# *Low Level Vision*

## Feature points: (Corner Detection, SIFT)

## Feature Points

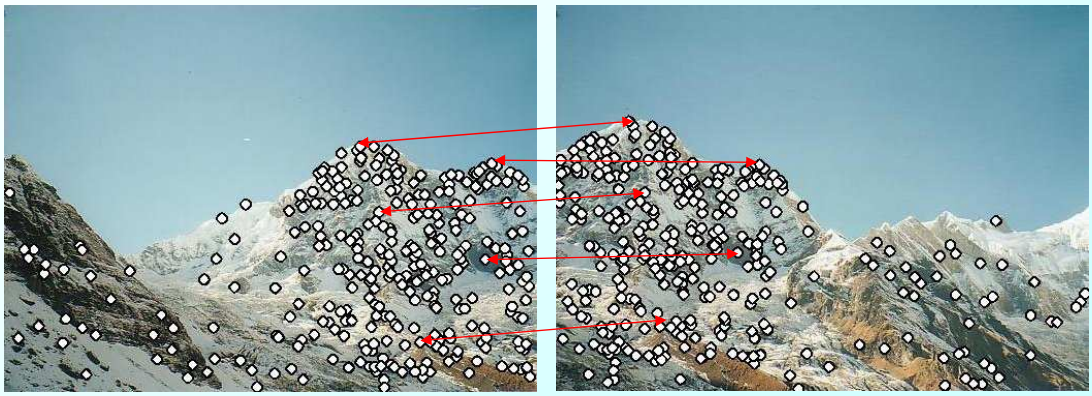
Analizzeremo adesso i cosiddetti ***feature points***, cioè i punti caratteristici di una immagine/scena che per la loro particolare natura (invarianza rispetto a ….) possono essere usati per molteplici scopi:

- Image alignment (homography, fundamental matrix)
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation

… altro!

# Matching

Per esempio data una coppia di immagini è possibile individuare i *feature points* in entrambe e quindi cercare una corrispondenza (matching) opportuna.

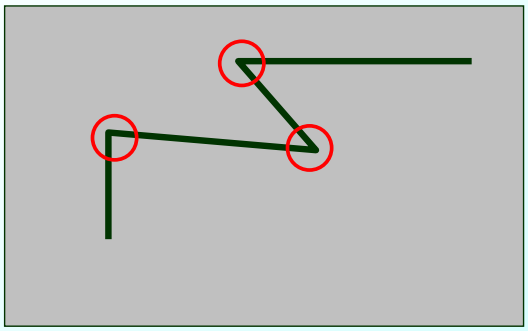


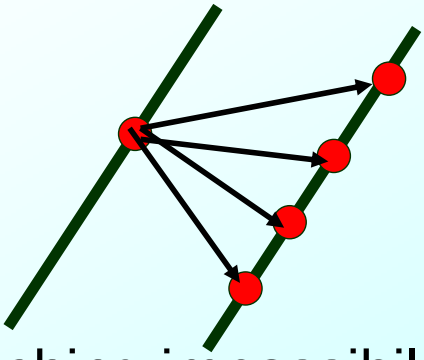
# Matching

E' quindi possibile comporre un "panorama" ottenuto utilizzando le coppie di "feature points" per allineare in maniera corretta le immagini:

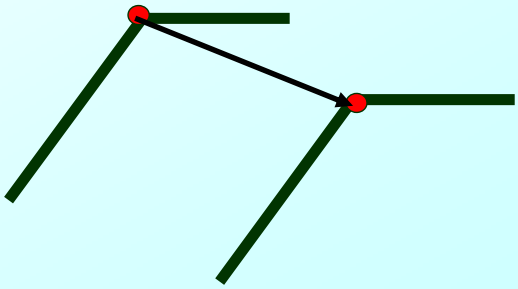
# Corner

I corner sono appunto dei punti caratteristici, che per loro natura si prestano bene ad essere utilizzati come ***feature points***.



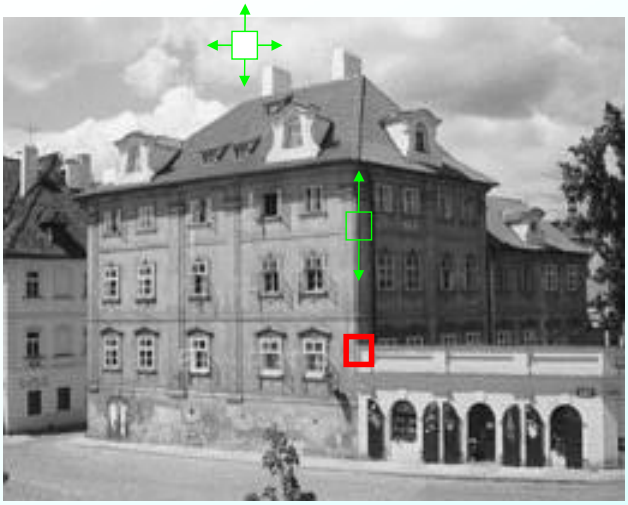


Matching impossibile



Matching più semplice

# Esempio

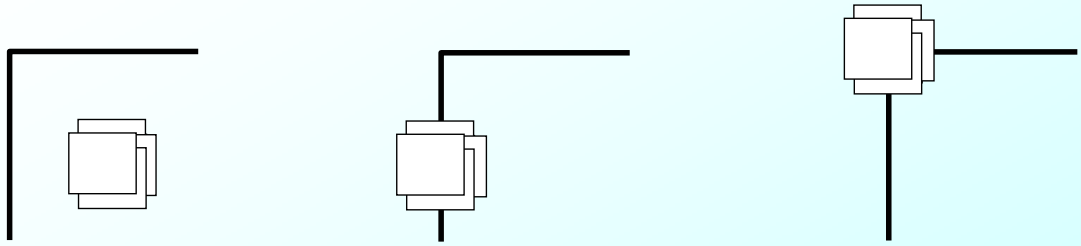


**Patches non caratterizzanti**

**Patches caratterizzanti**

# Moravec Corner Detector

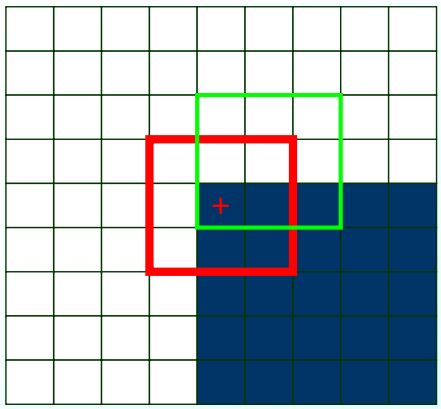
Considera una porzione di immagine (***Shifting Window***) che comprende il punto in esame e ne calcola la similarità con le porzioni vicine tramite **SSD** (**Sum of Squared Differences**).

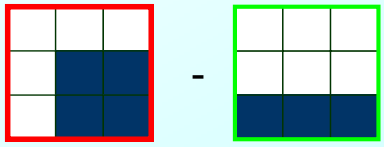


**Regione omogenea:** ***non*** ci sono variazioni in tutte le direzioni.

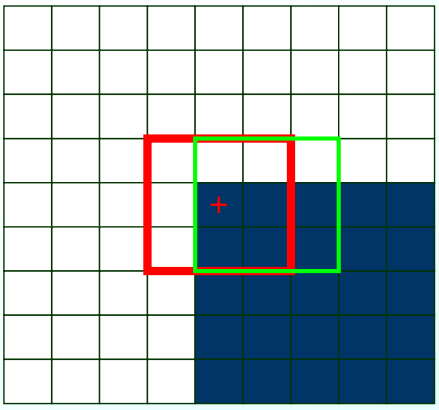
**Edge:** ***non*** ci sono variazioni lungo la direzione dell'edge.

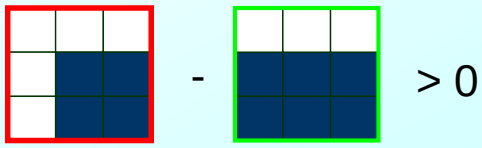
**Corner:** ci sono variazioni significative in tutte le direzioni.



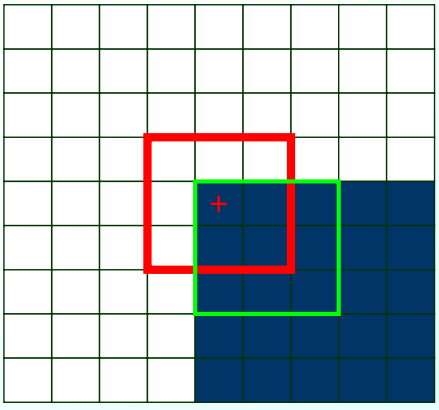


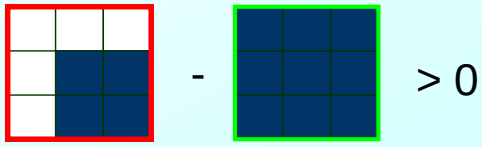

Demo su un punto di corner con un vicinato (**neighbourhood**) ben distinto



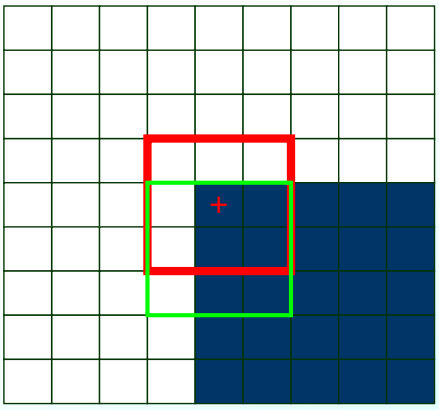


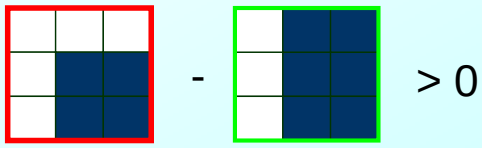
Demo su un punto di corner con un vicinato (**neighbourhood**) ben distinto



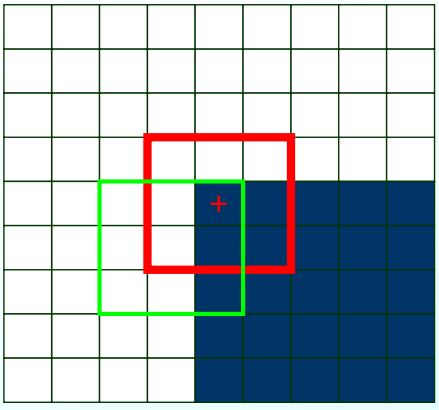


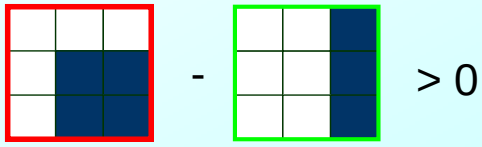
Demo su un punto di corner con un vicinato (**neighbourhood**) ben distinto



