

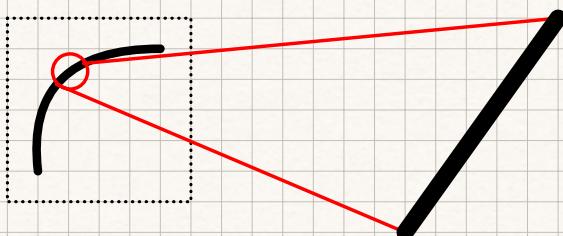
FEATURES DETECTION

Dato un'immagine vogliamo processarla per estrarre **locations** o meglio **keypoints** ovvero quelli che sono più probabili da trovare in altre immagini.

I Keypoints dovrebbero essere **distinti**, ovvero dobbiamo essere in grado di distinguere per differente posizione nell'immagine e **invarianti**: la stessa posizione può essere trovata in altre immagini anche se esse hanno trasformazioni geometriche e/o fotometriche.

Possiamo usare gli **edges** come Keypoint? Gli edges sono contengono abbastanza informazione. Solitamente una immagine contiene troppi edges, anche non rilevanti. **Gli edge non sono abbastanza unici.**

Possiamo usare gli **angoli** come Keypoints? Per oggetti molto semplici potrebbero bastare, ma per oggetti molto complessi cerca una volta **non sono unici abbastanza unici**. Inoltre gli angoli non sono scale invarianti.



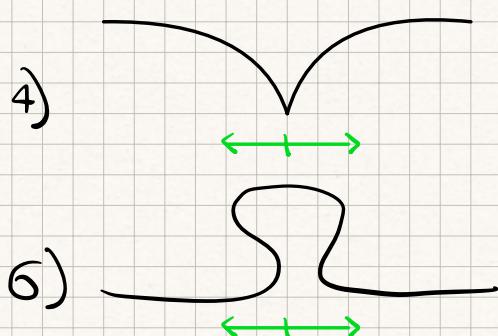
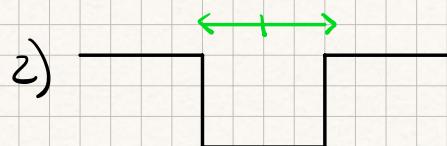
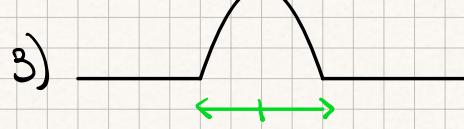
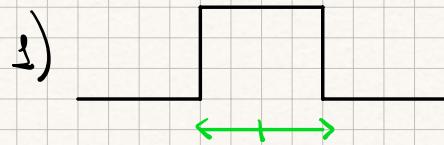
Questo qui è un angolo, ma se zoominiamo non lo è più per sue particolarità.

Altra idea: AREA

Ci concentreremo in aree delimitate della nostra immagine, dentro una ben definita localizzazione.

Questa area è detta **Blob**

Esempio di blob:



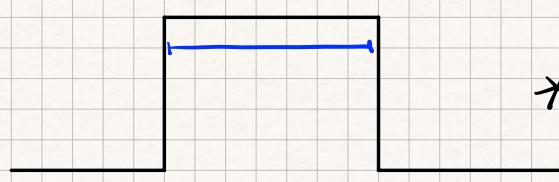
AREA BEN DEFINITA

LOCALIZZAZIONE BEN DEFINITA

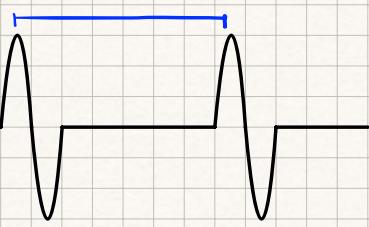
Nella sua forma più semplice un blob è formato da due **step signal**, ognuno dei quali corrisponde con un edge, ma ognuno di questi edge deve essere accoppiato l'uno con l'altro.

Vediamo cosa succede a varie blobs con il LOG.

