Elaborazione di Segnali Multimediali

Operazioni morfologiche

L.Verdoliva, D.Cozzolino

Le tecniche di enhancement studiate finora si basano tipicamente su operazioni di tipo lineare, tuttavia spesso può essere molto utile considerare un approccio di tipo non lineare quando si devono risolvere problemi che coinvolgono gli aspetti geometrici di un'immagine. Un potente metodo non lineare usa la morfologia matematica per rappresentare e descrivere le forme degli oggetti presenti in un'immagine. In tal caso il valore di ogni pixel dell'immagine in uscita è determinato mediante un confronto del pixel corrispondente dell'immagine in ingresso con i suoi vicini. Scegliendo opportunamente la dimensione e la forma del vicinato, è possibile costruire operazioni morfologiche che sono sensibili a certe specifiche strutture geometriche. Il vicinato può essere definito dalle posizioni che i pixel hanno rispetto a quello da elaborare mediante un elemento strutturante (*structuring element*, SE). In modo simile a quanto accade per il filtraggio bidimensionale, l'elemento strutturante rappresenta la maschera con cui modificare i pixel questa volta attraverso operazioni di logica booleana. Nel seguito descriveremo le operazioni basilari dapprima per le immagini binarie e poi per quelle su scala di grigio.

1 Immagini binarie

La matematica necessaria per descrivere gli operatori morfologici è la teoria degli insiemi. Cominciamo col considerare un'immagine $x(m,n)$ su due livelli i cui pixel assumono valore 0 per il background e 1 per l'oggetto presente (foreground). Se facciamo riferimento al solo foreground, le coordinate (m,n) di ogni suo elemento sono interi appartenenti a $\mathbf{Z}^2$ e x è la corrispondenza che assegna 0 o 1 ad ogni coppia di coordinate (m,n). Sia A un sottoinsieme di $\mathbf{Z}^2$ i cui elementi sono le coordinate dei pixel, se $w=(m,n)$ è un elemento di A allora si può scrivere:

$$w \in A$$

altrimenti

$$w \notin A$$

L'insieme delle coordinate dei pixel che soddisfa una determinata condizione si scrive come:

$$B = \{w|\text{condizione}\}$$

Per esempio l'insieme di tutte le coordinate di pixel che non appartengono ad A si chiama complemento di A e si indica con A^c:

$$A^c = \{w|w \notin A\}$$

L'unione di due insiemi:

$$C = A \cup B$$

è l'insieme di tutti gli elementi che appartengono ad A, B o entrambi. L'intersezione invece:

$$C = A \cap B$$

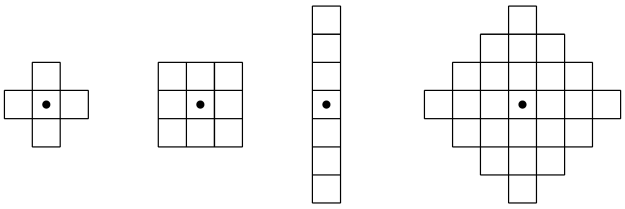

Figura 1: Esempi di elementi strutturanti.

è l'insieme di tutti gli elementi che appartengono ad entrambi gli insiemi. Infine definiamo la differenza come:

$$A - B = \{w | w \in A, \notin B\}$$

Provate ad applicare queste operazioni (complemento, unione, intersezione e differenza) alle immagini testo.y (256×256, uint8) e UTK.tif, code vi fornisce a tale scopo le funzioni logiche `np.logical_not`, `np.logical_or`, `np.logical_and`.

