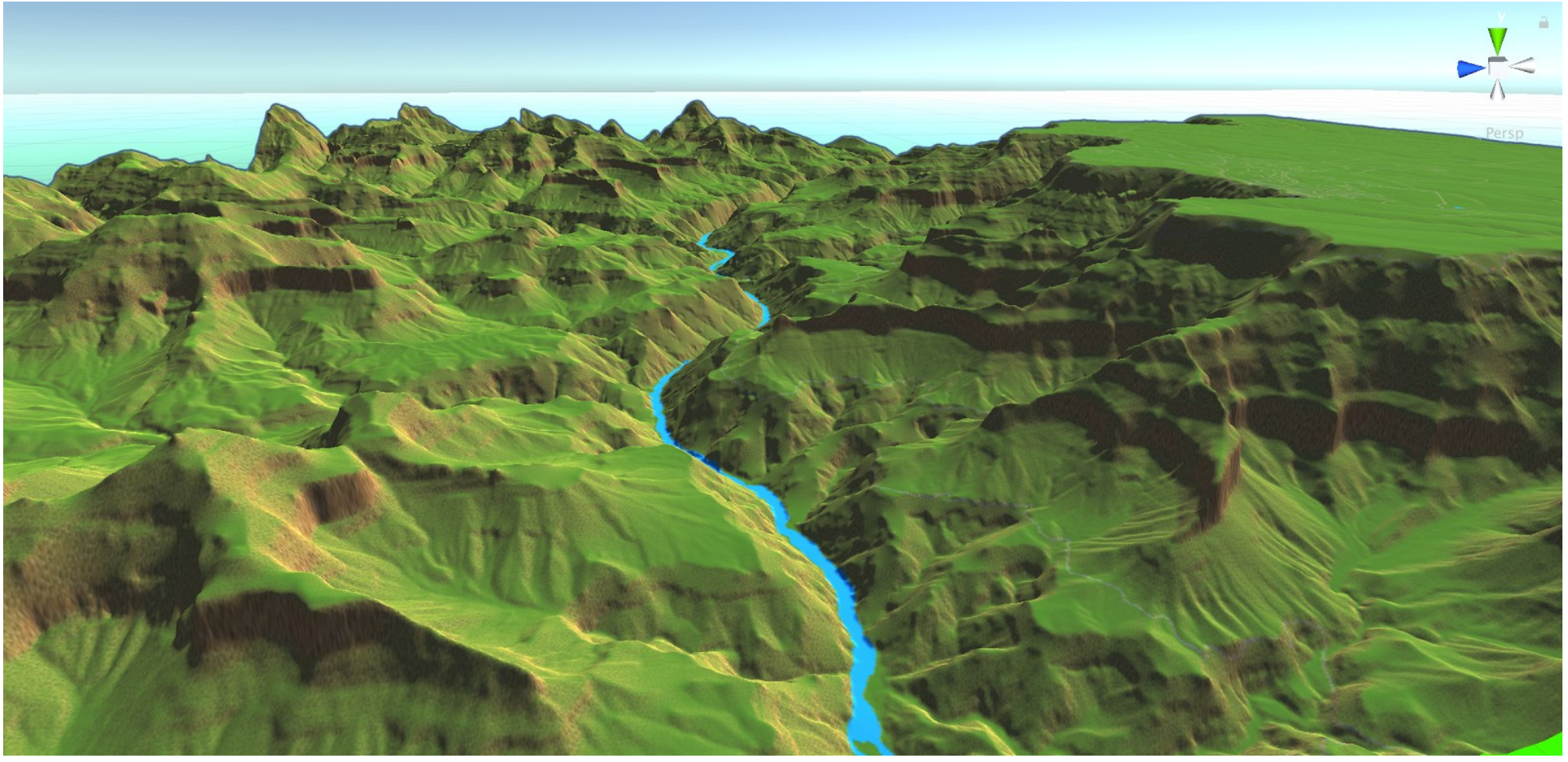
$$f * \mathcal{G}_\sigma * \mathcal{G}_{\sigma'} = f * \mathcal{G}_{\sigma''}$$

$$\sigma'' = \sqrt{\sigma^2 + \sigma'^2}$$

Cascade Gaussians: we can use smaller kernels

Gradient filtering

UNIVERSITÀ
DI PARMA



Gradient filtering

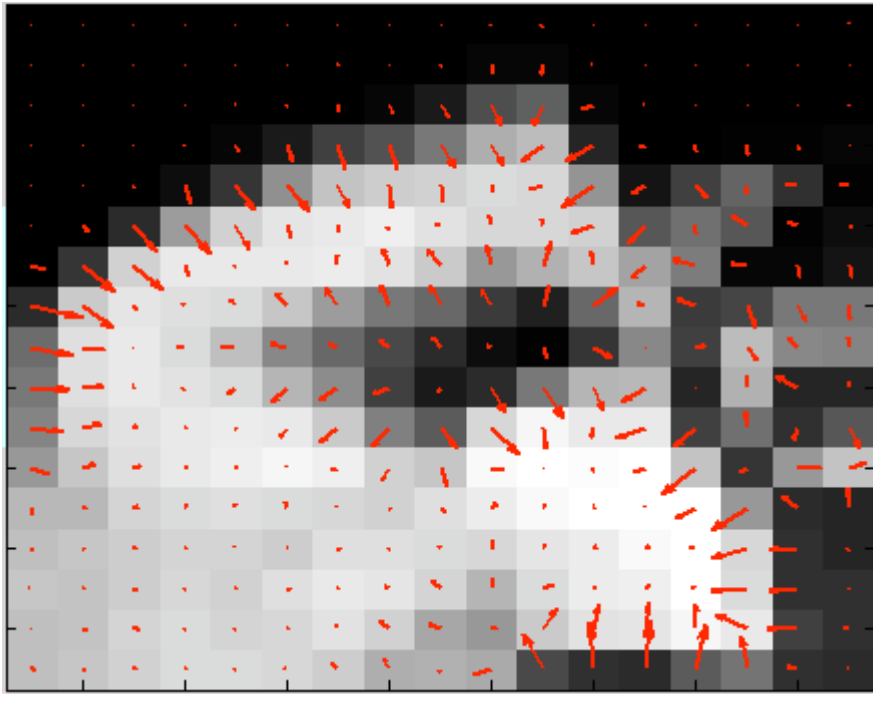
UNIVERSITÀ DI PARMA

- We can see images as 2D “surfaces” where pixel “intensity” encodes elevation
- Therefore we can use concepts like:
  - Uphill/Downhill
  - Peaks/Valleys
  - Discontinuities/Steepness
- Do you remember we can see images as functions?
  - I(x,y)
- For each (x,y)what function can say us something about local intensity variations?
  - Gradient

$$\nabla I(x,y) = \begin{bmatrix} \frac{\partial I(x,y)}{\partial x} \\ \frac{\partial I(x,y)}{\partial y} \end{bmatrix}$$

Gradient filtering

UNIVERSITÀ DI PARMA



Credits: Robert Collins

Gradient filtering

UNIVERSITÀ DI PARMA

- For each pixel we have a vector
  - Magnitude
  - Direction
- Images are discrete functions
  - How we can compute numerical derivates?

$$\frac{\partial I(x,y)}{\partial x} = \lim_{h \to 0} \frac{I(x+h,y) - I(x,y)}{h}$$

Gradient filtering

UNIVERSITÀ DI PARMA

- Taylor Series Expansion can be used
- Precision of the following formula is $O(h^2)$

$$\frac{\partial I(x,y)}{\partial x} \approx \frac{I(x+h,y)-I(x-h,y)}{2h} \qquad \frac{\partial I(x,y)}{\partial y} \approx \frac{I(x,y+h)-I(x,y-h)}{2h}$$