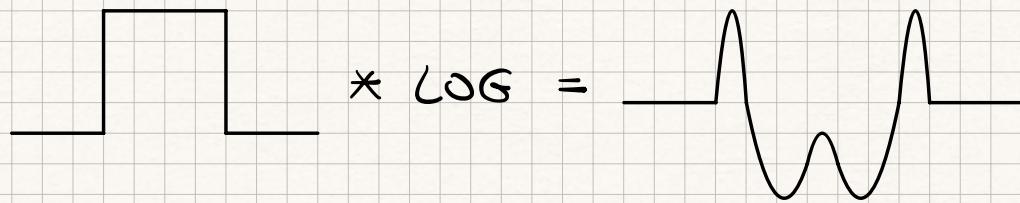
Per semplicità presupponiamo $G=1$.



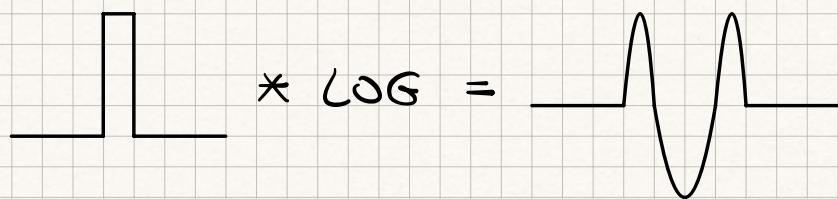
* LOG =



Proviamo ora con un blob più piccolo



Stiamo variando la larghezza del blob con un LOG con $\sigma = \text{cost}$. Più stringiamo più il contributo degli edges si sovrappone.

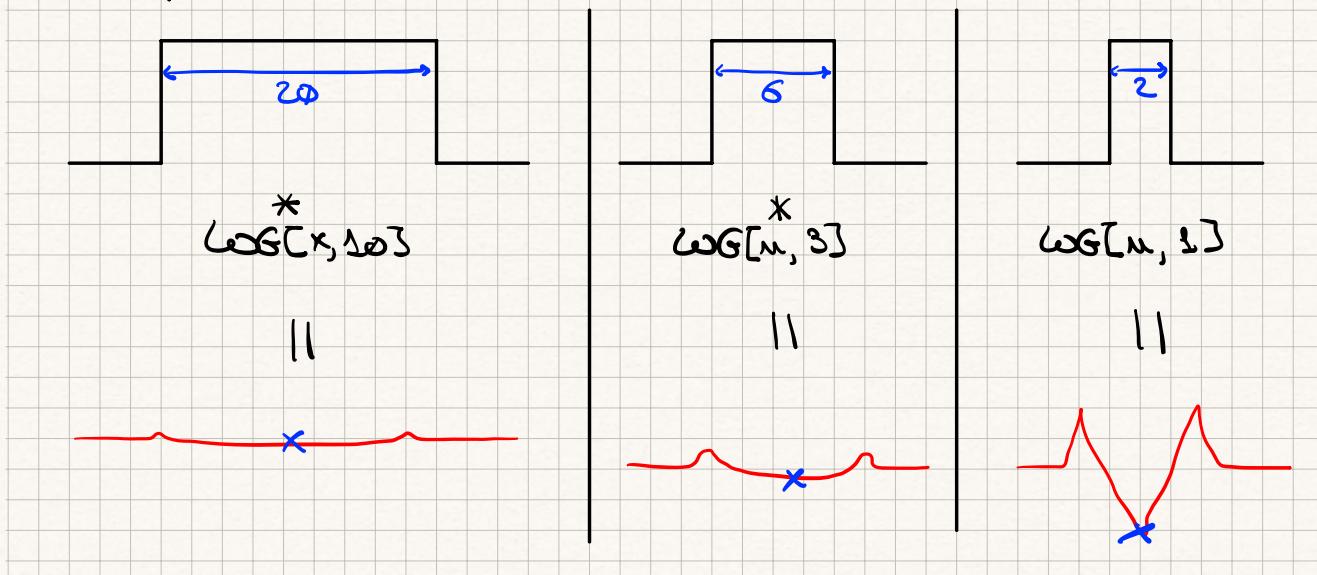


Vediamo che succede con $\sigma = \frac{\text{blob size}}{2}$.

