Elaborazione e Analisi delle Immagini

Giuseppe Bellisano

30 novembre 2016

Indice

Capitolo 1

Image segmentation

1.1 Cosa è l'image segmentation?

La segmentazione è uno dei passi più importanti nell'analisi delle immagini processate

Il suo scopo è quello di **dividere** l'immagine in parti che hanno una forte correlazione con oggetti o aree che si trovano nel mondo reale.

1.2 Quanti tipi di segmentazione abbiamo?

Abbiamo due tipi di segmentazione:

1. **Segmentazione Completa** in cui le regioni corrispondono direttamente con gli oggetti dell'immagine in ingresso. In pratica ogni elemento dell'immagine viene riconosciuto e isolato. Per ottenere questo risultato è necessario avere informazioni di processing di alto livello (high level processing) da utilizzare per comprendere il dominio del problema da risolvere (che ci permette di riconoscere gli oggetti/regioni)

2. **Segmentazione Parziale** in cui le regioni/segmenti di immagini non corrispondono direttamente con gli oggetti reali rappresentati nell'immagine. In questo caso l'immagine è divisa in regioni omogenee secondo una determinata proprietà scelta (colore, texture, riflettività).

   La segmentazione parziale è seguita da un ulteriori processi sino ad arrivare all'immagine finale di segmentazione che può essere trovata grazie all'aiuto di informazioni di alto livello.

1.3 In quali campi viene utilizzata la segmentazione?

Ecco alcune delle problematiche più comuni che possono essere risolte con la segmentazione:

- ottenere dati da immagini ambigue

- rumore nelle informazioni
- ottenere oggetti con discreto contrasto su di uno sfondo uniforme
- semplici lavori di raggruppamento come cellule del sangue, caratteri stampati

In generale è difficile ottenere una corretta e completa segmentazione; per questo spesso di utilizza una segmentazione parziale come input in un processo di alto livello.

1.4 Quali sono i metodi utilizzati per la segmentazione?

- Thresholding (Approccio Globale) che si basa sull'istogramma di alcune caratteristiche dell'immagine come il **gray level thresholding** o **soglia del livello di grigio**. Si ricercano i toni soglia che separano oggetti differenti nell'immagine. E' il metodo più semplice e veloce
- Segmentazione basata sugli Edge in cui ci cercano i contorni degli oggetti basandosi sulla discontinuità dei toni, cioè sugli edge
- Segmentazione basata sulle Regioni in cui si ricercano le regioni, cioè degli oggetti uniformi secondo un certo criterio (stesso tono di grigio, o tono di grigio che si differenzia al massimo di una certa soglia)
  - region growing
  - region splitting and merging

1.5 Cosa si intende con Approccio Globale: Gray level thresholding?

E' il sistema più semplice a livello concettuale di segmentazione e per questo è anche il sistema più efficiente computazionalmente e veloce.

Esso viene applicato alle immagini che contengono regioni o oggetti con un costante livello di riflettività o assorbimento di luce. Si sfrutta questa caratteristica per partizionare l'immagine grazie ai valori di intensità e/o alle proprietà di questi valori.

1.6 Quale è l'idea alla base del gray level thresholding con due elementi (mode) da separare?

1. Si supponga di avere un istogramma di intensità di una immagine *f(x,y)* che è composta da un oggetto bianco (level 255) e di uno sfondo scuro (level 0) e che si voglia estrarre l'oggetto dallo sfondo.

2. Inizialmente si seleziona una soglia T con cui si ottiene l'immagine segmentata *g(x, y)*:

$$g(x,y) = \begin{cases} 1 & se \quad f(x,y) > T \\ 0 & se \quad f(x,y) \leq T \end{cases}$$

3. Se il valore di intensità dell'immagine è superiore a quello della soglia, si assegna a *g(x, y) il valore 1, altrimenti 0*

1.7 Quale è l'idea alla base del gray level thresholding con tre elementi (mode) da separare?

- Maggiore è il numero di elementi (mode), più complessa sarà la segmentazione. Esaminiamo il caso di una immagine con due oggetti chiari posti su uno sfondo scuro.
- L'immagine segmentata risultante *g(x, y)* sarà:

$$g(x,y) = \begin{cases} a & se \quad f(x,y) > T_2 \\ b & se \quad T_1 < f(x,y) \leq T_2 \\ c & se \quad f(x,y) \leq T_1 \end{cases}$$

dove *a, b, c* sono tre diversi valori di intensità e T_1 e T_2 sono due differenti valori di soglia.

1.8 Nel gray level thresholding quanti e quali tipi di soglia possiamo avere?

Possiamo avere due tipi di soglia:

1. Soglia Globale
2. Soglia Variabile, che si divide in soglia locale o regionale

1.9 Il thresholding applicato ad immagini con rumore e variazioni di luminosità: cosa cambia?

Esaminando l'istogramma dell'immagine si nota che questo subisce delle variazioni a seconda che l'immagine si affetta da rumore o una intensità variabile.

1.10 Cosa è il single global threshold e quando si applica?

Il single global threshold si applica a quelle immagini in cui l'intensità degli oggetti e dello sfondo rende questi facilmente distinti. L'algoritmo che si applica all'intera immagine si basa sui seguenti passi:

1. Si seleziona una soglia T
2. Si effettua la segmentazione dell'immagine usando la soglia T producendo due gruppi di pixel:
   - G_1 dato da tutti i pixel che hanno una intensità $\geq T$
   - G_2 dato da tutti i pixel che hanno una intensità $< T$
3. Si calcolano i valori di **intensità medi** m_1 e m_2 rispettivamente per i gruppi di pixel G_1 e G_2
4. Si determina un nuovo livello di soglia T secondo la formula:

$$T = \frac{1}{2}(m_1 + m_2)$$

5. Si ripetono i passi dal 2 al 4 sino a quando la differenza tra i valori di T (delle successive iterazioni) è più piccolo rispetto al valore predefinito T_0

1.11 Quali sono le caratteristiche del single global threshold?

- E' un algoritmo semplice.
- Funziona bene nei casi in cui l'istogramma dell'immagine presenta delle campane (le mode) corrispondenti agli oggetti e allo sfondo separate da una valle.
- Il valore di soglia iniziale deve essere maggiore rispetto al minimo valore di intensità e minore rispetto al massimo valore di intensità nell'immagine.

1.12 Qual è l'algoritmo Matlab per il global thresholding?

Indicando con:

- f per l'immagine di input
- g per l'immagine di output
- T per la soglia

```
function [g]=iter_thresh(f)
T = 0.5 * (double(min(f(:)))+ double(max(f(:))));
flag = false;
while ~flag
  g = f >= T;
  Tnext = 0.5 * (mean(f(g)) + mean(f(~g)));
  flag = abs (T - Tnext) < 0.5;
  T = Tnext;
end
```

1.13 Cosa è l'Optimum global thresholding o metodo Otsu?

Meglio conosciuto come Metodo Otsu è un è un metodo di sogliatura automatica dell'istogramma nelle immagini digitali.

Viene definito come "ottimo" perché **massimizza la varianza tra le classi**. Un algoritmo di sogliatura (threshold) che fornisce la migliore separazione tra le classi in termini dei loro valori di intensità è il miglior threshold.

1.14 Come funziona il metodo Otsu?

- Si considerino:
  - {0, 1, 2, ... , L-1} siano gli L distinti livelli di intensità in una immagine
  - MxN sia la dimensione dell'immagine di M righe X N colonne
  - n_i indica il numero di pixel con intensità i
  - $MN = n_0 + n_1 + n_2 + \dot{,} + n_{L-1}$
- l'istogramma normalizzato ha componenti $p_i = \frac{n_i}{MN}$ da cui segue:

$$\sum_{i=0}^{L-1} p_i = 1 \quad p_i \geq 0$$

- Selezionando e utilizzando una soglia $T(k) = k$ con $0 < k < L-1$ con l-immagine di input, si ottengono le due classi c_1 e C_2 dove:
  - C_1 è la classe di tutti i pixel con valori di intensità compresi nel range $[0, k]$
  - C_2 è la classe di tutti i pixel con valori di intensità compresi nel range $[k+1, L-1]$
- Usando questa soglia, si può calcolare la *probabilità* $P_1(k)$ che un pixel appartenga alla classe C_1:

$$P_1(k) = \sum_{i=0}^{k} p_i$$

- La *probabilità* che il pixel invece appartenga alla classe C_2 è data da:

$$P_2(k) = \sum_{i=k+1}^{L-1} p_i = 1 - P_1(k)$$

- Il valore di *intensità media* di tutti i pixel della classe C_1 è:

$$m_1(k) = \sum_{i=0}^{k} iP(\frac{i}{C_1}) = \sum_{i=0}^{k} iP(\frac{C_1}{i})\frac{P(i)}{P(C_1)} = \frac{1}{P_1(k)}\sum_{i=0}^{k} ip_i$$

- Il valore di *intensità media* di tutti i pixel della classe C_2 è invece:

$$m_2(k) = \sum_{i=k+1}^{L-1} iP(\frac{i}{C_2}) = \frac{1}{P_2(k)}\sum_{i=k+1}^{L-1} ip_i$$

- La *media complessiva* sino al livello k è data da:

$$m(k) = \sum_{i=0}^{k} ip_i$$

- Invece l'*intensità media* dell'intera immagine (**la media globale**) è data da:

$$m_G = \sum_{i=0}^{L-1} ip_i$$

- Inoltre valgono le seguenti:

$$P_1 m_1 + P_2 m_2 = m_G$$

$$P_1 + P_2 = 1$$

- Per valutare la bontà della soglia al livello k si usa la *metrica normalizzata*:

$$\eta = \frac{\sigma_B^2}{\sigma_G^2}$$

  - dove σ_G^2 è la **varianza globale** ed è *costante*:

$$\sigma_G^2 = \sum_{i=0}^{L-1}(i - m_G)^2 p_i$$

  - dove σ_B^2 è la **varianza tra le classi** che indica la misura della *separabilità* tra le classi:

$$\sigma_B^2 = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 = P_1 P_2(m_1 - m_2)^2 = \frac{(m_G P_1 - m)^2}{P_1(1 - P_1)}$$

- (The farther the two means m 1 and m 2 are from each other the larger will be.)
- Poiché σ_G^2 è una costante ne consegue che η è una *misura della separabilità*. Inoltre massimizzare questa metrica, è identico a massimizza σ_B^2
- l'obiettivo principale è determinare il valore k di soglia che massimizza la varianza di classe.
- Introducendo nuovamente k si ha:

$$\eta(k) = \frac{\sigma_B^2}{\sigma_G^2}$$

e

$$\sigma_B^2(k) = \frac{[m_g P_1(k - m(k)]^2}{P_1(k[1 - P_1(k)]}$$

- La **soglia ottimale** è il valore k^* che massimizza $\sigma_B^2(k)$:

$$\sigma_B^2(k^*) = \max_{0 \leq k \leq L-1} \sigma_B^2(k)$$

- Per trovare K^* si deve valutare questa equazione per tutti i valori interi di k e scegliere il valore che produce il massimo $\sigma_B^2(k)$
- Se il massimo esiste *per più di un valore di k* è consuetudine mediare i diversi valori di k per cui $\sigma_B^2(k)$ è massima.
- La metrica normalizzataη valutata per il valore di soglia ottimale $\eta(k^*)$ può essere utilizzata per ottenere una stima quantitativa della separabilità delle classi, che a sua volta fornisce un'idea della facilità di sogliatura di una data immagine. Il suo valore oscilla tra 0 e 1:
  - il valore più basso è ottenibile solo da immagini con un singolo livello di intensità costante
  - Il limite superiore invece è ottenibile solo dalle immagini 2-valori con intensità uguali a 0 e $L - 1$

1.15 Spiegare le sogliature multiple

Il metodo di Otsu può essere esteso ad un numero arbitrario di soglie in modo da valutare un diverso numero di classi.

Generalmente i metodi che utilizzano più di due soglie vengono risolte con più valori di intensità (colore).

Un altro metodo è quello di suddividere l'immagine in *rettangoli sovrapposti*

- questo sistema viene utilizzato per compensare i casi di illuminazione non uniforme
- i rettangoli vengono scelti di piccole dimensioni in modo che l'illuminazione da essi coperta, sia uniforme

1.16 Illustra il metodo Otsu in Matlab

La funzione *graythresh* di Matlab permette di calcolare una soglia utilizzando il metodo di Otsu.

1.17 Tecniche per migliorare il global thresholding

- Applicare lo smoothing all'immagine prima del thresholding: è una tecnica che si applica quando il rumore non può essere ridotto e come metodo di sogliatura si seleziona il global thresholding. Di seguito un esempio di codice Matlab.

```
f=imread('large_septagon_gaussian_noise_mean_0_std_50_added.
    tif');
figure, imshow(f);
figure, imhist(f);
T=graythresh(f);
T=T*255
g=f>=T;
figure, imshow(g);
filt=fspecial('average', [5 5]);
ff=imfilter(f, filt);
figure, imshow(ff);
figure, imhist(ff);
ylim([0 16000])
T1=graythresh(ff);
T1=T1*255
g1=ff>=T1;
figure, imshow(g1);
```

- Thresholding variabile attraverso il partizionamento dell'immagine. Di seguito un esempio di codice Matlab.

Capitolo 2

Segmentazione

2.1 Formulazione matematiche di base

- R rappresenta l'intera regione dell'immagine
- la segmentazione è il processo che divide la regione R nelle sottoregioni $R_1, R_2, \ldots, R_n$ per cui valgono
  1. $\bigcup_{i=1}^{n} R_i = R$ ovvero la somma di tutte le sottoregioni è uguale all'intera regione R dell'immagine
  2. R_i è un insieme connesso dove $i = 1, 2, \ldots, n$
  3. $R_i \cap R_j = \emptyset \quad \forall i, j \quad i \neq j$
  4. $Q(R_i) = TRUE \quad i = 1, 2, \ldots, n$
  5. $Q(R_i \cup R_j) = FALSE$ per ogni regione adiacente R_i e R_j

2.2 Region growing

E' una procedura che raggruppa i pixel o le sottoregioni in regioni più grandi basandosi su criteri predefiniti.

L'approccio base è il seguente:

- si parte con un insieme di punti "seme"
- da queste grow regioni si aggiungono ad ogni seme i suoi pixel vicini che hanno dele predefinite proprietà simili a quelle del seme (come uno specifico range di intensità o un colore)
- è importante selezionare con attenzione i punti *da cui partire* in base al tipo di problema o all'immagine da processare.
  - le immagini satellitati dipendono dal colore
  - le immagini monocramatiche dipendono dai descrittori basati sui livelli di intensità e sulle proprietà spaziali (texture, momenti)
- devono essere usate le proprietà della connessione
- si deve formulare una regola di stop

2.3 Qual è l'algoritmo di un region growing?

- si assume che:
  - $f(x, y)$ è un'immagine in ingresso
  - $S(x, y)$ indica un array di semi contenente un 1s nei posti dei punti del seme e 0 negli altri punti
  - Q indica un predicato che verrà applicato ad ogni locazione (x, y)
- si assume che gli array f ed S abbiano la stessa dimensione
- si utilizza una 8-connessione

Con queste premesse si applica il seguente algoritmo:

1. si trovano tutte le componenti in $S(x, y)$
   - si erode ogni componente connessa sino ad ottenere un solo pixel
   - si etichetta (label) ogni pixel trovato con il valore 1
   - si etichettano tutti gli altri pixel in S con il valore 0
2. si realizza un'immagine f_q, tale che per ogni coppia di coordinate (x, y) si abbia che:
   - $F_q(x, y) = 1$ se l'immagine in ingresso soddisfa il predicato Q a queste coordinate
   - $f_q(x, y) = 0$ negli altri casi
3. si forma così un'immagine g realizzata aggiungendo ad ogni punto seme in S tutti i punti 1-valutati in f_q che sono 8-connessioni al punto seme
4. si etichetta ogni componente connessa in g con una etichetta di regione differente (1, 2, 3, ...). Questa è così l'immagine segmentata ottenuta attraverso l'algoritmo region growing

2.4 Region Splitting and Merging

2.4.1 Definizione

Si tratta di una procedura che suddivide un'immagine prima di tutto in un insieme arbitrario di regioni disgiunte e quindi le fonde (merge) e/o le suddivide in regioni che soddisfino i requisiti della segmentazione.

2.4.2 Region Splitting and Merging: algoritmo

- si assuma che:
  - R è la regione che rappresenta l'intera immagine
  - sia Q il predicato utilizzato

- si parte con l'intera regione R suddividendo questa in quadranti sempre più piccoli R_i, dove per ognuno di essi vale che $Q(R_i) = TRUE$.
  - se $Q(R) = FALSE$ si divide l'immagine in nuovi quadranti
  - se $Q = FALSE$ per ogni quadrante, si suddivide il quadrante in ulteriori sotto-quadranti e così via

2.4.3 Region Splitting and Merging: tecniche di rappresentazione

Una delle tecniche più utilizzata è denominata **Quadtree**:

- si tratta di un albero in cui cui ogni nodo ha 4 discendenti
- la radice corrisponde all'intera immagine
- ogni nodo corrisponde a 4 foglie

2.4.4 Caratteristiche

- se viene usato solo lo *splitting* la partizione finale contiene regioni adiacenti con proprietà identiche
- soddisfare i vincoli di segmentazione richiede la fusione solo delle regioni adiacenti i cui pixel soddisfano il predicato Q:
  - due regioni adiacenti R_i e R_j sono fuse solo se vale: $Q(R_i \cup R_j) = TRUE$

2.4.5 Algoritmo per lo splitting and merging

1. si suddivide in 4 quadranti disgiunti ogni regione R_j per cui $Q(R_i) = FALSE$
2. quando non è possibile un ulteriore suddivisione, si fonde ogni regione R_i e R_j adiacente per cui $Q(R_i \cup R_j) = TRUE$
3. ci si ferma quando non sono possibili ulteriori fusioni

Si è soliti indicare una dimensione minima quadregion oltre il quale nessuna ulteriore frazionamento sia effettuato.

una variante: la fusione di due qualsiasi regioni adiacenti R_i e R_j se ognuno soddisfa il predicato singolarmente.

2.4.6 Region growing Matlab

```
function [g, NR, SI, TI] = regiongrow(f, S, T)
f = double(f);
% if S is a scalar, obtain the seed image
if numel(S) == 1
SI = f == S;
S1 = S;
else
SI = bwmorph(S, 'shrink', Inf);
```

```
J = find(SI);
S1 = f(J); % Array of seed values
end
TI = false(size(f));
for K = 1:length(S1)
seedvalue = S1(K);
S = abs(f - seedvalue) <= T;
TI = TI | S;
end
[g, NR] = bwlabel(imreconstruct(SI, TI));
```

S can be an array (the same size as f) with a 1 at the coordinates of every seed point and 0s elsewhere. S can also be a single seed value. (Our example S = 255)

Similarly, T can be an array (the same size as f) containing a threshold value for each pixel in f. T can also be a scalar, in which case it becomes a global threshold. (Our example T = 65)

g is the result of region growing, with each region labelled by a different integer.

NR is the number of regions.

SI is the final seed image used by the algorithm.

TI is the image consisting of the pixels in f satisfied the threshold test.

Use function imreconstruct with SI as the marker image to obtain the regions corresponding to each seed in S.

bwlabel assigns a different integer to each connected region.

2.4.7 Region growing Matlab: le funzioni

- la funzione predefinita per implementare il quadtree è *qtdecomp* con la sintassi:

$$S = qtdecomp(f, @split_test, parameters)$$

  - f è l'immagine di ingresso
  - S è una matrice sparsa contenente la struttura quadtree
    - se $S(k,m) \neq 0$, allora (K,m) è l'estremo superiore sinistro del blocco in decomposizione e la dimensione del blocco è data da $S(k,m)$
  - la funzione $split_test$ è usata per determinare se una regione deve essere divisa o meno
  - $parameters$ indica eventuali parametri addizionali

- per ottenere i valori dei pixel della quadregion in una decomposizione quadtree, si utilizza la funzione **qtgetblk** con la sintassi:

$$[vals, r, c] = qtgetblk(f, S, m)$$

  - $vals$ è un array che contiene i valori dei blocchi di dimensione mXm nella decomposizione quadtree f

  - S è la matrice sparsa restituita dalla funzione *qtdecomp*
  - i parametri r, c sono dei vettori che contengono le coordinate della riga e della colonna degli angoli in alto a sinistra dei blocchi

- la funzione che implementa l'algoritmo di segmentazione è **splitmerge** che la seguente sintassi:

$$g = splitmerge(f, mindim, @predicate)$$

  beginitemize

- f è l'immagine di ingresso
- g è l'immagine in uscita
- *mindim* definisce la dimensione del più piccolo blocco nella decomposizione (potenza di 2)
- *predicate* è una funzione definita dall'utente che deve essere inclusa nell'ambiente (path) di Matlab. La sua sintassi è:
  - $flag = predicate(region)$
  - restituisce *true* (1) se i pixel nella regione soddisfa il predicato definito dal codice nella funzione
  - restituisce $false$ (0) negli altri casi
  -

```
function flag = predicate(region)
sd = std2(region);
m = mean2(region);
flag = (sd > 10) & (m > 0) & (m < 125);
```

2.4.8 Region Splittin and Merging in Matlab: splitmerge

```
function g = splitmerge(f, mindim, fun)
Q = 2^nextpow2(max(size(f)));
[M, N] = size(f);
f = padarray(f, [Q - M, Q - N], 'post');

S = qtdecomp(f, @split_test, mindim, fun);

Lmax = full(max(S(:)));

g = zeros(size(f));
MARKER = zeros(size(f));

for K = 1: Lmax
  [vals, r, c] = qtgetblk(f, S, K);
  if ~isempty(vals)
    for I = 1:length(r)
      xlow = r(I);
      ylow = c(I);
      xhigh = xlow + K -1;
```

```
      yhigh = ylow + K -1;
      region = f(xlow:xhigh, ylow:yhigh);
      flag = feval(fun, region);
      if flag
        g(xlow:xhigh, ylow:yhigh) = 1;
        MARKER(xlow, ylow) = 1;
      end
    end
  end
end

g = bwlabel(imreconstruct(MARKER, g));
g = g(1:M, 1:N);
```

2.4.9 Region Splitting and Merging in Matlab: split test

```
function v = split_test(B, mindim, fun)

k = size(B,3);
v(1:k) = false;

for I = 1:k
  quadregion = B(:, :, I);
  if size(quadregion, 1) <= mindim
    v(I) = false;
    continue;
  end

  flag = feval(fun, quadregion);
  if flag
    v(I) = true;
  end
end
```

Capitolo 3

Image processing con Matlab

3.1 Ridimensionamento

comando: imresize
possibili domande:

- Leggere un'immagine e rimpicciolirla o farne lo zoom di un valore intero
- Leggere un'immagine e rimpicciolirla o farne lo zoom di un valore decimale con interpolazione lineare

3.2 Trasformazioni geometriche

In Matlab è possibile effettuare una trasformazione geometrica affine specificando la matrice di trasformazione T attraverso il comando *maketform*. Per effettuare la trasformazione si usa il comando *imtransform*.
comando: maketform, imtransform, imrotate, impixelinfo (pixval),
possibili domande:

- Effettuare la rotazione di un'immagine qualsiasi
- Confrontare il risultato con quello ottenuto mediante la funzione *imrotate*
- Leggere l'immagine di 'lena' e realizzare l'ingrandimento di una zona dell'immagine usando la matrice di trasformazione T la sezione da ingrandire è intorno all'occhio di lena. Per individuare la sezione e, quindi, avere informazioni sulla posizione dei pixel potete usare il comando *impixelinfo* o *pixval* (in base alla versione di Matlab più o meno recente).
- Fare degli esperimenti modificando il tipo di interpolazione e notate l'effetto di blocchettatura causa
- La combinazione di diverse trasformazioni affini è ancora una trasformazione affine, che può essere ottenuta tramite il prodotto (matriciale) delle matrici che le definiscono.

- Scrivere una funzione dal prototipo `function g=rot_dist(f, alfa, c)` per realizzare prima una rotazione e poi una distorsione verticale.
- Creare l'immagine di ingresso usando il seguente comando `f = checkerboard(40);` in modo da generare una scacchiera su cui le modifiche risultano essere più facilmente visibili.

3.3 Rumore

- Aggiungere del rumore gaussiano bianco ad un immagine f con il comando `noisy = f + n con n = d*randn(size(f))` dove **d è la deviazione standard del rumore**.
- Rimuovere il rumore dall'immagine con i filtri a media mobile (al variare della dimensione della finestra).
- Valutare l'efficacia del filtraggio sia visivamente sia calcolando l'errore quadratico medio tra f e l'immagine "ripulita" .
- L'errore quadratico medio rappresenta una misura quantitativa per stabilire quanto l'immagine elaborata sia simile all'originale.
- L'MSE (Mean Squared Error) tra due immagini si definisce come:

$$MSE = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |f(m,n) - g(m,n)|^2$$

3.4 Smoothing con thresholding

- Consideriamo l'immagine 'telescopio.jpg', proveniente dal telescopio Hubble, in orbita intorno alla terra. Rilevare gli oggetti grandi realizzando le seguenti operazioni:
  - Visualizzare l'immagine
  - Applicare il filtro che effettua la media aritmetica su una finestra di dimensioni 15x15 e visualizzare il risultato
  - Applicare un'operazione a soglia per eliminare gli oggetti piccoli (considerare una soglia pari al 25 per cento del valore massimo presente nell'immagine filtrata)
  - Visualizzare il risultato dell'elaborazione

Capitolo 4

Morfologia

4.1 Introduzione

La morfologia è una branca della biologia che tratta le forme e la struttura di animali e piante. Nel contesto dell'elaborazione delle immagini, la **morfologia matematica** è uno strumento che è utile per:

- estrarre componenti dall'immagine che sia utili per rappresentare e descrivere regiorni come confini, scheletri, e gusci convessi
- attuare tecniche di processing come il filtraggio morfologico il thinning e il pruning (potatura)

4.1.1 Campi di utilizzo della morfologia matematica

- mage enhancement
- image restoration
- noise reduction
- space-time filtering
- image segmentation
- edge detection
- texture analysis
- particle analysis
- component analysis
- shape analysis
- feature generation
- feature detection
- skeletonization

- general thinning
- curve filling
- image compression

4.1.2 Il linguaggio della morfologia matematica

Il linguaggio alla base della morfologia matematica (denominato MM) è basato sulla **teoria degli insiemi**.

Si campionano le partizioni xy del piano in una griglia, con le coordinate del centro di ogni griglia che diventano un paio di elementi che formano lo spazio degli interni 2D $ZxZ(x^2)$ tale che è l'insieme di tutte le coppie di elementi ordinati $(a, b), a, b \in Z$

Gli **insiemi** nella MM rappresentano le forme degli oggetti in una immagine.

- l'insieme di tutti i pixel neri in una immagine binaria è una descrizione completa dell'immagine
- in una immagine binari gli insiemi in questione sono membri Z^2 dove ogni elemento di un insieme è una 2-tupla (o un 2-D vettore) le cui le coordinate x, y sono le coordinate di un pixel bianco (o nero a seconda della convenzione adottata) nell'immagine

4.1.3 Altri concetti della teoria degli insiemi

- sia A un insieme in z^2 con elementi (x, y):
  - se $w = (x, y)$ è un elemento di A, $\quad w \in A$
  - se $w = (x, y)$ non è un elemento di A, $\quad w \notin A$
- un insieme di pixel B che soddisfa una particolare condizione è scritto come:
$$B = \{w|\text{condizione}\}$$
- il **complemento di A** è l'insieme di tutte le coordinate di pixel che non appartengono ad A è indicato con A^c:
$$A^C = \{w|w \notin A\}$$
- l'**unione** di due insiemi A e B è l'insieme formato da tutti gli elementi di A e B ed è indicato come:
$$C = A \cup B$$
- l'**intersezione** di due insiemi A e B è l'insieme formato da tutti gli elementi che appartengono sia ad A che a B ed è indicato come:
$$C = A \cap B$$
- la **differenza** di due insiemi A e B è l'insieme degli elementi che appartengono ad A ma non a B ed è indicato come:
$$A \breve{} B = \{w|w \in A, w \notin B\}$$

- la **riflessione** di un insieme B indicato con $\hat{B}$:

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

- la **traslazione** di un insieme B di un punto $z = (z_1, z_2)$ è indicato con $(B)_z$ e definito come:

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

Indice analitico

Glossario

connesso pulsazione naturale.

trasformazione geometrica affine Esempi di affinità sono rotazioni, omotetie, traslazioni, rototraslazioni, riflessioni. Le affinità non sono necessariamente isometrie, non preservano cioè angoli e distanze, mentre mantengono sempre il parallelismo tra le rette..