Elaborazione di Segnali Multimediali
a.a. 2017/2018

Elaborazione di immagini a colori

L.Verdoliva

In questa esercitazione vedremo come si elaborano le immagini a colori in Matlab estendendo le tecniche di enhancement e filtraggio definite per immagini monocromatiche.

1 Lo spazio RGB

Il numero di bit usati per rappresentare ogni pixel nello spazio RGB è detta *profondità*; se per esempio ognuna delle tre componenti è a 8 bit, ogni pixel a colori ha una profondità di 24 e l'immagine è detta *full color*. In tal caso i possibili colori che si possono visualizzare sono $(2^8)^3 = 16777216$ e vengono spesso mostrati mediante il cubo RGB, i cui vertici sono rappresentati dai colori primari (R, G, B) e secondari (C, M, Y). Nella seguente tabella si mostrano i relativi valori RGB:

Colore		**Valore RGB**
Nero	1	[0 0 0]
Blu	2	[0 0 1]
Verde	3	[0 1 0]
Ciano	4	[0 1 1]
Rosso	5	[1 0 0]
Magenta	6	[1 0 1]
Giallo	7	[1 1 0]
Bianco	8	[1 1 1]

Proviamo allora a generare il cubo dei colori. Un cubo è individuato da 8 vertici che formano 6 lati. I vertici possono essere specificati mediante i valori RGB definiti nella tabella, mentre le facce da 4 valori (da 1 a 8) relativi ai 4 vertici e la cui associazione è sempre mostrata in tabella. Il codice è il seguente:

```
vertici = [0 0 0; 0 0 1; 0 1 0; 0 1 1; 1 0 0; 1 0 1; 1 1 0; 1 1 1];
facce = [1 5 6 2; 1 3 7 5; 1 2 4 3; 2 4 8 6; 3 7 8 4; 5 6 8 7];
colori = vertici;
patch('Vertices', vertici, 'Faces', facce, ...
      'FaceVertexCdata', colori, 'FaceColor', 'interp');
view([10 10 10]); axis square; axis off;
```

Il

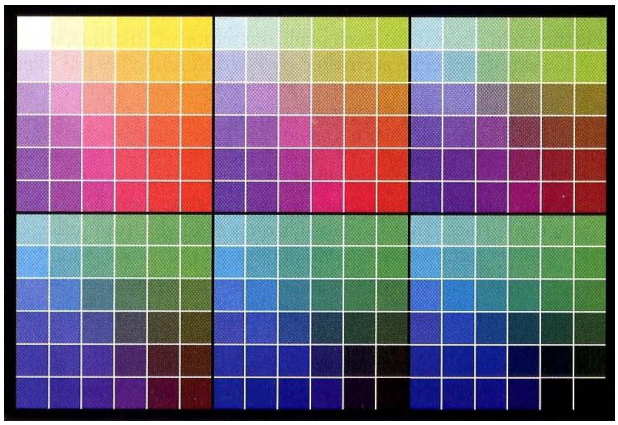

Figura 1: I 216 colori sicuri.

comando `patch` permette di creare il cubo specificando i vertici (`'Vertices'`), le facce (`'Faces'`), i colori di ogni vertice (`'FaceVertexCData'`) e di ogni lato (`'Facecolor'`) che vanno calcolati tramite interpolazione dei colori dei vertici usando l'opzione (`'interp'`). Se preferite eliminare la cornice nera che definisce il cubo dovete inserire un'ulteriore opzione nel comando `patch`: `'EdgeAlpha', 0`. Il comando `view` invece consente di settare l'angolo di vista da cui l'osservatore guarda il grafico tridimensionale. Se per esempio ponete `view(10,0,0)` potrete osservare il lato del cubo con il piano rosso-giallo-bianco-magenta, mentre con `view(0,0,-10)` quello nero-rosso-giallo-verde. Notate che il cubo è da intendersi pieno, cioè i punti rappresentativi di colori validi sono sia sulla sua superficie che al suo interno.

In realtà a meno di considerare monitor ad elevata risoluzione è molto difficile osservare su di uno schermo tutti questi colori. Spesso le schede video si limitano alla visualizzazione di soli 256 colori di questi solo 216 sono comuni alla maggior parte dei sistemi, nel senso che sono stati standardizzati al fine di garantirne la riproducibilità indipendentemente dalle capacità dell'hardware. Vengono perciò detti *safe colors* (colori sicuri o colori web) e si usano ogni qualvolta si vuole che i colori che si vedono su monitor diversi siano esattamente gli stessi. In tal caso, ognuna delle tre componenti RGB può assumere solo i valori 0, 51, 102, 153, 204, 255.

1.1 Esercizi proposti

1. *Lo spazio CMY e CMYK.* Se le componenti RGB sono state normalizzate nel range $[0, 1]$ è molto facile ottenere le componenti nello spazio CMY (Cyan, Magenta, Yellow), dato che queste componenti sono le complementari di quelle RGB:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

   Inoltre è possibile usare la funzione `imcomplement` di matlab direttamente sull'immagine a colori. Provate a scrivere una funzione con il seguente prototipo: `function y=rgb2cmy(x)`, che legge un'immagine a colori e determina la rappresentazione CMY dell'immagine fragole.jpg visualizzandone le componenti.

   Scrivete poi una funzione dal prototipo: `function y=rgb2cmyk(x)` che legge un'immagine a colori e determina la rappresentazione CMYK (Cyan, Magenta, Yellow, Black) di un'immagine visualizzandone le componenti. Tenete presente che quando si usano i pigmenti, il nero si ottiene usando uguali quantità

dei pigmenti ciano, magenta e giallo, e quindi la componente K è pari al minimo fra C, M ed Y, e le nuove componenti dei pigmenti si ottengono sottraendo K, cioè C'=C-K, e così via.

2. *Lo spazio HSI.*

   Il toolbox image di Matlab consente di effettuare in modo immediato il passaggio tra spazi di colore diverso, in particolare è possibile ottenere la rappresentazione HSV con il comando `rgb2hsv`. Il modello HSV è leggermente diverso da quello HSI, dato che il solido di riferimento è una piramide (a base esagonale o circolare) rovesciata, con la cima nell'origine a differenza del caso HSI in cui il modello è una doppia piramide a base triangolare, esagonale o circolare. Per questo motivo per il passaggio nello spazio HSI usate le funzioni disponibili sul sito del corso nella sezione materiale didattico.

   Visualizzate le componenti HSI dell'immagine fragole.jpg e quelle dell'immagine cubo.jpg che rappresenta proprio il cubo dei colori. In quest'ultimo caso si possono fare alcune interessanti considerazioni. Nell'immagine di tinta, si può notare la forte discontinuità lungo la linea a 45^o sul piano frontale del cubo, che è quello del rosso: si ha infatti lungo questa linea la transizione brusca tra valori alti (360^o) e valori bassi (0^o) della tinta, dovuta alla sua rappresentazione circolare. L'immagine di saturazione mostra valori più scuri verso il vertice del bianco, dove i colori diventano progressivamente meno saturi. Infine, nell'immagine di intensità, ogni pixel è semplicemente la media dei valori RGB del pixel corrispondente nell'immagine a colori.

2 Tecniche per l'elaborazione

I metodi di elaborazione di immagini su scala di grigi descritti nelle precedenti esercitazioni possono essere applicate facilmente alle immagini a colori, lavorando sulle singole componenti definite dallo spazio di colore in cui ci troviamo. Tuttavia, il risultato dell'elaborazione di ogni singola componente non è di solito equivalente all'elaborazione congiunta di tutti i piani (elaborazione vettoriale). Affinché il processing per componente e quello vettoriale siano equivalenti devono essere soddisfatte due condizioni:

1. l'algoritmo deve potersi applicare sia a scalari che vettori;
2. l'operazione su ogni componente del vettore deve essere indipendente dalle altre.

Ovviamente non sempre tali condizioni sono verificate, tuttavia di seguito focalizzeremo l'attenzione su quelle tecniche che operano sulle singole componenti dell'immagine. In teoria, ogni trasformazione può essere realizzata in uno qualunque degli spazi di colore, in pratica però alcune operazioni si adattano meglio a modelli specifici. Supponiamo per esempio di voler semplicemente modificare l'intensità di un'immagine. Nello spazio HSI basta modificare solo la componente I dell'immagine, lasciando le altre due inalterate. Invece, nello spazio RGB, così come in quello CMY, è necessario modificare tutte e tre le componenti. Sebbene la trasformazione nello spazio HSI coinvolga il minor numero di operazioni, ciò va barattato con il costo computazionale di convertire l'immagine da RGB a HSI. Di seguito analizzeremo le trasformazioni del colore (*color mapping*), in cui si elaborano i pixel di ogni piano di colore basandosi esclusivamente sul loro valore (sono in effetti le corrispondenti delle operazioni puntuali viste per le immagini monocromatiche) e le elaborazioni spaziali, in cui si effettua il filtraggio spaziale di ogni piano di colore.

2.1 Negativo

Nel cerchio mostrato in fig.1 i colori che sono l'uno opposto dell'altro si dicono complementari tra loro. L'interesse nei complementari deriva dal fatto che sono analoghi ai negativi delle immagini su scala di grigio. Sono quindi utili per enfatizzare i dettagli nascosti nelle regioni scure delle immagini a colori. Per ottenere il negativo dell'immagine basta effettuare il negativo di ogni componente RGB (nello spazio HSI invece bisogna utilizzare trasformazioni diverse per le tre componenti). Tenete presente che in matlab le operazioni elementari agiscono

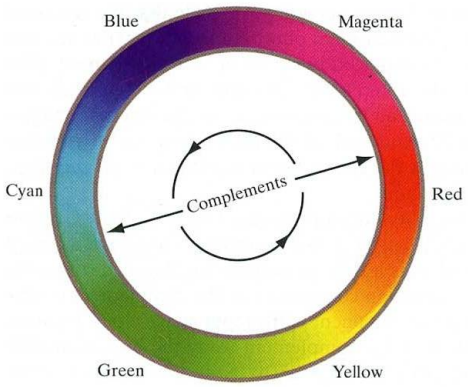

Figura 2: Cerchio dei colori.

direttamente sulle tre componenti dell'immagine. Provate allora a visualizzare il negativo dell'immagine fragole.jpg, noterete come somigli al negativo dei film a colori: il rosso è sostituito col ciano (suo complementare), il bianco con il nero e così via. Confrontate il risultato con quello che vi fornisce la funzione `imcomplement`.

2.2 Correzione di toni e colori

Le trasformazioni usate per modificare i toni di un'immagine sono selezionate in modo interattivo, nel senso che sperimentalmente si trova l'operazione che migliora la luminosità o il contrasto di un'immagine. Negli spazi RGB e CMY ciò significa che ognuna delle componenti sarà soggetta a tale trasformazione, mentre nello spazio HSI solo l'intensità sarà modificata. Per esempio, nel caso in cui si voglia migliorare il contrasto dell'immagine colori.jpg si può realizzare l'operazione potenza:

```
x = imread('colori.jpg');
figure(1); imshow(x,[]);
x=double(x);
y = x.^4;
y = y/max(y(:));   % normalizzazione nel range [0,1]
figure(2); imshow(y,[]);
```

La normalizzazione è resa necessaria dal fatto che um'immagine a colori in formato double viene visualizzata correttamente solo se normalizzata. Modificate il valore di γ e osservate gli effetti sull'immagine. Si consideri poi l'immagine montagna.jpg, che presenta un aspetto complessivamente molto scuro. Si determini il valore di γ che consente di migliorarne il contrasto. Provate a realizzare la stessa operazione nello spazio HSI (lavorando solo sull'intensità) e confrontate i risultati.

2.3 Equalizzazione

Ricordiamo che l'equalizzazione permette di ottenere un'immagine con istogramma approssimativamente piatto. In generale non ha molto senso equalizzare indipendentemente le componenti di un'immagine a colori, dato che viene modificata la proporzione con cui sono stati mescolati i colori primari. La modifica indipendente delle tre componenti cambia la tinta e la saturazione di ogni pixel introducendo una distorsione cromatica (gli istogrammi relativi alle tre immagini sono diversi quindi diversa risulterà la trasformazione applicata).

L'approccio corretto è quello di elaborare solo l'intensità del colore, lasciando tinta e saturazione intatti. Da questo punto di vista è corretto rappresentare l'immagine nello spazio HSI ed equalizzare solo la componente di intensità (I). In realtà se si volesse tener conto anche della percezione del colore da parte del sistema visivo umano (più sensibile al verde che al rosso e al blu) si potrebbe utilizzare il modello YUV o YCbCr in cui la luminanza Y è data da una combinazione lineare dei primari, con pesi non uguali, a differenza del modello HSI, in cui I pesa ugualmente le componenti RGB.

Considerate l'immagine a colori `volto.tiff` e provate ad effettuare l'equalizzazione sia nello spazio RGB su ognuna delle componenti che nello spazio HSI solo sulla componente I (ricordate di ritornare nello spazio RGB per la visualizzazione). Confrontate le immagini ottenute con i due approcci. Cosa potete osservare?

2.4 Color balancing

Spesso può essere utile realizzare un bilanciamento dei colori (*color balancing*) allo scopo di regolare le componenti di colore di un'immagine. Ci sono diversi possibili approcci, in ogni caso bisogna sempre tener conto che ogni operazione influenza l'equilibrio globale dei colori, dato che la percezione di un colore dipende anche da quelli circostanti. La proporzione di un colore può essere aumentata diminuendo la quantità di colore opposto. Allo stesso modo può essere diminuita aumentando la proporzione dei due colori immediatamente adiacenti oppure diminuendo la percentuale dei colori adiacenti al complementare. Se per esempio c'è troppo magenta in un'immagine allora può essere diminuito o rimuovendo del rosso e del blu oppure aggiungendo del verde.

Considerate l'immagine foto.jpg, che presenta un'elevata quantità di ciano, e, dopo aver effettuato la conversione nello spazio CMY, realizzate un bilanciamento delle componenti di colore cercando di riottenere l'immagine originale (memorizzata in foto_originale.tif). Se lavorate sulle componenti normalizzate tenete presente che per $\gamma > 1$ diminuite il peso del colore nell'immagine, il contrario accade per $\gamma < 1$.

2.5 Filtraggio spaziale

Supponiamo di considerare il filtro che realizza la media artitmetica dei valori che appartengono alla maschera, per cui se $\mathbf{x}(m,n)$ è l'immagine 3D nello spazio dei colori RGB, il risultato dell'elaborazione sarà:

$$\mathbf{y}(m,n) = \frac{1}{K} \sum_{m,n \in Mask} \mathbf{x}(m,n)$$

con K numero di pixel appartenenti alla maschera. D'altra parte è facile riconoscere che ogni componente dello spazio RGB ottenuta in uscita non è altro che la media aritmetica dell'elaborazione realizzata su ogni singola componente, per cui quando si effettua il filtraggio spaziale operare sull'immagine 3D o singolarmente su ogni componente è perfettamente equivalente. Per realizzare lo smoothing nello spazio RGB di un'immagine a colori memorizzata nella variabile `x` bisogna realizzare i seguenti passi:

```
R = x(:,:,1); G = x(:,:,2); B = x(:,:,3);
fR = imfilter(R,h);
fG = imfilter(G,h);
fB = imfilter(B,h);
y = cat(3,fR,fG,fB);
```

In realtà la funzione `imfilter` può operare direttamente nello spazio 3D, quindi il seguente comando fornisce direttamente il risultato: `y = imfilter(x,h)`. Se si volesse realizzare lo smoothing nello spazio HSI:

```
w = rgb2hsi(x);
H = w(:,:,1); S = w(:,:,2); I = w(:,:,3);
fI = imfilter(I,h,'replicate');
w = cat(3,H,S,fI);
y = hsi2rgb(w);
```

Si consideri allora l'immagine lenac.jpg, utilizzate un filtro di smoothing di dimensioni 5×5 operando su ogni singola componente RGB, provate poi a realizzare questa stessa operazione nello spazio HSI solo sulla componente di intensità. Confrontate le immagini ottenute sia visivamente che mostrando a video la differenza. Noterete che le due immagini non sono perfettamente identiche, ciò è dovuto al fatto che nello spazio RGB ogni pixel è pari al color medio dei pixel nella finestra 5×5, mentre mediare solo le intensità non altera il colore originale dei pixel (dato che tinta e saturazione non sono stati modificati). Questo effetto aumenta al crescere della dimensione del filtro. Ripetete l'esperimento usando l'immagine fiori.tif con una finestra di dimensioni 25×25.

Infine provate a realizzare un esperimento in cui effettuate l'enhancement di un'immagine a colori mediante un filtro di sharpening. Applicate il filtro laplaciano all'immagine fiori.jpg, lavorate sia in RGB che in HSI, e confrontate i risultati.

2.6 Esercizi proposti

1. *Variazione del colore.* Considerate l'immagine Azzurro.jpg, dove è presente un accappatoio azzurro, e generate l'immagine Rosso.jpg, in cui solo l'accappatoio diventa rosso. A tal fine, passate nello spazio di colore HSI, individuate la regione dell'immagine occupata dall'accappatoio in base ai suoi valori di tinta, saturazione e luminanza, e solo in tale regione operate una opportuna variazione della sola tinta.

2. *Equalizzazione.* L'immagine a colori X contenuta in Tiffany.tiff è chiaramente sovraesposta (oltre a presentare delle strisce errate sui bordi). Dopo aver sostituito valori ragionevoli al posto delle strisce errate, cercate di rendere l'immagine più gradevole seguendo due diversi approcci:

   (a) egualizzate singolarmente le tre componenti di colore;

   (b) convertite l'immagine in uno spazio di colore opportuno, egualizzate la sola componente di luminanza e antitrasformate.

   Confrontate le immagini risultanti con i due approcci.

3. *Filtraggio in frequenza.* Data l'immagine foto_originale.jpg, si vuole realizzare il filtraggio dell'immagine nel dominio della frequenza mediante il seguente filtro:

$$H(\mu,\nu) = \begin{cases} 1 & |\mu| \leq 0.10\,,\ |\nu| \leq 0.25, \\ 0 & \text{altrimenti} \end{cases}$$

   Scrivete il codice relativo e mostrate l'immagine filtrata.

4. *Demosaicking.* La maggior parte delle fotocamere digitali acquisisce una sola informazione di colore (rosso, verde, oppure blu) per ogni pixel, secondo il pattern mostrato nell'immagine Mosaic.bmp, e produce quindi, per ogni canale di colore, una matrice in cui solo alcuni dei valori sono non nulli. Gli altri valori sono poi ottenuti mediante interpolazione bilineare di quelli disponibili.

   L'immagine Fiori_mosaic.bmp contiene appunto, nelle tre componenti, le matrici prima dell'interpolazione. Scrivete una funzione `function y = demosa(x)` in cui estraete le tre componenti, effettuate l'interpolazione, ricomponete l'immagine e confrontate il risultato, sia visivamente che in termini di SNR, con l'immagine vera Fiori256.bmp.