UNIVERSITÀ DI BRESCIA
FACOLTÀ DI INGEGNERIA
Dipartimento di Elettronica per l'Automazione

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica
(Prof. Riccardo Cassinis)

Costruzione della libreria Observer

Elaborato di esame di: **Paolo Libardi, Daniel Giardina**

Consegnato il: **23 settembre 2003**

Sommario

Lo scopo del programma è di riconoscere i blob di un determinato colore presenti su un'immagine bitmap. Selezionando il colore dei blob desiderati, una tolleranza di toni e le dimensione minime dei blob, vengono evidenziati i blob e calcolato il loro baricentro. Successivamente è possibile "aggiustare" i blob, accrescendo le aree di colore mediante la funzione "region growing".

1. Introduzione

Questo programma è stato sviluppato con l'intenzione di rendere semplice il riconoscimento di un oggetto da parte di un robot mediante l'utilizzo di una telecamera.

É stata sviluppata un'interfaccia software che fornisce le coordinate di un oggetto rispetto a un' immagine, data una selezione da parte dell'utente del colore dell'oggetto che deve essere raggiunto dal robot. Si presuppone quindi che il robot non sia completamente autonomo in quanto la selezione deve avvenire da parte dell'utente umano.

La parte principale dell'elaborato è composta dall'implementazione delle funzioni di filtraggio, infatti queste devono essere il più veloci possibile, così da essere una base di partenza per un'eventuale implementazione in *real-time*, inoltre, devono essere robuste per elaborare immagini affette da problemi di luminosità che rendono difficile il riconoscimento dei colori reali.

Il programma, successivamente, punta ad un'interfaccia il più possibile semplice e completa. Permette infatti di mantenere visualizzata l'immagine di partenza per tutto il processo dell'elaborazione, di selezionare tutti i possibili parametri di filtraggio, di ritornare allo stato precedente dell'elaborazione.

2. Il problema affrontato

Il compito assegnato è trovare i blob di colore in un'immagine. Un blob è definito come un'area di colore omogeneo, a meno di variazioni dovute all'illuminazione, che rappresenta un oggetto all'interno di uno spazio fotografato. Il riconoscimento della posizione dell'area all'interno di un'immagine può fornire le indicazioni reali di posizione spaziale dell'oggetto desiderato.

Un blob è caratterizzato da:

- colore omogeneo
- area estesa

le problematiche legate all'identificazione dei blob sono molteplici. Infatti, prendendo in considerazione immagini in ingresso acquisite mediante telecamera o macchina fotografica, i due problemi più gravi sono la non uniformità del colore derivato dall'illuminazione degli oggetti e la mancanza di colori con molto contrasto tra oggetti vicini.

3. La soluzione adottata

Il programma è stato scritto in Borland Builder C++ 5.0, il quale estende il linguaggio di programmazione C++ a un linguaggio di programmazione a finestre. Il programma opera su immagini bitmap standard. É possibile caricare immagini di qualsiasi dimensione (il limite massimo è dato dalla memoria che mette a disposizione il sistema operativo). Esso opera sempre su copie delle immagini in memoria.

Abbiamo semplificato il problema della scelta dell'intervallo dei colori che caratterizzano il blob usando un'array di pixel chiamandolo "matrice di filtro". Si tratta di un'area di pixel di dimensioni variabili 1*1 3*3 5*5 7*7 9*9 il cui centro viene puntato dal cursore quando è posizionato sull'immagine da elaborare. Quest'area viene considerata come se contenesse tutte le informazioni sul colore che caratterizzano un blob. In pratica queste si riducono ai valori massimi e minimi di rosso, verde e blu dell'area, considerati come valori assoluti o relativi[1]. É possibile pensare a questi punti come gli angoli di un parallelepipedo contenuto nel cubo RGB (vedi appendice I). I colori che andranno a caratterizzare il blob saranno quindi quelli contenuti da questo parallelepipedo.

Concettualmente il programma svolge un percorso che può essere suddiviso in tre parti (figura 3.1):

- filtraggio del colore
- filtraggio delle dimensioni
- accrescimento dei blob

le prime due parti ricevono entrambe in ingresso un'immagine e ne generano una in uscita. La procedura di accrescimento, invece necessita dell'immagine contenente i blob e dell'immagine iniziale.

---

[1] l'operazione di passaggio da componenti di colore assolute a relative, è lineare ed è definita come:

$$r_{rel} = \frac{100 \times r_{ass}}{r_{ass} + g_{ass} + b_{ass}} \qquad g_{rel} = \frac{100 \times g_{ass}}{r_{ass} + g_{ass} + b_{ass}} \qquad b_{rel} = \frac{100 \times b_{ass}}{r_{ass} + g_{ass} + b_{ass}}$$

dove *r*, *g*, *b* sono le componenti rispettivamente di rosso, verde, blu, e i pedici rel e ass identificano rispettivamente le componenti assolute e relative.

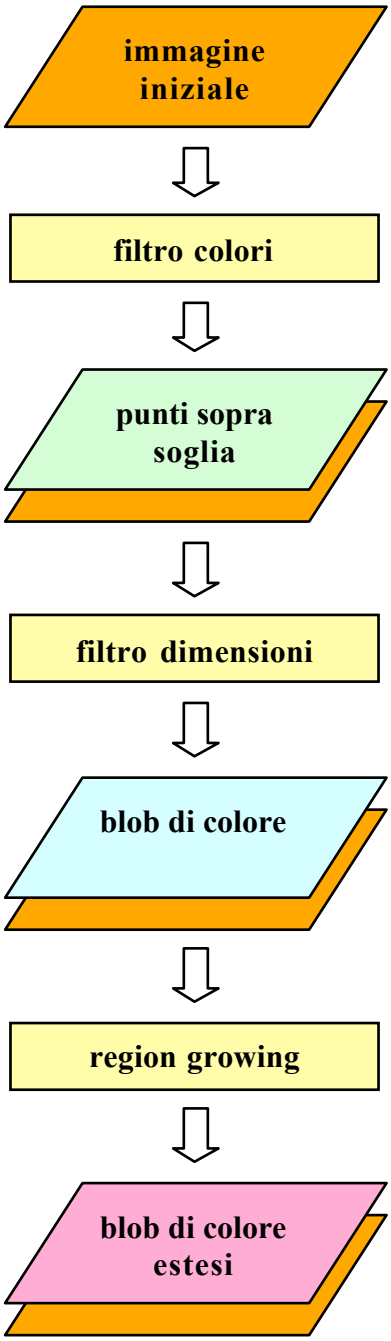


figura 3.1

schema di principio di observer

3.1. Passo 1: Filtro colore

Dopo aver definito i parametri cromatici che definiscono se un pixel è accettato o meno, il programma deve passare in rassegna pixel per pixel. I pixel dell'immagine di partenza vengono analizzati spostandosi prima dal pixel più alto a sinistra, poi in basso per tutta la colonna, quindi si passa alla colonna a destra, così via fino al pixel più in basso a destra. Ogni volta che un pixel viene riconosciuto come appartenente all'intervallo di colori definito viene scritto un pixel nero in una copia dell'immagine. Finito il processo, si avrà un'immagine con gli stessi attributi dell'originale (dimensioni, gradazioni di colore, ecc.) ma il cui "soggetto" sono i pixel che rientrano nei parametri definiti.