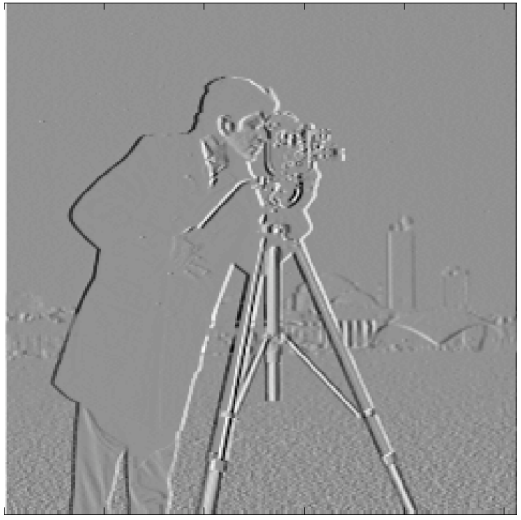
Gradient filtering

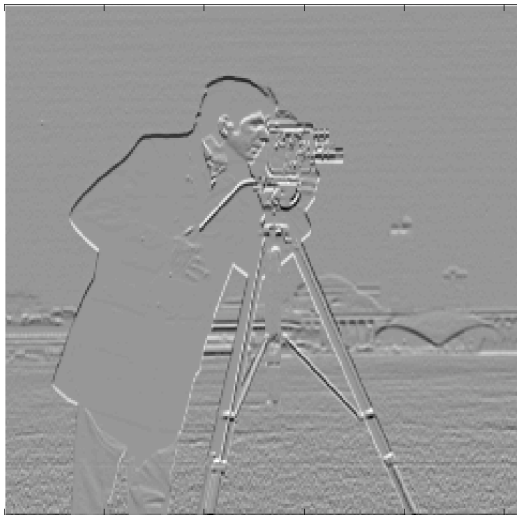
UNIVERSITÀ DI PARMA

$I(x,y)$

$$I_x = \frac{I(x+1,y) - I(x-1,y)}{2}$$

$$I_y = \frac{I(x,y+1) - I(x,y-1)}{2}$$





Credits: Robert Collins

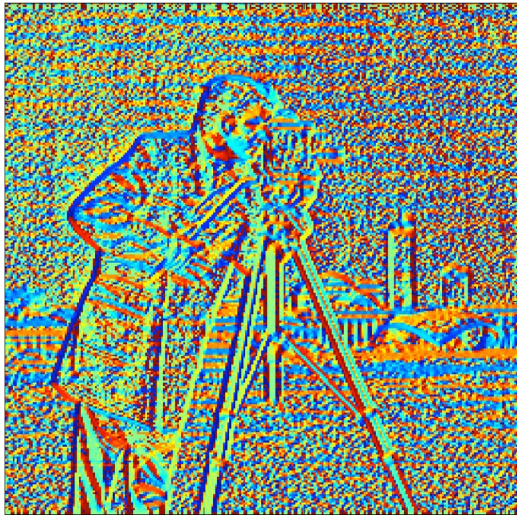
Gradient filtering

UNIVERSITÀ DI PARMA

$I(x, y)$

$\sqrt{I_x^2 + I_y^2}$

$\operatorname{atan} 2(I_x, I_y)$



Credits: Robert Collins

Gradient filtering

UNIVERSITÀ DI PARMA

- I_x and I_y can be easily computed using simple operators

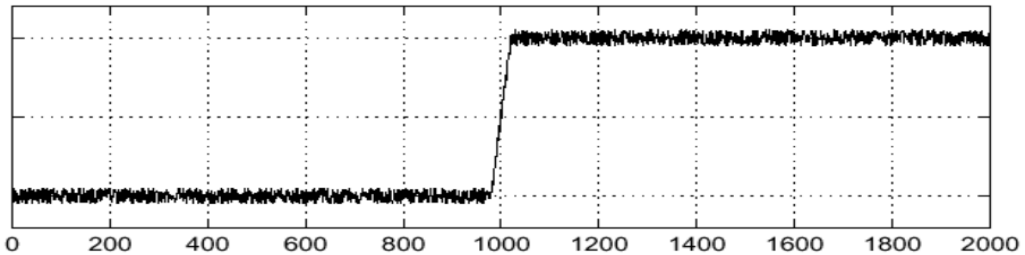
$$I_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * I \qquad I_x = \begin{bmatrix} +1 & 0 & -1 \end{bmatrix} * I$$

- Problem: noise

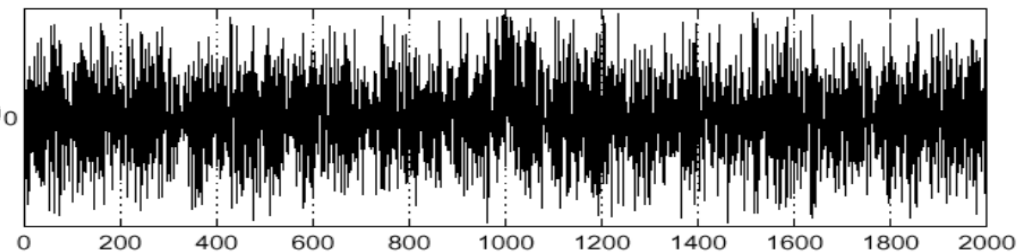
Edge filtering

Consider a single image row

$f(x)$



$\frac{d}{dx} f(x)$



Finding the “edge” is not so easy…

Source: A. Bobick

Gradient filtering

UNIVERSITÀ DI PARMA

- Usually we want to find *strong* gradient variations (edges)
- How to remove noise? ⟶ smoothing

$$I_y = \begin{bmatrix} +1 \\ 0 \\ -1 \end{bmatrix} * \left(\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} * I \right) \qquad I_x = \begin{bmatrix} +1 & 0 & -1 \end{bmatrix} * \left(\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} * I \right)$$

Gradient filtering

UNIVERSITÀ DI PARMA

- Usually we want to find *strong* gradient variations (edges)
- How to remove noise? ⟶ smoothing

$$I_y = \begin{bmatrix} +1 & 0 & -1 \\ +1 & 0 & -1 \\ +1 & 0 & -1 \end{bmatrix} * I \qquad I_x = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * I$$

- Prewitt

Gradient filtering

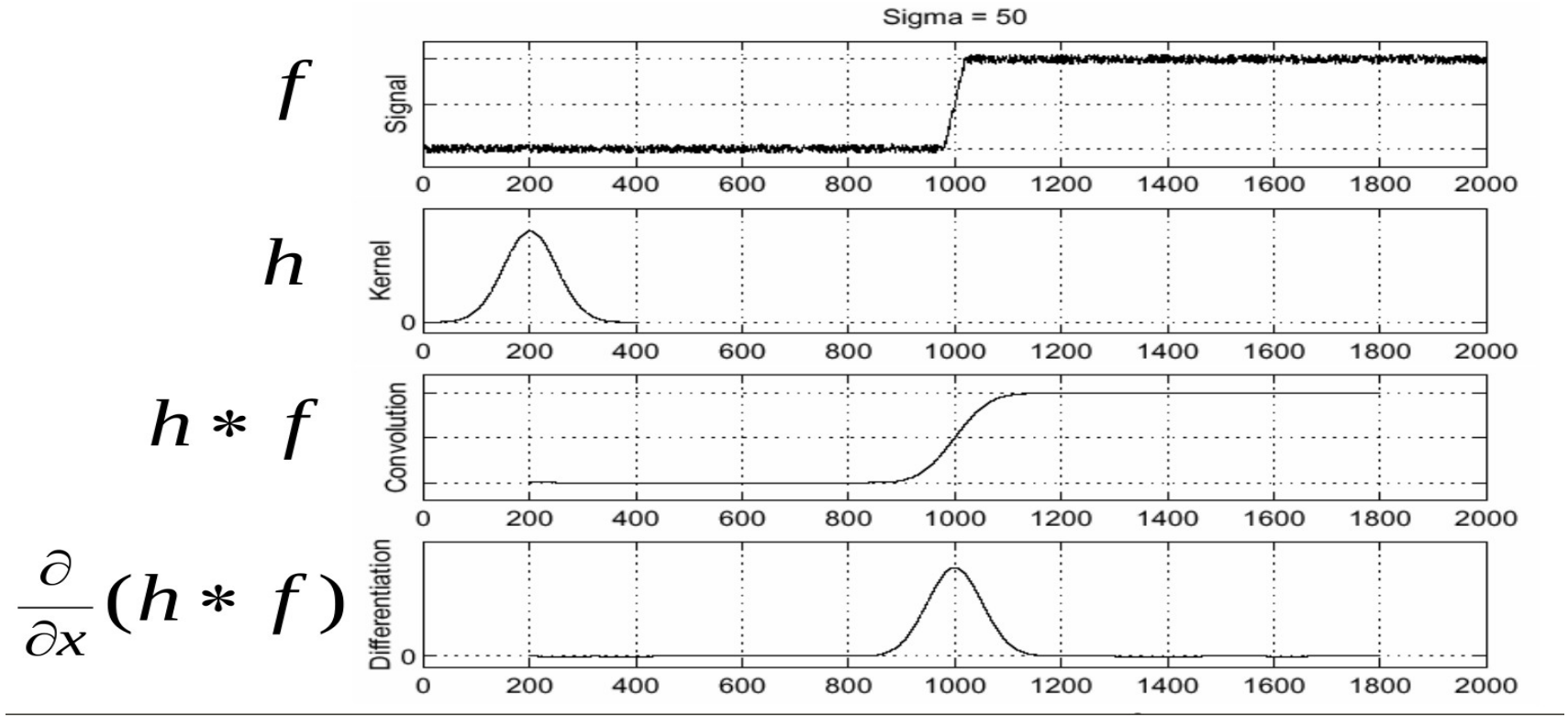
UNIVERSITÀ DI PARMA

- As seen for smoothing for prewitt operators we have the same weight
- **Sobel** operators slightly improves this

$$I_y = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * I \qquad I_x = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I$$

- For smoothing we also saw a gaussian approach

Gradient filtering



Where is the maximum gradient?