Oltre a queste operazioni basilari è necessario definire due operatori specifici per insiemi i cui elementi sono coordinate di pixel: la riflessione e la traslazione. La riflessione è definita come:

$$\widehat{B} = \{w | w = -b, b \in B\}$$

e modifica le generiche coordinate di un pixel (m, n) con $(-m, -n)$, determinando in questo modo la riflessione dell'oggetto che si sta considerando. Invece la traslazione di $z = (z_1, z_2)$ è:

$$(B)_z = \{w | w = b + z, b \in B\}$$

in tal caso le coordinate vengono tutte traslate in $(m + z_1, n + z_2)$. La riflessione e la traslazione risultano fondamentali per realizzare operazioni basate sui cosiddetti elementi strutturanti (*structuring elements*), che rappresentano piccoli insiemi usati per esaminare ed estrarre le proprietà di interesse di un'immagine. Esempi di elementi strutturanti sono mostrati di seguito (il punto nero ne indica l'origine):
Dal punto di vista implementativo è più comodo definire tali elementi come matrici rettangolari di 0 e 1, quindi, per esempio, per i primi due elementi le matrici corrispondenti sono:

0	1	0
1	**1**	1
0	1	0

1	1	1
1	**1**	1
1	1	1

dove il pixel in grassetto questa volta identifica l'origine, cioè la posizione del pixel che viene elaborato. In modo analogo al filtraggio spaziale, facendo scorrere queste finestre su tutta l'immagine e definendo opportunamente l'operazione booleana da realizzare, è possibile estrarre determinati tipi di geometrie presenti. Nei paragrafi seguenti si mostrano diversi esempi.

1.1 Dilatazione e erosione

Due operazioni morfologiche fondamentali sono la dilatazione e l'erosione: la dilatazione aggiunge pixel ai contorni di un oggetto, mentre l'erosione li rimuove. Il numero di pixel aggiunti o rimossi dipendono dalla forma e dalla dimensione dell'elemento strutturante. In modo più formale la dilatazione di A con B (elemento strutturante) è:

$$A \oplus B = \{z | (\widehat{B})_z \cap A \neq \emptyset\}$$

quindi la dilatazione di A con B è l'insieme di tutte le traslazioni tali che $\widehat{B}$ e A si sovrappongano almeno per un elemento. Il processo di ribaltare B rispetto alla sua origine e poi spostarlo in modo da scorrere tutta l'immagine (A) è analogo a ciò che si fa per la convoluzione spaziale. Questa operazione rende più spessi gli oggetti presenti in un'immagine binaria. Supponiamo per esempio di avere la seguente sezione di immagine:

$x(m,n) =$

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

e di considerare come elemento strutturante la matrice:

$b(m,n) =$

1	**1**	1

Questo significa che il vicinato di un pixel è costituito da i due pixel vicini nella direzione orizzontale. La dilatazione di $x(m,n)$ mediante l'elemento strutturante $b(m,n)$ fornisce in uscita il valore 1 se, facendo scorrere $b(-m,-n)$ su $x(m,n)$ nel vicinato del pixel in esame è presente almeno un 1. In questo caso si ottiene in uscita la seguente immagine:

$y(m,n) =$

1	1	0	0	0
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	0	1	1

Si noti che per ottenere anche i valori ai bordi l'immagine è stata estesa con valori nulli (zero-padding). Se l'elemento strutturante fosse stato:

$b'(m,n) =$

1	0	0
0	**1**	0
0	0	1

l'uscita sarebbe stata coincidente con l'immagine in ingresso $x(m,n)$ e questo perchè la forma scelta per la struttura è identica a quella presente nell'immagine. Vale allora ancora una volta il concetto per cui se si vogliono individuare linee nell'immagine bisogna cotruire strutture lineari, in generale bisogna scegliere gli elementi della struttura della stessa forma e dimensione degli oggetti da elaborare.

L'erosione invece fornisce il valore 0 se nel vicinato è presente almeno uno 0 (in tal caso l'estensione ai bordi è fatta con tutti 1). Formalmente:

$$A \ominus B = \{z | (B)_z \cap A^c \neq \emptyset\}$$

Se consideriamo sempre la stessa sezione di immagine $x(m,n)$ e l' elemento strutturante $b(m,n)$, allora è facile riconoscere che in uscita avremo un'immagine costituita da soli elementi nulli. Se invece si fa riferimento alla seguente sezione d'immagine:

$$x(m,n) = \begin{array}{|c|c|c|c|c|}\hline 0&0&0&0&0\\\hline 0&1&1&1&0\\\hline 0&1&1&1&0\\\hline 0&1&1&1&0\\\hline 0&0&0&0&0\\\hline\end{array}$$

e si usa sempre l'elemento $b(m,n)$ si ottiene in uscita:

$$y(m,n) = \begin{array}{|c|c|c|c|c|}\hline 0&0&0&0&0\\\hline 0&0&1&0&0\\\hline 0&0&1&0&0\\\hline 0&0&1&0&0\\\hline 0&0&0&0&0\\\hline\end{array}$$