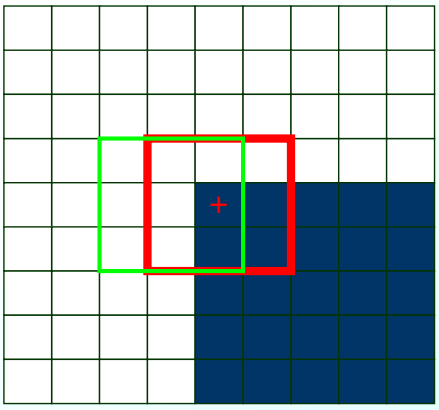


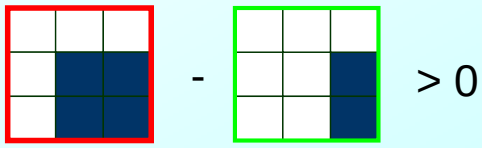
Demo su un punto di corner con un vicinato (**neighbourhood**) ben distinto



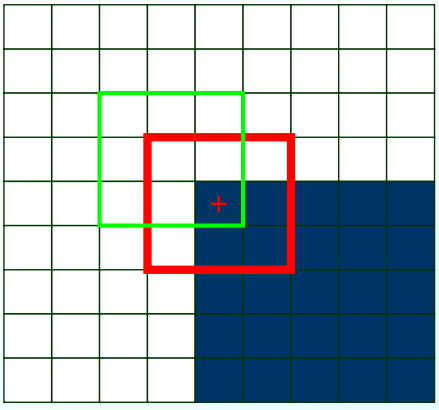


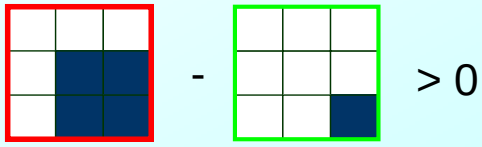
Demo su un punto di corner con un vicinato (**neighbourhood**) ben distinto



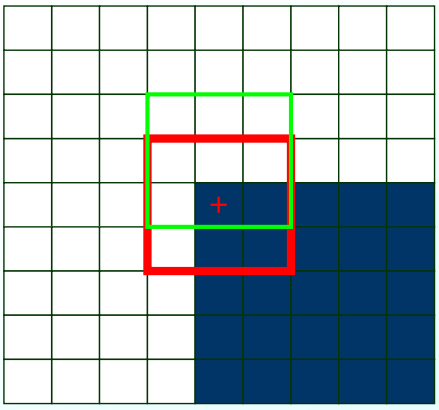


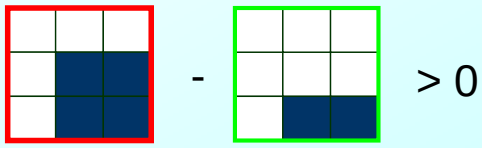
Demo su un punto di corner con un vicinato (**neighbourhood**) ben distinto





Demo su un punto di corner con un vicinato (**neighbourhood**) ben distinto



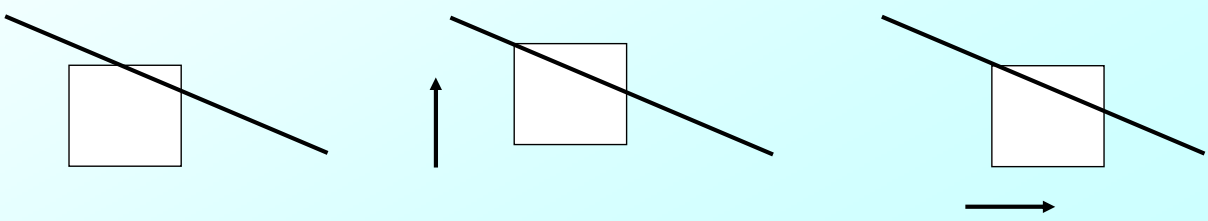



Demo su un punto di corner con un vicinato (**neighbourhood**) ben distinto

## Moravec Corner Detector

Ad ogni punto viene associata una corner **strength (intensità/forza)** pari al minimo tra tutte le SSD calcolate tra la porzione di immagine in esame e quelle vicine (le direzioni esaminate sono quella orizzontali, verticale e le due diagonali).

**Limiti:** l'operatore definito da Moravec non è isotropo. Se un edge è presente in una direzione diversa da quelle analizzate potrebbe essere confuso con un corner.



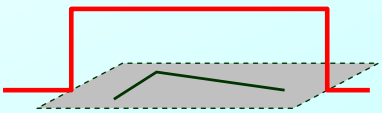
# Harris Corner Detector

Harris e Stephens migliorano l'approccio di Moravec evitando di calcolare direttamente la funzione SSD come differenza tra porzioni di immagini vicine. Dato uno shift (Δx, Δy) ed un punto di coordinate (x,y) la SSD è definita come:

$$E(x,y) = \sum_{(x_k,y_k)\in W} w_k \left(I(x_k,y_k) - I(x_k+\Delta x, y_k+\Delta y)\right)^2$$

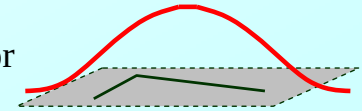
Con $(x_k,y_k)$ punti della finestra W centrata in (x,y), *I* l'immagine in esame e $w_k$ peso associato al punto:

Window function $w_k(x,y)$ =



1 in window, 0 outside

or



Gaussian

# Harris Corner Detector

Per evitare di calcolare la funzione nelle varie direzioni (approccio di Moravec) si considera il seguente sviluppo in serie di Taylor:

$$I(x_k+\Delta x, y_k+\Delta y) \approx$$

$$\approx I(x_k,y_k) + \left(I_x(x_k,y_k)\; I_y(x_k,y_k)\right)\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix}$$

# Harris Corner Detector

Utilizzando questa approssimazione e facendo un pò di conti la funzione SSD diventa:

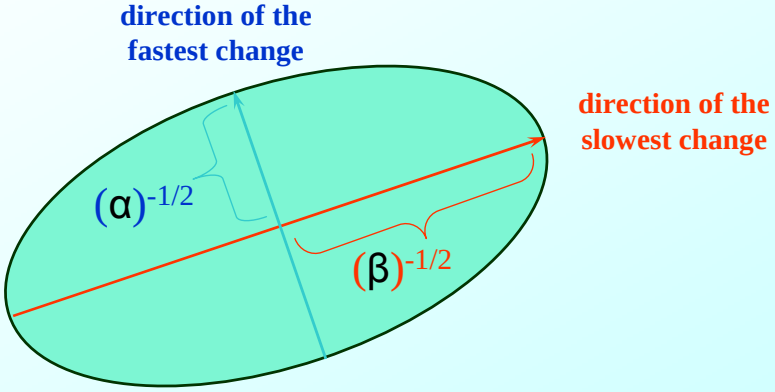
$$E(x,y) = (\Delta x \quad \Delta y) A(x,y) \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

$$A(x,y) = \begin{bmatrix} \sum_{(x_k,y_k)\in W} w_k \left(I_x(x_k,y_k)\right)^2 & \sum_{(x_k,y_k)\in W} w_k I_x(x_k,y_k) I_y(x_k,y_k) \\ \sum_{(x_k,y_k)\in W} w_k I_x(x_k,y_k) I_y(x_k,y_k) & \sum_{(x_k,y_k)\in W} w_k \left(I_y(x_k,y_k)\right)^2 \end{bmatrix}$$

La funzione trovata è strettamente legata alla funzione di autocorrelazione locale. La matrice A ne descrive la forma nell'origine.

# Harris Corner Detector

Gli autovalori $\alpha$, $\beta$ della matrice A sono proporzionali alle ***curvature*** principali dell'autocorrelazione locale e formano un descrizione **invariante per rotazione della matrice A**.



## Harris Corner Detector

Gli autovalori $\alpha$, $\beta$ della matrice A sono proporzionali alle ***curvature*** principali dell'autocorrelazione locale e formano un descrizione **invariante per rotazione della matrice A**.

**Tre** casi possono essere considerati:

- se entrambe le curvature sono **piccole** (la funzione di autocorrelazione è piatta), la regione in esame ha intensità approssimativamente **costante**;
- se una curvatura è **elevata** e l'altra è **bassa** c'è un **edge**;
- se **entrambe** le curvature sono **elevate** c'è un **corner**.

## Harris Corner Detector

Ad ogni punto dell'immagine *I* viene associato un valore ***R*** (chiamato corner response) funzione degli autovalori $\alpha$ e $\beta$ (viene così mantenuta l'invarianza rotazionale).

*R* viene così definito:

$$R=Det(A)-kTr(A)^2$$

Con *Det(A)* e *Tr(A)* determinante e traccia della matrice A. Si ricorda che:

$$Det(A)=\alpha\beta$$

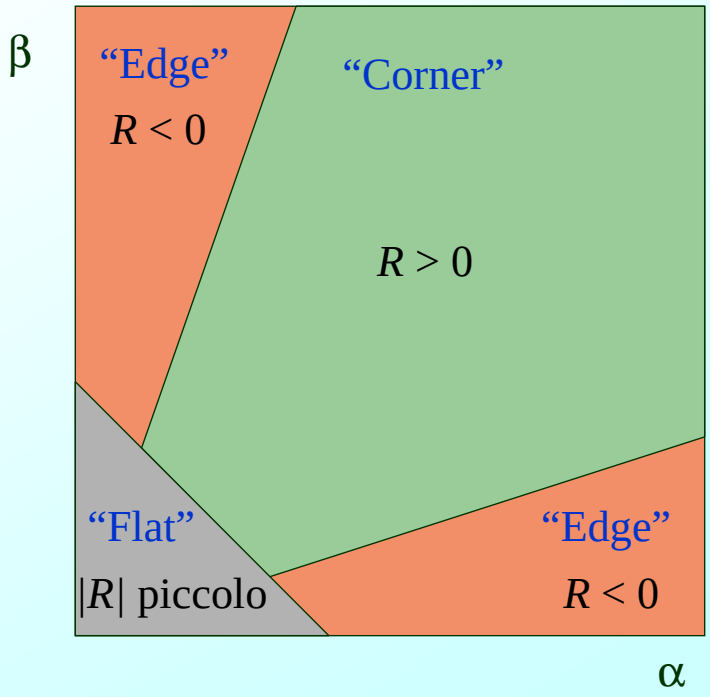
$$Tr(A)=\alpha+\beta$$

Il valore della costante *k* viene generalmente fissato nell'intervallo 0.04-0.06.

