



Corso di Visione Artificiale

Laurea Magistrale in Informatica (F94)

Docenti:

Raffaella Lanzarotti

Federico Pedersini

*Dipartimento di Informatica
Università degli Studi di Milano*



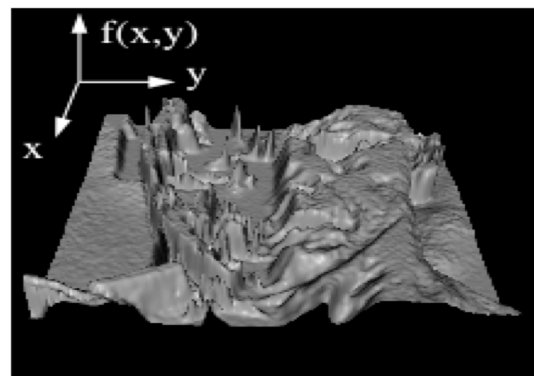
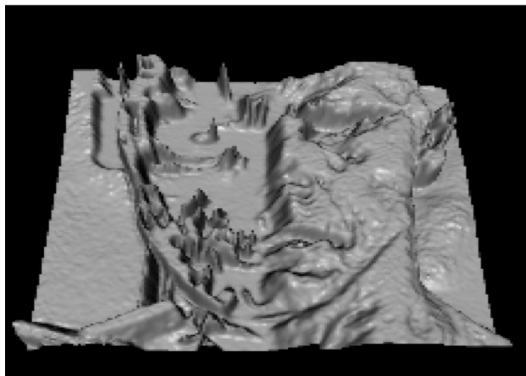
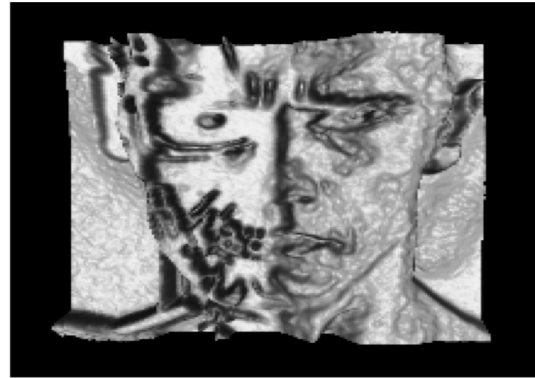
Filtraggio lineare di immagini

- ❖ **Filtraggio lineare nel dominio dello spazio**
 - convoluzione 2D
 - esempi di filtraggio lineare (blurring, sharpening)

- ❖ **Filtraggio nel dominio delle frequenze**
 - trasformata di Fourier 2D, spettro di un'immagine
 - filtraggio nel dominio delle frequenze
 - campionamento spaziale 2D / spettro / aliasing / Moiré
 - Applicazione: sotto-/sovra-campionamento

(Forsyth/Ponce: Capitolo 4)

Slide credits: varie sorgenti (citare)



source: S. Seitz



❖ **Immagine:** funzione definita sul **piano immagine:**

- Posizione $(x,y) \rightarrow f(x,y)$: **intensità** in posizione (x,y) $f: \mathbb{R}^2 \rightarrow \mathbb{R}$
- Piano immagine: dominio rettangolare, finito: $\langle x \in [X_{MIN}, X_{MAX}], y \in [Y_{MIN}, Y_{MAX}] \rangle$
- Immagine a **colori**: 3 componenti in ogni punto: $I_{RGB}(x,y) = \begin{bmatrix} I_R(x,y) \\ I_G(x,y) \\ I_B(x,y) \end{bmatrix}$

❖ **Immagine digitale:** immagine **campionata e quantizzata**

- **Campionamento:** dominio spaziale discretizzato: $I: f(i,j), i,j \in \mathbb{N}$
- **Quantizzazione:** valore della funzione discretizzato: $I: f(i,j), f \in \{I_{min}, I_{max}\} \subset \mathbb{N}$

$I: f(i,j)$

62	79	23	119	120	105
10	10	9	62	12	78
10	58	197	46	46	0
176	135	5	188	191	68
2	1	1	29	26	37
0	89	144	147	187	102

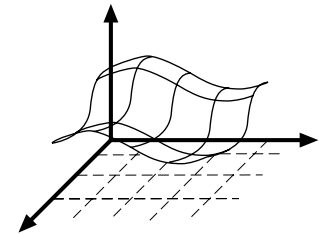
credits: S. Seitz



Filtraggio lineare di immagini

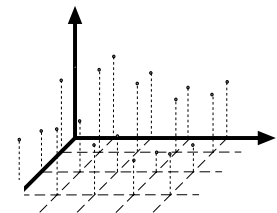
elaborazione dell'immagine descrivibile come una trasformazione applicata all'immagine da un **sistema lineare e spazio-invariante**, detto **filtro lineare**.

- filtraggio: descrivibile mediante l'operazione di **convoluzione**, nel dominio dell'immagine
- oppure come **prodotto**, nel dominio delle **frequenze spaziali**



Filtraggio lineare di immagini digitali:

filtraggio lineare su un dominio (spaziale) discreto



Nel dominio delle frequenze:

trasformata di Fourier su un dominio 2D → **FFT-2D**

Esempi applicativi: riduzione del rumore



Noise reduction:

come posso ridurre il **rumore** in immagini di una scena statica?



1. acquisisco tante (N) immagini uguali $I_i(x, y) = C(x, y) + n_i(x, y)$

...e faccio la media:

$$\bar{I}(x, y) = C(x, y) + \frac{1}{N} \sum_i n_i(x, y)$$

source: S. Seitz



Come posso ridurre il rumore, da immagini di una scena statica?



❖ *6 fotogrammi ripetuti...*



Come posso ridurre il rumore, da immagini di una scena statica?



- ❖ *...e sommati* 
- ❖ *E se non dispongo di immagini multiple?*

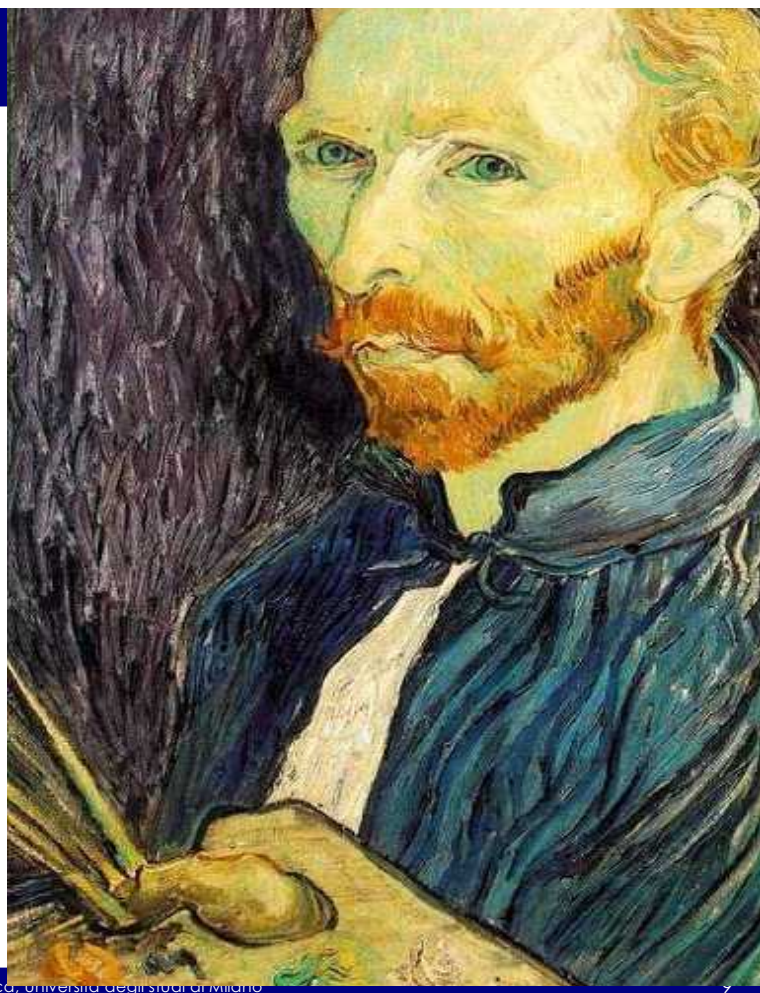
Motivazioni: ridimensionamento

- ❖ L'immagine è troppo grande.
Come possiamo ridurla
(ad es. di un fattore 2)?

Proposta:

- ❖ *prendiamo un pixel ogni 2*

Va bene?



source: S. Seitz

Visione Artificiale – F. Pedersini

Dip. Informatica, Università degli studi di Milano

Riduzione del rumore

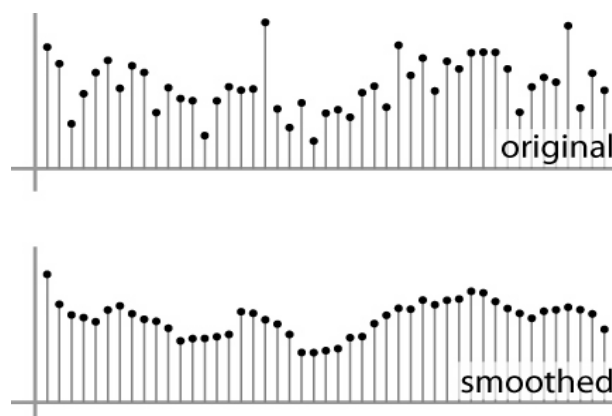


Riduzione del rumore in un'immagine

Approccio:

sostituiamo il valore di ogni pixel con la media dei suoi vicini

- ❖ **Media pesata (1D):**



source: S. Marschner

Visione Artificiale – F. Pedersini

Dip. Informatica, Università degli studi di Milano

10



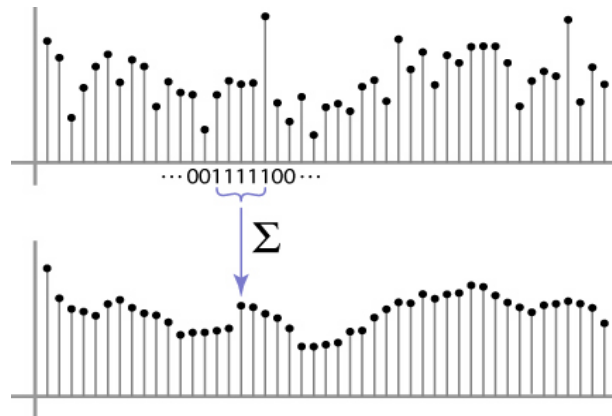
Riduzione del rumore in un'immagine

Approccio:

sostituiamo il valore di ogni pixel con la media dei suoi vicini

- ❖ Media pesata (1D):

- ❖ pesi uniformi: $[1, 1, 1, 1, 1] / 5$



source: S. Marschner



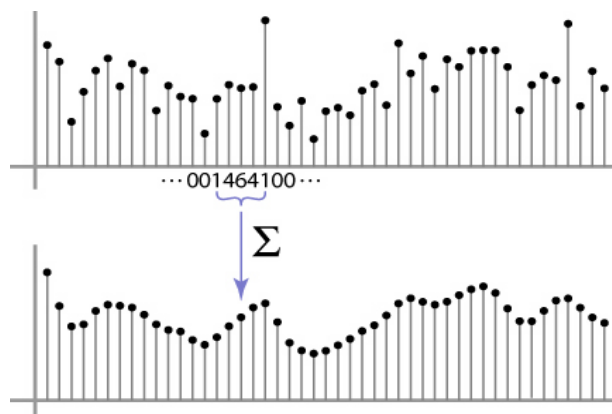
Riduzione del rumore in un'immagine

Approccio:

sostituiamo il valore di ogni pixel con la media dei suoi vicini

- ❖ Media pesata (1D):

- ❖ pesi **non** uniformi: $[1, 4, 6, 4, 1] / 16$



source: S. Marschner



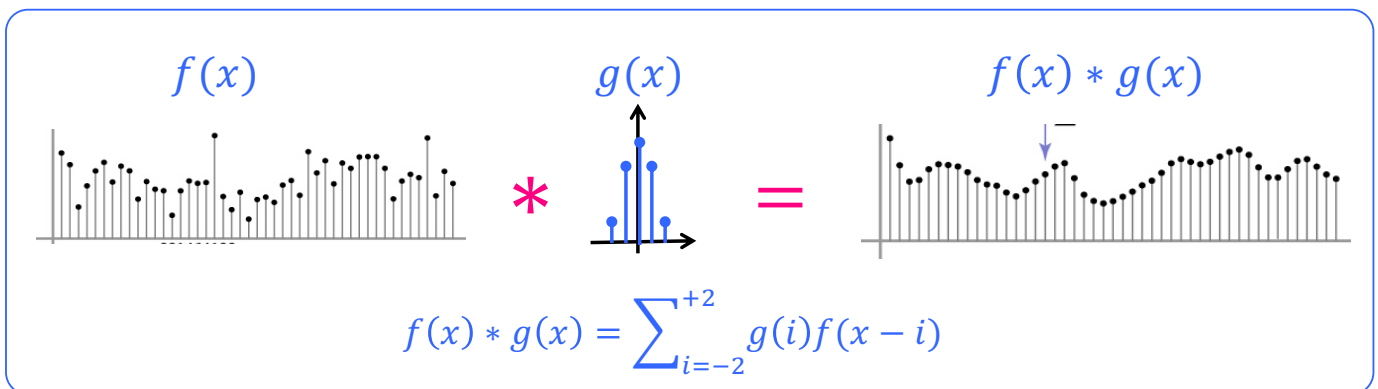
Approccio:

sostituiamo ogni **campione (pixel)** con una **media pesata** dei suoi vicini

La descrizione analitica di questa operazione è la

Convoluzione (discreta):