3.2. Passo 2: Filtro dimensioni

Arrivati a questo passo l'immagine da elaborare non è più l'originale, bensì l'immagine in bianco e nero ottenuta all'uscita del filtro colore. In quest'immagine, le aree estese di nero, che identificano i blob, sono mescolate a gruppi di punti sparsi, che possono essere considerarti come rumore. Come primo passo risulta importante definire la differenza tra blob e rumore. Abbiamo definito l'area minima di un blob come parametro per distinguere questi due tipologie di raggruppamenti di pixel.

Il filtro dimensioni agisce in maniera simile al filtro colore, infatti scandisce un'immagine pixel per pixel, colorando man mano un'immagine bianca (che possiede gli stessi attributi dell'immagine in ingresso) se i pixel che trova appartengono a un blob e cambiando colore quando identifica un nuovo blob. L'eccezione si verifica quando un pixel appartiene a due blob, in seguito viene spiegato come è stato risolto questo caso.

Per identificare i blob è stata costruita la classe **ColorBlob** (vedi appendice II). Essa contiene le informazioni necessarie per riconoscere i blob sull'immagine che viene creata con il filtro dimensioni e per calcolare i baricentri di ogni blob. I blob presenti sull'immagine sono identificati da un elemento della lista *lista_blob*, ogni volta che viene riconosciuto un nuovo blob viene aggiunto un elemento alla lista.

L'operazione di filtraggio dell'immagine avviene pixel per pixel, essa è facilmente comprensibile se si fa riferimento al pixel in questione e ai pixel contigui al pixel considerato istante per istante. I casi possibili sono:

- pixel bianco
- pixel isolato
- primo pixel del blob
- pixel con 1 blob vicino
- pixel con più blob vicini

3.2.1. pixel bianco

Il pixel viene riconosciuto come bianco e si passa subito al pixel successivo.

3.2.2. pixel isolato

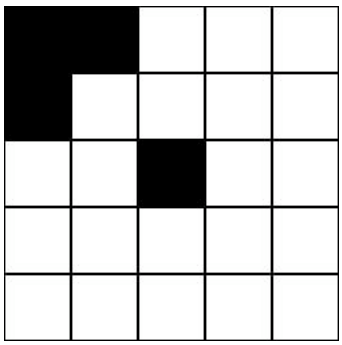


figura 3.2

pixel isolato

Il pixel è di colore nero e tutti i suoi pixel contigui sono tutti bianchi. Questo pixel non appartiene a nessun blob ed è sicuramente rumore[2]. Nell'immagine di uscita del filtro immagine sarà presente un pixel bianco.

Se si definisce la dimensione minima dei blob come unitaria o nulla, il controllo di pixel isolato non viene eseguito.

3.2.3. pixel non isolato

In questo sottoparagrafo e nei successivi si farà riferimento ai pixel contigui già scanditi. Essi, per tutti i pixel differenti dal bordo dell'immagine, sono quattro, analizzandoli è possibile arrivare a diverse conclusioni. Il colore dei pixel se diverso da bianco viene memorizzato in un vettore di quattro elementi α[3] .

---

[2] fanno eccezione eccetto i casi in cui la dimensione minima viene impostata a 0 o a 1.

[3] il vettore α, contiene nell'ordine i colori dei pixel a,b,c,d (figura 3.3) dell'immagine di uscita del filtro dimensioni, se uno di questi è bianco, non viene inserito all'interno del vettore. Il colore inserito è un intero e in forma esadecimale è rappresentabile in forma BBGGRR, dove BB, GG, RR sono rispettivamente i valori di blu, verde e rosso in scala RGB.

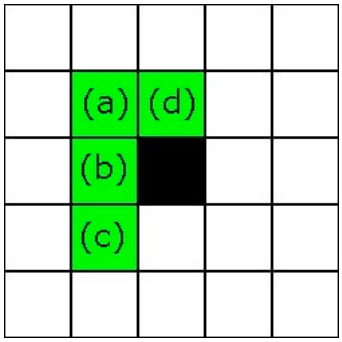


figura 3.3

pixel esaminati per la costruzione del vettore α

3.2.3.1 primo pixel del blob

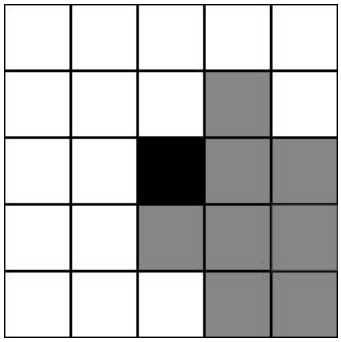


figura 3.4

primo pixel del blob

Il pixel è di colore nero, non isolato, e il vettore α è vuoto. Necessariamente il pixel è il primo pixel di un blob. A questo punto viene creato un nuovo elemento della classe **colorblob**, e aggiunto come elemento alla lista di blob. Nell'immagine di uscita dal filtro dimensioni, verrà aggiunto un pixel del colore scelto per il nuovo elemento **colorblob**.

3.2.3.2 pixel con un blob vicino

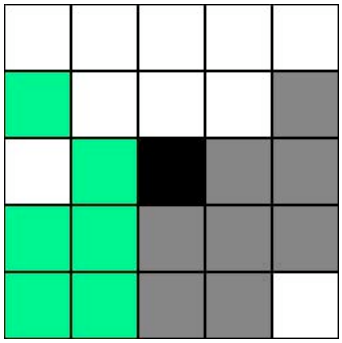


figura 3.5

pixel con un blob vicino

Il pixel è di colore nero, non isolato e il vettore α contiene uno o più elementi uguali tra loro. Viene ricercato il blob nella lista di blob, ne viene incrementata la massa nelle due dimensioni, l'area e viene salvato il proprio colore del blob. Nell'immagine di uscita dal filtro dimensioni viene aggiunto un pixel del colore del blob trovato (campo **colore**).

3.2.3.3 pixel con più blob vicini

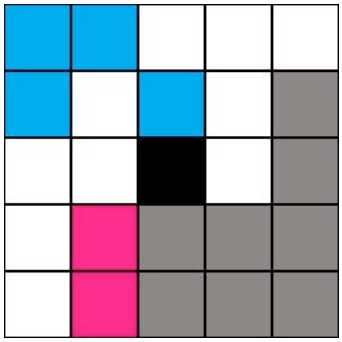


figura 3.6

pixel con 2 blob vicini

Il pixel è di colore nero, non isolato e il vettore α contiene almeno due elementi di colore diverso tra loro. Ci troviamo nel caso in cui si uniscono due o più blob che abbiamo identificato come distinti. In questo caso il pixel viene assegnato al blob di colore uguale al primo elemento del vettore α (come nel caso precedente). Successivamente viene salvato il campo **colore_blob** del blob trovato e copiato nel campo **colore_blob** di tutti gli elementi della lista con campo **colore_blob** uguale a

quello del secondo, terzo, quarto blob presenti nel vettore α. Alla fine di questa elaborazione potremmo avere un'immagine con aree colorate di colori distinti anche contigue. Le varie aree rappresentano gli elementi della lista *lista_blob* presente in memoria.

3.2.4. Ultima parte dell'elaborazione

Arrivati a questo punto possiamo dire che non tutti i blob presenti in *lista_blob* sono veri e propri blob. Infatti dato che è richiesto che un blob sia un'area isolata, questa caratteristica è assicurata solo dagli insiemi di blob aventi lo stesso numero nel campo **blob**. Definiamo *blob reali* tali insiemi.

L'ultima parte dell'elaborazione del filtro dimensione inizia con una scansione in cui viene calcolata l'area totale di tutti i blob con lo stesso **blob** (*blob reali*). Se l'area totale risulta minore della dimensione minima definita come parametro del filtro dimensioni, a questi blob viene assegnato il colore bianco. Altrimenti a tutti questi elementi (con lo stesso **blob**) viene assegnato un colore univoco. L'immagine di uscita del filtro viene ricolorata assegnando a ciascun pixel il nuovo colore del *blob reale* cui tale pixel appartiene.

Al termine di questa elaborazione gli elementi della lista di colore bianco vengono eliminati per velocizzare i successivi passaggi di calcolo dei baricentri.

3.2.5. Salvataggio baricentri

Il calcolo della posizione del baricentro di un blob avviene operando attraverso **massax, massay** e **num_pixel** (campi di **ColorBlob** vedi appendice II). Nella prima parte dell'elaborazione del filtro dimensioni essi vengono incrementati della coordinata x, della coordinata y e di 1 rispettivamente ogni volta che viene riconosciuto un pixel appartenente a un blob. Per ogni blob singolo il calcolo è:

$$x_b = \frac{\sum_{i=1}^{n} x_i \cdot m_i}{m_{tot}} = \frac{m \cdot \sum_{i=1}^{n} x_i}{n \cdot m} = \frac{\sum_{i=1}^{n} x_i}{n}$$

eq. 3.1

$$y_b = \frac{\sum_{i=1}^{n} y_i \cdot m_i}{m_{tot}} = \frac{m \cdot \sum_{i=1}^{n} y_i}{n \cdot m} = \frac{\sum_{i=1}^{n} y_i}{n}$$

eq. 3.2

dove x_b e y_b rappresentano le coordinate del baricentro, *m* la massa del punto (che sarà uguale per ogni pixel), $\sum_{i=1}^{n} x_i$ **massax,** $\sum_{i=1}^{n} y_i$ **massax** e *n* **num_pixel**.

Il calcolo finale delle coordinate del baricentro ($\hat{x}_b$, $\hat{y}_b$) di un *blob reale* sarà:

$$\hat{x}_b = \frac{\sum_k \sum_{i=1}^{n} x}{\sum_k n}$$

eq. 3.3

$$\hat{y}_b = \frac{\sum_k \sum_{i=1}^{n} y}{\sum_k n}$$

eq.3.4

dove si intende per $\sum_k \sum_{i=1}^{n} x$ e $\sum_k \sum_{i=1}^{n} y$ le sommatorie dei campi **massax**, **massay** per ogni *k* elemento con lo stesso valore del campo **blob** (*blob reale*).

3.3. Passo 3: *Region growing*

Per effettuare l'accrescimento dei blob (*region growing*) viene compiuta un'analisi pixel per pixel di una copia in bianco e nero dell'immagine di uscita dal filtro dimensioni, quando viene trovato un pixel nero si attiva una funzione che esamina ogni pixel circostante di colore bianco:

a) memorizzato il colore originale del pixel nero
b) viene memorizzato il colore originale del pixel bianco
c) viene effettuato un confronto tra i due colori memorizzati. Se la loro distanza ricade all'interno del *range* definito per il *region growing*, il pixel bianco viene colorato con il colore del blob del pixel nero (colore preso dall'immagine a colori di uscita del filtro dimensioni).

4. Modalità operative

4.1. Hardware & Software necessari

Il programma è eseguibile su qualsiasi personal computer con un sistema operativo Microsoft Windows.

4.2. Istallazione del software

É sufficiente eseguire l'archivio autoestraente `observinst.exe` presente nella directory `install/` presente sul cd-rom. Oppure è possibile copiare sul proprio hardisk la directory `programma/` presente sempre sul cd-rom.

4.3. Manuale d'uso

L'interfaccia è composta da 4 differenti schermate che rappresentano i 4 stati del programma.

La schermata iniziale, mostra solamente un'immagine di background.

Subito dopo aver selezionato un'immagine da analizzare si passa alla schermata successiva dove l'immagine selezionata occupa una buona parte della finestra, sotto di essa vengono visualizzati i parametri di Filtro colore e a fianco la loro rappresentazione grafica (insieme dei colori selezionati). Questa schermata rappresenta lo stato precedente all'elaborazione di filtraggio dei colori.

Terminato il filtraggio colori, nella nuova schermata, l'immagine iniziale è stata sostituita con l'immagine di uscita del filtro colori. I colore che è stato scelto per colorare questa immagine è il colore complementare alla media dei colori selezionati precedentemente nei parametri di filtro. Si può selezionare di visualizzare quest'immagine anche in bianco e nero. Per visualizzare l'immagine iniziale sotto questa è sufficiente selezionare il pulsante a destra in alto che contiene l'immagine originale rimpicciolita. Nella parte dello schermo in basso a sinistra sono visualizzati i parametri del precedente filtraggio. In basso a destra è visualizzato l'unico parametro

necessario per il filtro dimensioni: la dimensione minima dei blob. Questa schermata rappresenta lo stato precedente all'elaborazione del filtraggio delle dimensioni.

L'ultima schermata rappresenta lo stato finale (in cui sono presenti i blob) e di region growing. In basso a sinistra è presente un riassunto delle precedenti elaborazioni (in pratica è stato aggiunto il valore di dimensione di dimensione minima del blob al riassunto dei parametri di filtro colore), nella parte a destra dello schermo sono presenti i parametri di accrescimento dei blob. Di seguito sono inserite le immagini delle tre schermate.

