UNIVERSITÀ DEGLI STUDI DI BRESCIA
FACOLTÀ DI INGEGNERIA
CORSO DI LAUREA IN INGEGNERIA ELETTRONICA

ELABORATO DI ROBOTICA A.A. 1999/2000

CONFRONTO TRA ALGORITMI DI SEGMENTAZIONE DI IMMAGINI A COLORI

Marco Lazzaroni 27147
Luca Lorenzoni 27291

Indice

1. Introduzione

1.1. Tecniche di segmentazione

La segmentazione di un'immagine consiste nella divisione dell'immagine in regioni, sulla base di un certo criterio di appartenenza dei pixel ad una regione, di solito con l'obiettivo di riconoscere gli oggetti che compongono la scena.

Ogni algoritmo di segmentazione soddisfa requisiti che dipendono dal proprio campo di applicazione. I metodi di segmentazione sono quindi molto specifici e diversi tra loro. È comunque possibile definire delle tecniche generali, sulle quali tutti gli algoritmi di segmentazione si basano. Esse sono di seguito brevemente descritte.

1. ***Thresholding***. Ogni pixel di un'immagine a toni di grigio è di solito caratterizzato da un valore di luminanza. L'algoritmo, di base, fissa una soglia di luminanza, allo scopo di distinguere tra due regioni, ad esempio lo sfondo e gli oggetti dell'immagine. La soglia può essere fissata arbitrariamente, o in maniera automatica in base ad un certo criterio, usualmente statistico. Siccome i risultati di un tale algoritmo sono grezzi, è possibile migliorare la tecnica considerando varie soglie. È inoltre facile estendere questa tecnica ad immagini a colori.

2. ***Clustering***. Dalla rappresentazione dell'immagine nell'usuale spazio bidimensionale, si passa ad uno spazio delle caratteristiche, ad esempio lo spazio tridimensionale RGB, e si procede ad un partizionamento di tale spazio, allo scopo di definire le regioni. In questo modo si perdono le proprietà di vicinato tra i pixel, e quindi si possono generare regioni non connesse. Il partizionamento avviene secondo criteri diversi in dipendenza dello scopo della segmentazione. Il numero di partizioni può essere impostato a priori o automaticamente scelto.

3. ***Region growing***. Si basa su informazioni di vicinanza tra i pixel. All'inizio viene scelto un insieme di pixel, detti semi. Ogni regione è inizialmente costituita da uno di tali semi. In seguito, si cerca di ingrandirla unendo ad essa i pixel vicini, a condizione che soddisfino un certo criterio di somiglianza. Se il pixel non viene associato alla regione, sarà lasciato libero di aggregarsi ad un'altra regione o di diventare un nuovo seme. I vari algoritmi che prendono spunto da questa tecnica

si differenziano per la scelta del seme, e per l'eventuale successiva applicazione di metodi di accorpamento tra regioni.

4. ***Split and merge***. L'immagine viene suddivisa in quattro parti; ogni parte non sufficientemente omogenea viene ulteriormente suddivisa, fino a quando tutte le regioni ottenute soddisfano il criterio di omogeneità.

5. ***Estrazione dei contorni***. Viene utilizzato un filtro di estrazione dei contorni presenti nell'immagine, che vengono successivamente connessi. I pixel interni ad un contorno sono raggruppati e costituiscono una regione.

6. ***Texture***. In presenza di texture, ovvero aree di immagine caratterizzate da una successione di colori diversi che si ripete in maniera abbastanza regolare, è necessaria una pre-elaborazione al fine di trasformare le texture in regioni uniformi. Si può poi procedere con la segmentazione attraverso una delle precedenti tecniche.

1.2. Obbiettivi del progetto

Lo scopo di questo progetto è confrontare diversi algoritmi di segmentazione di immagini a colori. Come detto, nel corso degli ultimi anni sono stati proposte numerose tecniche diverse tra loro, ma ciascuna comunque riferibile ad un determinato filone. Tali tecniche possono essere *"supervised"*, in quanto necessitano di una corretta impostazione di parametri, diversi a seconda dell'immagine trattata, o *"unsupervised"*, che non hanno tale bisogno.

Abbiamo confrontato quattro algoritmi di segmentazione, due supervised e due unsupervised. I termini di paragone scelti sono i seguenti:

- l'*efficacia*, ovvero in che misura la segmentazione risulta 'buona'; in effetti è difficile valutare correttamente l'efficacia della segmentazione, a meno che non si abbia a che fare con immagini particolari; invece, data un'immagine generica non è possibile stabilire a priori quale sia la segmentazione ottima, anche perché dipende dal task che la richiede; qualche autore ha comunque cercato di fornire delle linee guida generali che definiscono le caratteristiche di una buona segmentazione;
- l'*efficienza*, ovvero le prestazioni dell'algoritmo in termini di tempo di esecuzione;
- l'*omogeneità* dei risultati, ovvero se una tecnica è applicabile con la stessa efficacia a diversi tipi di immagine.

Tali algoritmi sono stati provati su diverse immagini, tra le quali abbiamo individuato tre categorie:

- immagini ricavate dalla navigazione di un robot nello stesso ambiente con diversi illuminanti;

- color chart, ovvero immagini con gamme di colori particolarmente critiche ai fini della segmentazione;
- immagini generiche, tra cui alcune usate in letteratura, per verificare l'efficacia in un contesto reale.

2. Descrizione degli algoritmi analizzati

Abbiamo scelto di testare quattro algoritmi di segmentazione, due dei quali di tipo supervised e due di tipo unsupervised. I primi due sono basati sul region growing, il terzo sul clustering e il quarto sul trattamento delle texture.

In questo capitolo verrà analizzato a grandi linee il principio di funzionamento di ciascuno di essi.

2.1. Algoritmo di Adami e Mosma

L'algoritmo di Adami e Mosma è parte del loro progetto riguardante l'autolocalizzazione dei robot autonomi [Adami,1998]. Esso è basato sulla tecnica region growing, quindi produce regioni connesse. A causa della particolare applicazione di questo algoritmo, non tutti i pixel vengono associati ad una regione.

Essendo la velocità di esecuzione un fattore critico nell'ambito della navigazione autonoma, è stata fatta la scelta di rappresentare i pixel nello spazio RGB, che è lo spazio usato per la memorizzazione delle immagini. Ciò porta a risultati soddisfacenti, magari con un leggero sforzo aggiuntivo nella scelta dei parametri rispetto ad altri spazi. Inoltre, siccome le regioni troppo piccole o troppo grandi non forniscono informazioni sufficienti al problema dell'autolocalizzazione, esse vengono scartate per diminuire il carico computazionale.

La scelta del seme è fatta per via esaustiva, in quanto vengono fatti scorrere tutti i pixel dell'immagine fino a trovare il primo punto non ancora assegnato. Una volta scelto il seme, si procede alla definizione della regione controllando il vicinato di ordine 8 del seme, e i vicinati successivi. A questo scopo, viene introdotta una misura di somiglianza tra pixel, la distanza euclidea. Se non c'è troppa differenza tra il colore medio della regione e il colore del pixel di cui si sta valutando l'opportunità dell'accorpamento, il pixel apparterrà alla regione. In realtà si tiene anche in considerazione la differenza di colore tra pixel attuale e seme. Il compito di definire una soglia entro la quale un pixel appartiene alla regione è lasciato all'utente.

Non vengono tenute in considerazione le texture, perché è stata preferita la velocità di esecuzione.

2.2. Algoritmo di Pujas e Aldon implementato da Lazzaroni e Roncati

L'algoritmo di Pujas e Aldon [Pujas,1995] è stato implementato da Alberto Lazzaroni e Giorgio Roncati [Lazzaroni,1999] nell'ambito del progetto di robotica. Esso si basa sulla tecnica region growing ed è supervised. Inoltre è robusto rispetto alla variazione delle condizioni di illuminazione della scena. A tal fine, opera nello spazio HSV (Hue, Saturation, Value).

L'immagine da segmentare può presentare, sotto cattive condizioni di luce, delle regioni dette acromatiche. Per ogni pixel P, quindi, viene definito un tasso di cromaticità c(P); a seconda del suo valore, si tiene in maggior considerazione l'intensità V piuttosto che la tinta H.

Si possono individuare i tre passi principali dell'algoritmo:
1. graph computation
2. growing process
3. over-merging

La graph computation consiste in una fase di preprocessing per il calcolo delle somiglianze tra pixel. Per ogni coppia di pixel adiacenti viene computata una funzione di somiglianza basata su: tasso di cromaticità c(P) e di acromaticità 1-c(P) e due soglie H e V che rappresentano rispettivamente la massima distanza ammissibile in tinta e intensità tra due pixel simili, e che devono essere impostate dall'utente.

Il growing process consiste nell'espansione di una regione a partire da un pixel seme. Viene introdotta una nuova funzione di somiglianza tra pixel e regione, del tutto simile alla precedente, ma in cui vengono associati ad ogni regione i propri valori medi di H, S e V. Le nuove soglie di somiglianza sono denominate Hr e Vr, e devono essere impostate dall'utente. Il procedimento è il seguente: si parte da un pixel seme scelto casualmente e, se i suoi vicini sono sufficientemente simili, vengono accorpati al seme.

Dopo la fase di growing di solito l'immagine è sovrasegmentata, sia per una cattiva scelta dei semi, sia per il rumore contenuto nell'immagine. L'over-merging viene quindi effettuato per eliminare le regioni troppo piccole, che vengono associate alla regione più somigliante tra quelle adiacenti. Le regioni non accorpabili vengono scartate. Questa fase necessita dell'impostazione di due soglie di somiglianza tra regioni, Hrr e Vrr, da parte dell'utente.

2.3. Algoritmo di Dorin Comaniciu e Peter Meer

Comaniciu e Meer [Comaniciu,1997] presentano una tecnica generale per individuare le caratteristiche significative di un'immagine. Esso si basa sul metodo del clustering: data un'immagine, ad ogni pixel viene associato un vettore di caratteristiche che viene mappato in uno spazio delle caratteristiche. Le regioni ad alta densità in tale spazio corrispondono alle caratteristiche significative dell'immagine. Comunque, durante tutto il procedimento, non si utilizzano solo i vettori di caratteristiche, ma anche le informazioni nel dominio dell'immagine, allo scopo di tenere conto anche dei vincoli spaziali. Inoltre, il metodo è applicabile a qualsiasi spazio delle caratteristiche.

Per determinare le regioni ad alta densità viene impiegata una finestra di ricerca caratterizzata da quanti meno parametri possibile: nel caso migliore, in cui lo spazio delle caratteristiche è isotropico, essa è una sfera. Il metodo utilizzato per posizionare tale finestra, chiamato *mean shift algorithm*, si basa sulle seguenti considerazioni. Se, all'interno di uno spazio delle caratteristiche isotropico, caratterizzato da una funzione di densità di probabilità unimodale p(x), si considera una sfera S_X di raggio r centrata in x, essa contiene tutti i vettori y tali che $\|y - x\| \le r$. Definendo z=y-x e $\mu = E[z/S_x] = E[x/x \in S_x] - x$, ovvero μ=differenza tra media locale e centro della sfera, si può dimostrare che μ è direttamente proporzionale al gradiente di p(x), e inversamente a p(x)[*]. Le regioni ad alta densità hanno p(x) elevati e $\nabla p(x)$ bassi, quindi sono quelle in cui lo shift medio μ è molto piccolo. Quindi, il *mean shift algorithm* funziona nel seguente modo:

1. Scelta del raggio r della finestra di ricerca.
2. Scelta della locazione iniziale della finestra.
3. Computazione del vettore di shift medio e traslazione del centro della finestra di una misura pari a tale valore.
4. Ripetizione del passo 3 fino alla convergenza.

Questo algoritmo funziona però con funzioni di densità di probabilità unimodali. Per eliminare questo vincolo la locazione iniziale della finestra viene scelta a caso, e l'algoritmo trova la regione ad alta densità più vicina; esso viene poi applicato ad un certo numero di finestre di ricerca. Inoltre, ogni regione trovata deve essere convalidata dai vincoli derivanti dal dominio dell'immagine. Il procedimento generale consiste quindi nei seguenti passi:

1. Mappare il dominio dell'immagine nello spazio delle caratteristiche.
2. Definire un numero adeguato di finestre di ricerca centrate in posizioni casuali nello spazio.

[*] Per la dimostrazione si rimanda all'articolo di Comaniciu e Meer.

3. Trovare i centri delle regioni ad alta densità attraverso il *mean shift algorithm* applicato ad ogni finestra.
4. Convalidare le regioni estratte attraverso i vincoli derivanti dal dominio dell'immagine, per costruire la palette delle caratteristiche.
5. Associare, utilizzando le informazioni nel dominio dell'immagine, a tutti i vettori di caratteristiche il proprio elemento della palette.

Come si può notare, il metodo descritto può essere applicato ad ogni spazio delle caratteristiche. Per il problema della segmentazione di immagini a colori, la tecnica deve però essere maggiormente specificata, e contenere numerose procedure ad-hoc. Ad esempio, viene utilizzato un criterio per definire la somiglianza tra colori.

A seconda dello scopo della segmentazione, è stato introdotto un parametro: la risoluzione della segmentazione. È possibile scegliere tra tre classi: sottosegmentazione, sovrasegmentazione, quantizzazione. La prima ha un ampio margine di tolleranza per la somiglianza tra colori, quindi nella palette sono presenti solo i colori principali, e vengono riconosciuti solo i principali oggetti dell'immagine. La seconda ha un margine di tolleranza minore, quindi le regioni risultanti sono molte e piccole, e dovranno essere assemblate in una seconda fase con una base di conoscenza più ampia; l'obiettivo è il riconoscimento di oggetti. La terza è quella con la risoluzione più alta: l'immagine segmentata contiene tutti i colori importanti nell'immagine; si usa per query content-based. Per i nostri scopi, l'unica che fornisce buoni risultati si è rivelata la sottosegmentazione. Avendo imposto questa scelta, l'algoritmo risulta totalmente unsupervised.

Le immagini sono di solito memorizzate nello spazio RGB, ma, per garantire l'isotropia dello spazio delle caratteristiche, si è scelto il L*u*v*, che è legato al RGB da trasformazioni non lineari. L'illuminante scelto è il daylight standard D_{65}. Lo spazio L*u*v* non è perfettamente isotropico, ma è comunque soddisfacente.

Vengono di seguito riportati tutti i passi dell'algoritmo.

1. *Definizione dei parametri della segmentazione.* A seconda della classe di segmentazione scelta, l'algoritmo imposta: il raggio r della finestra di ricerca, il minimo numero di elementi N_{min} richiesti per avere un colore significativo, il numero minimo di pixel contigui N_{con} richiesti per avere una regione significativa. In particolare, il raggio non dipende unicamente dalla classe di segmentazione, ma anche dall'"attività visiva" dell'immagine: infatti un'immagine con larghe regioni omogenee deve essere trattata con risoluzione più alta, quindi con raggio minore, rispetto ad un immagine con molte aree *textured*. A questo scopo, si considera la matrice di covarianza globale, si valuta la varianza σ e si prende r proporzionale a σ, in modo tale da avere raggi più grandi se l'immagine è poco omogenea.
2. *Definizione della finestra di ricerca.* La finestra viene scelta a caso, ma vengono esaminati più candidati per avere la certezza che essa si trovi vicina ad una

regione significativa. La scelta casuale è fatta nel dominio dell'immagine, considerando qualche piccolo gruppo di pixel con i propri vicini, mappando tali pixel nello spazio delle caratteristiche, e analizzando se i vicini appartengono ad una regione più ampia: in tal caso, con molta probabilità, la locazione della finestra è quella voluta.

3. *Mean shift algorithm.* Viene applicato alla finestra scelta, in modo da posizionarla sulla regione ad alta densità.
4. *Rimozione della finestra trovata.* I pixel appartenenti alla finestra trovata sono rimossi sia dallo spazio delle caratteristiche che dal dominio dell'immagine. Anche gli 8 vicini di tali pixel, indipendentemente dalle loro caratteristiche, sono rimossi dal dominio dell'immagine, perché di solito hanno colori "strani".
5. *Iterazioni.* Si ripetono i passi 2-3-4 fino a quando il numero di vettori di caratteristiche appartenenti alla finestra scelta diventa minore di N_{min}.
6. *Determinazione della palette iniziale.* Una regione è significativa se è basata su almeno N_{min} vettori e, allo stesso modo, se esistono almeno N_{min} pixel connessi che sono rappresentati da essa. Vengono quindi eliminate le regioni che non sottostanno a queste condizioni.
7. *Determinazione della palette finale.* Tutti i pixel vengono associati ad un colore della palette determinata in precedenza. Innanzitutto, i pixel appartenenti ad una regione ad alta densità sono associati al colore del centro della finestra. Poi le finestre assumono volume doppio. I nuovi pixel incorporati nelle finestre vengono associati al colore del centro della finestra solo se hanno almeno un vicino già associato ad essa. La media delle caratteristiche dei pixel associati ad una finestra diviene il loro colore rappresentativo ed è contenuto nella palette finale. Alla fine, i pochi pixel ancora non classificati sono associati al colore più vicino della palette finale.
8. *Postprocessing.* Consiste nella rimozione di tutte i gruppi di pixel di dimensione minore di N_{con}. Tali pixel sono associati al colore maggioritario tra i vicini; in caso di parità, al colore più simile nello spazio delle caratteristiche.

2.4. Algoritmo di Deng, Manjunath, Shin

Questa tecnica di segmentazione [Deng,1999], chiamata JSEG dai suoi autori, nasce con lo scopo di realizzare un algoritmo automatico che funzioni bene su di una grande varietà di immagini. Per automatico si intende dire che non sarà necessario tarare alcun parametro per ciascuna immagine di cui si desidera la segmentazione.

La segmentazione risulta particolarmente ostica in presenza di immagini di tipo *textured*. Un'area di un'immagine è textured quando il suo colore non è omogeneo ma presenta una successione di colori diversi che si ripete in maniera abbastanza regolare; a titolo di esempio si può citare il tetto di un'abitazione, un terreno ghiaioso o le venature del legno di un mobile. Si può quindi affermare che l'algoritmo si basa

sull'elaborazione di texture e, come spiegato in seguito, su una successiva estrazione di contorni.

Gli autori nel loro approccio fanno alcune assunzioni:
- ogni regione dell'immagine contiene un texture pattern uniformemente distribuito;
- l'informazione in termini di colori di ciascuna regione dell'immagine può essere rappresentata da pochi colori quantizzati
- i colori di due regioni vicine devono essere distinguibili.

L'algoritmo procede ad un partizionamento dell'immagine, fatto in maniera di minimizzare una certa funzione di costo. Al fine di completare il processo di minimizzazione, vengono generate delle mappe dette *J-images*, che misurano le disomogeneità dell'immagine su scale di dimensioni differenti. Ma passiamo ad analizzare in dettaglio l'algoritmo JSEG.

Il primo passo consiste in una quantizzazione dei colori dell'immagine. Dopo questa operazione, ad ogni colore viene assegnata una label. Si definisce il concetto di *classe di colore*, che è l'insieme dei pixel dell'immagine che dopo la quantizzazione risultano avere lo stesso colore. A questo punto i pixel dell'immagine vengono sostituiti con le labels delle corrispondenti classi di colore. Questa nuova immagine fatta di labels viene detta *mappa delle classi*.

Viene quindi definita una misura J, che dà un indice della distribuzione delle labels nel piano bidimensionale dell'immagine. Se l'immagine è costituita da più regioni omogenee, il valore di J sarà alto; se invece le labels sono uniformemente sparse per l'immagine, J tenderà ad essere nullo. Nel primo caso si può supporre che una buona segmentazione consista nell'assegnare una regione a ciascuna delle aree omogenee, mentre nel secondo caso la scelta è una sola grossa regione in quanto essa è di per sé omogenea.

Si definisce quindi una nuova misura J', costruita sulla base di J, nella quale si considera una particolare segmentazione dell'immagine. J' si calcola come media dei J delle singole regioni.

La segmentazione 'ottima' sarà quindi quella a cui sarà associato il minimo valore J'. Purtroppo la minimizzazione di J' sull'intera immagine è un problema troppo complesso. Tuttavia si può notare che la misura J, applicata ad un'area circolare dell'immagine, indica se quell'area appartiene al centro della regione o se piuttosto è localizzata al confine tra due regioni.

Definiamo ora la *J-image* come un'immagine a toni di grigio in cui i valori di ciascun pixel sono il valore di J, calcolato su di un'area centrata in quel pixel, che

chiameremo d'ora in poi valore locale di J. Quindi questi valori sono più alti in corrispondenza dei confini tra le regioni.

L'algoritmo JSEG genera più J-images, diverse solo per la scelta del raggio dell'area su cui calcolare J. In questo modo sono ricavati confini su diverse scale di grandezza.

Sulla base della J-image a scala più grande si determinano i minimi valori locali di J. Questa è un'operazione delicata in quanto il numero di minimi scelti è determinante ai fini del risultato finale; il problema viene risolto calcolando media e varianza dei valori locali di J dell'insieme dei punti di minimo. L'insieme dei punti di minimo viene incrementato finché la varianza non oltrepassa una certa soglia, fissata dagli autori.

I punti dell'insieme così definito individuano piccole aree attorno ad essi, dette *valli*. Le regioni della segmentazione finale vengono cresciute attorno alle valli pixel per pixel. Ecco le principali regole usate in questa fase:

- togliere eventuali buchi nelle valli accresciute
- calcolare i valori locali di J nelle parti non assegnate ad alcuna regione ed assegnare il pixel alla regione adiacente col valore di J più prossimo
- passare a J-images a scala più piccola e ripetere l'operazione.

Dopo questo passaggio l'immagine ottenuta è solitamente sovrasegmentata, ed alcune regioni vengono unite in base alla somiglianza dei colori: si crea cioè una tabella da cui si ricava la coppia di regioni a somiglianza massima e le si unisce. Si continua fino ad arrivare ad una certa soglia di somiglianza.

Questa soglia ed altri parametri dell'algoritmo sono stati ottimizzati dagli autori e producono una segmentazione ugualmente buona su una vasta gamma di immagini. È tuttavia possibile alterare manualmente tali parametri.

3. Implementazione e adattamento degli algoritmi

Uno dei nostri compiti è stato quello di arricchire l'implementazione esistente degli algoritmi per consentirne una valutazione dell'efficacia e dell'efficienza. A tal fine abbiamo operato le seguenti modifiche:

- Uniformazione dell'interfaccia, allo scopo di lanciare l'esecuzione degli algoritmi nello stesso modo, ove possibile.
- Misura dei tempi di elaborazione.
- Uniformazione dell'output. In generale, l'output fornito da un algoritmo di segmentazione può essere uno dei seguenti:
    - una serie di immagini, ciascuna delle quali riporta una delle regioni ottenute;
    - un'immagine in cui i pixel di una regione assumono un colore uniforme;
    - un'immagine in cui sono segnati i confini tra le regioni.

  Siccome i quattro algoritmi fornivano inizialmente output diversi, abbiamo deciso di uniformarli in modo che tutti producessero almeno gli ultimi due elencati.
- Uniformazione dei tipi di dati, in modo che tutti gli algoritmi funzionassero anche con immagini piuttosto grandi.
- Generazione di report, contenenti informazioni sull'immagine in ingresso, sui tempi di esecuzione, sul numero e sulle caratteristiche delle regioni individuate.

Tutte le immagini usate nei test, i report generati, le immagini prodotte, e i sorgenti degli algoritmi sono contenuti nel CD accluso. Di seguito verranno analizzati l'implementazione degli algoritmi, il loro adattamento alle nostre necessità e verranno fornite le modalità d'uso.

3.1. Algoritmo di Adami e Mosma

L'algoritmo di Adami e Mosma, sviluppato in ambiente Linux, è utilizzato all'interno del progetto di autolocalizzazione dei robot autonomi. Esso produce una matrice, detta di copertura, in cui ad ogni pixel viene associata la rispettiva regione di appartenenza. In base a tale matrice, abbiamo arricchito il codice, in modo da generare un'immagine nel formato a regioni di colore uniforme. Il formato elettronico delle immagini, sia di ingresso che di uscita, può essere ppm, pgm, tif, tng, xpm, jpeg o eim, in quanto sfrutta la libreria grafica Imlib (si veda [Imlib]).

Inoltre, per i nostri scopi, l'algoritmo è stato isolato ed è stato reso stand-alone. Per semplificare l'implementazione e la leggibilità, abbiamo deciso di non utilizzare alcune classi C++ che erano state previste originariamente per il trattamento delle matrici. Il comando da eseguire è il seguente:

```
segm imm_in.ext imm_out.ext soglia min_pixel max_pixel
```

in cui:

- `imm_in` e `imm_out` sono i nomi delle immagini di ingresso e di uscita; `imm_out` avrà il formato a regioni di colore uniforme; per produrre il formato a contorni è necessario eseguire il programma `edges`, spiegato in seguito;
- `soglia` è il valore di somiglianza minimo tra pixel della stessa regione;
- `min_pixel` e `max_pixel` sono il numero minimo e massimo di pixel di una regione.

Per i nostri scopi, a `max_pixel` assegneremo un valore grande, in modo che non vi sia un limite massimo per il numero di pixel appartenenti ad una regione. Quindi, solo `min_pixel` influirà sui risultati; i pixel di una regione di cardinalità non sufficiente rimarranno inclassificati. Nell'output a regioni di colore uniforme un colore grigio (R=128, G=128, B=128) indica che il pixel non è stato assegnato ad alcuna regione. Il programma `edges` tratterà le aree grigie come normali regioni.

Abbiamo notato che, per problemi di rumore dell'immagine, alcune regioni risultano avere dei piccoli 'buchi' di pochi pixel; in effetti, per gli scopi a cui era preposto, essi erano ininfluenti; invece, per una migliore segmentazione, potrebbe essere utile un algoritmo basato sul '*majority game*' proposto in [Von Neumann,1944], di cui proponiamo una implementazione nel cd accluso. Esso, in sostanza, assegna un pixel alla regione maggioritaria tra i pixel vicini. È però da notare che esso rallenta l'esecuzione dell'algoritmo. Essendo spesso numerosi i pixel non assegnati a una regione, è opportuno considerarli come un'ulteriore regione ai fini dell'esecuzione di questo algoritmo.

3.2. Algoritmo di Pujas e Aldon implementato da Lazzaroni e Roncati

Originalmente, l'implementazione di questo algoritmo, sviluppato in ambiente Linux, produceva una serie di immagini, chiamate immseg00, immseg01, …, ciascuna delle quali conteneva una regione. Abbiamo quindi apportato delle modifiche in modo da produrre l'immagine di output in formato a regioni di colore uniforme; in tale formato un colore grigio indica che il pixel non appartiene ad alcuna regione; tramite il lancio successivo del programma `edges` si ottiene il formato a contorni.

Tale applicazione considera le aree grigie come regioni vere e proprie. Anche questo algoritmo sfrutta la libreria Imlib, quindi accetta i formati indicati al paragrafo 4.1.

I parametri dell'algoritmo, come indicato in [Lazzaroni,1999], devono essere specificati nel file valori.dat, che deve risiedere nella stessa directory dell'eseguibile, nell'ordine seguente:

1. *Soglia su db*: rappresenta la somiglianza minima tra un pixel e una regione affinché pixel possa essere considerato appartenente alla regione. Variabile in [0.5,1[.
2. *Soglia su H tra pixel*: rappresenta la massima differenza tra le tinte di due pixel simili. Variabile in [0,6[.
3. *Soglia su V tra pixel*: rappresenta la massima differenza tra le intensità di due pixel simili. Variabile in [0,255].
4. *Soglia su Hrp tra regione e pixel*: rappresenta la massima differenza tra la tinta di un pixel e la media delle tinte dei pixel di una regione. Variabile in [0,6[
5. *Soglia su Vrp tra regione e pixel*: rappresenta la massima differenza tra l'intensità di un pixel e la media delle intensità dei pixel di una regione. Variabile in [0,255].
6. *Compressione del cp*: consente di controllare la pendenza della superficie che rappresenta il tasso di cromaticità (figura 3-1). Variabile in [0,1].
7. *Numero minimo di pixel di una regione da accorpare*: area al di sotto della quale una regione viene considerata troppo piccola e quindi accorpabile.
8. *Soglia su Hrr tra regioni*: rappresenta la massima differenza tra le tinte di due regioni simili. Variabile in [0,6[.
9. *Soglia su Vrr tra regioni*: rappresenta la massima differenza tra le intensità di due regioni simili. Variabile in [0,255].
10. *Soglia massima su l_ir tra regioni*: somiglianza minima tra due regioni affinché una sia accorpabile all'altra. Variabile in [0,1].

Il comando da eseguire è il seguente:

```
indici imm_in.ext imm_out.ext
```

in cui `imm_in` e `imm_out` sono i nomi delle immagini di ingresso e uscita.

3.3. Algoritmo di Comaniciu e Meer

Questo algoritmo, il cui codice ci è stato reso disponibile in ambiente Linux, è stato probabilmente sviluppato in C++ sotto un altro sistema operativo di tipo Unix, e quindi ha richiesto la correzione di qualche problema di portabilità verso Linux. Inoltre, originariamente, l'output era un'immagine con formato a regioni di colore uniforme e un'immagine in bianco e nero in cui comparivano i soli contorni. Per produrre il formato in cui i contorni vengono tracciati sull'immagine di partenza, è necessario utilizzare il programma `edges`.

L'algoritmo richiede e produce solamente immagini in formato ppm. E' possibile sfruttare il programma `convert` presente in molte distribuzioni di Linux per convertire le immagini in tale formato.

Dopo la modifica dell'interfaccia, il comando da eseguire è il seguente:

```
segm imm_in.ppm imm_out.ppm
```

in cui `imm_in` e `imm_out` sono i nomi delle immagini di ingresso e uscita. Viene prodotto anche un file `imm_out.pgm` contenente i soli contorni.

3.4. Algoritmo di Deng, Manjunath e Shin

L'implementazione di questo algoritmo è disponibile solamente in versione eseguibile in ambiente Solaris, SGI, o Windows; i nostri test sono stati effettuati in quest'ultimo ambiente. Al fine di misurare i tempi e di rendere l'utilizzo il più uniforme possibile rispetto agli algoritmi precedenti, abbiamo realizzato una applicazione esterna sviluppata in linguaggio C. Essa consente di lavorare su molti formati di immagine, sfruttando la versione Windows del programma `convert`. L'output prodotto dall'algoritmo originale è quello a contorni sovrapposti all'immagine d'ingresso. La nostra applicazione, basandosi sui risultati di tale algoritmo, produce anche il formato a regioni di colore uniforme.

L'esecuzione dell'algoritmo viene lanciata tramite il comando:

```
jseg imm_in.ext imm_out.ext
```

in cui `imm_in` e `imm_out` sono i nomi delle immagini di ingresso e uscita.

3.5. L'applicazione 'edges'

L'applicazione `edges` da noi sviluppata consente di convertire un'immagine nel formato a regioni di colore uniforme in una nel formato a contorni sovrapposti all'originale. Essa funziona in ambiente Linux e necessita della libreria Imlib, e quindi supporta i formati indicati nel paragrafo 4.1.

Il comando da eseguire è il seguente:

```
edges imm_input.ext imm_uniforme.ext imm_contorni.ext
```

in cui:
- `imm_input` è il nome dell'immagine sottoposta a segmentazione;

- `imm_uniforme` è il nome dell'immagine risultato della segmentazione in formato a regioni di colore uniforme;
- `imm_contorni` è il nome dell'immagine in formato a contorni sovrapposti all'originale che verrà prodotta da questa applicazione.

4. Testing degli algoritmi

4.1. Condizioni di test

Tutti gli algoritmi sono stati provati su un PC dotato di processore Pentium II, con clock a 350 Mhz, e con 64Mb di RAM.

La distribuzione Linux utilizzata per i primi tre algoritmi, ovvero quello di Adami e Mosma, quello di Pujas e Aldon, e quello di Comaniciu e Meer, è la RedHat 6.1.

Per il quarto algoritmo, ovvero quello di Deng, Manjunath e Shin, è stato utilizzato il sistema operativo Windows 98.

4.2. Conteggio delle regioni

Per una corretta lettura dei risultati, è necessario tenere in considerazione le diverse scelte effettuate dagli autori in tal senso.

Per l'algoritmo di Comaniciu e Meer una regione è costituita da gruppi di pixel simili anche non contigui, mentre per gli altri algoritmi una regione è sempre un'area connessa.

Inoltre, per quanto riguarda gli algoritmi di Adami-Mosma e Pujas-Aldon, non tutti i pixel vengono assegnati ad una regione. Nell'output a regioni di colore uniforme un colore grigio (R=128, G=128, B=128) indica che il pixel non è stato assegnato ad alcuna regione. Il programma `edges` tratterà le aree grigie come normali regioni.

4.3. Test su immagini di navigazione

Sono riportati di seguito i risultati della segmentazione applicati a cinque fotografie dello stesso ambiente in diverse condizioni di illuminazione.

Per i due algoritmi supervised, abbiamo deciso di impostare gli stessi parametri per tutte le cinque immagini, in modo da verificare l'indipendenza della tecnica di

segmentazione dalle condizioni di illuminazione. Tali parametri sono di seguito riportati:

- Algoritmo di Adami e Mosma

Soglia:	100
Numero minimo pixel:	400
Numero massimo pixel:	300000

- Algoritmo di Pujas e Aldon

Soglia su db:	0.77
Soglia su H tra pixel:	3.5
Soglia su V tra pixel:	35
Soglia su Hrp tra regione e pixel:	3
Soglia su Vrp tra regione e pixel:	25
Compressione del cp:	0.25
Numero minimo di pixel di una regione di pixel da accorpare:	1000
Soglia su Hrr tra regioni:	5.6
Soglia su Vrr tra regioni:	30
Soglia massima su l_ir tra regioni:	0.2

File di origine:
navigazione/originali/ing-day.tif

Numero colori: 30127

Dimensioni : 384 x 256

Adami e Mosma

Numero regioni: 17
Tempo impiegato: 0.3791 s
Throughput: 259339 pixel/s

Pujas e Aldon

Numero regioni: 17
Tempo impiegato: 17.8124 s
Throughput: 5519 pixel/s

Comaniciu e Meer

Numero regioni: 12
Tempo impiegato: 3.0861 s
Throughput: 31854 pixel/s

Deng, Manjunath e Shin

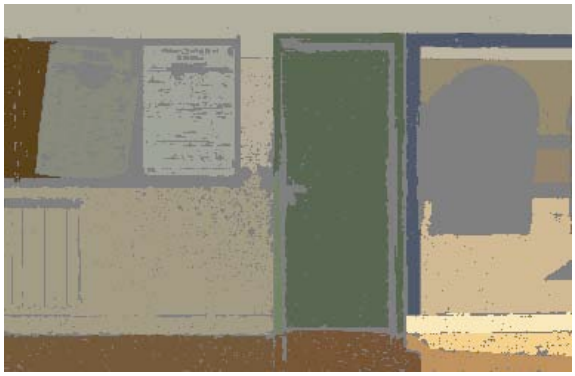
Numero regioni: 39
Tempo impiegato: 76.1200 s
Throughput: 1291 pixel/s

File di origine:
navigazione/originali/ing-flash.tif

Numero colori: 32958

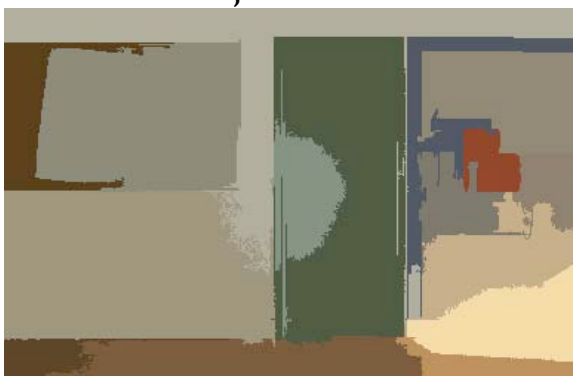
Dimensioni : 384 x 256

Adami e Mosma



Numero regioni: 19
Tempo impiegato: 0.3651 s
Throughput: 269269 pixel/s

Pujas e Aldon

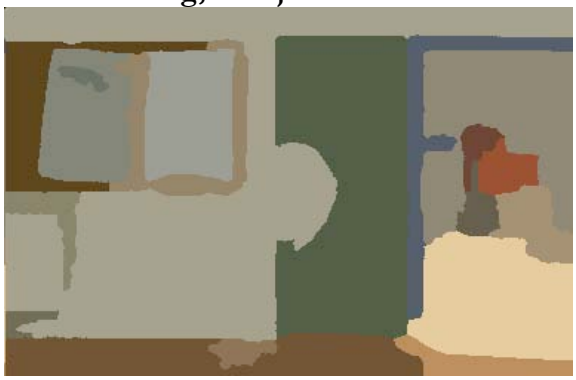


Numero regioni: 16
Tempo impiegato: 18.7277 s
Throughput: 5249 pixel/s

Comaniciu e Meer

Numero regioni: 14
Tempo impiegato: 3.4441 s
Throughput: 28542 pixel/s

Deng, Manjunath e Shin



Numero regioni: 19
Tempo impiegato: 64.3200 s
Throughput: 1528 pixel/s

File di origine:
navigazione/originali/ing-fluo.tif

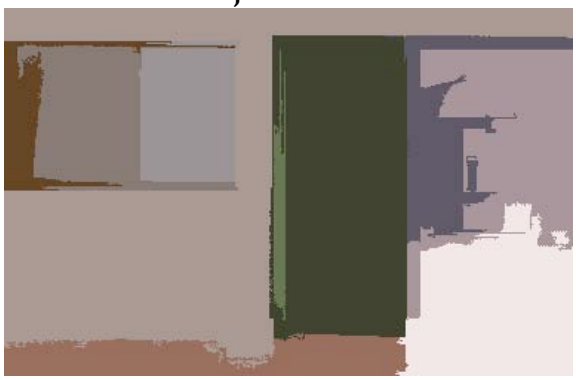
Numero colori: 33142

Dimensioni : 384 x 256

Adami e Mosma

Numero regioni: 16
Tempo impiegato: 0.3679 s
Throughput: 267215 pixel/s

Pujas e Aldon



Numero regioni: 10
Tempo impiegato: 17.9397 s
Throughput: 5480 pixel/s

Comaniciu e Meer

Numero regioni: 11
Tempo impiegato: 3.0472 s
Throughput: 32260 pixel/s

Deng, Manjunath e Shin

Numero regioni: 28
Tempo impiegato: 66.3500 s
Throughput: 1482 pixel/s

File di origine:
navigazione/originali/ing-off.tif

Numero colori: 29263

Dimensioni : 384 x 256

Adami e Mosma

Numero regioni: 19
Tempo impiegato: 0.3616 s
Throughput: 271836 pixel/s

Pujas e Aldon

Numero regioni: 19
Tempo impiegato: 17.3780 s
Throughput: 5657 pixel/s

Comaniciu e Meer

Numero regioni: 11
Tempo impiegato: 2.8625 s
Throughput: 34341 pixel/s

Deng, Manjunath e Shin

Numero regioni: 32
Tempo impiegato: 76.4000 s
Throughput: 1287 pixel/s

File di origine:
navigazione/originali/ing-tung.tif

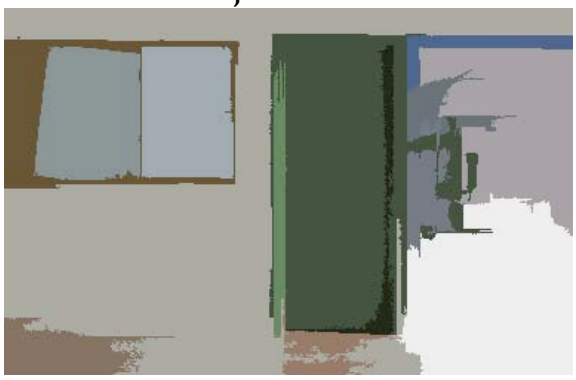
Numero colori: 34501

Dimensioni : 384 x 256

Adami e Mosma

Numero regioni: 18
Tempo impiegato: 0.3664 s
Throughput: 268312 pixel/s

Pujas e Aldon

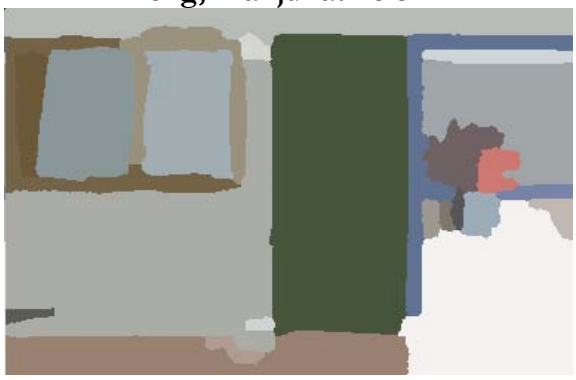


Numero regioni: 14
Tempo impiegato: 17.0300 s
Throughput: 5772 pixel/s

Comaniciu e Meer

Numero regioni: 11
Tempo impiegato: 3.1706 s
Throughput: 31005 pixel/s

Deng, Manjunath e Shin



Numero regioni: 25
Tempo impiegato: 56.4600 s
Throughput: 1741 pixel/s

4.4. Test su colorchart

Sono riportati di seguito i risultati della segmentazione applicati a dieci colorchart. Essi sono immagini con gamme di colori particolarmente critiche ai fini della segmentazione.

Per i due algoritmi supervised, abbiamo deciso di impostare gli stessi parametri per tutte le dieci immagini, in modo da verificare il funzionamento con lo stesso tipo di immagine. Tali parametri sono di seguito riportati:

- Algoritmo di Adami e Mosma

Soglia:	20
Numero minimo pixel:	1000
Numero massimo pixel:	300000

- Algoritmo di Pujas e Aldon

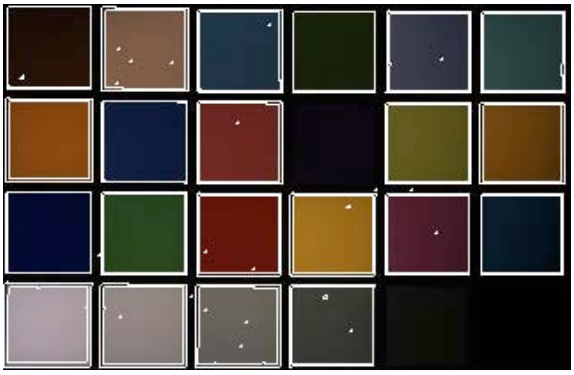
Soglia su db:	0.95
Soglia su H tra pixel:	0.9
Soglia su V tra pixel:	255
Soglia su Hrp tra regione e pixel:	0.9
Soglia su Vrp tra regione e pixel:	255
Compressione del cp:	0.95
Numero minimo di pixel di una regione di pixel da accorpare:	1500
Soglia su Hrr tra regioni:	0.9
Soglia su Vrr tra regioni:	255
Soglia massima su l_ir tra regioni:	0.5

File di origine:
colorchart/originali/a_gamma1.tif

Numero colori: 12868

Dimensioni : 320 x 213

Adami e Mosma



Numero regioni: 22
Tempo impiegato: 0.2489 s
Throughput: 273875 pixel/s
Quadrati riconosciuti: 21
Quadrati parzialmente riconosciuti: 0

Pujas e Aldon

Numero regioni: 23
Tempo impiegato: 26.4697 s
Throughput: 2575 pixel/s
Quadrati riconosciuti: 16
Quadrati parzialmente riconosciuti: 6

Comaniciu e Meer

Numero regioni: 18
Tempo impiegato: 1.8128 s
Throughput: 37599 pixel/s
Quadrati riconosciuti: 22
Quadrati parzialmente riconosciuti: 0

Deng, Manjunath e Shin

Numero regioni: 21
Tempo impiegato: 36.9100 s
Throughput: 1847 pixel/s
Quadrati riconosciuti: 18
Quadrati parzialmente riconosciuti: 2

File di origine:
colorchart/originali/a10_gamma1.tif

Numero colori: 17338

Dimensioni : 320 x 211

Adami e Mosma

Numero regioni: 24
Tempo impiegato: 0.2402 s
Throughput: 281079 pixel/s
Quadrati riconosciuti: 23
Quadrati parzialmente riconosciuti: 0

Pujas e Aldon

Numero regioni: 19
Tempo impiegato: 21.0420 s
Throughput: 3209 pixel/s
Quadrati riconosciuti: 17
Quadrati parzialmente riconosciuti: 1

Comaniciu e Meer

Numero regioni: 16
Tempo impiegato: 1.9137 s
Throughput: 35283 pixel/s
Quadrati riconosciuti: 23
Quadrati parzialmente riconosciuti: 0

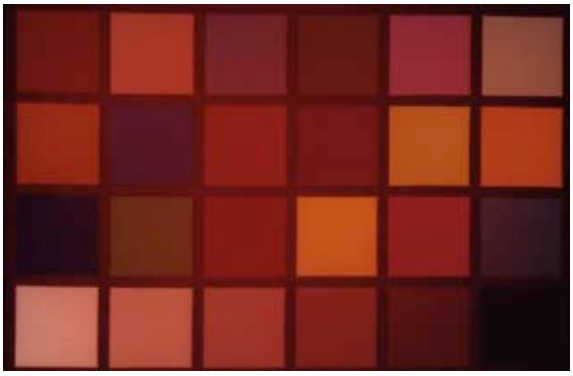
Deng, Manjunath e Shin

Numero regioni: 23
Tempo impiegato: 33.7200 s
Throughput: 1875 pixel/s
Quadrati riconosciuti: 19
Quadrati parzialmente riconosciuti: 2

File di origine:
colorchart/originali/a3_gamma1.tif

Numero colori: 10470

Dimensioni : 320 x 211



Adami e Mosma

Numero regioni: 22
Tempo impiegato: 0.2384 s
Throughput: 283248 pixel/s
Quadrati riconosciuti: 21
Quadrati parzialmente riconosciuti: 0

Pujas e Aldon

Numero regioni: 13
Tempo impiegato: 29.8478 s
Throughput: 2262 pixel/s
Quadrati riconosciuti: 9
Quadrati parzialmente riconosciuti: 3

Comaniciu e Meer

Numero regioni: 15
Tempo impiegato: 1.6238 s
Throughput: 41582 pixel/s
Quadrati riconosciuti: 16
Quadrati parzialmente riconosciuti: 6

Deng, Manjunath e Shin

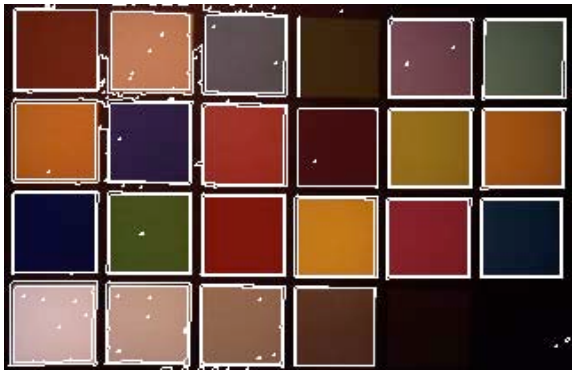
Numero regioni: 20
Tempo impiegato: 34.2700 s
Throughput: 1970 pixel/s
Quadrati riconosciuti: 13
Quadrati parzialmente riconosciuti: 3

File di origine:
colorchart/originali/a4_gamma1.tif

Numero colori: 16822

Dimensioni : 320 x 212

Adami e Mosma



Numero regioni: 22
Tempo impiegato: 0.2416 s
Throughput: 280759 pixel/s
Quadrati riconosciuti: 21
Quadrati parzialmente riconosciuti: 0

Pujas e Aldon

Numero regioni: 24
Tempo impiegato: 29.9399 s
Throughput: 2266 pixel/s
Quadrati riconosciuti: 19
Quadrati parzialmente riconosciuti: 2

Comaniciu e Meer

Numero regioni: 17
Tempo impiegato: 2.0831 s
Throughput: 32567 pixel/s
Quadrati riconosciuti: 22
Quadrati parzialmente riconosciuti: 0

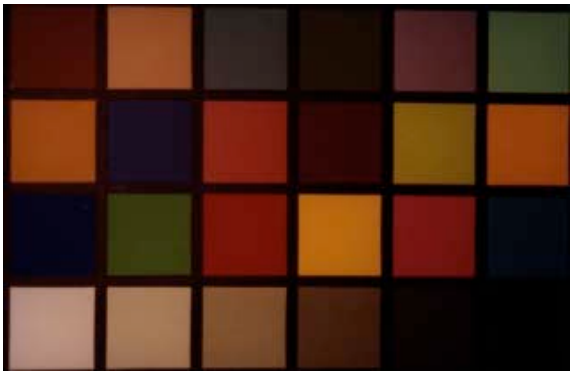
Deng, Manjunath e Shin

Numero regioni: 25
Tempo impiegato: 36.1400 s
Throughput: 1877 pixel/s
Quadrati riconosciuti: 20
Quadrati parzialmente riconosciuti: 2

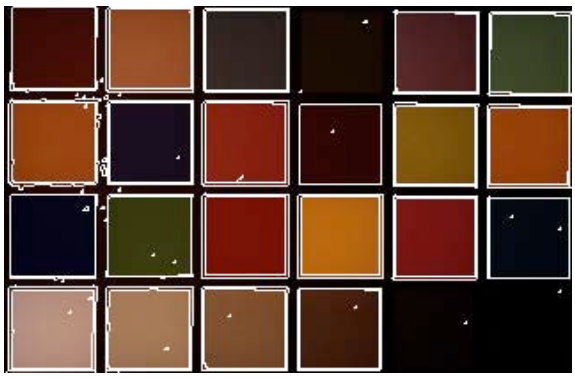
File di origine:
colorchart/originali/a8_gamma1.tif

Numero colori: 11696

Dimensioni : 320 x 211

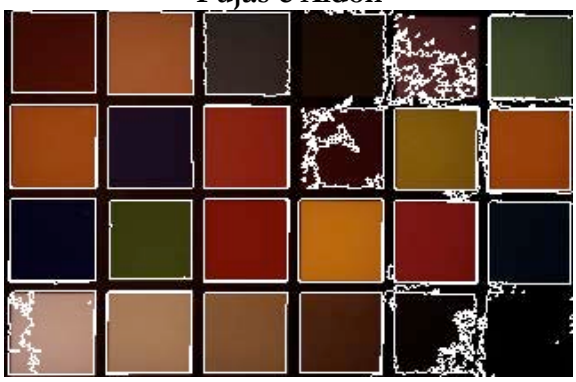


Adami e Mosma



Numero regioni: 22
Tempo impiegato: 0.2390 s
Throughput: 282558 pixel/s
Quadrati riconosciuti: 21
Quadrati parzialmente riconosciuti: 0

Pujas e Aldon



Numero regioni: 24
Tempo impiegato: 24.4846 s
Throughput: 2758 pixel/s
Quadrati riconosciuti: 18
Quadrati parzialmente riconosciuti: 4

Comaniciu e Meer

Numero regioni: 15
Tempo impiegato: 1.6463 s
Throughput: 41012 pixel/s
Quadrati riconosciuti: 21
Quadrati parzialmente riconosciuti: 1

Deng, Manjunath e Shin

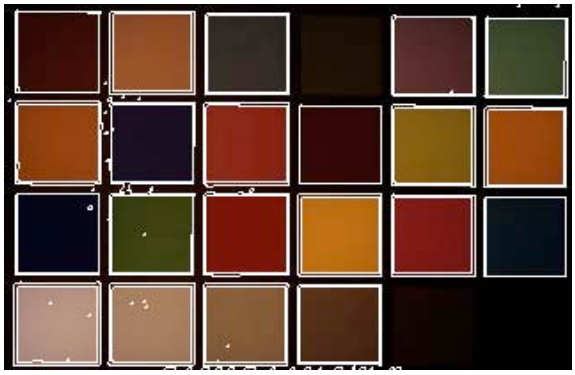
Numero regioni: 19
Tempo impiegato: 34.8200 s
Throughput: 1939 pixel/s
Quadrati riconosciuti: 15
Quadrati parzialmente riconosciuti: 1

File di origine:
colorchart/originali/a81_gamma1.tif

Numero colori: 12031

Dimensioni : 320 x 213

Adami e Mosma



Numero regioni: 22
Tempo impiegato: 0.2486 s
Throughput: 274171 pixel/s
Quadrati riconosciuti: 21
Quadrati parzialmente riconosciuti: 0

Pujas e Aldon

Numero regioni: 21
Tempo impiegato: 30.0466 s
Throughput: 2268 pixel/s
Quadrati riconosciuti: 17
Quadrati parzialmente riconosciuti: 3

Comaniciu e Meer

Numero regioni: 17
Tempo impiegato: 1.8439 s
Throughput: 36965 pixel/s
Quadrati riconosciuti: 22
Quadrati parzialmente riconosciuti: 1

Deng, Manjunath e Shin

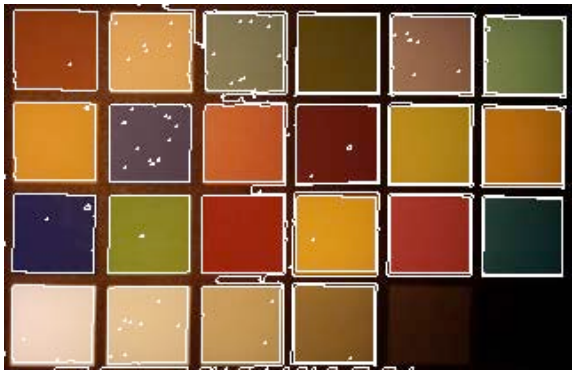
Numero regioni: 18
Tempo impiegato: 36.4700 s
Throughput: 1869 pixel/s
Quadrati riconosciuti: 17
Quadrati parzialmente riconosciuti: 0

File di origine:
colorchart/originali/a84_gamma1.tif

Numero colori: 22754

Dimensioni : 320 x 213

Adami e Mosma



Numero regioni: 23
Tempo impiegato: 0.2457 s
Throughput: 277452 pixel/s
Quadrati riconosciuti: 22
Quadrati parzialmente riconosciuti: 0

Pujas e Aldon

Numero regioni: 21
Tempo impiegato: 36.7768 s
Throughput: 1853 pixel/s
Quadrati riconosciuti: 13
Quadrati parzialmente riconosciuti: 4

Comaniciu e Meer

Numero regioni: 15
Tempo impiegato: 1.9071 s
Throughput: 35741 pixel/s
Quadrati riconosciuti: 21
Quadrati parzialmente riconosciuti: 1

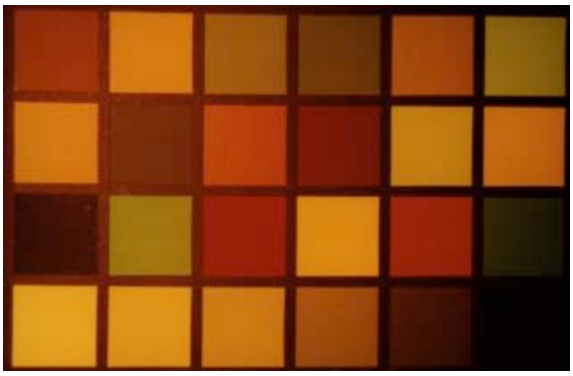
Deng, Manjunath e Shin

Numero regioni: 25
Tempo impiegato: 37.5700 s
Throughput: 1814 pixel/s
Quadrati riconosciuti: 17
Quadrati parzialmente riconosciuti: 4

File di origine:
colorchart/originali/aa1_gamma1.tif

Numero colori: 10512

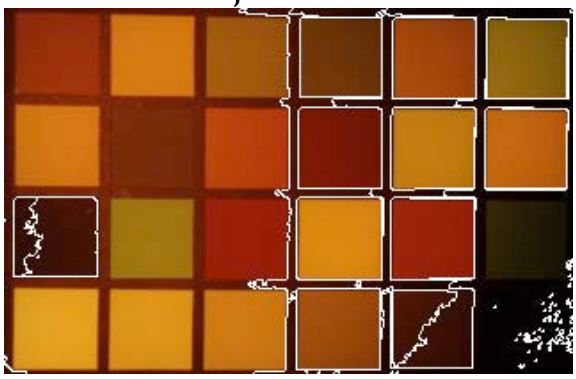
Dimensioni : 320 x 212



Adami e Mosma

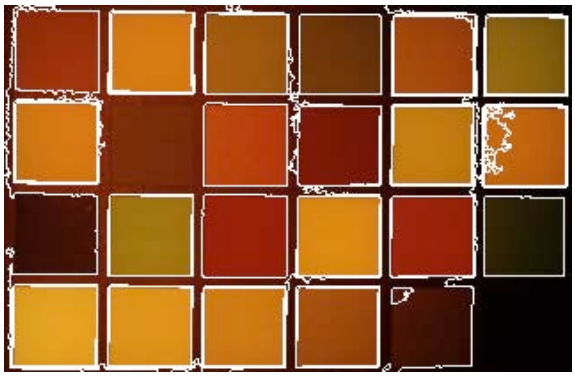
Numero regioni: 24
Tempo impiegato: 0.2515 s
Throughput: 269737 pixel/s
Quadrati riconosciuti: 21
Quadrati parzialmente riconosciuti: 1

Pujas e Aldon



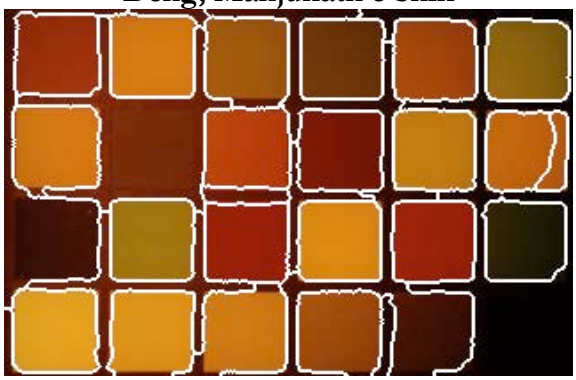
Numero regioni: 14
Tempo impiegato: 22.1732 s
Throughput: 3060 pixel/s
Quadrati riconosciuti: 9
Quadrati parzialmente riconosciuti: 2

Comaniciu e Meer



Numero regioni: 10
Tempo impiegato: 1.2445 s
Throughput: 54513 pixel/s
Quadrati riconosciuti: 19
Quadrati parzialmente riconosciuti: 1

Deng, Manjunath e Shin



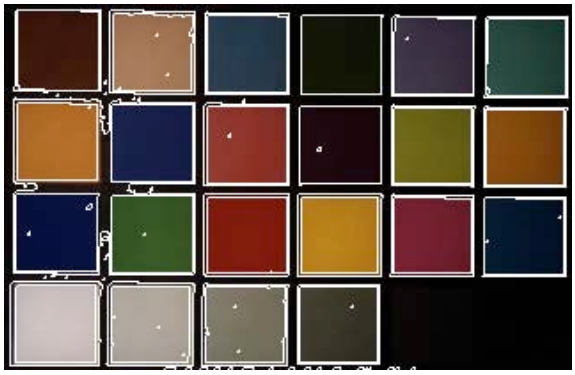
Numero regioni: 30
Tempo impiegato: 36.4200 s
Throughput: 1863 pixel/s
Quadrati riconosciuti: 14
Quadrati parzialmente riconosciuti: 6

File di origine:
colorchart/originali/aa10_gamma1.tif

Numero colori: 16642

Dimensioni : 320 x 213

Adami e Mosma



Numero regioni: 23
Tempo impiegato: 0.2414 s
Throughput: 282310 pixel/s
Quadrati riconosciuti: 22
Quadrati parzialmente riconosciuti: 0

Pujas e Aldon

Numero regioni: 20
Tempo impiegato: 24.6283 s
Throughput: 2768 pixel/s
Quadrati riconosciuti: 17
Quadrati parzialmente riconosciuti: 1

Comaniciu e Meer

Numero regioni: 18
Tempo impiegato: 2.0818 s
Throughput: 32741 pixel/s
Quadrati riconosciuti: 22
Quadrati parzialmente riconosciuti: 0

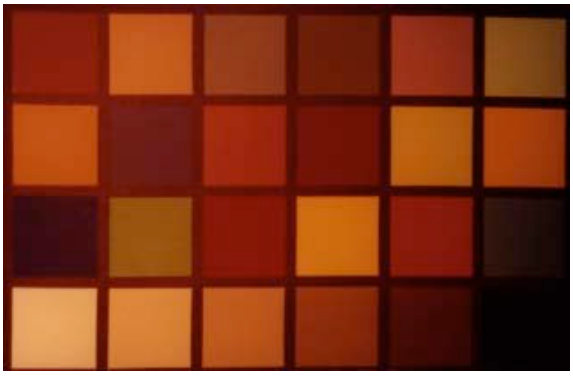
Deng, Manjunath e Shin

Numero regioni: 21
Tempo impiegato: 37.4600 s
Throughput: 1820 pixel/s
Quadrati riconosciuti: 20
Quadrati parzialmente riconosciuti: 0

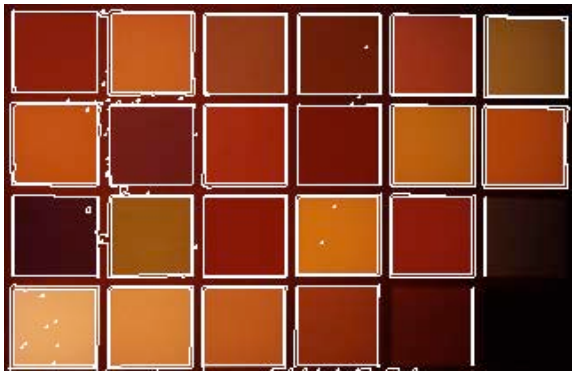
File di origine:
colorchart/originali/aa4_gamma1.tif

Numero colori: 10911

Dimensioni : 320 x 214

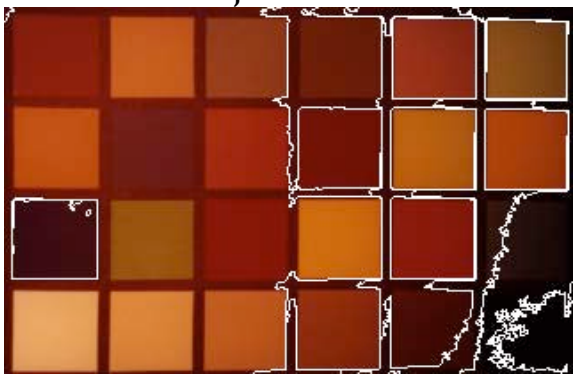


Adami e Mosma



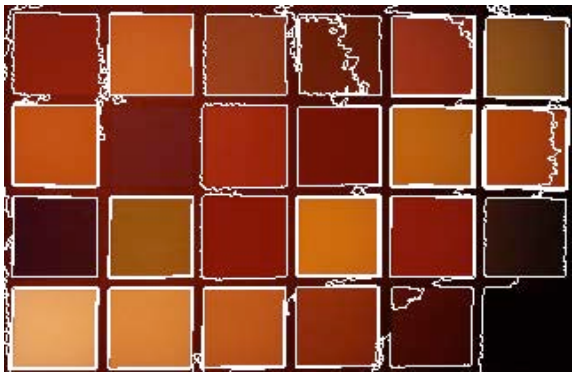
Numero regioni: 22
Tempo impiegato: 0.2418 s
Throughput: 283182 pixel/s
Quadrati riconosciuti: 21
Quadrati parzialmente riconosciuti: 0

Pujas e Aldon



Numero regioni: 12
Tempo impiegato: 19.8192 s
Throughput: 3455 pixel/s
Quadrati riconosciuti: 6
Quadrati parzialmente riconosciuti: 3

Comaniciu e Meer



Numero regioni: 13
Tempo impiegato: 1.4338 s
Throughput: 47761 pixel/s
Quadrati riconosciuti: 17
Quadrati parzialmente riconosciuti: 4

Deng, Manjunath e Shin

Numero regioni: 28
Tempo impiegato: 39.9300 s
Throughput: 1715 pixel/s
Quadrati riconosciuti: 13
Quadrati parzialmente riconosciuti: 6

4.5. Test su immagini generiche

Sono riportati di seguito i risultati della segmentazione applicati a cinque immagini generiche e i parametri scelti per gli algoritmi supervised.

Le prime quattro sono immagini di test usate frequentemente in letteratura, mentre la quinta è uno screenshot televisivo.

Prima immagine: woman.tif

- Algoritmo di Adami e Mosma

Soglia:	900
Numero minimo pixel:	400
Numero massimo pixel:	300000

- Algoritmo di Pujas e Aldon

Soglia su db:	0.95
Soglia su H tra pixel:	0.9
Soglia su V tra pixel:	255
Soglia su Hrp tra regione e pixel:	4.9
Soglia su Vrp tra regione e pixel:	255
Compressione del cp:	0.25
Numero minimo di pixel di una regione di pixel da accorpare:	400
Soglia su Hrr tra regioni:	2.9
Soglia su Vrr tra regioni:	50
Soglia massima su l_ir tra regioni:	0.6

Seconda immagine: smhouse.tif

- Algoritmo di Adami e Mosma

Soglia:	400
Numero minimo pixel:	100
Numero massimo pixel:	300000

- Algoritmo di Pujas e Aldon

Soglia su db:	0.9
Soglia su H tra pixel:	2.9
Soglia su V tra pixel:	255
Soglia su Hrp tra regione e pixel:	2.9
Soglia su Vrp tra regione e pixel:	255
Compressione del cp:	0.6
Numero minimo di pixel di una regione di pixel da accorpare:	400
Soglia su Hrr tra regioni:	2.9
Soglia su Vrr tra regioni:	255
Soglia massima su l_ir tra regioni:	0.6

Terza immagine: path.tif

- Algoritmo di Adami e Mosma

Soglia:	200
Numero minimo pixel:	200
Numero massimo pixel:	300000

- Algoritmo di Pujas e Aldon

Soglia su db:	0.95
Soglia su H tra pixel:	0.9
Soglia su V tra pixel:	255
Soglia su Hrp tra regione e pixel:	4.9
Soglia su Vrp tra regione e pixel:	255
Compressione del cp:	0.25
Numero minimo di pixel di una regione di pixel da accorpare:	600
Soglia su Hrr tra regioni:	2.9
Soglia su Vrr tra regioni:	50
Soglia massima su l_ir tra regioni:	0.6

Quarta immagine: hand.tif

- Algoritmo di Adami e Mosma

Soglia:	400
Numero minimo pixel:	600
Numero massimo pixel:	300000

- Algoritmo di Pujas e Aldon

Soglia su db:	0.75
Soglia su H tra pixel:	0.9
Soglia su V tra pixel:	255
Soglia su Hrp tra regione e pixel:	4.9
Soglia su Vrp tra regione e pixel:	255
Compressione del cp:	0.25
Numero minimo di pixel di una regione di pixel da accorpare:	600
Soglia su Hrr tra regioni:	2.9
Soglia su Vrr tra regioni:	50
Soglia massima su l_ir tra regioni:	0.6

Quinta immagine: tv.tif

- Algoritmo di Adami e Mosma

Soglia:	2700
Numero minimo pixel:	200
Numero massimo pixel:	300000

- Algoritmo di Pujas e Aldon

Soglia su db:	0.77
Soglia su H tra pixel:	5.5
Soglia su V tra pixel:	155
Soglia su Hrp tra regione e pixel:	5.9
Soglia su Vrp tra regione e pixel:	155
Compressione del cp:	0.25
Numero minimo di pixel di una regione di pixel da accorpare:	500
Soglia su Hrr tra regioni:	5.5
Soglia su Vrr tra regioni:	50
Soglia massima su l_ir tra regioni:	0.1

File di origine:
generiche/originali/woman.tif

Numero colori: 200

Dimensioni : 116 x 261

Adami e Mosma
Numero regioni: 6
Tempo impiegato: 0.1076 s
Throughput: 281457 pixel/s

Pujas e Aldon
Numero regioni: 18
Tempo impiegato: 15.3314 s
Throughput: 1975 pixel/s

Comaniciu e Meer
Numero regioni: 4
Tempo impiegato: 0.2872 s
Throughput: 105430 pixel/s

Deng, Manjunath e Shin
Numero regioni: 10
Tempo impiegato: 12.9600 s
Throughput: 2336 pixel/s

File di origine:
generiche/originali/smhouse.tif

Numero colori: 9603

Dimensioni : 255 x 192

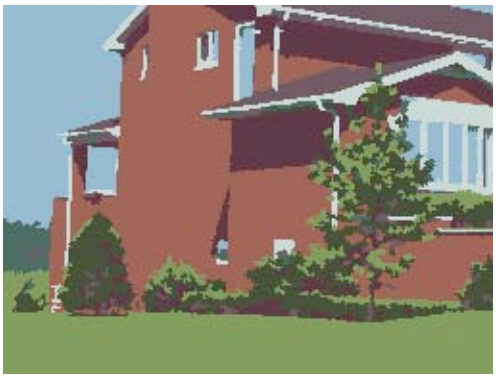
Adami e Mosma

Numero regioni: 26
Tempo impiegato: 0.1861 s
Throughput: 263106 pixel/s

Pujas e Aldon

Numero regioni: 16
Tempo impiegato: 28.0254 s
Throughput: 1747 pixel/s

Comaniciu e Meer



Numero regioni: 8
Tempo impiegato: 1.0976 s
Throughput: 44605 pixel/s

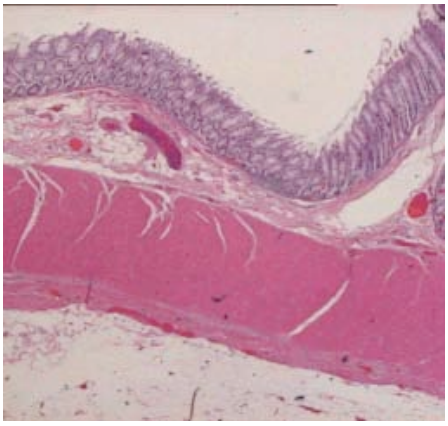
Deng, Manjunath e Shin

Numero regioni: 17
Tempo impiegato: 21.6900 s
Throughput: 2257 pixel/s

File di origine:
generiche/originali/path.tif

Numero colori: 187

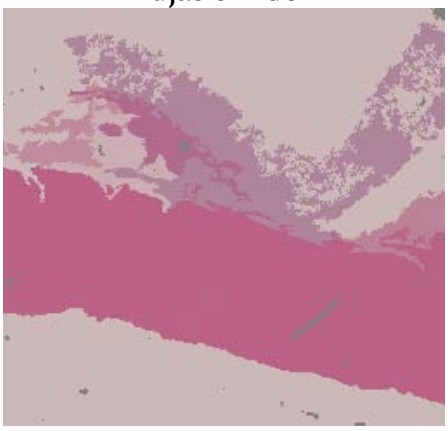
Dimensioni : 255 x 241



Adami e Mosma

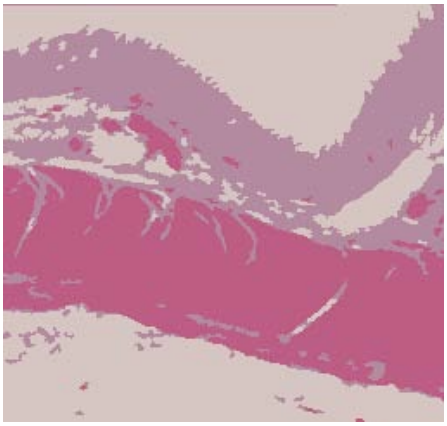
Numero regioni: 14
Tempo impiegato: 0.2420 s
Throughput: 253995 pixel/s

Pujas e Aldon



Numero regioni: 8
Tempo impiegato: 17.2749 s
Throughput: 3563 pixel/s

Comaniciu e Meer



Numero regioni: 3
Tempo impiegato: 0.5441 s
Throughput: 112945 pixel/s

Deng, Manjunath e Shin

Numero regioni: 11
Tempo impiegato: 32.5700 s
Throughput: 1887 pixel/s

File di origine:
generiche/originali/hand.tif

Numero colori: 200

Dimensioni : 303 x 243

Adami e Mosma

Numero regioni: 6
Tempo impiegato: 0.2975 s
Throughput: 247505 pixel/s

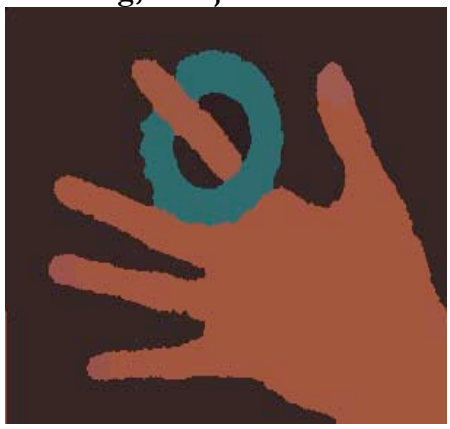
Pujas e Aldon

Numero regioni: 10
Tempo impiegato: 41.2713 s
Throughput: 1784 pixel/s

Comaniciu e Meer

Numero regioni: 3
Tempo impiegato: 0.8028 s
Throughput: 91712 pixel/s

Deng, Manjunath e Shin



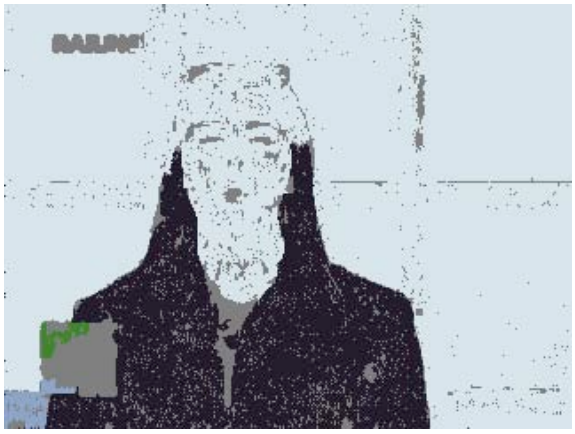
Numero regioni: 9
Tempo impiegato: 46.7900 s
Throughput: 1574 pixel/s

File di origine:
generiche/originali/tv.tif

Numero colori: 137

Dimensioni : 320 x 240

Adami e Mosma



Numero regioni: 4
Tempo impiegato: 0.2867 s
Throughput: 267877 pixel/s

Pujas e Aldon

Numero regioni: 13
Tempo impiegato: 13.4486 s
Throughput: 5711 pixel/s

Comaniciu e Meer

Numero regioni: 4
Tempo impiegato: 0.8185 s
Throughput: 93827 pixel/s

Deng, Manjunath e Shin

Numero regioni: 18
Tempo impiegato: 37.9000 s
Throughput: 2026 pixel/s

5. Analisi dei risultati

In questo capitolo confronteremo i risultati degli algoritmi di segmentazione presentati. Per ogni categoria di immagine riassumeremo i dati relativi all'elaborazione di ogni algoritmo, analizzeremo tali dati e daremo una valutazione qualitativa della segmentazione.

Inoltre, per i colorchart, i risultati verranno valutati numericamente, poiché in tal caso è possibile stabilire quale sia la segmentazione ottima: se l'immagine contiene 24 quadrati di colore diverso più uno sfondo, la segmentazione ottima deve produrre 25 regioni. Quindi, per i colorchart, misureremo anche il numero medio di quadrati riconosciuti.

5.1. Immagini di navigazione in diverse condizioni di illuminazione

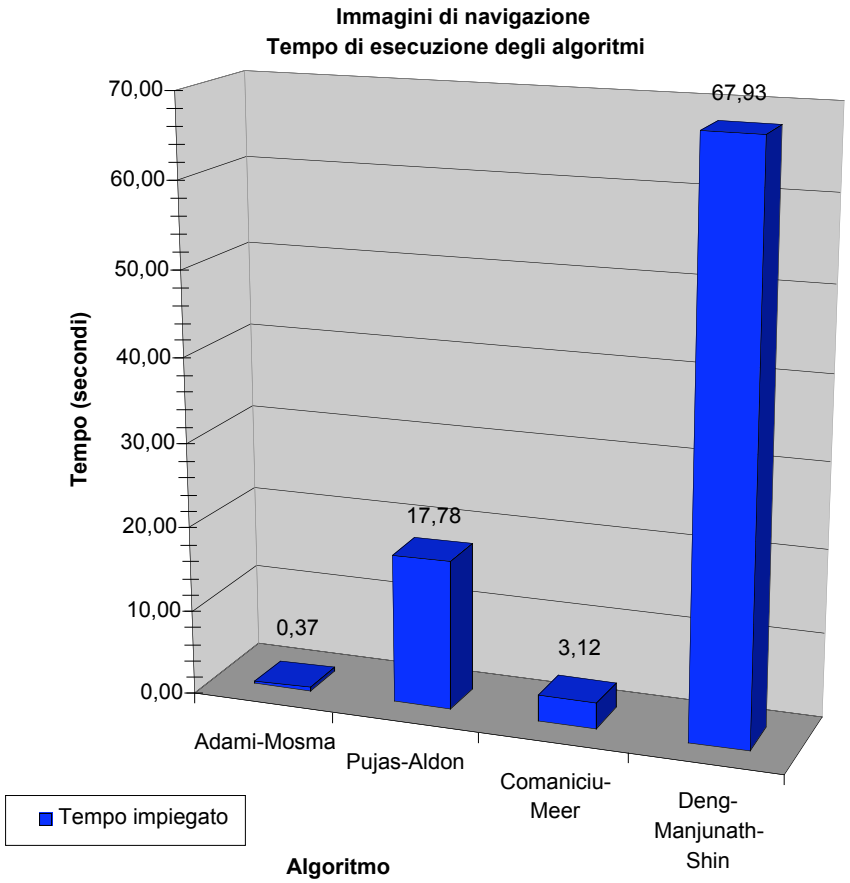
Nella tabella qui sotto e nel grafico alla pagina successiva sono riportati i tempi di esecuzione per la segmentazione delle cinque immagini con i diversi algoritmi.

	Adami-Mosma	Pujas-Aldon	Comaniciu Meer	Deng-Manjunath-Shin
Ing-day.tif	0,38	17,81	3,09	76,12
Ing-flash.tif	0,37	18,73	3,44	64,32
Ing-fluo.tif	0,37	17,94	3,05	66,35
Ing-off.tif	0,36	17,38	2,86	76,40
Ing-tung.tif	0,37	17,03	3,17	56,46
Media	**0,37**	**17,78**	**3,12**	**67,93**

Tabella 1. *Tempo (secondi) di esecuzione degli algoritmi applicati alle immagini elencate.*

I quattro algoritmi producono risultati qualitativi non troppo diversi tra di loro. Si può dire che la segmentazione è genericamente discreta. Per quanto riguarda invece i tempi, variano di due ordini di grandezza, e vista la qualità dei risultati, queste differenze non sono giustificate. In particolare, per gli algoritmi supervised, quello di Adami-Mosma è notevolmente più veloce, e addirittura quello di Pujas-Aldon, che

richiede un notevole sforzo per la taratura dei parametri, risulta più lento di quello di Comaniciu-Meer, che è unsupervised. I tempi di Deng-Manjunath-Shin sono invece talmente alti che è improponibile l'uso per una navigazione in tempo reale. Inoltre, nonostante le cinque immagini siano di dimensioni uguali, la varianza tra i tempi dell'ultimo algoritmo è piuttosto grande.



Per quanto riguarda il numero delle regioni indicato nel capitolo 5, il quarto algoritmo sembra produrne un numero molto più grande. È comunque da ricordare che gli algoritmi di Pujas-Aldon e Comaniciu-Meer hanno un altro modo di contare le regioni (si veda il paragrafo 5.2), che li porta ad indicare un numero più basso. Quindi, da un'analisi più approfondita, è proprio l'algoritmo di Comaniciu-Meer a produrre una sovrasegmentazione maggiore, anche se l'algoritmo è automaticamente impostato su dei parametri che, secondo gli autori, produrrebbero una sottosegmentazione. Tale sovrasegmentazione consente da un lato di riconoscere particolari quali il telefono e la maniglia, ma dall'altro non permette una corretta individuazione della porta. L'unico algoritmo che riconosce correttamente la porta in tutte le immagini è quello di Adami-Mosma. Esso in effetti tende a sottosegmentare. Invece, gli algoritmi di Deng-Manjunath-Shin e di Pujas-Aldon, possono essere visti

come vie di mezzo, anche se quest'ultimo in qualche caso tende a sottosegmentare più di Adami-Mosma, ma in molti altri non riconosce correttamente la porta.

5.2. Colorchart

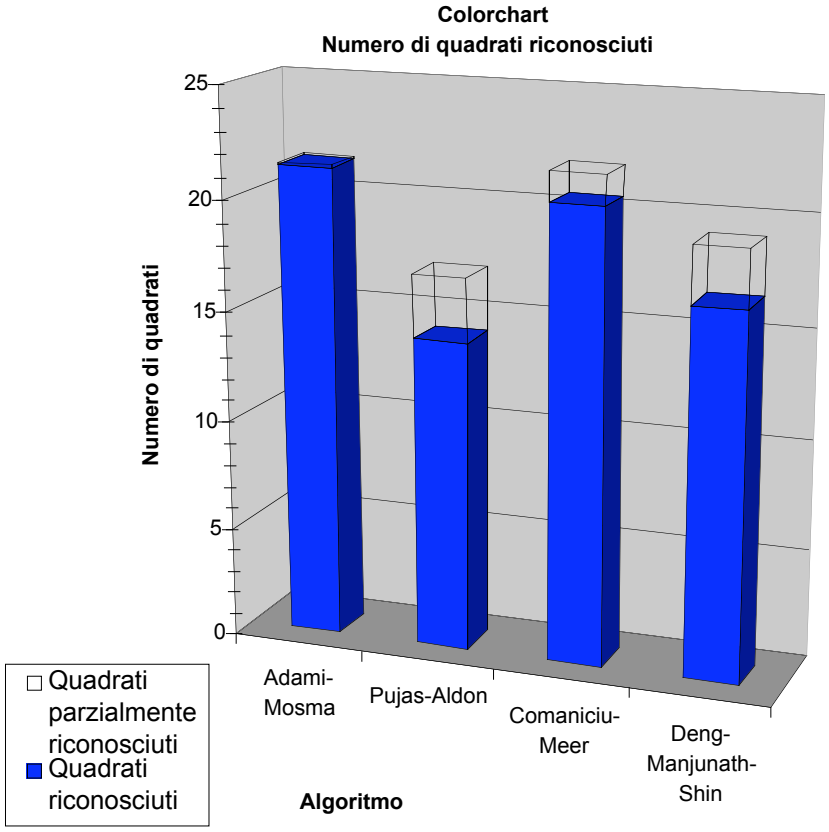
Presentiamo di seguito i risultati forniti dagli algoritmi applicati ai colorchart. Oltre che sui tempi impiegati, è possibile fare un confronto di numero di quadrati: in questo modo si può misurare quantitativamente l'efficacia della segmentazione.

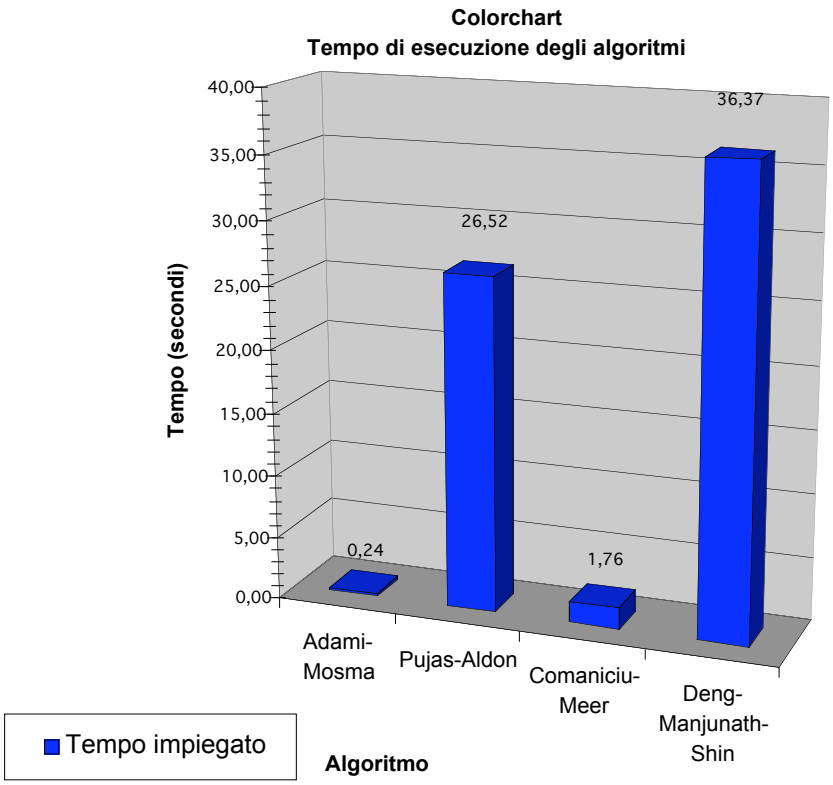
	Adami-Mosma	Pujas-Aldon	Comaniciu-Meer	Deng-Manjunath-Shin
a_gamma1.tif	0,25	26,47	1,81	36,91
a10_gamma1.tif	0,24	21,04	1,91	33,72
a3_gamma1.tif	0,24	29,85	1,62	34,27
a4_gamma1.tif	0,24	29,94	2,08	36,14
a8_gamma1.tif	0,24	24,48	1,65	34,82
a81_gamma1.tif	0,25	30,05	1,84	36,47
a84_gamma1.tif	0,25	36,78	1,91	37,57
aa1_gamma1.tif	0,25	22,17	1,24	36,42
aa10_gamma1.tif	0,24	24,63	2,08	37,46
aa4_gamma1.tif	0,24	19,82	1,43	39,93
Media	**0,24**	**26,52**	**1,76**	**36,37**

Tabella 2. Tempo (secondi) di esecuzione degli algoritmi applicati alle immagini elencate

	Adami-Mosma	Pujas-Aldon	Comaniciu-Meer	Deng-Manjunath-Shin
a_gamma1.tif	21+0	16+6	22+0	18+2
a10_gamma1.tif	23+0	17+1	23+0	19+2
a3_gamma1.tif	21+0	9+3	16+6	13+3
a4_gamma1.tif	21+0	19+2	22+0	20+2
a8_gamma1.tif	21+0	18+4	21+1	15+1
a81_gamma1.tif	21+0	17+3	22+1	17+0
a84_gamma1.tif	22+0	13+4	21+1	17+4
Aa1_gamma1.tif	21+1	9+2	19+1	14+6
Aa10_gamma1.tif	22+0	17+1	22+0	20+0
Aa4_gamma1.tif	21+0	6+3	17+4	13+6
Media	**21,4+0,1**	**14,1+2,9**	**20,5+1,4**	**16,6+2,6**

Tabella 3. Numero di quadrati riconosciuti dagli algoritmi per ciascuna delle immagini elencate. 'a+b' *indica che sono stati riconosciuti completamente* a *quadrati, mentre altri* b *solo parzialmente.*


Colorchart
Numero di quadrati riconosciuti
Numero di quadrati
25
20
15
10
5
0
Adami-Mosma
Pujas-Aldon
Comaniciu-Meer
Deng-Manjunath-Shin
Algoritmo
Quadrati parzialmente riconosciuti
Quadrati riconosciuti


Colorchart
Tempo di esecuzione degli algoritmi
Tempo (secondi)
40,00
35,00
30,00
25,00
20,00
15,00
10,00
5,00
0,00
0,24
26,52
1,76
36,37
Adami-Mosma
Pujas-Aldon
Comaniciu-Meer
Deng-Manjunath-Shin
Algoritmo
Tempo impiegato

Dall'analisi dei risultati l'algoritmo di Adami-Mosma risulta essere decisamente il migliore. In tutte le immagini la segmentazione sfiora l'ottimo, anche se, pur essendo supervised, l'algoritmo è stato eseguito con lo stesso set di parametri per tutte le immagini. Peraltro, per questa categoria di immagini, la taratura è piuttosto semplice perché praticamente è limitata alla scelta della soglia di somiglianza tra pixel. D'altra parte, una tecnica region-growing dovrebbe essere la migliore per immagini di questo tipo, in cui colori molto vicini sono pur sempre divisi da regioni di colore nero.

Risultati di poco inferiori si ottengono dall'algoritmo di Comaniciu-Meer; tale algoritmo, pur essendo unsupervised, si comporta molto bene con queste immagini, particolarmente critiche. Questo si deve sia alla scelta della soglia dipendente dalla matrice di covarianza dell'immagine, sia alla notevole considerazione dei vincoli spaziali. Il tempo di esecuzione richiesto è però molto maggiore (circa 7 volte).

L'algoritmo di Pujas-Aldon, anche dopo una laboriosa taratura dei parametri, fatta comunque una volta per tutte le immagini, presenta risultati mediocri. Inoltre, si può osservare che la varianza del numero di quadrati riconosciuti è piuttosto alta; l'algoritmo sembra mal sopportare le immagini che hanno un maggiore contenuto di rosso. Un discorso analogo vale per l'algoritmo di Deng-Manjunath-Shin, sebbene le differenze siano meno evidenti ed esso non abbia bisogno di tarature.

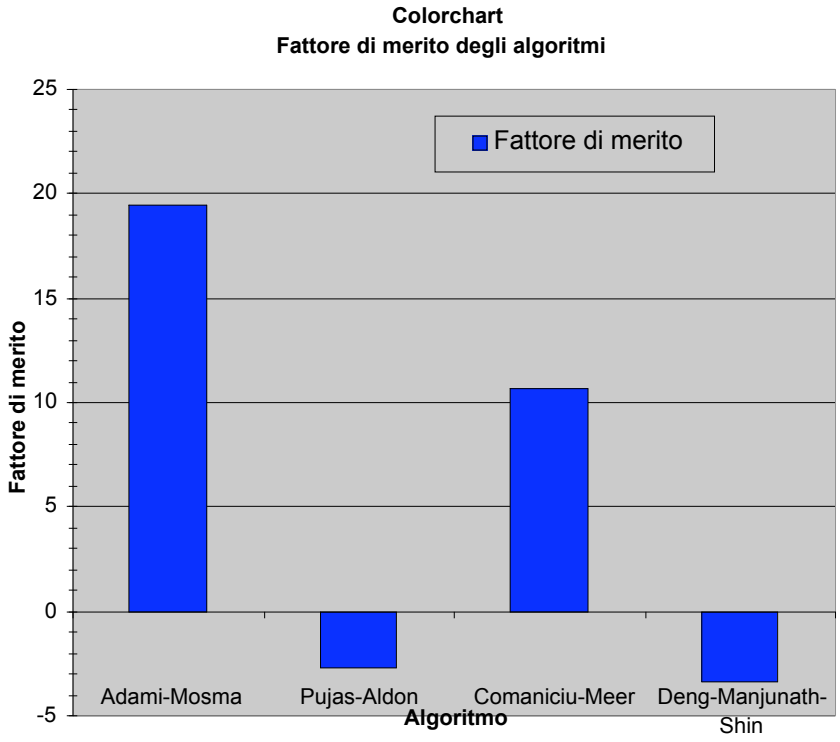
L'analisi dei tempi di esecuzione conferma la bontà dell'algoritmo di Adami-Mosma; Pujas-Aldon e Deng-Manjunath-Shin sono sconsigliabili perché in questo caso sono lenti e poco efficaci.

Per dare una valutazione il più possibile oggettiva, introduciamo un fattore di merito F così definito:

$$F = 10 \cdot \log\left(\frac{\text{numero di quadrati riconosciuti}}{\text{tempo impiegato}} \right)$$

Questo fattore di merito è quindi alto quando l'algoritmo fornisce buoni risultati in tempi veloci. Poiché vengono combinati risultati e tempi, esso dà un'indicazione del migliore compromesso tra efficacia e efficienza.

Alla pagina successiva è riportato il grafico dei fattori di merito degli algoritmi.

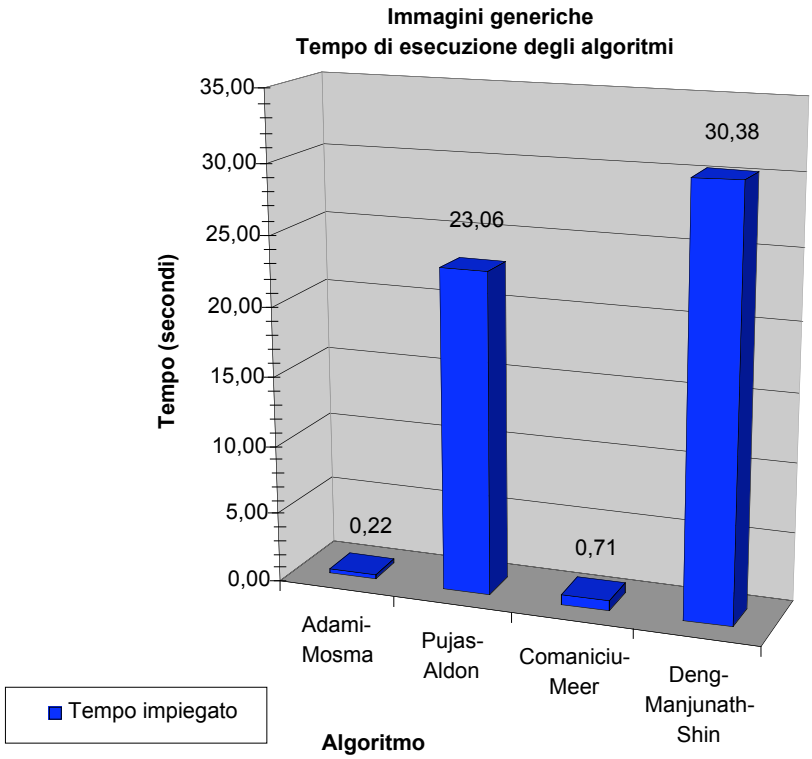


5.3. Immagini generiche

Nella tabella seguente sono riportati i tempi di esecuzione degli algoritmi applicati ad immagini generiche. Viene inoltre presentato un grafico che riassume tali dati. Ricordiamo che, in questo caso, per ogni immagine e per ogni algoritmo supervised è stata fatta una diversa impostazione dei parametri.

	Adami-Mosma	Pujas-Aldon	Comaniciu-Meer	Deng-Manjunath-Shin
hand.tif	0,30	41,27	0,80	46,79
path.tif	0,24	17,25	0,54	32,57
smhouse.tif	0,19	28,03	1,10	21,69
tv.tif	0,29	13,45	0,82	37,90
woman.tif	0,11	15,33	0,29	12,96
Media	**0,22**	**23,06**	**0,71**	**30,38**

Tabella 4. *Tempo (secondi) di esecuzione degli algoritmi applicati alle immagini elencate*



Per quanto riguarda l'efficacia della segmentazione, si può dire in generale che per questa categoria di immagini l'algoritmo di Deng-Manjunath-Shin viene rivalutato. Invece, l'algoritmo di Adami-Mosma offre dei risultati meno soddisfacenti di quanto si era riscontrato per i colorchart.

In particolare, per la casa (smhouse.tif) l'algoritmo di Deng-Manjunath-Shin produce i risultati migliori grazie allo sfruttamento della tecnica delle texture; ad esempio, la finestra a destra, nella quale si susseguono aree azzurre ed aree bianche, viene riconosciuta come un unico oggetto; un discorso analogo vale per le piante. L'algoritmo di Comaniciu-Meer, pur rilevando un numero limitato di regioni di colore diverso, tende a sovrasegmentare. L'algoritmo di Pujas-Aldon funziona leggermente meglio, senza tuttavia giustificare un tempo di esecuzione così alto. L'algoritmo di Adami-Mosma invece non riesce a riconoscere le piante, in quanto costituite da piccole regioni di colori troppo diversi.

Per l'immagine path.tif valgono le considerazioni fatte per la casa. La segmentazione migliore è quella di Deng-Manjunath-Shin, che presenta regioni più compatte di quelle di Comaniciu-Meer, che tuttavia dà in output solo 3 regioni di colore diverso. L'algoritmo di Adami-Mosma scarta troppe regioni, mentre quello di Pujas-Aldon ha tempi troppo alti.

L'immagine della mano (hand.tif) presenta delle texture sullo sfondo, costituite però di colori non troppo diversi tra loro, quindi l'algoritmo migliore risulta quello di Comaniciu-Meer. Invece, gli algoritmi basati su region growing non riescono a riconoscere bene lo sfondo. Ottimo risultato fornisce l'algoritmo di Deng-Manjunath-Shin, ma in questo caso esso ha lo svantaggio di avere tempi di esecuzione più alti.

L'immagine della donna (woman.tif) non richiede particolari abilità nel riconoscimento di texture, quindi l'algoritmo di Comaniciu-Meer rappresenta il miglior compromesso fra il risultato (praticamente perfetto) e i tempi. Qualche difficoltà presenta l'algoritmo di Adami-Mosma, e risulta molto complicata l'impostazione dei parametri per l'algoritmo di Pujas-Aldon.

L'immagine tv.tif ha richiesto un notevole sforzo di taratura dei parametri per entrambi gli algoritmi supervised, probabilmente a causa del rumore elevato. Gli algoritmi migliori sono i due unsupervised, e quello di Comaniciu-Meer si lascia preferire dopo l'analisi dei tempi.

I tempi più alti sono stati impiegati dall'algoritmo di Deng-Manjunath-Shin, ma essi sono quasi sempre giustificati da un migliore riconoscimento delle texture. Anche l'algoritmo di Pujas-Aldon presenta tempi abbastanza alti, ma non fornisce risultati migliori di quello di Comaniciu-Meer, che funziona bene e ha tempi dell'ordine del secondo. L'algoritmo di Adami-Mosma richiede i tempi più bassi, ma presenta i risultati peggiori.

6. Conclusioni

L'obbiettivo di questo progetto è stato confrontare quattro algoritmi di segmentazione. Due di essi sono supervised e basati sulla tecnica region growing, gli altri due sono unsupervised e basati sul clustering o sull'estrazione di contorni. Uno di essi prevede inoltre il riconoscimento delle texture. Sono state confrontate l'efficacia, l'efficienza e l'omogeneità dei risultati dei quattro algoritmi. A tale scopo, si sono resi necessari interventi sui codici in modo di rendere le interfacce e gli output omogenei e di effettuare la misura dei tempi.

L'algoritmo di Adami-Mosma, basato sul region growing, si è rivelato il più veloce in assoluto e presenta buoni risultati se applicato a immagini non troppo complesse, in cui le aree sono abbastanza grandi, hanno colore sufficientemente omogeneo e c'è una netta divisione tra regioni adiacenti. Esso necessita di una taratura di tre parametri che risulta non troppo complicata. Per gli scopi con i quali è nato, scarta le regioni troppo piccole, mentre la scelta più comune è quella di accorparle alle regioni adiacenti. Gli output presentano dei piccoli disturbi, sotto forma di pixel 'liberi' interni ad una regione, che potrebbero essere eliminati ad esempio tramite un *majority game* (paragrafo 3.1). Lo spazio dei colori scelto è il RGB, ma comunque l'algoritmo fornisce ottimi risultati anche con illuminanti diversi.

L'algoritmo di Pujas-Aldon è stato creato con lo scopo di essere robusto alla variazione delle condizioni di illuminazione della scena. A tal fine, opera nello spazio HSV. Esso è basato sulla tecnica del region growing e necessita dell'impostazione di ben dieci parametri diversi. I risultati ed i tempi di esecuzione sono notevolmente influenzati dai parametri, ma per avere risultati accettabili i tempi diventano piuttosto alti – anche di due ordini di grandezza rispetto all'algoritmo di Adami-Mosma – ed è difficile capire in quale direzione muoversi nello spazio dei parametri per migliorare la qualità dell'output. Non sempre si ottengono buoni risultati e per di più anche tra le immagini caratterizzate da diversi illuminanti questo algoritmo non si è rivelato il migliore.

L'algoritmo di Comaniciu-Meer rappresenta in generale il miglior compromesso tra tempi di esecuzione e qualità del risultato. Dà risultati ugualmente buoni su qualsiasi tipo di immagine da noi testato. Non necessita della taratura di alcun parametro ed è basato sulla tecnica del clustering, ma dà una certa rilevanza anche all'informazione di vicinanza tra i pixel. In questo modo si uniscono i vantaggi delle

due tecniche. I risultati sono un po' sovrasegmentati, anche se la versione dell'algoritmo da noi scelta è dichiarata sottosegmentante dall'autore.

L'algoritmo di Deng-Manjunath-Shin dà i risultati migliori quando le immagini sono complicate e presentano delle aree textured. Infatti è prevista un'apposita procedura per il riconoscimento delle texture, al termine della quale, l'algoritmo prevede l'estrazione dei contorni. I tempi richiesti sono però decisamente lunghi – maggiori di quelli di Pujas-Aldon – ed aumentano notevolmente con l'aumentare delle aree textured; in assenza di tali aree, i tempi impiegati non sono comunque giustificati, perché l'efficacia non è molto diversa da quella dell'algoritmo di Comaniciu-Meer.

Concludendo, la scelta tra questi algoritmi dipende dal tipo di immagine considerata e dai vincoli imposti dal task che richiede la segmentazione. Si può dire che dovendo operare su immagini non troppo complesse e avendo dei vincoli temporali molto stretti la scelta ricade sull'algoritmo di Adami-Mosma. In realtà, se il task prevede attività in tempo reale, esso è l'unica via percorribile. Se i vincoli temporali sono meno restrittivi, o avendo a disposizione una macchina più potente, l'algoritmo di Comaniciu-Meer è il più adatto, e ha il vantaggio di lavorare bene su immagini di qualsiasi tipo. Se le immagini da segmentare sono molto complesse e con texture complicate, è necessario non avere vincoli temporali ed affidarsi all'algoritmo di Deng-Manjunath-Shin.

Bibliografia

[Adami,1998] N. Adami, P. Mosma, "Metodologie di autolocalizzazione per robot autonomi basate sulle caratteristiche di visione cromatica degli insetti", Tesi di laurea in Ingegneria Elettronica, Università degli Studi di Brescia, 1997-98.

[Pujas,1995] Pujas Ph., Aldon M.J., "Robust Color Image Segmentation", Laboratoire d'Informatique, de Robotique et de Microéletronique de Montpellier, ICAR '95.

[Lazzaroni,1999] A. Lazzaroni, G. Roncati, "La segmentazione robusta di immagini colorate", Elaborato di Robotica, Università degli Studi di Brescia, 1999-2000.

[Comaniciu,1997] D. Comaniciu, P. Meer, "Robust analysis of feature spaces: color image segmentation", Proc. IEEE Conf. on Computer Vision and Pattern Recognition, giugno 1997.

[Deng, 1999] Y. Deng, B. S. Manjunath, H. Shin, "Color image segmentation", Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 1999.

[Imlib] "The Imlib programmers guide", disponibile on-line all'indirizzo http://cvs.labs.redhat.com/lxr/source/imlib

[Von Neumann,1944] Von Neumann, J. Morgenstern, "Theory of games and economic behavior", Princeton University Press, 1944.

Anil K. Jain, "Fundamentals of digital image processing" (cap. 9), Prentice Hall International Editions, 1989.

Siti internet consultati:

http://www.caip.rutgers.edu/~comanici/segm_images.html

http://maya.ece.ucsb.edu/JSEG/

Altri siti Internet interessanti:

http://telin.rug.ac.be/~pds/source_code/wshed.html

http://www.univ-reims.fr/Labos/SciencesExa/Leri/membre/luc/SEGMENTE/segmente.html

Pubblicazioni correlate:

Lim, Lee, Seoul, "On the color image segmentation algorithm based on the thresholding and fuzzy c-means techniques", Pattern Recognition, vol. 23, 1993.

Haddon, Boyce, "Image segmentation by unifying region and boundary information", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 12, 1990.

Healey, "Segmenting images using normalized color", IEEE Trans. on Syst., Man., Cybern., vol. 22, 1992.

S. C. Zhu, A. Yuille, "Region competition : unifying snakes, region growing, and Bayes/MDL for multiband image segmentation", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 18, settembre 1996.

Moghaddamzadeh, Bourbakis, "A fuzzy region growing approach for segmentation of color images", Pattern Recognition, vol. 30, 1997.

Tremlau, Borel, "Region growing and merging algorithm to color segmentation", Pattern Recognition, vol. 30, 1997.

Guo Dong Guo, Shan Yu, Song De Ma, "Unsupervised Segmentation of Color Images", International Conference on Image Processing, ICIP'98, Chicago, Illinois, 299-302, 1998

F. Ziliani and B. Jensen "Unsupervised Image Segmentation Using the Modified Pyramidal Linking Approach". In Proceedings of the 5th IEEE International Conference on Image Processing (ICIP'98), vol. 3, pp. 303-307, Chicago, USA, October 4-7, 1998.

A. Banerjee, P. Burlina, F. Alajaji, "Image segmentation and labeling using the Polya Urn Model", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 8, settembre 1999.