Azzolini Riccardo 2019-10-23

Operatori di smoothing

1 Operatori di smoothing

Gli **operatori di smoothing** sono **filtri passa-basso**: enfatizzano le basse frequenze e attenuano le alte frequenze. Essi determinano uno sfocamento dell'immagine, che è utile per:

- eliminare piccoli dettagli;
- ridurre il rumore nell'immagine;
- attenuare falsi contorni, dovuti a una quantizzazione non accurata.

Essendo filtri passa-basso, tutti questi operatori lasciano inalterate le zone a valori costanti.

2 Operatori lineari di smoothing

Un operatore lineare - convolutivo di smoothing ha la formula generale

$$g(x,y) = \frac{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)}{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)}$$

dove:

- per una maschera $m \times n$, si hanno $m = 2a + 1$ e $n = 2b + 1$;
- il denominatore è una costante, quindi lo si può portare all'interno delle sommatorie al numeratore per ricondursi alla formulazione generale dei filtri lineari.

Esistono due varianti di questo tipo di filtro:

- operatore di **media** (o *box filter*), nel quale tutti i coefficienti della maschera sono uguali a 1;

$$\frac{1}{9} \cdot \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \hline \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \hline \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \hline \end{array}$$

- **media pesata**, con pesi inversamente proporzionali alla distanza dal centro, che riduce gli effetti indesiderati di blurring.

$$\frac{1}{16} \cdot \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \\ \hline \frac{2}{16} & \frac{4}{16} & \frac{2}{16} \\ \hline \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \\ \hline \end{array}$$

Siccome i coefficienti sono positivi, questi operatori eseguono una sorta di integrazione dei valori dei pixel nell'intorno. Se tale intorno è piccolo, ad esempio 3×3, i pixel integrati sono generalmente correlati tra loro, e quindi la media è vicina al valore originale del pixel centrale. Invece, con un intorno più grande, si ha un effetto più forte (tranne nelle zone omogenee, che questi operatori lasciano sempre invariate), perché alla media partecipano pixel meno correlati tra loro, quindi i valori di output tendono a deviare maggiormente dagli originali, perdendo così il loro significato.

3 Eliminazione dei dettagli

Il filtro di media si può applicare per eliminare i dettagli di piccole dimensioni. Infatti, proprio perché sono piccoli, con una maschera di media sufficientemente grande essi tendono ad assumere valori molto simili al background.

Quest'operazione è utile, ad esempio, prima di eseguire una sogliatura per l'estrazione degli oggetti.

4 Rumore

Si definisce **rumore** la componente del segnale acquisito che devia dal segnale ideale. Le principali cause del rumore sono:

- variazioni di sensitività dei sensori di acquisizione;
- errori di quantizzazione e/o trasmissione;
- fenomeni fisici legati alla natura della radiazione elettromagnetica.

4.1 Image independent noise

L'**image independent noise** (rumore indipendente dal segnale immagine) viene generalmente descritto da un modello additivo,

$$f(i,j) = s(i,j) + n(i,j)$$

dove:

- $f(i,j)$ è l'immagine acquisita;
- $s(i,j)$ è l'immagine ideale;
- $n(i,j)$ è la componente di rumore.

4.1.1 Detector noise

Il **detector noise** è il rumore (indipendente dal segnale immagine) dovuto alla natura discreta della radiazione elettromagnetica: il sensore cattura il segnale in termini di conteggio di fotoni, e si hanno variazioni più sensibili in condizioni di scarsa radiazione (illuminazione) o di elevata temperatura. Questo tipo di rumore è anche detto **gaussiano**, perché le variazioni di livelli di grigio che provoca, cioè i valori di $n(i,j)$, sono distribuiti secondo una gaussiana con media 0, caratterizzata da una determinata deviazione standard σ.

4.1.2 Rumore impulsivo

Il **rumore impulsivo**, detto anche **sale e pepe** (**salt and pepper / data drop out noise**), è principalmente dovuto a errori nella trasmissione del segnale: i pixel corrotti da tale rumore assumono valori pari a 0 (nero) oppure al massimo (bianco, ad esempio 255 se l'immagine è a 8 bit), mentre gli altri pixel rimangono inalterati. Esso è caratterizzato dalla percentuale di pixel corrotti.

4.2 Image dependent noise

L'**image dependent noise** (rumore dipendente dal segnale immagine) si genera solitamente quando la radiazione incidente viene riflessa da una superficie la cui rugosità è nell'ordine di grandezza della lunghezza d'onda della radiazione: nell'immagine che cattura la radiazione riflessa, si ha allora un rumore detto *speckle*. Esso è descritto da un modello moltiplicativo, non lineare.

5 Filtro di media per l'eliminazione del rumore

Solitamente, il rumore è distribuito sulle alte frequenze, quindi può essere eliminato con filtri passa-basso, cioè di smoothing. Essi, però, degradano anche le alte frequenze "desiderate", corrispondenti ai piccoli dettagli e ai bordi, ma questi sono spesso rappresentati da variazioni dei valori di grigio più ampie rispetto a quelle dovute al rumore, quindi possono rimanere accettabili anche dopo lo smoothing.

