

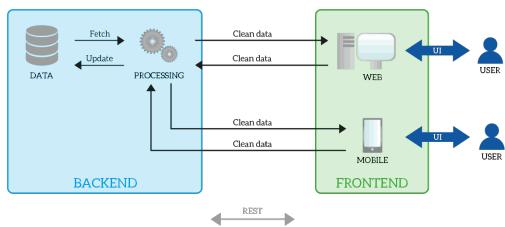


Università
di Catania

Relazione: Laboratorio multimedia (LM-18 Informatica) – A.A 2020/2021
A cura di Leocata Danilo – 1000022576

Introduzione

Sono stati presi in esame algoritmi di image processing, analizzandone i casi d'uso. Gli algoritmi sono stati poi scritti in codice python, con materiale reperibile su [GitHub](#) commentato opportunamente per dimostrare i motivi della scelta implementativa utilizzando le librerie opencv, numpy ed imageio. Oltre ad essere stati testati tramite eseguibile, sono stati integrati in una web-application realizzata con front-end in svelte (un framework che utilizza javascript ed html) e back-end in flask. Le istruzioni per eseguire il progetto si trovano all'interno del README.md nella [repository del progetto](#). È stata inoltre implementata una funzione PSNR per valutarne il risultato dopo l'applicazione dell'algoritmo rispetto all'immagine originale.



È stata realizzata una semplice interfaccia dalla quale sarà possibile caricare una generica immagine ed applicare uno dei filtri implementati tramite chiamate REST. Nella fase di upload l'immagine sarà codificata in base64, non è stato previsto un database ma per semplicità verrà scritta su disco, dopo sarà possibile applicare vari filtri di image processing. Le immagini presentate all'interno della relazione saranno tutte presenti nella cartella dedicata all'interno della repository in quanto l'import dell'immagine all'interno del documento e la conseguente dimensione ne compromette la visualizzazione.

Filtro di media aritmetica

$$\hat{f}(x,y) = \frac{1}{mn} \sum_{(s,t) \in S_{xy}} g(s,t)$$

Dove m ed n sono le dimensioni della finestra, g è l'immagine corrotta nell'area definita da S_{xy} che è l'insieme delle coordinate della finestra. Il valore di \hat{f} è il valore dell'immagine restaurata nel punto $g(x,y)$. Attenua le variazioni locali di un'immagine e ne riduce di conseguenza il rumore.

Algoritmo implementato



Filtro mediano

È un tipo di filtro spaziale la cui risposta si basa sull'ordinamento dei valori dei pixel contenuti nell'area dell'immagine inglobata dal filtro. La posizione relativa determina la risposta del filtro. Il filtro mediano è il più noto tra quello delle statistiche d'ordine che sostituisce il valore di un pixel con il mediano dei livelli di intensità nel suo intorno, matematicamente:

$$f(x,y) = \text{mediano}\{g(s,t)\}$$

Con $(s,t) \in S_{xy}$

Dove il valore del pixel in (x,y) è incluso nel calcolo della mediana. I filtri mediani sono abbastanza noti perché, per alcuni tipi di rumori casuali, hanno eccellenti capacità di riduzione del rumore, sfocando molto meno rispetto a filtri di smoothing di dimensioni simili. Da risultati eccellenti in presenza di rumore a impulsi (bipolare e unipolare). Incrementando l'ordine della finestra si andrà ad aumentare l'ordine del mediano. Ripetuti passaggi di questo filtro sfocano l'immagine ulteriormente, quindi sarebbe meglio limitare il numero di applicazioni successive