# Harris Corner Detector

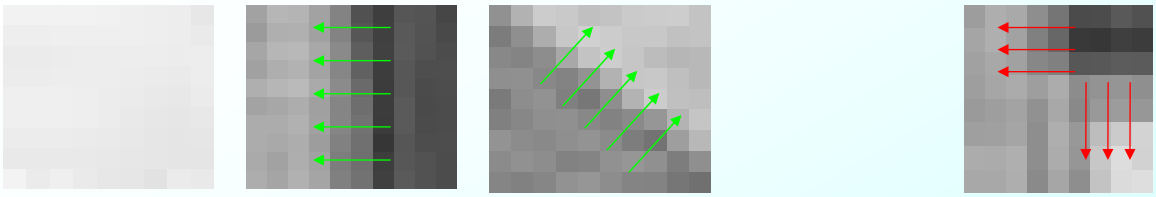
$$R = \alpha\beta - k(\alpha+\beta)^2$$

- *R* è grande per un **corner**
- *R* è negativo con valori elevati per un **edge**
- |*R*| è piccolo per una regione **piatta**





# Harris Corner Detector

Riepilogando l'Harris corner detector:

- considera i punti con elevata corner response (R maggiore di una soglia);
- Classifica come punti interessanti quelli che sono massimi locali di *R*.

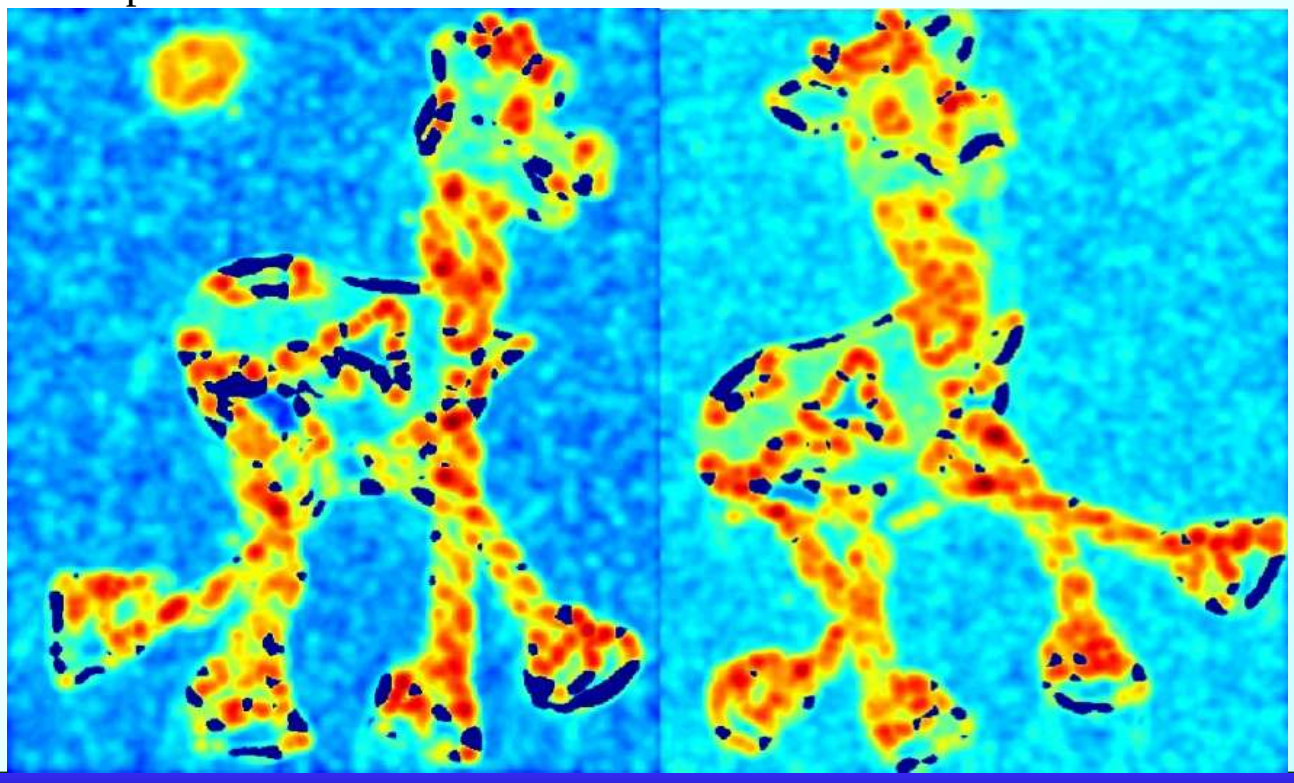
# Harris Corner Detector (esempio 1)

# Harris Corner Detector (esempio 1)

# Harris Corner Detector (esempio 2)

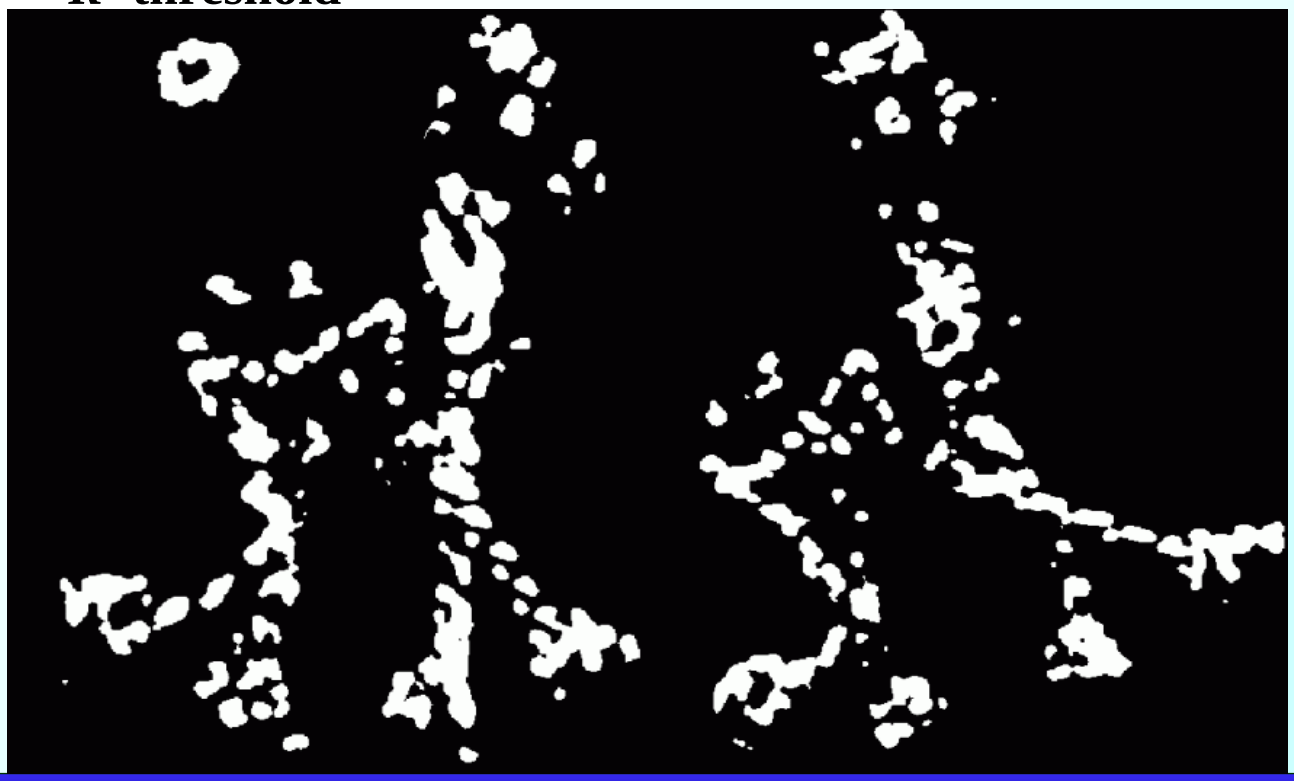
# Harris Corner Detector (esempio 2)

Risposta ai corner $R$



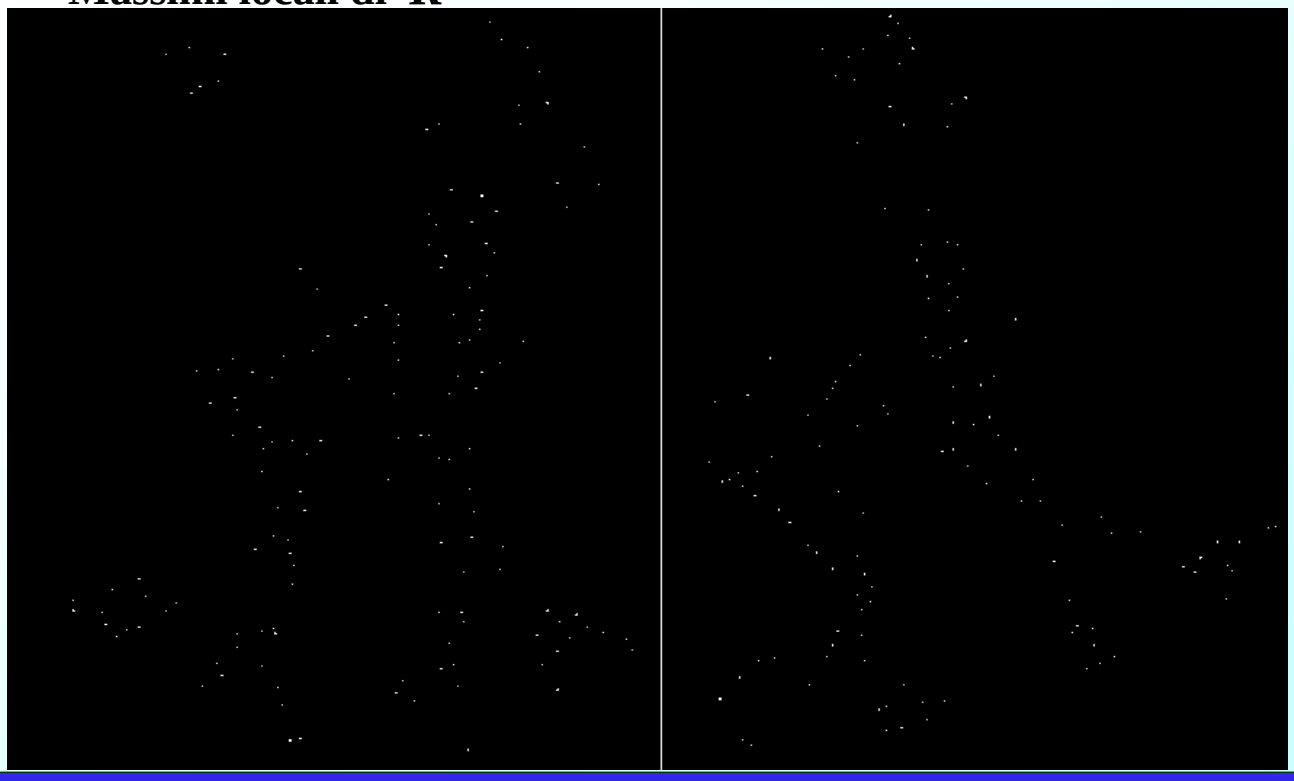
# Harris Corner Detector (esempio 2)