In particolare, il filtro di media è efficace nella rimozione del rumore gaussiano, ma non di quello impulsivo, perché i pixel corrotti costituiscono degli "spike" (grosse variazioni dei livelli di grigio rispetto ai valori reali del segnale), che influenzano eccessivamente il calcolo della media, quindi i punti di rumore possono addirittura tendere a dilatarsi.

6 Smoothing con sogliatura

Una delle possibili soluzioni per ridurre la degradazione dei bordi provocata dai filtri di smoothing è l'introduzione di un'operazione di sogliatura: il risultato finale $g'(x, y)$ è quello calcolato dal filtro, $g(x, y)$, se la differenza tra questo e il valore originale $f(x, y)$ non supera una certa soglia T, altrimenti il pixel rimane inalterato.

$$g'(x,y) = \begin{cases} g(x,y) & \text{se } |g(x,y) - f(x,y)| < T \\ f(x,y) & \text{altrimenti} \end{cases}$$

Questo metodo ha però degli svantaggi:

- aumenta il costo computazionale;
- non è facile tarare la soglia.

7 Filtro gaussiano

Il **filtro gaussiano** è un filtro di media pesata con coefficienti derivati da una funzione gaussiana bidimensionale.

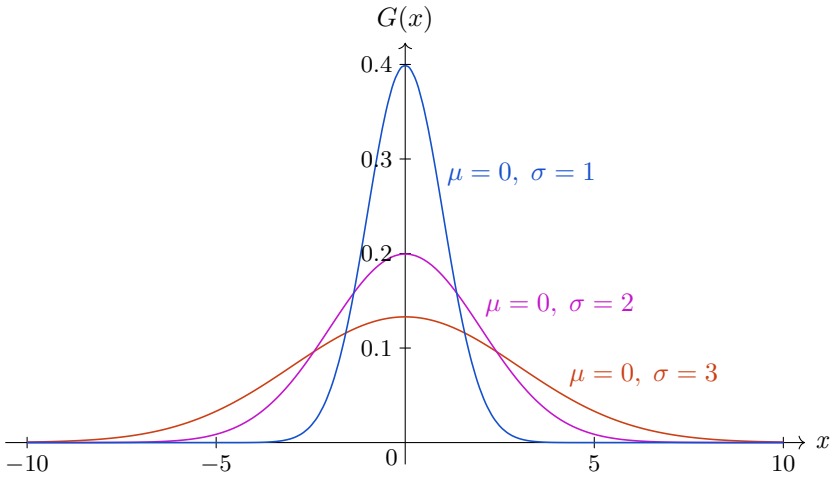
La gaussiana unidimensionale con media μ e deviazione standard σ ha la formula:

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Tale funzione:

- raggiunge in $x = \mu$ il valore massimo $G(\mu) = \frac{1}{\sigma\sqrt{2\pi}}$;

- ha punti di flesso in $x = \mu \pm \sigma$.



Una gaussiana bidimensionale centrata in 0 si ottiene come prodotto di due gaussiane orientate lungo gli assi x e y. La sua formula è:

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

La maschera a valori discreti del filtro gaussiano si ottiene campionando questa gaussiana bidimensionale. L'effetto del filtro dipende dal valore del parametro σ: più esso è alto, maggiore è l'effetto di smoothing, perché all'aumentare di σ diminuiscono i pesi dei pixel vicini al centro e aumentano quelli dei pixel più esterni.

Il filtro gaussiano ha un comportamento simile al filtro di media "normale", quindi anch'esso è efficace nella rimozione del rumore gaussiano, ma non di quello impulsivo.

7.1 Dimensione della maschera

È necessario un criterio per dimensionare la maschera in modo che essa rappresenti una buona approssimazione della gaussiana: una maschera più grande costituisce un'approssimazione migliore, ma aumenta il costo computazionale dell'operazione di filtraggio.

Con σ piccolo, è sufficiente una maschera di dimensioni minori, perché i valori della gaussiana diminuiscono più rapidamente allontanandosi dal centro. Perciò, il criterio di dimensionamento deve dipendere da σ.

L'area sottesa alla gaussiana unidimensionale è pari a

- 0.68 per $|x| \leq \sigma$

- 0.95 per $|x| \leq 2\sigma$
- 0.99 per $|x| \leq 3\sigma$

Si può allora definire un kernel $(2k+1)\times(2k+1)$, con $k = 3\sigma$, una buona approssimazione della gaussiana. Esso, però, tende a essere piuttosto grande:

- per $\sigma = 1$ è 7×7;
- per $\sigma = 2$ è 13×13;
- per $\sigma = 3$ è 19×19.

Spesso, quindi, per velocizzare l'elaborazione, si preferisce porre $k = 2\sigma$, ottenendo così un kernel più piccolo che è un'approssimazione comunque accettabile.