Source: A. Bobick

Gradient filtering

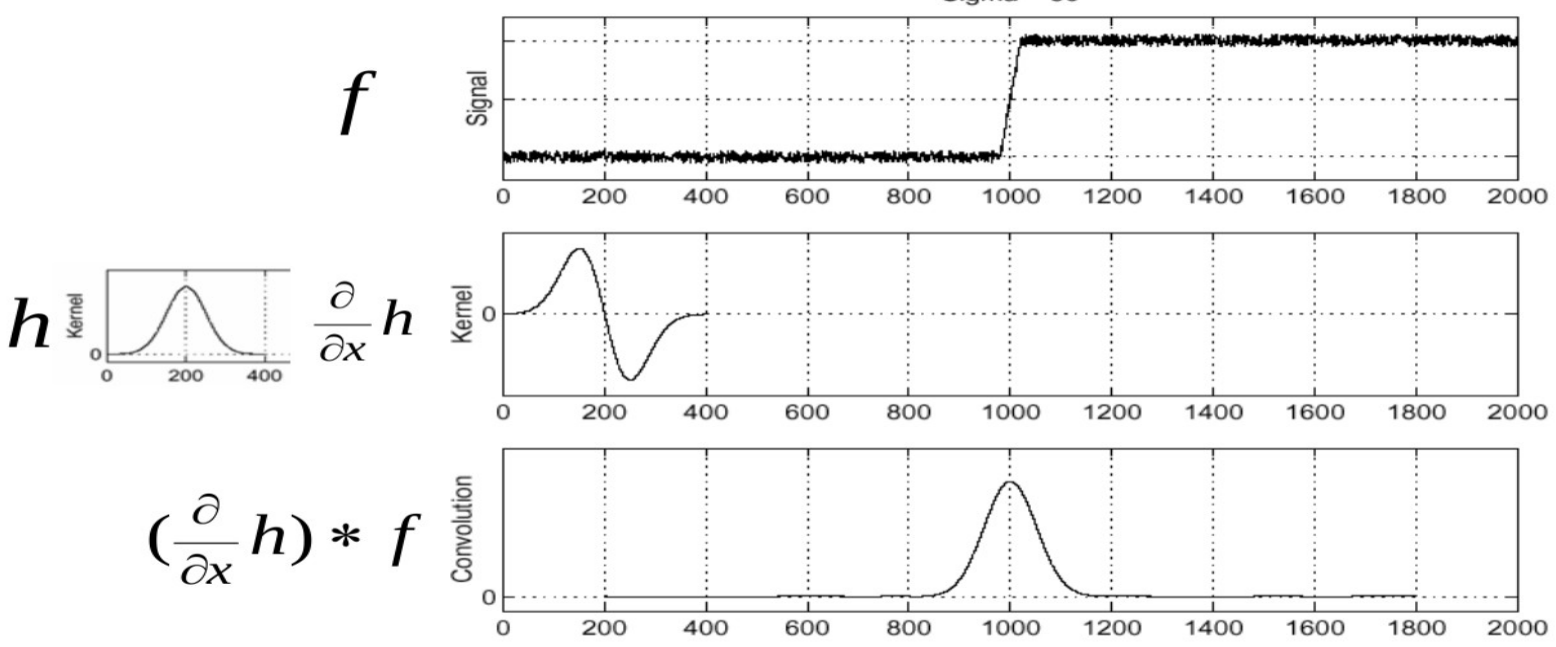
UNIVERSITÀ DI PARMA

- Introducing Prewitt/Sobel we distributed the convolution
  - Namely we applied the derivative operator to smoothing one
- Derivative theorem of convolution

$$\frac{\partial}{\partial x}(f * g) = \left(\frac{\partial}{\partial x} f\right) * g = f * \left(\frac{\partial}{\partial x} g\right)$$

Gradient filtering

- This saves us one operation: $\frac{\partial}{\partial x}(h * f) = (\frac{\partial}{\partial x} h) * f$



And if we want to find local maxima?

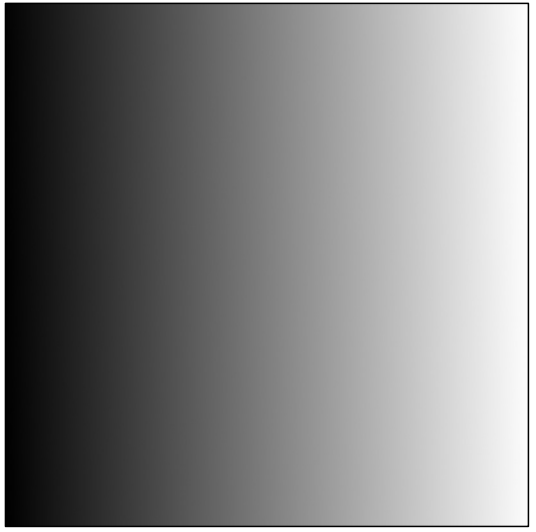
Source: A. Bobick

Gradient filtering

UNIVERSITÀ DI PARMA

- 1st derivative filters
- Where is the edge?
  - Magnitude peaks

$$\sqrt{\frac{\partial}{\partial x} I(x, y)}$$

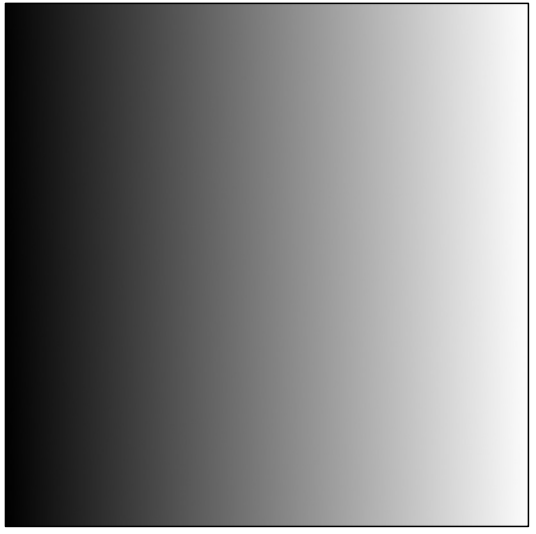


Gradient filtering

UNIVERSITÀ DI PARMA

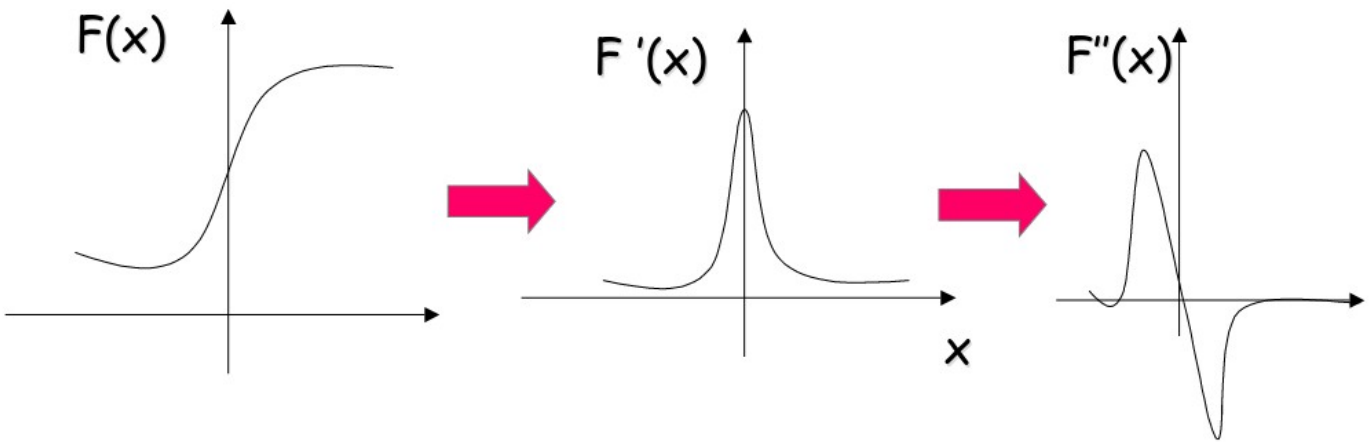
- 1st derivative filters
- Where is the edge?
  - Magnitude peaks

$$\sqrt{I_x^2 + I_y^2} > Th$$



Gradient filtering

- Using 1st derivative filters maxima detect edges
  - Not simple to exactly find them
- What about 2nd derivative filters?



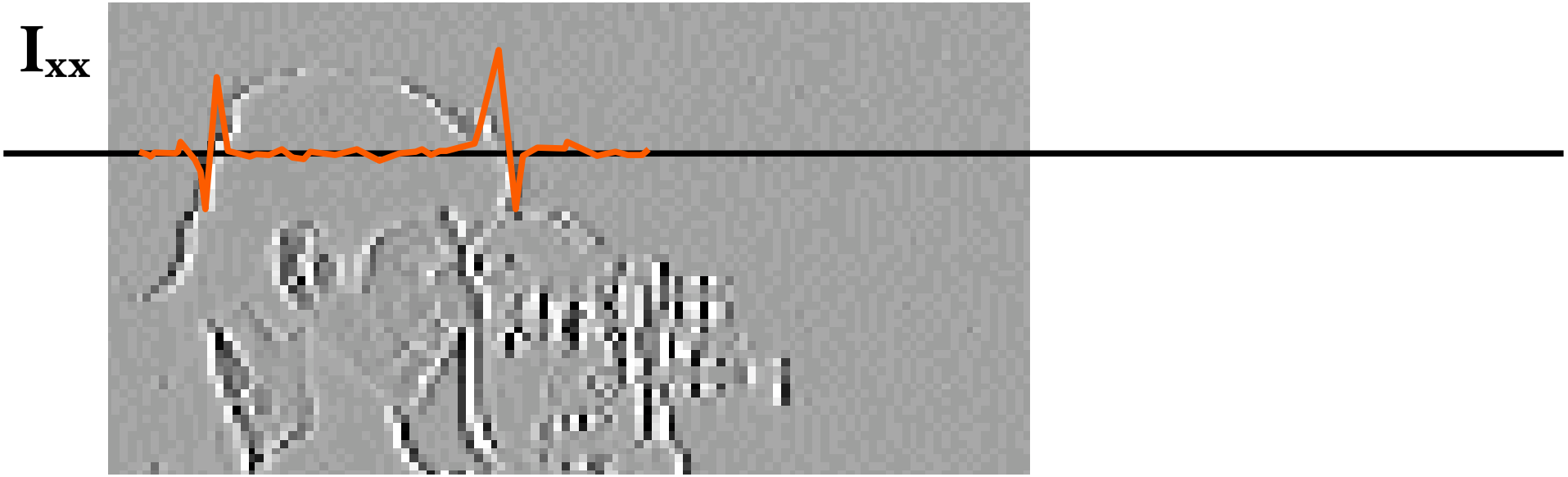
Source: R. Collins

Gradient filtering

UNIVERSITÀ DI PARMA

- Pro: easy localization of edges
- Cons: 2nd derivative is 0 also when 1st one is 0

I_{xx}



Source: R. Collins

Gradient Filtering: Laplacian operator

UNIVERSITÀ DI PARMA

- Laplacian operator ∇^2 can be used to combine I_{xx} and I_{yy}

$$\nabla^2 I(x,y) = I_{xx}(x,y) + I_{yy}(x,y) \approx \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} * I$$