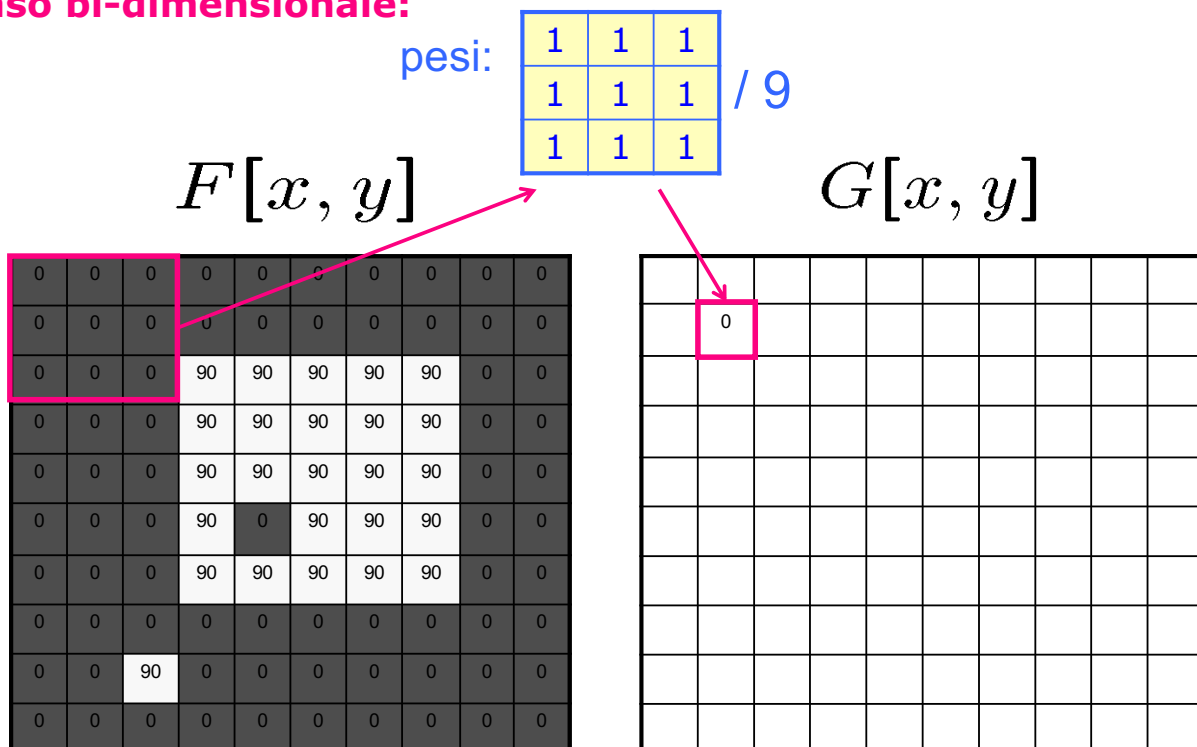
$$(f * g)(x) = f(x) * g(x) = \sum_k f(x - k) g(k)$$



Convoluzione in 2D



Caso bi-dimensionale:





Caso bi-dimensionale:

pesi: $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} / 9$

$F[x, y]$

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	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

$G[x, y]$

	0	10							

source: S. Seitz



Caso bi-dimensionale:

pesi: $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} / 9$

$F[x, y]$

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

$G[x, y]$

	0	10	20						

source: S. Seitz



Caso bi-dimensionale:

pesi:

1	1	1
1	1	1
1	1	1

 / 9

$F[x, y]$

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

$G[x, y]$

	0	10	20	30					

source: S. Seitz



Caso bi-dimensionale:

pesi:

1	1	1
1	1	1
1	1	1

 / 9

$F[x, y]$

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

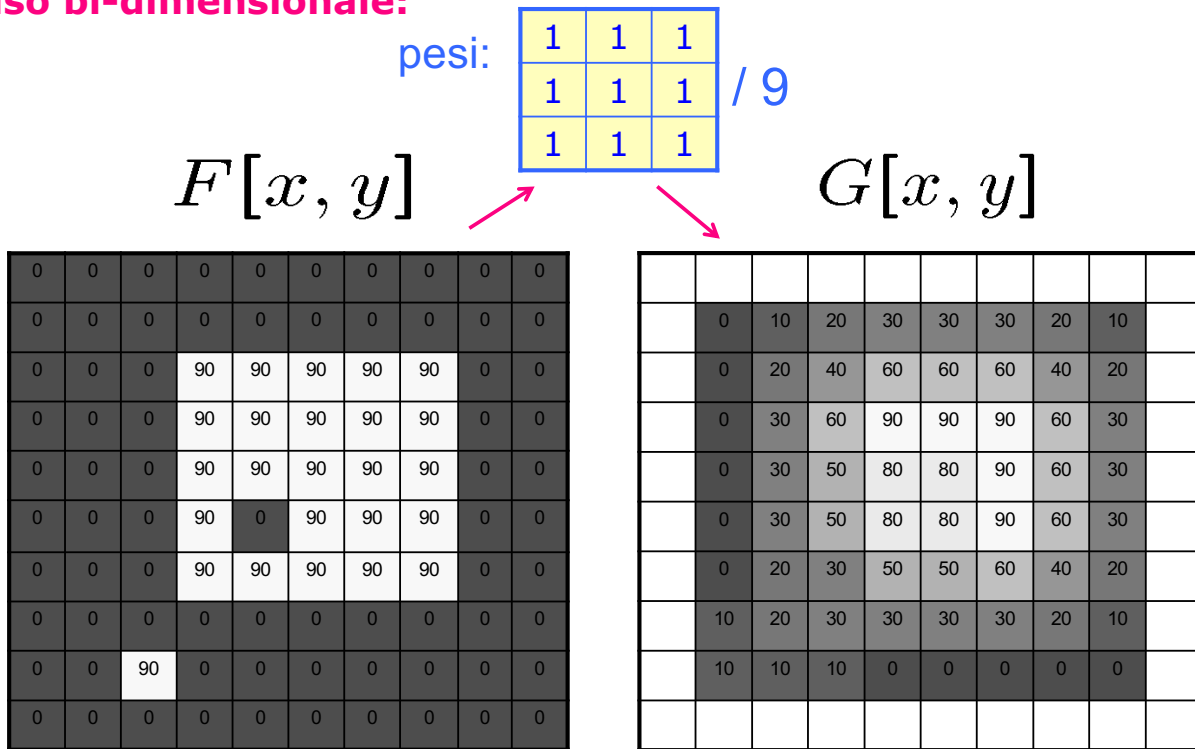
$G[x, y]$

	0	10	20	30	30				

source: S. Seitz



Caso bi-dimensionale:



source: S. Seitz

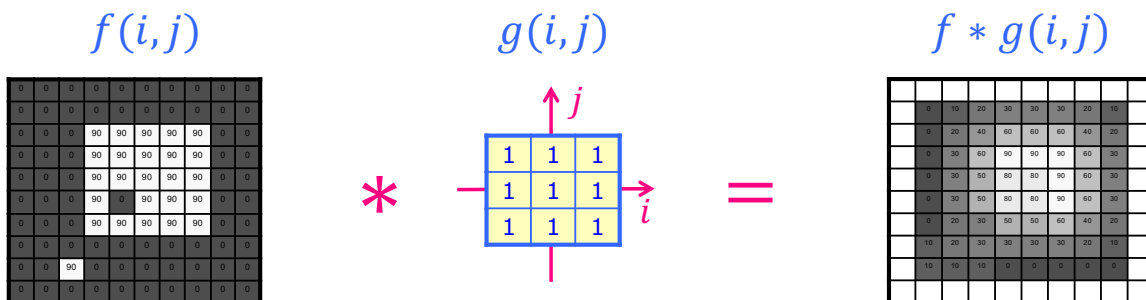
Convoluzione in 2D



La descrizione analitica di questa operazione è la

Convoluzione 2D (discreta):

$$(f * g)(i, j) = f(i, j) * g(i, j) = \sum_{h, k} g(h, k) f(i - h, j - k)$$



$$(f * g)(i, j) = \sum_{h, k=-1}^{+1} g(h, k) f(i - h, j - k)$$

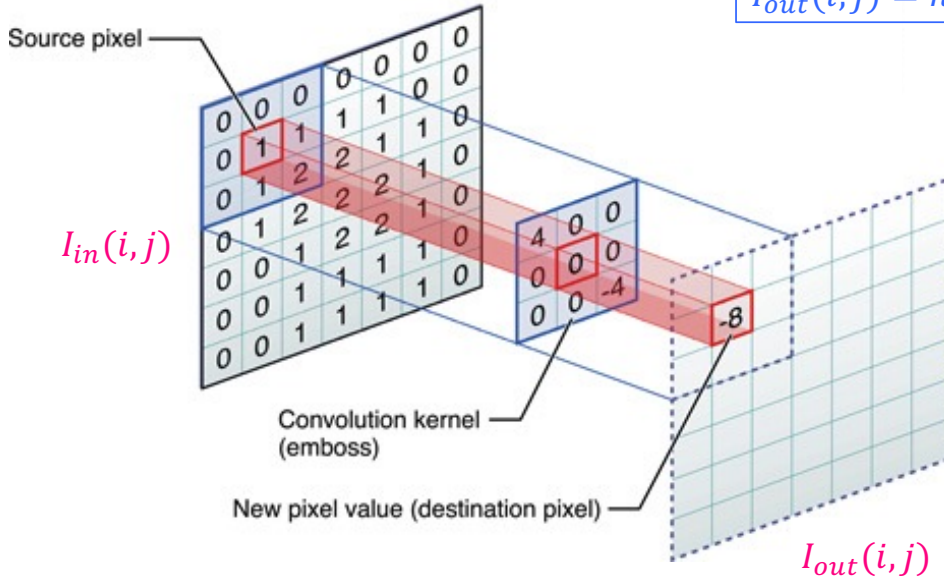
credits: F. Durand



La convoluzione (filtri lineari)



$$I_{out}(i,j) = h(i,j) * I_{in}(i,j)$$



Proprietà della convoluzione



Sistemi lineari spazio-invarianti: Filtri lineari



❖ Se un sistema $S[\cdot]$ gode delle proprietà di:

❖ **Linearità:** $S(f(i,j) + g(i,j)) = S(f(i,j)) + S(g(i,j))$

❖ **Spazio-invarianza:** $S(f(i,j)) = h(i,j) \rightarrow S(f(i-h, j-k)) = h(i-h, j-k)$

Ogni operatore lineare spazio-invariante può essere definito come una convoluzione dell'ingresso con la sua **risposta all'impulso** $h(i,j)$:

$$I_{out}(i,j) = h(i,j) * I_{in}(i,j)$$

Proprietà della convoluzione:

❖ **commutativa:** $a * b = b * a$

➢ Non c'è differenza concettuale tra immagine e filtro

❖ **associativa:** $a * (b * c) = (a * b) * c$

➢ Applicare filtri in cascata è equivalente a un filtro costituito dalla cascata di tutti

$$(((a * b_1) * b_2) * b_3) = a * (b_1 * b_2 * b_3)$$

❖ **distributiva:** $a * (b + c) = (a * b) + (a * c)$

❖ **linearità:** $ka * b = a * kb = k(a * b)$

❖ **identità:** impulso unitario: $\delta = [\dots, 0, 0, 1, 0, 0, \dots] \rightarrow a * \delta = a$

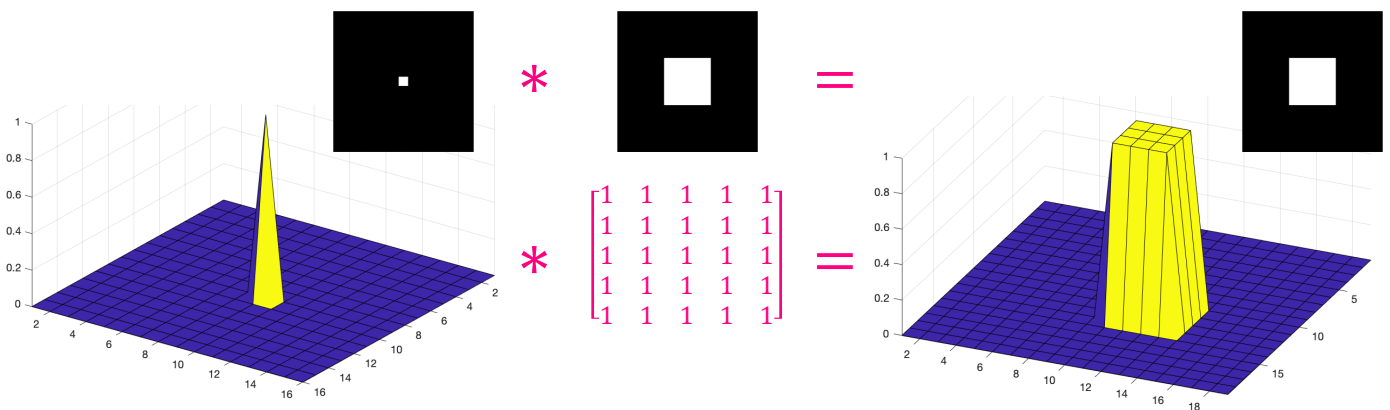
Data un'immagine $f(i, j)$
 e un sistema con risposta all'impulso $h(i, j)$,
 la risposta del sistema all'immagine
 (risultato del filtraggio): $g(i, j)$



$$g(i, j) = h(i, j) * f(i, j)$$

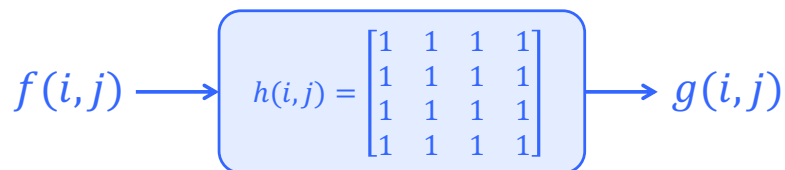
$$(f * h)(i, j) = \sum_{h, k} h(h, k) f(i - h, j - k)$$

❖ $h(i, j)$: **risposta all'impulso** del sistema (filtro)

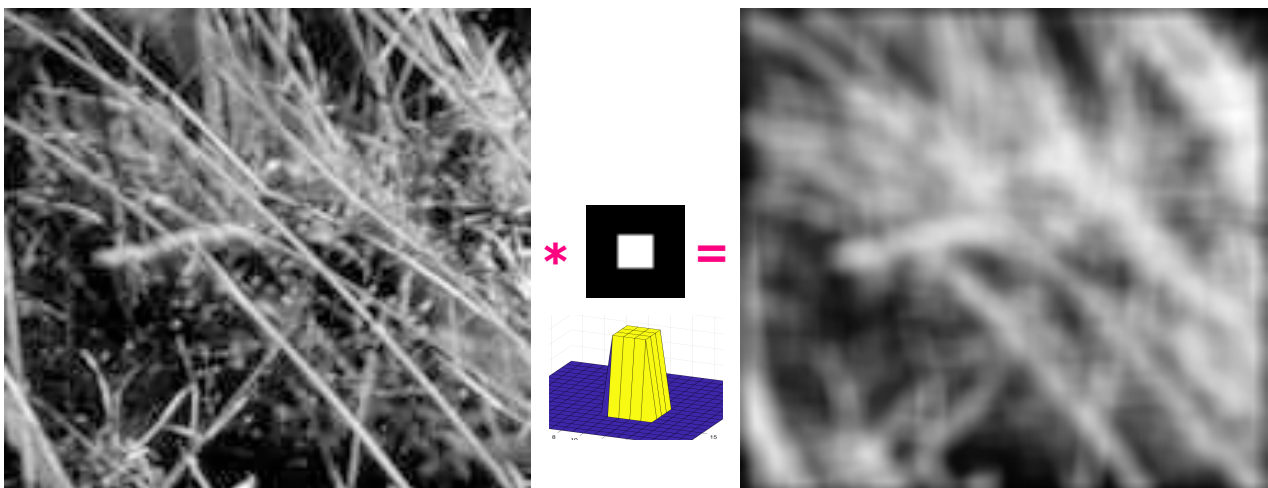


Smoothing – raffinamento

Box filter: convoluzione con risposta all'impulso "box"



$$g(i, j) = h(i, j) * f(i, j)$$

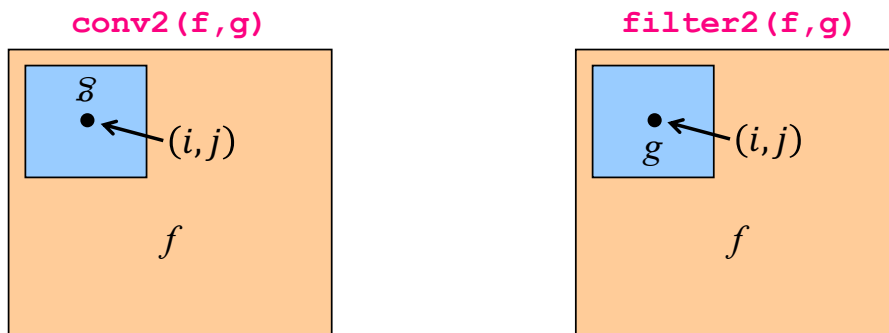


credits:
D.Forsyth



Operatori di **convoluzione 2D** in **MATLAB**:

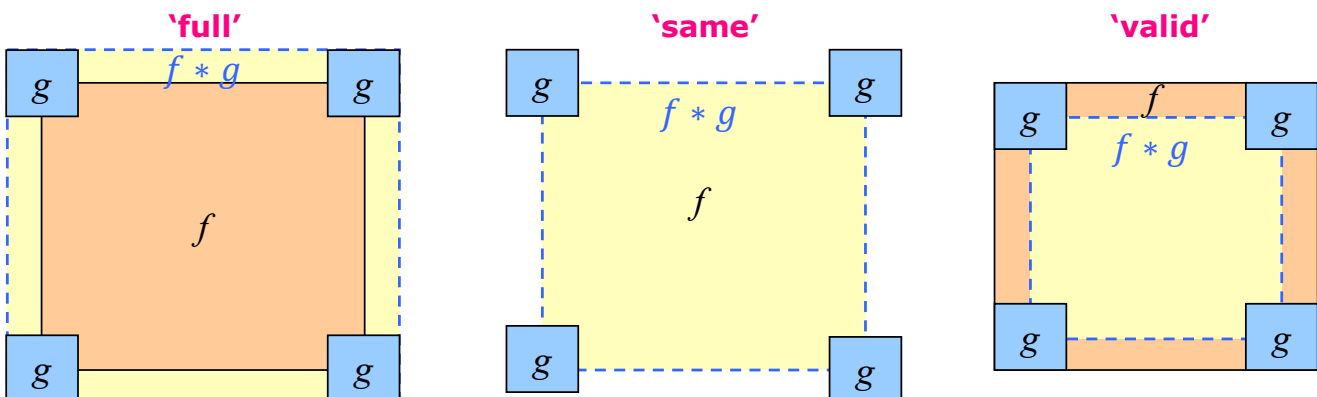
- ❖ `conv2(f, g, 'shape')` $C(i,j) = \sum_{h,k} f(i-h, j-k) \cdot g(h,k)$
- ❖ `filter2(f, g, 'shape')`
`xcorr2(f, g)` $C(i,j) = \sum_{h,k} f(i+h, j+k) \cdot g(h,k)$
- ❖ se $g(i,j)$ è *simmetrico* \rightarrow `conv2()` e `filter2()` coincidono



Convoluzione 2D: effetti di bordo



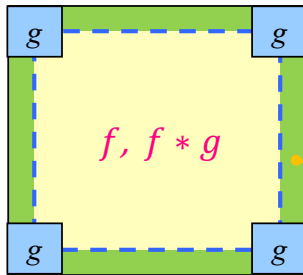
- ❖ Operatori di convoluzione 2D in MATLAB:
`conv2(f, g, 'shape')`
`filter2(f, g, 'shape')`
- ❖ che dimensione ha l'immagine (matrice) risultante?
 - `shape = 'full'`: output size is **sum of sizes** of **f** and **g**
 - `shape = 'same'`: output size is **same as f** (default)
 - `shape = 'valid'`: output size is **difference of sizes** of **f** and **g**



source: L. Cinque



`imfilter(f, g) = conv2(f, g, 'same')` + gestione effetti di bordo:



❖ Come riempio il bordo?

- clip filter (black):
`imfilter(f, g)`
- wrap around:
`imfilter(f, g, 'circular')`
- copy edge:
`imfilter(f, g, 'replicate')`
- reflect across edge:
`imfilter(f, g, 'symmetric')`

credits: S. Marschner

Smoothing – defocusing



Defocusing

Lo smoothing del box filter **non** si comporta esattamente come uno 'sfuocamento' della lente

- ❖ Lo sfuocamento della lente corrisponde alla **convoluzione con la MTF** (Modulation Transfer Function) della lente
- ❖ **MTF** – Modulation Transfer Function: risposta all'impulso della lente

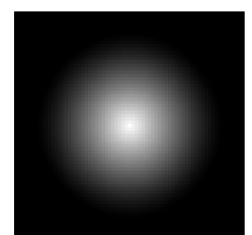
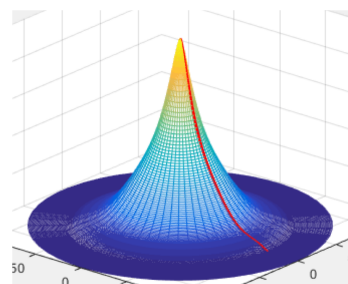
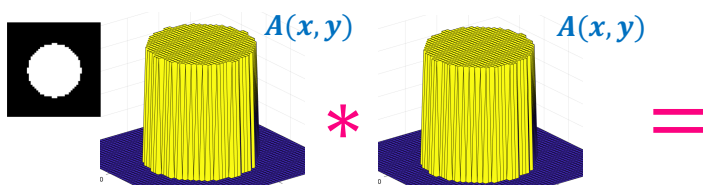
$A(x, y)$: lens aperture

$$MTF(x, y) = A(x, y) * A(x, y)$$



➔ MTF assomiglia molto a una Gaussiana 2D!

MTF – circular aperture



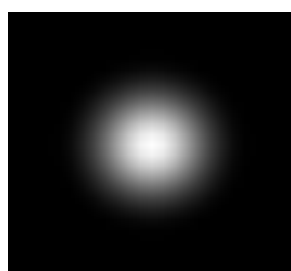
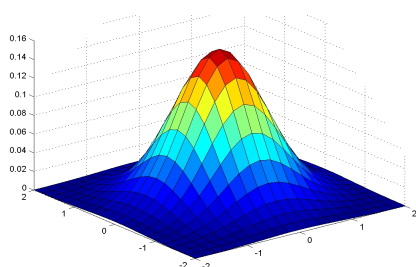
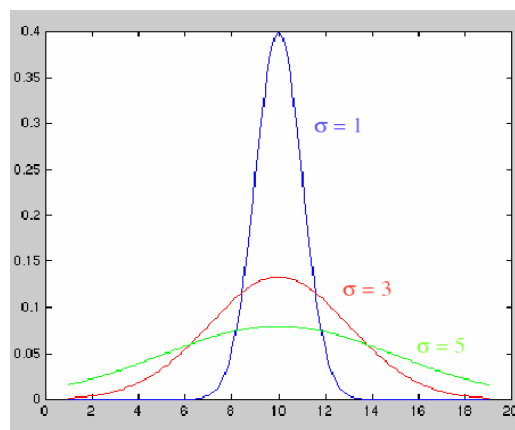


Filtraggio gaussiano

Risposta all'impulso: **gaussiana 2D**

- ❖ il coefficiente rende unitaria l'area della gaussiana → la convoluzione non "amplifica" l'immagine

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



$\sigma = 1$ pixel – 5 x 5 window

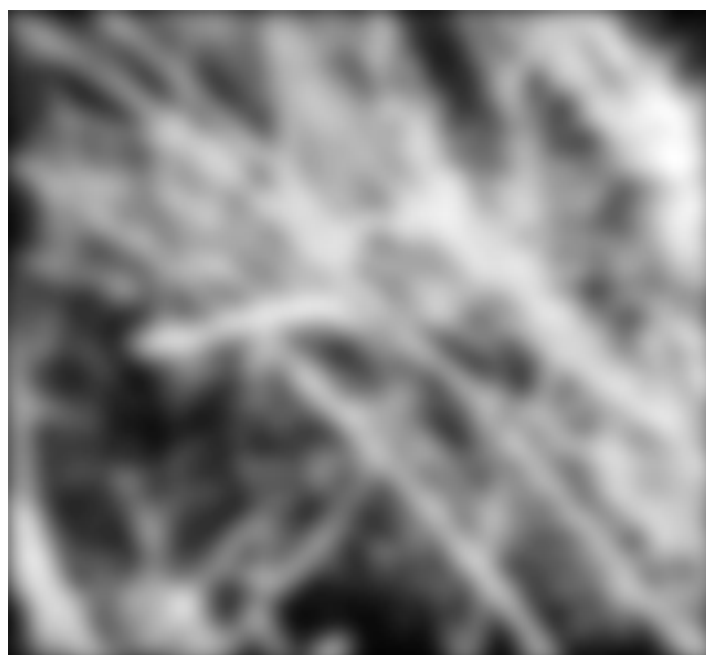
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

credit: C. Rasmussen



source: L. Cinque

- ❖ kernel: Gaussiana 2D





Proprietà del filtraggio gaussiano:

- ❖ **Buon filtro "passa-basso"**
 - decadimento veloce della risposta in frequenza
- ❖ **Convoluzione del kernel con se stesso ancora Gaussiano**
 - Due convoluzioni in cascata con un kernel gaussiano di dev. std.: σ è equivalente a convolvere una volta con un kernel di dev. std.: $\sqrt{2} \sigma$
- ❖ **Kernel separabile:**
 - Kernel 2D separabile in due kernel 1D!
 - 1 convoluzione 2D \rightarrow 2 convoluzioni 1D in cascata (righe \rightarrow colonne)
 - Complessità di calcolo si riduce (da $\mathbf{N^2 M^2}$ a $\mathbf{2N M^2}$)

$$\begin{aligned}
 G_{\sigma}(x, y) &= \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \\
 &= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right)\right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{y^2}{2\sigma^2}\right)\right)
 \end{aligned}$$

2D convolution
(center location only)

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix}$$

The filter factors
into a product of 1D
filters:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Perform convolution
along rows:

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix} = \begin{bmatrix} & & \\ 11 & 18 & 18 \\ & & \end{bmatrix}$$

Followed by convolution
along the remaining column:

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}^T * \begin{bmatrix} & & \\ 11 & 18 & 18 \\ & & \end{bmatrix}^T = \begin{bmatrix} & & \\ & 65 & \\ & & \end{bmatrix}$$

source: K. Grauman

Gaussian filters

*Filtraggio
gaussiano 2D
con kernel 1D*

- ❖ immagine: $M \times M$
- ❖ filtro: $N \times N$

Complessità:

- ❖ kernel 2D: $N^2 M^2$
- ❖ kernel 1D: $2N M^2$

Esempio kernel 1D:

- ❖ 25 'tap' ($N=25$)
- ➔ $625 M^2 \rightarrow 50 M^2$
- (speed-up: 12,5x)

[MATLAB LIVE Script:
GaussianFiltering](#)



Originale



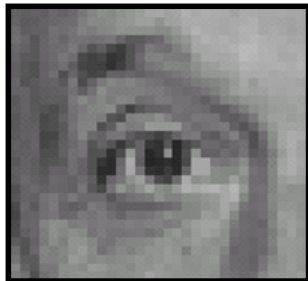
Filtraggio 1D orizzontale



Filtraggio 1D verticale



Filtraggio 2D

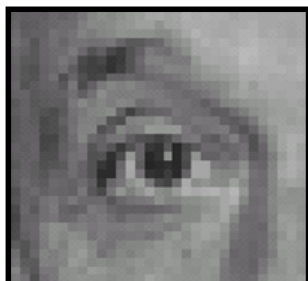


Original

0	0	0
0	1	0
0	0	0

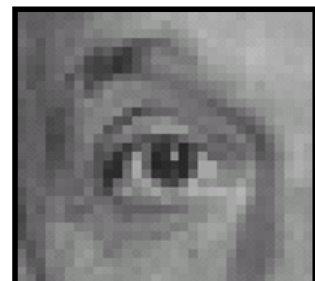
?

source: D. Lowe



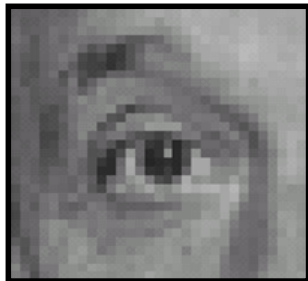
Original

0	0	0
0	1	0
0	0	0



Filtered
(no change)

source: D. Lowe

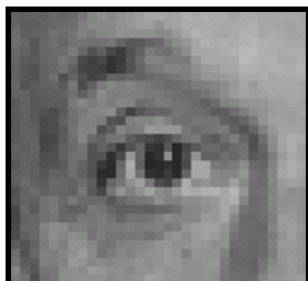


Original

0	0	0
0	0	1
0	0	0

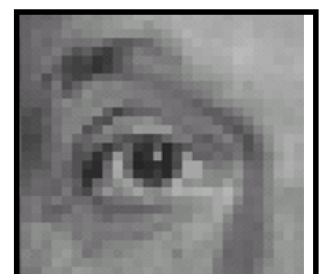
?

source: D. Lowe



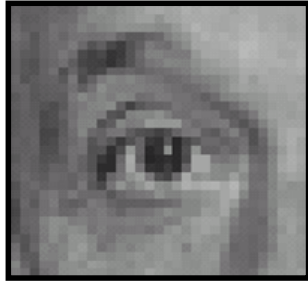
Original

0	0	0
0	0	1
0	0	0



Shifted left
by 1 pixel

source: D. Lowe

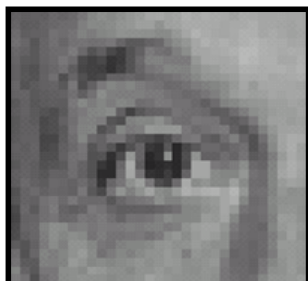


Original

1	1	1
1	1	1
1	1	1

?

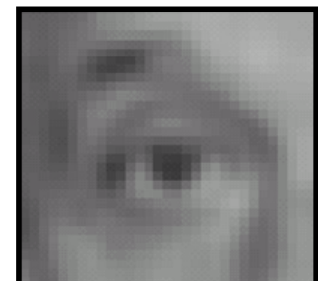
source: D. Lowe



Original

1	1	1
1	1	1
1	1	1

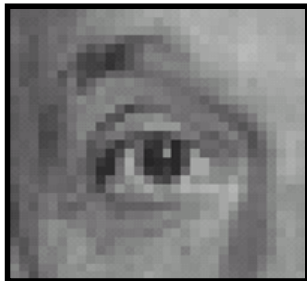
9



Blurring
(with a box filter)

coefficiente di normalizzazione
→ area unitaria

source: D. Lowe



Original

0	0	0
0	2	0
0	0	0

-

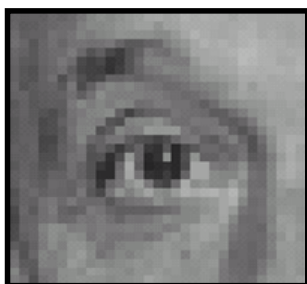
1	1	1
1	1	1
1	1	1

?

9

Il filtro risultante ha area unitaria

source: D. Lowe



Original

0	0	0
0	2	0
0	0	0

-

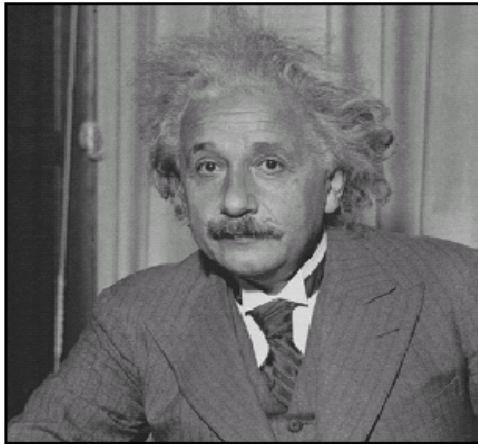
1	1	1
1	1	1
1	1	1



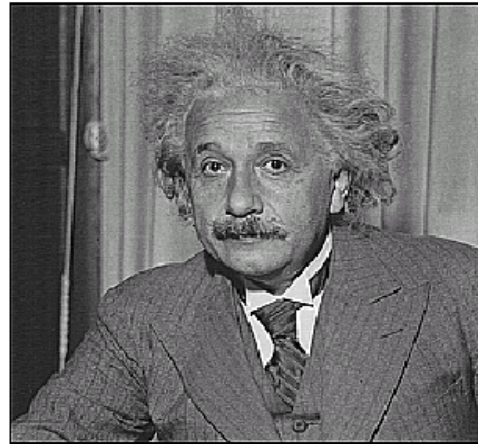
9

Sharpening filter:
Accentua le differenze rispetto alla media locale

source: D. Lowe



before



after

source: D. Lowe

Sharpening revisited

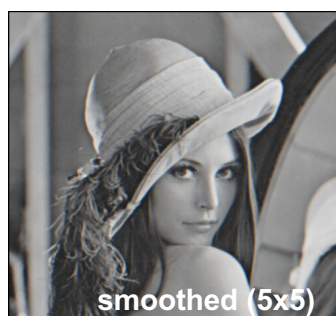


What does blurring take away?



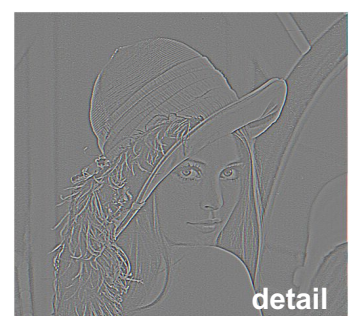
original

–



smoothed (5x5)

=



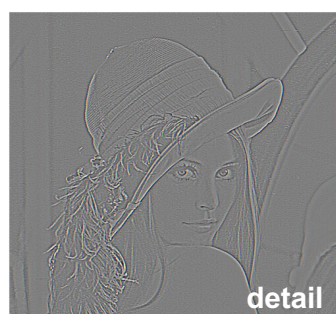
detail

Let's add it back:



original

+ α

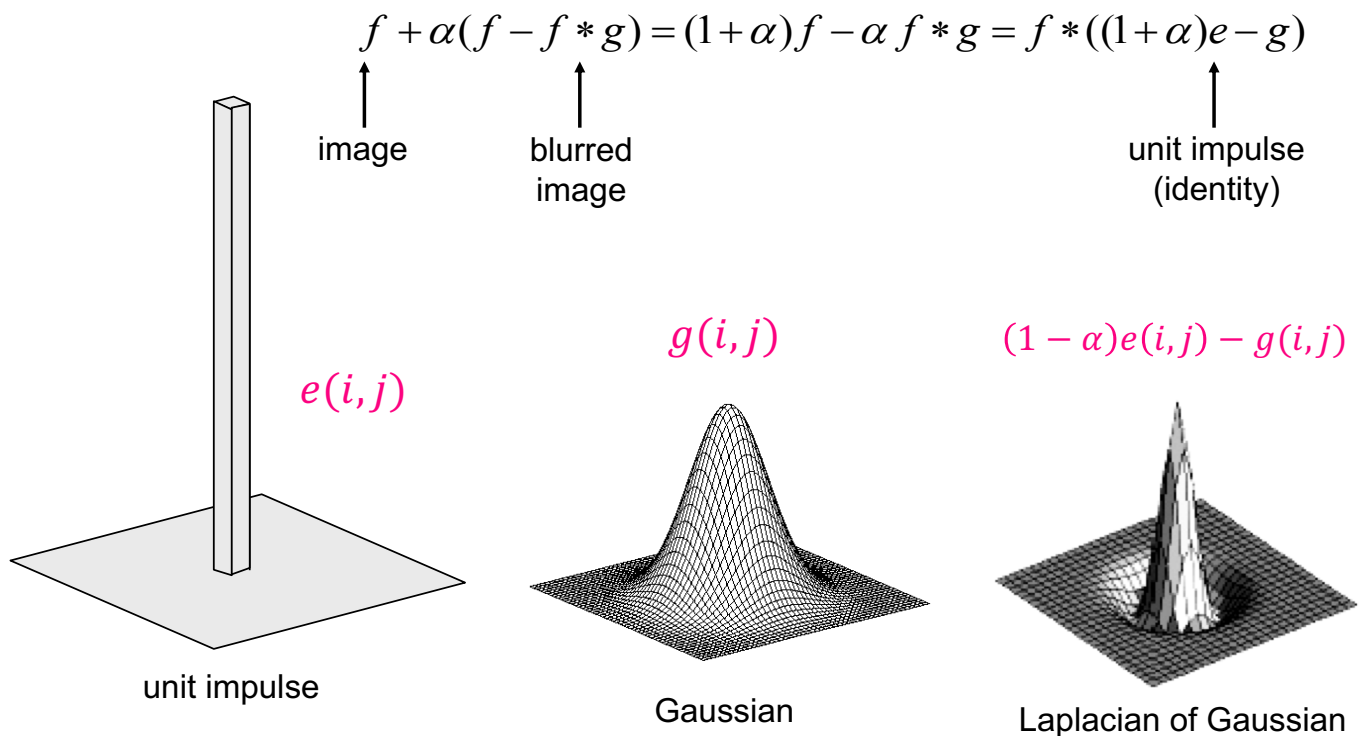


detail

=



sharpened



source: L. Cinque



Filtraggio lineare di immagini

- ❖ Filtraggio lineare nel dominio dello spazio
 - convoluzione 2D
 - esempi di filtraggio lineare (blurring, sharpening)
- ❖ Filtraggio nel dominio delle frequenze
 - trasformata di Fourier 2D, spettro di un'immagine
 - filtraggio nel dominio delle frequenze
 - campionamento spaziale 2D / spettro / aliasing / Moiré
 - Applicazione: sotto-/sovra-campionamento

(Forsyth/Ponce: Capitolo 4)

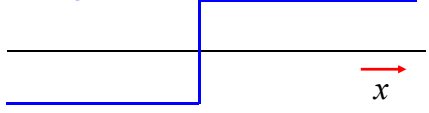
Slide credits: varie sorgenti (citare)



Trasformata di Fourier:

- ❖ rappresentazione di una **funzione periodica** come **somma di sinusoidi**
 - a frequenze multiple della frequenza fondamentale (armoniche)

Spatial frequency analysis of a step edge

$$f(x) = \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{otherwise} \end{cases}$$


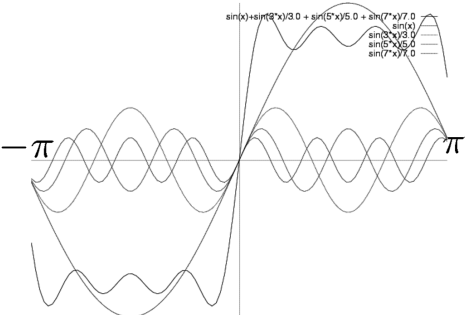
Fourier decomposition

Fourier Series

$$f(x) = \sum_n a_n \sin nx$$

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin nx \, dx$$

$$= \frac{2}{\pi} \int_0^{\pi} \sin nx \, dx = \begin{cases} \frac{4}{n\pi} & \text{if } n \text{ odd} \\ 0 & \text{otherwise} \end{cases}$$

$$f(x) = \sum_{n=1}^{\infty} \frac{4}{(2n-1)\pi} \sin(2n-1)x$$


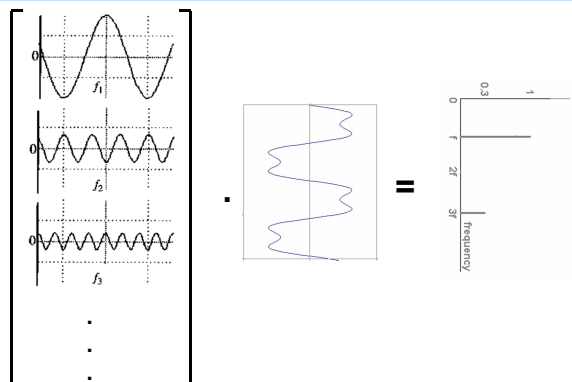
slide by
A. Zisserman



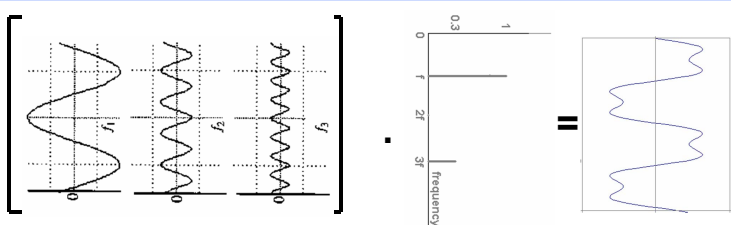
Trasformata di Fourier

- ❖ può essere vista come un "cambio di base":

Analisi



Sintesi



slides by
A. Zisserman

Analisi: dalla funzione allo spettro

$$S(f) = \int_{-\infty}^{+\infty} s(x) e^{-j2\pi f x} \, dx$$

Sintesi: dallo spettro alla funzione

$$s(x) = \int_{-\infty}^{+\infty} S(f) e^{+j2\pi f x} \, df$$

f: frequenza – dimensione reciproca a quella del dominio: tempo [s] → frequenza [Hz]
spazio [m] → frequenza spaziale [m⁻¹]



Immagine: funzione su **dominio spaziale 2D**

$$f: \mathfrak{R}^2 \rightarrow \mathfrak{R} \quad \mapsto \quad x, y \rightarrow f(x, y) \quad x, y: [\text{m} \mid \text{pixel} \mid \dots]$$

❖ **Spettro: funzione su dominio delle frequenze spaziali**

$$x, y \rightarrow f(x, y) \quad \xleftrightarrow{\mathfrak{S}} \quad u, v \rightarrow F(u, v) \quad u, v: [\text{m}^{-1} \mid \text{pixel}^{-1} \mid \dots]$$

❖ **Trasformata di Fourier 2D:**

$$f(x, y) \quad \xleftrightarrow{\mathfrak{S}} \quad \mathfrak{S}[f(x, y)] = F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j2\pi(ux+vy)} dx dy$$

- u, v : frequenze spaziali [cicli/m, cicli/pixel]
- $e^{-j2\pi(ux+vy)}$: esponenziale complesso (fasore)

$$e^{-j2\pi(ux+vy)} = \cos 2\pi(ux + vy) - j \sin 2\pi(ux + vy)$$

❖ **Operatore inverso: antitrasformata di Fourier 2D:**

$$F(u, v) \quad \xleftrightarrow{\mathfrak{S}^{-1}} \quad \mathfrak{S}^{-1}[F(u, v)] = f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) e^{j2\pi(ux+vy)} du dv$$

Trasformata di Fourier 2D

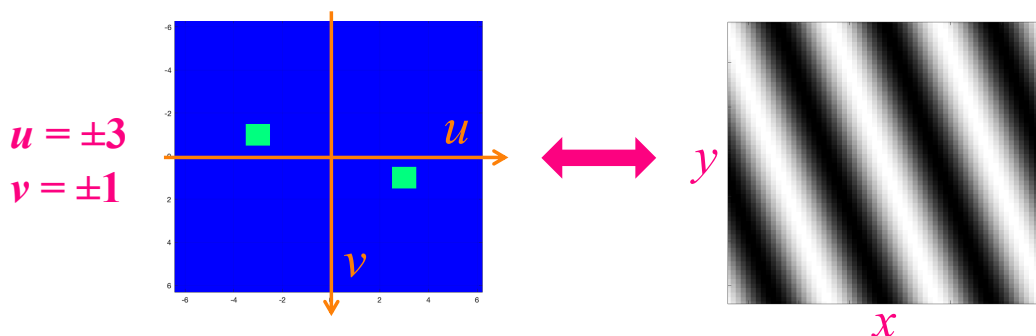
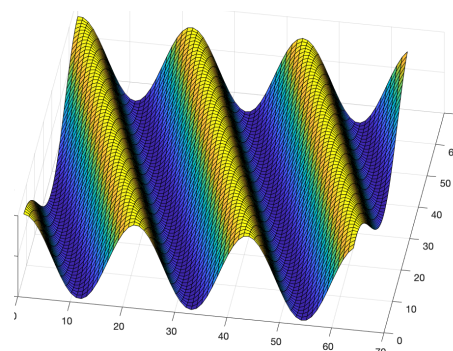


Trasformata di Fourier 2D:

- ❖ rappresentazione di una funzione 2D come combinazione lineare di sinusoidi/cosinusoidi

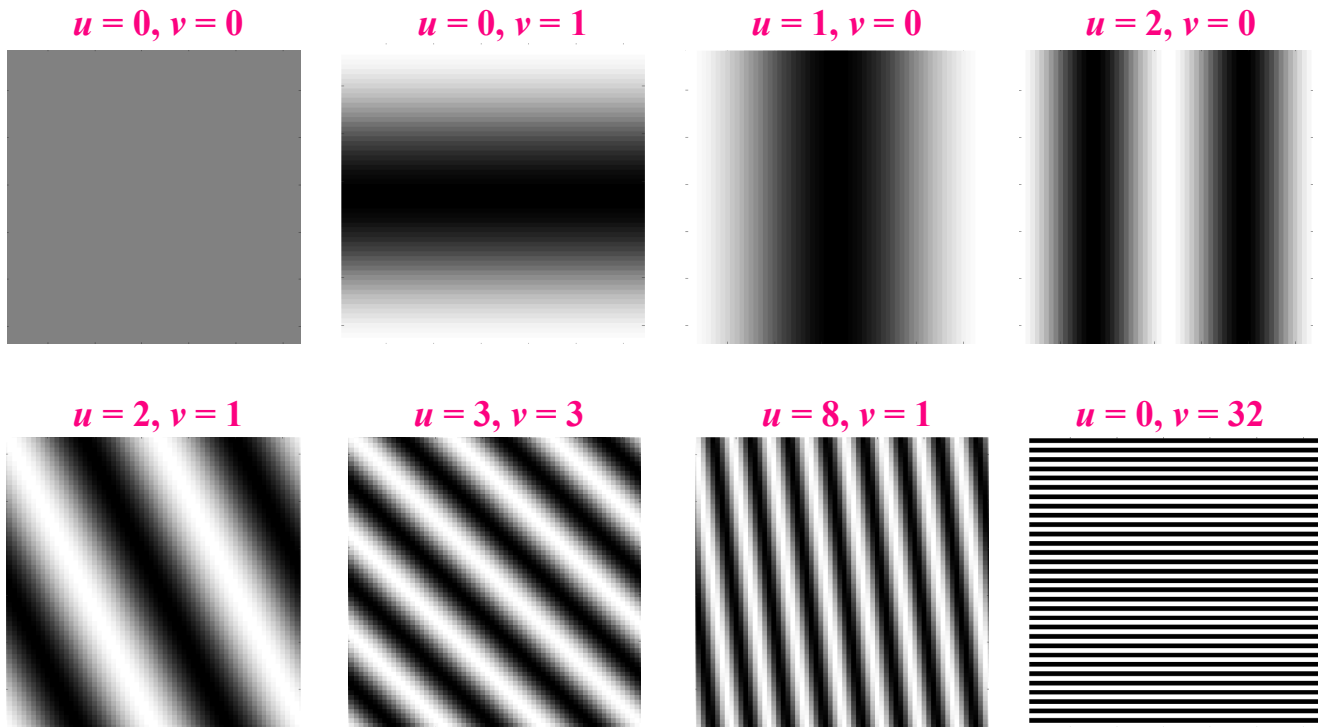
Sinusoide 2D – $S_{u,v}(x, y)$

$$S_{u,v}(x, y) = e^{-j2\pi(ux+vy)} = \cos(2\pi(ux + vy)) + j \sin(2\pi(ux + vy))$$





❖ Funzioni-base (cosinusoidi)



Trasformata di Fourier 2D



Fourier Transform pairs (2D):

Proprietà e
coppie notevoli
trasformata/
antitrasformata

impulso:
derivata:
coseno:
Gaussiana:
box – (rect() in 2D):
scalatura:
'pettine' di impulsi:
convoluzione/prodotto:
traslazione:
rotazione:

Function	Fourier transform
$g(x, y)$	$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x, y) e^{-i2\pi(ux+vy)} dx dy$
$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathcal{F}(g(x, y))(u, v) e^{i2\pi(ux+vy)} du dv$	$\mathcal{F}(g(x, y))(u, v)$
impulso: $\delta(x, y)$	1
derivata: $\frac{\partial f}{\partial x}(x, y)$	$u\mathcal{F}(f)(u, v)$
coseno: $0.5\delta(x + a, y) + 0.5\delta(x - a, y)$	$\cos 2\pi au$
Gaussiana: $e^{-\pi(x^2+y^2)}$	$e^{-\pi(u^2+v^2)}$
box – (rect() in 2D): $box_1(x, y)$	$\frac{\sin u}{u} \frac{\sin v}{v}$
scalatura: $f(ax, by)$	$\frac{\mathcal{F}(f)(u/a, v/b)}{ab}$
'pettine' di impulsi: $\sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \delta(x - i, y - j)$	$\sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \delta(u - i, v - j)$
convoluzione/prodotto: $(f * g)(x, y)$	$\mathcal{F}(f)\mathcal{F}(g)(u, v)$
traslazione: $f(x - a, y - b)$	$e^{-i2\pi(au+bv)} \mathcal{F}(f)$
rotazione: $f(x \cos \theta - y \sin \theta, x \sin \theta + y \cos \theta)$	$\mathcal{F}(f)(u \cos \theta - v \sin \theta, u \sin \theta + v \cos \theta)$

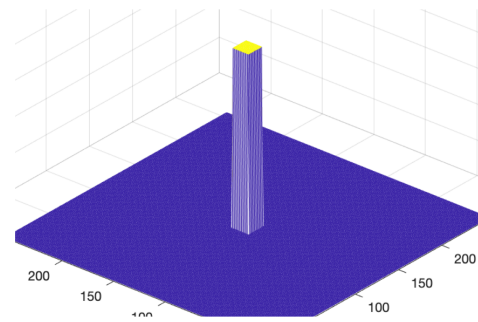


Fourier Transform pairs (2D):

Coppie notevoli trasformata/antitrasformata

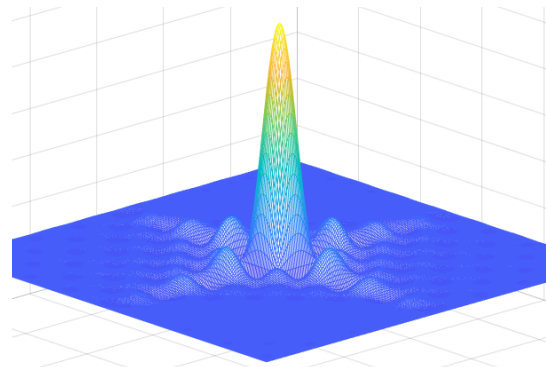
❖ **Box 2D** (versione 2D di: $rect(x)$):

$$f(x, y) = \text{box}\left(\frac{x}{a}, \frac{y}{b}\right) = \begin{cases} 1 & |x| < \frac{a}{2}, |y| < \frac{b}{2} \\ 0 & \text{altrove} \end{cases}$$



❖ **Sinc 2D:**

$$F(u, v) = \text{sinc}(a u) \cdot \text{sinc}(b v)$$



Fourier Transform pairs (2D):

Coppie notevoli trasformata/antitrasformata

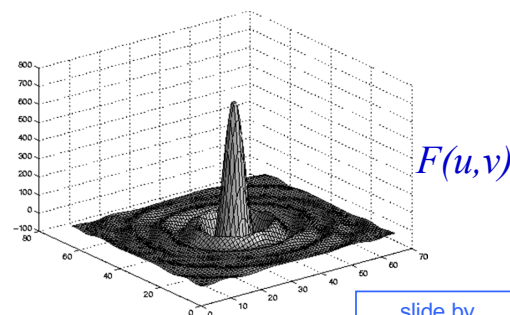
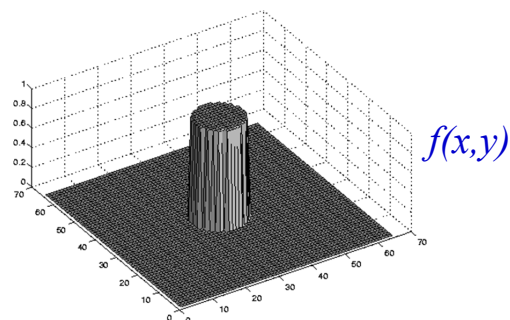
Circular disk unit height and radius a centred on origin

$$f(x, y) = \begin{cases} 1, & |r| < a, \\ 0, & |r| \geq a. \end{cases}$$

$$F(u, v) = F(\rho) = a J_1(\pi a \rho) / \rho$$

where $J_1(x)$ is a Bessel function.

- rotational symmetry
- a '2D' version of a sinc



slide by
A. Zisserman

Fourier Transform pairs (2D):

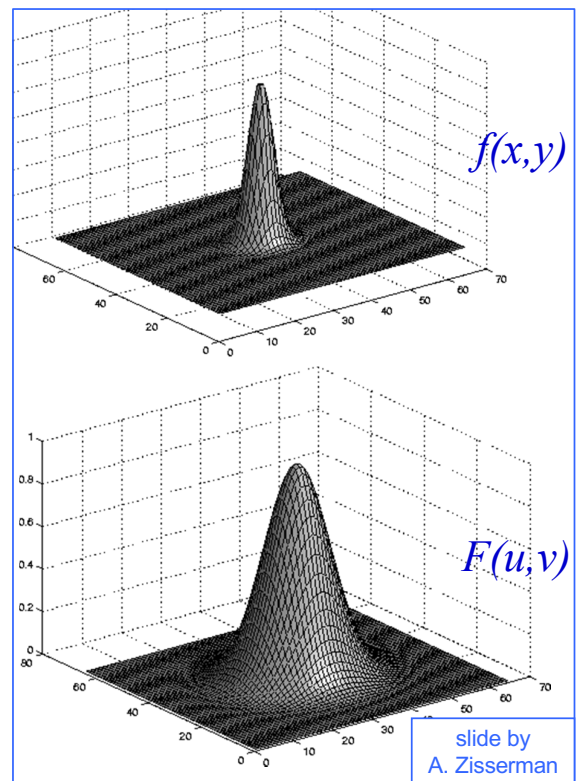
Coppie notevoli trasformata/antitrasformata

- ❖ **Gaussiana 2D** – varianza σ^2 :

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

- ❖ **Gaussiana 2D** – varianza $1/\sigma^2$:

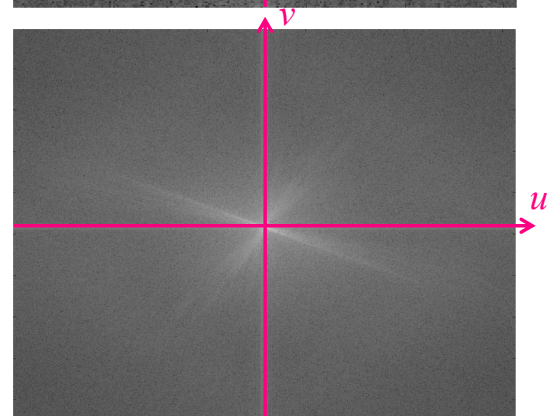
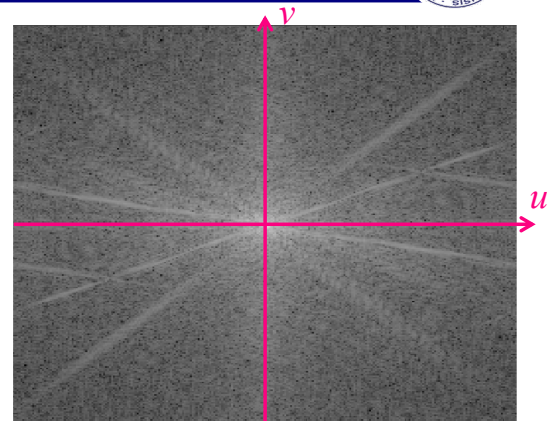
$$F(u, v) = e^{-2\pi^2\sigma^2(x^2+y^2)}$$



Trasformata di Fourier 2D

Esempi:

L: immagine
R: |spettro|





Campionamento 2D: discretizzazione del dominio

- ❖ Immagini digitali: dominio **discreto**

$$f(x, y) \rightarrow f(i, j)$$

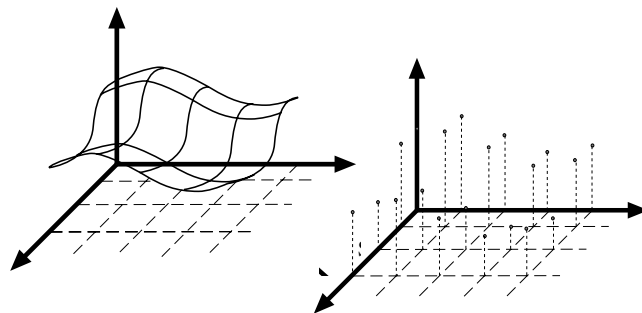
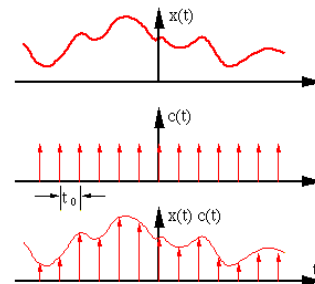
Campionamento: discretizzazione del dominio

- ❖ **In 1D:**

$$\begin{aligned} f_s(i) &= f(x = i \Delta_x) = \sum_h f(x) \cdot \delta(x - h \Delta_x) \\ &= f(x) \sum_h \delta(x - h \Delta_x) \end{aligned}$$

- ❖ **In 2D:**

$$\begin{aligned} f_s(i, j) &= f(x = i \Delta_x, y = j \Delta_y) \\ &= \sum_h \sum_k f(x, y) \delta(x - h \Delta_x, y - k \Delta_y) \\ &= f(x, y) \sum_h \sum_k \delta(x - h \Delta_x, y - k \Delta_y) \end{aligned}$$



Campionamento e ricostruzione



Come rappresento la trasformata di Fourier dell'immagine campionata?

- ❖ L'immagine digitale $f(i, j)$ è campionata, quindi a dominio **discreto** e **limitato**
 → ha trasformata su un dominio **continuo** e **illimitato nelle frequenze**

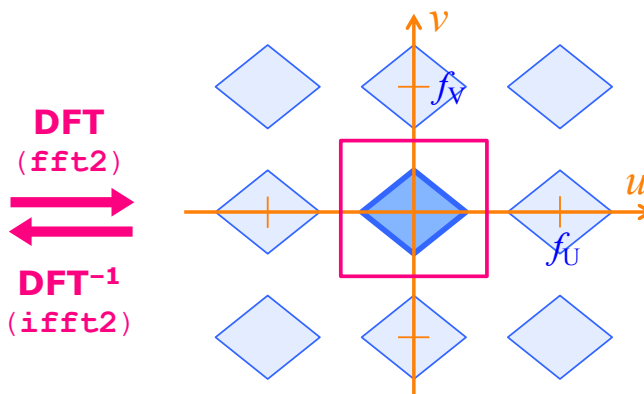
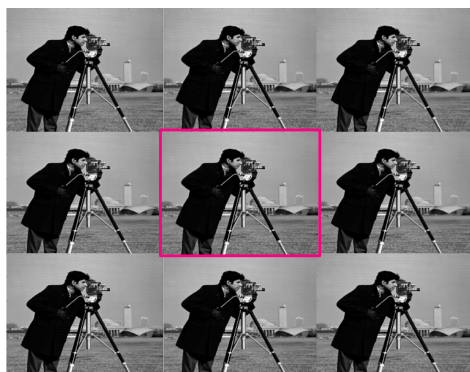
$f(i, j)$: dominio discreto e limitato → $F(u, v)$: dominio continuo e illimitato

➤ $F(u, v)$ non è rappresentabile numericamente!

- ❖ **Soluzione: periodicizzo l'immagine digitale**

Immagine digitale **discreta e periodica** → spettro **periodico e discreto**

$f(i, j)$: dominio discreto e limitato (1 periodo) → $F(u, v)$: dominio discreto e limitato



DFT
(fft2)
 ⇌
DFT-1
(ifft2)



DFT/FFT: Discrete/Fast Fourier Transform

- ❖ Immagine: $M \times N$ punti (pixel)
- ❖ Spettro: $M \times N$ punti

$$f(i, j) \xleftrightarrow{DFT} F(u, v) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} f(i, j) e^{-j2\pi \left(\frac{iu}{M} + \frac{jv}{N} \right)}, \quad \begin{cases} 0 \leq u \leq M-1 \\ 0 \leq v \leq N-1 \end{cases}$$

$$F(u, v) \xleftrightarrow{DFT^{-1}} f(i, j) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} F(u, v) e^{j2\pi \left(\frac{iu}{M} + \frac{jv}{N} \right)}, \quad \begin{cases} 0 \leq i \leq M-1 \\ 0 \leq j \leq N-1 \end{cases}$$

Frequenze spaziali

- ❖ frequenza orizzontale u , u/M : $0 \leq u \leq M-1 \left[\frac{\text{cicli}}{\text{immagine } h.} \right]$ $0 \leq \frac{u}{M} \leq \frac{M-1}{M} \left[\frac{\text{cicli}}{\text{pixel } h.} \right]$
- ❖ frequenza verticale v , v/M : $0 \leq v \leq N-1 \left[\frac{\text{cicli}}{\text{immagine } v.} \right]$ $0 \leq \frac{v}{N} \leq \frac{N-1}{N} \left[\frac{\text{cicli}}{\text{pixel } v.} \right]$

Trasformata di Fourier 2D – frequenze spaziali

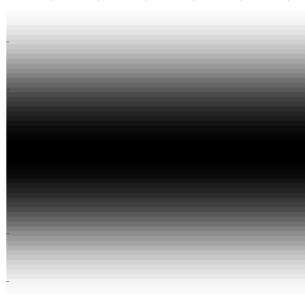


- ❖ Funzioni-base (cosinusoidi): u, v : cicli/immagine (M,N=64)

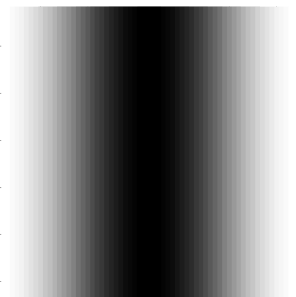
$u = 0, v = 0$



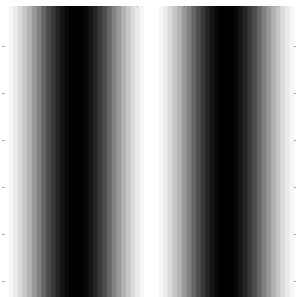
$u = 0, v = 1$



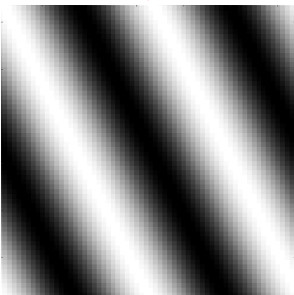
$u = 1, v = 0$



$u = 2, v = 0$



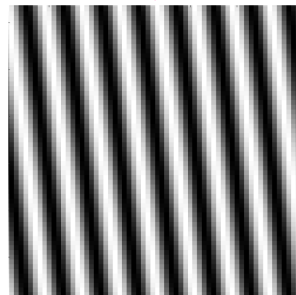
$u = 2, v = 1$



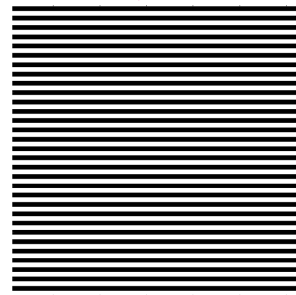
$u = 3, v = 3$



$u = 8, v = 1$

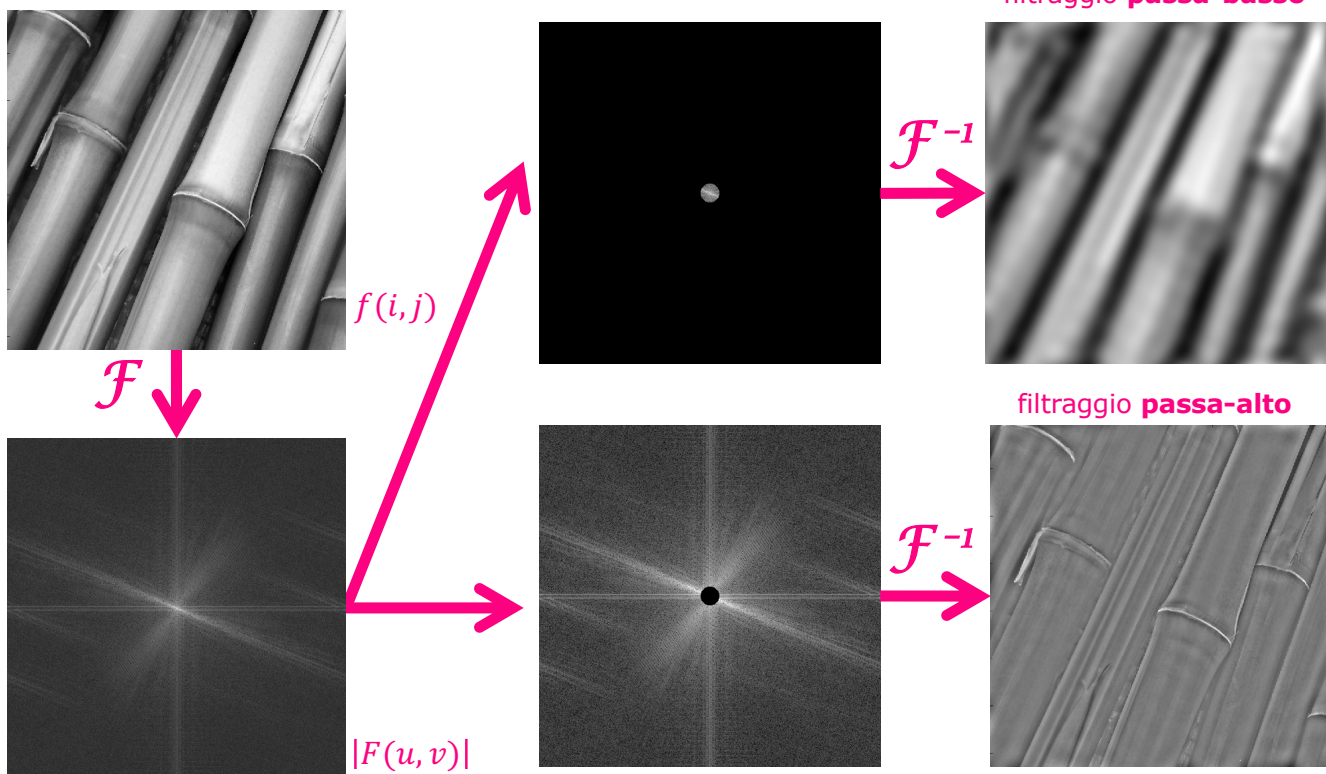


$u = 0, v = 32$





Applicazioni: filtri lineari, nel dominio della frequenza (spaziale)



Filtraggi nel dominio delle frequenze



Nel dominio degli spazi

Risposta $g(i,j)$ in uscita dal filtro $h(i,j)$:

- ❖ $h(i,j)$: risposta all'impulso del filtro

$$g(i,j) = f(i,j) * h(i,j) = \sum_{h,k} h(h,k) f(i-h, j-k)$$

Nel dominio delle frequenze spaziali

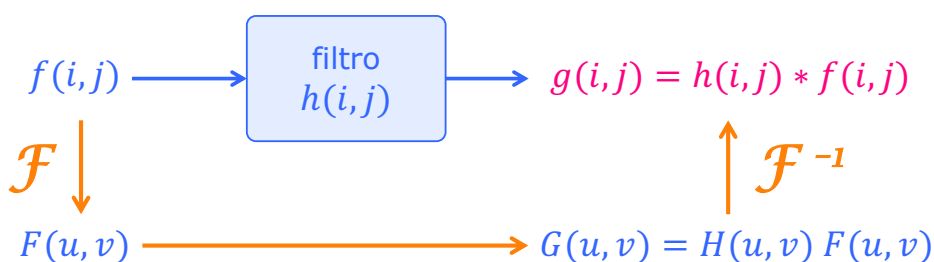
Spettro $G(u,v)$ in uscita dal filtro $H(u,v)$:

- ❖ $H(u,v)$: funzione di trasferimento o risposta in frequenza del filtro

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

Spazi: **Convoluzione:** $O(N^2 M^2)$

Frequenze: Trasformata \rightarrow **Prodotto** \rightarrow Antitrasformata: $O(N^2 + 2 N^2 \log(N^2)) = O(N^2 (1 + 4 \log(N)))$

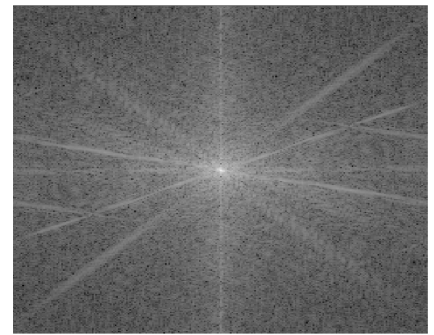




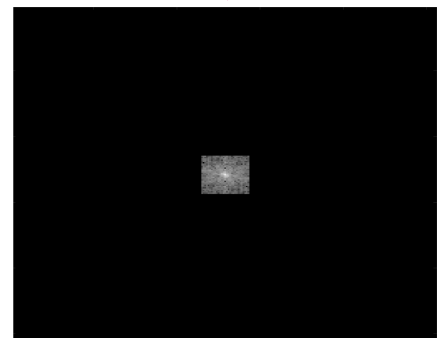
Filtraggio
passa-basso
in frequenza



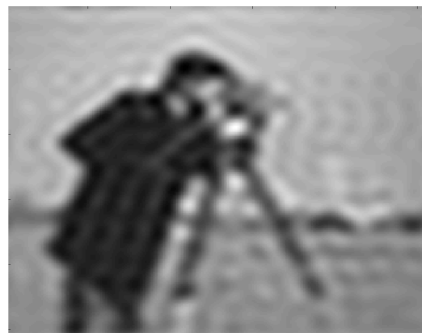
\mathcal{F}
→



prodotto ↓ $box\left(\frac{u,v}{16}\right)$



\mathcal{F}^{-1}
←



Complessità:

FFT: $N^2 \log(N^2)$

prodotto: N^2

IFFT: $N^2 \log(N^2)$

= $2 \cdot 256^2 \cdot 16$

+ 256^2

= $256^2 \cdot 33$

≈ $2 \cdot 10^6$



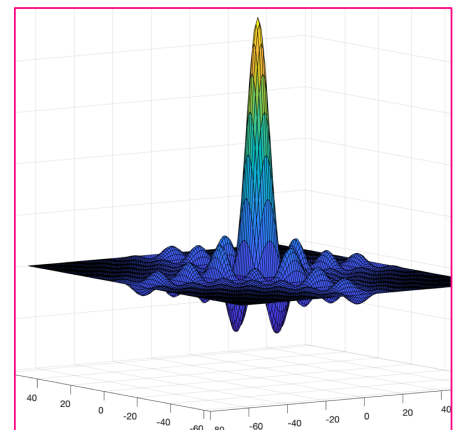
Filtraggio
passa-basso
in frequenza

❖ Analoga
operazione negli
spazi:

Convoluzione con



*



kernel di convoluzione
(128 x 128 punti)

$$H(u, v) = box\left(\frac{u, v}{16}\right)$$

↓

$$h(i, j) = \text{sinc}(16 i) \cdot \text{sinc}(16 j)$$

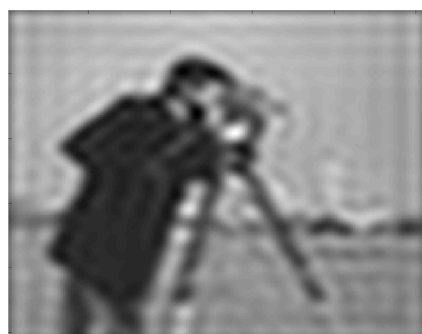
Complessità:

convoluzione:

$N^2 M^2$

= $256^2 \cdot 128^2$

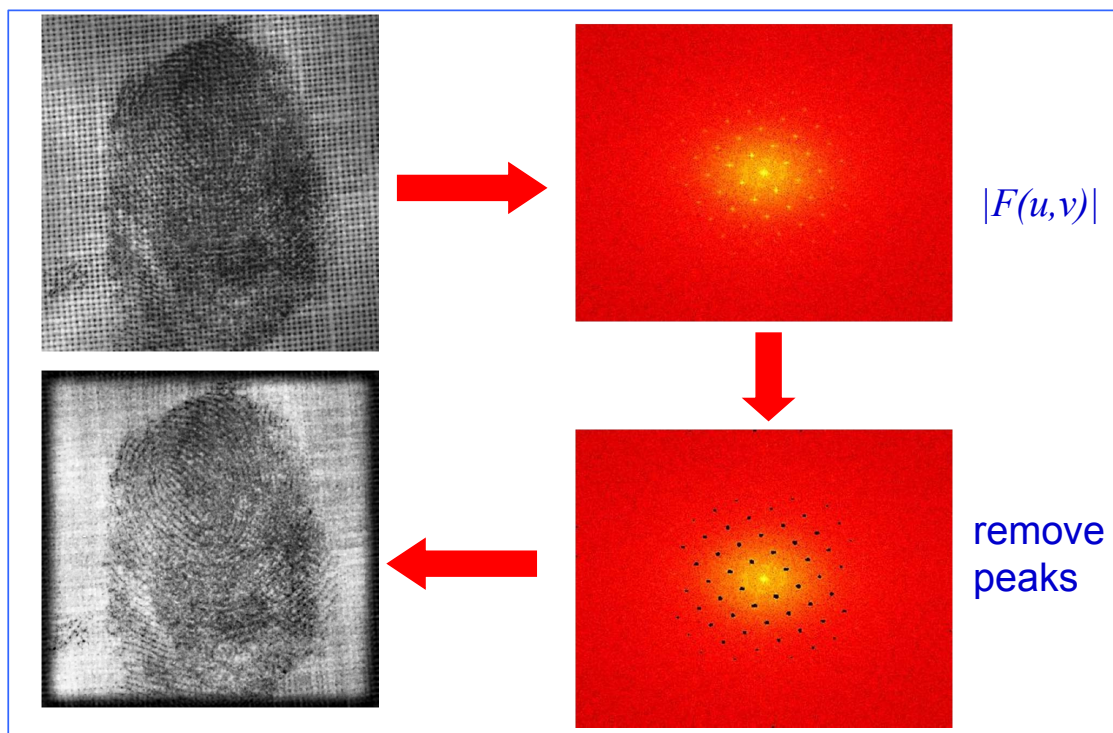
= $2^{30} \approx 10^9$





Esempi: applicazione forense

Rimozione pattern periodico del background



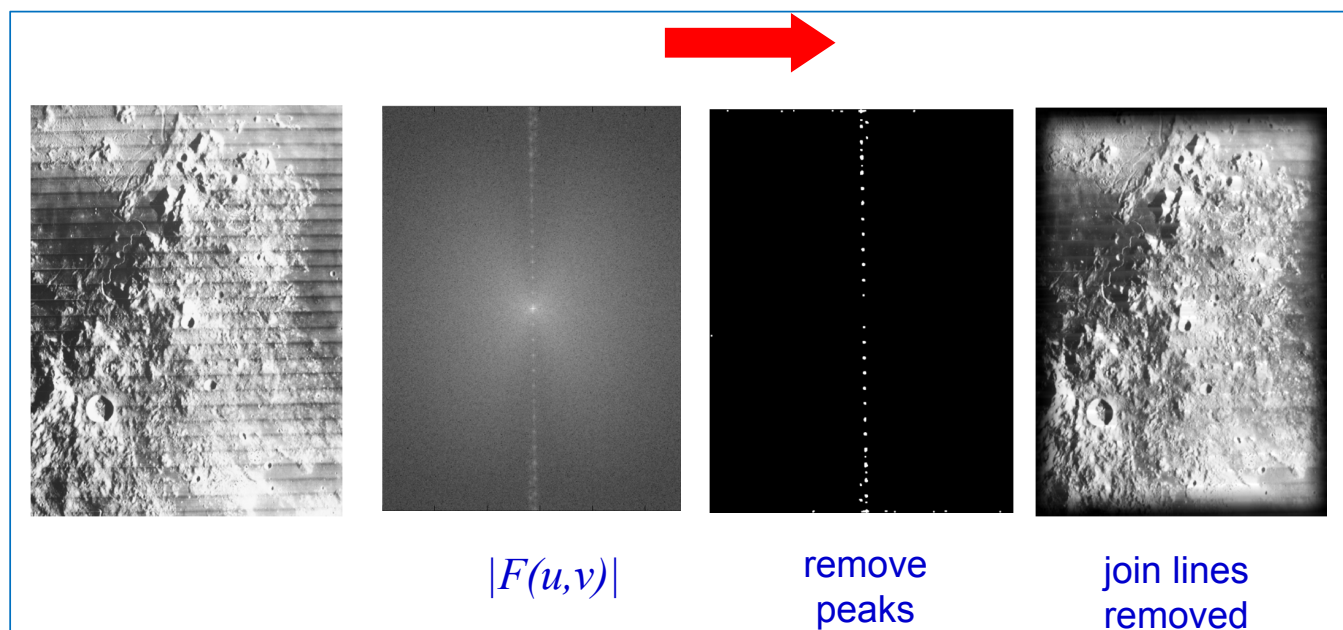
slide by
A. Zisserman



Esempi: elaborazione immagini

Immagine da modulo orbitante intorno alla Luna (1966)

slide by
A. Zisserman





Campionamento 2D: discretizzazione del dominio

In 2D: $f_s(i, j) = f(x = i\Delta_x, y = j\Delta_y) = f(x, y) \sum_h \sum_k \delta(x - h\Delta_x, y - k\Delta_y)$

❖ Nel dominio delle frequenze:

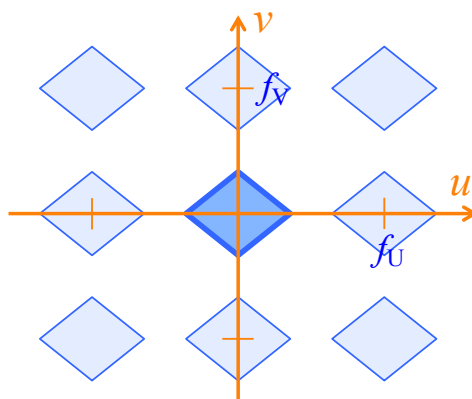
ripetizione periodica (periodo: $1/\Delta$) dello spettro originale:

$$F_s(u, v) = F(u, v) * \sum_h \sum_k \delta\left(u - \frac{h}{\Delta_x}, v - \frac{k}{\Delta_y}\right) = \sum_h \sum_k F(u - h f_x, v - k f_y)$$

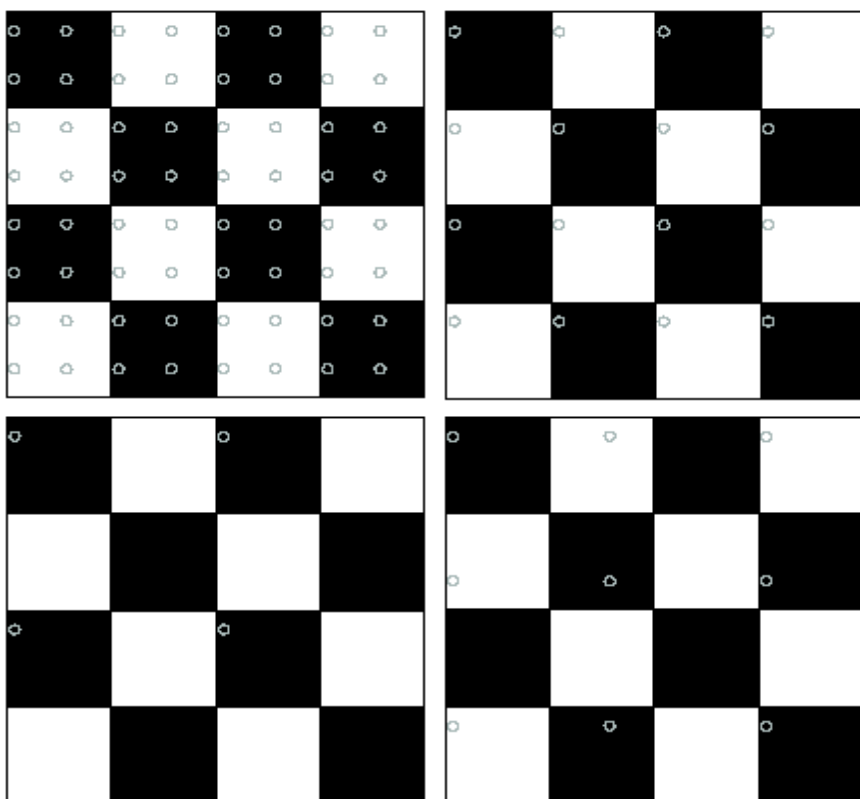
❖ dove: $f_x = \frac{1}{\Delta_x}, f_y = \frac{1}{\Delta_y}$

Teorema del campionamento (Nyquist):

se lo spettro è limitato in banda a $(f_u/2, f_v/2)$, è possibile ricostruire la funzione (immagine) originale a partire dai suoi campioni.



Aliasing - esempio



Resample the checkerboard by taking one sample at each circle.

Top left and top right: reasonable results.

Bottom left is all black and bottom right has checks that are too big.

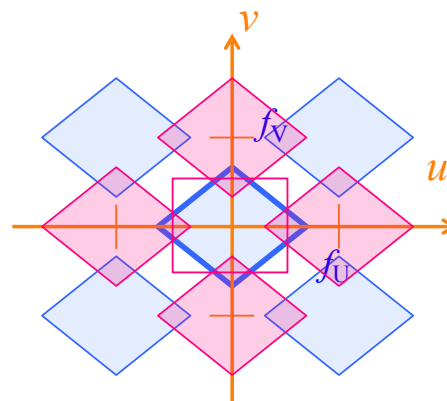
Aliasing

- ❖ Se il teorema del campionamento non è rispettato, non posso ricostruire l'immagine originale:

Filtraggio anti-Aliasing

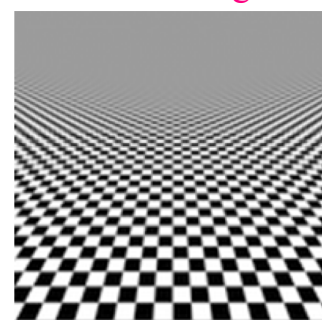
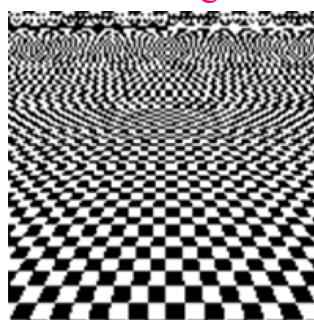
- ❖ Applico un filtro passa-basso, che azzeri tutte le frequenze al di fuori del rettangolo:

$$H(u, v) = 0, \quad |u| \geq \frac{f_U}{2} \text{ oppure } |v| \geq \frac{f_V}{2}$$



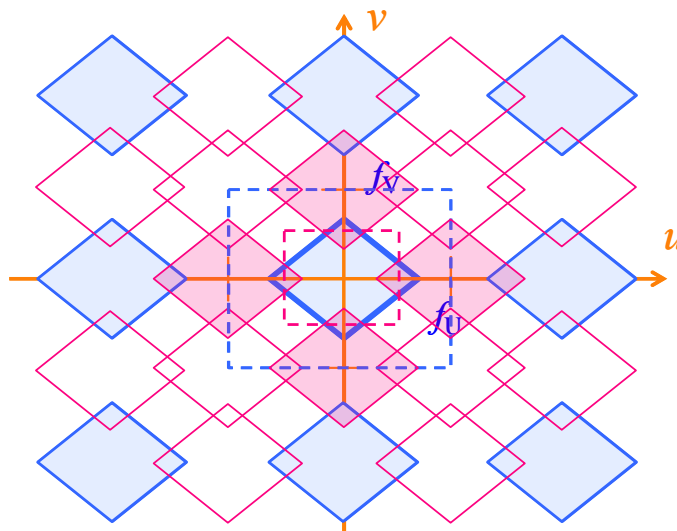
aliasing

no aliasing



Sottocampionamento e aliasing

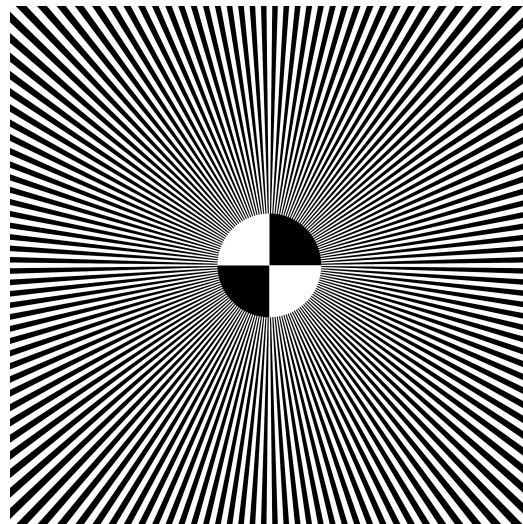
- ❖ Sottocampionamento: **1 campione ogni N**
- ❖ Nelle frequenze: frequenze spaziali **si riducono di 1/N**
- ❖ le repliche spettrali **si avvicinano di N volte** → si può generare **ALIASING!**



Sottocampionamento e aliasing

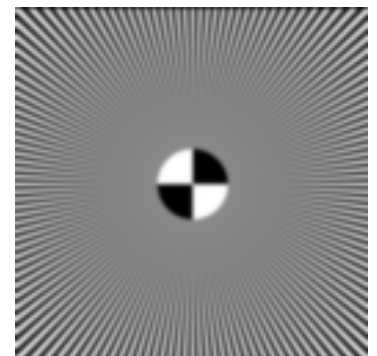
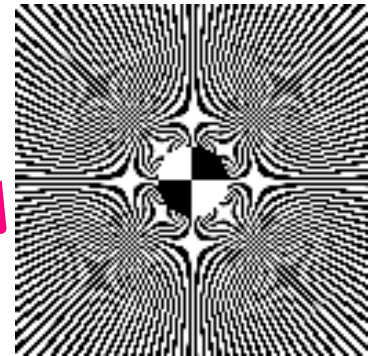
- ❖ Soluzione: **filtraggio passa-basso** → **subsampling**
- ❖ in genere si utilizza un **kernel Gaussiano**
 - *separabile (calcolo efficiente)*
 - *qualità visiva del filtraggio (assenza di 'ringing')*

MATLAB LIVE Script:
Subsampling



2048x2048

1/16 subsampling



128x128

Motivazioni: ridimensionamento

- ❖ L'immagine è troppo grande. Come possiamo ridurla (ad es. di un fattore 2)?

Proposta:

- ❖ *prendiamo un pixel ogni 2*

Va bene?

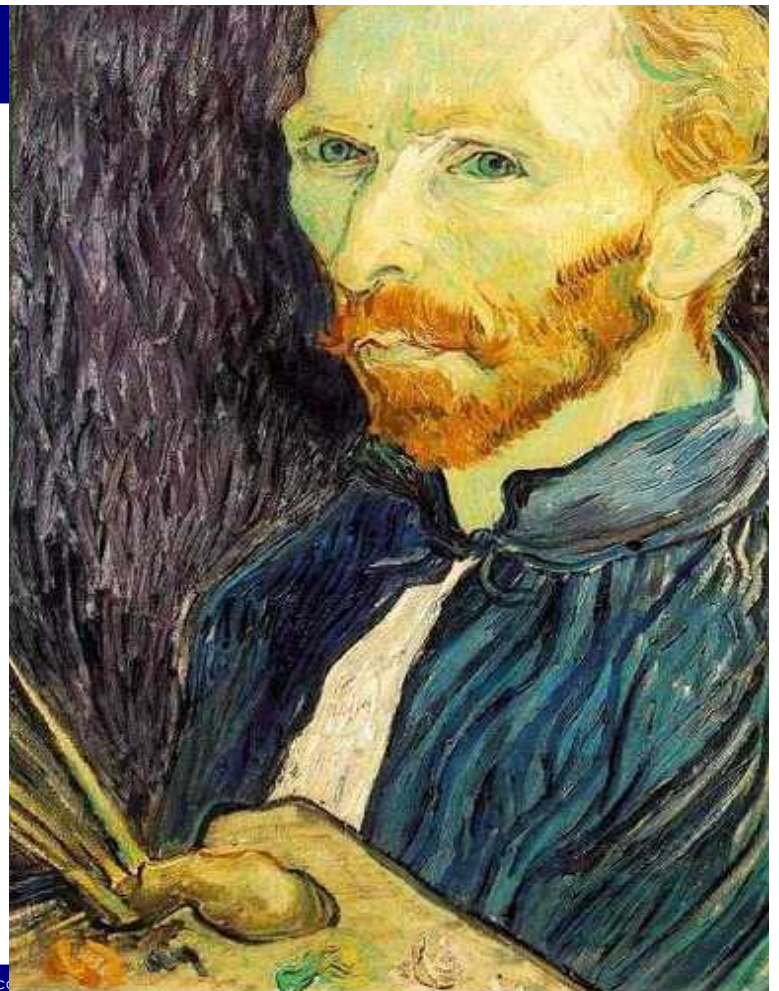
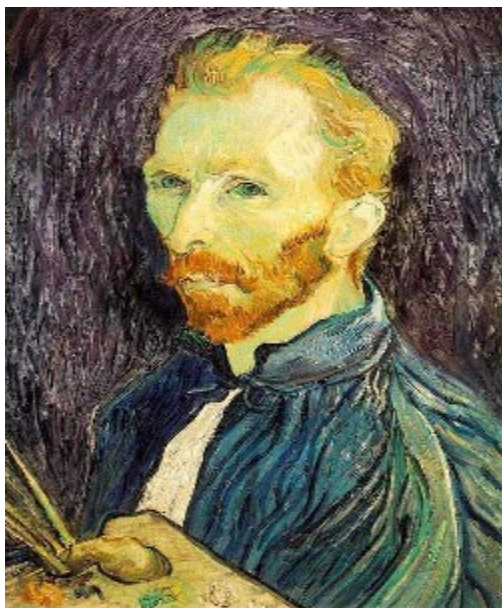




Image subsampling (50%):

Eliminazione di una riga e una colonna ogni 2

→ 1/2 righe, 1/2 colonne



1/2



1/4



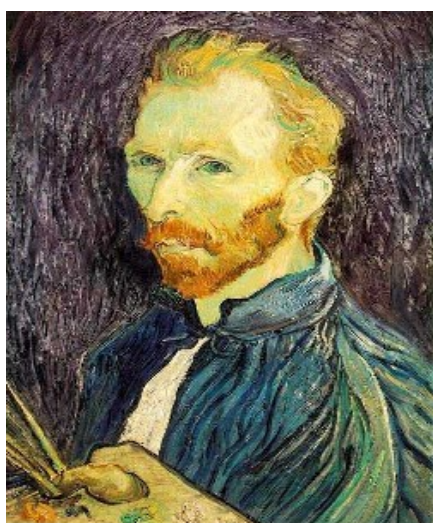
1/8

source: S. Seitz

Sottocampionamento



❖ Image subsampling (senza prefiltraggio)



1/2



1/4
(2x zoom)



1/8
(4x zoom)

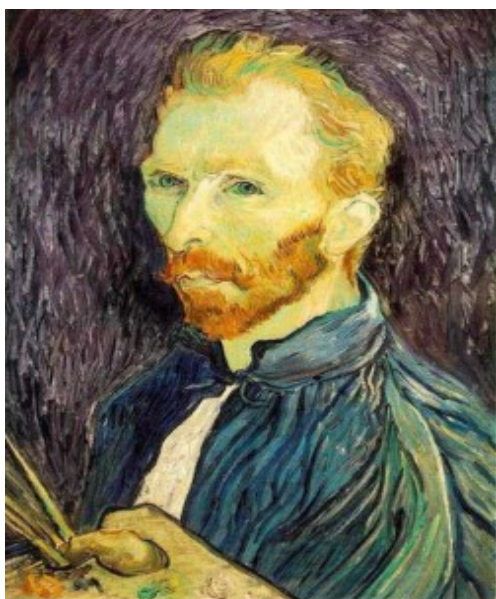
source: S. Seitz



Gaussian (low-pass) pre-filtering

Solution: **filter** the image, *then* **subsample**

- Filter size should double for each $\frac{1}{2}$ size reduction



Gaussian $\rightarrow 1/2$



$G \rightarrow 1/4$



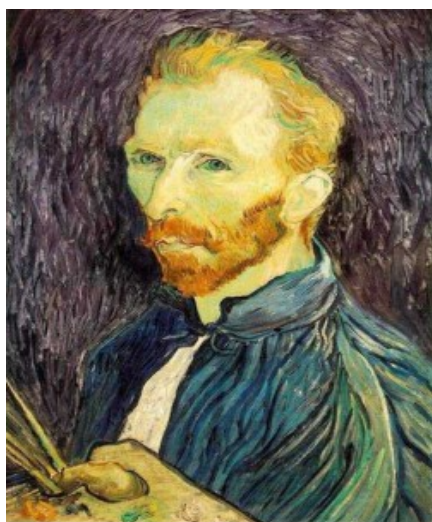
$G \rightarrow 1/8$

source: S. Seitz

Sottocampionamento



- ❖ con prefiltraggio mediante kernel gaussiano



$1/2$



$1/4$
(2x zoom)



$1/8$
(4x zoom)

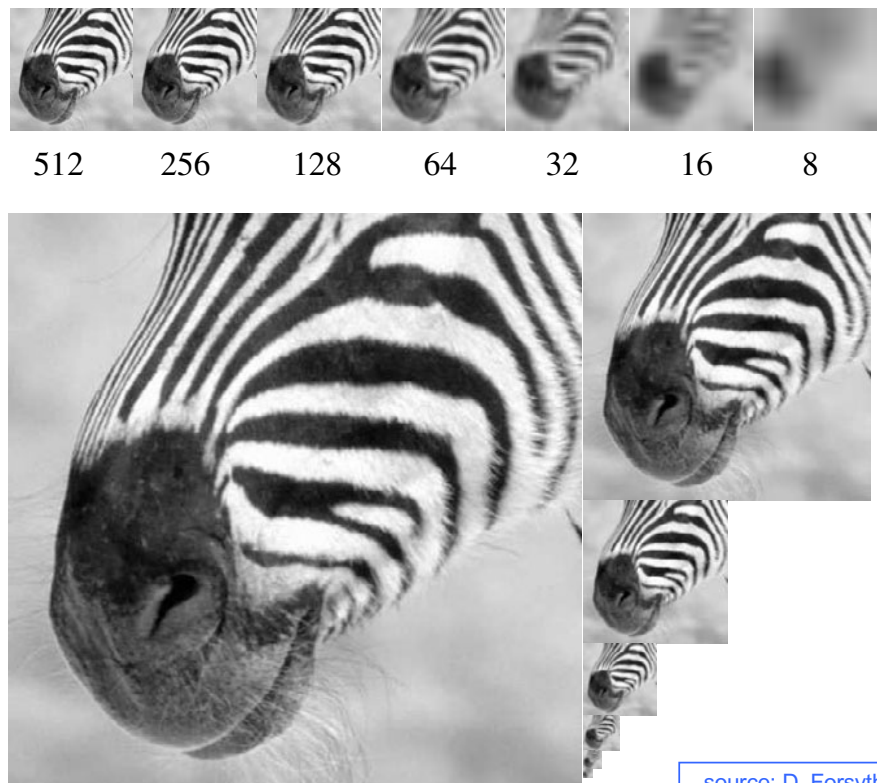
source: S. Seitz

Piramide Gaussiana (Gaussian pyramid)

- ❖ A partire dall'immagine originale, generazione di layer multipli, ogni volta di dimensioni dimezzate

Subsampling:

- ❖ ogni volta applico un filtro gaussiano con funzione **anti-aliasing**



source: D. Forsyth

Applicazione: Pattern Detection mediante correlazione

Pattern Detection mediante correlazione

- ❖ La convoluzione con $h(x, y)$ è il modo **ottimo** per rivelare il pattern $h(x, y)$ in immagini rumorose (rumore bianco)
- ❖ Tecnica ottima per rivelare un pattern di forma nota immerso in rumore gaussiano bianco

correlazione con $h(x, y)$ = convoluzione con $\bar{h}(x, y)$

$$f * g = \sum_{h,k} f(h,k) g(i-h, j-k) \quad \Leftrightarrow \quad \text{xcorr}(f, g) = \sum_{h,k} f(h,k) g(i+h, j+k)$$

- ❖ Tecnica utilizzata (in 1D) nei radar per rivelare l'eco del segnale trasmesso: la posizione dell'eco mi dice la distanza dell'oggetto rilevato