Ad esempio, il kernel ottenuto con $\sigma = 1$ e $k = 2\sigma$ è:

$$\frac{1}{273} \cdot \begin{array}{|c|c|c|c|c|} \hline 1 & 4 & 7 & 4 & 1 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 7 & 26 & 41 & 26 & 7 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 1 & 4 & 7 & 4 & 1 \\ \hline \end{array}$$

8 Filtro mediano

Il **filtro mediano** non è un operatore convolutivo, ma appartiene invece alla famiglia degli **operatori order-statistics**. Esso usa infatti una finestra "vuota", senza coefficienti: legge direttamente i valori sottesi alla finestra nell'immagine di input e ne calcola la mediana (cioè il 50° percentile), ordinandoli e selezionando quello nella posizione centrale della sequenza.

Tale filtro è efficace nella riduzione del rumore sale e pepe, perché la presenza di eventuali outlier (spike) non influenza il risultato (a meno che gli outlier siano particolarmente numerosi in un certo intorno). Inoltre, esso non degrada eccessivamente le zone di transizione (bordi) dell'immagine.

8.1 Altri operatori order-statistics

Esistono anche altri operatori order-statistics, che però non sono filtri di smoothing.

- L'operatore di minimo seleziona il valore più basso tra i pixel dell'intorno, dilatando quindi le zone scure.

- L'operatore di massimo seleziona il valore più alto presente nell'intorno del pixel, dilatando così le zone chiare.

Ad esempio, per una maschera 3×3, se i valori sottesi dell'immagine sono z_i, con $i = 1, 2, \ldots, 9$,

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

l'operatore di minimo assegna al pixel centrale il valore

$$R = \min\{z_i \mid i = 1, 2, \ldots, 9\}$$

mentre il massimo assegna il valore

$$R = \max\{z_i \mid i = 1, 2, \ldots, 9\}$$

Questi operatori sono utili per il processo di estrazione dei bordi di un'immagine: essi permettono di eliminare la frammentazione dei bordi, dilatando le zone chiare nell'immagine prodotta da un operatore di estrazione dei bordi.

9 Conservative smoothing

Il filtro di **conservative smoothing** sacrifica la soppressione del rumore a favore della conservazione dei dettagli. Esso è progettato principalmente per rimuovere gli spike (rumore sale e pepe), mentre è meno efficace per il rumore gaussiano.

Esso cerca di rendere il valore del pixel da elaborare "consistente" con i valori dei pixel vicini:

- se il valore del pixel centrale è compreso tra i valori minimo e massimo dei pixel nel suo intorno, esso è lasciato inalterato;
- altrimenti, l'intensità del pixel centrale è riportata al valore minimo dell'intorno (se era più bassa) o al massimo (se era più alta).

Rispetto al filtro mediano, il conservative smoothing taglia ancora meno le alte frequenze, ma in compenso tende a non rimuovere completamente il rumore impulsivo (perché non elimina uno spike se nel suo intorno ne è presente un altro).

10 Confronto tra filtri di smoothing

I filtri lineari (media, filtro gaussiano, ecc.):

- possono rimuovere efficacemente il rumore additivo (es. gaussiano);
- degradano i bordi e i piccoli dettagli;
- non rimuovono totalmente il rumore impulsivo.

Invece, gli operatori non lineari (filtro mediano e conservative smoothing):

- preservano l'informazione spaziale, poiché riutilizzano valori esistenti nell'intorno del pixel originale;
- sono efficaci nella rimozione del rumore impulsivo;
- sono meno efficaci nella rimozione del rumore additivo, soprattutto se molto invasivo.

11 Crimmins Speckle Removal

Il **Crimmins Speckle Removal** è un algoritmo iterativo che riduce il rumore (sia impulsivo che gaussiano) confrontando ciascun pixel con i suoi 8 vicini e "aggiustandolo" gradualmente per renderlo più simile a essi:

1. se il pixel è più scuro (oltre una certa soglia) del suo vicino a nord, viene schiarito;
2. se il pixel è più scuro del suo vicino a sud, viene schiarito;
3. se il pixel è più chiaro del suo vicino a nord, viene scurito;
4. se il pixel è più chiaro del suo vicino a sud, viene scurito;
5. la procedura si ripete poi con i vicini a est/ovest, sud-est/nord-ovest, e nord-est/sud-ovest, in modo da analizzare tutto l'intorno a 8 del pixel.

All'aumentare delle iterazioni, però, l'effetto di smoothing non rimane limitato agli spike (i pixel con differenza rispetto ai vicini superiore alla soglia fissata), ma tende a propagarsi anche ai loro vicini, perché, modificando un pixel per eliminare uno spike, può succedere che il cambiamento di valore faccia diventare spike i suoi vicini.

Questo algoritmo è molto efficace nel rimuovere rumore caratterizzato da deviazioni di poche unità rispetto ai valori reali (ad esempio, il rumore gaussiano non troppo invasivo), senza degradare di molto l'immagine.