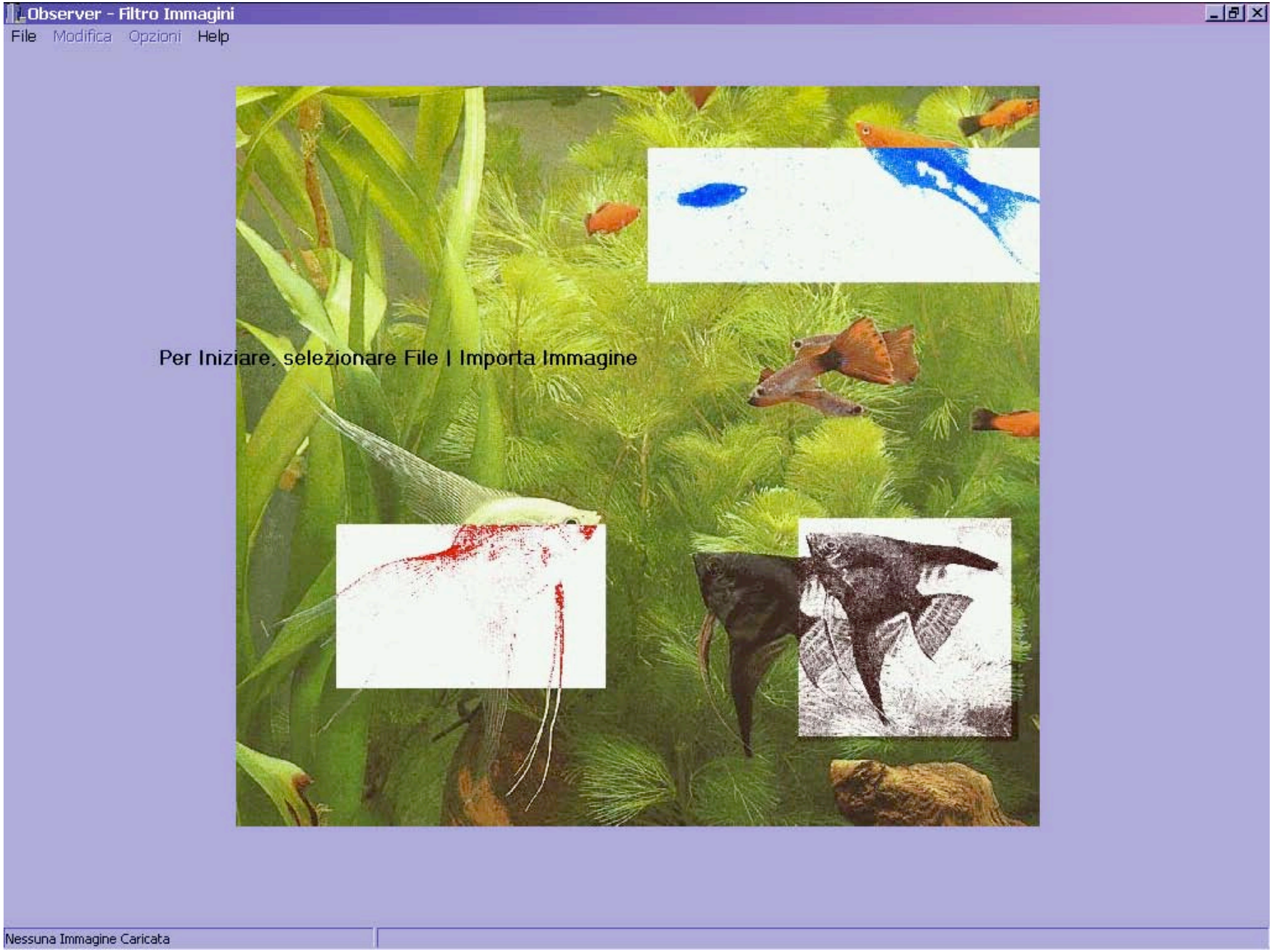
Immagini di seguito:

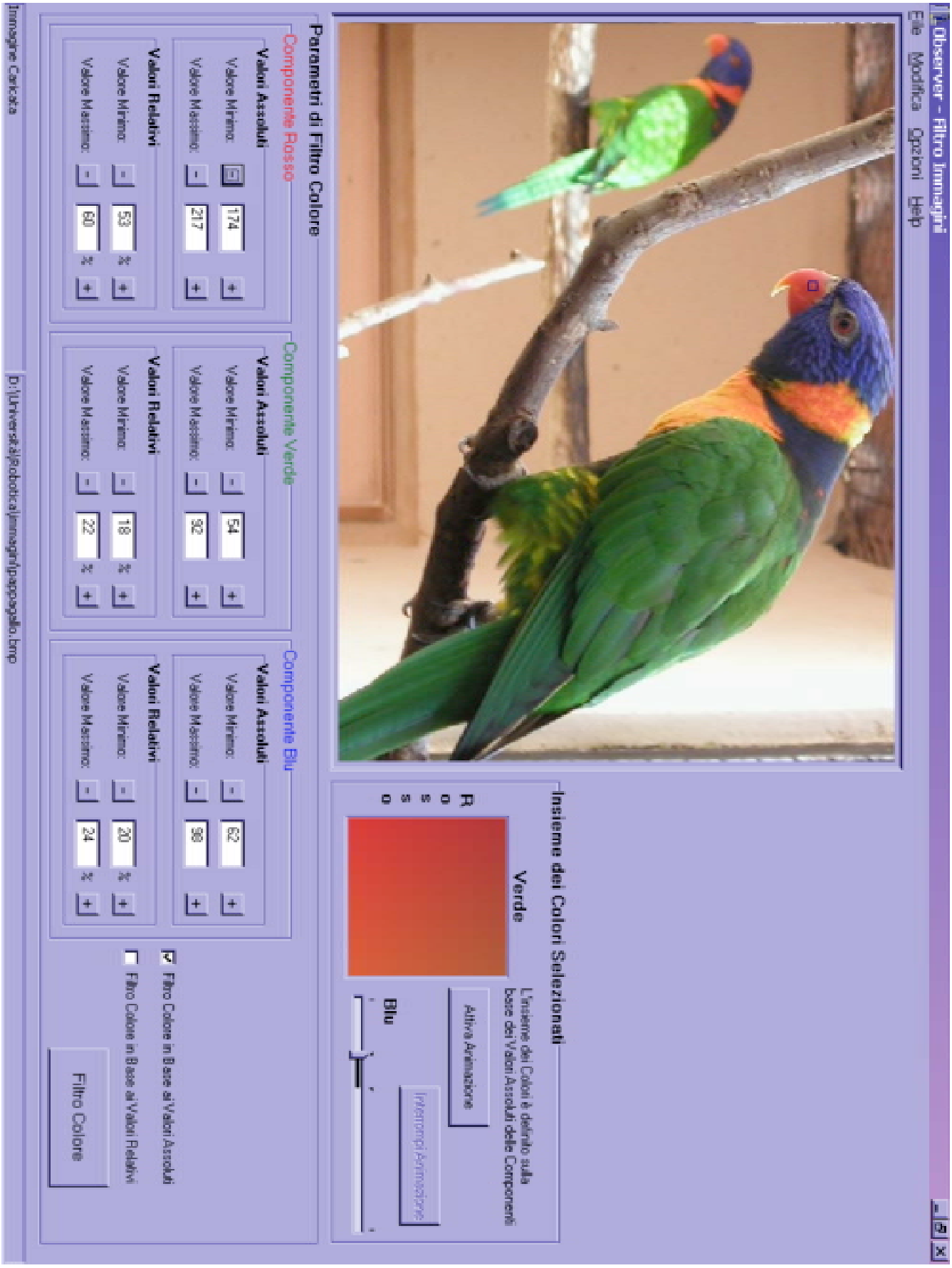
a pagina 13, figura 4.1 **schermata iniziale**

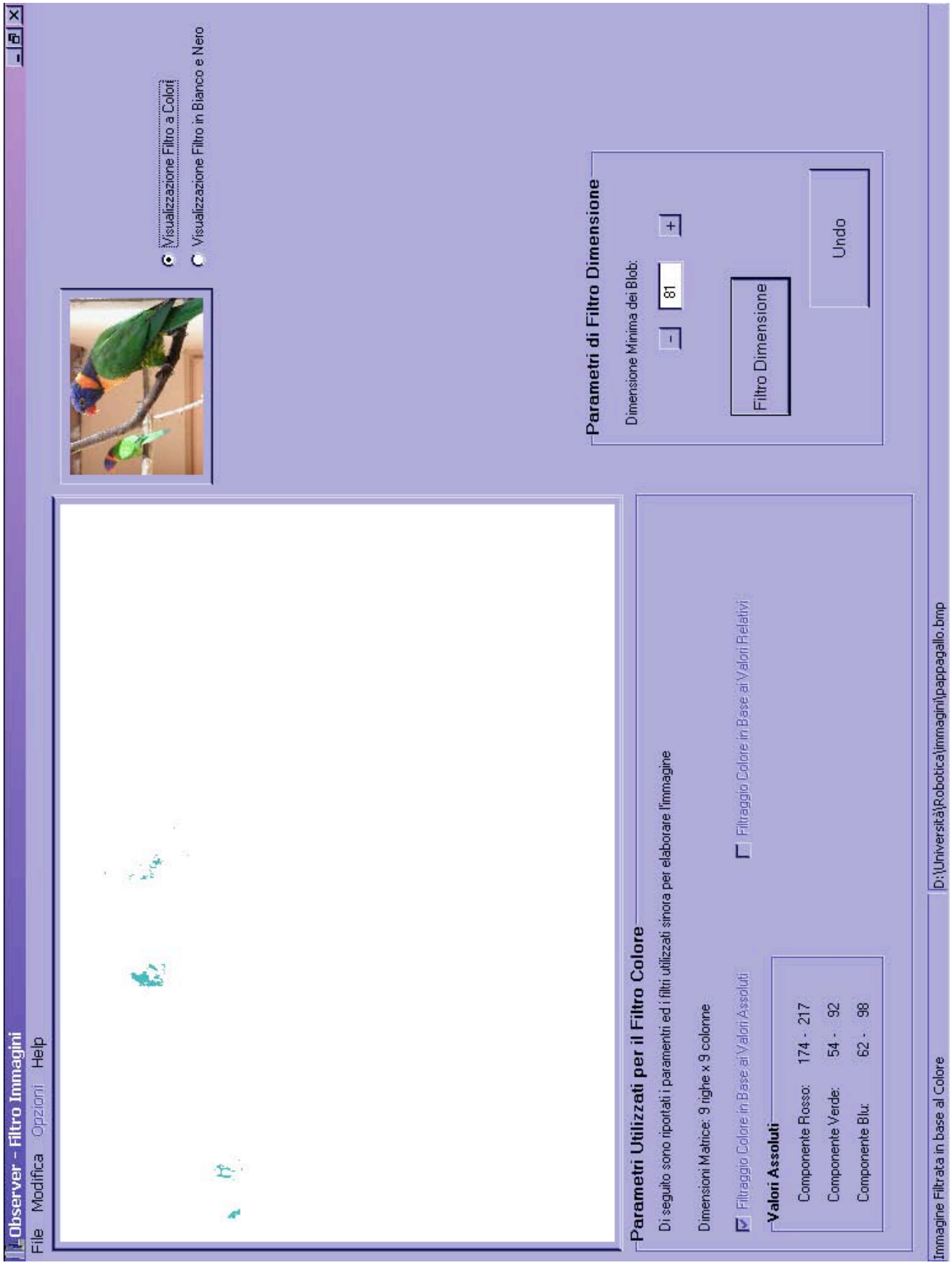
a pagina 14, figura 4.2 **prima schermata**

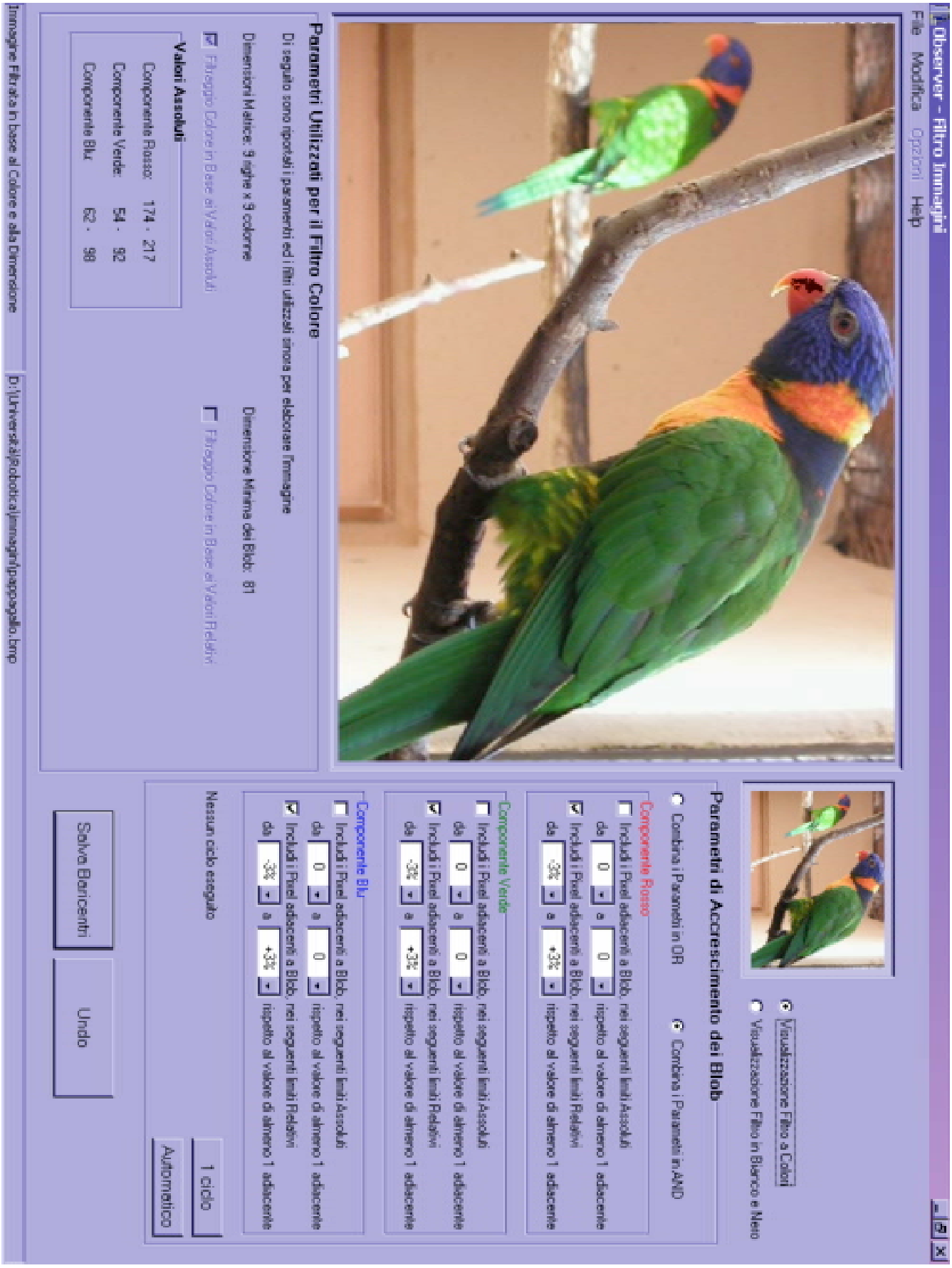
a pagina 15, figura 4.3 **seconda schermata**

a pagina 16, figura 4.4 **schermata finale**.


Observer - Filtro Immagini
File Modifica Opzioni Help
Per Iniziare, selezionare File | Importa Immagine
Nessuna Immagine Caricata


Observer - Filtro Immagini
File Modifica Opzioni Help
Insieme dei Colori Selezionati
L'insieme dei Colori è definito sulla base dei Valori Assoluti delle Componenti
Rosso
Verde
Blu
Attiva Animazione
Interrompi Animazione
Parametri di Filtro Colore
Componente Rosso
Componente Verde
Componente Blu
Valori Assoluti
Valori Relativi
Valore Minimo:
Valore Massimo:
174
217
53
60
54
92
18
22
62
98
20
24
Filtro Colore in Base ai Valori Assoluti
Filtro Colore in Base ai Valori Relativi
Filtro Colore
Immagine Caricata
D:\Universita\Robotica\Immagini\pappagallo.bmp


Observer - Filtro Immagini
File Modifica Opzioni Help
Visualizzazione Filtro a Colori
Visualizzazione Filtro in Bianco e Nero
Parametri di Filtro Dimensione
Dimensione Minima dei Blob:
-
81
+
Filtro Dimensione
Undo
Parametri Utilizzati per il Filtro Colore
Di seguito sono riportati i parametri ed i filtri utilizzati sinora per elaborare l'immagine
Dimensioni Matrice: 9 righe x 9 colonne
Filtraggio Colore in Base ai Valori Assoluti
Filtraggio Colore in Base ai Valori Relativi
Valori Assoluti
Componente Rosso: 174 - 217
Componente Verde: 54 - 92
Componente Blu: 62 - 98
Immagine Filtrata in base al Colore
D:\Università\Robotica\immagini\pappagallo.bmp


Observer - Filtro Immagini
File Modifica Opzioni Help
Visualizzazione Filtro a Colori
Visualizzazione Filtro in Bianco e Nero
Parametri di Accrescimento dei Blob
Combina i Parametri in OR
Combina i Parametri in AND
Componente Rosso
Componente Verde
Componente Blu
Nessun ciclo eseguito
1 ciclo
Automatico
Parametri Utilizzati per il Filtro Colore
Di seguito sono riportati i parametri ed i filtri utilizzati onora per elaborare l'immagine
Dimensioni Matrice: 9 righe x 9 colonne
Dimensione Minima dei Blob: 81
Filtraggio Colore in Base ai Valori Assoluti
Filtraggio Colore in Base ai Valori Relativi
Valori Assoluti
Componente Rosso: 174 - 217
Componente Verde: 54 - 92
Componente Blu: 62 - 98
Salva Baricentri
Undo
Immagine Filtrata in base al Colore e alla Dimensione
D:\Universita\Robotica\Immagini\pappagallo.bmp

4.3.1. caricamento immagine

figura 4.5

selezione immagine

La selezione dell'immagine all'interno delle directory del computer si esegue in maniera canonica. Se il programma effettua errori di apertura, questi vengono segnalati.

4.3.2. visualizzazione immagine

Una volta selezionata l'immagine questa viene caricata nella "finestra principale" e ridimensionata alle dimensioni della finestra (600*450 pixel). Questo ridimensionamento può essere percepito come distorsione se l'immagine risulta più alta che larga. Dato che le normali telecamere hanno un rapporto fra le due dimensioni di circa 4:3 , questo tipo di ridimensionamento non crea problemi.

La risoluzione dell'immagine visualizzata è 74 dpi (lo standard di visualizzazione in ambiente microsoft), questo comporta solo una differente visualizzazione, il programma lavora su immagini della stessa densità dell'originale.

4.3.3. selezione colore

figura 4.6

selezione colore

La prima azione per riconoscere un blob è selezionarne il colore. Per selezionarlo, la via più semplice è la selezione diretta sull'immagine. Spostando il cursore sull'immagine viene visualizzato il contorno della matrice del colore.

Per modificare la dimensione della matrice, basta selezionare Opzioni -> Dimensione Matrice sulla barra degli strumenti. É possibile selezionare matrici quadrate di dimensioni 1,9,25,49,81 pixel.

Per visualizzare in maniera comoda è possibile modificare il colore del cursore sull'immagine (il contorno della matrice di colore) selezionando Opzioni-> Bordo Regione di filtro.

4.3.4. impostazioni filtraggio colore

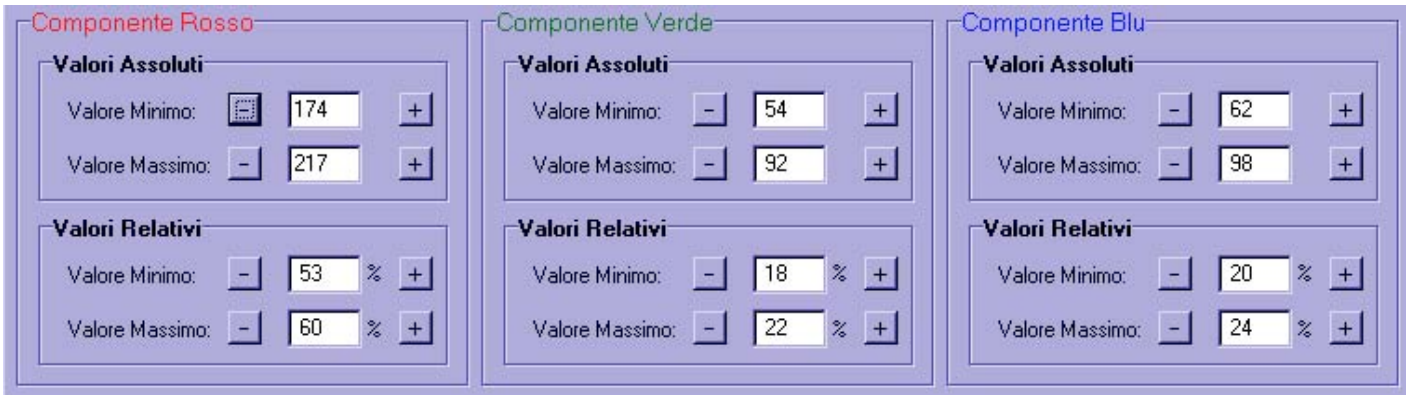


figura 4.7a

parametri di filtro colore

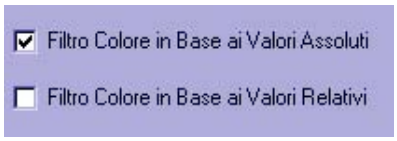


figura 4.7b

parametri di filtro colore (*continua*)

Selezionando un'area sull'immagine, vengono settati automaticamente i parametri di filtraggio. Vengono infatti impostati come margini del filtro colore i valori massimi e minimi delle componenti di colore (sia in termini relativi, che assoluti) della regione che definisce la matrice di filtro. Questi vengono visualizzati automaticamente e inoltre è possibile modificare ogni singolo parametro immettendo il valore manualmente nel campo apposito o utilizzando i pulsanti attivi a lato di ogni campo.

É inoltre possibile filtrare l'immagine sia in base ai margini cromatici relativi che assoluti, o entrambi[4], per fare questo basta selezionare un apposito campo prima di eseguire la scansione.

4.3.5. insieme dei colori selezionati

figura 4.8

rappresentazione dell'insieme dei colori selezionati

Per avere un'idea dell'insieme dei colori che caratterizza il filtro colore, è presente un'immagine che rappresenta i colori selezionati.

Essa è definita come una sezione sugli assi R e G di un parallelepipedo (definito al capitolo 3) appartenente al cubo RGB all'altezza B. É possibile variare manualmente

[4] in questo caso il programma filtra ogni pixel controllando che esso appartenga all'intersezione del range dei parametri cromatici relativi con il range dei parametri cromatici assoluti .

l'altezza B con una barra a fianco dell'immagine, o automaticamente attivando l'animazione con l'apposito pulsante.

figura 4.9

risultato elaborazione filtro dimensioni con immagine iniziale di sfondo

4.3.6. selezione visualizzazione

figura 4.10

selezione immagine di sfondo e visualizzazione colori

Come spiegato nel paragrafo 4.3 è possibile visualizzare l'immagine iniziale come sfondo dell'immagine di uscita sia del filtro colori che del filtro dimensioni. L'operazione si esegue selezionando il pulsante che contiene l'immagine iniziale rimpicciolita. Quest'opzione è stata inserita per fornire un aiuto all'utente nella visualizzazione di ciò che il programma riconosce come blob. Inoltre è possibile

colorare i risultati di entrambi i filtri in nero selezionando la visualizzazione in bianco e nero.

4.3.7. impostazioni filtraggio dimensioni

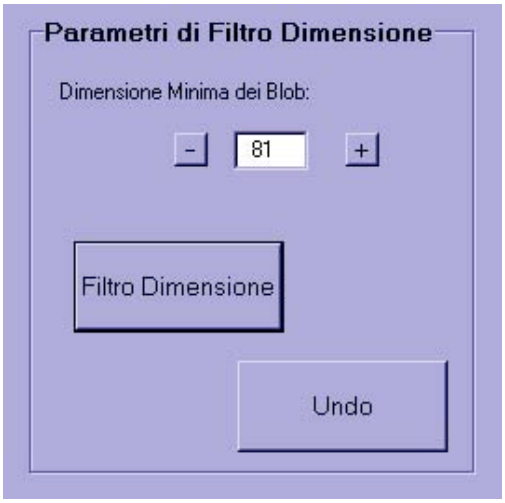


figura 4.11

parametri di filtro dimensioni

I parametri di filtraggio dimensioni si riducono alla selezione di un singolo parametro: la dimensione minima del blob. Il valore di default è pari all'area in pixel della matrice colori selezionata.

4.3.8. *region growing*

Effettuato il filtraggio dimensioni, è possibile aumentare le dimensioni dei blob mediante il *region growing*. I parametri che definiscono i pixel che verranno aggiunti sono selezionabili mediante una maschera concettualmente molto simile a quella per selezionare i parametri di filtro colore (figure 4.7a 4.7b e 4.14).

Una volta selezionati i parametri è possibile effettuare più volte il *region growing*. Ogni volta che viene eseguito, l'accrescimento lavora al massimo aggiungendo un pixel al contorno di ogni blob. É possibile far funzionare il *region growing* in modalità 'Automatico', la procedura viene eseguita finché ci sono pixel che rientrano nei parametri, fino a un massimo di 50 ripetizioni. Questo limite è stato inserito poiché, se si definiscono dei parametri "larghi", è possibile che i blob vengano estesi fino a ricoprire l'intera immagine.

Le figure 4.13 e 4.14 mostrano due esempi *region growing*, effettuato sul blob riconosciuto (figura 4.9) con i parametri definiti in figura 4.12.

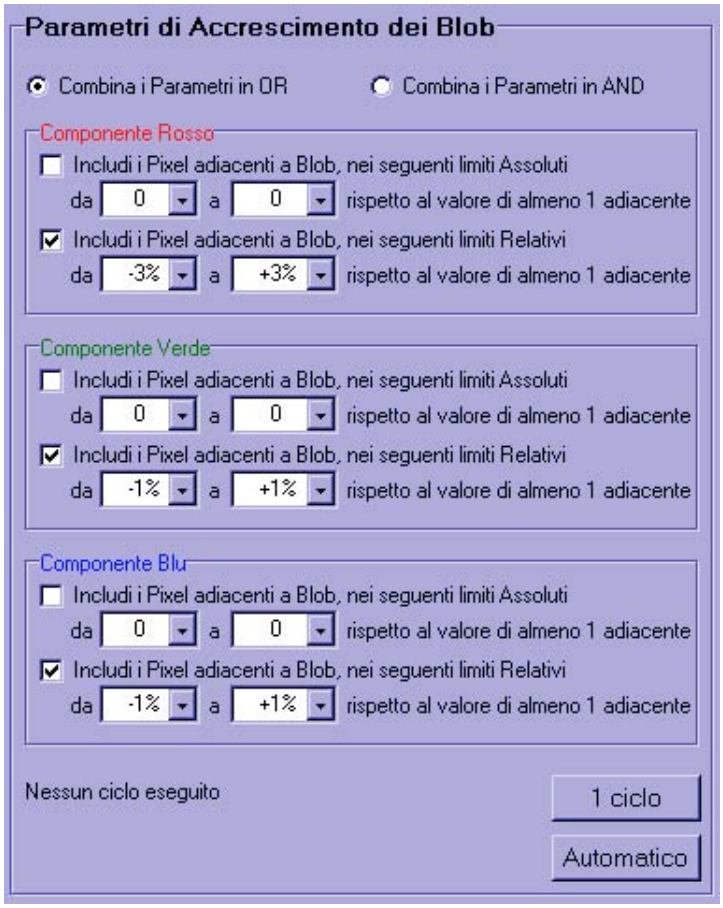


figura 4.12

parametri di region growing

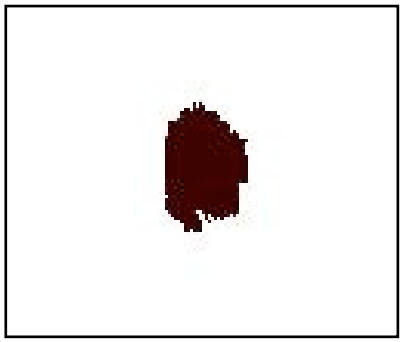


figura 4.13a

il blob dopo 10 elaborazioni di region growing *(sfondo bianco)*

figura 4.13b

il blob dopo 10 elaborazioni di region growing *(immagine iniziale di sfondo)*

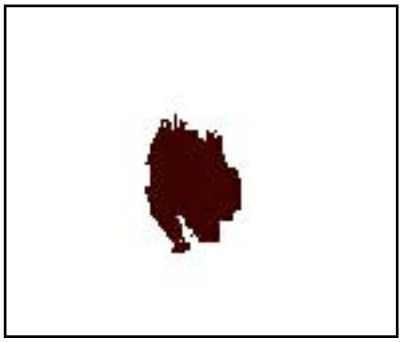


figura 4.14a

il blob dopo 20 elaborazioni di region growing *(sfondo bianco)*

figura 4.14b

il blob dopo 20 elaborazioni di region growing *(immagine iniziale di sfondo)*

4.3.9. salvataggio baricentri

Una volta eseguito il filtro dimensioni è possibile salvare i baricentri dei blob riconosciuti in un file di testo. Il nome e la posizione del file sono selezionabili in maniera standard. Il file di uscita è formato da un'intestazione, che commenta il file, e da una tabella formattata come 'numero del blob' 'x baricentro' 'y baricentro' 'area del blob'. Le coordinate del baricentro e l'area del blob sono espressi in pixel.

```
# blob: x baricentro, y baricentro, area del blob

Le coordinate del Baricentro si riferiscono ai pixel dell'immagine.

  1: 284.01,  92.90,    317
```

figura 4.15

contenuto del file `baricentri.txt`

5. Conclusioni e sviluppi futuri

Attualmente:

- il programma è in grado di elaborare qualunque immagine bitmap
- possiede un'elaborazione veloce nella parte di analisi pixel per pixel
- possiede un'interfaccia grafica *user friendly*
- offre una completa personalizzabilità di ogni parametro di impostazione dei vari filtraggi

I possibili sviluppi di questo software possono essere:

- elaborazione *realtime* delle immagini e del calcolo dei baricentri (per il raggiungimento di un oggetto dal robot)
- selezione di colore di filtro da parte dell'utente non tramite valori numerici ma in maniera grafica
- unione dei blob contigui dopo il *region growing*

Appendice I: Sistema RGB di rappresentazione del colore

Esistono vari sistemi per rappresentare un'immagine a colori. Quello *RGB* (Red-Green-Blue), senza dubbio tra i più conosciuti, è un sistema di riferimento in cui ogni colore viene individuato dalle sue componenti rossa, verde e blu. Queste rappresentano, infatti, i colori fondamentali grazie ai quali è possibile ottenerne qualunque altro, ovvero ogni colore può essere interpretato come una combinazione lineare dei tre fondamentali. Immaginando di prendere un sistema di riferimento cartesiano in tre dimensioni, ed associando ad ogni asse uno dei tre colori base, nasce *il cubo RGB*.

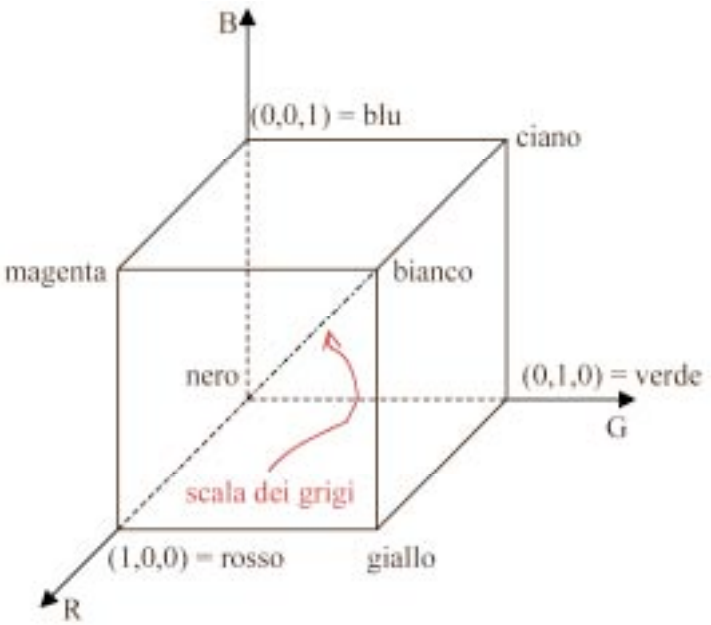


figura A1.1

cubo RGB

Ogni punto all'interno del cubo rappresenta un colore e le sue proiezioni sui tre assi indicano le sue componenti di rosso, verde e blu. Il vertice del cubo che coincide con l'origine degli assi corrisponde al colore con tutte le componenti nulle, cioè il nero. Percorrendo i tre spigoli del cubo che partono dall'origine, si osservano tutte le gradazioni dei tre colori fondamentali. Il vertice del cubo opposto all'origine corrisponde al bianco, che è il colore con tutte le tre componenti al massimo valore. Lungo gli spigoli del cubo che partono dal vertice che rappresenta il bianco, si possono osservare altri tre colori in tutte le loro gradazioni: il ciano, il magenta e il giallo.

Ogni colore si può, quindi, esprimere anche in funzione delle componenti ciano, magenta e giallo, che rappresentano così una seconda terna di colori fondamentali. Il

sistema CMY (Cyan-Magenta-Yellow) è complementare a quello RGB, e viene spesso utilizzato in tipografia in quanto l'assenza di colore si identifica nel bianco, che è solitamente il colore del supporto tipografico. Al contrario, per realizzare il colore nero si devono utilizzare tutti i tre colori. Per motivi analoghi, il sistema RGB è idoneo per applicazioni in cui lo sfondo è nero. In ambito informatico, tra i due sistemi menzionati, quello RGB è sicuramente il più utilizzato e, nel seguito, si farà sempre riferimento a questo. Inoltre, solitamente viene dedicato un byte ad ognuna delle tre componenti, quindi queste possono assumere un qualsiasi valore compreso tra 0 e 255 e il numero complessivo di combinazioni (cioè colori) ottenibili è 256^3=16.777.216. Questo numero ragguardevole si avvicina al massimo numero di colori che l'occhio umano può distinguere. Le immagini a colori di alta qualità hanno, quindi, una profondità di 24 bit (8 per ciascuna delle tre componenti) e vengono anche dette *true color*. C'è, infine, un terzo sistema molto utilizzato nell'ambito dell'elaborazione delle immagini, il sistema HSI (Hue-Saturation-Intensity) o HSV (Hue-Saturation-Value), in cui ogni colore viene rappresentato univocamente da tre componenti dette *tinta*, *saturazione* e *luminosità*.

Appendice II: Classe ColorBlob

ColorBlob	
int colore	colore dei pixel appartenenti a questo elemento, viene utilizzato per identificare univocamente questo elemento nella lista
long num_pixel	numero totali dei pixel appartenenti a questo elemento
long massaX, massaY	somma delle coordinate dei pixel appartenenti al blob nelle 2 direzioni
int blob	identificativo degli elementi adiacenti sull'immagine tra loro (*blob reale*)
int colore_blob	colore del *blob reale*
bool bar	flag di supporto per il calcolo dei baricentri
Long GetNumPixel()	restituisce il numero dei pixel appartenenti all'elemento
void AddPixel(int x, int y)	aggiunge un pixel all'elemento e incrementa massaX e massaY
int GetColore()	restituisce il colore identificativo dell'elemento
void SetColore(int c)	imposta il colore identificativo dell'elemento
int GetBlob()	restituisce il numero della *blob reale* a cui appartiene l'elemento
void SetBlob(int b)	imposta il numero del *blob reale* a cui appartiene l'elemento
int GetColoreBlob()	restituisce il colore identificativo del *blob reale* dell'elemento
void SetColoreBlob(int c)	imposta il colore identificativo del *blob reale* dell'elemento
bool Bar()	restituisce il valore della variabile bar
void ResetBar()	imposta il valore della variabile bar a false
long AddMassa(float *mX, float *mY)	incrementa *mX e *mY di massaX e massaY rispettivamente e restituisce il numero dei pixel appartenenti all'elemento

tabella A2.1

Bibliografia

[1] Cagno, M., Manenti, V., Panteghini, M.: "Tobor ricerca e afferra lattine", Laboratori di Robotica Avanzata, Facoltà di Ingegneria, Università di Brescia, settembre 2001.

[2] Carletti, M.: "Realizzazione di un sistema stereoscopico ibrido omnidirezionale/pin-hole e tecniche di visione artificiale per la robotica autonoma", Facoltà di Ingeneria, Università degli studi di Parma, marzo 2002.

Indice