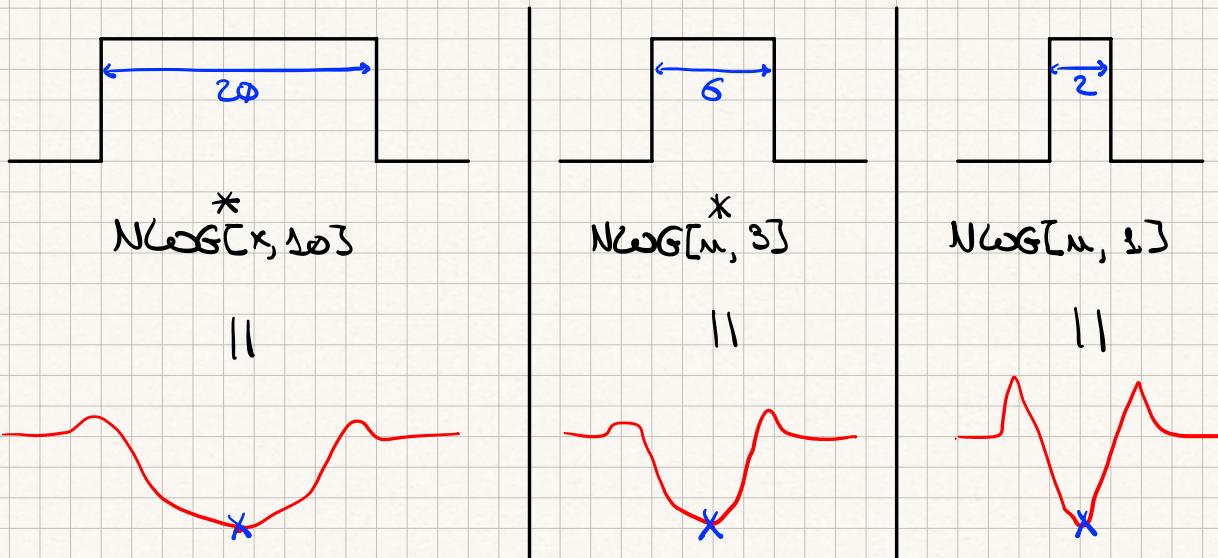
Ma prima introduciamo il **NORMALIZED LOG**:

$$N\text{LOG}[x, y, \sigma] = \sigma^2 \text{LOG}[x, y, \sigma]$$

Ora facciamo la convoluzione



δ chiaro che c'è un problema di scale, poiché i minimi ci sono ma sono molto allontanati per grandi σ .



Dobbiamo capire ora come settare il miglior $\sigma = \sigma^*$ per cui il NLLG convoluto con l'immagine raggiunge il suo valore minore. σ^* è detto **characteristic scale**.

Possiamo ora trovare la posizione x^*, y^* e σ^* di un blob semplicemente risolvendo:

$$x^*, y^*, \sigma^* = \underset{x, y, \sigma}{\operatorname{argmin}} (NLLG[x, y, \sigma] * f[x, y])$$

Come implementare un algoritmo per risolvere questo problema? Dobbiamo calcolare convolutioni per scale differenti: $\sigma_0, \sigma_1, \sigma_2, \sigma_3, \dots$ (una sorta di grid search per σ). Iniziamo da un σ_0 iniziale (**base-scale** σ_0) per poi continuare con $\sigma_i = s^i \sigma_0$ con $s > 1$.

σ è una costante.

Praticamente, cercare σ , vuol dire cercare la scala migliore per riuscire a vedere bene i Key Point. Come se prendessi l'immagine e la "avvicinassi" per vedere meglio.

DOG

In pratica l'operatore NLG non è mai esplicitamente usato, ma si usa una approssimazione più veloce da calcolare.

"Ho appena letto che questa equazione non la riuscirà."

$$\text{LOG}[x, y, \sigma] = \frac{1}{\sigma} g_\sigma[x, y, \sigma]$$

\Downarrow

$$g_\sigma[x, y, \sigma] = \partial \text{LOG}[x, y, \sigma] \quad *$$

Spansione di Taylor del primo ordine:

$$g[x, y, \sigma + \Delta\sigma] \approx g[x, y, \sigma] + \Delta\sigma g_\sigma[x, y, \sigma] \quad **$$

Utilizzando * e ** otteniamo:

$$\underbrace{g[x, y, \sigma + \Delta\sigma] - g[x, y, \sigma]}_{\text{Difference of Gaussian}} = (\Delta\sigma) \sigma \text{LOG}[x, y, \sigma]$$

DOG operator

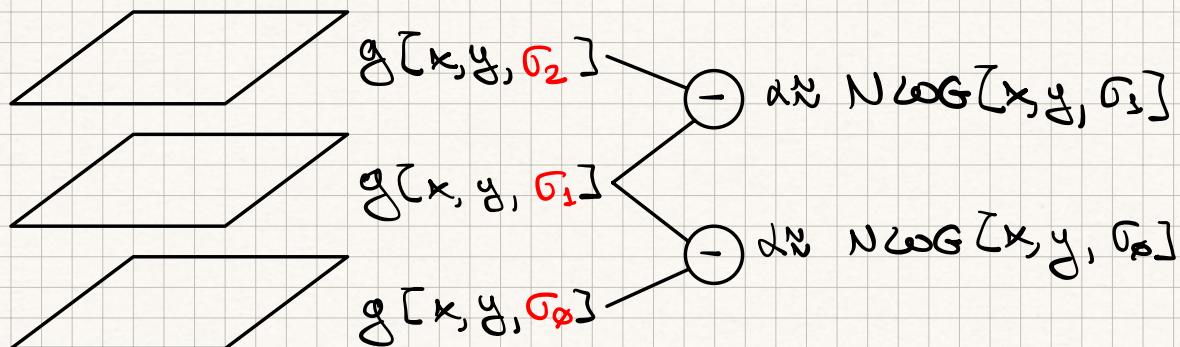
Se sostituiamo $\rho = (\sigma + \Delta\sigma)/\sigma$ otteniamo:

$$g[x, y, \rho\sigma] - g[x, y, \sigma] \approx (\rho-1) \sigma^2 \text{NLOG}[x, y, \sigma] =$$

$$= (\rho-1) \text{NLOG}[x, y, \sigma]$$

Quindi la differenza è che con $g[x, y, \rho\sigma] - g[x, y, \sigma]$ abbiamo una convoluzione con ben due gaussiane mentre dell'altra parte $(\rho-1) \text{NLOG}[x, y, \sigma]$ con solo NLOG.

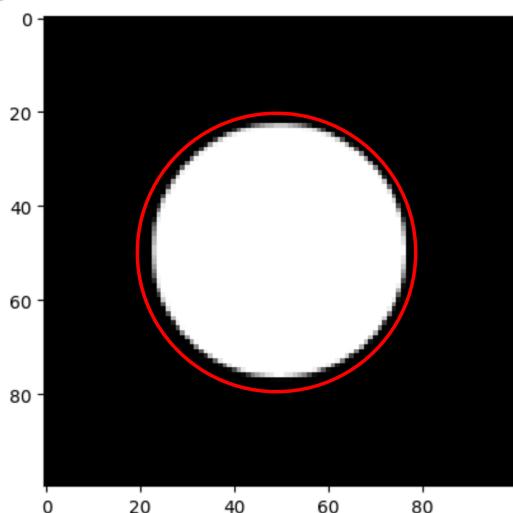
Quindi il problema di prima può essere risolto con le convoluzioni di $f[x, y]$ con il filtro Gaussiano con diverse scale $\sigma_1, \sigma_2, \dots$ e poi calcolando le differenze con due risultati impilati:



Il discorso è che comunque abbiamo bisogno di calcolare le convoluzioni con i filtri gaussiani nella fase di Feature Descriptor, quindi possiamo risparmiare le stesse, invece di calcolare nuove convoluzioni!