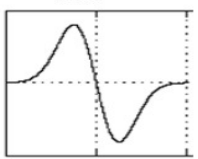
Source: R. Collins

Gradient filtering

- Consider $\frac{\partial^2}{\partial x^2}(h * f)$

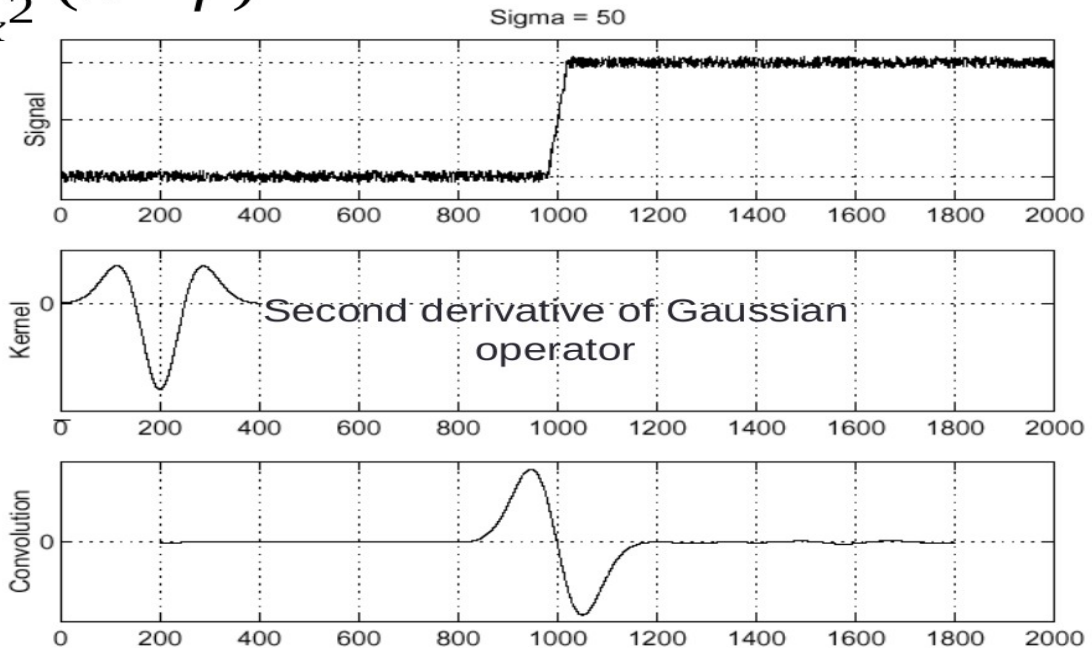
f

$\frac{\partial}{\partial x} h$



$\frac{\partial^2}{\partial x^2} h$

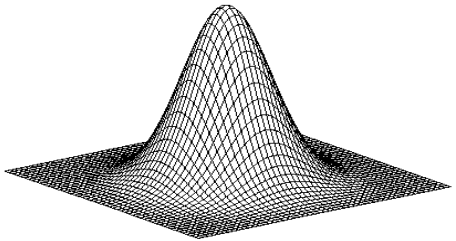
$\frac{\partial^2}{\partial x^2}(h * f)$



The maxima is where we have a zero crossing

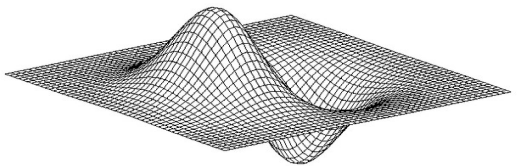
Source: A. Bobick

Gradient filtering



Gaussian

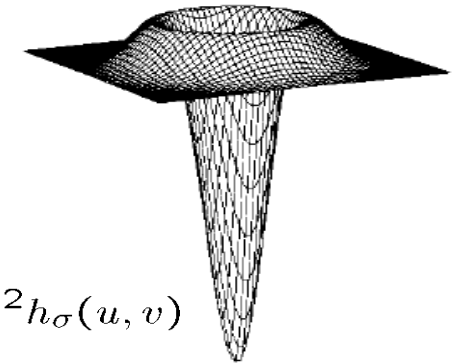
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



x derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian or LoG filter

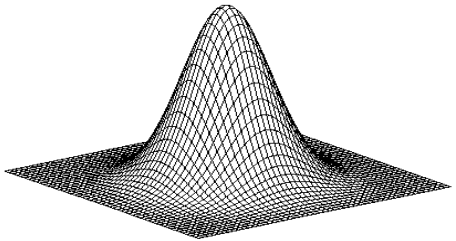


$$\nabla^2 h_\sigma(u,v)$$

∇^2 is the Laplacian operator (sum of 2nd derivatives):

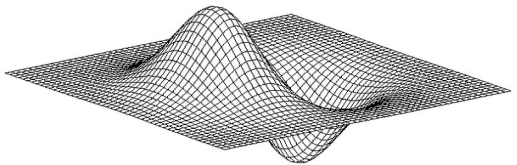
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Gradient filtering



Gaussian

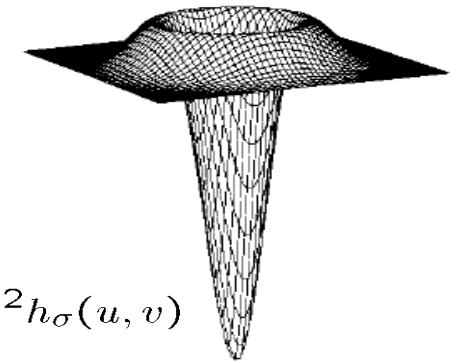
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



x derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

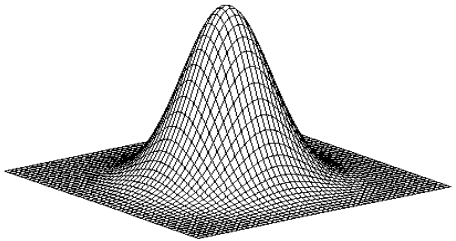
Laplacian of Gaussian or LoG filter



$$\nabla^2 h_\sigma(u,v)$$

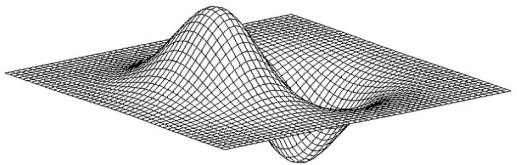
Laplacian(Gaussian * f)
=
LoG * f

Gradient filtering



Gaussian

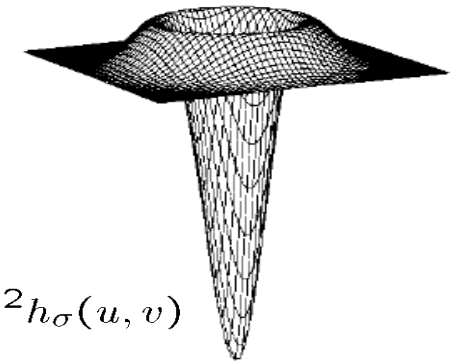
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



x derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian or LoG filter



$$\nabla^2 h_\sigma(u,v)$$

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

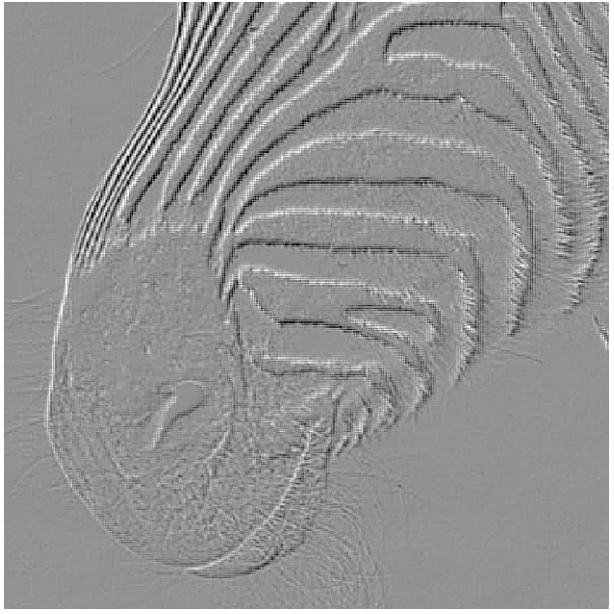
Often approximated by

0	1	0
1	-4	1
0	1	0

LoG Example

UNIVERSITÀ DI PARMA

- Band pass filter
  - Suppress low frequencies (2nd derivative) but also high ones (gaussian)

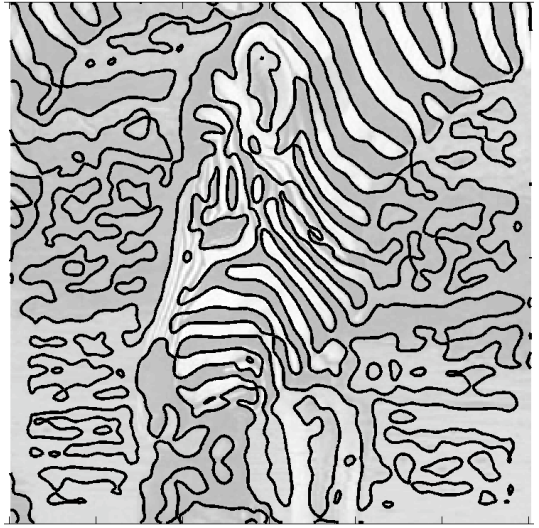


Source: R. Collins

LoG Example: zero crossing

UNIVERSITÀ DI PARMA

- Used as “edge” detector, extract “raw” zero crossings:
  - Different sigmas (2 & 4)
  - Like a “equal elevation” on a map



Source: R. Collins

Back to convolution issues

UNIVERSITÀ DI PARMA

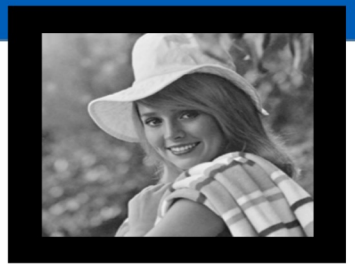
- We now should have undestood basic convolution principles
- We can add further complexity
  - Border issues
  - Padding
  - Stride

Borders issue

UNIVERSITÀ DI PARMA

- How we deal with border/corner pixels in a convolution?
  - Actually the same issue for other operations that involve a mask/kernel
- Basically two approaches
  - Result of the processing is smaller than original image
  - Same size with border/corner pixels with *unknown* value

Borders issue



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

0	10	20	30	30	30	20	10		
0	20	40	60	60	60	40	20		
0	30	60	90	90	90	60	30		
0	30	50	80	80	90	60	30		
0	30	50	80	80	90	60	30		
0	20	30	50	50	60	40	20		
10	20	30	30	30	30	20	10		
10	10	10	0	0	0	0	0		

?	?	?	?	?	?	?	?	?	?
?	0	10	20	30	30	30	20	10	?
?	0	20	40	60	60	60	40	20	?
?	0	30	60	90	90	90	60	30	?
?	0	30	50	80	80	90	60	30	?
?	0	30	50	80	80	90	60	30	?
?	0	20	30	50	50	60	40	20	?
?	10	20	30	30	30	30	20	10	?
?	10	10	10	0	0	0	0	0	?
?	?	?	?	?	?	?	?	?	?

Padding

- **Padding** is a technique to “enlarge” images adding a given amount of pixels to each image border
- The most common approach is the **zero padding**
  - Added pixels are set to 0

Zero Padding

UNIVERSITÀ DI PARMA

- Padding is often use before a convolution to obtain same size result
- The “extra” pixels can be set using different policies
  - Again: zero padding is the most used

Zero Padding

UNIVERSITÀ DI PARMA

- For a 4x4 image

20	128	60	100
50	66	72	20
73	98	65	89
20	170	190	200

Zero Padding

UNIVERSITÀ DI PARMA

- Zero padding 1 → output size is **6x6**

0	0	0	0	0	0
0	20	128	60	100	0
0	50	66	72	20	0
0	73	98	65	89	0
0	20	170	190	200	0
0	0	0	0	0	0

- When we use a 3x3 kernel we obtain a 4x4 resulting image

Zero Padding

UNIVERSITÀ DI PARMA

- Zero padding 2 → output size is **8x8**

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	20	128	60	100	0	0
0	0	50	66	72	20	0	0
0	0	73	98	65	89	0	0
0	0	20	170	190	200	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

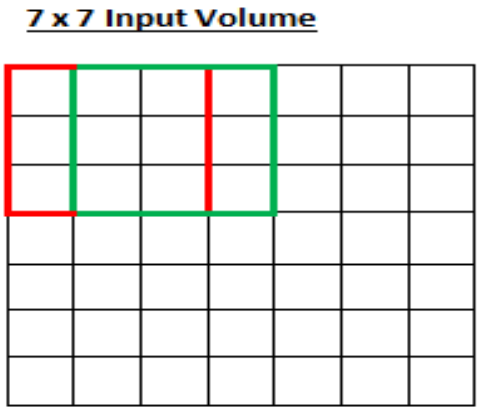
Stride

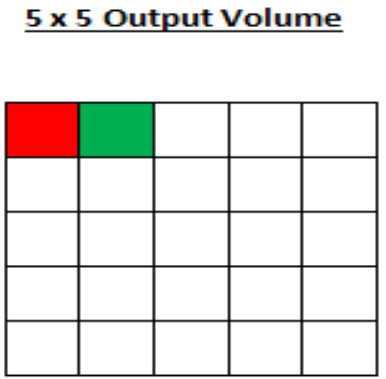
- The stride is a parameter used to set kernel movements on the image
- Since now we assumed that kernel is horizontally and vertically moved on each row/column $\longrightarrow$ stride is 1
- We can move a kernel skipping some pixel $\longrightarrow$ stride > 1
- A stride > 1 leads to subsampling images!
- Moreover we can have different strides for rows/columns

Stride

UNIVERSITÀ DI PARMA

- Stride = 1 ⟶ “normal convolution”. Kernel is moved on every pixel (vertically & horizontally)
- I.e for 7x7 image, a 3x3 kernel and a stride value as 1 we will obtain a **5x5 image**

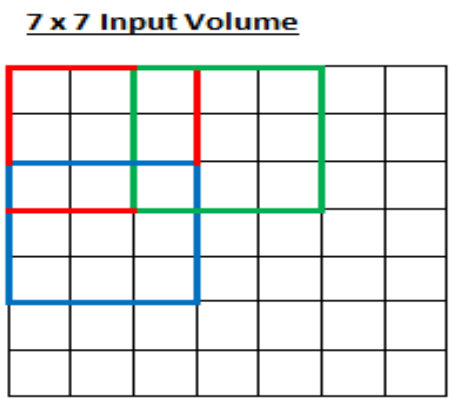


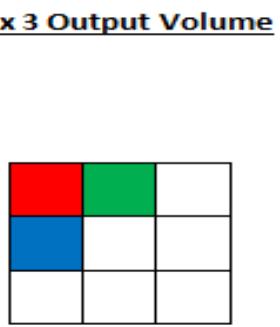


Stride

UNIVERSITÀ DI PARMA

- Stride = 2 ⟶ kernel is moved skipping 1 column for each horizontal move and 1 row for each vertical move
- I.e for 7x7 image, a 3x3 kernel and a stride value as 1 we will obtain a 3**x3 image**





Stride: output image size

UNIVERSITÀ DI PARMA

- Given a image featuring n columns
- Using a kernel with f columns, with padding p and stride s, the number of columns of the output image is:

$$n_{new} = \left\lfloor \frac{n + 2 * p - f}{s} + 1 \right\rfloor$$

- The result must be the largest integer value equal or smaller than n_{new}
- The same formula can be used for computing the new number of rows

UNIVERSITÀ DI PARMA

Image Filtering Local & NON Linear operations

Non linear filters

UNIVERSITÀ DI PARMA

- Since now we illustrate the so-called **linear filters**
  - The result is a linear sum of neighborhood values
  - Linear filters can easily be combined:
    - The result of the processing of the sum of 2 signals can be obtained as the sum of the processing of the 2 signals

$$S(\alpha * f[x,y] + \beta * g[x,y]) = \alpha * S(f[x,y]) + \beta * S(g[x,y]) \; for\; all\; \alpha, \beta \in \mathbb{R}$$

- Linear filters can be easily combined and can also be analyzed in the frequency domain (Fourier transform) but they are not always the most efficient solution

Non linear filters

UNIVERSITÀ DI PARMA

- Median pooling
- Max/Min pooling
- *Average pooling*
- Bilateral filter

Median pooling

UNIVERSITÀ DI PARMA

- For each neighborhood we select the median value

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

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

4

- Is a smoothing operation
- Can cope with salt & pepper noise
- Sorting needed → computationally expensive

Source:Szeliski

Median pooling

UNIVERSITÀ DI PARMA

Shot noise

5x5 average

5x5 median

Source: Collins

Max/Min pooling

UNIVERSITÀ DI PARMA

- Select the largest/smallest value in the neighborhood

20	128	60	100
50	66	72	20
73	98	65	89
20	170	190	200

128	128	100
98	98	89
170	190	200

- Can be used to selectively remove salt & pepper noise

Average pooling

UNIVERSITÀ DI PARMA

- We already saw this guy!
- Box filter (linear)

