Elaborazione di Segnali Multimediali
a.a. 2017/2018

Rappresentazione delle immagini

L.Verdoliva

In questa prima lezione vedremo come si rappresentano diversi tipi immagini digitali in Matlab. Oltre ai comandi base di Matlab, useremo le funzioni disponibili per l'elaborazione di immagini contenute nell'Image Processing Toolbox.

1 Immagini su scala di grigio

Un'immagine digitale (monocromatica) può essere rappresentata in Matlab come una matrice bidimensionale il cui generico campione $x(m,n)$ (*pixel*) varia nel range $[0, K-1]$. Le immagini sono quindi rappresentate con K livelli di grigio, e ogni pixel è codificato con $\log_2 K$ bit. Scriviamo i comandi Matlab per leggere da file l'immagine su scala di grigi dorian.jpg e che ci consentono di memorizzarne i valori in una matrice:

```
x = imread('dorian.jpg');       % leggiamo il file
[M,N] = size(x);                % memorizziamo le dimensioni in M e N
whos;                           % cosa abbiamo adesso in memoria?
```

Attraverso il comando `imread` è possibile leggere immagini con formato standard come JPEG, TIFF, GIF, BMP ed altri. Con il comando `help imread` potete verificare esattamente quali sono i formati supportati dalla versione di Matlab che state utilizzando.

Se l'immagine invece è in formato *grezzo* (cioè il file non ha un header, per cui l'estensione può essere qualsiasi) è necessario conoscerne le dimensioni e il formato. Per esempio, se l'immagine è 512×512 e il tipo di dato è intero senza segno su 8 bit, la lettura avviene con i seguenti comandi:

```
fid = fopen('house.y','r');       % apertura del file in lettura
x = fread(fid,[512 512],'uint8'); % lettura dei dati
x = x';                           % immagine trasposta
fclose(fid);                      % rilascio del file
whos x;                           % quali sono le caratteristiche di x?
```

Matlab legge per colonne quindi l'immagine va trasposta per una corretta visualizzazione. Fate attenzione al fatto che le immagini che volete leggere devono trovarsi nella directory corrente o nel path; in alternativa potete dare il percorso completo. Notate (usando `whos`) come il comando `imread` memorizzi i valori come interi senza segno su 8 bit, mentre `fread` li converta in double (64 bit). Quando si usa `imread` conviene comunque effettuare una conversione in double, perché molte operazioni in Matlab non sono definite sul tipo uint8. La conversione si ottiene facilmente col comando `x=double(x)`.

Per quanto riguarda invece la visualizzazione si può usare il seguente comando:

```
figure(1);          % apre la figura 1
imshow(x);          % visualizza l'argomento come immagine
```

Se $x(m,n) = k$, il comando `imshow` disegna nel punto (m,n) un pixel col k-esimo colore disponibile. Se `x` è di tipo *double*, 0 viene visualizzato come nero, 1 come bianco e tutti i valori intermedi come grigi su 256 possibili valori (di default). Se, invece, `x` è *uint8* (*uint16*), 255 (65535) è visualizzato come bianco. Se si vuole utilizzare un diverso range di visualizzazione rispetto a quelli di default appena descritti, è possibile specificare la dinamica dei dati nel seguente modo:

```
imshow(x,[low high]);    % visualizzazione nel range [low high]
imshow(x,[]);            % visualizzazione nel range [min(x) max(x)]
```

Se non si forniscono indicazioni sul valore minimo e massimo, allora i dati vengono scalati settando low e high al minimo e massimo valore presenti. Provate a visualizzare sia l'immagine dorian.jpg che granelli.jpg utilizzando le diverse opzioni del comando `imshow`. Come mai la visualizzazione varia così tanto solo per la seconda immagine?
Se l'immagine ha dimensioni molto grandi e se la si vuole analizzare con dettaglio avendo una serie di informazioni sui valori dei pixel si può usare il comando `imtool(x)` oppure `imtool('dorian.jpg')`; ovviamente in quest'ultimo caso l'immagine non sarà memorizzata nel workspace di matlab. Noterete come si sono aperte due finestre:

- *Overview window*. Mostra l'immagine all'interno di un rettangolo, le cui dimensioni possono essere modificate per stabilire quali parti dell'immagine si vogliono visualizzare nella finestra principale;
- *Image Information window*. Permette di avere molte informazioni sull'immagine: facendo scorrere il mouse sull'immagine si conosce la posizione e il valore di ogni pixel, inoltre è possibile esaminare i valori dei pixel in specifiche regioni cliccando il secondo pulsante della finestra.

