Elaborazione di Segnali Multimediali
a.a. 2017/2018

Segmentazione

In questa esercitazione vengono implementate alcune delle tecniche base per la segmentazione di immagini. In particolare, si focalizzerà l'attenzione sugli algoritmi *egde-based*, che si basano su elaborazioni locali per il rilevamento di discontinuità e producono una mappa di contorni, e sugli algoritmi *class-based*, che, invece, si basano su elaborazioni globali e forniscono una mappa di etichette.

1 Tecniche edge-based

Le tecniche edge-based si basano sulle discontinuità presenti in un'immagine per individuare gli oggetti che la compongono. In particolare, si usa un filtro passa-alto che enfatizza le variazioni dell'intensità luminosa e un'operazione di thresholding, che conserva solo quelle più forti presumibilmente associate ai bordi. In questo modo si ottiene la mappa dei contorni, che potrebbe poi essere ulteriormente elaborata, attraverso l'operazione di thinning, nel caso in cui i bordi prodotti presentino uno spessore troppo grande. Di seguito sono presentati diversi esempi basati su questo principio.

1.1 Point e line detection

Per rilevare punti isolati in un'immagine si può procedere nel seguente modo:

- filtrare l'immagine utilizzando la seguente maschera:

-1	-1	-1
-1	8	-1
-1	-1	-1

- sottoporre l'immagine filtrata ad un'operazione di thresholding per individuare il punto isolato.

Applicate questa procedura all'immagine turbina.jpg, che mostra un'immagine a raggi X in cui è presente una porosità in alto a destra, in cui si trova un singolo pixel nero. Scegliendo la soglia pari al 90% del livello di grigio più grande in valore assoluto, è possibile rilevare il punto isolato. E' evidente che questo approccio si basa sul fatto che i punti da rilevare si trovino su uno sfondo omogeneo.

Se invece si vogliono individuare delle linee in un'immagine (*line detection*), è necessario definire più maschere in base alla direzione che caratterizza la linea da estrarre:

-1	-1	-1
2	2	2
-1	-1	-1

-1	-1	2
-1	2	-1
2	-1	-1

-1	2	-1
-1	2	-1
-1	2	-1

2	-1	-1
-1	2	-1
-1	-1	2

Provate ad applicare queste maschere all'immagine quadrato.jpg e confrontate le 4 mappe che evidenziano contorni orizzontali, verticali e obliqui (nelle due direzioni). Quindi effettuate l'elaborazione che vi permette di produrre un'unica mappa in cui sono presenti linee di qualsiasi tipo nell'immagine. Fate attenzione alla scelta della soglia e al fatto che le maschere così definite vi permettono di rilevare bordi spessi un pixel.

1.2 Edge detection

Le tecniche di edge detection si basano tipicamente sul calcolo della derivata prima o seconda per rilevare i bordi presenti in un'immagine. Prevedono di solito le seguenti operazioni:

1. smoothing, necessario dal momento che l'operazione di derivazione è molto sensibile al rumore presente in un'immagine;
2. individuazione degli edge point tramite il calcolo del modulo del gradiente dell'immagine oppure mediante valutazione degli zero-crossing del laplaciano;
3. thresholding (nel caso del gradiente) per produrre la mappa dei contorni.

1.2.1 Filtraggio basato sul gradiente

L'ampiezza del gradiente in un'immagine valuta quanto sono intense le variazioni di luminosità locali presenti in regioni definite dalle dimensioni della maschera utilizzata, ed è definita per un'immagine $x(m, n)$ come:

$$|\nabla \mathbf{x}| = \sqrt{\left(\frac{\partial x}{\partial m}\right)^2 + \left(\frac{\partial x}{\partial n}\right)^2}$$

Il calcolo richiede la conoscenza delle derivate direzionali, che possono essere valutate in diversi modi in base a come si definisce la derivata lungo una direzione. Se si usa la differenza prima si ottengono le maschere:

0	0
-1	1

-1	0
1	0

Mentre per la differenza centrale si ha:

0	0	0
-1	0	1
0	0	0

0	-1	0
0	0	0
0	1	0

Per cercare di ridurre la sensibilità al rumore è possibile definire le maschere in cui viene inglobata anche l'operazione di smoothing, così come accade per le maschere di *Prewitt*:

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

e per quelle di *Sobel*:

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Realizzate l'edge detection dell'immagine house.y di dimensioni 512×512 usando le maschere di Sobel per il calcolo del gradiente e scegliendo il valore della soglia pari a `T = 1.5*mean(grad(:))`. Visualizzate l'immagine originale, il gradiente e la mappa dei contorni (usate il nero per i contorni e il bianco per lo sfondo). Fate degli esperimenti cambiando le maschere e il valore della soglia. Provate a considerare anche le maschere che danno più importanza ai bordi lungo la direzione obliqua, come quelle di *Roberts*:

0	1
-1	0

1	0
0	-1

e di *Sobel*:

0	1	2
-1	0	1
-2	-1	0

-2	-1	0
-1	0	1
0	1	2

1.2.2 Smoothing

Applichiamo la procedura descritta nel paragrafo precedente all'immagine angiogramma.jpg, che rappresenta un angiogramma, ottenuto tramite l'introduzione di un catetere in un'arteria per raggiungere la zona in cui viene introdotto il mezzo di contrasto allo scopo di rilevare eventuali malformazioni nei vasi sanguigni. La mappa dei contorni ottenuta è particolarmente rumorosa, per cui potrebbe essere utile pre-elaborare l'immagine con un filtro di smoothing. A tale scopo consideriamo una maschera $k \times k$ che approssima un filtro gaussiano bidimensionale con deviazione standard σ:

```
k = 25; sigma = 4;
G = fspecial('gaussian',k,sigma);
```

Fate attenzione ai valori che scegliete per k e σ, infatti se si sceglie una gaussiana con elevata varianza sarà anche necessario estrarre più valori per rappresentarla, scegliendo quindi una maschera di dimensioni maggiori. Una regola empirica è quella di generare una maschera $k \times k$ con k pari al più piccolo valore intero maggiore o uguale di 6σ. Provate a realizzare lo smoothing dell'immagine anche con il filtro media aritmetica e confrontate i risultati con quello gaussiano, al variare della dimensione della maschera.

1.2.3 Laplaciano di una gaussiana

Le tecniche che si basano sulla derivata seconda fanno invece ricorso al laplaciano di un'immagine. Tuttavia esso presenta due problemi: è estremamente sensibile al rumore e crea bordi doppi, che rendono più difficile l'operazione di segmentazione. Per questo motivo si cerca di sfruttare di questo operatore la proprietà per cui i passaggi per lo zero (*zero-crossing*) del laplaciano sono in grado di individuare la posizione delle discontinuità. Un interessante esempio è il *Laplaciano di una Gaussiana* (LoG) in cui l'operatore laplaciano è combinato con lo smoothing.

In matlab mediante il seguente comando `mappa = edge(x,'log',thresh,sigma)` implementerete il laplaciano di una gausiana con deviazione standard σ e soglia pari a `thresh` all'immagine `x`. Applicate questo algoritmo all'immagine Angio.16bit.png. Noterete come i risultati dell'elaborazione presenti numerosi anelli chiusi (*spaghetti effect*), che è il principale svantaggio di questo metodo, oltre al fatto che in molte applicazioni i passaggi per lo zero vanno calcolati con un'accuratezza maggiore di quella che il metodo indicato può fornire.

1.2.4 Canny edge-detector

Per realizzare l'algoritmo di Canny è possibile usare sempre il comando `edge`:

```
mappa = edge(x,'canny',thresh,sigma)
```

In questo caso `thresh` è un vettore costituito da due elementi $[T_{low}T_{high}]$. Applicate il canny edge detector all'immagine casa.y (di dimensioni 512×512) con $\sigma = 1$, $T_{high} = 0.13$ e $\texttt{T}_{\texttt{low}} = \texttt{T}_{\texttt{high}}/4$. Confrontate la mappa dei contorni con quelle che si ottengono con le tecniche precedenti. Provate poi a modificare i parametri dell'algoritmo, come le dimensioni del filtro di smoothing, ovvero la sua deviazione standard, e i valori delle due soglie e osservate gli effetti sulla mappa prodotta. Testate l'algoritmo anche sull'immagine headCT.tif, (fate attenzione a settare opportunamente le due soglie).

1.3 Esercizi proposti

1. Si vuole effettuare l'edge detection dell'immagine van.tif. A tale scopo scrivete una funzione dal prototipo `function y = edge_detect1(x,T)` in cui valutate il gradiente dell'immagine mediante la maschera di Sobel, quindi generate la mappa dei contorni tramite un'operazione di thresholding con soglia `T` da determinare empiricamente.

   Scrivete una nuova funzione `function y = edge_detect2(x,T)` in cui applicate il Laplaciano di una Gaussiana, Infine, confrontate le mappe ottenute con le due strategie.

2. Data l'immagine dowels.tif, la si vuole segmentare in modo che nella mappa siano presenti solo i contorni degli oggetti chiari circolari. Scrivete il codice matlab che permette di raggiungere questo risultato.

2 Tecniche class-based

Le tecniche class-based, a differenza di quelle descritte in precedenza, si basano su un'elaborazione globale dell'immagine per realizzare la segmentazione. Gli approcci che esamineremo usano operazioni di thresholding o clustering per produrre la mappa di etichette, i cui valori identificano la regione associata ai pixel dell'immagine.

Data l'estrema semplicità di implementazione, la segmentazione basata su threshoding è molto utilizzata in varie applicazioni. Cominciamo col considerare un'immagine monocromatica costituita da un oggetto (foreground) su uno sfondo (background), in modo che i pixel dell'oggetto e quelli dello sfondo presentino livelli di intensità raggruppati in due modi dominanti. Un modo ovvio per estrarre gli oggetti dallo sfondo è quello di selezionare una soglia T che separi i due modi. Pertanto ogni pixel in posizione (m, n) per cui risulta $x(m, n) \geq T$ è classificato come *object point* altrimenti come *background point*. La mappa delle etichette, $y(m, n)$, si ottiene quindi nel modo seguente:

$$y(m,n) = \begin{cases} 1 & x(m,n) \geq T \\ 0 & x(m,n) < T \end{cases}$$

Quando T è costante si parla di thresholding globale; in tal caso un modo per selezionare il valore della soglia è quello di osservare l'istogramma dell'immagine oppure procedere in modo interattivo considerando diverse soglie fin quando il risultato diventa soddisfacente per l'osservatore. Una procedura automatica può essere realizzata applicando il seguente algoritmo:

1. selezionare una stima iniziale per T;
2. segmentare l'immagine usando T, in modo da produrre due gruppi di pixel: C_1, costituito da tutti i gruppi di pixel con valori di grigio $> T$ e C_2, costituito da tutti i gruppi di pixel con valori di grigio $\leq T$;
3. calcolare i valori medi μ_1 e μ_2 per i due gruppi;
4. calcolare una nuova soglia pari a $T = \frac{1}{2}(\mu_1 + \mu_2)$;
5. ripetere i passi 2,3 e 4 fin quando la differenza tra due valori della soglia in due iterazioni successive non sia minore di un parametro predefinito T_0.

In Matlab l'algoritmo si traduce nel seguente codice (soglia inizializzata al valor medio dei livelli di grigio):

```
Ti  = mean(X(:));           % inizializzazione della soglia
T0  = 0; diff = 1;
while (abs(diff) > T0)
   mask1 = (X > Ti);
   mask2 = (X <= Ti);
   C1 = X.*mask1;
   C2 = X.*mask2;
   mu1 = sum(C1(:))/sum(mask1(:));
   mu2 = sum(C2(:))/sum(mask2(:));
   Tf  = (mu1 + mu2)/2;
   diff = Tf - Ti;
   Ti = Tf;
end
```

Applicate questo algoritmo per determinare la soglia e poi applicate il thresholding all'impronta digitale impronta.jpg visualizzando il risultato della segmentazione.

Se l'immagine presenta più oggetti di interesse allora l'istogramma sarà multimodale e la procedura descritta sopra può essere generalizzata fissando K valori medi (se K è il numero di oggetti presenti) che permettono di realizzare la prima partizione, definendo le regioni di appartenenza dette anche *cluster*, e individuare i valori delle soglie. La procedura viene iterata fino a convergenza e prevede i seguenti passi:

1. selezionare casualmente un set di K medie;
2. per ogni pixel x_i calcolare la distanza (euclidea) dalle K medie:
$$D(x_i, m_k) = \|x_i - m_k\|^2 \qquad \forall k = 1, \cdots, K$$
3. assegnare il pixel x_i al cluster C_j cui corrisponde la media più vicina;
4. ricalcolare le medie per ogni cluster;
5. ripetere i passi 3 e 4 fino a convergenza.

Notate come nell'associazione valore pixel-etichetta non si tenga assolutamente in conto del contesto, cioè il fatto di avere a che fare con un'immagine piuttosto che con un vettore è ininfluente ai fini della segmentazione. L'algoritmo è anche noto come K-means e matlab lo implementa usando il comando `idx = kmeans(x,K)`, dove `x` è il vettore con i dati dell'immagine da elaborare, `K` il numero di classi, e `idx` il vettore contenente le etichette di ogni punto:

```
[M N] = size(x);
x = x(:);
idx = kmeans(x,K);
y = reshape(idx,M,N); imshow(y,[]);
```

Applicate questo algoritmo, al variare del numero di classi K, alle immagini rice.png e coins.png, che appartengono alla libreria di immagini di matlab.

L'algoritmo può poi essere generalizzato ulteriormente qualora si avesse a che fare con immagini a colori. In tal caso il clustering va effettuato sulla base dei vettori che determinano lo spazio di rappresentazione dei pixel

dell'immagine. I vettori rappresentativi di ogni pixel vengono suddivisi in K gruppi caratterizzati da K medie secondo il criterio a minimo errore quadratico medio. Applichiamo K-means utilizzando lo spazio di colori RGB. Leggiamo quindi un'immagine a colori e facciamo attenzione a fornire correttamente l'ingresso, che deve essere una matrice bidimensionale in cui ogni riga è la rappresentazione del colore dei pixel dell'immagine:

```
[M N L] = size(X);
for i = 1:L
    temp = X(:,:,i);
    x(:,i) = temp(:);
end
idx = kmeans(x,K);
y = reshape(idx,M,N); imshow(y,[]);
```

Tenete presente che matlab vi dà la possibilità di usare diversi tipi di distanze da minimizzare nello spazio L-dimensionale, oltre a quella euclidea (di default). Applicate l'algoritmo alle immagini a colori Fiori256.jpg e lenac.jpg, scegliendo opportunamente il valore da assegnare a K.

2.1 Esercizi proposti

1. *Thresholding locale*. Si vuole segmentare l'immagine yeast.tif; a tale scopo applicate l'algoritmo k-means su 3 classi e visualizzate la mappa ottenuta. Noterete come in questo modo sia possibile isolare le aree luminose dal background, tuttavia non risulti possibile individuare gli anelli grigio chiaro che circondano i cerchi bianchi. Per questo motivo si vuole segmentare l'immagine usando una soglia variabile basata sulle proprietà statistiche locali dell'immagine stessa. Scrivete la funzione `function mappa = thresholding_locale(x)` in cui:

   (a) si calcola l'immagine delle deviazioni standard locali, $\sigma_L(m,n)$, usando una finestra 3×3;

   (b) si ottiene la mappa binaria usando una soglia diversa per ogni posizione (m,n) come:

   $$g(m,n) = \begin{cases} 1 & x(m,n) > a\sigma_L(m,n) \ \text{ AND } \ x(m,n) > b\, m_G \\ 0 & \text{altrimenti} \end{cases}$$

   dove m_G è la media globale, e si pone $a = 30$ e $b = 1.5$.

   Si confrontino le mappe ottenute mediante i due approcci.

2. *Metodo di Otsu*. Si vuole segmentare su 2 classi mediante thresholding l'immagine molecole.tif avente dinamica $[0, K-1]$ e media m_G. Per determinare la soglia scrivete la funzione `function t1 = thresholding1(x)` che calcola empiricamente il valore della soglia globale seguendo un approccio di tipo iterativo. In realtà il thresholding globale dà un risultato poco soddisfacente a causa del fatto che come si può osservare dall'istogramma le due regioni risultano fortemente sovrapposte. Scrivete quindi una nuova funzione `function t2 = thresholding2(x)` in cui:

   (a) si calcola l'istogramma normalizzato $p(k)$ e quello cumulativo $P(k)$, $k = 1, \ldots, K$;

   (b) si calcolano le medie cumulative $m(k) = \sum_{i=0}^{k} i\, p(i), \ \ k = 1, \ldots, K$;

   (c) solo per i valori di k per cui risulta $0 < P(k) < 1$ si calcola la seguente quantità:

   $$\sigma_B^2(k) = \frac{[m_G P_(k) - m(k)]^2}{P(k)[1 - P(k)]} \quad k = 1, \ldots, K$$

   La soglia non è altro che il valore di k per cui σ_B^2 è massimo.

   Si confronti la mappa di segmentazione ottenuta mediante i due approcci.