$R$>threshold



# Harris Corner Detector (esempio 2)

Massimi locali di $R$

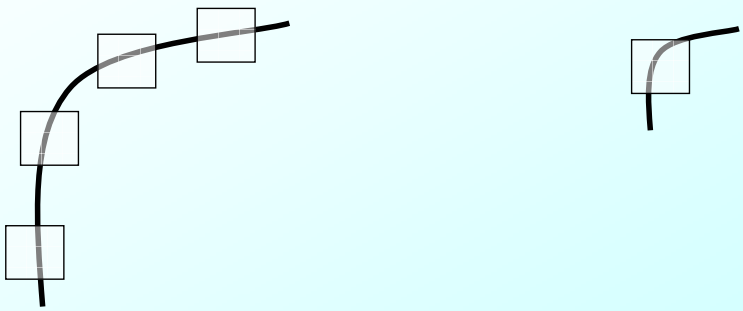


# Harris Corner Detector (esempio 2)

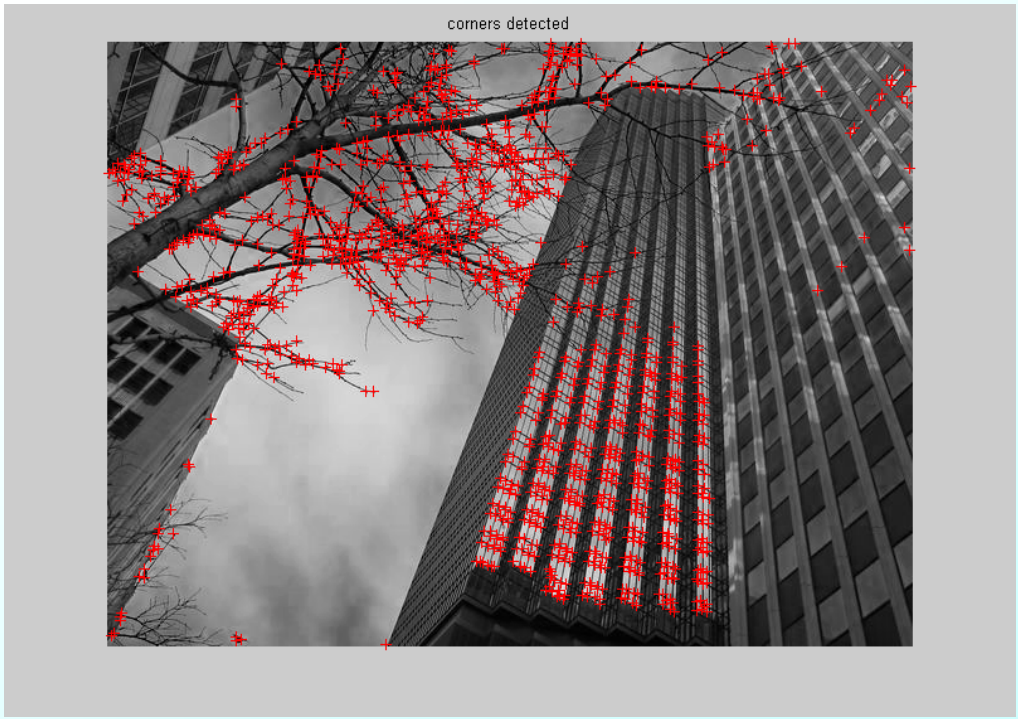
# Harris Corner Detector

L'Harris corner detector risulta essere invariante alle rotazioni e parzialmente invariante ai cambi di illuminazione.

**Limiti**: non è invariante alla scala.



Lo stesso arco a scale differenti viene riconosciuto come edge (a sinistra) e corner (a destra).




Attenzione alla scala!!!

## Demo e Materiale on-line

Confronto tra tecniche per la ricerca dei corner:

http://rfv.insa-lyon.fr/~jolion/Cours/ptint.html

MATLAB and Octave Functions for Computer Vision and Image Processing

http://www.csse.uwa.edu.au/~pk/research/matlabfns/

# Rappresentazione Multiscala


Tutti gli oggetti che ci circondano esistono come entità significative solo all'interno di un certo range di scale. Se ad esempio consideriamo il concetto di albero, esso non ha significato se utilizziamo scale dei kilometri o dei centimetri. In questi casi sarebbe più opportuno parlare di foreste o di foglie.

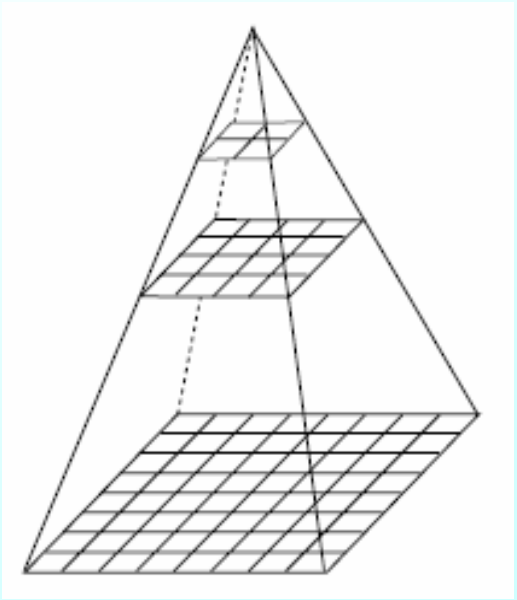
# Rappresentazione Multiscala

Nell'analisi delle immagini, molto spesso, vengono utilizzati degli operatori locali la cui bontà dei risultati dipende fortemente dalla relazione tra la grandezza degli operatori applicati e quella delle strutture dell'immagine da analizzare.

Nel caso in cui non si abbiano informazioni a priori risulta utile considerare delle rappresentazioni **multiscala** dell'immagine.

# Piramidi Gaussiane

La piramide è una collezione di rappresentazioni di un'immagine. In genere ogni livello ha dimensioni pari ad un quarto rispetto al precedente.



In una piramide Gaussiana ogni livello è smussato tramite la convoluzione con un kernel gaussiano e campionato per ottenere il layer successivo.

Sia S l'operatore di campionamento e $G_\sigma$ il kernel gaussiano, la relazione che lega due livelli successivi è:

$$P(I)_{n+1}=S(G_\sigma * P(I)_n)$$

$$P(I)_1=I$$

# Piramide Gaussiana (esempio)

## Piramide Gaussiana (applicazione)

**Ricerca di corrispondenze tra punti in immagini differenti** (visione stereo, analisi del moto).

La ricerca di tali corrispondenze è inefficiente se fatta nell'immagine originale (molti punti da esaminare). Un approccio migliore consiste nell'effettuare la ricerca a partire da una versione smussata e sottocampionata dell'immagine e successivamente raffinare la stima guardando nelle versioni più dettagliate dell' immagine.

## Scale-Space

La scale-space è una rappresentazione multiscala basata su un parametro di scala **continuo**. Essa viene ottenuta attraverso la convoluzione del segnale con una famiglia di kernel Gaussiani con σ crescente.

Dato un segnale $f : \Re^N \to \Re$

la sua rappresentazione multiscala $L : \Re^N \times \Re_+ \to \Re$

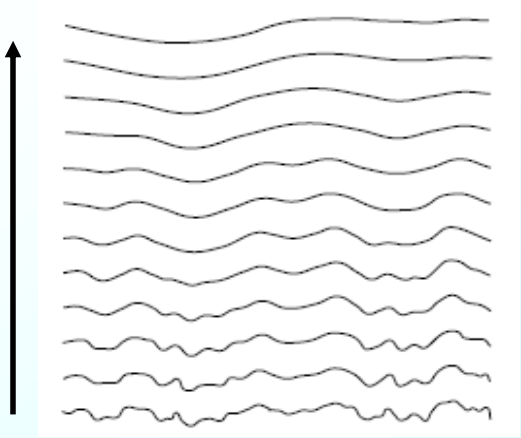
è definita da: $L(\cdot, t) = g(\cdot, t) * f \qquad L(\cdot, 0) = f$

con $t \in \Re_+$ parametro di scala,

$g : \Re^N \times \Re_+ \setminus \{0\} \to \Re$ kernel gaussiano $g(x;t) = \frac{1}{(2\pi t)^{\frac{N}{2}}} e^{-\frac{x^T x}{2t}}$

$x \in \Re^N$ con deviazione standard σ pari a: $\sigma = \sqrt{t}$

# Scale-Space



Segnale monodimensionale smussato tramite la convoluzione con una famiglia di kernel gaussiani con valori di σ crescenti.

L'idea che sta alla base della rappresentazione scale-space è quella di creare una famiglia di segnali nei quali le informazioni fine-scale vengono progressivamente soppresse.