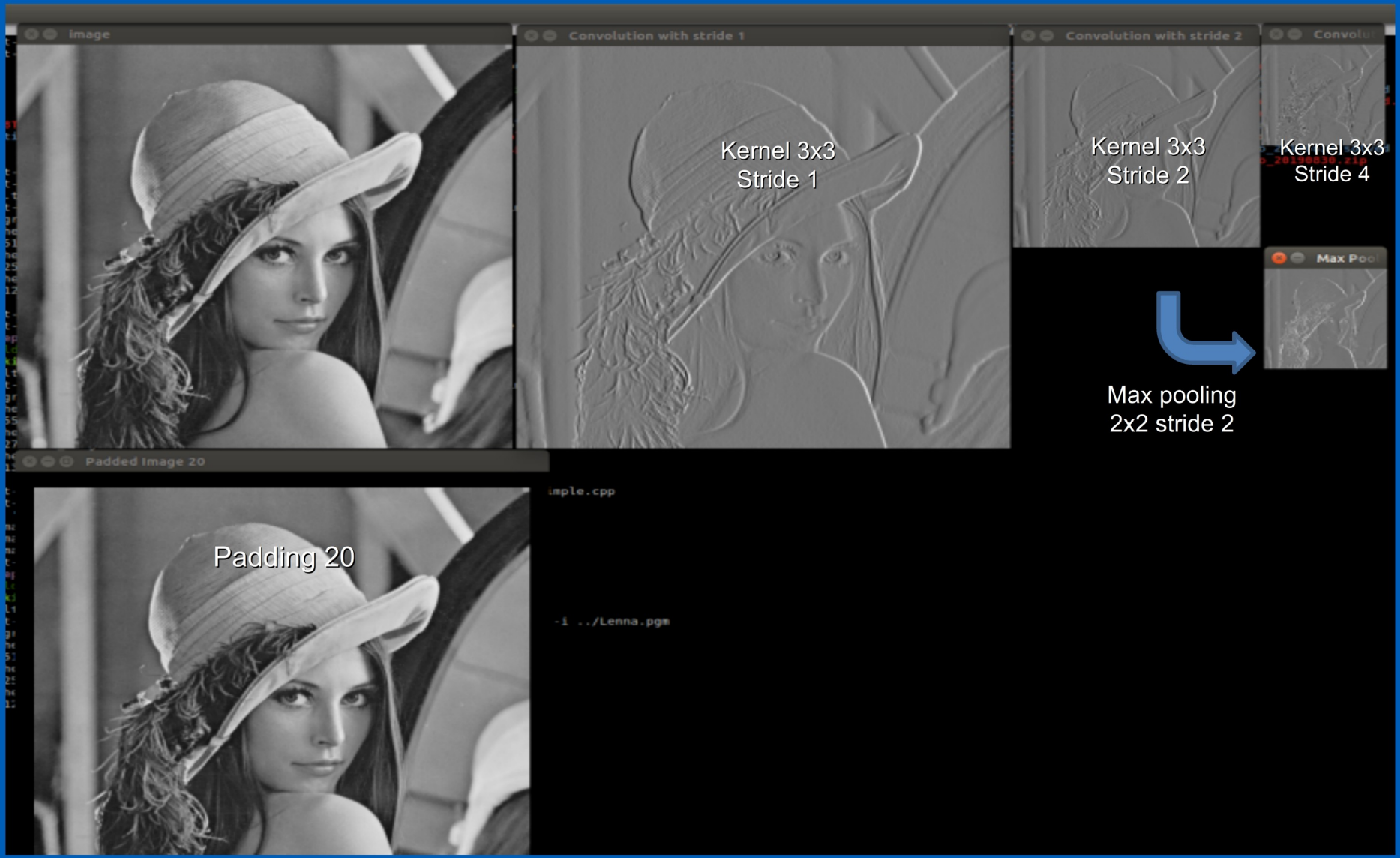
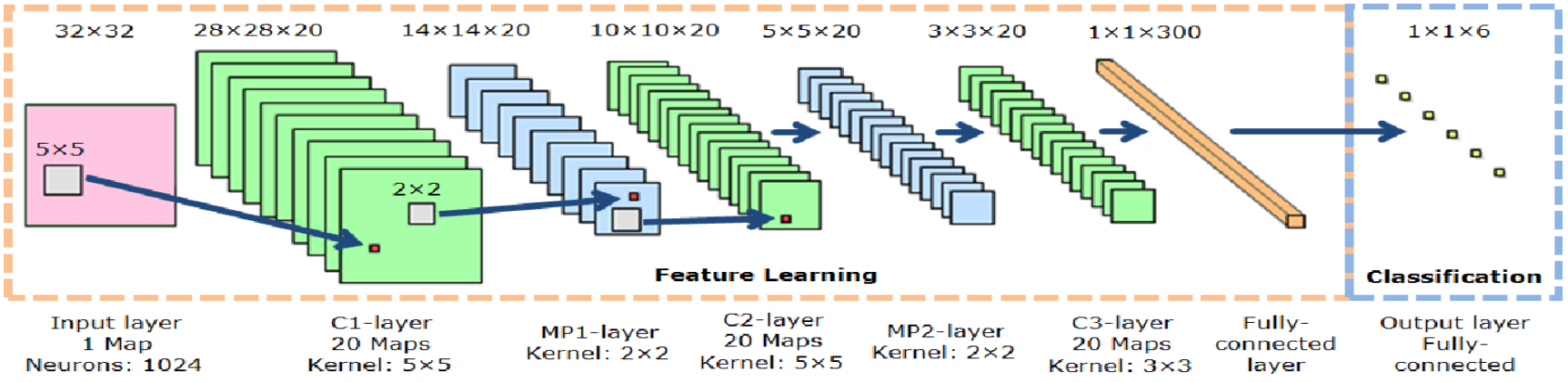


image
Convolution with stride 1
Kernel 3x3
Stride 1
Convolution with stride 2
Kernel 3x3
Stride 2
Kernel 3x3
Stride 4
Max Pool
Max pooling
2x2 stride 2
Padded Image 20
Padding 20
imple.cpp
-i ../Lenna.pgm

Convolutional Neural Network

UNIVERSITÀ DI PARMA

- Convoluzioni con stride e max pooling sono gli elementi base di una **Convolutional Neural Network** (CNN):

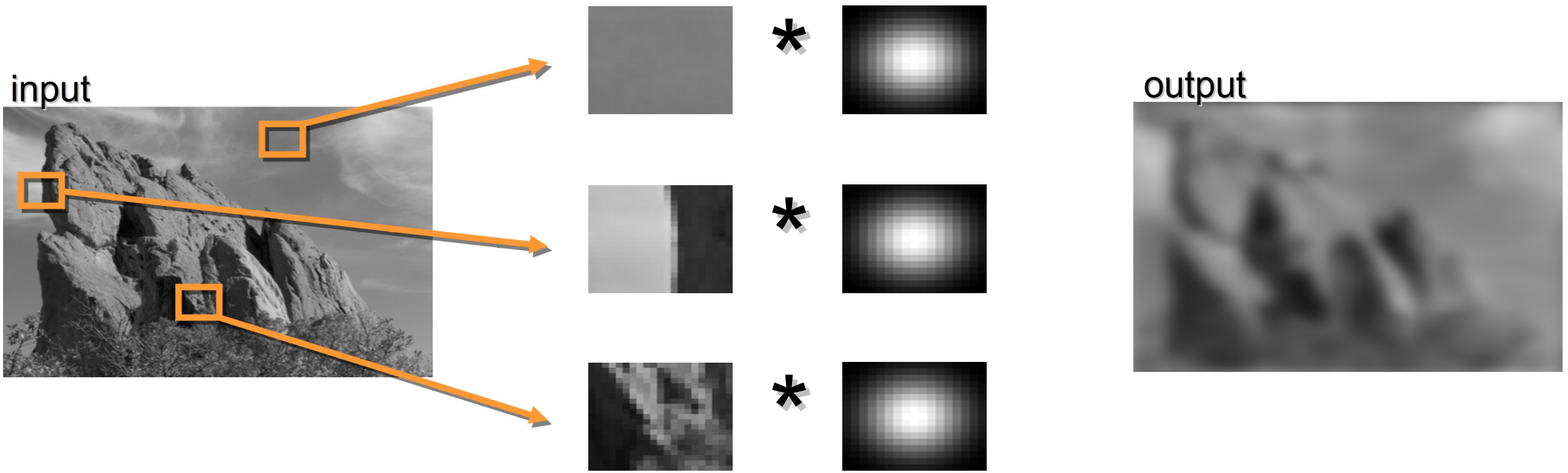


Bilateral filter

UNIVERSITÀ DI PARMA

- Smoothing vs Edges
  - Smoothing is mainly used for noise reduction
  - Anyway also edges can be affected…
- Gaussian filters do not depend on image content
  - Strong gradient variations are treated like uniform areas

Bilateral filter



Same Gaussian kernel everywhere.

Slides courtesy of Sylvian Paris

Bilateral filter

UNIVERSITÀ DI PARMA

- Bilateral filter exploits a specific weight for adapting a gaussian kernel to image content
- w(i,j,k,l) ⟶ weighting coefficient

$$g(i,j)=\frac{\sum_{k,l} f(i+k,j+l)\,w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

Source: Szeliski

Bilateral filter

UNIVERSITÀ DI PARMA

- The weighting coefficient can be obtained as the composition of 2 kernels:
  - Gaussian filtering ⟶ **Domain** Kernel
    - Independent from image content
  - Weight correction ⟶ **Range** Kernel
    - Dependent on (local) image content
- Bilateral filtering is space variant since kernel depends on pixel value
  - Non linear filtering

Bilateral filter

UNIVERSITÀ DI PARMA

- The domain kernel is a Gaussian Kernel

$$d(i,j,k,l) = e^{-\left[\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2}\right]}$$

- d(i,j,k,l) does not depend on image content

Source: Szeliski

Bilateral filter

UNIVERSITÀ DI PARMA

- The range kernel is data dependent

$$r(i,j,k,l) = e^{-\left[\frac{\|f(i,j) - f(i+k,j+l)\|^2}{2\sigma_r^2}\right]}$$

- r(i,j,k,l) uses the **vector distance** between center pixel and neighboring one!

Source: Szeliski

Bilateral filter

UNIVERSITÀ DI PARMA

- The overall kernel is

$$w(i,j,k,l) = e^{-\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2} - \frac{\|f(i,j)-f(i+k,j+l)\|^2}{2\sigma_r^2}}$$

- d(i,j,k,l) uses the **vector distance** between center pixel and neighboring one!

