Corso di

Visione Artificiale

Laurea Magistrale in Informatica (F94)

Docenti:

Raffaella Lanzarotti

Federico Pedersini

Dipartimento di Informatica

Università degli Studi di Milano

Estrazione e localizzazione di "features" da immagini

(features extraction)

- Estrazione di contorni
  - Gradiente dell'immagine
  - Estrazione di contorni (edge detection)
- Localizzazione di punti significativi
  - Operatore di Harris
  - Estrazione di features invarianti – SIFT

(Forsyth/Ponce: Capitolo 5)

Slide credits: varie sorgenti (citate)

Edge detection

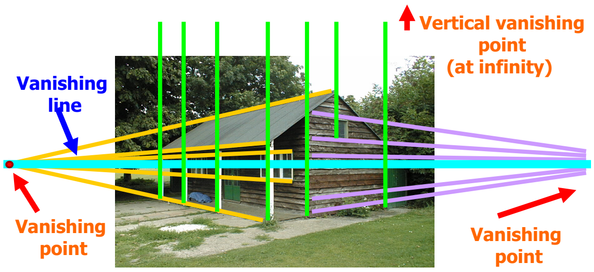
Perché i contorni (edges) sono importanti?

- Informazioni salienti in una scena, fondamentali per riconoscere oggetti
  - *Early vision* [Poggio, Marr]: il cervello 'estrae' i contorni principali dalla scena osservata

- Localizzazione precisa → portatori di informazioni geometriche preziose
  - Contorni → ingombri
  - Linee di fuga → prospettiva

Applicazioni:

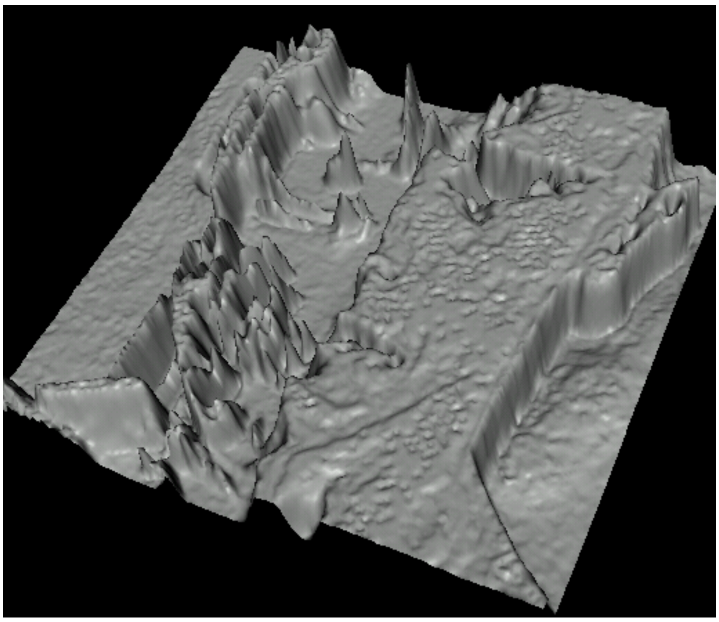
- Object recognition
- Camera calibration
- 3D reconstruction



source: J. Hayes

Image gradient

Contorno → brusco salto di luminanza

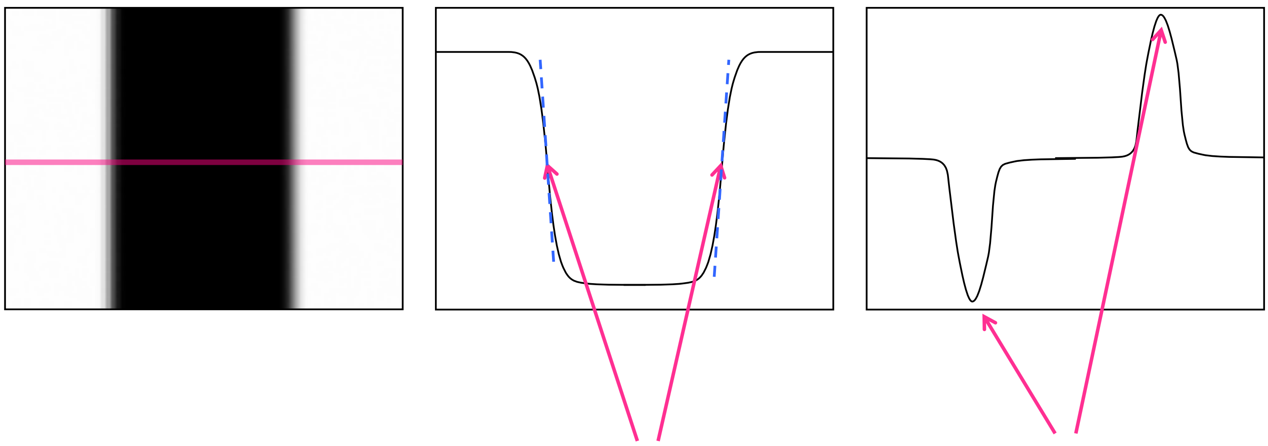


Slide credits: N. Snavely

Image gradient

Contorno → brusco salto di luminanza

immagine f(x,y) | **intensity function** (along horizontal scanline) | → **derivata prima**



edges: ***massima pedenza***

edges: ***minimi/massimi della derivata prima***

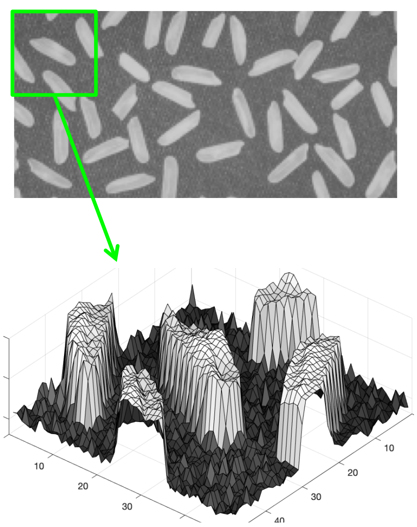
Source: L. Lazebnik
Slide credits: N. Snavely

Image gradient

Derivata prima in 2D → gradiente dell'immagine

$$\nabla I(x,y) = \overrightarrow{G(x,y)} = \left\{\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right\}$$

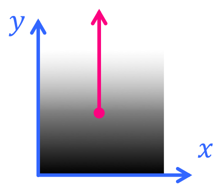
- Gradiente = "vettore-pendenza"
  - direzione: massima pendenza (a salire) della funzione $I(x,y)$
  - modulo: proporzionale alla pendenza



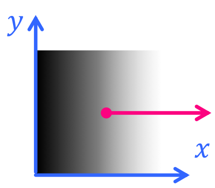
$$\nabla f(x,y) = \begin{bmatrix} \dfrac{\partial f(x,y)}{\partial x} \\ \dfrac{\partial f(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

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

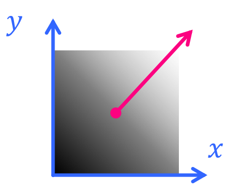
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$



$\Delta I = \{0, f_y\}$



$\Delta I = \{f_x, 0\}$



$\Delta I = \{f_x, f_y\}$

Come calcolare il gradiente di un'immagine?

Immagine **campionata** $I(i,j)$: dominio spaziale (i,j) discreto
→ derivata = differenze finite

Derivata **orizzontale**: $$f_X(i,j) = \frac{\partial f(i,j)}{\partial i} \approx \frac{f(i+1,j) - f(i-1,j)}{2} = \boldsymbol{f} * \boldsymbol{d_X(i,j)}$$

$d_X(i,j)$: $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$

Derivata **verticale**: $$f_Y(i,j) = \frac{\partial f(i,j)}{\partial j} \approx \frac{f(i,j+1) - f(i,j-1)}{2} = \boldsymbol{f} * \boldsymbol{d_Y(i,j)}$$

$d_Y(i,j)$: $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$

Operatori tipici:

Prewitt

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

$P_Y(i,j)$

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

$P_X(i,j)$

Sobel

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

$S_Y(i,j)$

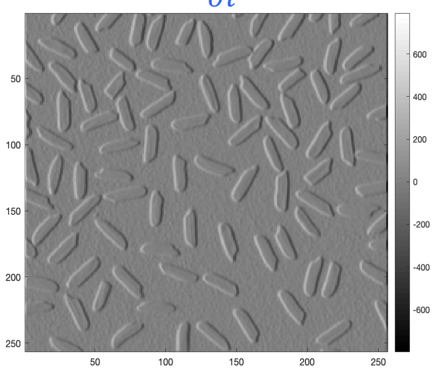
-1	0	1
-2	0	2
-1	0	1

$S_X(i,j)$

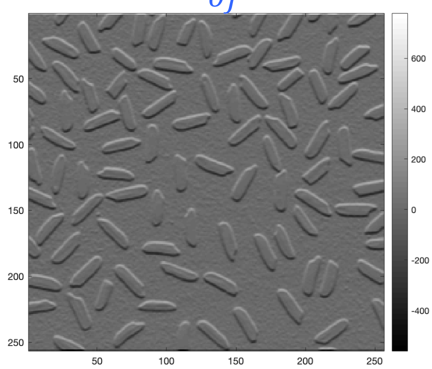
Sobel: esempio

$f(i,j)$

$$f_x(i,j) = \frac{\partial f(i,j)}{\partial i} = f * S_X$$

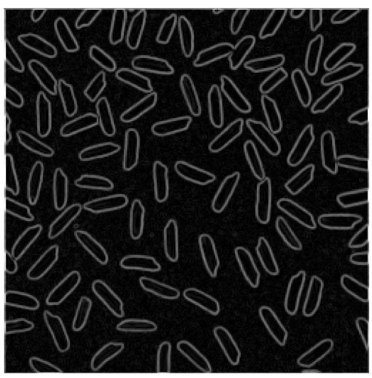


$$f_y(i,j) = \frac{\partial f(i,j)}{\partial j} = f * S_Y$$



Modulo del gradiente:

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

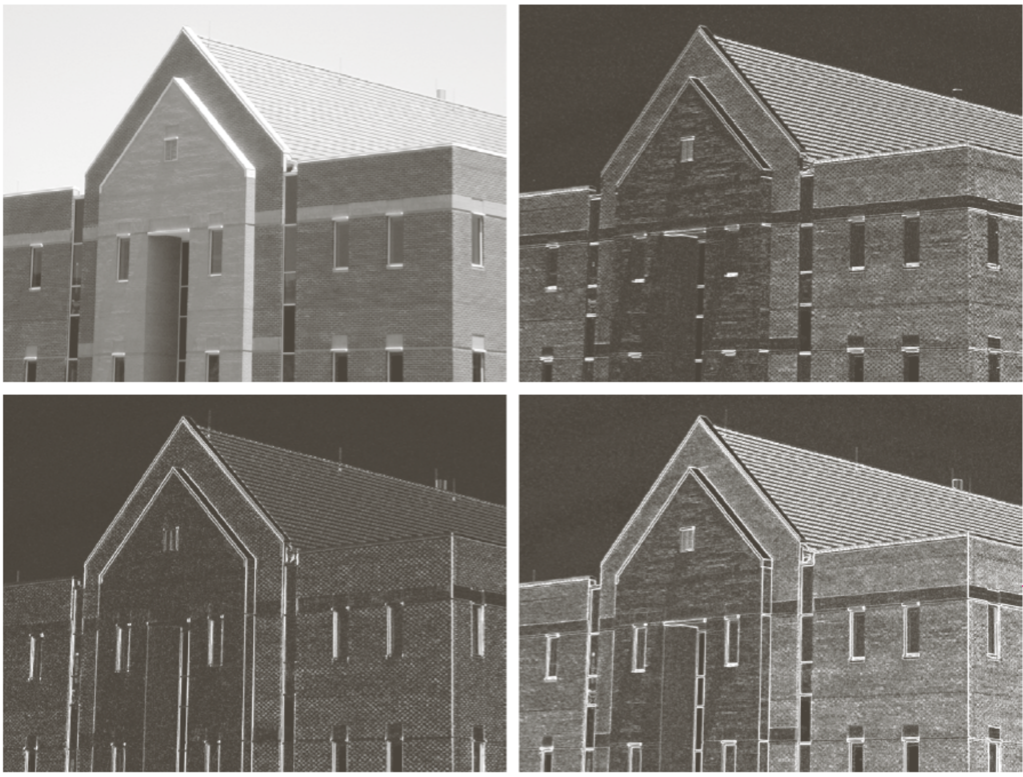


Calcolo del gradiente

Operatore di Sobel:

f $\quad f * S_Y$

$f * S_X$ $\quad |\nabla f(x, y)|$



presenza significativa di rumore

Calcolo del gradiente

Operatore di Sobel:

Selezione edges significativi

Approccio possibile: soglia minima sul valore del modulo

- es: mantengo il 33% degli edges con il modulo più alto

➔ perdo anche edges significativi

Calcolo del gradiente

Problema: **il gradiente è sensibile al rumore**

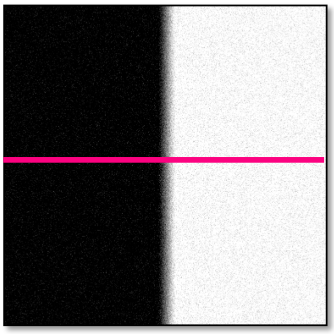
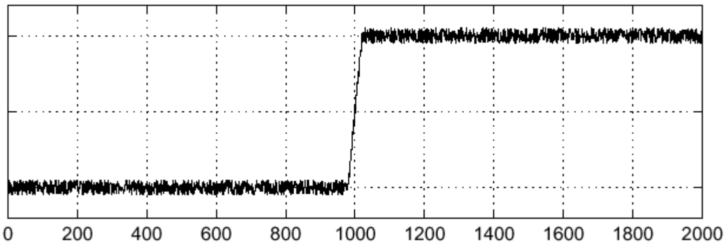
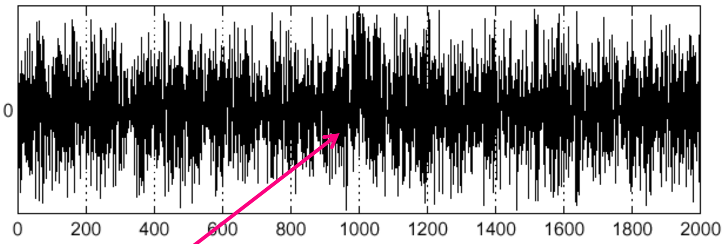


Immagine rumorosa

$f(x)$



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



Dov'è l'*edge*?