In Python, le operazioni morfologiche sono incluse nel modulo `morphology` di SKImage. Per le operazioni di dilatazione e erosione si possono usare i seguenti comandi:

```
import skimage.morphology as morph
b  = np.array([[1,1,1]], np.bool)
y1 = morph.binary_dilation(x,b)
y2 = morph.binary_erosion(x,b)
```

In realtà il modulo `morphology` mette a disposizione diverse funzioni (`square`, `rectangle`, `diamond`, `octagon`, `disk`, `star`) per definire in modo semplice l'elemento strutturante, per esempio se volete un elemento strutturante circolare di raggio 7 pixel, si può usare il seguente comando:

```
b  = morph.disk(7)
```

1.1.1 Esercizi proposti

1. *Enhancement di un testo a bassa risoluzione.* Una delle più semplici applicazioni della dilatazione è riempire i vuoti presenti nei caratteri di un'immagine di testo, come l'immagine testo_fax.tif (su cui in precedenza avevamo utilizzato un filtraggio passa-basso). Dato che la massima lunghezza delle interruzioni è pari a due pixel, l'elemento strutturante che si può usare è:

   $$b(m,n) = \begin{array}{|c|c|c|}\hline 0&1&0\\\hline 1&\mathbf{1}&1\\\hline 0&1&0\\\hline\end{array}$$

   Realizzate la dilatazione dell'immagine e osservate il risultato dell'elaborazione. Provate a modificare la forma dell'elemento strutturante e osservate gli effetti sull'immagine elaborata.

2. *Rimozione di oggetti piccoli in un'immagine.* Un'applicazione dell'erosione è quella di eliminare dettagli irrilevanti in un'immagine. L'immagine quadrati.tif è composta da rettangoli di dimensioni 1, 3, 5, 7, 9 e 15 pixel per lato. Supponiamo di voler eliminare tutti i quadrati eccetto i più grandi, allora dobbiamo applicare l'operazione di erosione usando un elemento strutturante di dimensione un po' più piccola degli oggetti che vogliamo conservare, per esempio quadrate di 13×13 pixel. Dato che in questo

modo si conservano solo porzioni dei quadrati più grandi, allora bisogna effettuare anche un'operazione di dilatazione per recuperare la dimensione originale dei quadrati, usando sempre lo stesso elemento strutturante.

3. Data l'immagine forme.tif realizzate l'erosione dell'immagine scegliendo l'elemento strutturante in modo da generare in uscita un'immagine in cui è presente il solo quadrato centrale.

1.2 Opening e closing

Le operazioni di dilatazione ed erosione spesso sono combinate assieme nell'image processing; in particolare se si fa seguire la dilatazione all'erosione si parla di apertura (*opening*):

$$A \circ B = (A \ominus B) \oplus B$$

mentre se accade l'inverso di chiusura (*closing*):

$$A \bullet B = (A \oplus B) \ominus B$$

Consideriamo l'immagine circbw.png, immagine binaria che rappresenta un circuito e di voler rimuovere piccoli oggetti, conservando forma e dimensioni degli oggetti più grandi. Per esempio, se vogliamo rimuovere le linee dal circuito e conservare solo le forme rettangolari dei microchip il codice è:

```
x = io.imread('circbw.tif')
s = morph.rectangle(40,30)
z = morph.binary_erosion(x,s)
y = morph.binary_dilation(z,s)
```

In realtà code prevede i due comandi `morph.binary_opening` e `morph.binary_closing` per realizzare le operazioni di apertura e chiusura.

1.2.1 Esercizi proposti

1. *Riduzione del rumore in un'immagine.* Le operazioni morfologiche possono essere usate per costruire filtri simili concettualmente ai filtri spaziali studiati in precedenza. L'immagine impronta.tif mostra la sezione di un'impronta digitale corrotta da rumore, che si manifesta con pixel chiari sullo sfondo e pixel scuri sull'impronta. Per eliminare il rumore senza modificare le strutture che caratterizzano l'impronta si sceglie come elemento strutturante un quadrato 3×3 e si procede nel seguente modo:
   (a) si effettua l'erosione dell'immagine che rimuove completamente il rumore sullo sfondo (dato che le componenti di rumore sono più piccole dell'elemento strutturante), ma aumenta le dimensioni degli elementi di rumore presenti sull'impronta, che però possono essere ridotte o addirittura eliminate se si effettua la dilatazione. La sequenza di queste due operazioni è l'opening.
   (b) L'operazione precedente ha creato dei vuoti all'interno dell'impronta, bisogna allora effettuare nuovamente la dilatazione, che però rende più spesse le regioni dell'impronta, per cui è necessario realizzare ancora un'erosione. La sequenza di queste due operazioni è il closing.

1.3 Estrazione dei bordi

I contorni di un'immagine binaria si possono ottenere attraverso l'operazione di erosione con un opportuno elemento strutturante e poi sottraendo il risultato all'immagine originale. cioè:

$$\beta(A) = A - (A \ominus B)$$

Per comprendere meglio perché bisogna realizzare queste operazioni consideriamo un semplice esempio e supponiamo che l'immagine binaria di cui si vogliono estrarre i contorni sia la seguente:

$$x(m,n) = \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline 1&1&1&0&1&1&1&1&1&0\\\hline 1&1&1&0&1&1&1&1&1&0\\\hline 1&1&1&1&1&1&1&1&1&1\\\hline 1&1&1&1&1&1&1&1&1&1\\\hline 1&1&1&1&1&1&1&1&1&1\\\hline\end{array}$$

Usiamo poi come elemento strutturante un matrice quadrata 3×3 e realizziamo l'operazione di erosione:

$$z(m,n) = \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline 0&0&0&0&0&0&0&0&0&0\\\hline 0&1&0&0&0&1&1&1&0&0\\\hline 0&1&0&0&0&1&1&1&0&0\\\hline 0&1&1&1&1&1&1&1&1&0\\\hline 0&0&0&0&0&0&0&0&0&0\\\hline\end{array}$$

Sottraendo $z(m,n)$ da $x(m,n)$ si ha:

$$y(m,n) = \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline 1&1&1&0&1&1&1&1&1&0\\\hline 1&0&1&0&1&0&0&0&1&0\\\hline 1&0&1&1&1&0&0&0&1&1\\\hline 1&0&0&0&0&0&0&0&0&1\\\hline 1&1&1&1&1&1&1&1&1&1\\\hline\end{array}$$

Realizzate questa procedura sull'immagine volto.tif.

1.4 Trasformazione Hit-or-Miss

Questa trasformazione è uno strumento base per la shape detection (individuazione delle forme). Mostriamo l'idea su cui si fonda con un esempio. Supponiamo di avere la seguente sezione di immagine:

$$A = \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline 0&1&0&0&1&1&1&1&0&0\\\hline 0&1&0&0&0&0&0&1&1&0\\\hline 1&1&1&0&0&1&0&1&1&1\\\hline 0&1&0&0&1&1&1&0&1&0\\\hline 0&0&0&0&0&1&0&0&0&0\\\hline\end{array}$$

Supponiamo inoltre di voler identificare le posizioni della seguente configurazione di pixel:

0	1	0
1	**1**	1
0	1	0

La sezione di immagine contiene tale configurazione di pixel in due diverse posizioni. Consideriamo quindi due elementi strutturanti:

$$B_1 = \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & \mathbf{1} & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \qquad B_2 = \begin{array}{|c|c|c|}\hline 1 & 0 & 1 \\ \hline 0 & \mathbf{0} & 0 \\ \hline 1 & 0 & 1 \\ \hline \end{array}$$

L'erosione dell'immagine A con B_1 fornisce la posizione dei pixel del foreground che hanno vicini a nord, sud, est e ovest:

$$A \ominus B_1 = \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

Tuttavia in questo modo catturiamo le forme che includono questo tipo di struttura, ma che non sono perfettamente identiche alla struttura stessa. Per escludere dall'analisi quelle diverse bisogna esaminare cosa succede allo sfondo dell'immagine considerando quindi il complemento di A (A^c). Infatti, basta individuare in A^c le posizioni dei pixel che contengono i vicini a nord-est, sud-est, sud-ovest e nord-ovest; il che è equivalente a realizzare l'erosione di A^c con l'elemento strutturante B_2, si ottiene così:

$$A^c \ominus B_2 = \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline \end{array}$$

A questo punto dall'intersezione degli insiemi ottenuti dalle due operazioni di erosione otteniamo le posizioni dei pixel cercate. In conclusione la trasformaione hit-or-miss individua tutte le posizioni dei pixel che hanno una struttura come B_1 (hit, colpisci), ma che non hanno nessuno dei pixel contenuti in B_2 (miss, manca):

$$A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2)$$

Questa operazione è implementata in code con i seguenti comandi:

```
C1 = morph.binary_erosion(A,S1)
C2 = morph.binary_erosion(np.logical_not(A),S2)
C  = np.logical_and(C1,C2)
```

Supponiamo di voler individuare la posizione dei pixel degli oggetti in figura quadrati.tif che hanno vicini solo a est e sud, e non ne hanno nelle direzioni nord-est, nord, nord-ovest, ovest e sud-ovest. Bisogna allora definire i due elementi strutturanti:

```
S1 = np.array([[0,0,0],[0,1,1],[0,1,0]], np.bool)
S2 = np.array([[1,1,1],[1,0,0],[1,0,0]], np.bool)
```

Verificate la correttezza del risultato mostrando l'immagine elaborata.

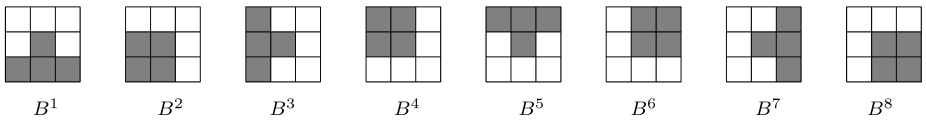

Figura 2: Sequenza di elementi strutturanti usati per il thinning.

1.5 Thinning e Skeletonization

Un'operazione estremamente utile (soprattutto dopo operazioni di thresholding di un'immagine) è il thinning (assottigliamento). Il thinning riduce la forma di un oggetto rendendo le linee più sottili, ciò si può realizzare utilizzando una sequenza di elementi strutturanti:

$$\{B\} = \{B^1, B^2, B^3, \ldots, B^n\}$$

dove B^i è una versione ruotata di B^{i-1}. Definiamo allora il thinning come

$$A \otimes \{B\} = ((\ldots((A \otimes B^1) \otimes B^2)\ldots) \otimes B^n)$$

dove

$$A \otimes B^i = A - (A \circledast B^i)$$

l'operazione base rimuove via via dall'immagine originale il risultato di un'operazione di estrazione delle posizioni di determinate configurazioni di pixel. L'elaborazione complessiva consiste poi nel rendere A più sottile prima usando l'elemento strutturale B^1, quindi il risultato viene elaborato con B^2 e così via. L'intero processo viene ripetuto fino a quando non si verificano ulteriori cambiamenti. Di seguito si mostrano un insieme di elementi strutturanti tipicamente usati per il thinning:
Proviamo ad applicare questa operazione al risultato dell'elaborazione dell'esempio 1 nel paragrafo 1.2.1 che mostra un'immagine binaria di un'impronta digitale in cui i bordi (ridge) sono piuttosto spessi. Può essere conveniente renderli più sottili attraverso una procedura di thinning:

```
y = morph.thin(x, 1)
```

Confrontate il risultato di questa elaborazione con quella che applica la procedura 2 volte, e poi infinite volte (`np.inf`).

La skeletonization (scheletrizzazione) è un'altra operazione che riduce gli oggetti di un'immagine binaria ad un insieme di linee sottili che conservano l'informazione rilevante sulla forma dell'oggetto:

```
y = morph.skeletonize(x)
```

Applicate questa operazione sempre all'immagine dell'impronta digitale e confrontate il risultato ottenuto con la procedura di thinning.

2 Immagini su scala di grigio

Estendiamo alle immagini su scala di grigio le operazioni di dilatazione ed erosione, (le operazioni di apertura e chiusura si definiscono esattamente come per le immagini binarie), che poi useremo per sviluppare alcuni semplici algoritmi morfologici. Indichiamo ancora una volta con $x(m, n)$ l'immagine monocromatica e con $b(m, n)$ l'elemento strutturante, che si comporta concettualmente come per le immagini binarie.

L'erosione di x attraverso un elemento strutturante b è definita per ogni posizione (m, n) come il valore minimo dell'immagine quando l'origine di b è posizionata proprio in (m, n). Formalmente:

$$[x \ominus b](m, n) = \min_{(s,t)\in b}\{x(m + s, n + t)\}$$

L'erosione seleziona quindi il valore minimo di x da tutti i valori contenuti nella regione coincidente con b. Se per esempio b è un elemento strutturante 3×3, per ogni posizione (coincidente con l'origine di b) va calcolato il valore minimo in un intorno quadrato di dimensione 3. Analogamente la dilatazione è definita come il valore massimo dell'immagine nella finestra indicata da $\widehat{b}$, dove $\widehat{b} = b(-m, -n)$:

$$[x \oplus b](m, n) = \max_{(s,t)\in b}\{x(m - s, n - t)\}$$

Dato che l'erosione in scala di grigio calcola il valore di intensità minimo in ogni regione definita dall'elemento strutturante, l'immagine elaborata risulta più scura di quella originale. Per le immagini su scala di grigio si possono usare le funzioni `morph.dilation` e `morph.erosion` per realizzare le operazioni di dilatazione ed erosione. Provate a realizzare l'erosione dell'immagine circuito.tif con un disco di raggio 2: le intensità dei piccoli punti chiari sono state ridotte, mentre sono diventati più spessi gli oggetti scuri. Ripetete l'esperimento realizzando la dilatazione, osserverete che gli effetti ottenuti in questo caso sono esattamente il contrario.

2.1 Smoothing morfologico

Dato che l'operazione di apertura sopprime i dettagli brillanti più piccoli di uno specifico SE e che la chiusura sopprime dettagli scuri, spesso queste operazioni sono usate insieme per realizzare i filtri morfologici utili per lo smoothing e la riduzione del rumore in un'immagine. Consideriamo la figura supernova.tif che mostra la nebulosa del Cigno, fotografata a raggi X dal telescopio Hubble della NASA. Supponiamo che la regione centrale dell'immagine sia l'oggetto di interesse e che le piccole componenti che la circondano sia rumore. Per rimuovere tale rumore si puo effettuare prima l'apertura e poi la chiusura con un SE che è un disco di raggio specifico. Provate a mostrare il risultato dell'elaborazione al variare del raggio del disco (1, 3 e 5) e commentate il risultato.

Spesso si usa una procedura che realizza il filtraggio sequenziale alternato, cioè le operazioni di apertura-chiusura vengono iterate fin quando non si ottiene il risultato desiderato. Ovviamente in questo modo si produce un effetto di blurring più marcato di quello visto nell'esempio precedente.

2.2 Gradiente morfologico

La dilatazione e l'erosione possono essere usate congiuntamente per ottenere il gradiente morfologico di un'immagine x:

$$y = (x \oplus b) - (x \ominus b)$$

La dilatazione rende le regioni più spesse e l'erosione le assottiglia, la loro differenza allora enfatizza i confini tra le regioni. Le aree omogenee non vengono alterate (fintanto che l'SE è relativamente piccolo) per cui saranno eliminate dopo la differenza. Il risultato è un'immagine in cui si vedono bene i bordi tra gli oggetti e in cui non è presente il contributo di zone omogenee, esattamente ciò che si ottiene con un filtro che realizza la derivata.

In figura headCT.tif si mostra una scansione tomografica (TAC) di un cranio. Realizzate il gradiente morfologico usando come elemento strutturante una finestra 3×3 di tutti 1 e osservate i risultati nei diversi stadi di elaborazione (immagine erosa, dilatata, differenza). Ripetete l'esperimento per l'immagine vista_aerea.tif.

2.3 Trasformazioni top-hat e bottom-hat

.

Se si combina la sottrazione con le operazioni di apertura e chiusura si ottengono le trasformazioni *top-hat* e *bottom-hat*. La trasformazione top-hat è:

$$T_{hat}(x) = x - (x \circ b)$$

mentre la bottom-hat:

$$B_{hat}(x) = (x \bullet b) - x$$

Una delle principali applicazioni di questa trasformazione è quella di rimuovere oggetti da un'immagine usando un elemento strutturante che non si abbini a quelli da rimuovere. L'operazione differenza produce un'immagine in cui sono presenti solo le componenti rimosse. La trasformazione top-hat si usa per oggetti chiari su uno sfondo scuro (*white top-hat*), mentre la bottom-hat per il contratrio (*black top-hat*).

Le trasformazioni top-hat correggono efficacemente gli effetti di un'illuminazione non uniforme (*shading correction*). Questa operazione è fondamentale quando si vuole realizzare la segmentazione. Consideriamo l'immagine granelli_riso.tif in cui sono presenti chicci di riso su uno sfondo disomogeneo: più luminoso nella parte superiore. Se provate a segmentare questa immagine con k-means con $K = 2$ noterete come l'illuminazione non uniforme causa degli errori di segmentazione nell'area scura (diversi granelli di riso non vengono estratti dal background) così come nella parte in alto a sinistra dell'immagine alcuni pixel non vengono classificati correttamente. Allo scopo di regolarizzare lo sfondo si vogliono realizzare i seguenti passi (trasformazione top-hat):

1. un'operazione di opening (con un disco di raggio 40) per rimuovere gli oggetti presenti nell'immagine (che non vanno modificati) e stimare il solo background;

2. creare uno sfondo più uniforme sottraendo l'immagine di background a quella originale.

Ripetete la segmentazione sull'immagine così prodotta e confrontate la mappa con quella precedente.

2.4 Segmentazione di regioni con texture tramite operazioni morfologiche

.

In figura blob.tif si mostra un'immagine rumorosa con cerchi scuri su uno sfondo chiaro. In particolare sono presenti due regioni distinte, a destra la regione è caratterizzata da cerchi di dimensioni maggiori rispetto a quella di sinistra. L'obiettivo è trovare un confine tra le due regioni in base al loro contenuto tessiturale. Dato che gli oggetti di interesse sono più scuri rispetto al background, se realizziamo l'operazione di chiusura con un elemento strutturante più grande dei cerchi piccoli, questi saranno rimossi.

Provate allora ad utilizzare come elemento strutturante un disco di raggio 30 (il raggio dei cerchi è pari circa a 25); l'immagine ottenuta è costituita da cerchi grandi su uno sfondo chiaro. A questo punto se effettuiamo l'operazione di apertura con un elemento strutturante che è un disco di raggio maggiore della distanza tra i cerchi otterremo un'immagine in cui le zone chiare tra i blob vengono rimosse. L'immagine fornisce cioè la segmentazione delle due regioni con texture diversa. Se si vuole ottenere la linea di separazione tra le due regioni basta realizzare un gradiente morfologico con un elemento strutturante 3×3 di tutti 1.