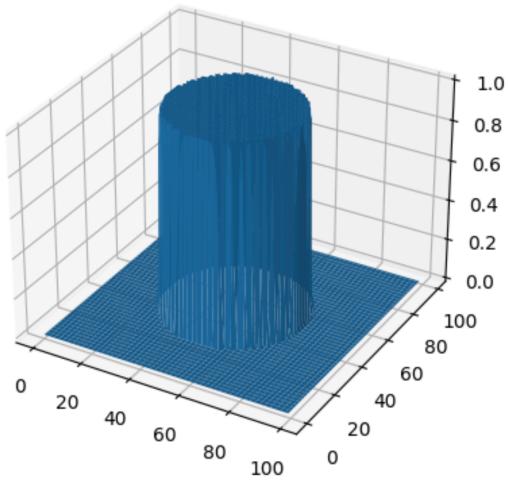
Esempio Blob Detection

1)



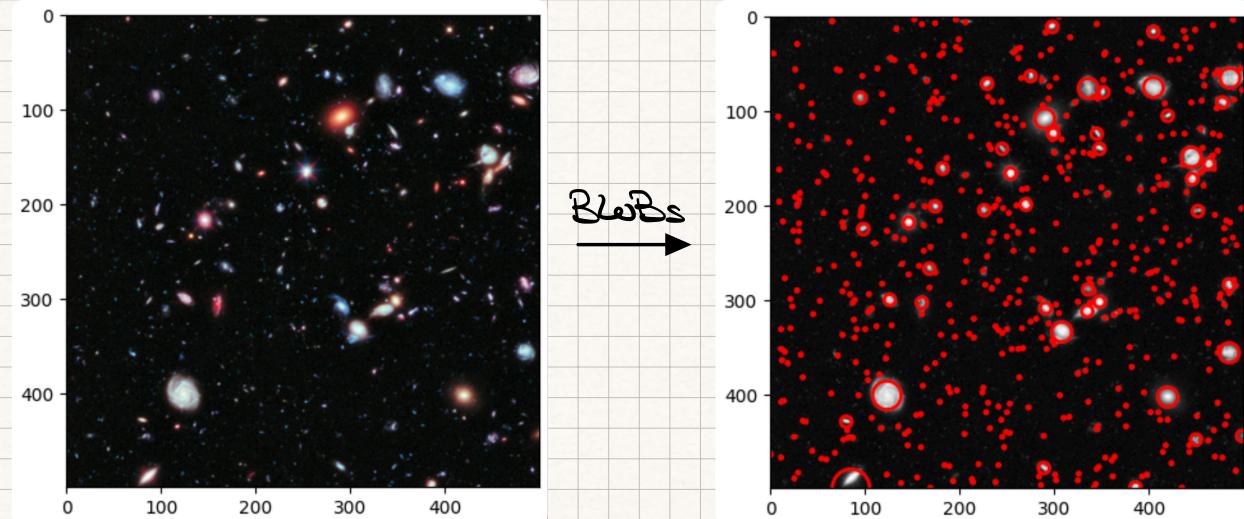
Qui abbiamo un unico blob, che fa riferimento al cerchio bianco.

Guardando la funzione $f[x, y]$ della scala dei grigi della nostra immagine abbiamo:



Da qui possiamo intuire meglio l'area.

2) Proviamo con un'immagine più complessa:



Per quanto riguarda $f[x, y]$ nella scala dei grigi:

