Appunti di Elaborazione di Segnali Multimediali
a.a. 2017/2018

La segmentazione

L.Verdoliva

La segmentazione è il processo che permette di partizionare un'immagine in regioni (o oggetti) costituenti sulla base di specifici criteri. Il livello cui deve essere condotta la suddivisione dipende dal problema da risolvere, vale a dire il processo termina non appena si riescono ad isolare gli oggetti di interesse. Ottenere una buona segmentazione dell'immagine è uno dei compiti più difficili dell'image processing e la sua accuratezza influenza considerevolmente l'eventuale successo o fallimento di procedure di analisi computerizzata che ne fanno uso. La segmentazione può fornire in uscita la mappa dei contorni, i cui elementi non nulli identificano i contorni tra gli oggetti oppure la mappa delle etichette, i cui elementi sono in corrispondenza univoca con i pixel dell'immagine e ne identificano la regione di appartenenza.

Data l'eterogeneità delle possibili applicazioni, non esiste una tecnica di segmentazione che sia universalmente buona. Esistono però concetti di base del trattamento di immagini che sono comuni a molti approcci. Di fatto, gli algoritmi di segmentazione si basano su due proprietà fondamentali delle immagini: discontinuità e similarità. Nella prima categoria ricadono quelle tecniche che realizzano la partizione basandosi su repentini cambiamenti della luminosità, mentre nel secondo caso l'approccio è quello di suddividere l'immagine in regioni che sono simili sulla base di un fissato criterio.

Questa trattazione si limiterà alla solo analisi di alcune fra le tecniche edge-based e class-based. Le tecniche *edge-based* si basano sull'idea che nel passare da una regione d'interesse ad un'altra si riscontrano rapidi cambiamenti nei valori di intensità. Lo scopo dunque è il rilevamento (e la classificazione) delle discontinuità (*edges*) dell'immagine (livelli di grigio). Invece, le tecniche *class-based* che esamineremo si basano sull'analisi dell'intera immagine per raggruppare i pixel attraverso operazioni di thresholding o clustering. Le prime producono la mappa dei contorni, mentre le seconde la mappa delle etichette.

1 Tecniche edge-based

Questo approccio sfrutta il fatto che gli oggetti che costituiscono un'immagine sono caratterizzati da valori di intensità molto diversi, per cui la capacità di rilevare le discontinuità equivale alla possibilità di individuare gli oggetti che la compongono. Un modo molto semplice per rilevare i bordi è attraverso un filtraggio spaziale in cui si assegnano opportunamente i coefficienti della maschera. Questa operazione è poi seguita da un processo di thresholding, che permette di produrre una mappa binaria in cui i contorni sono identificati solo dai pixel i cui valori di intensità superano una prefissata soglia T. La mappa così ottenuta spesso è caratterizzata da

Figura 1: Diagramma a blocchi di un generico schema di edge detection

contorni piuttosto spessi, per cui tipicamente si applica una procedura per renderli più sottili (thinning). In base a come si definisce la maschera è possibile individuare:

1. punti isolati (*point detection*);
2. linee (*line detection*);
3. bordi di qualsiasi tipo (*edge detection*).

Nel seguito tratteremo separatamente queste elaborazioni.

1.1 Point Detection

Cominciamo col considerare il problema di rilevare punti isolati in un'immagine, che definiamo come quei pixel il cui livello di grigio è significativamente diverso dallo sfondo, tipicamente omogeneo, in cui sono inseriti. In tal caso si può usare la seguente maschera:

$$h(m,n) = \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

L'idea che è alla base di questo approccio è che un punto isolato assumerà un valore molto diverso dai pixel circostanti, per cui quando la maschera è centrata proprio sul punto da rilevare produrrà una risposta molto grande, tale risposta sarà invece nulla in aree di intensità costante, dato che la somma di tutti i coefficienti della maschera è pari a zero.

Applichiamo questa procedura all'immagine di figura 2, che mostra un'immagine a raggi X in cui è presente una porosità in alto a destra, dove si trova un singolo pixel nero nella posizione con coordinate (446, 250). Scegliendo la soglia pari al 90% del livello di grigio più grande in valore assoluto, è possibile rilevare il punto isolato.

1.2 Line Detection

Per individuare le linee in un'immagine, usando la stessa strategia vista nel paragrafo precedente, basta modificare la maschera. In particolare, si possono definire più maschere in base alla direzione che caratterizza la linea da estrarre:

$$h_1(m,n) = \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline 2 & 2 & 2 \\ \hline -1 & -1 & -1 \\ \hline \end{array} \qquad h_2(m,n) = \begin{array}{|c|c|c|}\hline -1 & -1 & 2 \\ \hline -1 & 2 & -1 \\ \hline 2 & -1 & -1 \\ \hline \end{array}$$

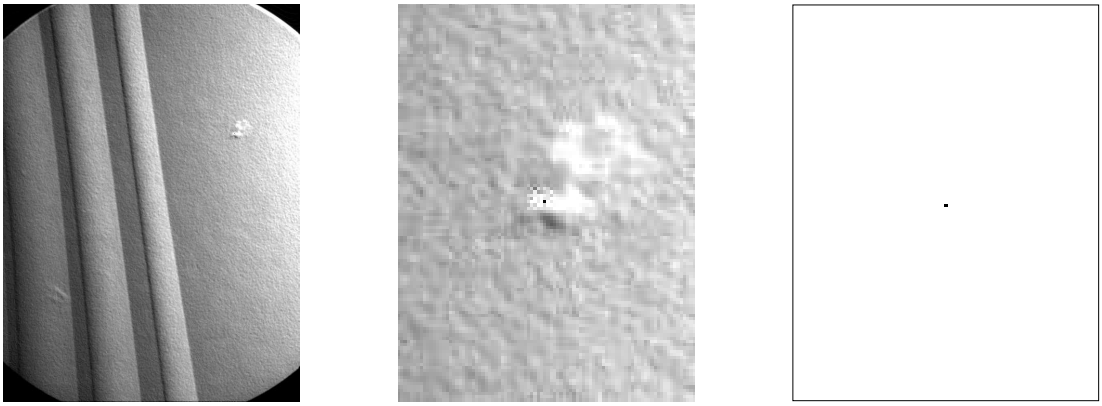

Figura 2: Immagine da elaborare, zoom della regione in cui è presente il punto isolato e mappa relativa.

$$h_3(m,n) = \begin{array}{|c|c|c|}\hline -1 & 2 & -1 \\ \hline -1 & 2 & -1 \\ \hline -1 & 2 & -1 \\ \hline \end{array} \qquad h_4(m,n) = \begin{array}{|c|c|c|}\hline 2 & -1 & -1 \\ \hline -1 & 2 & -1 \\ \hline -1 & -1 & 2 \\ \hline \end{array}$$

Se l'immagine viene filtrata separatamente con queste 4 maschere si otterranno 4 immagini, indichiamole con $y_1(m,n)$, $y_2(m,n)$, $y_3(m,n)$ e $y_4(m,n)$, ognuna delle quali è caratterizzata da bordi nella direzione orizzontale, diagonale ($+45°$), verticale e diagonale ($-45°$), rispettivamente. Se, per esempio, per il punto relativo alla posizione m_0, n_0 della prima immagine $y_1(m_0, n_0)$ risulta:

$$|y_1(m_0, n_0)| > |y_i(m_0, n_0)| \qquad i = 2, 3, 4$$

allora il pixel in posizione (m_0, n_0) è associato ad una linea orizzontale. Si noti come la direzione che si vuole estrarre sia pesata con un coefficiente più grande rispetto alle altre direzioni. Per esempio, supponiamo di usare la prima maschera per la seguente sezione di immagine:

$$x(m,n) = \begin{array}{|c|c|c|c|c|}\hline 1 & 1 & 1 & 1 & 1 \\ \hline 5 & 5 & 5 & 5 & 5 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline \end{array}$$

L'uscita presenta la massima risposta laddove sono presenti linee, di spessore pari a un pixel, orientate orizzontalmente:

$$y(m,n) = \begin{array}{|c|c|c|c|c|}\hline -6 & -9 & -9 & -9 & -6 \\ \hline 16 & 24 & 24 & 24 & 16 \\ \hline -6 & -9 & -9 & -9 & -6 \\ \hline \end{array}$$

Consideriamo come esempio l'immagine di figura 3, se siamo interessati ad individuare tutte le linee orientate a $-45°$ bisognerà filtrare l'immagine usando l'ultima delle maschere indicate. Dallo zoom (figura 4) si può notare quanto sia più forte la risposta del segmento obliquo in basso a destra piuttosto che quello in alto a sinistra e questo e legato al fatto che la componente in basso è spessa esattamente un pixel.

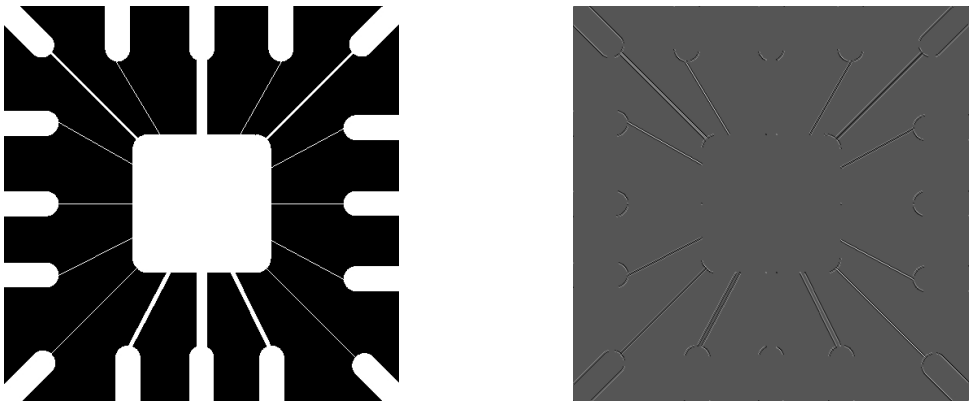

Figura 3: Immagine originale e filtrata con la maschera $h_4(m, n)$.

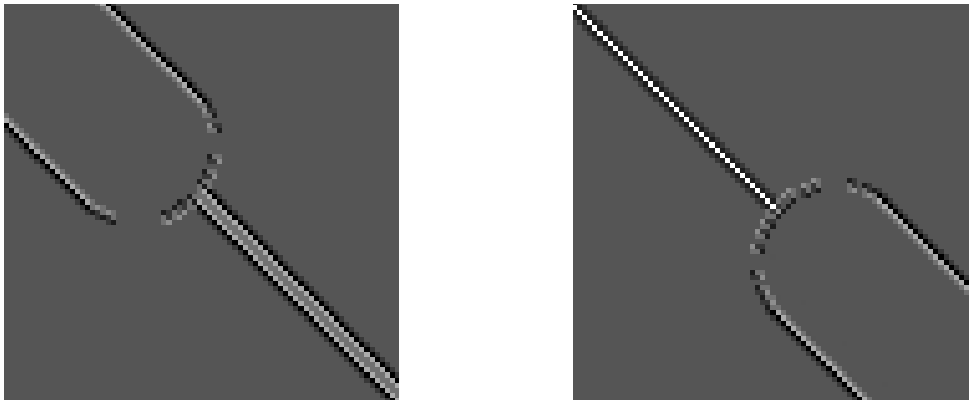

Figura 4: Zoom dell'immagine filtrata nelle regioni in cui sono presenti linee oblique.

Se si vogliono determinare quali sono i punti che producono valori più elevati, basta effettuare ancora una volta un'operazione di thresholding in cui la soglia è scelta come il valore massimo presente nell'immagine originale. In questo modo otteniamo un'immagine caratterizzata da linee di spessore pari proprio a un pixel orientate nella direzione $-45°$. In realtà nell'immagine (figura 5) sono anche presenti punti isolati, che possono essere rivelati con la tecnica descritta in precedenza e poi eliminati.

1.3 Edge Detection

Sebbene il rilevamento di punti e linee sia molto importante nei processi di segmentazione, l'*edge detection* rappresenta di gran lunga l'approccio più comune per individuare le discontinuità nei livelli di grigio in un'immagine. Per affrontare e risolvere correttamente il problema è necessario definire un bordo. In figura 6 a sinistra è mostrata una regione in cui è presente una transizione ideale tra due livelli di luminosità e il corrispondente profilo di una sua sezione (*step edge*). Nella pratica, a causa di imprecisioni nell'acquisizione dell'immagine, il bordo non presenta una netta transizione tra i livelli di grigio, ma è caratterizzato da una variazione graduale che può essere ben approssimata da un modello a rampa (figura 6 a destra).

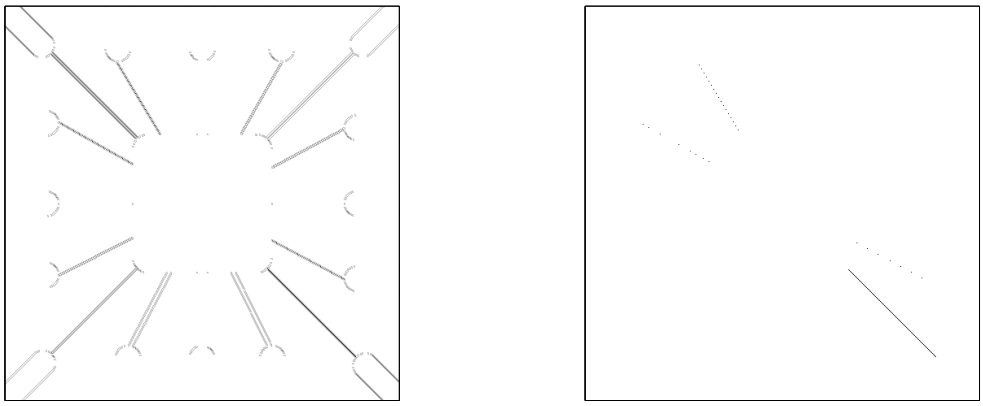

Figura 5: Valore assoluto dell'immagine filtrata e mappa binaria ottenuta dopo il thresholding.

Se il bordo presentasse una netta transizione, l'operazione che ne consentirebbe la rivelazione sarebbe la derivazione. In figura 7 (a sinistra) si mostrano la derivata prima e la derivata seconda nel caso di edge a rampa: la derivata prima risulta costante per i punti appartenenti alla rampa, mentre è zero nelle aree costanti, quindi la sua ampiezza può essere usata per stabilire se un pixel dell'immagine è associato ad un bordo; la derivata seconda, invece, fornisce due valori in corrispondenza di ogni bordo (doppio bordo), Nella pratica il profilo di un edge a rampa è simile a quello mostrato in figura 7 a destra. In tal caso la derivata prima assume valore massimo intorno al centro del bordo per cui dopo averla calcolata si può usare un'operazione di thresholding per valutarne i massimi locali. La derivata seconda presenta, invece, l'interessante proprietà di attraversare lo zero intorno al punto centrale del bordo. Questa proprietà detta di *zero-crossing* può risultare molto utile per localizzare il centro di bordi spessi.

E' molto importante poi fare un'ulteriore considerazione. Spesso si ha a che fare con immagini rumorose, e anche se il rumore risulta quasi impercettibile nell'immagine, le operazioni di derivazione sono estremamente sensibili alla sua presenza. In figura 8 si mostrano due esempi in cui alla sezione di immagine di figura 7 è stato aggiunto rumore gaussiano, si può notare come, anche se dall'immagine e dal profilo difficilmente si riesca a percepire la presenza di rumore, la derivata prima e soprattutto la derivata seconda risultino fortemente distorte. Per questo

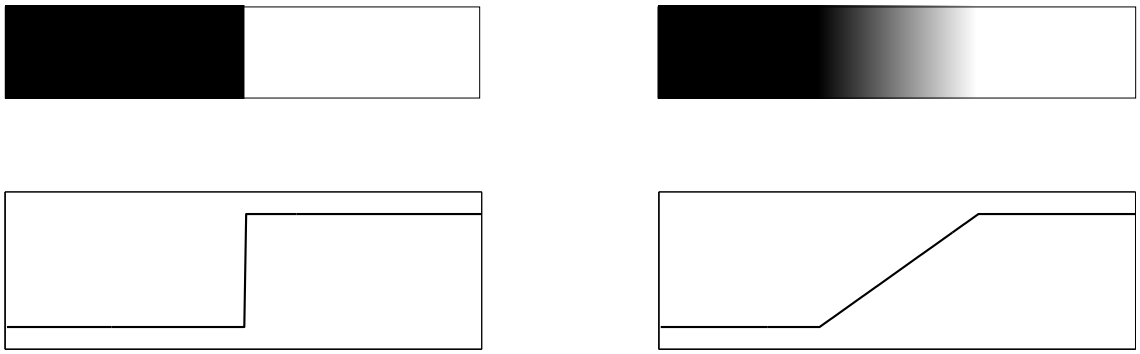

Figura 6: Bordo ideale e modello a rampa.

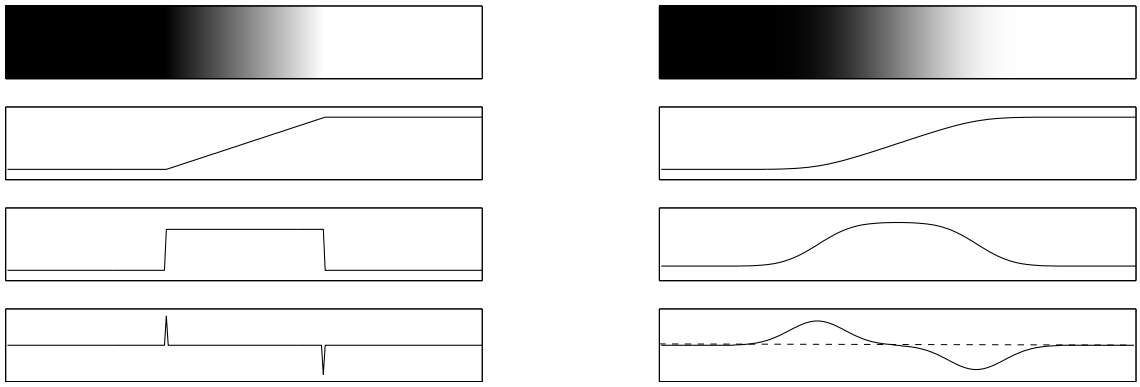

Figura 7: Derivata prima e seconda del bordo schematizzato con un modello a rampa.

motivo è necessario effettuare lo smoothing delle immagini prima di applicare l'operazione di derivazione. Esiste però un trade-off tra la rivelazione corretta del bordo e l'individuazione della sua posizione. Infatti, per migliorare l'effetto del filtraggio del rumore, e quindi ridurre la probabilità di errata classificazione, è necessario aumentare la dimensione della maschera associata al filtro, d'altra parte in questo modo peggiora la capacità di localizzare perfettamente il bordo.

In conclusione per classificare un pixel dell'immagine come un bordo, la transizione su scala di grigi associata a quel punto deve essere molto più grande dello sfondo. Diremo allora che un pixel dell'immagine è un *edge point* se la derivata prima in quel punto è superiore ad una soglia fissata. Un insieme connesso di tali punti è un bordo (*edge*). In alternativa possiamo definire gli edge point come i passaggi per lo zero della derivata seconda. E' bene tener presente che queste definizioni non garantiscono il successo nell'individuazione dei bordi in un'immagine. Il risultato in entrambi i casi è una mappa di contorni (*edge map*).

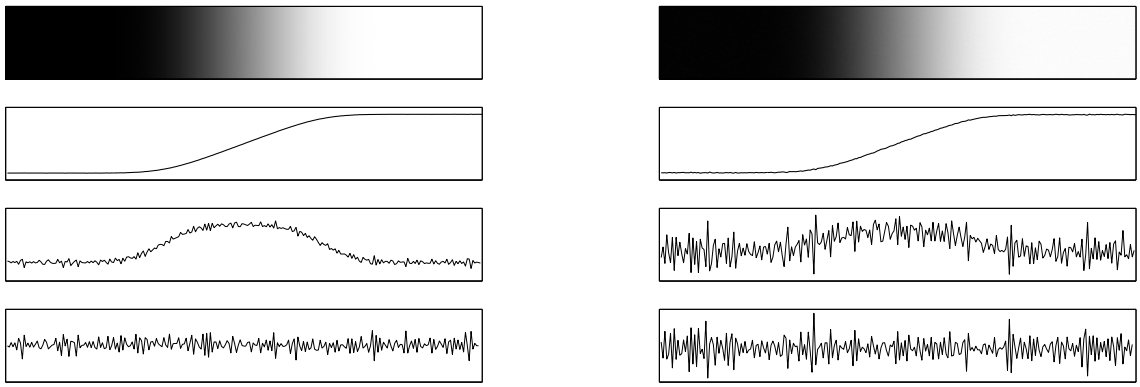

Figura 8: Esempi in cui è stato aggiunto rumore gaussiano.

1.3.1 Tecniche che si basano sulla derivata prima

Ricordiamo che il calcolo della derivata prima di un'immagine digitale si basa su approssimazioni del gradiente bidimensionale, definito per un'immagine $x(m,n)$ come il vettore

$$\nabla \mathbf{x} = \frac{\partial x}{\partial m}\mathbf{i}_m + \frac{\partial x}{\partial n}\mathbf{i}_n$$

la cui ampiezza è:

$$|\nabla \mathbf{x}| = \nabla x = \sqrt{\left(\frac{\partial x}{\partial m}\right)^2 + \left(\frac{\partial x}{\partial n}\right)^2} \tag{1}$$

Per semplificare il calcolo questa quantità è approssimata usando i valori assoluti come:

$$\nabla x \simeq \left|\frac{\partial x}{\partial m}\right| + \left|\frac{\partial x}{\partial n}\right|$$

Questa approssimazione ancora si comporta come una derivata, cioè è nulla in aree di intensità costante e i valori risultano proporzionali al livello di variazione di intensità nelle aree non omogenee. Nella pratica è comune chiamare *gradiente* l'ampiezza del gradiente o una sua approssimazione. Si definisce invece direzione del vettore gradiente $\nabla \mathbf{x}$ nel punto (m,n) la quantità:

$$\alpha(\nabla \mathbf{x}) = \tan^{-1}\left(\frac{\partial x/\partial n}{\partial x/\partial m}\right) \tag{2}$$

La direzione di un bordo nel punto (m,n) è perpendicolare alla direzione del vettore gradiente in quel punto, è infatti la direzione di massima variazione dell'intensità. Consideriamo allora un'area 3×3 che rappresenta la sezione dell'immagine che viene elaborata:

$$\begin{bmatrix} x(m-1,n-1) & x(m-1,n) & x(m-1,n+1) \\ x(m,n-1) & x(m,n) & x(m,n+1) \\ x(m+1,n-1) & x(m+1,n) & x(m+1,n+1) \end{bmatrix}$$

Esistono molti possibili filtri che approssimano la derivata se si vuole stimare il gradiente. Cominciamo con il caso più semplice. Due semplici schemi per la derivata lungo la direzione orizzontale sono la differenza prima e la differenza centrale:

$$\frac{\partial x}{\partial m} = x(m,n) - x(m-1,n) \tag{3}$$

$$\frac{\partial x}{\partial m} = x(m+1,n) - x(m-1,n) \tag{4}$$

Entrambe queste derivate rispondono fortemente a bordi orientati lungo la direzione verticale e forniscono risposta nulla se il bordo è perfettamente orizzontale. Stesso discorso vale per derivate lungo la direzione verticale. Queste derivate possono essere espresse mediante le maschere dei filtri che le implementano nel seguente modo per la differenza prima:

$h_1(m,n) =$

0	0
1	-1

$h_2(m,n) =$

-1	0
1	0


Figura 9: Immagine originale e immagine gradiente.

Mentre per la differenza centrale si ha:

$$h_1(m,n) = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 1 & \mathbf{0} & -1 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \qquad h_2(m,n) = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline 0 & \mathbf{0} & 0 \\ \hline 0 & 1 & 0 \\ \hline \end{array}$$

Il punto in grassetto indica la posizione dell'origine. In figura 9 a destra si mostra il gradiente dell'immagine della casa raffigurata a sinistra. Si può osservare come il gradiente assume valori elevati laddove sono presenti brusche variazioni nei livelli di grigio.

Per quanto detto nel paragrafo precedente la difficoltà nell'uso di questi filtri è l'estrema sensibilità al rumore, questo può essere ridotto però inglobando l'operazione di smoothing in ognuno dei filtri. Consideriamo per esempio la differenza centrale in una direzione e un filtro media aritmetica su 3 campioni nella direzione ortogonale. Definiamo allora le seguenti risposte impulsive:

$$h_a(m) = [1\ 1\ 1] \qquad h_b(n) = [-1\ 0\ 1]$$

Poiché h_a dipende solo da m e h_b da n, facendo il prodotto matriciale tra i due possiamo individuare un filtro bidimensionale separabile che effettua la derivata, ma realizza anche lo smoothing:

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Si ottengono in questo modo le maschere di *Prewitt*:

$$h_1(m,n) = \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \qquad h_2(m,n) = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

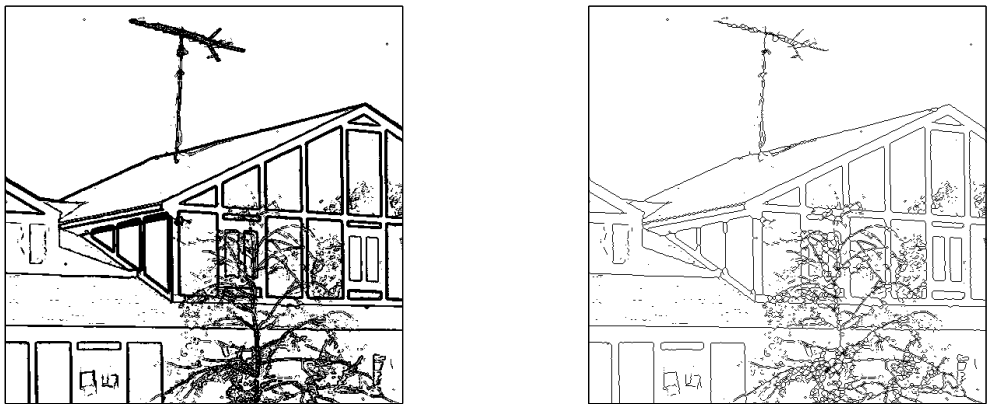

Figura 10: Mappa dei contorni ottenuta dopo il thresholding e dopo un'operazione di thinning.

Se modifichiamo i pesi del filtro di smoothing, otteniamo uno dei più usati rivelatori di bordi, l'operatore di *Sobel*:

$$h_1(m,n) = \begin{array}{|c|c|c|}\hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \qquad h_2(m,n) = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

Si noti come la somma di tutti i coefficienti delle maschere sia sempre pari a zero, infatti l'operatore di derivata deve fornire un contributo nullo per aree con luminosità costante. Queste maschere possono essere modificate in modo da dare una risposta maggiore lungo la direzione diagonale per esempio per le maschere di Sobel si ha:

$$h_1(m,n) = \begin{array}{|c|c|c|}\hline 0 & 1 & 2 \\ \hline -1 & 0 & 1 \\ \hline -2 & -1 & 0 \\ \hline \end{array} \qquad h_2(m,n) = \begin{array}{|c|c|c|}\hline -2 & -1 & 0 \\ \hline -1 & 0 & 1 \\ \hline 0 & 1 & 2 \\ \hline \end{array}$$

Il calcolo del gradiente (o meglio della sua ampiezza) permette quindi di stabilire per ogni pixel dell'immagine se sono presenti bordi valutando i massimi locali attraverso il confronto con una soglia T, tipicamente prestabilita. Se

$$|\nabla x(m_0, n_0)| \geq T$$

allora il pixel nella posizione (m_0, n_0) è classificato come *edge point*. Analizzando la mappa così ottenuta si può notare che l'insieme dei punti selezionati formano bordi piuttosto spessi (figura 10 a sinistra), anziché contorni di ampiezza nulla. Per questo motivo il thresholding viene fatto seguire da un'operazione di *thinning* che ha proprio l'obiettivo di assottigliare i bordi (figura 10 a destra). In effetti quest'ultima operazione serve proprio a stabilire con più precisione se il pixel è effettivamente un massimo locale. La scelta della soglia è il risultato di un compromesso: una soglia elevata tende a rimuovere eventuale rumore, ma potrebbe anche eliminare dei pixel che appartengono ad un bordo per cui la mappa risulta essere caratterizzata da contorni più frammentati, di contro una soglia bassa potrebbe creare falsi edge point.

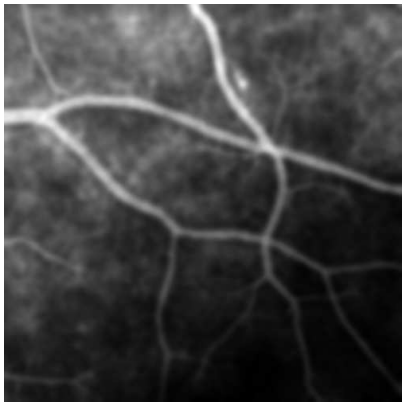
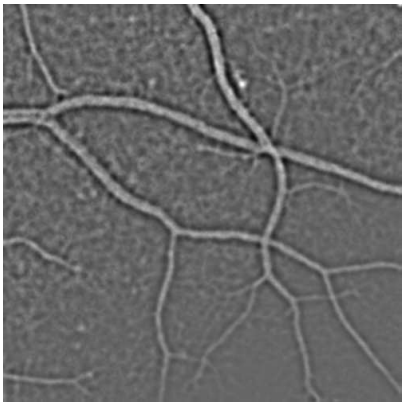

Figura 11: Angiogramma e immagine del Laplaciano.

1.3.2 Tecniche che si basano sulla derivata seconda

La derivata seconda di un'immagine viene invece calcolata usando l'operatore Laplaciano definito come

$$\nabla^2 x = \frac{\partial^2 x}{\partial m^2} + \frac{\partial^2 x}{\partial n^2}$$

Ricordiamo che una possibile maschera che lo implementa (e che usa la differenza prima per calcolare la derivata) è la seguente:

$$h(m,n) = \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline -1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}$$

Ovviamente è possibile ottenere altri tipi di maschere se si modifica la definizione della derivata prima. In figura 11 si mostra il laplaciano di un'angiogramma.

Per ridurre la sensibilità al rumore della derivata seconda si usa un filtro di smoothing, prima di valutare i passaggi per lo zero che individuano le posizioni dei bordi.

1.3.3 Marr-Hildreth edge-detector

Un interessante esempio di tecnica basata sul laplaciano è il Marr-Hildreth edge detector, in cui l'operatore laplaciano è combinato con lo smoothing di tipo gaussiano per ridurre il rumore. La funzione gaussiana ha una serie di interessanti proprietà: ha un andamento dolce ed è ben localizzata sia nello spazio che in frequenza, fornendo così un buon compromesso tra la necessità di evitare edge falsi e minimizzare gli errori nella posizione degli stessi. Un filtro gaussiano ha la seguente espressione:

$$g(m,n) = \exp\left(-\frac{m^2+n^2}{2\sigma^2}\right)$$

dove σ è la deviazione standard. Questo filtro crea un effetto di blurring tanto più marcato quanto più alta è σ. L'algoritmo prevede di realizzare prima la convoluzione con un filtro gaussiano e poi calcolare il Laplaciano, cioè:

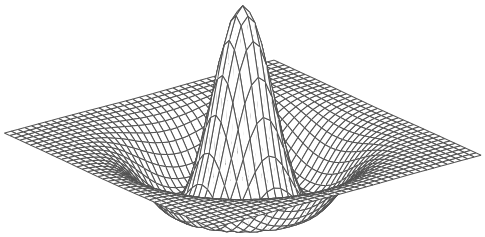

Figura 12: Laplaciano di una gaussiana.

1. $y(m,n) = x(m,n) * g(m,n)$;
2. $\nabla^2 y(m,n) = \frac{\partial^2 y}{\partial m^2} + \frac{\partial^2 y}{\partial m^2}$

D'altra parte poiché la derivata seconda è un operatore lineare risulta:

$$\nabla^2 y(m,n) = \nabla^2(x(m,n) * h(m,n)) = x(m,n) * \nabla^2 g(m,n)$$

ma allora anziché filtrare l'immagine e poi calcolarne il laplaciano, è possibile effettuare la convoluzione dell'immagine con il Laplaciano della funzione gaussiana (Laplacian of Gaussian, LoG), che può essere calcolato una volta per tutte e non dipende dall'immagine in ingresso. Il laplaciano di una gaussiana è anche chiamato *Mexican hat* a causa della sua forma (figura 12) ed ha la seguente espressione analitica:

$$\begin{aligned}
\nabla^2 g(m,n) = \frac{\partial^2 g}{\partial m^2} + \frac{\partial^2 g}{\partial n^2} &= \frac{\partial}{\partial m}\left[-\frac{m}{\sigma^2}\, e^{-\frac{m^2+n^2}{\sigma^2}}\right] + \frac{\partial}{\partial n}\left[-\frac{n}{\sigma^2}\, e^{-\frac{m^2+n^2}{\sigma^2}}\right] \\
&= \left[\frac{m^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{m^2+n^2}{\sigma^2}} + \left[\frac{n^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{m^2+n^2}{\sigma^2}} \\
&= \left[\frac{m^2+n^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{m^2+n^2}{\sigma^2}}
\end{aligned}$$

In figura 13 si mostra il LoG dell'immagine di figura 12 e i passaggi per lo zero relativi. Dalla figura si può notare come i bordi abbiano spessore nullo, quindi non è necessario effettuare il thinning, ma dal momento che anche una lieve transizione di intensità produce uno zero-crossing, l'immagine è caratterizzata da numerosi contorni chiusi (*spaghetti effect*).

1.3.4 Canny edge-detector

Il Canny edge detector è un metodo molto potente per la rivelazione dei bordi in un'immagine e cerca di migliorare il risultato dell'elaborazione sostituendo all'operazione di thresholding i due seguenti passi:

- rendere più sottili (*thinning*) i contorni direttamente sull'immagine gradiente attraverso una procedura nota come *Nonmaxima Suppression*;
- applicare un'operazione di thresholding con doppia soglia (*Hysteresis Thresholding*).

In totale i passi da realizzare sono i seguenti:

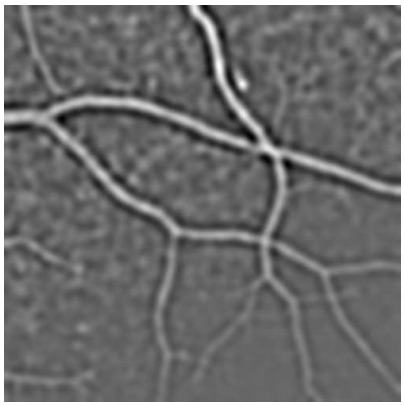
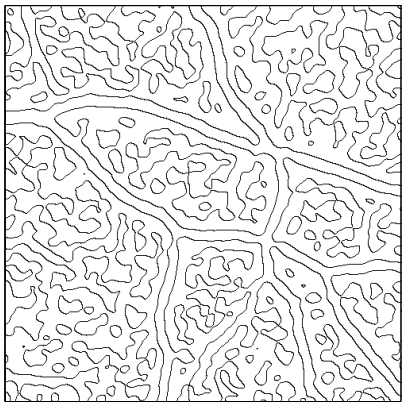

Figura 13: LoG e zero crossing dell'immagine di figura 12.

1. l'immagine viene filtrata con un filtro gaussiano per ridurre il rumore;
2. per ogni pixel si calcola il gradiente usando per la valutazione delle derivate direzionali un filtro che è la derivata direzionale di una gaussiana[1].
3. per ogni pixel si stima la direzione del gradiente, e lo si classifica come edge point solo se lungo tale direzione il gradiente è maggiore di quello relativo ai due pixel adiacenti, altrimenti il valore del pixel è posto a zero (*Nonmaxima suppression*).
4. si usa una doppia soglia allo scopo di eliminare i pixel che presentano un gradiente minore della soglia bassa (dovuto probabilmente a rumore) e conservare quelli che superano la soglia alta (classificati come edge point). I pixel che presentano un gradiente compreso tra le due soglie vengono conservati solo se vicini ai pixel che hanno superato la soglia alta.

Data una funzione gaussiana

$$g(m,n) = \exp\left(-\frac{m^2+n^2}{2\sigma^2}\right)$$

le derivate lungo la direzione orizzontale e verticale di $g(m,n)$ sono:

$$\begin{cases} \frac{\partial g}{\partial m} = -\frac{m}{\sigma^2}\ \exp\left(-\frac{m^2+n^2}{2\sigma^2}\right) \\ \frac{\partial g}{\partial m} = -\frac{n}{\sigma^2}\ \exp\left(-\frac{m^2+n^2}{2\sigma^2}\right) \end{cases}$$

La maschera del filtro relativo alla derivata della gaussiana lungo la direzione orizzontale servirà a calcolare $\partial x/\partial m$, quello lungo la direzione verticale $\partial x/\partial n$. Quest'ultima maschera si può calcolare trasponendo la prima, data la completa simmetria esistente tra le espressioni delle due derivate. A questo punto bisogna realizzare l'algoritmo nonmaxima suppression, cioè annullare i valori che non sono localmente dei massimi, confrontandoli con i vicini nella direzione del gradiente. In particolare si tenga presente che intorno ad ogni pixel sono presenti 8 vicini,

[1]Nell'articolo pubblicato nel 1986, *a computational approach to edge detection*, Canny ha dimostrato che nell'ipotesi di considerare un bordo ideale corrotto da rumore gaussiano il filtro rivelatore ottimo può essere approssimato dalla derivata di una gaussiana.

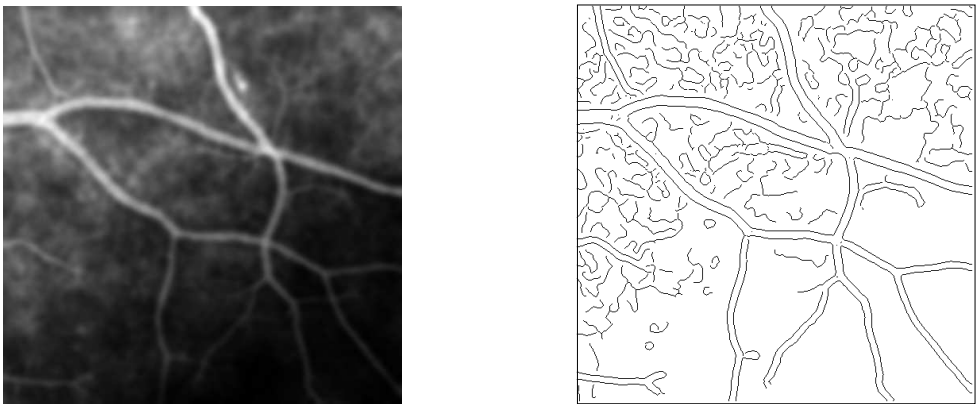

Figura 14: Angiogramma e mappa binaria ottenuta con l'algoritmo di Canny.

quindi 4 possibili direzioni, per ognuna delle quali è necessario realizzare l'algoritmo nonmaxima suppression. Per l'hysteresis thresholding è necessario fissare due soglie: T_{high} e T_{low}. Si stabilisce quindi se un pixel appartiene ad un edge oppure no adottando la seguente strategia: se il relativo valore del gradiente

- supera la soglia T_{high}, allora è classificato come un bordo (edge);
- minore della soglia T_{low}, viene annullato;
- se è compreso tra T_{low} e T_{high}, viene conservato solo se è vicino a pixel classificati come edge.

In figura 14 si mostra ancora una volta l'angiogramma e la mappa binaria ottenuta con l'algoritmo di Canny.

2 Tecniche class-based

Le tecniche class-based, a differenza di quelle esaminate nel paragrafo precedente si basano su un'elaborazione globale dell'immagine per realizzare la segmentazione. Inoltre, in uscita producono una mappa di etichette, ognuna delle quali è associata ad una regione.

2.1 Thresholding

Un esempio di segmentazione mediante thresholding è quello che fa uso dell'istogramma. Questa tecnica funziona se l'immagine ha caratteristiche molto semplici, cioè è costituita da uno sfondo (background) e un oggetto (foreground) e il relativo istogramma è di tipo bimodale. Supponiamo per esempio di considerare l'immagine $x(m,n)$ che mostra un'impronta digitale e visualizziamo il relativo istogramma (figura 15). Quest'ultimo rileva come i pixel dell'immagine presentino i livelli di grigio raggruppati in due modi dominanti. Un modo molto semplice per estrarre l'impronta dallo sfondo è quello di selezionare una soglia opportuna che li separi e poi ricavare la mappa come

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

Dalla figura si può determinare empiricamente il valore della soglia T, pari circa a 125. In questo modo viene prodotta la mappa binaria mostrata in figura 16 (a sinistra). Tuttavia se all'immagine si sovrappone rumore gaussiano l'istogramma non presenta più i due modi divisi perfettamente (figura 17), ma è presente un certo livello di sovrapposizione. In tal caso la scelta empirica della soglia si sposta al valore T pari a 150 e si ottiene la mappa binaria mostrata in figura 16 (a destra). Evidentemente non è sempre possibile procedere empiricamente a meno che non si operi in modo interattivo, selezionando la soglia fino a quando non si ottiene un risultato soddisfacente.

In situazioni più critiche un modo per selezionare la soglia è mediante l'utilizzo di un modello. L'istogramma dell'immagine si può interpretare infatti come una distribuzione di probabilità, che nel nostro caso si può modellare come una mistura di due densità, una per il background e l'altra per il foreground. Detta X la variabile aleatoria associata al generico pixel dell'immagine, la distribuzione di probabilità $p_X(x)$ si può esprimere come:

$$p_X(x) = P_1\, p_{X|1}(x|1) + P_2\, p_{X|2}(x|2)$$

dove $p_1 = \Pr(X \in C_1)$ è la probabilità che un pixel appartenga al foreground (classe 1), mentre $p_2 = \Pr(X \in C_2)$ che appartenga al background (classe 2); $p_{X|1}$ è la densità di probabilità associata ai pixel che appartengono al foreground, mentre $p_{X|2}$ quella associata ai pixel del background. Il valore della soglia può essere determinato minimizzando la probabilità di errore, cioè la probabilità di classificare un pixel appartenente allo sfondo come oggetto e viceversa:

$$\begin{aligned}\Pr(E) &= \Pr(\{X > T, X \in C_1\}) + \Pr(\{X < T, X \in C_2\}) \\ &= P_1 \Pr(X > T | X \in C_1) + P_2 \Pr(X < T | X \in C_2) \\ &= P_1 \int_T^{+\infty} p_{X|1}(x|1)\, dx + P_2 \int_{-\infty}^{T} p_{X|2}(x|2)\, dx\end{aligned}$$

Minimizzando $\Pr(E)$ rispetto a T si ha:

$$\frac{d\Pr(E)}{dT} = -P_1\, p_{X|1}(T|1) + P_2\, p_{X|2}(T|2) = 0 \tag{5}$$

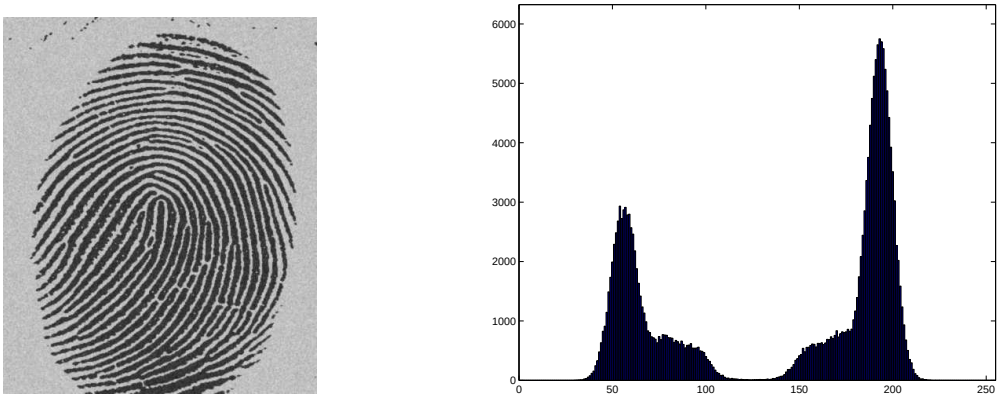

Figura 15: Immagine e relativo istogramma.

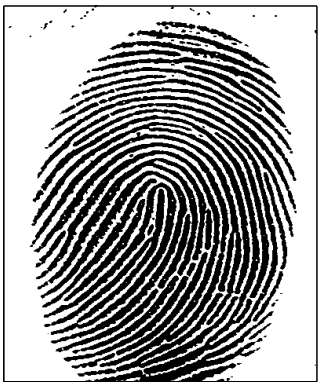


Figura 16: Mappa binaria ottenuta dall'immagine di figura 14 (a sinistra) e dall'immagine di figura 16 (a destra) scegliendo la soglia dall'istogramma.

Per determinare il valore della soglia è necessario conoscere le distribuzioni di probabilità condizionate. Nell'ipotesi di considerare un modello gaussiano, allora $p_{X|1}(x|1) \sim \mathcal{N}(\mu_1, \sigma^2)$ e $p_{X|2}(x|2) \sim \mathcal{N}(\mu_2, \sigma^2)$, per cui la (5) diventa:

$$P_2 \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(T-\mu_2)^2}{2\sigma^2}} = P_1 \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(T-\mu_1)^2}{2\sigma^2}}$$

cioè

$$\begin{aligned} \ln\left(\frac{P_2}{P_1}\right) &= \frac{-(T-\mu_1)^2 + (T-\mu_2)^2}{2\sigma^2} \\ 2\sigma^2 \ln\left(\frac{P_2}{P_1}\right) &= -T^2 - \mu_1^2 + 2T\mu_1 + T^2 + \mu_2^2 - 2T\mu_2 \\ &= 2T(\mu_1 - \mu_2) - (\mu_1 - \mu_2)(\mu_1 + \mu_2) \end{aligned}$$


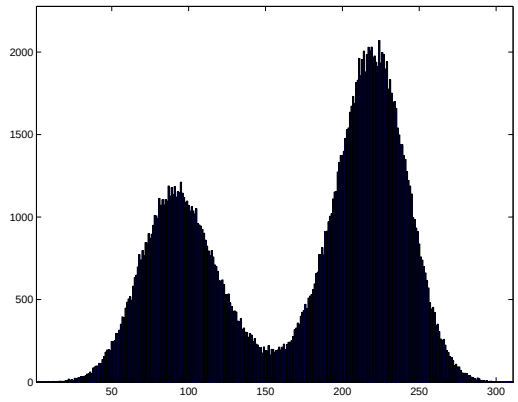

Figura 17: Immagine rumorosa e relativo istogramma.

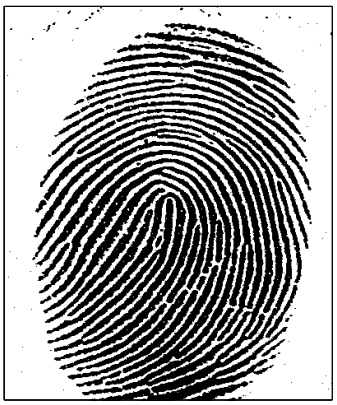

Figura 18: Mappa binaria dell'immagine di figura 16 ottenuta con l'algoritmo K-means.

quindi la soglia ha la seguente espressione analitica:

$$T = \frac{\mu_1 + \mu_2}{2} - \frac{\sigma^2}{\mu_2 - \mu_1} \ln\left(\frac{P_2}{P_1}\right)$$

Interpretiamo questo risultato. Se $P_1 = P_2$ la soglia coincide con il valor medio, cosa del tutto ragionevole vista la perfetta simmetria del problema. Se invece le due probabilità sono sbilanciate la soglia si modifica, ad esempio se $P_1 > P_2$ aumenta, polarizzando la decisione a favore della classe 1. Questa regola è del tutto ragionevole, e implica che per i pixel non facilmente classificabili, quelli a metà strada fra μ_1 e μ_2, va portata in conto anche la conoscenza a priori disponibile, e cioè il fatto che una classe sia più probabile dell'altra. In particolare, se l'immagine è poco rumorosa (σ^2 piccolo rispetto a $\mu_2 - \mu_1$) la soglia si sposta poco, e quindi l'informazione a priori pesa poco rispetto al valore osservato. Al contrario, in presenza di un forte rumore (σ^2 grande) il valore osservato è molto inaffidabile e quindi pesa molto l'informazione a priori. Il limite di questo procedimento è la conoscenza dei parametri del modello, che non sono sempre facili da stimare.

2.2 K-means

Un modo alternativo all'uso del modello è quello di stimare la soglia mediante una procedura iterativa che cerca di adattarsi alla distribuzione dei dati. Dato che l'algoritmo non tiene conto della posizione del pixel dell'immagine, ma solo del suo valore di intensità, possiamo vettorizzare l'immagine e considerare quindi i dati memorizzati nella sequenza $x(i)$ per $i = 1, \cdots, N$ dove N è il numero di pixel presenti nell'immagine. Indichiamo poi con K il numero di classi presenti nell'immagine (nell'esempio del paragrafo precedente $K = 2$). L'algoritmo è di tipo iterativo e prevede i seguenti passi:

1. Scegliamo $T_1, T_2, \cdots, T_{K-1}$ soglie, che individueranno $C_1, C_2, \cdots, C_K$ regioni. La regione (detta anche *cella*) $k - esima$ contiene tutti i pixel per cui risulta $T_{k-1} \leq x(i) \leq T_k$ (si pone per definizione $T_0 = -\infty$ e $T_K = +\infty$).

2. Individuiamo $y_1, y_2, \cdots, y_K$ valori rappresentativi di ogni regione, detti anche esempi o *template*, in modo da minimizzare la distanza (euclidea) con i dati. Per esempio y_1 sarà

scelto in modo che:

$$y_1 = \text{argmin}_y \sum_{C_1} [x(i) - y]^2$$

dove la notazione sintetica $\sum_{C_1}$ indica la somma su tutti gli $x(i) \in C_1$. Derivando rispetto a y_1 ed eguagliando a zero si ha:

$$\begin{aligned} \frac{d}{dy_1} \sum_{C_1} [x(i) - y_1]^2 &= \frac{d}{dy_1} [\sum_{C_1} x(i)^2 + N_1 y_1^2 - 2y_1 \sum_{C_1} x(i)] \\ &= 2y_1 N_1 - 2 \sum_{C_1} x(i) = 0 \end{aligned}$$

da cui:

$$y_1 = \frac{1}{N_1} \sum_{C_1} x(i)$$

dove N_1 è il numero di pixel che appartengono alla classe C_1. Più in generale il template k-esimo si calcola come:

$$y_k = \frac{1}{N_k} \sum_{C_k} x(i)$$

Quindi il template della regione C_k coincide proprio con il valor medio dei dati che appartengono a quella regione.

3. Determiniamo le nuove soglie come $T_k = \frac{y_k + y_{k+1}}{2}$ e quindi le nuove partizioni dei dati;

4. ripetiamo i passi 1, 2 e 3 fino a convergenza, ad esempio, fino a quando tutti i template sono diventati stabili a meno di un ϵ prefissato.

In figura 18 si mostra la mappa binaria ottenuta con questo algoritmo per l'impronta rumorosa di figura 17. L'algoritmo K-means è molto robusto, converge sempre ad una soluzione ottima (nel senso del minimo errore quadratico medio) anche se l'ottimo può non essere quello globale e può essere facilmente esteso alle immagini a colori, multispettrali o a dati di natura diversa. Il maggior problema che si incontra nella pratica è quello della *cluster validation* cioè la scelta di K, il numero di classi, unico parametro dell'algoritmo. Per immagini non banali, non è affatto ovvio quante classi ci siano, e un valore errato di K conduce inevitabilmente ad una sotto/sovra segmentazione.

In figura 19 si mostra un'immagine monocromatica con diversi oggetti su uno sfondo scuro e la relativa segmentazione usando 3 classi, mentre in figura 20 si mostra la corrispondente immagine a colori segmentata sempre usando 3 classi.

Riferimenti bibliografici

[1] R.C.Gonzalez and R.E.Woods: *Digital Image Processing, 3rd Ed.*, Prentice Hall, 2008.

[2] A.Bovik, *The essential guide to Image Processing*, Academic Press, 2009.

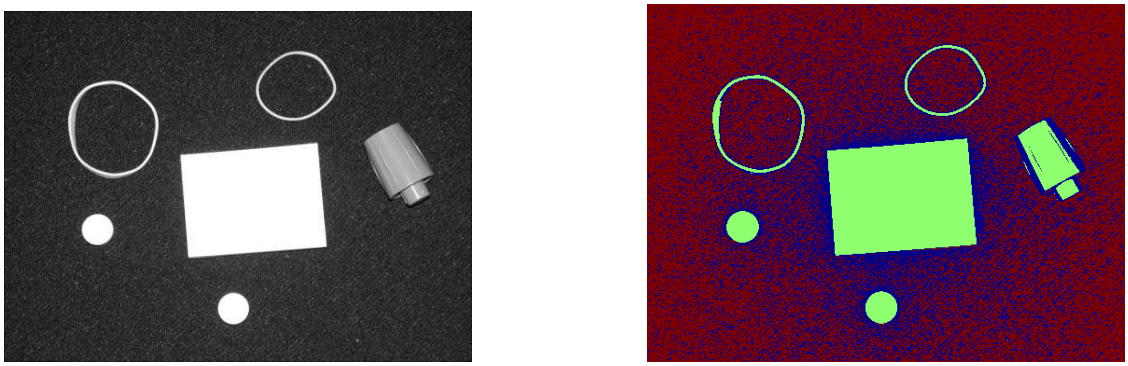

Figura 19: Immagine su scala di grigi e segmentazione con 3 classi.

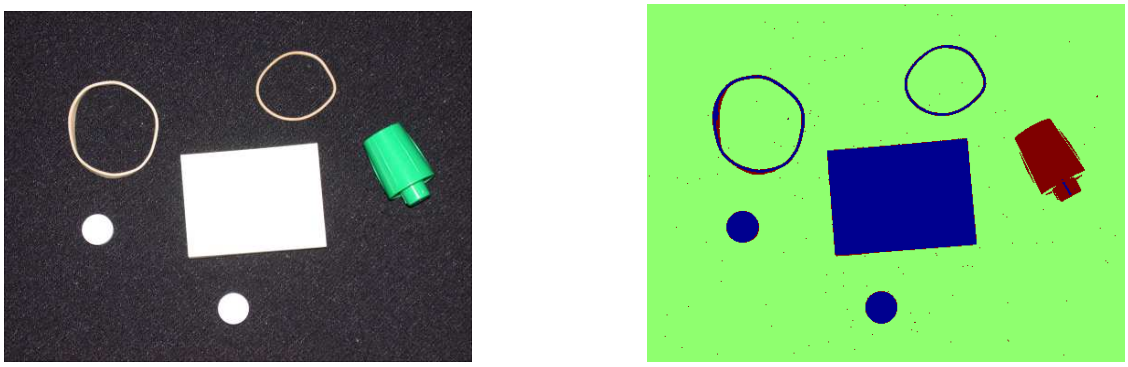

Figura 20: Immagine a colori e segmentazione con 3 classi.