Source: Szeliski

Bilateral filter

UNIVERSITÀ DI PARMA

0.1	0.2	0.0	200	213	222
0.0	0.2	0.3	201	223	247
0.2	0.7	0.1	210	201	233
0.4	0.3	0.5	220	216	235
0.3	0.4	0.3	225	250	221
0.6	0.5	0.6	215	243	200

domain kernel $\sigma=2$

0.77	0.88	0.77
0.88	1.0	0.88
0.77	0.88	0.77

range kernel with $\sigma=5$

0.999	0.999	0.0
0.992	1.0	0.0
0.999	0.996	0.0

w

0.76	0.87	0.0
0.872	1.0	0.0
0.76	0.87	0.0

Bilateral filter

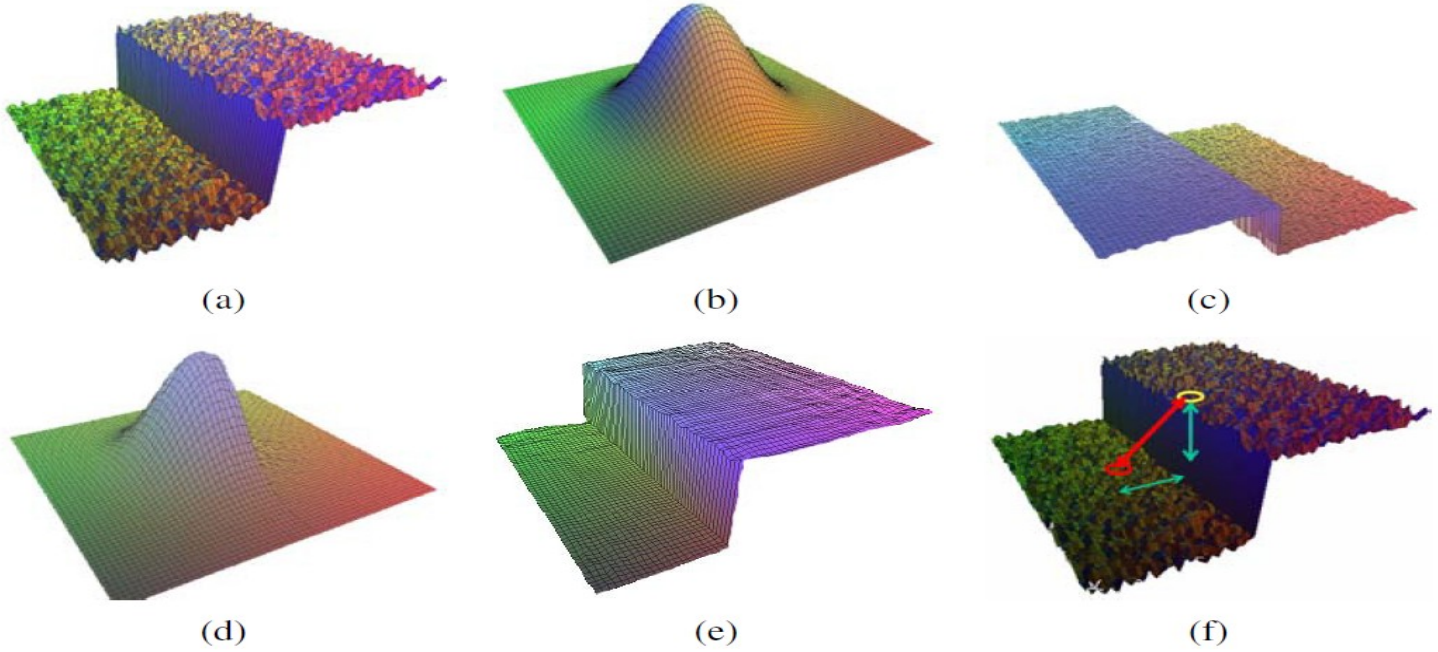
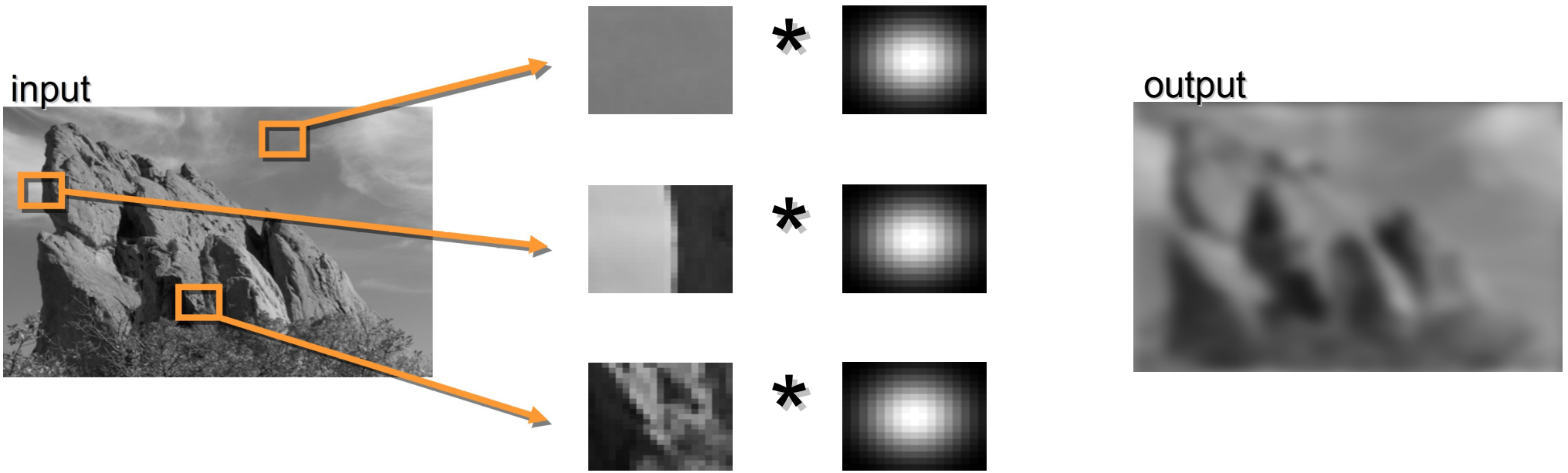


Figure 3.20 Bilateral filtering (Durand and Dorsey 2002) © 2002 ACM: (a) noisy step edge input; (b) domain filter (Gaussian); (c) range filter (similarity to center pixel value); (d) bilateral filter; (e) filtered step edge output; (f) 3D distance between pixels.

Source:Szeliski

Bilateral filter



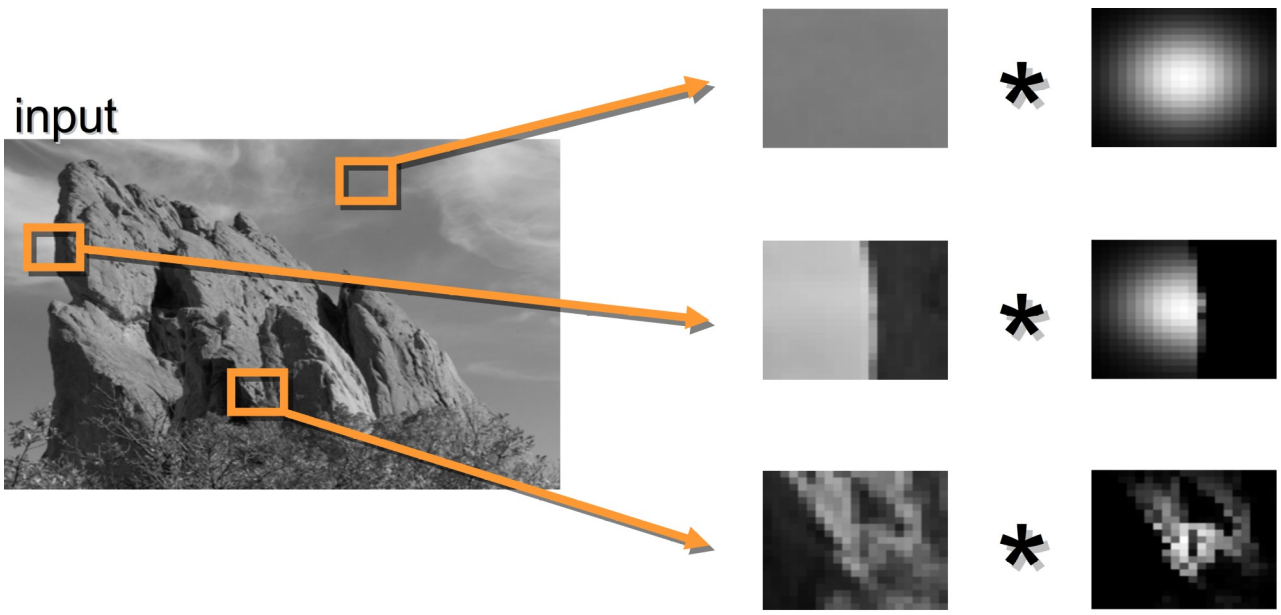
Same Gaussian kernel everywhere.

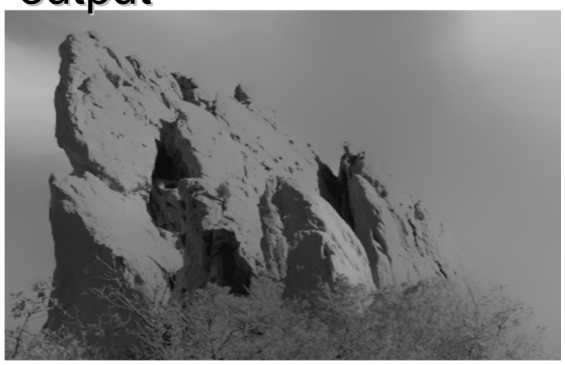
Slides courtesy of Sylvian Paris

Bilateral filter

UNIVERSITÀ DI PARMA

Maintains edges when blurring!





The kernel shape depends on the image content.

