Azzolini Riccardo 2019-10-18

Operazioni aritmetiche e logiche

1 Operazioni aritmetiche

Le **operazioni aritmetiche** si applicano *pixel per pixel* a due o più immagini della stessa dimensione: ogni pixel del risultato è calcolato a partire dai pixel aventi le stesse coordinate nelle diverse immagini di input.

Le operazioni aritmetiche sono:

- **somma**: $R(x, y) = A(x, y) + B(x, y)$
- **sottrazione**: $R(x, y) = A(x, y) - B(x, y)$
- **moltiplicazione** e **divisione** (equivalenti):

$$R(x, y) = A(x, y) \cdot B(x, y) \qquad R(x, y) = \frac{A(x, y)}{B(x, y)}$$

Come caso particolare, queste operazioni si possono applicare anche a una singola immagine, combinando i valori dei pixel con una costante:

$$\begin{aligned} R(x, y) &= A(x, y) + C \\ R(x, y) &= A(x, y) - C \\ R(x, y) &= A(x, y) \cdot C \\ R(x, y) &= \frac{A(x, y)}{C} \end{aligned}$$

1.1 Somma

Solitamente, l'operatore di somma non è utilizzato come funzione singola, bensì composto all'interno di procedure di analisi nelle quali viene impiegato ripetutamente e/o combinato con altri operatori.

Un esempio di applicazione è la **riduzione del rumore** mediante **image averaging**. Il **rumore** presente in un'immagine è definito come una componente del segnale acquisito che devia dal segnale ideale:

$$g(x, y) = f(x, y) + r(x, y)$$

dove f è il segnale ideale, g è il segnale acquisito e r è il rumore (additivo).

L'idea dell'image averaging è ridurre il rumore sommando più immagini $g_i(x, y)$ della stessa scena:

$$\overline{g}(x, y) = \frac{1}{K} \sum_{i=1}^{K} g_i(x, y)$$

Si può dimostrare che, se il rumore è scorrelato (non dipende dal segnale) e con media 0, il valore atteso di $\overline{g}(x, y)$ è $f(x, y)$.

Per applicare questa tecnica, è però necessario che la scena sia pressoché costante, altrimenti non è possibile acquisire un numero sufficiente di immagini.

La somma di un'immagine con una costante, invece, permette di effettuare uno schiarimento.

1.1.1 Scaling

Se si sommano K immagini, ciascuna con valori nel range $[0, L-1]$, l'immagine risultante ha range $[0, K(L-1)]$. Ad esempio, un'immagine a 8 bit ha range $[0, 255]$, quindi una somma di K immagini a 8 bit ha range $[0, K \cdot 255]$.

Per riportare i valori nel range di visualizzazione, si effettua uno scaling, dividendo per K i valori risultanti dalla somma:

$$\frac{g(x, y)}{K}$$

Osservazione: Lo scaling è necessario solo se si vuole visualizzare l'immagine. Altrimenti, le trasformazioni successive si possono applicare direttamente sul range di valori ottenuto dall'operazione.

1.2 Sottrazione

La sottrazione può essere utilizzata per enfatizzare le differenze tra due immagini, permettendo anche di individuare quelle non visibili percettivamente.

Un altro uso della sottrazione è l'eliminazione della variazione di illuminazione dello sfondo, per rendere più facilmente analizzabili gli oggetti di interesse.

Sottraendo una costante, invece, si ottiene uno scurimento dell'immagine.

1.2.1 Scaling

Nella differenza tra due immagini, ad esempio a 8 bit, il massimo range possibile di valori risultanti è $[-255, 255]$. Se si vogliono riportare i valori nel range di visualizzazione $[0, 255]$, sfruttando tutta la dinamica, la procedura di scaling consiste nel:

1. considerare il minimo valore di grigio dell'immagine differenza;
2. sommare il suo negativo a tutti i pixel $g(x, y)$, producendo così un'immagine con valori che sono positivi e partono da 0;
3. moltiplicare per la costante $\frac{255}{Max}$, dove Max è il massimo dell'immagine ottenuta al punto 2.

1.3 Moltiplicazione e divisione

Un'applicazione della moltiplicazione (o, equivalentemente, della divisione) è l'operazione di **shading correction**, che serve a correggere la presenza di zone più chare/scure in un'immagine, dividendola per (o, equivalentemente, moltiplicandola per il reciproco di) una seconda immagine che contiene solo le differenze di illuminazione.

La moltiplicazione può anche essere usata per il **mascheramento**, cioè per isolare le regioni di interesse (RoI: *Region of Interest*) e azzerare le altre. A tale scopo, l'immagine originale viene moltiplicata per una **maschera**: un'immagine binaria che ha pixel bianchi (1) nelle zone da isolare, cioè lasciare inalterate, e pixel neri (0) nelle zone da escludere.

La moltiplicazione per una costante C, invece, effettua uno scaling dell'immagine, permettendo di schiarirla (se $C > 1$) o scurirla (se $0 < C < 1$). Si possono ottenere effetti simili con l'addizione/sottrazione di una costante, ma la moltiplicazione dà risultati migliori perché preserva meglio il contrasto dell'immagine.

1.4 Blending

L'operatore di **blending** implementa una combinazione lineare di due immagini:

$$R(x, y) = wA(x, y) + (1 - w)B(x, y)$$

Il parametro w, compreso tra 0 e 1, determina il peso di ciascuna immagine nella combinazione. In particolare, con $w = 0.5$ le due immagini hanno peso uguale.

1.5 Gestione dell'overflow

In pratica, gli operatori aritmetici sono spesso implementati in modo da produrre solo risultati nel range di visualizzazione (ad esempio $[0, 255]$). I valori che superano il massimo consentito possono essere gestiti in due modi:

- troncamento al massimo (che causa un effetto di saturazione, perché si ha una perdita di informazione);
- *wrapping around* con ripartenza dei valori dallo zero (ad esempio, $255 + 1$ dà come risultato 0, $255 + 2$ dà 1, ecc.).

In generale, nell'applicazione degli operatori è meglio evitare che l'overflow si verifichi: ad esempio, prima di sommare una costante che provocherebbe l'overflow di alcuni pixel si può effettuare uno scaling con fattore minore di 1, che riduca i valori presenti nell'immagine abbastanza da prevenire l'overflow in seguito alla somma.

2 Operatori logici

Gli **operatori logici** si applicano pixel per pixel, ma, a differenza degli operatori aritmetici, processano i pixel come stringhe di cifre binarie, applicandosi bit a bit.

2.1 AND

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

L'operatore di AND permette di mascherare le regioni di interesse dell'immagine, analogamente alla moltiplicazione. Ciò può essere utile, ad esempio, per effettuare operazioni di enhancement solo su una certa zona di un'immagine:

1. si estrae la zona da elaborare mediante mascheramento;
2. si effettuano su di essa le operazioni desiderate;
3. si ricombina la zona elaborata con il resto dell'immagine (mediante una somma).

Inoltre, l'AND effettua l'intersezione tra due immagini, quindi lo si può utilizzare per isolare le parti che *non* hanno subito mutamenti (al contrario della sottrazione). A tale scopo, esso funziona correttamente solo se si hanno *oggetti chiari* (assimilabili al valore logico 1) su *sfondo scuro* (assimilabile a 0): in caso contrario, bisogna prima invertire i valori di grigio delle due immagini (mediante l'operatore di negativo/NOT). Comunque,

i risultati migliori si ottengono lavorando su immagini binarie, quindi spesso conviene effettuare una sogliatura prima di applicare l'AND.

Infine, l'AND si può effettuare con una costante per eseguire il bit slicing, cioè mantenere solo alcuni bit di ciascun pixel e azzerare gli altri. Ad esempio, per un'immagine a 8 bit, applicando l'AND con la costante 128 (10000000) si seleziona solo il bit più significativo (il che è equivalente a una sogliatura con soglia 128).

2.2 OR

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

L'OR funziona in modo complementare all'AND:

- Può essere utilizzato per il mascheramento, ma seleziona le zone che nella maschera sono a 0, e rende completamente *bianche* (invece che nere, come fa l'AND) le zone da escludere, che nella maschera sono indicate da valori 1.
- Permette di individuare le parti non cambiate tra due immagini che hanno oggetti scuri su sfondo chiaro (al contrario dell'AND). Comunque, come nel caso dell'AND, per ottenere risultati affidabili è necessario lavorare su immagini binarie.

2.3 XOR

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

L'operatore XOR evidenzia le zone che non sono identiche in due immagini. Come gli altri operatori logici, per avere risultati affidabili deve essere applicato su immagini binarie.

2.4 NOT

Nell'elaborazione di un'immagine binaria può essere necessario cambiare la polarità. Ad esempio, se si hanno due immagini con oggetti neri su sfondo bianco, l'operatore OR effettua l'unione dello sfondo. Invece, usando prima il NOT per invertire le immagini di input (oggetti bianchi su sfondo nero), quando poi si applica l'OR viene realizzata l'unione degli oggetti.

Il NOT equivale al negativo di un'immagine, $s = 255 - t$.

3 Operazioni insiemistiche

Le **operazioni insiemistiche** si applicano a insiemi di pixel appartenenti a regioni all'interno dell'immagine. Si assume che i pixel di una regione (insieme) abbiano la stessa intensità, dato che queste operazioni non definiscono i valori di grigio dei pixel che costituiscono il risultato.