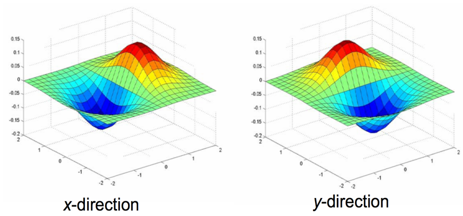
Source: S. Seitz

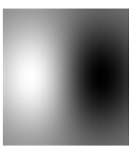
Edge detection

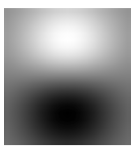
Edge detection con **filtraggio gaussiano**

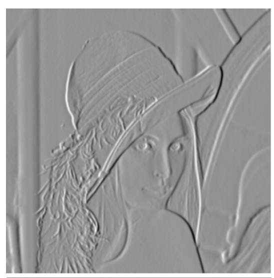
1. *Filtraggio gaussiano* (σ) → $f(i,j) = I(i,j) * G(i,j)$
2. *Calcolo derivate orizzontale / verticale* → $f_x(i,j) = \frac{\partial f(i,j)}{\partial i}$; $f_y(i,j) = \frac{\partial f(i,j)}{\partial j}$
3. *Calcolo modulo del vettore gradiente* → $|\nabla f(i,j)| = \sqrt{f_x^2(i,j) + f_y^2(i,j)}$

DoG
(derivative of gaussian)

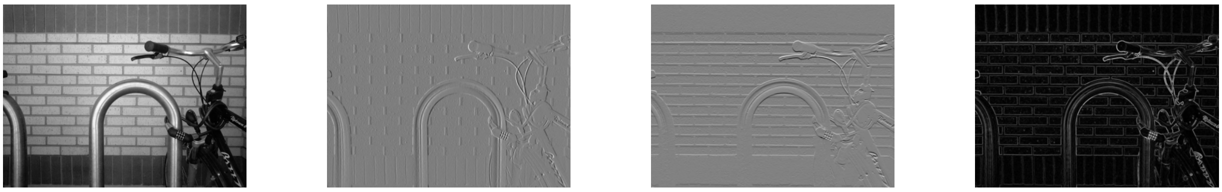








Calcolo del gradiente con filtraggio gaussiano: esempio



- Immagine: $I(x,y)$
- derivata orizzontale:
  $I_x(x,y) = I(x,y) * S_x$
- derivata verticale:
  $I_y(x,y) = I(x,y) * S_y$
- Gradient magnitude:
  $|\nabla I(x,y)| = \sqrt{I_x^2 + I_y^2}$

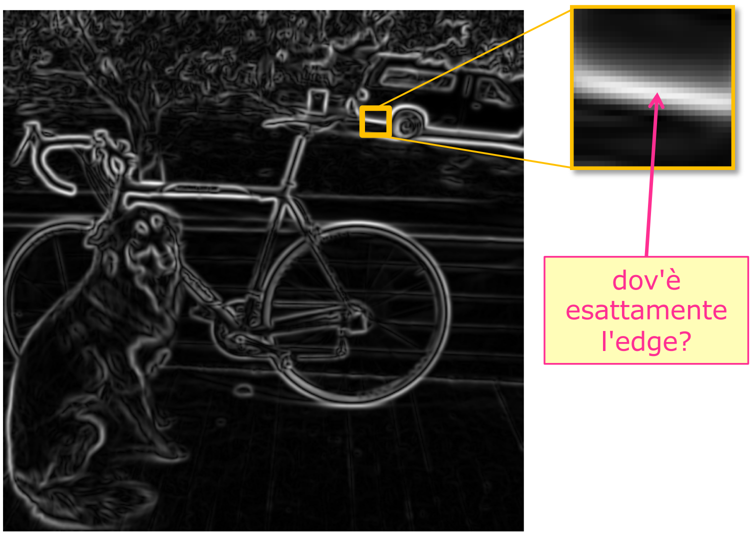
Source: Wikipedia – slide credits: N. Snavely

Localizzazione dei contorni

Localizzazione del contorno

La posizione esatta del contorno: la cresta del modulo del gradiente

original image



modulo del gradiente

Image credit: Joseph Redmon

Localizzazione dei contorni

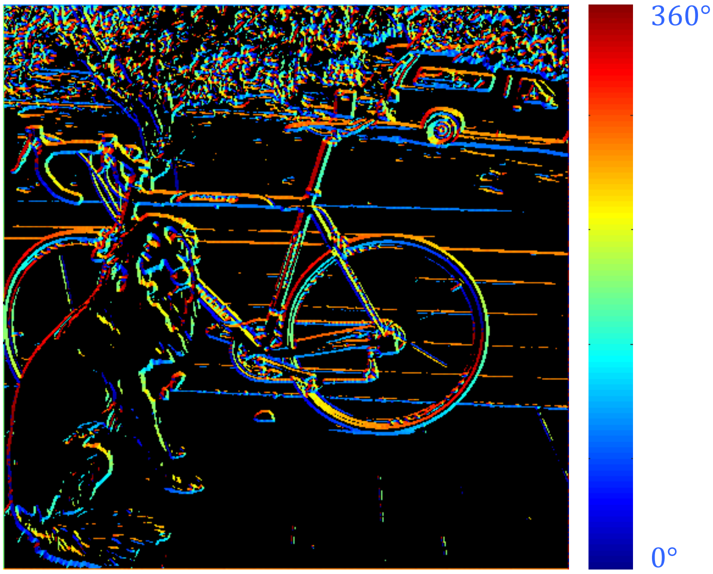
Edge: la **cresta** del modulo del gradiente:

- punto di massimo locale del modulo del gradiente...
  ...lungo la direzione locale del vettore gradiente

$$|\nabla I(x,y)| = \sqrt{f_X^2 + f_Y^2}$$

$$\vartheta = \angle \nabla I(x,y) = \mathrm{atan2}(f_X, f_Y)$$

modulo



direzione

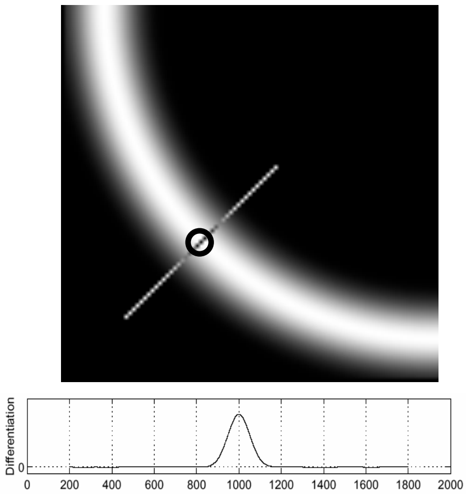
Non-maxima suppression

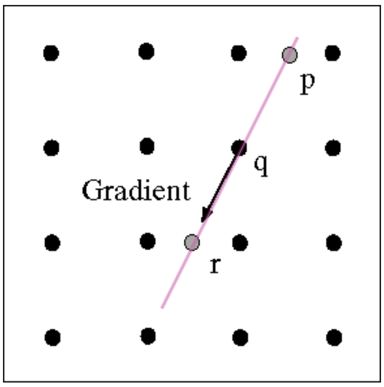
Procedura di determinazione del punto di massimo (sulla cresta del gradiente):

Non-maxima suppression:

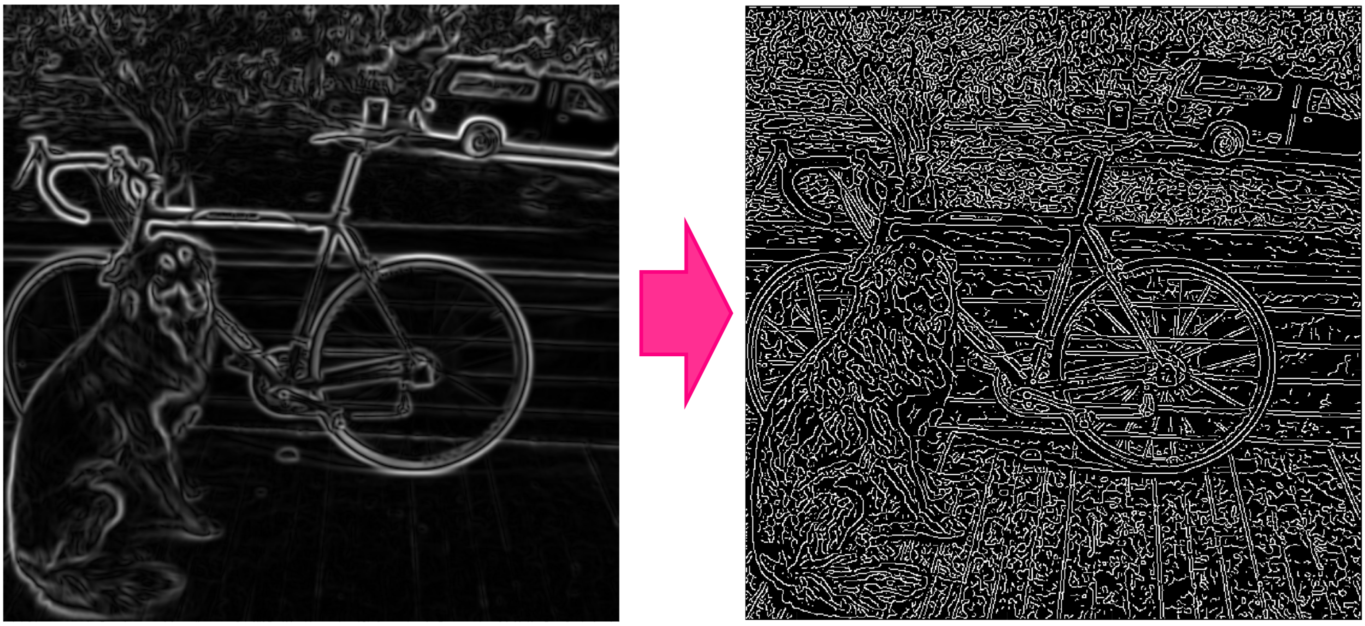
- *Check if pixel **q** is local maximum along gradient direction*
  - *requires interpolating pixels **p** and **r***

$$if \quad |\nabla f(\mathbf{q})| > |\nabla f(\mathbf{p})| \quad and \quad |\nabla f(\mathbf{q})| > |\nabla f(\mathbf{r})| \quad then \quad q \in \{Edges\}$$





Non-maxima suppression



modulo del gradiente

punti di massimo

Sogliatura dei contorni

Sensibilità ai massimi dovuti a rumore

➔ **Sogliatura**

seleziono soltanto i punti di massimo del modulo, se superiore a una soglia ***T***

$$(x, y) \in \{\text{Edges}\} \quad \leftrightarrow \quad |\nabla I(x, y)|_{MAX} > T$$

Problema:

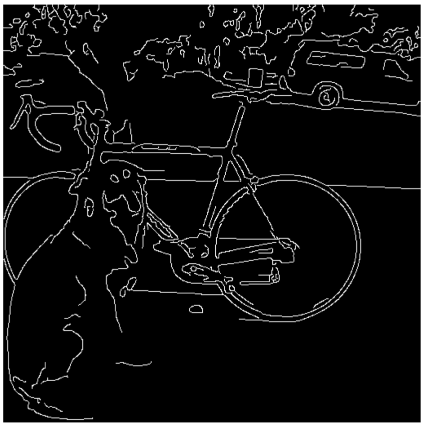
- soglia bassa → presenza edges non significativi (rumore)
- soglia alta → assenza edges significativi

Sogliatura dinamica – isteresi

Soluzione: **Isteresi** → utilizzo due soglie: T_{low} , T_{high}

- *Seleziono un punto di edge se il modulo è:*
  - superiore alla soglia T_{high} oppure
  - superiore alla soglia T_{low} ma vicino a un edge

$$(x,y) \in \{E\} \;\; sse: \;\; |\nabla I(x,y)|_{MAX} > T_{high} \;\; oppure \;\; \begin{cases} |\nabla I(x,y)|_{MAX} > T_{low} \\ \exists p \in N(x,y): \; p \in \{E\} \end{cases}$$



Canny edge detector

Edge detection: metodo di Canny

J. Canny (1986), ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

1. *Filtraggio gaussiano* (σ) → *calcolo derivate h/v* → f_X, f_Y
2. *Calcolo gradiente (modulo, direzione)*
3. *Non-maxima suppression*
4. *Sogliatura con isteresi (2 soglie)*
   - *Define two thresholds: low and high*
   - *Use the high threshold to start edge curves and the low threshold to continue them*

MATLAB: `edge_image = edge( image, 'canny' );`

Proprietà:

- Tuttora lo standard per edge detection
- Prestazioni dipendenti dalla scelta dei parametri:
  - σ del filtraggio gaussiano (scala)
  - soglie alta e bassa: T_{low} ; T_{high}

Demo on web: http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/

Canny edge detector

Edge detection: metodo di Canny

Scelta della scala (σ)

La scelta ottima della scala σ dipende dal comportamento desiderato:

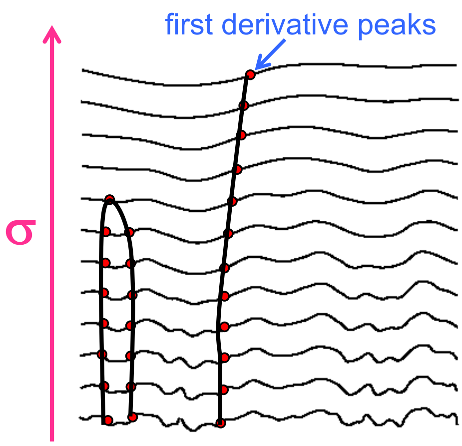
- σ grande: edge di oggetti a "larga scala"
- σ piccolo: edge di oggetti a "piccola scala"

original — Canny with $\sigma = 1$ — Canny with $\sigma = 2$

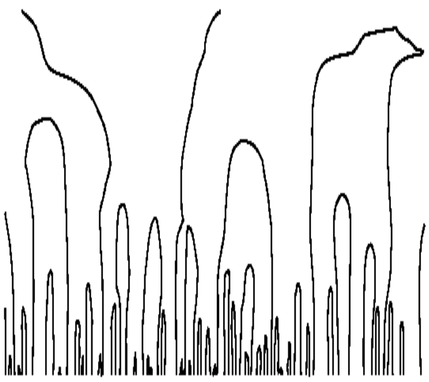
Source: S. Seitz

Scale space

- Properties of scale space
  - edge position may shift with increasing scale (σ)
  - two edges may merge with increasing scale
  - an edge may ***not*** split into two with increasing scale
- *Multi-scale edge detection (edge focusing)* [Bergholm, 1987]
  1. determino gli edges significativi a valori alti di scala (σ alta)
  2. riduco progressivamnte la scala e inseguo la posizione accurata degli edges



Gaussian-filtered signal



Estrazione e localizzazione di "features" da immagini

(features extraction)

- Estrazione di contorni
  - Gradiente dell'immagine
  - Estrazione di contorni (edge detection)

- Localizzazione di punti significativi
  - Operatore di Harris
  - Estrazione di features invarianti – SIFT

(Forsyth/Ponce: Capitolo 5)

Slide credits: varie sorgenti (citate)

Localizzabilità di una feature

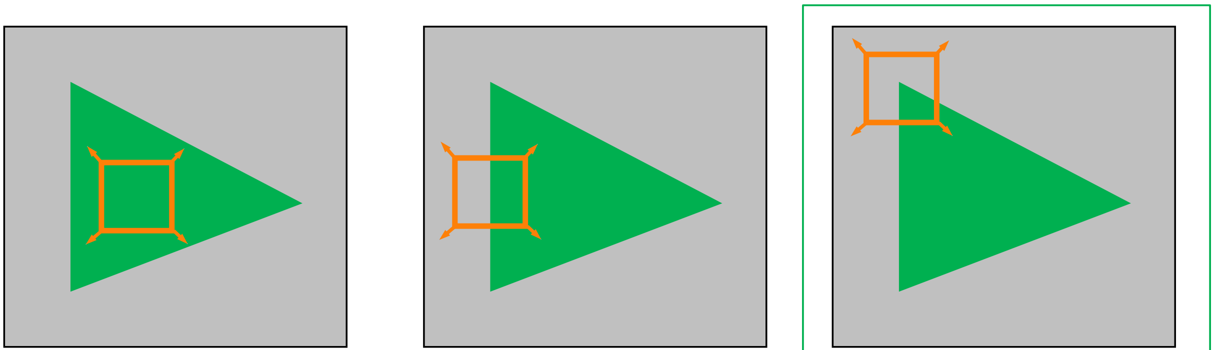
Feature:
particolare di un'immagine localizzabile senza ambiguità e con precisione

Localizzabilità di una *'feature'*:

- *Come cambia l'immagine nella finestra (feature), spostandola?*

Principio di **localizzazione**: ogni spostamento, in qualunque direzione, per poter essere rilevato, deve causare un cambiamento dell'immagine nella finestra

casi possibili:



"flat" region: no change in all directions

"edge": no change along the edge direction

"corner": significant change in all directions

Credit: S. Seitz, D. Frolova, D. Simakov

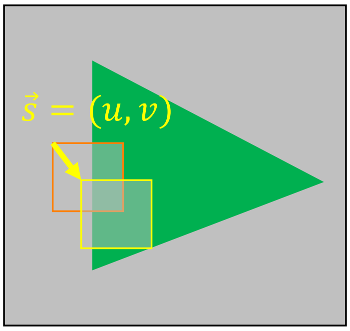
Credit: S. Seitz, D. Frolova, D. Simakov

Determinabilità della localizzazione di una feature

Misura dell'**errore di localizzazione**

Consideriamo uno spostamento $\vec{s} = (u, v)$ della finestra ***W***

- *Quanto cambia l'immagine in **W**?*



Misura del cambiamento:

- Confronto, pixel a pixel, i pixel della finestra ***W*** prima e dopo lo spostamento $\vec{s}$
- Sommo i quadrati delle differenze tra pixel (*Sum of Squared Differences – **SSD***)

Definisco **l'errore SSD:** $E(u, v)$

- Dato lo spostamento $\vec{s} = (u, v)$:

$$E(u,v) = \sum_{(x,y)\in W} \left[I(x+u, y+v) - I(x,y)\right]^2$$

Slide credits: Noah Snavely

Small motion assumption

SSD Error: $$E(u,v) = \sum_{(x,y)\in W} \left[I(x+u, y+v) - I(x,y)\right]^2$$

Approssimo $I(x+u\,,\ y+v)$ mediante lo sviluppo in serie di Taylor:

$$I(x+u\,, y+v) = I(x,y)\ +\ \frac{\partial I}{\partial x}u\ +\ \frac{\partial I}{\partial y}v\ +\ \{termini\ di\ ordine\ sup.\}$$

"If the motion (u, v) is small (*small motion assumption*), then the *first-order approximation* is good enough"

$$I(x+u\,, y+v)\ \approx\ I(x,y)\ +\ \frac{\partial I}{\partial x}u\ +\ \frac{\partial I}{\partial y}v\ =\ I(x,y)\ + [I_x \quad I_y]\begin{bmatrix} u \\ v \end{bmatrix}$$

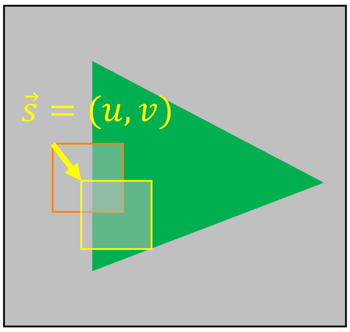
$$dove:\ I_x = \frac{\partial I}{\partial x}\ ;\ I_y = \frac{\partial I}{\partial y}$$

Sostituendo nella formula di $E(x, y)$ ottengo...

Slide credits: Noah Snavely

Corner detection

misura dell'**errore di localizzazione** (cont.)



Consideriamo uno spostamento $\vec{s} = (u, v)$ della finestra $\boldsymbol{W}$

- *Quanto cambia l'immagine in **W**?*

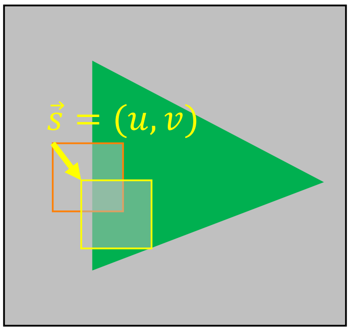
$$I(x+u\,, y+v) \;\approx\; \underbrace{I(x,y) \;+ I_x\, u \;+ I_y\, v}$$

$$\begin{aligned}
E(u,v) &= \sum_{(x,y)\in W} \left[I(x+u, y+v) - I(x,y) \right]^2 \\
&\approx \sum_{(x,y)\in W} \left[I(\cancel{x,y}) + I_x u + I_y v - I(\cancel{x,y}) \right]^2 \\
&\approx \sum_{(x,y)\in W} \left[I_x u + I_y v \right]^2
\end{aligned}$$

Slide credits: Noah Snavely

Corner detection

Misura dell'**errore di localizzazione**



Consideriamo uno spostamento $\vec{s} = (u, v)$ della finestra W

- *Quanto cambia l'immagine in W?*

$$\begin{aligned}
E(u,v) &\approx \sum_{(x,y)\in W} \left[I_x u + I_y v \right]^2 \\
&\approx Au^2 + 2Buv + Cv^2
\end{aligned}$$

$$A = \sum_{(x,y)\in W} I_x^2 \qquad B = \sum_{(x,y)\in W} I_x I_y \qquad C = \sum_{(x,y)\in W} I_y^2$$

➔ $E(u,v)$ approssimato localmente da una forma quadratica

Slide credits: Noah Snavely

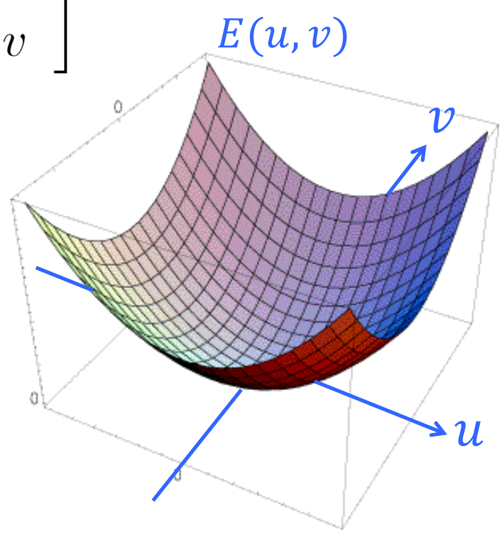
The second moment matrix

$E(u,v)$ approssimato localmente da una forma quadratica

$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$
$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{\boldsymbol{H}} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$
$$B = \sum_{(x,y)\in W} I_x I_y$$
$$C = \sum_{(x,y)\in W} I_y^2$$

***H*: second moment matrix**



Slide credits: Noah Snavely

Let's try to understand its shape

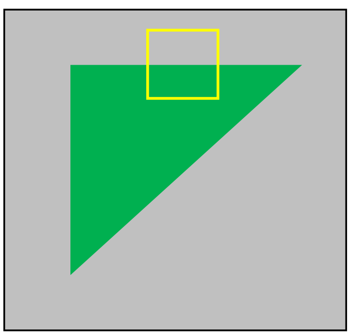
The second moment matrix: horizontal edge

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{\boldsymbol{H}} \begin{bmatrix} u \\ v \end{bmatrix}$$

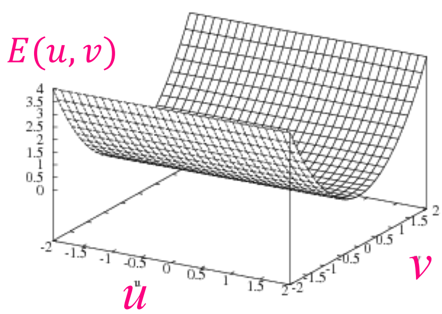
$$A = \sum_{(x,y)\in W} I_x^2$$
$$B = \sum_{(x,y)\in W} I_x I_y$$
$$C = \sum_{(x,y)\in W} I_y^2$$

Esempio:

- **edge orizzontale** ➔ $I_x = 0$



$$H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}$$



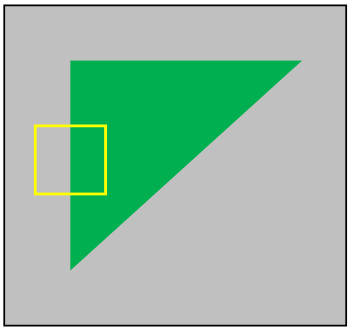
The second moment matrix: vertical edge

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

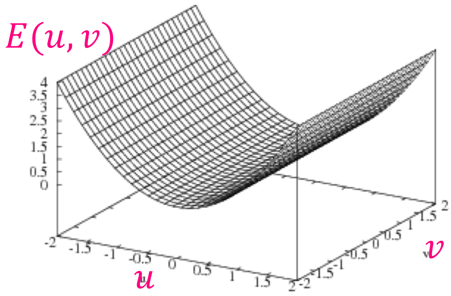
$$A = \sum_{(x,y)\in W} I_x^2$$
$$B = \sum_{(x,y)\in W} I_x I_y$$
$$C = \sum_{(x,y)\in W} I_y^2$$

Esempio:

- **edge verticale** ➔ $I_y = 0$



$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$



The second moment matrix: general case

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$ (H)

$$A = \sum_{(x,y)\in W} I_x^2 \quad B = \sum_{(x,y)\in W} I_x I_y \quad C = \sum_{(x,y)\in W} I_y^2$$

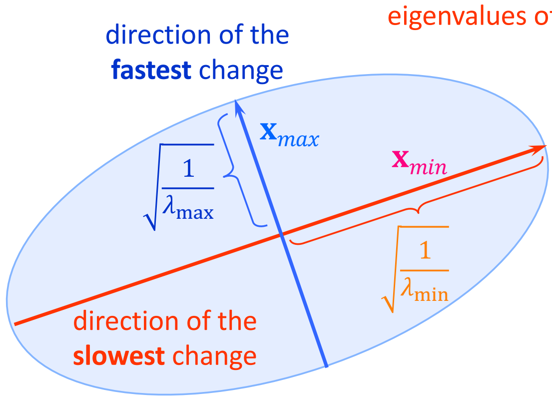
L'equazione: $[u \;\; v] \; H \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$ è **un'ellisse**

Autovalori/autovettori di H

$$\boldsymbol{H} \cdot \mathbf{x} = \lambda\, \mathbf{x} \;\rightarrow\; \begin{matrix} \lambda_1, \lambda_2 \\ \mathbf{x}_1, \mathbf{x}_2 \end{matrix}$$

$$\rightarrow \begin{matrix} \boldsymbol{H} \cdot \mathbf{x}_{min} = \lambda_{min}\, \mathbf{x}_{min} \\ \boldsymbol{H} \cdot \mathbf{x}_{max} = \lambda_{max}\, \mathbf{x}_{max} \end{matrix}$$

$\lambda_{\max}$, $\lambda_{\min}$: eigenvalues of ***H***



*We can visualize **H** as an ellipse with:*

- *axis lengths determined by the eigenvalues of **H** and*
- *orientation determined by the eigenvectors of **H***

Slide credits: Noah Snavely

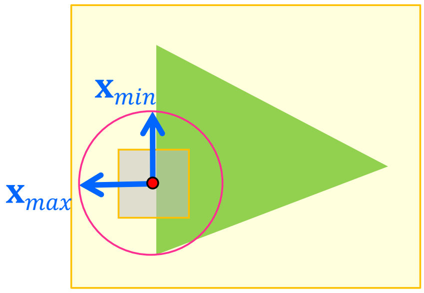
The second moment matrix

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2 \quad B = \sum_{(x,y)\in W} I_x I_y \quad C = \sum_{(x,y)\in W} I_y^2$$

Autovalori/autovettori e localizzabilità:

$$\rightarrow \quad \begin{aligned} \boldsymbol{H} \cdot \mathbf{x}_{min} &= \lambda_{min}\, \mathbf{x}_{min} \\ \boldsymbol{H} \cdot \mathbf{x}_{max} &= \lambda_{max}\, \mathbf{x}_{max} \end{aligned}$$