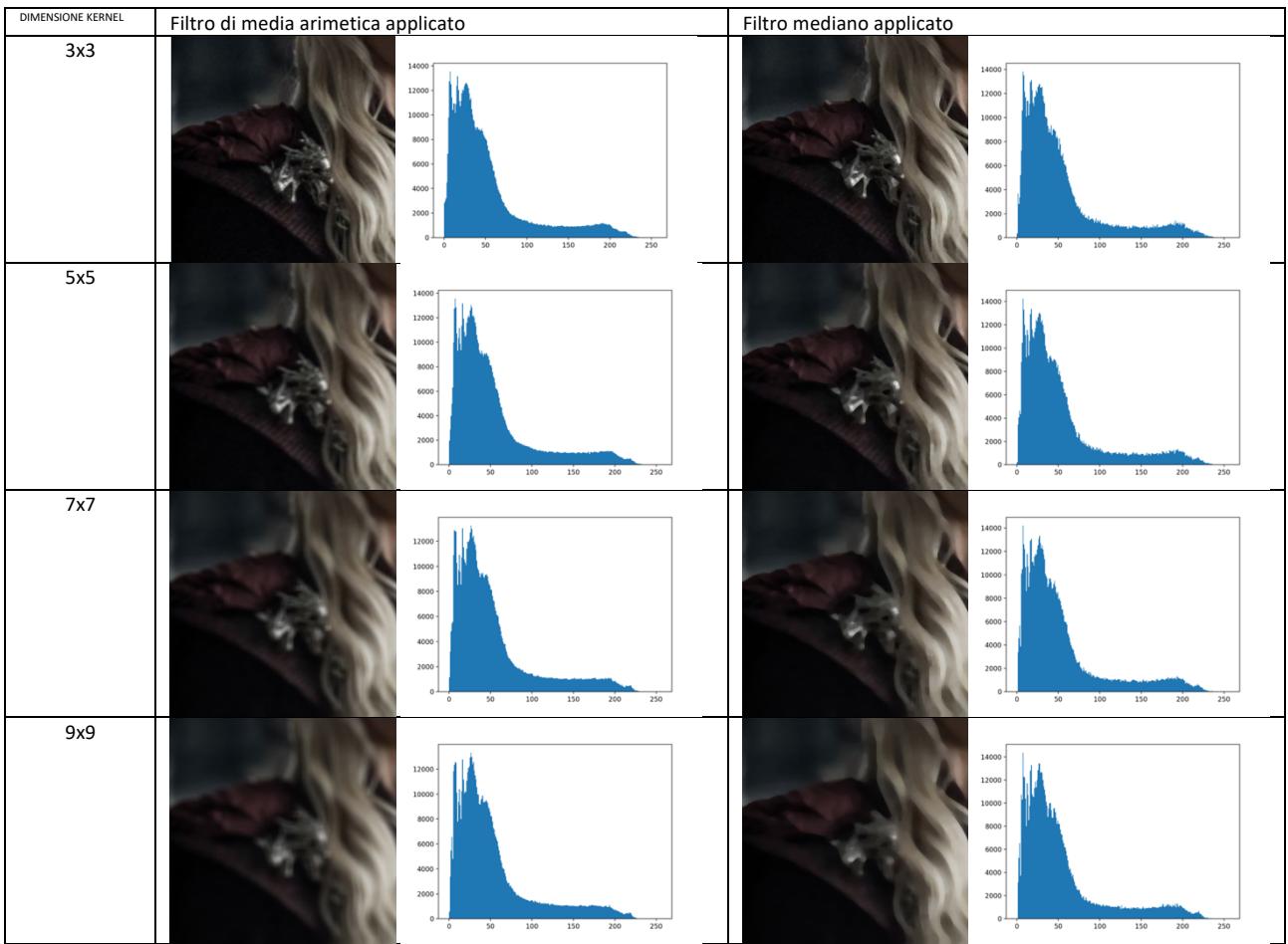
Algoritmo implementato



Esempi di applicazione

Consideriamo di seguito un dettaglio dell'immagine di test per notarne le differenze all'aumentare della finestra. Il psnr calcolato fa riferimento all'intera immagine di input.

Immagine originale	Valori PSNR filtro di media aritmetica	Valori PSNR filtro mediano
3x3	34.45654233586586	35.42571604637124
5x5	32.451949432323744	32.89428905228828
7x7	31.641010733959668	31.949368778731028
9x9	31.127911423634067	31.4052257114982



Aumentando la dimensione della finestra il filtro di media tende a sfocare solamente i valori dell'istogramma, sfocando l'immagine, mentre il mediano tende ad allargare i contorni e formare delle chiazze uniforme di colore. Notiamo anche la formazione di una banda nera a destra, dovuto alla scelta implementativa, in entrambi i casi.

Un esempio di applicazione su immagine rumorosa

Consideriamo di seguito una generica immagine con rumore sale e pepe:



Immagine originale

Applichiamo di seguito filtro mediano e di media con rispettivamente kernel 3x3 e 5x5:

Mediano	33.20249007580863	31.31734579773987
Media aritmetica	29.782979821760893	30.99426617900852

	DIMENSIONE DELLA FINESTRA: 3x3	DIMENSIONE DELLA FINESTRA: 5x5
Mediano		
Media artimetica		

Risulta dunque inutile l'applicazione del filtro di media aritmetica in questo caso. Nonostante il mediano con finestra 5x5 causi una perdita quasi totale di una perdita di contorni a livello visivo, risulta migliore del filtro di media applicato con una finestra minore.