Esplorate i diversi pulsanti disponibili e cercate di comprendere cosa fanno. Tenete presente che `imtool` mostra l'immagine al 100 %, cioè ad ogni pixel dell'immagine corrisponde un pixel dello schermo. Ricordate che per chiudere le immagini aperte con il comando `imtool` è necessario digitare `imtool close all` e non semplicemente `close all`.

In Matlab dopo aver elaborato un'immagine e' possibile salvarla in un file con un ben determinato formato con il comando `imwrite`. Nell'ipotesi in cui i valori da scrivere su file siano memorizzati nella variabile `x`:

```
x = uint8(x);            % tipo di dato supportato da JPEG
imwrite(x,'immagine.jpg','JPEG','Quality',70);
```

In questo modo si sta anche effettuando automaticamente una compressione JPEG con fattore di qualità 70 (tale parametro varia da 0 (qualità peggiore) a 100 (qualità migliore)). Provate a salvare un'immagine di prova a diversi fattori di qualità e visualizzatela.

Se però l'immagine che volete salvare è double e non volete perdere informazione la cosa migliore è salvare i dati con `fwrite` senza un header, nel seguente modo:

```
fid = fopen('immagine.y','wb'); % apertura del file in scrittura
fwrite(fid,x,'double');         % scrittura dei dati
fclose(fid);                    % rilascio del file
```

Attenti al fatto che se volete avere coerenza nella lettura dei dati, in quest'ultimo caso va effettuata un'operazione di trasposizione prima di salvare l'immagine. Un altro modo di salvare i dati in Matlab è quello di usare il comando `save`, molto utile qualora si vogliano per esempio salvare tutte le variabili che si trovano nel workspace. Ovviamente se volete salvarne solo alcune di variabili, basta elencarle dopo il nome del file:

```
save dati.mat           % salva tutte le variabili del workspace
save dati.mat x y       % salva le variabili x e y
```

Il comando `load` permette di caricare nuovamente i dati nel workspace.

2 Immagini a colori

Un'immagine a colori in formato standard (JPG, BMP, TIF ...) può facilmente essere importata in Matlab con il comando `imread`. In questo modo si crea una variabile che è una matrice di dimensioni $M \times N \times 3$, le cui componenti coincidono con le componenti di rosso, verde e blu dell'immagine (spazio RGB). Le tre immagini possono essere visualizzate singolarmente su scala di grigi e definiscono le quantità di rosso, verde e blu da mescolare per ottenere il pixel a colori sullo schermo.

```
x = imread('fragole.jpg'); figure; imshow(x);
R = x(:,:,1);
G = x(:,:,2);
B = x(:,:,3);
figure; imshow(R); title('componente di rosso');
figure; imshow(G); title('componente di verde');
figure; imshow(B); title('componente di blu');
```

Il comando `imshow` visualizza un'immagine a colori, operando in modo diverso se i dati sono di tipo uint8 oppure double. Nel primo caso i valori della variabile devono essere compresi tra 0 e 255 (il che si verifica ad esempio usando `imread`). Nel secondo caso, invece, devono essere normalizzati tra 0 e 1.

Estraendo poi le tre immagini da `x` è possibile visualizzare le componenti RGB (Red, Green, Blue): i valori più luminosi si ottengono in corrispondenza delle componenti di colore che sono presenti maggiormente nell'immagine. Facciamo un esperimento annullando la prima componente (R) e poi ricostruiamo l'immagine:

```
M = size(x,1);          % numero di righe di x
N = size(x,2);          % numero di colonne di x
R = zeros(M,N);         % annullamento della componente di rosso
y = cat(3,R,G,B);
figure; imshow(y);
```

Provate ad annullare singolarmente anche le altre due componenti e visualizzate l'immagine ricostruita.