*Gli autovalori e gli autovettori di **H** definiscono le direzioni di spostamento caratterizzate dalla massima (λ_{max}, $\mathbf{x}_{max}$) e minima (λ_{min}, $\mathbf{x}_{min}$) variazione dell'errore di localizzazione $E(x,y)$*

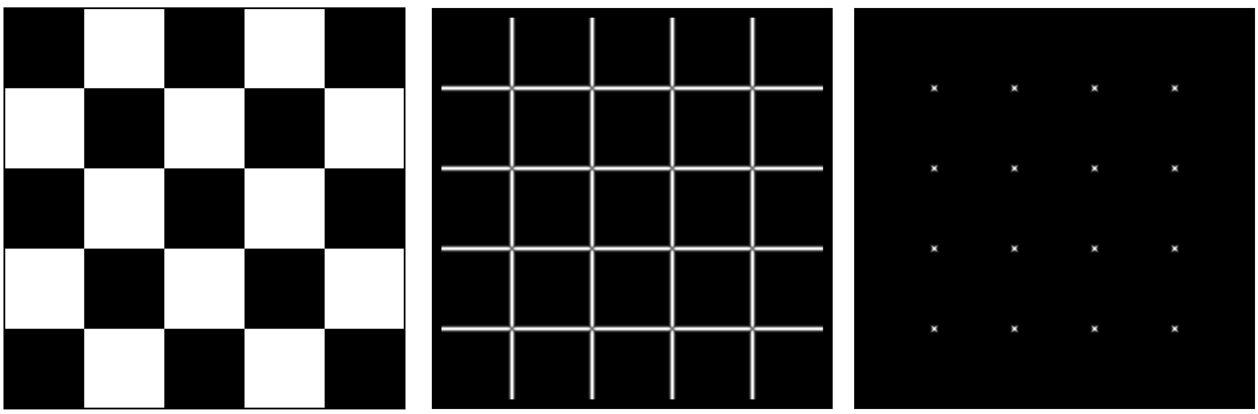
- $\mathbf{x}_{max}$ = direzione di massimo incremento di $E(x,y)$
- λ_{max} = velocità di incremento di $E(x,y)$ in direzione $\mathbf{x}_{max}$
- $\mathbf{x}_{min}$ = direzione di minimo incremento di $E(x,y)$
- λ_{min} = velocità di incremento di $E(x,y)$ in direzione $\mathbf{x}_{min}$

Slide credits: Noah Snavely

Corner detection

- Perché λ_{max}, x_{max}, λ_{min}, and x_{min} sono rilevanti per la feature detection?
  - qual è il nostro obiettivo? **localizzabilità**

→ $E(u,v)$ deve essere grande in tutte le direzioni
  - il minimo di $E(u,v)$ deve essere grande, per tutti i versori [u v]
  - tale minimo coincide con l'autovalore minore di ***H***, λ_{min}

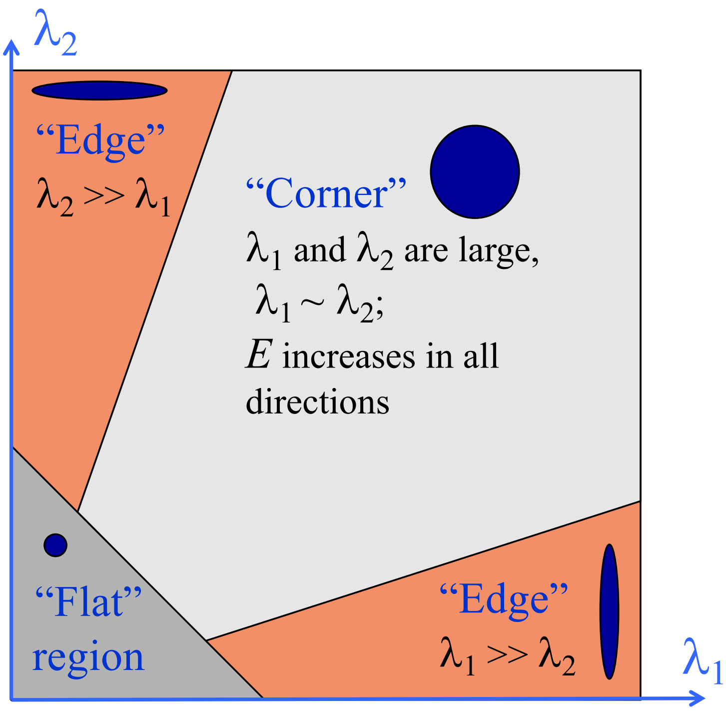


$I(i,j)$ $\lambda_{max}(i,j)$ $\lambda_{min}(i,j)$

Classification of image points using eigenvalues of *H*:

$$\mathbf{H}\cdot\mathbf{x}=\lambda\,\mathbf{x} \;\rightarrow\; \lambda_1,\lambda_2$$

Matlab: `L = eig(H);`

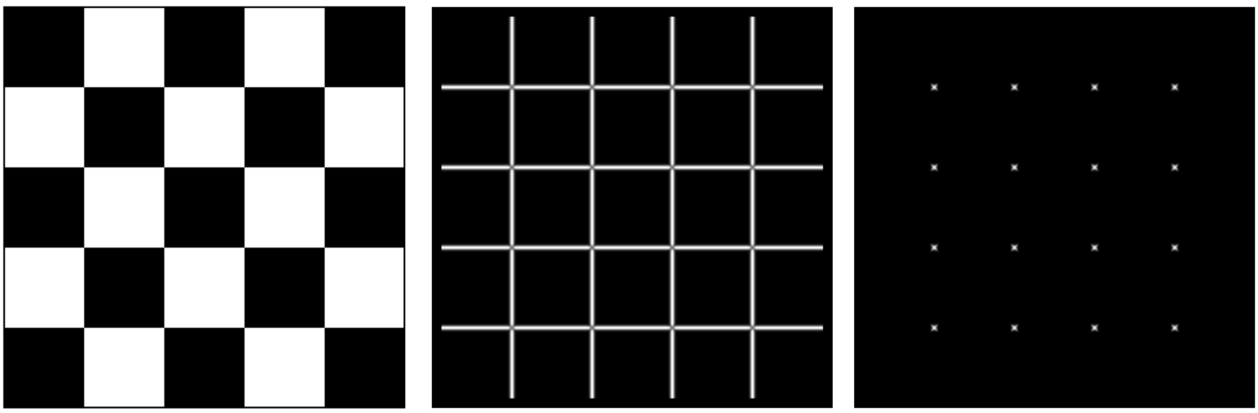


λ_1 and λ_2 are small;
E is almost constant in all directions

Slide credits: Noah Snavely

Corner detection algorithm:

- Compute the gradient at each point in the image
- Create the ***H*** matrix from the gradient (at each desired point)
- Compute the eigenvalues → λ_{min} , λ_{max}
- Find points with large response (λ_{min} > threshold)
- Choose those points where λ_{min} is a local maximum as features

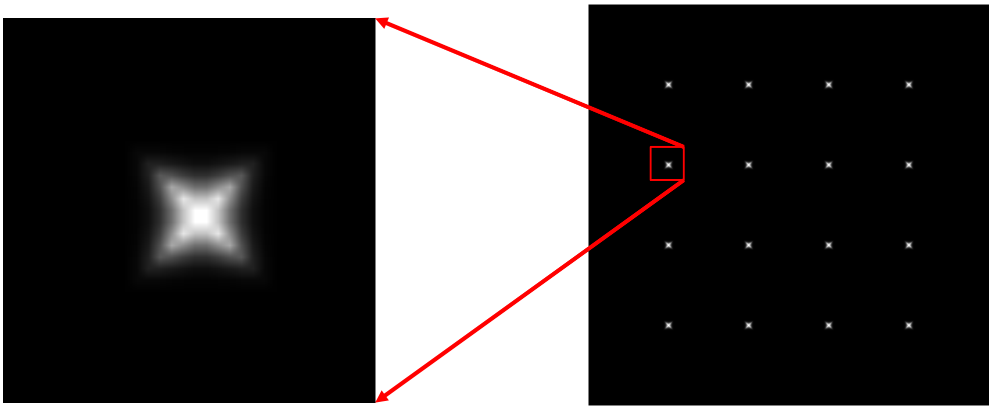


$I(i,j)$ $\lambda_{max}(i,j)$ $\lambda_{min}(i,j)$

Corner detection summary

Corner detection algorithm:

- Compute the gradient at each point in the image
- Create the ***H*** matrix from the gradient (at each desired point)
- Compute the eigenvalues → λ_{min} , λ_{max}
- Find points with large response (λ_{min} > threshold)
- Choose those points where λ_{min} is a local maximum as **features**



Slide credits: Noah Snavely

The Harris operator

"Harris operator" *is a variant of λ_{min} the for feature detection*

Harris corner detector:

- Definisce il parametro *response function **R**:*

$$R = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$
$$\alpha = 0.05$$

si dimostra che:

$$\lambda_1\lambda_2 = \det(\mathbf{H}) = h_{11}h_{22} - h_{21}h_{12}$$

$$\lambda_1 + \lambda_2 = \mathrm{tr}(\mathbf{H}) = h_{11} + h_{22}$$

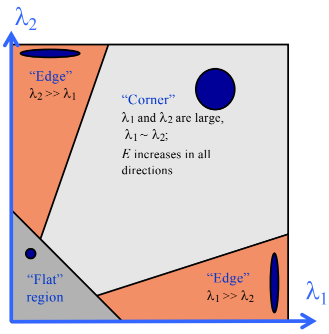
$$H = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

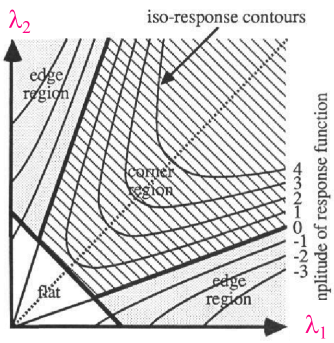
Quindi:

$$\boxed{R = \sum I_x^2 + \sum I_y^2 - \left(\sum I_x I_y\right)^2 - \alpha\left(\sum I_x^2 + \sum I_y^2\right)^2}$$

- Risposta simile a λ_{min}, ma calcolo molto più efficiente

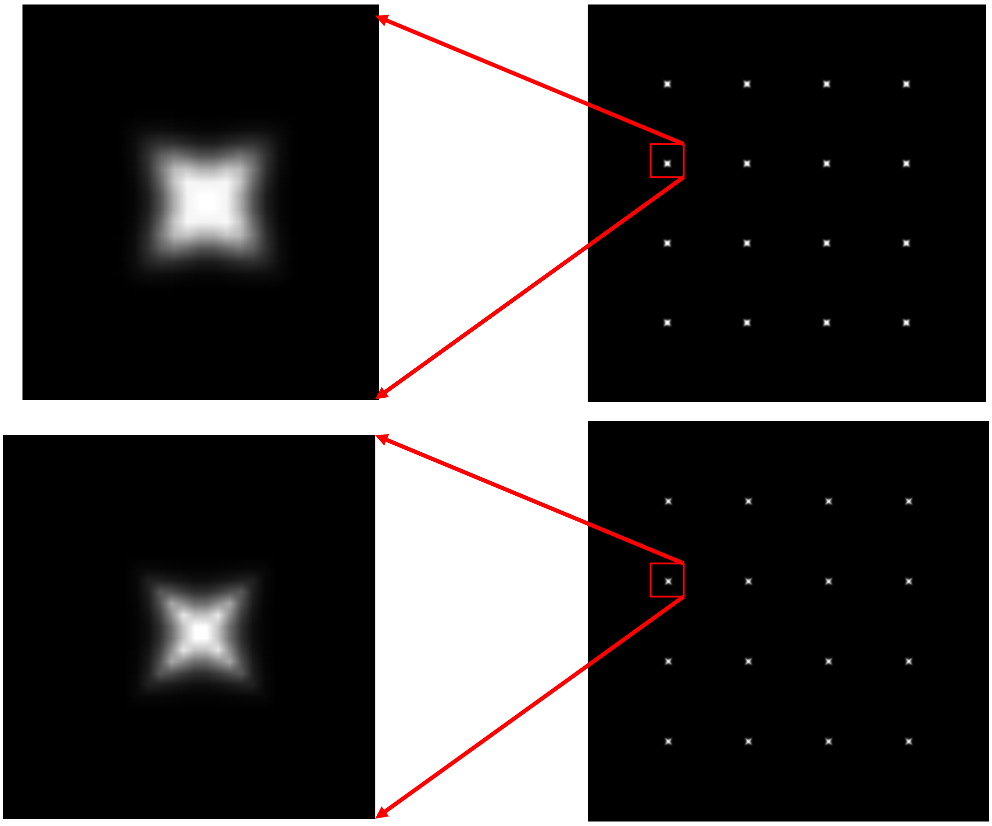
→ il più popolare, tra i corner detectors





C. Harris and M. Stephens (1988), A Combined Corner and Edge Detector, *Alvey Vision Conference,* 15, 1988.

The Harris operator



Harris operator $R(x,y)$

$\lambda_{min}(x,y)$

Pesatura della finestra

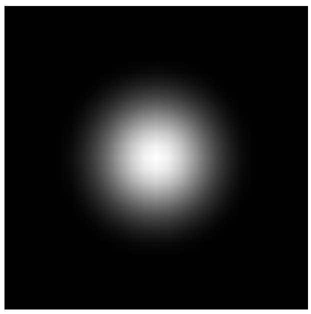
Pesatura della finestra ***W***:

- In practice, using a simple window ***W*** doesn't work too well

$$H = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

- Instead, we'll *weigh* each derivative value based on its distance from the center pixel

$$H = \sum_{(x,y)\in W} w_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

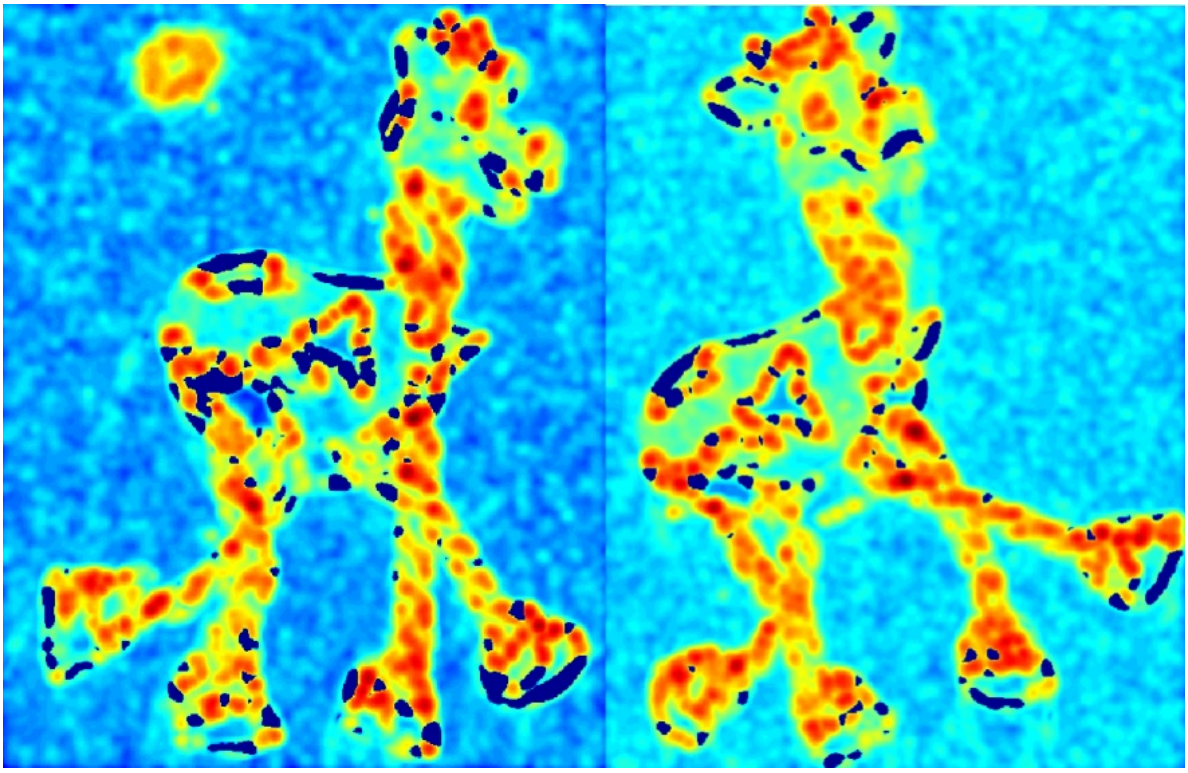


$w_{x,y}$

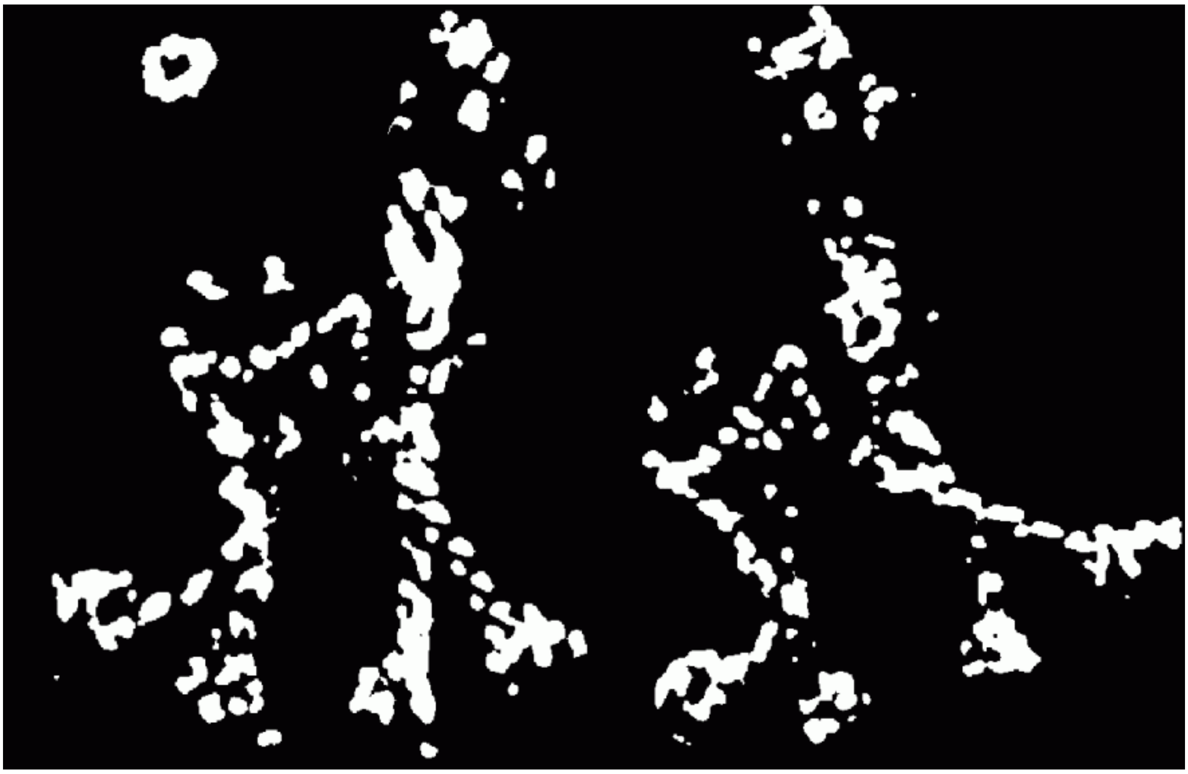
Slide credits: Noah Snavely

Harris detector example

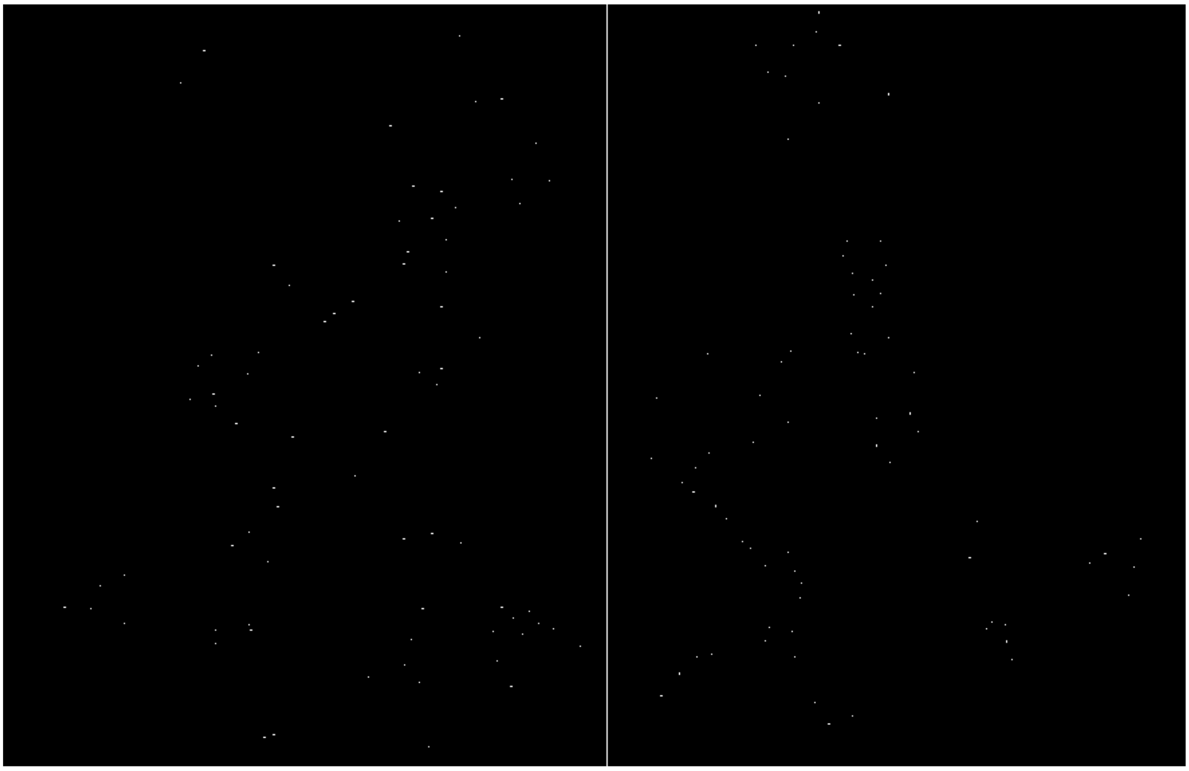
Valori di $R(x, y)$ (red=high, blue=low)



Threshold ($R(x,y)$ > value)



Massimi locali di $R(x,y)$

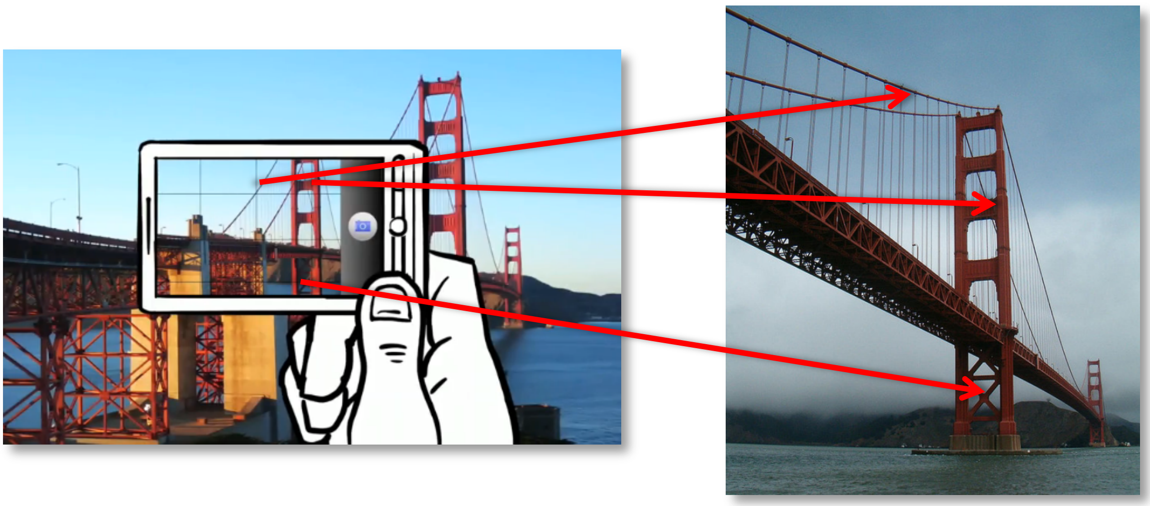


Harris features

Feature matching

Feature matching:

- Localizzazione di particolari in un'immagine
- Riconoscimento dello stesso particolare in immagini diverse

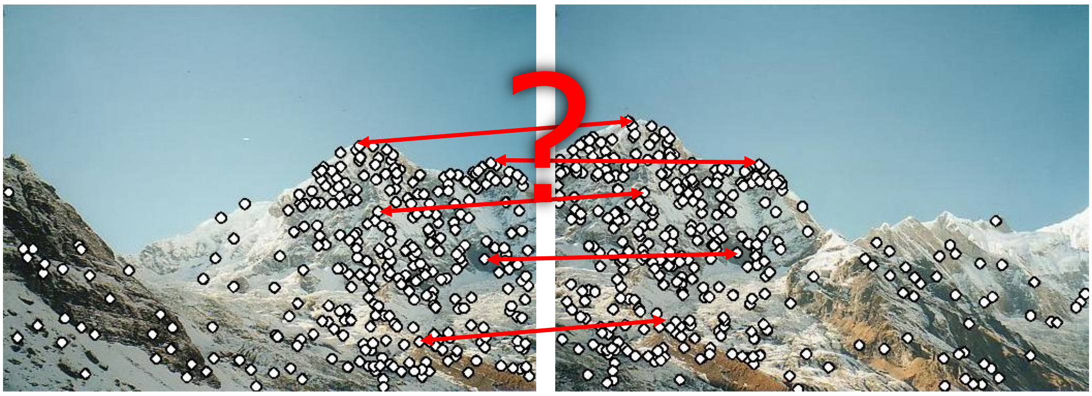


Slide credits: Noah Snavely

Feature descriptors

Localizzazione ➔ Harris corner detector

Next question: *how to match them?*



Ho bisogno di un descrittore del punto (feature descriptor) che dia:

- risposte differenti per feature differenti
- risposte simili per lo stessa feature in una vista diversa:
  - in immagini diverse può risultare più chiara/scura, ruotata, ingrandita/rimpicciolita

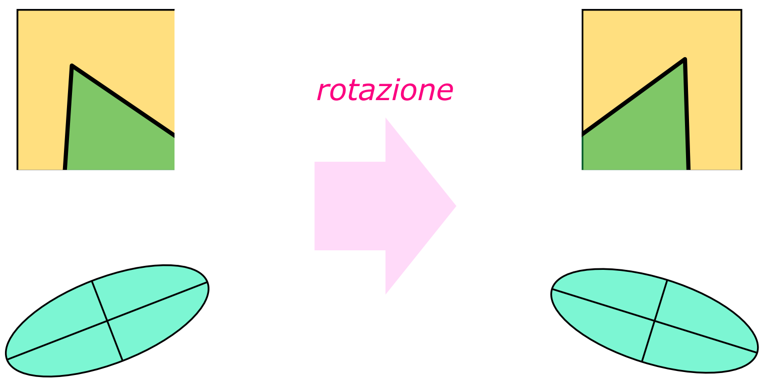
➔ risposta invariante alla luminosità, alla rotazione, alla scala

Slide credits: Noah Snavely

Harris Detector: Invariance Properties

Harris corner detector

- Proprietà rispetto alla rotazione



- L'ellisse ruota con l'immagine → gli autovettori ruotano, ma i suoi autovalori non cambiano