Filtro bilaterale¹

È un filtro non lineare che tende a preservare i contorni e ridurre il rumore con filtri per le immagini. Sostituisce la densità di ogni pixel con una media pesata dell'intensità dei pixel vicini. I pesi possono essere basati secondo una distribuzione gaussiana, che dipendono non solo dalla distanza euclidea dei pixel (in matematica, una distanza tra due punti, in particolare è una misura della lunghezza del segmento avente per estremi i due punti) ma anche dalle differenze radiometriche come la differenza di range, l'intensità di colore, distanza di profondità. Queste operazioni tendono a preservare i bordi

Il filtro è definito dalla seguente formula:

$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|),$$

Dove il termine di normalizzazione W_p è dato dalla formula:

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

Dove

- I^{filtered} è l'immagine filtrata,
- I è l'immagine di input da filtrare
- x rappresenta le coordinate del pixel corrente che deve essere filtrato
- Ω è la finestra centrata in x , quindi $x_i \in \Omega$ in un altro pixel
- f_r è il range kernel per le differenze di smoothing in intensità (questa funzione può essere una gaussiana)
- g_s è il kernel spaziale (o dominio) per la differenza di smoothing in coordinate (questa funzione può essere una gaussiana)

Il peso W_p è assegnato usando la chiusura spaziale (usando il kernel spaziale g_s) e le differenze di intensità (usando il range kernel f_r). Consideriamo un pixel (i, j) per il quale vogliamo eliminare il rumore usando i suoi pixel vicini e uno dei pixel è in (k, l) . Quindi, assumendo che il range e i kernel spaziali siano gaussiani, il peso assegnato al pixel (k, l) per eliminare il rumore il pixel (i, j) è dato dalla formula:

$$w(i, j, k, l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2} - \frac{\|I(i, j) - I(k, l)\|^2}{2\sigma_r^2}\right),$$

Dove σ_d e σ_r sono parametri di smoothing e $I(i, j)$ ed $I(k, l)$ sono le intensità dei pixel $I(i, j)$ e $I(k, l)$. Una volta aver calcolato i pesi, applichiamo la normalizzazione:

$$I_D(i, j) = \frac{\sum_{k,l} I(k, l) w(i, j, k, l)}{\sum_{k,l} w(i, j, k, l)},$$

dove I_d è il l'intensità prima di rumore di (i, j)

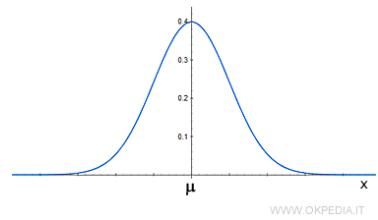
- All'aumentare di σ_r (parametro di range), il filtro bilaterale assomiglia gradualmente alla convoluzione gaussiana perché il range gaussiano si allarga e si appiattisce, che significa che diventa quasi costante nell'intervallo di intensità dell'immagine.
- All'aumentare del parametro spaziale σ_d , le caratteristiche più "importanti"/dettagliate subiscono una sorta di "levigazione"

Algoritmo implementato

Limitazioni

Il filtro bilaterale può introdurre vari tipi di artefatti, quali: l'effetto staircase che tende a formare altopiani di intensità, unificando una zona e soprattutto gradient reversal che comporta l'introduzione di falsi bordi nell'immagine. La causa è data dal fatto che quando un pixel hanno alcuni pixel simili attorno, la media pesata gaussiana è instabile. In questo caso, il risultato probabilmente esibisce "profili indesiderati" attorno ai bordi, usualmente osservati in detail enhancement o compressione HDR.

Esistono diverse estensioni al filtro che si occupano di questi artefatti, come il filtro bilaterale in scala che utilizza un'immagine ridotta per calcolare i pesi. Anche filtri alternativi, come il filtro guidato, sono stati proposti come un'alternativa efficiente senza queste limitazioni.



I esempio di distribuzione gaussiana (o normale)

¹ https://en.wikipedia.org/wiki/Bilateral_filter

Output bilateral filter

PSNR	$\sigma_r = 0.1$	$\sigma_r = 0.2$	$\sigma_r = 0.4$
$\sigma_s = 2$	36.0480067641424	34.37136279370226	33.95546897875178
$\sigma_s = 4$	34.647387894308956	32.941345545252844	32.438785292990424
$\sigma_s = 8$	34.647387894308956	32.941345545252844	32.438785292990424



Guided image filtering²

Il guided filter “computa” un’immagine di output considerando il contenuto dell’immagine guida che può essere l’immagine di input stessa o un’altra. Può essere utilizzato come un operatore che preserva i bordi, come il bilateral filter, di cui è pure migliore in quanto assume comportamenti migliori in prossimità di essi. Può anche trasferire le strutture dell’immagine di guida all’output del filtraggio, consentendo nuove applicazioni di filtraggio come il dehazing (rimozione di effetto “nebbia”) e il guided feathering (sfumatura guidata dell’immagine). Inoltre, il filtro guidato ha naturalmente un algoritmo di tempo lineare veloce e non approssimativo, indipendentemente dalle dimensioni del kernel e dalla gamma d’intensità ed è attualmente uno dei filtri di per la conservazione dei bordi più veloci. È inoltre dimostrato che è sia efficace che efficiente in una grande varietà di applicazioni di computer vision, computer grafica compresi edge-aware smoothing, valorizzazione dei dettagli, compressione HDR, opacità / sfumatura dell’immagine, dehazing, joint upsampling etc.

In primo luogo, definiamo un processo di filtraggio generale lineare di traduzione-variante, che coinvolge un’immagine di guida I , un’immagine di input di filtraggio p e un’immagine di uscita q . Sia I che p sono dati possono essere identici. L’output di filtraggio ad un pixel i è espresso come media ponderata:

$$q_i = \sum_j W_{ij}(I)p_j$$

dove i e j sono indici di pixel. Il kernel del filtro W_{ij} è una funzione dell’immagine di guida I e indipendente da p . Questo filtro è lineare rispetto a p .

Implementazione:

L’assunzione chiave del filtro guidato è un modello lineare locale tra l’immagine guida I e l’output q . Assumendo che q sia una trasformazione lineare di I in una finestra w_k centrata sul pixel k

$$q_i = \mathbf{a}_k I_i + b_k, \forall i \in \omega_k \quad (\text{eq. 1})$$

Dove (a_k, b_k) sono alcuni coefficienti lineari considerati costanti in w_k . Usiamo una finestra quadrata di raggio r . Questo modello lineare locale assicura che q ha un bordo solo se ho un bordo in quanto $\nabla q = a \nabla I$. Per determinare i coefficienti lineari (a_k, b_k) abbiamo la necessità di introdurre un limite dall’input p . L’output q è ottenuto sottraendo alcune componenti non volute come rumore/textures

$$q_i = p_i - n_i$$

Dobbiamo trovare una soluzione che minimizzi la differenza tra q e p mantenendo il modello lineare di eq 1. Minimizziamo la seguente funzione costato nella finestra w_k

$$E(a_k, b_k) = \sum_{i \in \omega_k} ((a_k I_i + b_k - p_i)^2 + \epsilon a_k^2)$$

Dove epsilon è il parametro di regolarizzazione che penalizza grandi ak. La soluzione d’ellequazione sopra è data da:

$$a_k = \frac{\frac{1}{|\omega|} \sum_{i \in \omega_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \epsilon},$$

$$b_k = \bar{p}_k - a_k \mu_k.$$

Dove μ_k e σ_k^2 sono la media e la varianza di I in ω_k , $|\omega|$ è il numero di pixel in ω_k e $\bar{p}_k = \frac{1}{|\omega|} \sum_{i \in \omega_k} p_i$ è la media di p in ω_k . Avendo ottenuto i coefficienti lineari (a_k, b_k) possiamo ottenere l’output q_i dalla eq1

Ogni pixel i sarà coinvolto in tutte le finestre sovrapposte ω_k , di conseguenza il valore di q_i è diverso in base alla dimensione della finestra. Una strategia semplice è calcolare la media di tutti i possibili valori di q_i . Quindi dopo aver calcolato i coefficienti lineari (a_k, b_k) per tutte le finestre:

$$q_i = \frac{1}{|\omega|} \sum_{k|i \in \omega_k} (a_k I_i + b_k)$$

Notiamo dunque che $\sum_{k|i \in \omega_k} a_k = \sum_{k|i \in \omega_k} a_k$ data dalla simmetria delle finestre (box window), riscriviamo l’equazione principale come:

$$q_i = \bar{a}_i I_i + \bar{b}_i$$

Dove \bar{a}_i e \bar{b}_i sono le media dei coefficienti di tutte le finestre sovrapposte su i .

Algoritmo implementato

Estensione per immagini a colori

Il filtro di guida può essere facilmente esteso alle immagini a colori. In questo caso l’input sarà multicanale e sarà necessario applicare il filtro ad ogni canale indipendentemente. Nel caso in cui la immagine guida sia multicanale dobbiamo riscrivere l’equazione principale come:

$$q_i = \mathbf{a}_k^T \mathbf{I}_i + b_k, \quad \forall i \in \omega_k$$

² <http://mmlab.ie.cuhk.edu.hk/pdf/pami12Guided%20Image%20Filtering.pdf>

Dove I_i è un vettore color 3x1, a_k è un vettore coefficiente 3x1 coefficiente, mentre q_i e b_k sono scalari. Quindi il filtro bilaterale diventerà

$$\mathbf{a}_k = (\Sigma_k + \epsilon \mathbf{U})^{-1} \left(\frac{1}{|\omega|} \sum_{i \in \omega_k} \mathbf{I}_i p_i - \mu_k \bar{p}_k \right)$$

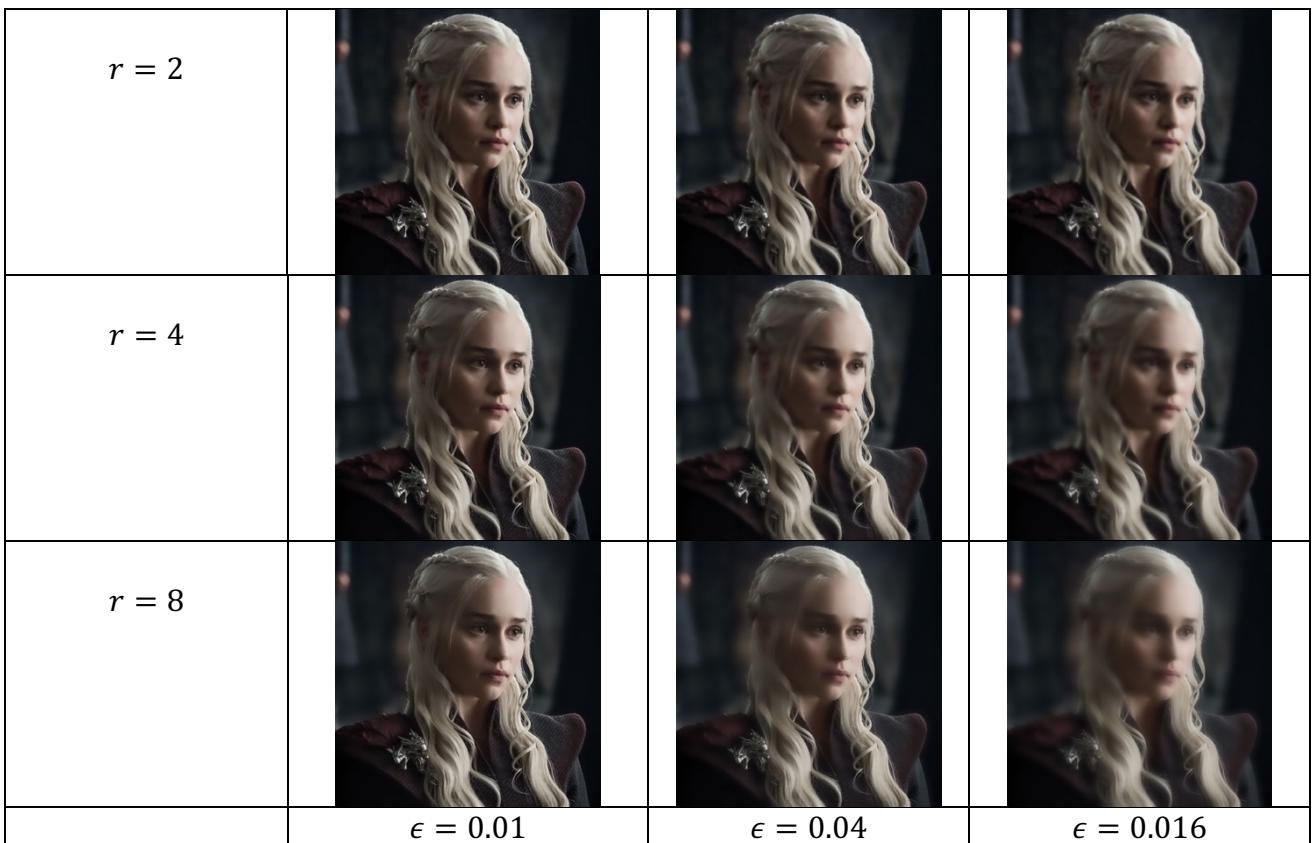
$$b_k = \bar{p}_k - \mathbf{a}_k^T \mu_k,$$

$$q_i = \bar{\mathbf{a}}_i^T \mathbf{I}_i + \bar{b}_i.$$

Dove Σ_k è la matrice covariante 3x3 di I in w_k e \mathbf{U} è una matrice identità 3x3. Preserva i bordi che non sono distinguibili nelle immagini in bianco e nero.

Output guided filter

PSNR	$\sigma_r = 0.1$	$\sigma_r = 0.2$	$\sigma_r = 0.4$
$\sigma_s = 2$	36.46200728635879	33.85884174789064	33.17172787656282
$\sigma_s = 4$	35.98756388492862	33.00000268502831	32.199713885645636
$\sigma_s = 8$	35.49662869424799	32.05739044562387	31.147644066700035



Canny

Come ultimo, ma non per importanza, viene implementato l'algoritmo di Canny, un operatore per il riconoscimento dei bordi. L'algoritmo è implementato seguendo su tre principi fondamentali:

- Devono essere individuati per la maggior parte (buon riconoscimento)
- Devono essere più vicini ai contorni reali dell'immagine (buona localizzazione)
- Il contorno deve essere marcato solo una volta e il rumore presente nell'immagine non ne deve influenzare la scelta (risposta minima)

Si può dividere in quattro fasi:

1. Riduzione del rumore: viene utilizzato in questo caso un filtro basato sulla derivata prima di una gaussiana. Dato che sarà presente del rumore l'immagine viene sottoposta a convoluzione con un filtro gaussiano dove nel risultato di output non vi sarà alcun disturbo di livello significativo
2. Ricerca del gradiente della luminosità di un'immagine: conoscendo a priori che un contorno di un'immagine può puntare verso una direzione qualsiasi, quindi usa quattro filtri differenti per individuare i contorni orizzontale, verticale e le due diagonali a cui è stato applicato precedentemente il filtro gaussiano. Per ciascun pixel verrà considerata la sola direzione relativa al filtro che dà il valore maggiore. Questa direzione combinata con il valore ottenuto applicando il filtro, corrisponde a quella in cui si ha il massimo gradiente di luminosità in ciascun punto dell'immagine.
3. Soppressione dei non-massimi: avremo una mappa dei gradienti che fornirà il valore di intensità luminosa in ciascun punto dell'immagine. Una forte intensità indicherà una forte presenza di contorno, tuttavia solo i punti corrispondenti a dei massimi locali sono considerati come appartenenti ad un contorno e saranno presi in considerazione dai successivi step di elaborazione. Un massimo locale si ha nei punti in cui la derivata del gradiente si annulla
4. Individuazione dei contorni con sogliatura e isteresi: l'estrazione dei contorni dalla mappa introdotta precedentemente si esegue con sogliatura e isteresi. Si definiscono due soglie, una bassa e una alta che vengono confrontate con il gradiente in ciascun punto. Considerando il valore del gradiente:
 - a. Se inferiore alla soglia bassa, il punto è scartato
 - b. Se superiore alla soglia alta, il punto è accettato come parte di un contorno
 - c. Se compreso tra due soglie sarà accettato solo se contiguo ad un punto precedentemente accettato

Al termine di questo step si ottiene un'immagine binaria dove ciascun pixel sarà marcato come appartenente o no ad un contorno.

L'algoritmo dipenderà dunque da due parametri:

- Dimensione del filtro gaussiano: a filtri più piccoli corrisponde una minore sfocatura, di conseguenza permettono di riconoscere contorni più netti, mentre filtri più grandi sono più utili per il riconoscimento di contorni più ampi e sfumati
- La grandezza delle due soglie: una soglia settata ad un valore troppo alto può provare la perdita di informazioni significative, mentre una soglia settata ad un valore troppo basso può far sì che informazioni irrilevanti possano essere considerati durante il calcolo.

Output canny edge detections

Applicando il filtro con gli stessi valori di soglia (0.1 e 0.8) ma aumentando la finestra conferiamo ciò detto precedentemente:

