Università degli Studi di Catania

FACOLTÀ DI SCIENZE MATEMATICHE FISICHE E NATURALI
Corso di Laurea Specialistica in Informatica

Computer Vision

Segmentazione di Immagini:
tecniche di soglia e region growing

Progetto di:
Alfonso Ridolfo
Matricola: B05/000210

Docente:
Prof. Sebastiano Battiato

Ottobre 2007

1 Introduzione

Dovendo realizzare un progetto per il corso di **Computer Vision**, ho scelto di affrontare una tra le tematiche più interessanti di questa disciplina: la segmentazione. Ho realizzato dei plugin per il noto software di image processing *ImageJ* che implementano delle possibili soluzioni a questa problematica utilizzando due diversi metodi: il primo consiste nel sottoporre l'immagine ad un'operazione di soglia; il secondo invece costruisce delle regioni a partire da dei punti (detti semi) iniziali, basandosi su opportuni criteri di similarità.

2 Segmentazione tramite soglia

Il plugin *Thresholding Segmentation* implementa due diversi algoritmi di segmentazione eseguita tramite l'applicazione di una soglia all'immagine in input. Il primo algoritmo utilizza, nel calcolo della soglia, informazioni relative all'immagine nel suo complesso; il secondo invece fa uso di dati che esprimono caratteristiche locali dell'input.

2.1 Global Thresholding

L'algoritmo di soglia globale esegue un calcolo iterativo della soglia T da applicare all'immagine di input.

1. $$T_0 = m_l + \frac{M_l - m_l}{2} \tag{1}$$

   dove m_l e M_l sono rispettivamente i valori di minimo e massimo dell'immagine (in termini di livello di grigio o di luminanza se l'immagine in input è a colori).

2. Si esegue una partizione dell'immagine di input I in 2 insiemi di pixel G_1 e G_2 tali che risulti:

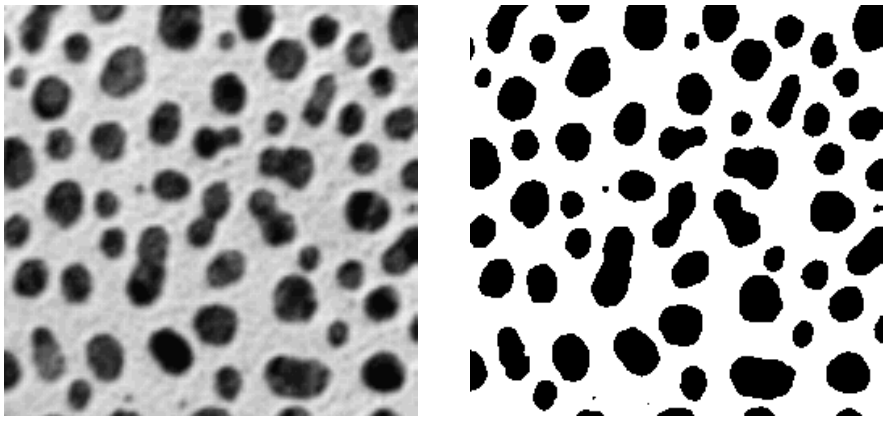
   $$\forall p_1 \in I : f(x_1, y_1) > T_0 \Longrightarrow p_1 \in G_1 \tag{2}$$
   $$\forall p_2 \in I : f(x_2, y_2) \leq T_0 \Longrightarrow p_2 \in G_2 \tag{3}$$

   laddove (x_1, y_1) e (x_2, y_2) sono le rispettive coordinate dei punti p_1 e p_2, mentre la f è una funzione che restituisce il valore dei pixel.

3. Per ciascuno dei 2 sottoinsiemi G_1 e G_2 si calcola il livello di grigio medio, rispettivamente μ_1 e μ_2

4. Si calcola un nuovo valore di soglia con la seguente formula:

   $$T_1 = \frac{1}{2}(\mu_1 + \mu_2) \tag{4}$$



(a) (b)

Figura 1: Immagine di prova `blobs` (a) e output del plugin (b), ottenuto con $T_t = 0$

5. Si iterano i passi dal 2 al 4 finchè non risulta

$$|T_{n-1} - T_n| \leq T_t \tag{5}$$

dove T_n rappresenta la soglia calcolata all'n-esima iterazione e T_t una soglia di terminazione.

Al fine di ottenere risultati migliori si possono apportare delle modifiche all'algoritmo di base: nel caso in cui si debba elaborare un immagine nella quale gli oggetti e lo sfondo occupano più o meno la stessa estensione, il passo 1 può essere modificato come segue:

$$T_0 = \mu \tag{6}$$

dove μ è la media dei valori dei pixel dell'intera immagine. Questo tipo di scelta non sembra invece consigliabile nel caso in cui uno dei due gruppi di pixel (oggetti e sfondo) prevalga sull'altro nell'input. Entrambi le modalità di avvio della procedura sono disponibili nel plugin e la scelta va effettuata in fase di avvio. Il parametro T_t influenza la rapidità con cui l'algoritmo termina; scegliendo un valore abbastanza vicino allo 0 si ottengono risultati intuitivamente migliori e più precisi. I più grossi pregi del plugin sono le buone prestazioni in termini di rapidità di terminazione, sulla quale è anche possibile agire tramite il parametro T_t, e l'efficacia in situazioni "semplici"; in effetti nel caso in cui le immagini provengano da ambienti le cui condizioni sono soggette ad un certo tipo di controllo (ad esempio applicazioni per l'ispezione a livello industriale) l'algoritmo riesce spesso a fornire risultati soddisfacenti come si può vedere in figura 1. Come visibile in figura 2, i risultati di questo algoritmo applicato ad immagini con illuminazione non uniforme non sono invece altrettanto positivi.

(a) (b)

Figura 2: oggetto con illuminazione non uniforme (a) e output del plugin (b), ottenuto per $T_t = 0$

2.2 Local Thresholding

La maggior parte dei metodi di soglia di tipo globale richiedono determinate peculiarità statistiche nell'input: l'istogramma deve essere essenzialmente di tipo bimodale con mode ben distinte. È possibile trovare questo tipo di situazione solo in determinati domini applicativi. Allo scopo di ottenere risultati positivi anche in presenza di immagini di altro tipo possono essere utilizzate informazioni sui bordi degli oggetti da identificare. Queste informazioni possono essere ottenute tramite gli operatori gradiente e laplaciano.
Sfruttando le caratteristiche dei risultati dell'applicazione di questi filtri derivativi sull'input, si cercano i bordi degli oggetti di interesse nell'immagine. Individuando valori significativi del gradiente è possibile localizzare i bordi di un oggetto, sui quali è poi possibile trarre ulteriori informazioni tramite il risultato della convoluzione con filtro laplaciano che permette di distinguere tra i due "versanti" di un bordo: il lato chiaro di un bordo fornisce output di valore negativo, mentre dal lato scuro della transizione avremo un output positivo. Imponendo che:

$$G(x, y) = |\nabla f(x, y)| \tag{7}$$

e che:

$$L(x, y) = |\nabla^2 f(x, y)| \tag{8}$$

dove $|\nabla f(x, y)|$ e $|\nabla^2 f(x, y)|$ rappresentano rispettivamente il modulo del gradiente ed il Laplaciano calcolati nel punto (x, y) ho implementato un algoritmo che ricerca gli oggetti di interesse nell'immagine creando una mappa $s(x, y)$

dell'immagine in input nel seguente modo:

$$s(x,y) = \begin{cases} 0 & \text{se } G(x,y) < T, \\ 1 & \text{se } G(x,y) \geq T \wedge L(x,y) < 0, \\ 2 & \text{se } G(x,y) \geq T \wedge L(x,y) \geq 0. \end{cases}$$

dove T è una soglia. I valori del modulo del gradiente vengono calcolati tramite le due componenti $G_x(x,y)$ e $G_y(x,y)$ tramite la formula:

$$G(x,y) \equiv \sqrt{G_x(x,y)^2 + G_y(x,y)^2}. \tag{9}$$

Il calcolo delle due componenti è stato effettuato tramite le maschere $Sobel - X$ e $Sobel - Y$:

$$\mathbf{Sobel - X} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad \mathbf{Sobel - Y} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Il kernel utilizzato per il calcolo del Laplaciano è invece il seguente:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Le informazioni ottenute in questo passaggio vengono utilizzate per generare un immagine segmentata a due livelli di grigio: i pixel degli oggetti di interesse vengono posti a 255 mentre i pixel dello sfondo vengono posti a 0. La procedura seguita per individuare gli oggetti si basa sulla struttura della mappa $s(x,y)$: si trovano pixel etichettati con 0 nelle zone omogenee, pixel etichettati con 1 sul lato chiaro di un bordo e pixel etichettati con 2 sul lato scuro di un bordo. Seguendo questo ragionamento si procede con la scansione in orizzontale ed in verticale della $s(x,y)$ alla ricerca della sequenza:

$$(...)(1,2)(0 \vee 1)(2,1)(...) \tag{10}$$

tenendo presente che (...) rappresenta una combinazione di 1, 2 e 0, tale sequenza individua una linea di scansione su cui è presente un oggetto scuro su sfondo chiaro (in caso di oggetti chiari su sfondo scuro bisogna invertire le etichette 1 e 2 nella creazione della $s(x,y)$). I pixel corrispondenti a tali sequenze vengono posti uguali a 255 nell'output mentre i restanti vengono posti a 0. L'impostazione di un valore adeguato per il parametro T risulta spesso decisiva per L'efficienza dell'algoritmo (e quindi del plugin che lo implementa). Le condizioni di illuminazione non uniforme possono essere trattate in maniera molto più efficace rispetto alla segmentazione su base globale come si può notare in figura 3.

(a)

(b)

Figura 3: oggetto con illuminazione non uniforme (a) e output del plugin di threshold locale (b), ottenuto per $T_g = 11$

3 Segmentazione tramite Region Growing

Le tecniche di segmentazione basate sull'applicazione di una soglia all'input hanno spesso il vantaggio di essere computazionalmente molto efficienti; tuttavia in molte applicazioni i risultati forniti possono essere piuttosto insoddisfacenti. Tra i diversi metodi proposti nel corso degli anni ho scelto di implementare una versione dell'algoritmo di *Region Growing*. L'algoritmo basa la sua efficacia sulla possibilità di distinguere tra bordi e zone omogenee scandendo una mappa del gradiente e su due considerazioni:

- sia G_p il valore del gradiente dell'immagine per cui il $p\%$ dei pixel presentano valori minori o uguali ad esso, data una scelta opportuna di p ,si può ragionevolmente supporre che i punti per cui risulta

  $$G(x, y) > G_p$$

  siano pixel di bordo;

- sia (x, y) il seme di una regione, il valore (o il colore) di ciascun pixel della stessa non deve discostarsi troppo da $f(x, y)$.

Ponendo $p = 95$ e siano (x_s, y_s) le coordinate del seme, dalle precedenti considerazioni seguono i criteri di similarità adoperati nella procedura di espansione per i quali un pixel di coordinate (x, y) viene annesso solo se:

- $G(x, y) \leq G_{95} \wedge |f(x, y) - f(x_s, y_s)| \leq T_i$
- $G(x, y) > G_{95} \wedge |f(x, y) - \mu_R| \leq 3\sigma_R$

dove μ_R e σ_R sono rispettivamente la media e la deviazione standard dei valori dei pixel della regione, T_i è una soglia che misura la differenza di intensità massima tra il seme ed il resto dei pixel della regione. Nel caso di immagini a colori le differenze in valore assoluto utilizzate nelle equazioni precedenti sono sostituite da distanze euclidee tra i punti individuati dalle tre componenti di colore nello spazio RGB. Il plugin (*Region Growing Segmentation T.jar*) applica all'immagine un filtro mediano allo scopo di ridurre il rumore nell'input, dopodichè costruisce una mappa del gradiente con le modalità descritte nella sezione relativa al *Local Thresholding* [2.2]. Da questa mappa estrae l'istogramma necessario per il calcolo di G_{95}. Pensando all'istogramma come una distribuzione di frequenza, G_{95} rappresenta il 95-esimo percentile della stessa e può essere ottenuto sommando le frequenze cumulative relative di ciascun valore del gradiente fino a giungere ad un valore pari a $0,95$ (le frequenze cumulative relative sono facilmente calcolabili a partire dalle frequenze assolute fornite dall'istogramma).
Partendo dal fatto che il seme appartiene alla regione, l'espansione utilizza una coda di supporto e procede come segue:

1. se il pixel di coordinate (x, y) appartiene alla regione secondo i criteri di similarità, lo si annette alla regione e si inseriscono gli otto suoi vicini nella coda. Si tiene traccia dei pixel già accodati con una mappa (un array nella fattispecie) di valori booleani, evitando così di accodare (e quindi testare) un pixel più di una volta;

2. si effettua un operazione di *dequeue* sulla coda e se si ottiene un pixel si ritorna al passo 1 altrimenti la coda è vuota e la procedura termina.

Il plugin permette di scegliere il numero delle regioni da individuare e la soglia T_i, attendendo dopo la scelta l'input dei semi tramite click del mouse sull'immagine.
In figura 4 si può osservare un esempio di output dall'elaborazione di un'immagine. Ho implementato anche una variante del plugin nel quale il primo criterio di appartenenza è modificato nel seguente modo:

$$G(x, y) \leq G_{95} \wedge f(x, y) \leq T_{ai} \tag{11}$$

dove T_{ai} è una soglia di intensità approssimata con la seguente

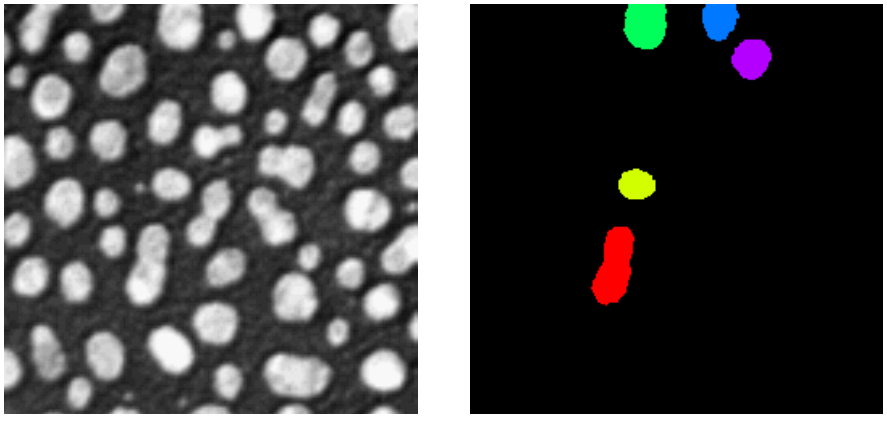
- Dall'immagine in input si estraggono i pixel che sono stati selezionati come bordo dalla procedura che calcola G_{95} (ossia i pixel per cui risulta $G(x, y) > G_{95}$;

- $T_{ai} = \mu_e$ dove μ_e è la media dei valori dei pixel di bordo selezionati al passo precedente.

Questa variante del software (contenuta in *Region Growing Segmentation.jar*) fornisce risultati soddisfacenti solo con determinati tipi di immagine, ma ha il vantaggio di non richiedere in input una soglia all'utente, stimando una possibile soglia con il metodo appena descritto. Ne abbiamo un esempio di elaborazione in figura 5.

(a) (b)

Figura 4: Immagine di prova `forme uneven` (a) e output del plugin di region growing, ho selezionato la forma in basso a destra presente nell'input ed ho ottenuto l'output (b), con $T_i = 30$



(a) (b)

Figura 5: Immagine di prova `blobs` (a) e output del plugin di region growing, ho selezionato 5 dei *blobs* presenti nell'input ed ho ottenuto l'output (b)