Harris detector: invariante alla rotazione

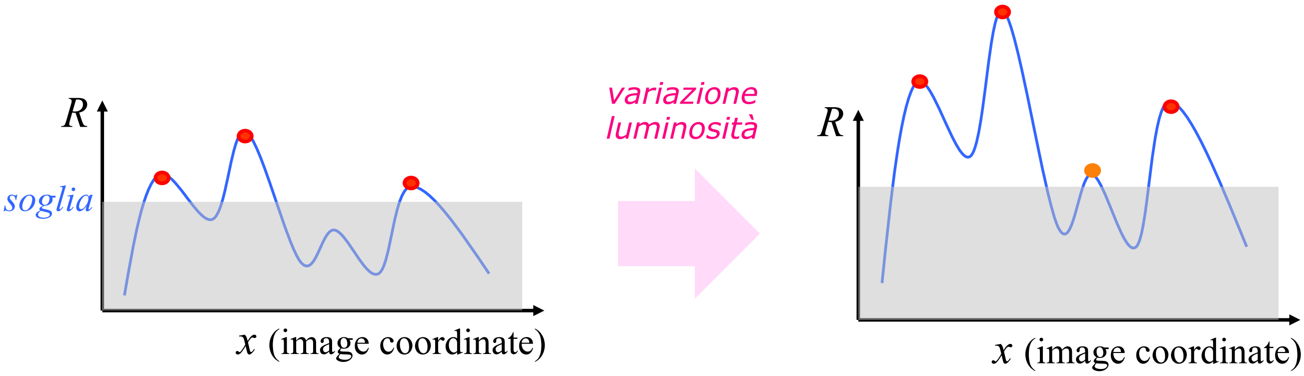
Slide credits: Noah Snavely

Harris Detector: Invariance Properties

Variazione di intensità luminosa: $\boldsymbol{I'} \rightarrow a\boldsymbol{I} + b$

- Operatore di Harris: ***H*** è funzione delle derivate prime

➔ $R = f(\boldsymbol{H}) \rightarrow I'_{x,y} = a\, I_{x,y} \rightarrow R(\boldsymbol{I'}) = a\, R(\boldsymbol{I})$



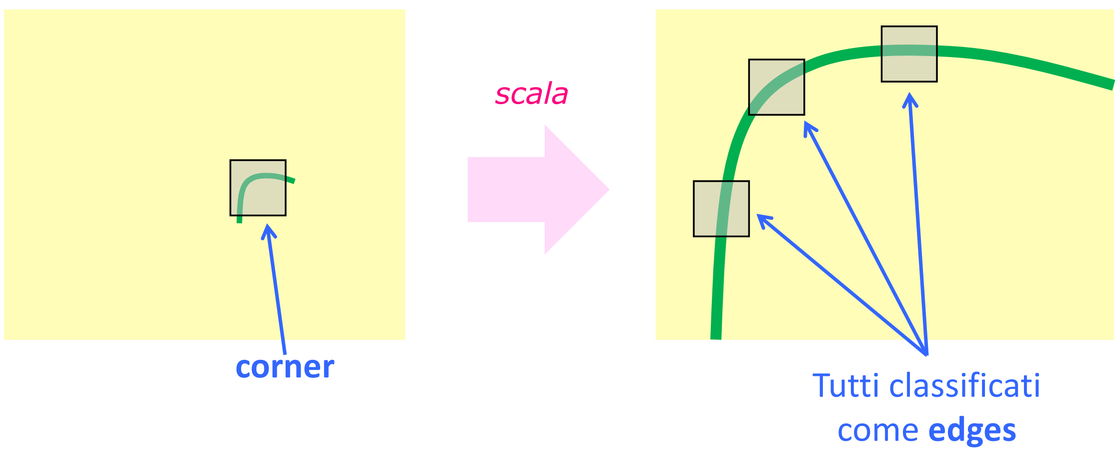
Harris: parzialmente invariante alle variazioni di intensità

Slide credits: Noah Snavely

Harris Detector: Invariance Properties

Variazione di scala (es: vista ingrandita)

- la localizzabilità di un particolare dipende dalla scala:

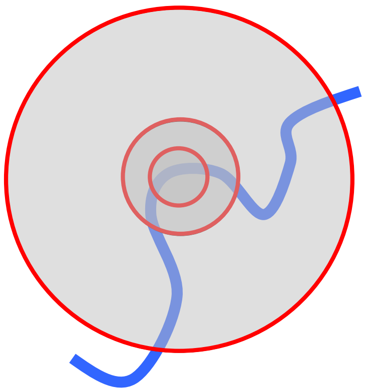


*Harris: **NON invariante** alle variazioni di **scala***

Slide credits: Noah Snavely

Scale-invariant Feature Detection

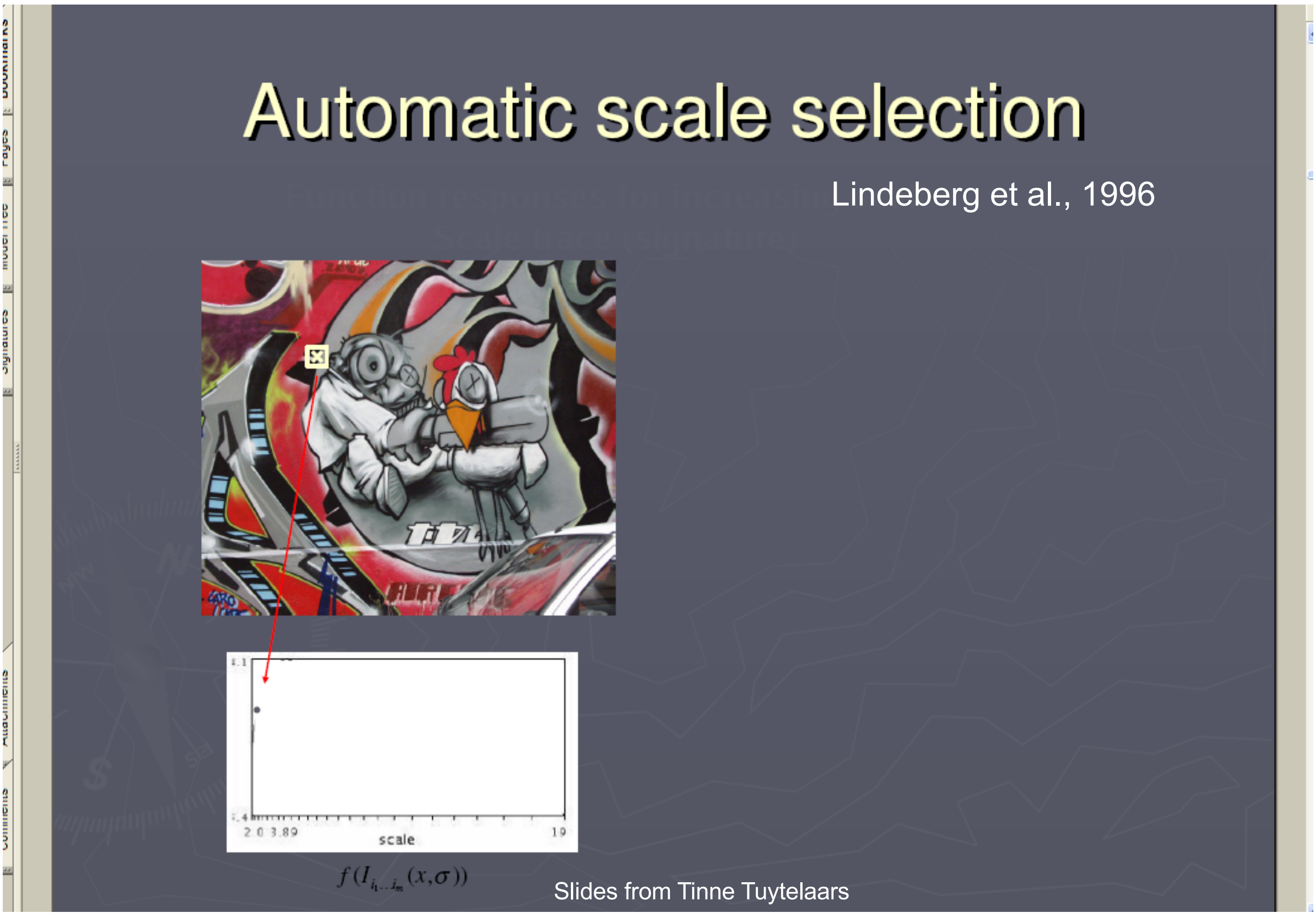
- come posso localizzare una feature, alla "sua" scala?
- qual è la scala più adatta ad una feature?



Idea: trovare la scala che fornisce il massimo valore di localizzabilità f

- massimo locale, al variare di posizione e scala
- Esempio di definizione di localizzabilità f: Harris operator