# Scale-Space (esempio)

Immagine originale

σ=1

σ=2

σ=4

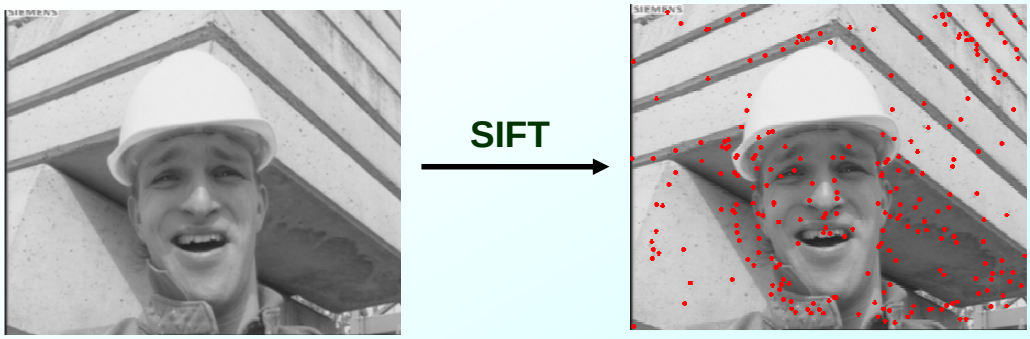
## Scale-Space (altro esempio)

## Scale Invariant Feature Transform (SIFT)

- *Scale-invariant Feature Transform* (SIFT) è un algoritmo/tecnica pubblicata da David Lowe nel 2004 per l'estrazione di *feature* (point-based) da immagini.

- Le feature estratte risultano essere:
  - invarianti alla scala dell'immagine, alla rotazione, alla traslazione e parzialmente invarianti al cambio del punto di vista e di illuminazione.

# SIFT: Esempio




Ad ogni punto individuato viene associato un descrittore che consente di individuarlo in immagini diverse.

# SIFT (passi principali)

I passi principali dell'algoritmo di estrazione delle SIFT sono i seguenti:

- **Individuazione degli estremi locali nello scale-space:** si cercano punti interessanti su tutte le scale e posizioni dell'immagine utilizzando una funzione **DoG** (***Difference of Gaussian***). L'approccio utilizzato è quello del filtraggio in cascata (*cascade filtering approach*), che consente di determinare le posizioni e la scala delle feature candidate ad essere punti chiave e che, in un secondo momento, vengono caratterizzate con maggior dettaglio.
- **Localizzazione dei keypoint:** per ciascun punto candidato viene costruito un modello dettagliato per determinarne posizione e scala. I punti vengono inoltre selezionati secondo misure di stabilità.
- ….

## SIFT (passi principali)

- **Generazione delle orientazioni:** per ottenere l'invarianza rotazionale, ad ogni punto chiave (*keypoint*) vengono associate una o più orientazioni calcolate in base ai gradienti locali dell'immagine.
- **Generazione del descrittore:** a partire dai gradienti locali dell'immagine, alla scala selezionata e nell'intorno del punto chiave, viene costruito il descrittore.

## Estremi Locali "Particolari"

- Per trovare punti dell'immagine che sono invarianti a cambiamenti di scala si cercano *feature* stabili su tutte le scale possibili, usando una funzione di scala nota come spazio-scala (*scale-space*). Come dimostrato in letteratura, l'unica possibile funzione che costituisce un Kernel Scale-Space è la funzione Gaussiana.
- Lo Scale-Space di una immagine è definito come una funzione $L(x,y,\sigma)$ data dalla convoluzione in $x$ e $y$ di una funzione Gaussiana, variabile in scala, $G(x,y,\sigma)$ con l'immagine $I(x,y)$:

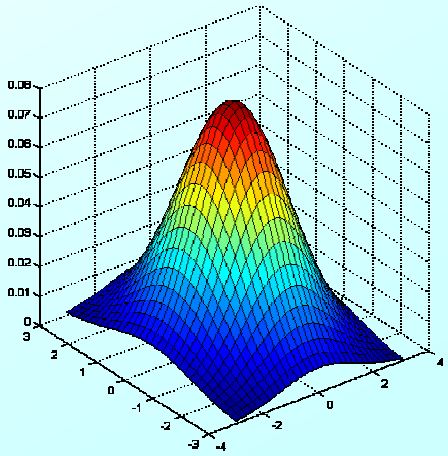
$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y)$$

# Analisi Scale-Space

- L'analisi scale-space di un'immagine è basata sul filtraggio con un filtro di scala variabile, che proietta l'immagine in uno spazio di scala;
- Si utilizza un filtro gaussiano con $\sigma$ variabile che proietta l'immagine *I(x,y)* in uno spazio tridimensionale *L(x,y,σ).*

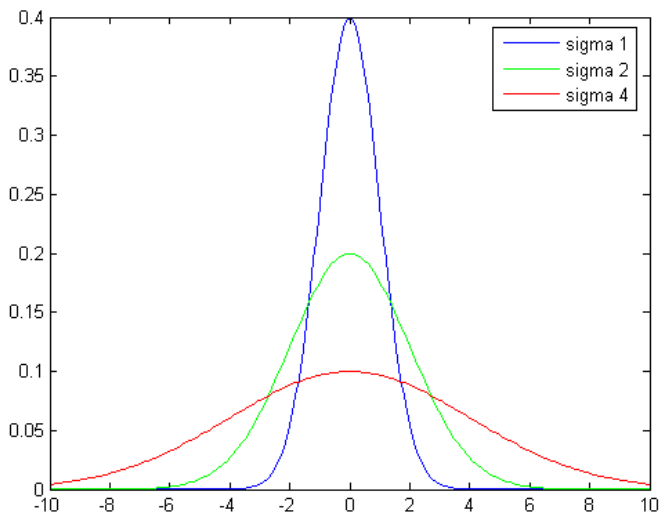
$$L(x,y,\sigma) = I(x,y) * G(x,y,\sigma)$$

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



# Analisi Scale-Space

- Convoluzione con la gaussiana, quanti campioni scegliere?




Il numero di campioni "significativi" cresce al crescere del valore di σ.

L'intervallo di campioni [-3σ, 3σ] è un buon compromesso tra accuratezza e numero non elevato di campioni.

## Analisi Scale-Space

La costruzione di uno scale-space completo può essere computazionalmente onerosa. A valori di sigma elevati corrispondono infatti kernel di convoluzione sempre più grandi.

Operazioni necessare per effettuare la convoluzione con una gaussiana (per pixel):

| σ | Kernel [-3σ, 3σ] | Somme ($N^2$-1) | Moltiplicazioni ($N^2$) |
|---|---|---|---|
| 1 | 7x7 | 48 | 49 |
| 2 | 13x13 | 168 | 169 |
| 4 | 25x25 | 624 | 625 |
| 8 | 49x49 | 2400 | 2401 |

## Analisi Scale-Space

Per diminuire il peso computazionale relativo alla costruzione dello scale-space si possono sfruttare le seguenti proprietà:

- Il kernel gaussiano è separabile;

- Il kernel gaussiano è “chiuso” rispetto all’operazione di convoluzione.

## Separabilità

- La funzione gaussiana è separabile nel prodotto di componenti mono-dimensionali. Tale proprietà permette il calcolo della convoluzione con un kernel gaussiano NxN tramite una serie di due convoluzione mono-dimensionali.

- Dal punto di vista computazionale la convoluzione con un kernel NxN comporta $N^2$ moltiplicazioni e $N^2$-1 addizioni. Due convoluzioni monodimensionali comportano invece 2*N moltiplicazioni e 2N-2 somme.

  La complessità computazionale diventa O(N) (prima era $O(N^2)$).

## Separabilità

Consideriamo ad esempio una gaussiana con $\sigma = 0.5$. Il suo kernel bidimensionale è:

$$G_{xy}=\begin{bmatrix} 0.0117 & 0.0862 & 0.0117 \\ 0.0862 & 0.6366 & 0.0862 \\ 0.0117 & 0.0862 & 0.0117 \end{bmatrix}$$

I kernel monodimensionali sono:

$$g_x=[0.1080 \quad 0.7979 \quad 0.1080],\ g_y=g_x';$$

Per effettuare la convoluzione con l'immagine $I$ i passi sono i seguenti:

$$I_1=I*g_x \ I_2=I_1*g_y \qquad \text{oppure} \qquad I_1=I*g_y \ I_2=I_1*g_x$$

## Kernel Gaussiano e Chiusura

- La convoluzione di due funzioni gaussiane di varianza $\sigma_1^2$, $\sigma_2^2$ è ancora una gaussiana la cui varianza $\sigma_3^2$ è pari alla somma delle varianze $\sigma_1^2$,$\sigma_2^2$. Vale quindi la seguente relazione:

$$\sigma_3^2 = \sigma_1^2+\sigma_2^2$$

- Tale proprietà risulta molto utile nella costruzione dello scale-space. Ad ogni passo si sfruttano le convoluzioni precedentemente effettuate.

## Kernel Gaussiano e Chiusura

Supponiamo di voler trovare l'immagine prodotta dalla convoluzione di un immagine $I$ con un kernel gaussiano con $\sigma = \sqrt{2}$ e di aver effettuato in precedenza la convoluzione con un kernel gaussiano con $\sigma$=1.

Sia $I_3$ l'immagine che vogliamo ricavare ed $I_1$ quella già convoluta a nostra disposizione.

Possiamo utilizzare l'immagine $I_1$ per produrre $I_3$?
Quale valore di $\sigma$ dobbiamo utilizzare?

## Kernel Gaussiano e Chiusura

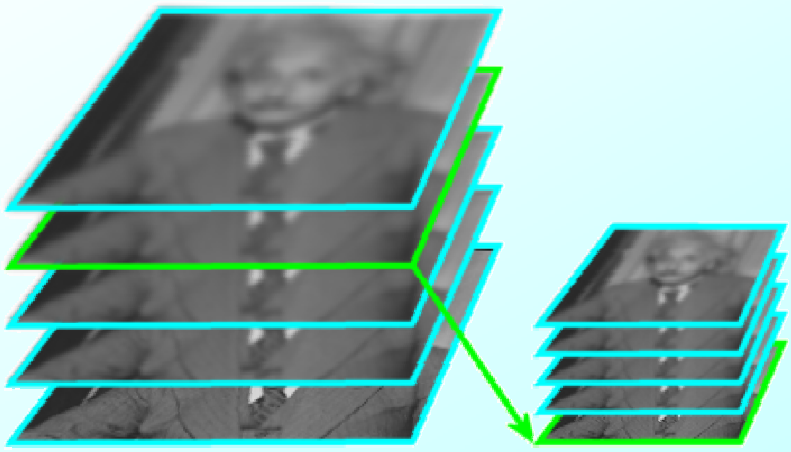
- Sfruttando la proprietà delle convoluzioni in cascata ($\sigma_3^2 = \sigma_1^2 + \sigma_2^2$) e ponendo $\sigma_3^2 = 2$ e $\sigma_1^2 = 1$ ricaviamo $\sigma_2^2 = 1$.

Questa proprietà ci ha permesso di risparmiare varie operazioni. La convoluzione con un kernel $\sigma = \sqrt{2}$ comporta 11 prodotti (per pixel) mentre quella con $\sigma$ pari 1 solo 7 (valutazione analoga può essere fatta per le somme).

## Analisi Scale-Space

- Per velocizzare la costruzione dello *scale-space* l'immagine viene sottocampionata ad intervalli regolari e nuovamente filtrata, dando origine ad insiemi di immagini detti **ottave**.

  La prima immagine di ogni ottava ha associato un valore di $\sigma$ pari al doppio rispetto alla corrispondente immagine nell'ottava precedente.

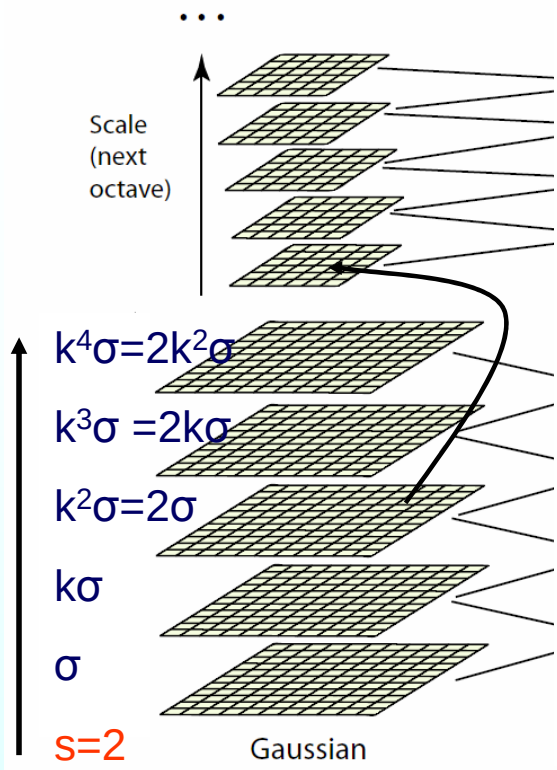


# Analisi Scale-Space

Costruzione dello scale-space:

- L'immagine iniziale viene convoluta con varie gaussiane producendo immagini separate da un fattore ***k*** nello *space-scale*.
- Ogni ottava viene divisa in un numero intero ***s*** di intervalli tali che $k=2^{1/s}$.
- Dopo aver processato un'intera ottava, l'immagine che ha $\sigma = 2\sigma_0$ ($\sigma_0$ deviazione standard relativa alla prima immagine dell'ottava) viene sottocampionata di un fattore 2.
- In ogni ottava occorre calcolare **s**+3 immagini per permettere una corretta estrazione degli estremi locali.

# Analisi Scale-Space




Nell'esempio ogni ottava viene divisa in s=2 intervalli.

Si calcolano s+3=5 immagini con valori di sigma crescenti di un fattore $k=2^{1/s}=2^{1/2}$

Ovviamente $k^s=2$ per cui il valore di sigma iniziale viene raddoppiato ed è proprio questa l'immagine da sottocampionare per l'ottava successiva.

## Analisi Scale-Space

Per trovare dei punti interessanti nello scale-space viene utilizzata la funzione **DoG** (**Difference of Gaussian**):

$$D(x,y,\sigma)=(G(x,y,k\sigma)-G(x,y,\sigma))*I(x,y)=L(x,y,k\sigma)-L(x,y,\sigma)$$

$$L(x,y,\sigma)=I(x,y)*G(x,y,\sigma)$$

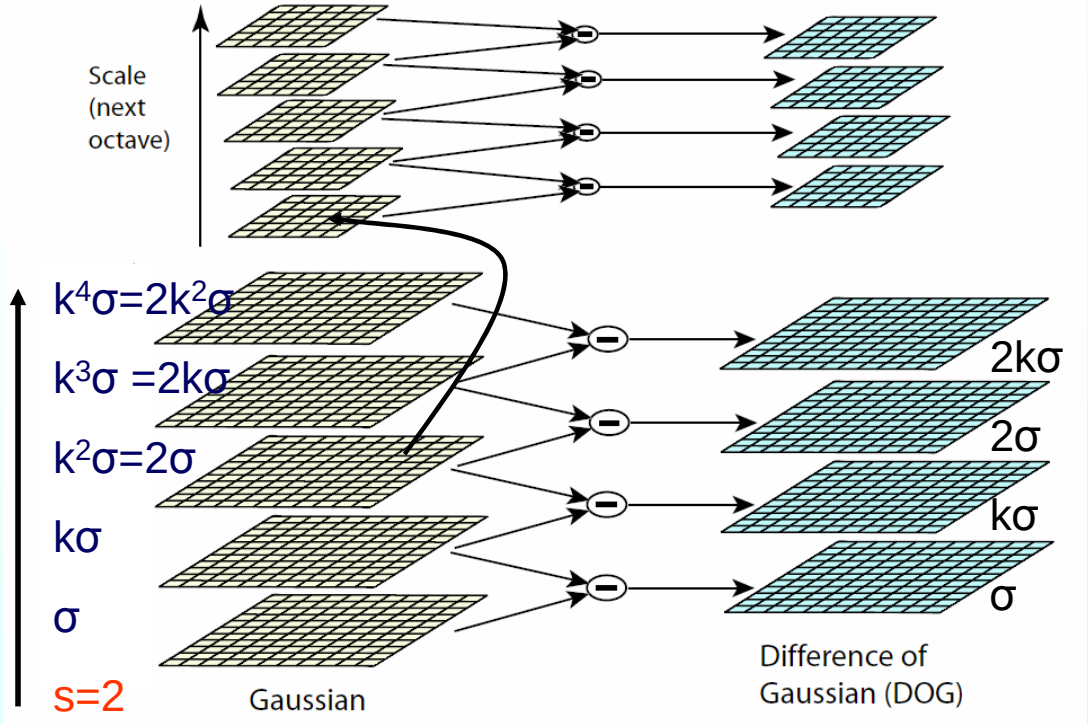
## Analisi Scale-Space

- La funzione **DoG** è molto veloce da calcolare a partire dallo scale-space (basta effettuare la differenza tra immagini convolute vicine tra loro).

- Inoltre è una buona approssimazione della funzione LoG (Laplaciano di Gaussiana) normalizzata in scala. Attraverso accurati confronti si è provato che gli estremi locali del LoG producono delle feature più stabili rispetto ad altre funzioni quali gradiente, Hessiana ed Harris corner function.

# Analisi Scale-Space

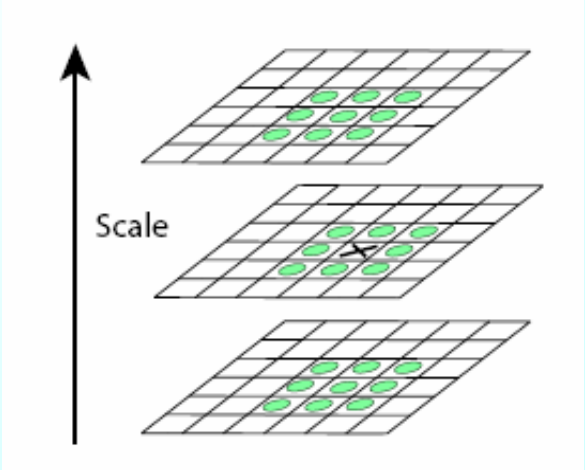
Creazione delle **DoG** a partire dallo scale-space.




# Analisi Scale-Space

Gli estremi locali della funzione D(x,y,σ) costituiscono i punti di interesse cercati. Per rilevarli ogni campione viene confrontato con i suoi 8 vicini nell'immagine corrente e con i nove vicini della **scala superiore ed inferiore**.

Un punto viene considerato massimo o minimo locale se è più grande o più piccolo di tutti i suoi vicini.



# Keypoint Localization

Viene testata la stabilità dei punti chiave, analizzando il **contrasto**, valutando la presenza di bordi o linee e interpolandone la posizione per aumentare la precisione.

Viene adattata una funzione quadratica 3D ai punti locali campionati per determinare la posizione interpolata dell'estremo locale. Gli esperimenti mostrano che si ha un miglioramento in termini di matching e di stabilità.

# Keypoint Localization

Sviluppiamo in serie di Taylor (arrestandoci al secondo ordine) la funzione D(x,y,σ):

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x$$

Con D e le derivate valutate nell'estremo locale. L'estremo di tale funzione, calcolato mettendo a zero le derivate è:

$$x = -\frac{\partial^2 D}{\partial x^2}^{-1} \frac{\partial D}{\partial x}$$

Se lo shift trovato è maggiore di 0.5 nelle varie dimensioni l'estremo si trova vicino ad un altro punto. In questo caso, il punto viene cambiato e l'interpolazione viene effettuata rispetto al nuovo campione.

# Keypoint Localization

Il valore della funzione D(x,y,σ) valutata nell'estremo risulta utile nell'eliminare estremi instabili con basso constrasto:

$$D(\hat{x}) = D + \frac{1}{2}\frac{\partial D^T}{\partial x}\hat{x}$$

Lowe [5] nella sua implementazione scarta tutti gli estremi con

$$\left|D(\hat{x}) < 0.03\right|$$

Da notare che tale valore è relativo ad immagini a valori in [0,1].

# Keypoint Localization

L'eliminazione degli estremi a basso contrasto non è sufficiente a garantire la stabilità. La DoG ha infatti forti risposte anche in presenza di edge, anche se in tal modo la posizione non è ben determinata.

Per risolvere il problema basta considerare l'Hessiana (i suoi autovalori sono proporzionali alle curvature principali di D):

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

# Keypoint Localization

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Sia $\alpha$ l'autovalore più grande e $\beta$ quello più piccolo.

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta$$

$$Det(H) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta$$

Sia r il rapporto tra l'autovalore più grande ed il più piccolo ($\alpha$=r$\beta$).

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}$$

# Keypoint Localization

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}$$

La quantità $(r+1)^2/r$ è minima quando i due autovalori hanno lo stesso valore e cresce al crescere di r. Quindi per imporre che il rapporto tra curvature principali (r) sia sotto una certa soglia occorre che:

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}$$

Occore notare che questo procedimento ci ha evitato il calcolo degli autovalori della matrice H.

# Keypoint Localization

## ESEMPIO

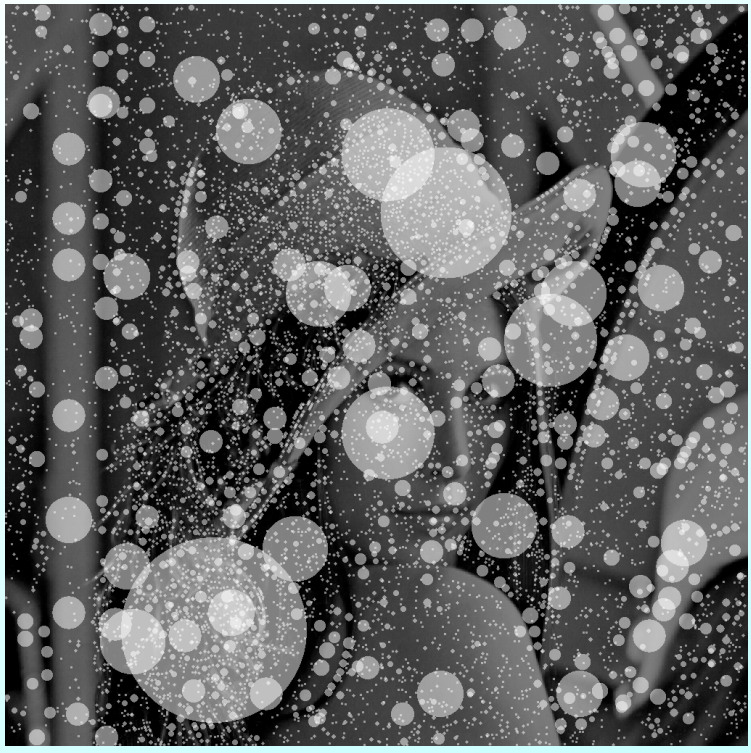
**1.Immagine**

# Keypoint Localization

## ESEMPIO

1.Immagine

**2.Estremi locali della DoG** (7951)



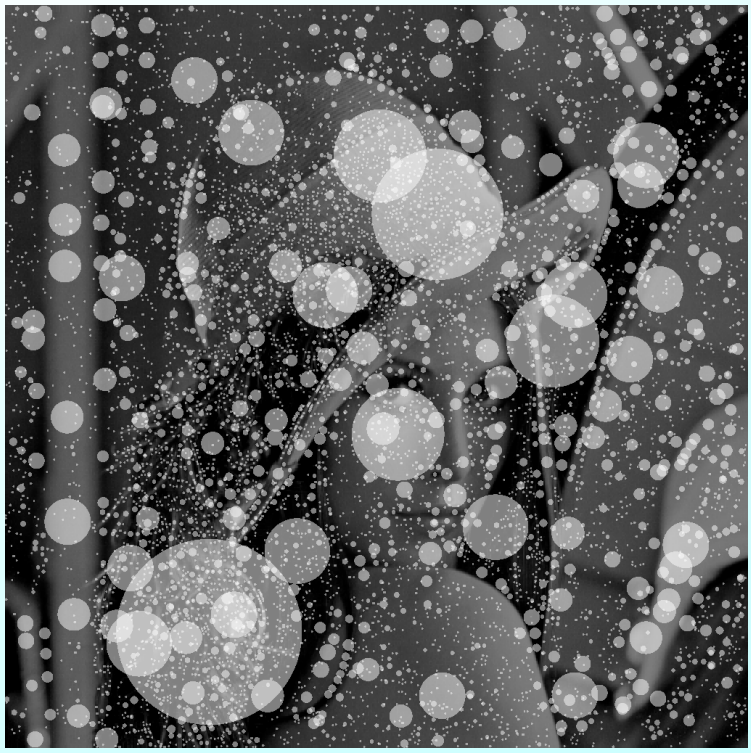
# Keypoint Localization

## ESEMPIO

1.Immagine

2.Estremi locali della DoG (7951)

**3.Keypoint localizzati**



# Keypoint Localization

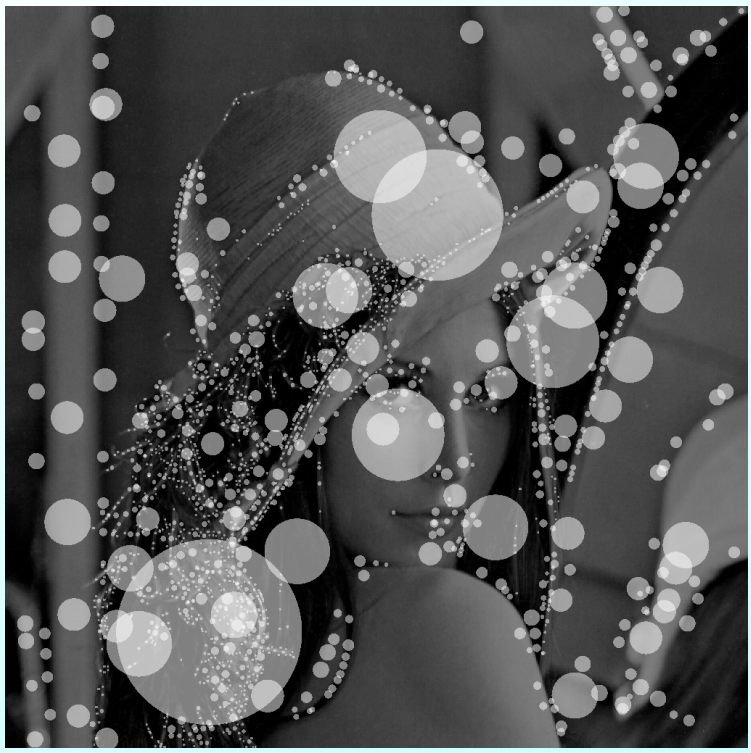
## ESEMPIO

1.Immagine

2.Estremi locali della DoG (7951)

3.Keypoint localizzati

**4.Keypoint rimasti dopo il filtraggio basato sul contrasto** (1962)



# Keypoint Localization

## ESEMPIO

1.Immagine

2.Estremi locali della DoG (7951)

3.Keypoint localizzati

4.Keypoint rimasti dopo il filtraggio basato sul contrasto (1962)

**5.Keypoint rimasti dopo il filtraggio basato sul rapporto tra le curvature principali** (1309)

# Orientation Assignment

Ad ogni punto chiave viene associata una direzione basandosi sulle proprietà locali dell'immagine. La scala del keypoint viene utilizzata per scegliere l'immagine smussata con la scala più vicina.

Per ogni campione dell'immagine L(x,y) vengono calcolati magnitudo ed orientazione del gradiente:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

## Orientation Assignment

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

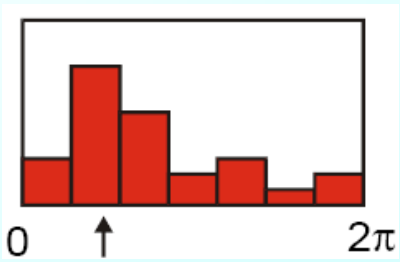
$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

A partire dalle orientazioni dei gradienti dei punti in un intorno del keypoint viene costruito un istogramma delle orientazioni.

L'istogramma ha 36 bin e copre l'intero range delle orientazioni (360°). Ogni campione aggiunto nell'istogramma vien e pesato in base alla magnitudo del suo gradiente e ad una finestra circolare Gaussiana con deviazione standard pari a 1.5 volte la scala corrente.

## Orientation Assignment

Per determinare l'orientazione del punto chiave viene selezionato il valore associato al picco più alto dell'istogramma ed eventuali altri picchi che superino l'80% del valore del picco massimo. Nel caso in cui siano stati selezionati più picchi verranno creati vari keypoint con la stessa posizione, la stessa scala ma differente orientazione.




Per ottenere una maggiore accuratezza, viene effettuato un fitting parabolico considerando i tre punti più vicini al picco selezionato.

# Orientation Assignment

## **ESEMPIO** (continua)

1.Immagine

2.Estremi locali della DoG (7951)

3.Keypoint localizzati

4.Keypoint rimasti dopo il filtraggio basato sul contrasto (1962)

**5.Keypoint rimasti dopo il filtraggio basato sul rapporto tra le curvature principali** (1309)

# Orientation Assignment

## **ESEMPIO** (continua)

1.Immagine

2.Estremi locali della DoG (7951)

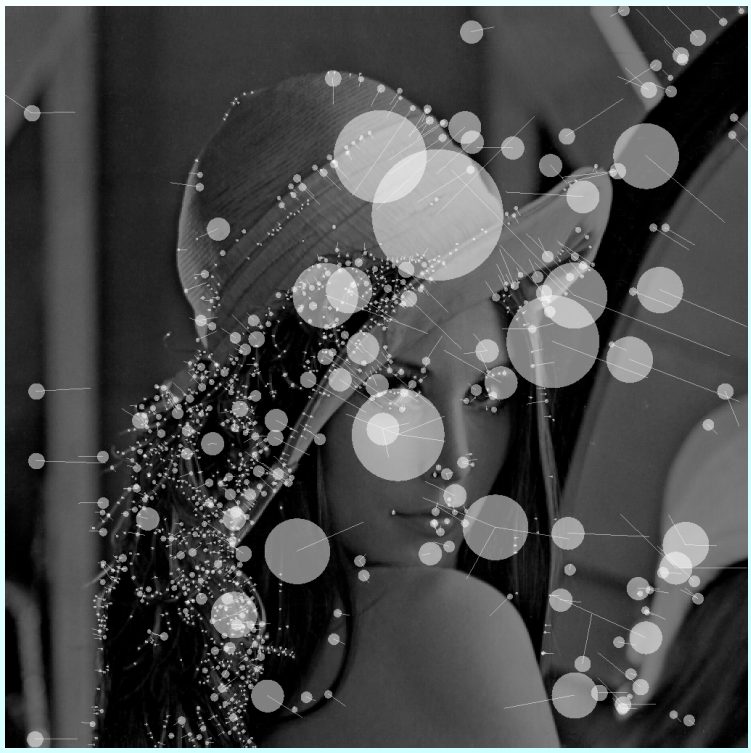
3.Keypoint localizzati

4.Keypoint rimasti dopo il filtraggio basato sul contrasto (1962)

5.Keypoint rimasti dopo il filtraggio basato sul rapporto tra le curvature principali (1309)

**6.Keypoint orientati** (1551)

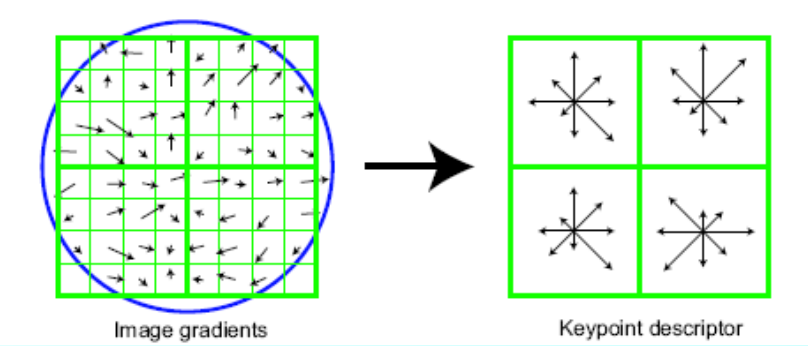
In presenza di più orientazioni principali i keypoint vengono replicati: il loro numero quindi aumenta.



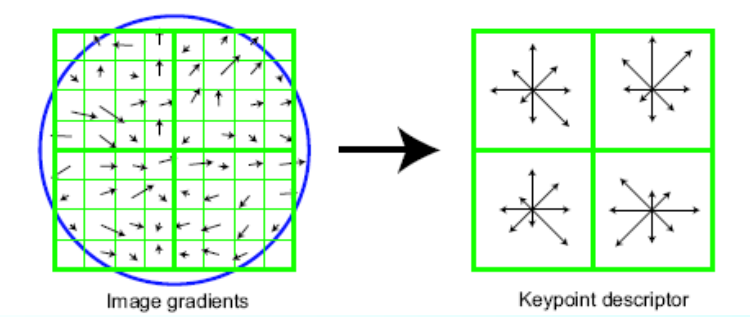
# Keypoint Descriptor

Ad ogni keypoint è stata assegnata una scala, una posizione ed un'orientazione.

Occorre assegnare ai punti chiave anche un descrittore che risulti invariante, il più possibile, ai cambiamenti di illuminazione e punti di vista 3D.




# Keypoint Descriptor

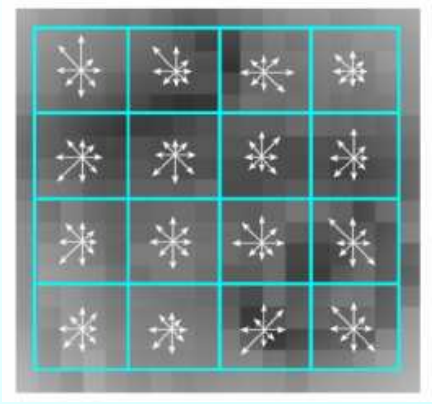



Per il calcolo dei descrittori vengono prima di tutto calcolati magnitudo ed orientazione del gradiente nei punti che si trovano in un intorno del keypoint in esame.

Per ottenere invarianza rotazionale le coordinate del descrittore e le orientazioni del gradiente vengono ruotate relativamente a quelle del punto chiave. Viene inoltre assegnato un peso alle magnitudo in base ad una funzione gaussiana con $\sigma$ pari alla metà della finestra del descrittore.

## Keypoint Descriptor

I campioni vengono accumulati in istogrammi di orientazioni che riassumono il contenuto di regioni. Ogni bin dell'istogramma contiene la somma delle magnitudo dei gradienti vicini alla direzione che rappresenta.



Per evitare cambi bruschi dovuti a semplici shift tra istogrammi o orientazioni, viene utilizzata un'interpolazione trilineare per distribuire il valore di ogni gradiente negli istogrammi adiacenti.

## Keypoint Descriptor

- Il descrittore è formato da un vettore contenente i valori di tutti gli istogrammi delle orientazioni (altezza dei bin). Lowe ha ottenuto i migliori risultati con 4x4 istogrammi con 8 bin ciascuno. Ogni descrittore è dunque un array di 128 (4*4*8) elementi.
- Per rendere il descrittore meno sensibile ai cambi di illuminazione vengono effettuate le seguenti operazioni: il vettore viene normalizzato all'unità di lunghezza (invarianza a cambiamenti affini di illuminazione); i valori presenti nel vettore maggiori di 0.2 vengono sogliati ed il vettore viene nuovamente normalizzato (parziale invarianza a cambi non lineari di illuminazione).

## Keypoint Matching

In molte applicazioni (object recognition, stereo vision, motion analysis), occorre mettere in corrispondenza punti trovati in immagini diverse. Le sift trovano punti “interessanti” nelle varie immagini ed associano loro dei descrittori. A partire dai descrittori è possibile mettere in relazione i punti trovati nelle varie immagini; dato un punto nell’immagine $I_a$ con descrittore $d_{pa}$ il punto corispondente nell’immagine $I_b$ è quello che ha il descrittore più vicino (es. distanza euclidea) a $d_{pa}$.

Può accadere che non tutti i punti presenti in un’immagine hanno un corrispondente nell’altra immagine (occlusioni, nuovi oggetti nella scena,…). Occorre dunque avere un criterio che permetta di capire se un punto ha un buon match.

**L’utilizzo di una soglia fissa sulla distanza tra descrittori non si è dimostrata una buona scelta.**

## Keypoint Matching

Lowe [5] utilizza come misura della bontà del matching la seguente relazione:

$$r = \frac{d1}{d2}$$

*d1* distanza euclidea tra il descrittore $d_{pa}$ ed il descrittore con minima distanza ($d_{pb1}$ relativo ai punti dell’immagine $I_b$) da $d_{pa}$

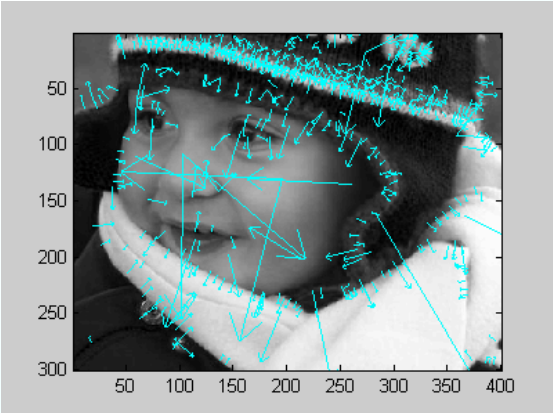
*d2* distanza euclidea tra il descrittore dpa ed il descrittore con minima distanza da $d_{pa}$ dopo $d_{pb1}$

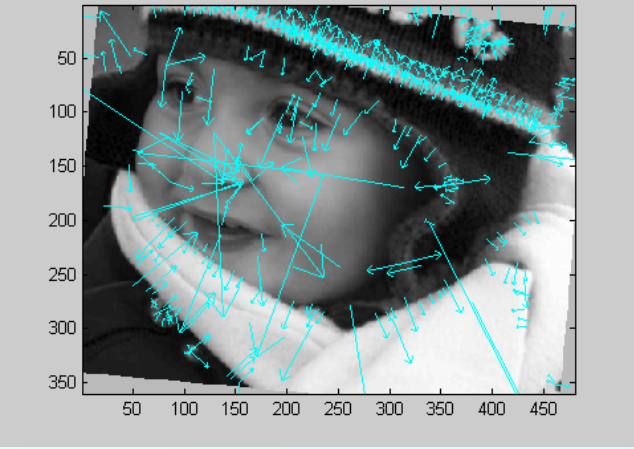
Nel caso di matching corretto r assumerà valori non elevati. In particolare [5] se *r* è inferiore a 0.6 il matching è corretto, altrimenti il punto con descrittore $d_{pa}$ non deve essere preso in considerazione.

# Sift (esempio)


Anche in presenza di rotazioni, occlusioni, cambiamenti di scala ed illuminazione il treno e la rana vengono riconosciuti.

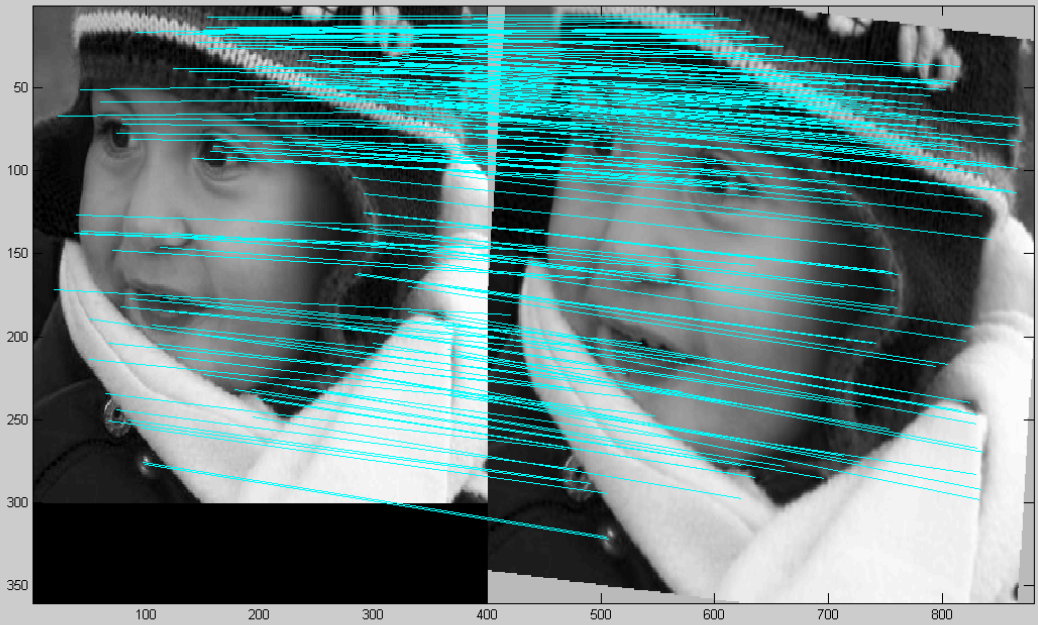
# SIFT (Esempio)




La seconda immagine è stata ruotata, ingrandita e leggermente modificata nel contrasto e nella luminosità globale

# Sift (Esempio)




# References


- [1] Witkin, A.P. 1983. Scale-space filtering. In International Joint Conference on Artificial Intelligence,Karlsruhe, Germany, pp. 1019-1022.
- [2] Koenderink, J.J. 1984. The structure of images. Biological Cybernetics, 50:363-396.
- [3] Lindeberg, T. 1994. Scale-space theory: A basic tool for analysing structures at different scales. Journal of Applied Statistics, 21(2):224-270.
- [4] Mikolajczyk, K. 2002. Detection of local features invariant to affine transformations, Ph.D. thesis, Institut National Polytechnique de Grenoble, France.
- [5] Lowe, D. G. 2004. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision Vol. 60(2), p.9.

# References


- Harris, C. and Stephens, M. 1988. A combined corner and edge detector. In Fourth Alvey Vision Conference, Manchester, UK, pp. 147-151.
- Burt P.J., Adelson E.H. (1983) “The Laplacian Pyramid as a Compact Image Code”, *IEEE Transactions on Communications*, Vol. COM-9, No. 4, pp532-540.