Nel caso in cui si voglia costruire una struttura multidimensionale costituita da diverse immagini a colori è possibile farlo nel seguente modo:

```
z = cat(4,x1,x2,x3,x4,x5);          % creazione array multiframe
figure(1); imshow(z(:,:,:,3));      % mostra la terza immagine a video
figure(2);                          % mostra due immagini affiancate
subplot(1,2,1); imshow(z(:,:,:,1));
subplot(1,2,2); imshow(z(:,:,:,2));
figure(3); montage(z);              % mostra tutte le immagini
```

3 Immagini MRI

Proviamo adesso a visualizzare immagini ottenute tramite Risonanza Magnetica presenti nel database di Matlab:

```
load mri                  % carica le immagini MRI
figure; montage(D);       % visualizzazione
```

La variabile `D` comprende 27 slice orizzontali di dimensione 128×128 di una scansione MRI del cervello. Noterete come la visualizzazione è piuttosto scura. Il problema è che la dinamica dei dati è compresa in un range piuttosto piccolo ($[0, 88]$). Per migliorare la visualizzazione e aumentare il contrasto, basta allora specificare il massimo e il minimo dei valori compresi nel range:

```
figure; montage(D,[0 88]);       % visualizzazione
```

Attenzione: i comandi di visualizzazione non modificano il valore dei dati memorizzati nella matrice.

4 Immagini HDR

Un'immagine HDR (High Dynamic Range) presenta un intervallo tra le aree visibili più chiare e quelle più scure molto ampio. In Matlab esiste il comando `hdrread` per la loro lettura:

```
x = hdrread('office.hdr');
imshow(x);
```

Potrete notare che l'immagine risulta molto scura, ciò è dovuto al fatto che il monitor del PC presenta un limitato range per la visualizzazione che risulta essere inadatto a riprodurre un'immagine HDR. E' necessario pertanto effettuare il *tone mapping* che consente di ridurre la dinamica in modo opportuno:

```
y = tonemap(x);
imshow(y);
```

4.1 Esercizi proposti

1. *Lettura e visualizzazione di un'immagine*. Provate a scrivere una funzione in grado di visualizzare le immagini sia in formato JPEG che in formato grezzo. Il prototipo delle funzioni deve essere rispettivamente: `function x=vediJPG(nomefile)` e `function x=vediRAW(nomefile,nRighe,nColonne,tipo)`.

2. *Rappresentazione in falsi colori.* La rappresentazione in pseudocolori (o falsi colori) consiste nel visualizzare a colori un'immagine monocromatica. L'obiettivo è quello di migliorare l'interpretazione di un'immagine su livelli di grigio o semplicemente visualizzarla più facilmente. Infatti, l'occhio umano è in grado di discriminare migliaia di variazioni di colore rispetto a qualche decina di livelli di grigio.

   Un esempio di rappresentazione in falsi colori riguarda le immagini multispettrali. Queste sono un insieme di immagini, ognuna delle quali è stata rilevata dal satellite in una diversa banda spettrale. Considerate le 4 immagini telerilevate di Washington, in cui è presente una regione con il fiume Potomac. Le prime tre immagini sono state rilevate nelle tre bande del visibile, mentre la quarta è nel vicino infrarosso. Provate a visualizzare le prime tre immagini come immagine a colori e poi sostituite alla componente R, la quarta immagine, quindi confrontate le due figure. Noterete come nella seconda immagine il fiume (vegetazione) è più facilmente discriminabile dalla città. Infatti il vicino infrarosso è molto sensibile alle parti di una scena contenenti biomasse, e l'immagine mostra effettivamente in maniera chiara le differenze tra le componenti naturali (in rosso) e le costruzioni, composte principalmemte da asfalto e cemento (tendenti al blu).