Slides courtesy of Sylvian Paris

Bilateral filter

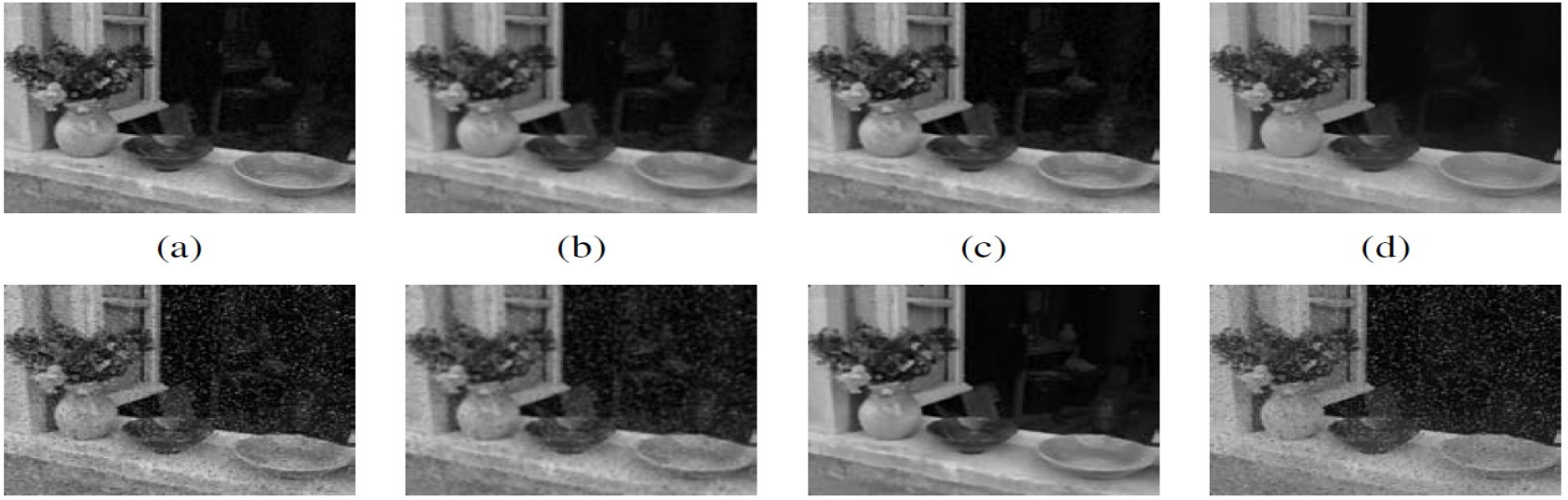


Figure 3.18 Median and bilateral filtering: (a) original image with Gaussian noise; (b) Gaussian filtered; (c) median filtered; (d) bilaterally filtered; (e) original image with shot noise; (f) Gaussian filtered; (g) median filtered; (h) bilaterally filtered. Note that the bilateral filter fails to remove the shot noise because the noisy pixels are too different from their neighbors.

Szeliski

UNIVERSITÀ DI PARMA

Image Filtering
Integral Image

Integral Image

UNIVERSITÀ DI PARMA

- Often convolutions imply to repeatedly compute the sum of a region.

$$S(i,j)=\sum_{k=0}\sum_{l=0} F(k,l)$$

- This is an expensive operation → $O(n^2)$

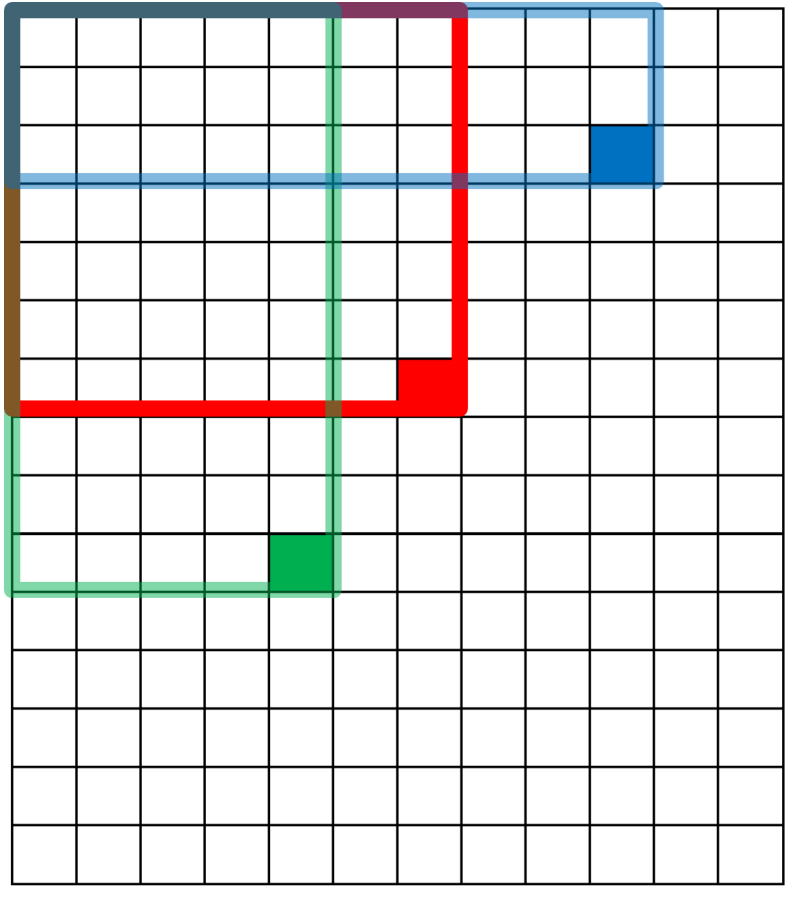
- The **integral image** can lead to O(n)
- Also known as **summed area table**

243	239	240	225	206	185	188	218	211	206	216	225
242	239	218	110	67	31	34	152	213	206	208	221
243	242	123	58	94	82	132	77	108	208	208	215
235	217	115	212	243	236	247	139	91	209	208	211
233	208	131	222	219	226	196	114	74	208	213	214
232	217	131	116	77	150	69	56	52	201	228	223
232	232	182	186	184	179	159	123	93	232	235	235
232	236	201	154	216	133	129	81	175	252	241	240
235	238	230	128	172	138	65	63	234	249	241	245
237	236	247	143	59	78	10	94	255	248	247	251
234	237	245	193	55	33	115	144	213	255	253	251
248	245	161	128	149	109	138	65	47	156	239	255
190	107	39	102	94	73	114	58	17	7	51	137
23	32	33	148	168	203	179	43	27	17	12	8
17	26	12	160	255	255	109	22	26	19	35	24

Integral Image

UNIVERSITÀ DI PARMA

- Each pixel (x,y) contains the sum of other pixels contained in the rectangle (0,0)-(x,y)



Integral Image

UNIVERSITÀ DI PARMA

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

4				

Integral Image

UNIVERSITÀ DI PARMA

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

4	7			

Integral Image

UNIVERSITÀ DI PARMA

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

4	7	14		

Integral Image

UNIVERSITÀ DI PARMA

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

4	7	14	23	

Integral Image

UNIVERSITÀ DI PARMA

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

4	7	14	23	26

Integral Image

UNIVERSITÀ DI PARMA

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

4	7	14	23	26
6				

Integral Image

UNIVERSITÀ DI PARMA

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

4	7	14	23	26
6	14			

Integral Image

- A fast way to compute the summed area table is:
  - Initialize integral image I using same values of original image
  - Start from (0,0) and move row by row
  - Update each “cell” as
    - $I(x, y) = I(x, y) + I(x-1, y) + I(x, y-1) - I(x-1, y-1)$

Integral Image

UNIVERSITÀ DI PARMA

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

4	7	14	23	26
6	14	7	9	1
5	6	4	7	2
1	1	4	3	2
0	2	3	4	1

Integral Image

UNIVERSITÀ DI PARMA

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

4	7	14	23	26
6	14	28	9	1
5	6	4	7	2
1	1	4	3	2
0	2	3	4	1

Integral Image

UNIVERSITÀ DI PARMA

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

4	7	14	23	26
6	14	28	46	1
5	6	4	7	2
1	1	4	3	2
0	2	3	4	1

Integral Image

UNIVERSITÀ DI PARMA

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

4	7	14	23	26
6	14	28	46	50
11	25	43	68	74
12	27	49	77	85
12	29	54	86	95

Integral Image

UNIVERSITÀ DI PARMA

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

4	7	14	23	26
6	14	28	46	50
11	25	43	68	74
12	27	49	77	85
12	29	54	86	95

Let's try to compute the sum of that area 7+9+1+4+7+2+4+3+2 = 39

Integral Image

UNIVERSITÀ DI PARMA

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

4	7 B	14	23	26 C
6	14	28	46	50
11	25	43	68	74
12	27 D	49	77	85 A
12	29	54	86	95

Consider the same area on the Integral Image and cells A, B, C, D

Integral Image

UNIVERSITÀ DI PARMA

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

4	7 B	14	23	26 C
6	14	28	46	50
11	25	43	68	74
12	27 D	49	77	85 A
12	29	54	86	95

The result is A+B-C-D ⟶ 85 + 7 -26 – 27 = 39

Integral Image

UNIVERSITÀ DI PARMA

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

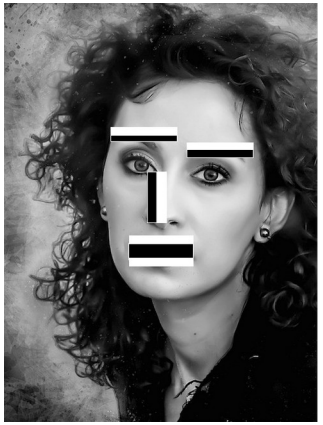
4	7 B	14	23	26 C
6	14	28	46	50
11	25	43	68	74
12	27 D	49	77	85 A
12	29	54	86	95

The result is A+B-C-D ⟶ 85 + 7 -26 – 27 = 39

Integral Image

UNIVERSITÀ DI PARMA

- Examples of usage
  - Filter box: given a n×n kernel on a M×N image
    - No integral image, separable kernels → O(M×N×2n)
    - Computing integral image → O(2×M×N)
  - Haar like features
    - Binary patterns used for face detection



UNIVERSITÀ DI PARMA

Image Filtering