Slide credits: Noah Snavely



Automatic scale selection

$f(I_{i_1 \ldots i_m}(x,\sigma))$

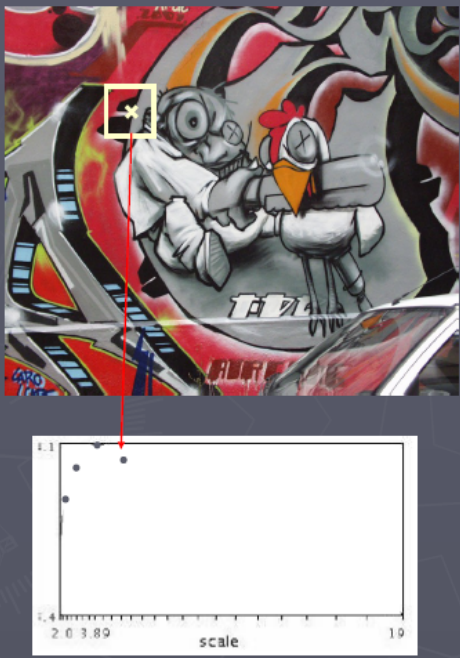
Slides from Tinne Tuytelaars

Automatic scale selection

$f(I_{i_1 \ldots i_m}(x,\sigma))$

Slides from Tinne Tuytelaars

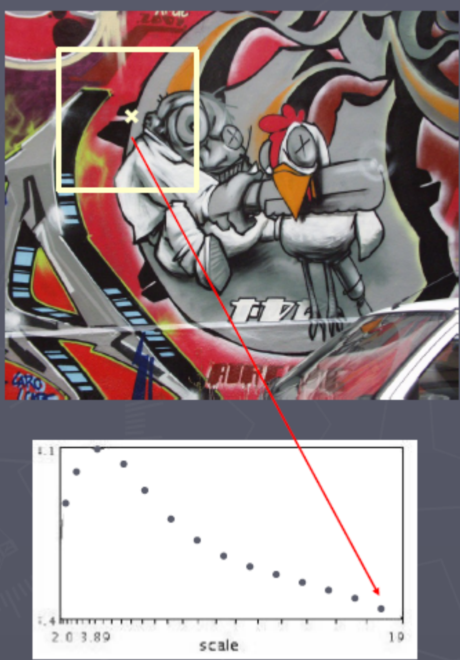
Automatic scale selection



$f(I_{i_1 \ldots i_m}(x,\sigma))$

Slides from Tinne Tuytelaars

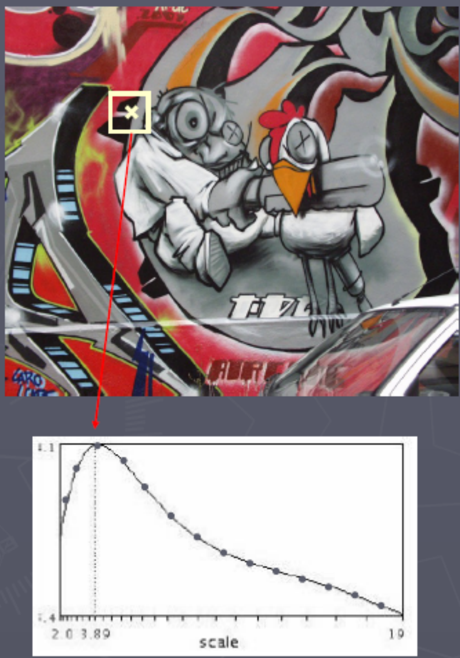
Automatic scale selection



$f(I_{i_1 \ldots i_m}(x,\sigma))$

Slides from Tinne Tuytelaars

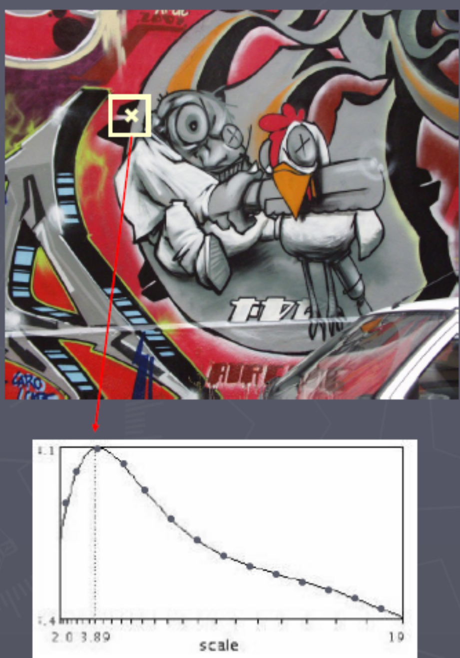
Automatic scale selection



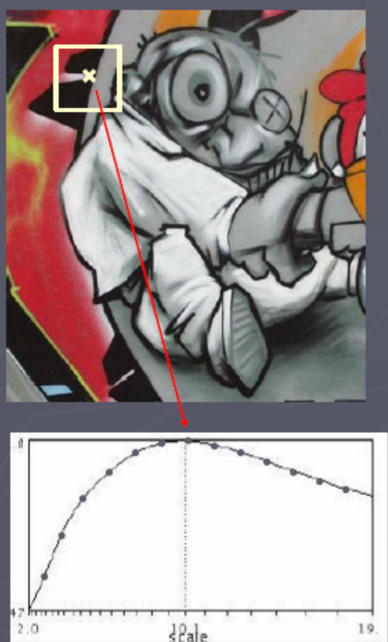
$f(I_{i_1 \ldots i_m}(x,\sigma))$

Slides from Tinne Tuytelaars

Automatic scale selection

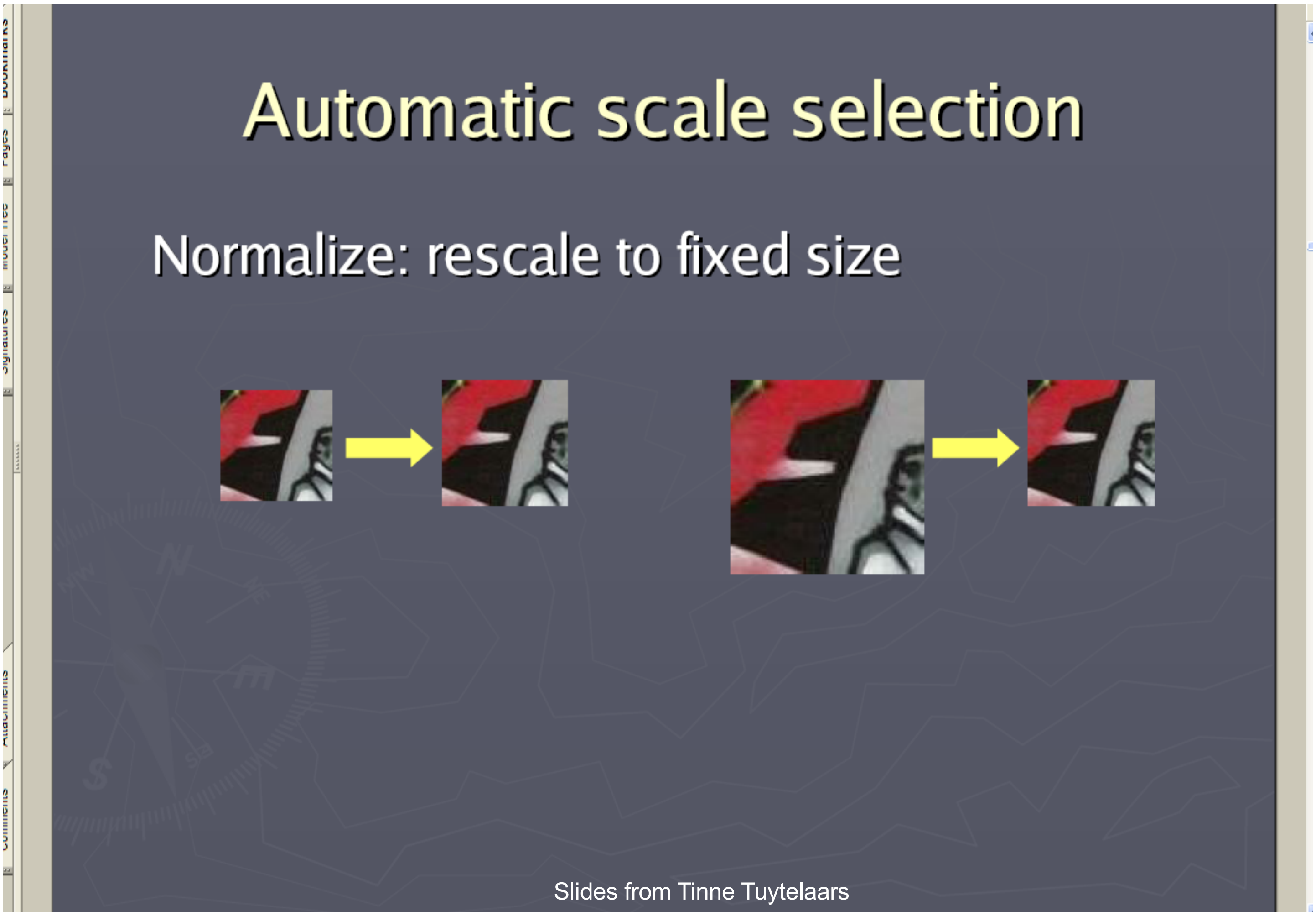


$f(I_{i_1 \ldots i_m}(x,\sigma))$



$f(I_{i_1 \ldots i_m}(x',\sigma'))$

Slides from Tinne Tuytelaars



Implementation

Implementazione mediante **piramide gaussiana**

- Anziché calcolare $f(W)$ su finestre W sempre più grandi, si può utilizzare una finestra W di dimensione fissa, su immagini progressivamente scalate (piramide gaussiana)

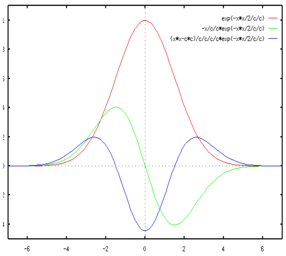
Slides from Tinne Tuytelaars

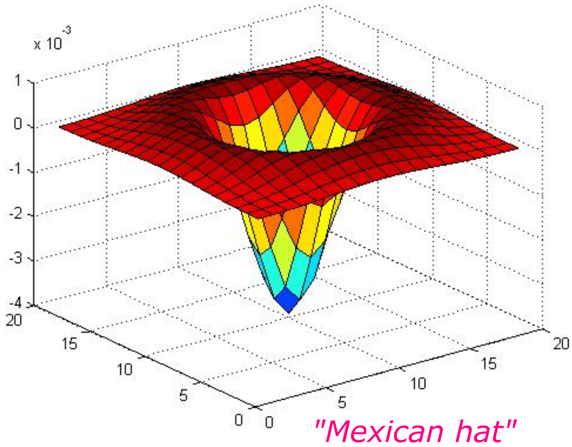
Laplacian of Gaussian

Feature detector alternativi a **Harris:**

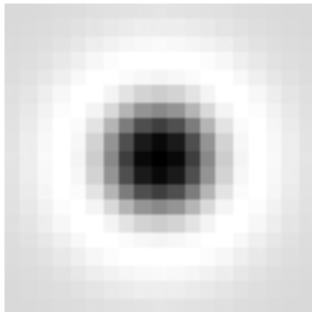
Laplacian of Gaussian (LoG)

$$g(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \Longrightarrow \quad \boxed{\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}}$$





"Mexican hat"



Slide credits: Noah Snavely

Laplacian of Gaussian

Feature detector alternativi a **Harris:**

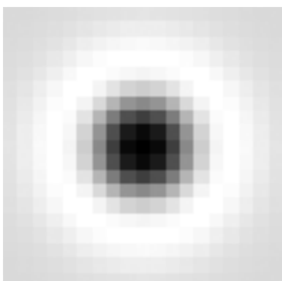
Laplacian of Gaussian (LoG)

Vantaggio: più semplice ed efficiente di Harris

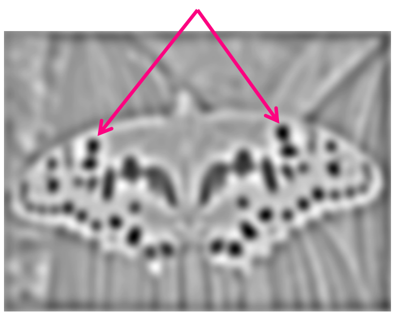
Corner/blob detector:

- Filtraggio immagine con **LoG:** $L_\sigma(x,y) = I(x,y) * \nabla^2 g(x,y,\sigma)$
- Corner ➔ minimi locali di $L_\sigma(x,y)$

$*$



$=$



Slide credits: Noah Snavely

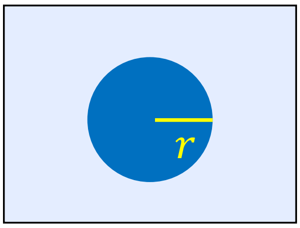
Laplacian of Gaussian – scale selection

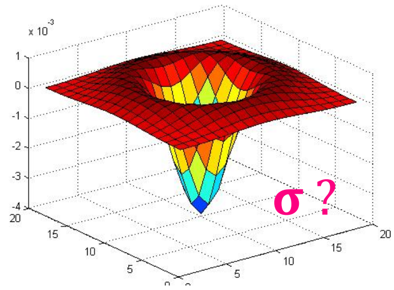
Selezione della scala

- Data la dimensione di una feature, a che scala σ LoG produce la risposta ottima?

Es: rivelazione ottima di un cerchio di raggio r

image

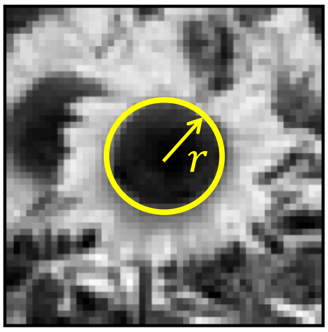


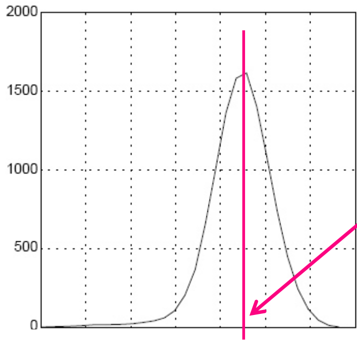


Laplacian

Scala caratteristica: la scala che produce la risposta massima del LoG

T. Lindeberg (1998). Feature detection with automatic scale selection. *International Journal of Computer Vision* **30** (2) pp 77—116, 1998.





Scala caratteristica:

$$\sigma \approx \frac{r}{\sqrt{2}}$$

Slide credits: Noah Snavely

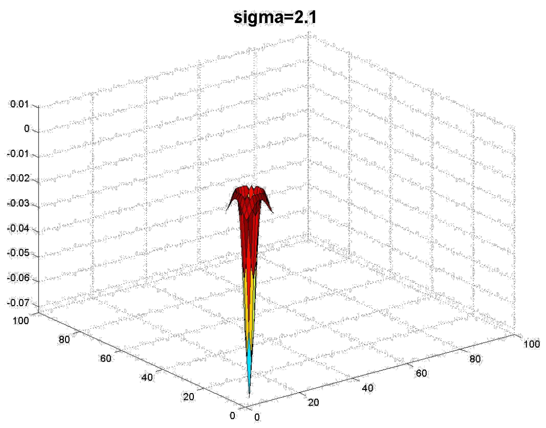
Scale-invariant feature detection

Esempio: blob detection

Original image

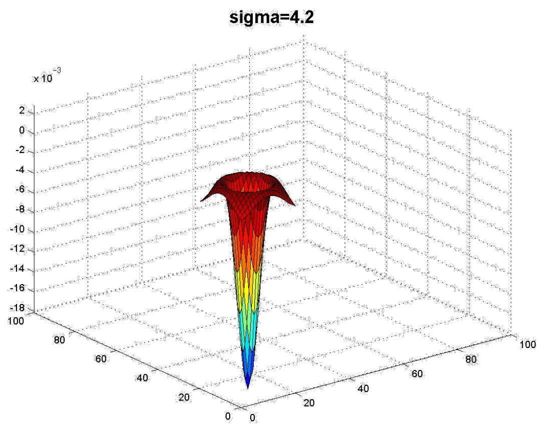
Esempio: blob detection

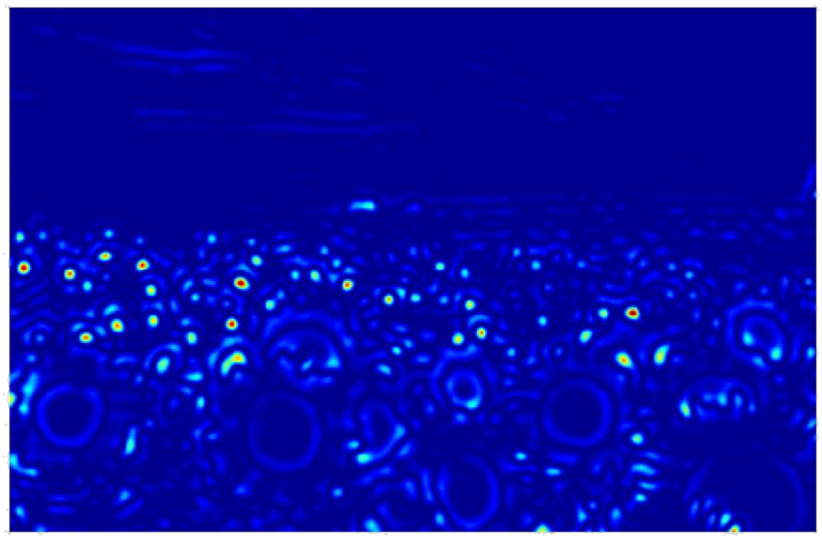
Filtraggio con **LoG:** $\sigma = 2.1$



Esempio: blob detection

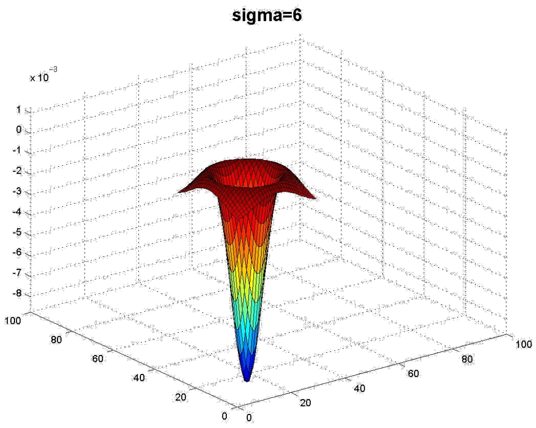
Filtraggio con **LoG:** $\sigma = 4.2$

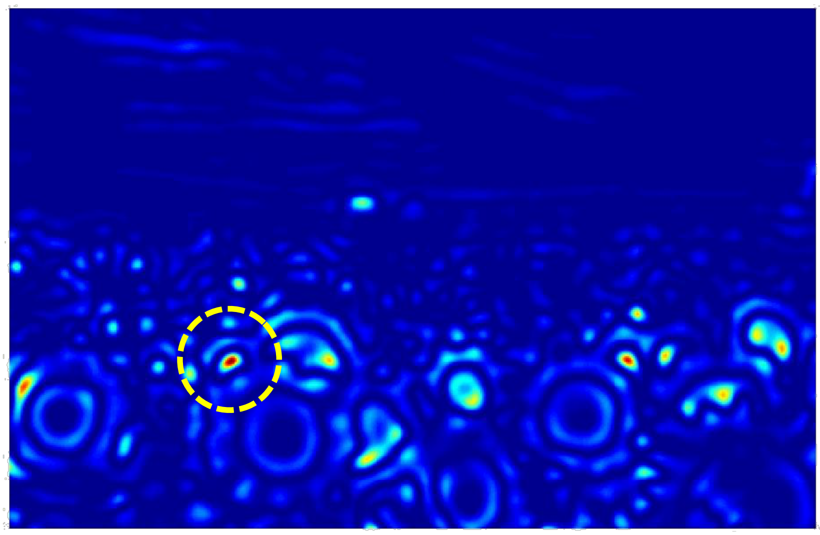




Esempio: blob detection

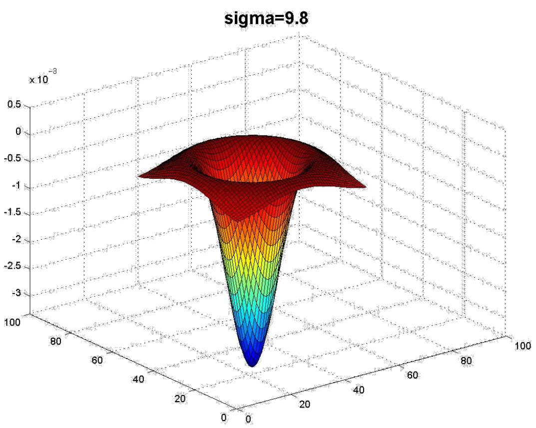
Filtraggio con **LoG:** $\sigma = 6$

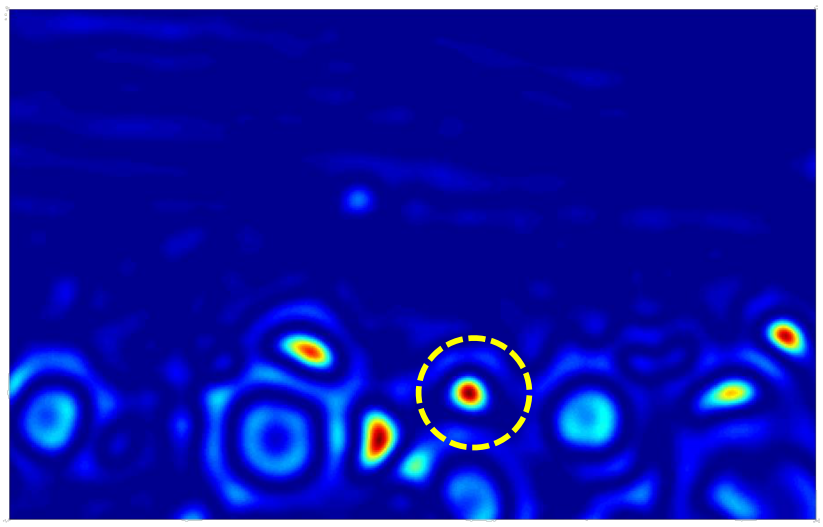




Esempio: blob detection

Filtraggio con **LoG:** $\sigma = 9.8$

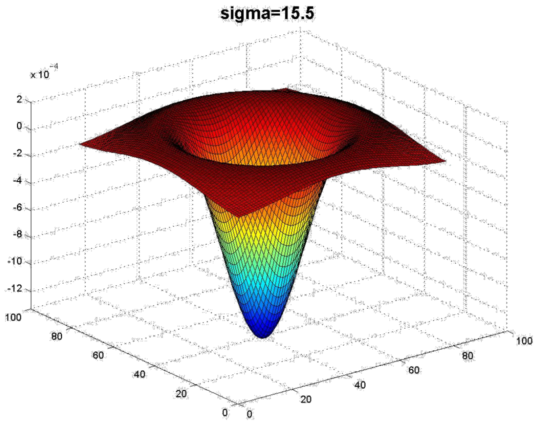


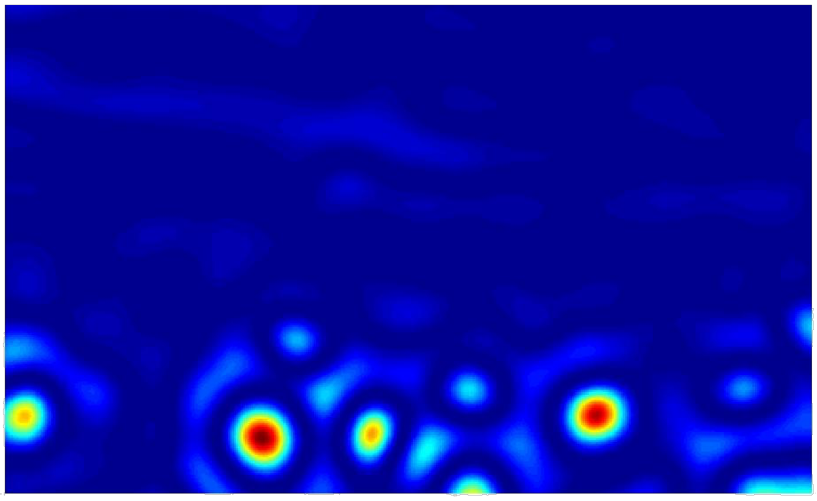


Scale-invariant feature detection

Esempio: blob detection

Filtraggio con **LoG:** $\sigma = 15.5$





Difference of Gaussians (DoG)

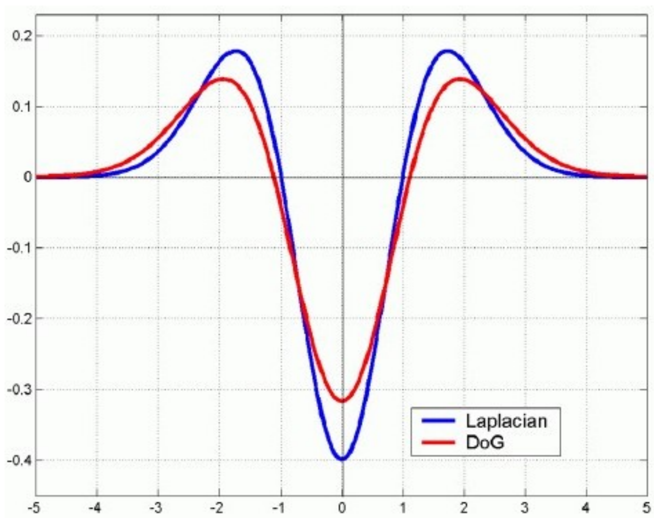
- il filtro ***LoG*** può essere ben approssimato da una differenza di due Gaussiane con σ differente

Laplacian of Gaussian:

$$LoG = \sigma^2 \left(\frac{\partial^2 g(x,y,\sigma)}{\partial x^2} + \frac{\partial^2 g(x,y,\sigma)}{\partial y^2} \right)$$

Difference of Gaussian (DoG):

$$DoG = g(x,y,k\sigma) - g(x,y,\sigma)$$



Vantaggi:

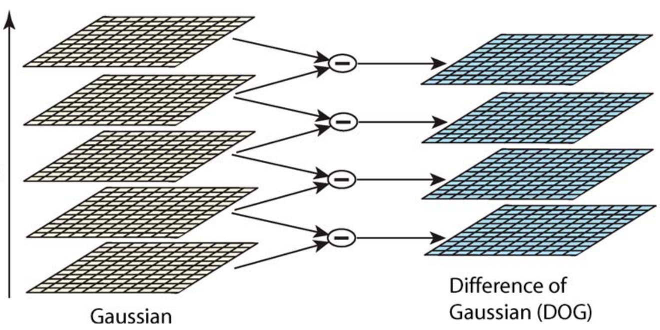
- **DoG** è più semplice → computazionalmente più efficiente

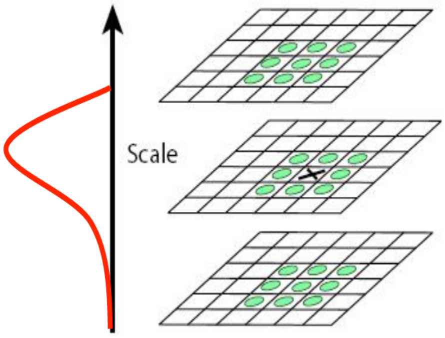
SIFT: Scale-Invariant Feature Transform

SIFT – Scale-Invariant Feature Transform [Lowe 2004]
Tecnica di feature detection invariante al punto di osservazione

SIFT method:

- Costruzione efficiente dello *scale space* (Gaussian pyramid → DOG)
- Ricerca features (*keypoints*) nello spazio (x, y, σ) (DOG)
- Caratterizzazione di ogni feature in uno spazio a 32-128 dimensioni





David G. Lowe. **Distinctive image features from scale-invariant keypoints**. *Int. J. Computer Vision*, 60 (2), pp. 91-110, 2004.

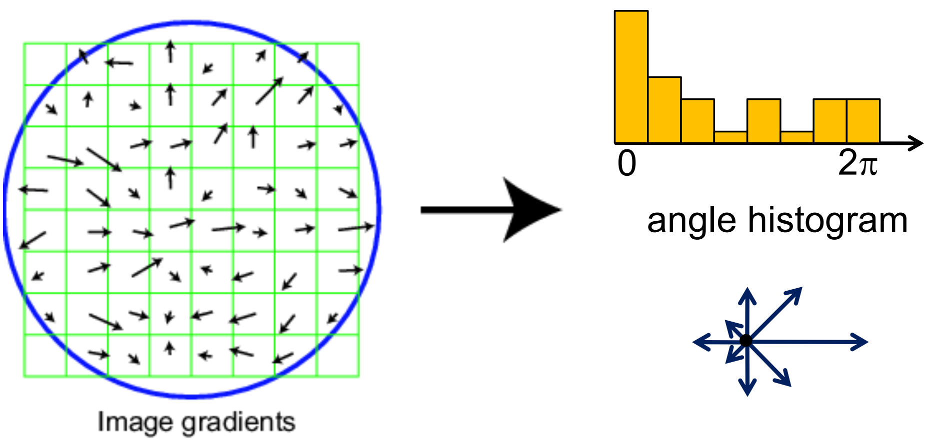
Adapted from slide by David Lowe

SIFT: Scale-Invariant Feature Transform

Adapted from slide by David Lowe

SIFT method (continua):

- Considero una finestra 8x8 (16x16) centrata sulla feature
- Calcolo orientazione dell'edge (dir. gradiente – 90º) di ogni pixel
- Eliminazione edge "deboli" (sogliatura del modulo del gradiente)
- Calcolo istogramma delle orientazioni degli edge superstiti
  (pesando i contributi con una finestra gaussiana centrata sulla feature)



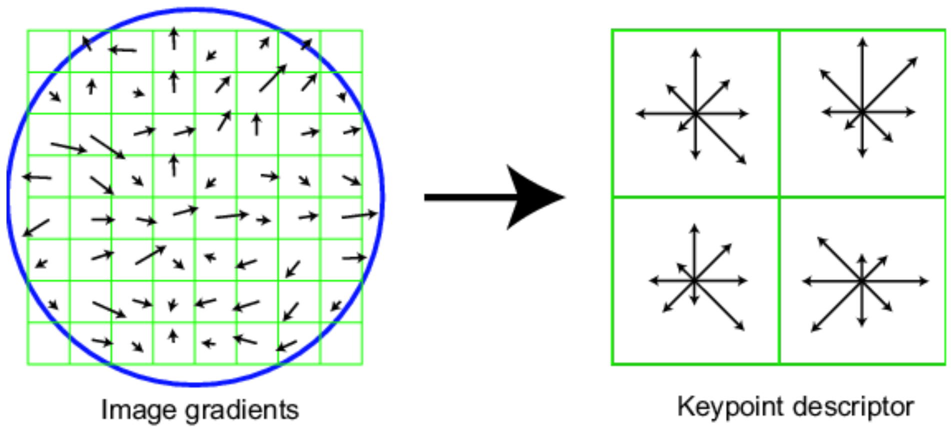
David G. Lowe. **Distinctive image features from scale-invariant keypoints**. *Int. J. Computer Vision*, 60 (2), pp. 91-110, 2004.

SIFT descriptor

Adapted from slide by David Lowe

SIFT method (continua):

- Suddivisione della finestra 8x8 (16x16) in una griglia di 2x2 (4x4) celle
- Calcolo dell'istogramma delle orientazioni per ogni cella
  - Ogni cella corrisponde a 4x4 pixel della finestra originaria
  - Le direzioni sono ruotate rispetto alla orientazione del keypoint (invarianza alla rotazione)
- 4 (16) celle * 8 orientazioni ➔ descrittore a 32 (128) dimensioni
  - il descrittore viene normalizzato a lunghezza unitaria (invarianza all'illuminazione)



Gestione delle ambiguità

Gestione delle ambiguità di matching

- Come decidere i casi ambigui?

➔ Calcolo la somiglianza (SSD) delle finestre candidate al match

- f_2 is best SSD match to f_1 in I_2
- f_2' is 2nd best SSD match to f_1 in I_2
- f_2'/f_2 fornisce valori alti per match ambigui

I_1 I_2

Gestione delle ambiguità

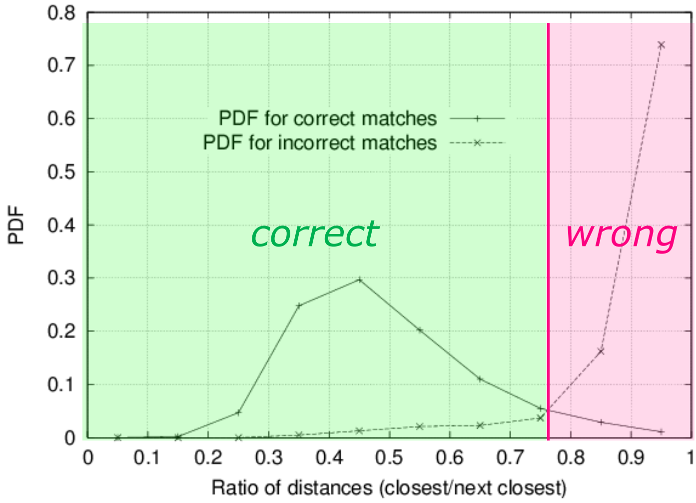
Se ho più candidati, come determinare il match corretto?

➔ un match corretto ha una somiglianza (SSD) molto maggiore di quelli errati

Idea: confronto tra la somiglianza del miglior candidato al match e quella del secondo miglior candidato

$$Ratio = \frac{SSD(best\ match)}{SSD(2^{nd}\ best\ match)}$$

- Low ratio
  ➔ probabile match corretto
- High ratio
  ➔ probabile ambiguità
- Threshold $T = 0.8$
  provides good separation



SIFT

SIFT: prestazioni

Tecnica di riconoscimento (e matching) di features molto robusta

- robusta ai cambi di prospettiva
- robusta alle differenze di illuminazione
- *Fast and efficient (can run in real time)*

http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT