Elaborazione delle Immagini

Laboratorio 3

Obiettivi:

- Applicare filtri non lineari
- Elaborazione di immagini a colori
- Filtraggio di immagini a colori
- Filtri di edge

Ricordate: per processare le immagini è sempre conveniente trasformale in valori double tra 0 e 1 con **im2double**.

Ricordate: imshow visualizza le immagini in modo corretto se hanno valori tra 0 e 255 (uchar8), se hanno valori tra 0 e 1 (double) o sono valori logici.

Ricordate: se volete saperne di più sulle funzioni Matlab usate, consultate l'help o la documentazione con i seguenti comandi da console:
help <funzione>
doc <funzione>

Scrivete il codice di ogni esercizio in uno script separato (labX_1.m, labX_2.m, ...)

---

a1. Caricate l'immagine '**lawrence2.png**' in una variabile **im.**

b1. Create una immagine **imF** filtrando **im** utilizzando il comando **medfilt2**. Il comando applica un filtro mediano sull'immagine. E' necessario specificare la dimensione del filtro. Usate un 3x3.

c1. In una stessa finestra visualizzate **im** e **imF**

d1. Filtrate l'immagine **im** cambiando la dimensione del filtro e visualizzate i risultati. Cosa notate?

---

(2)

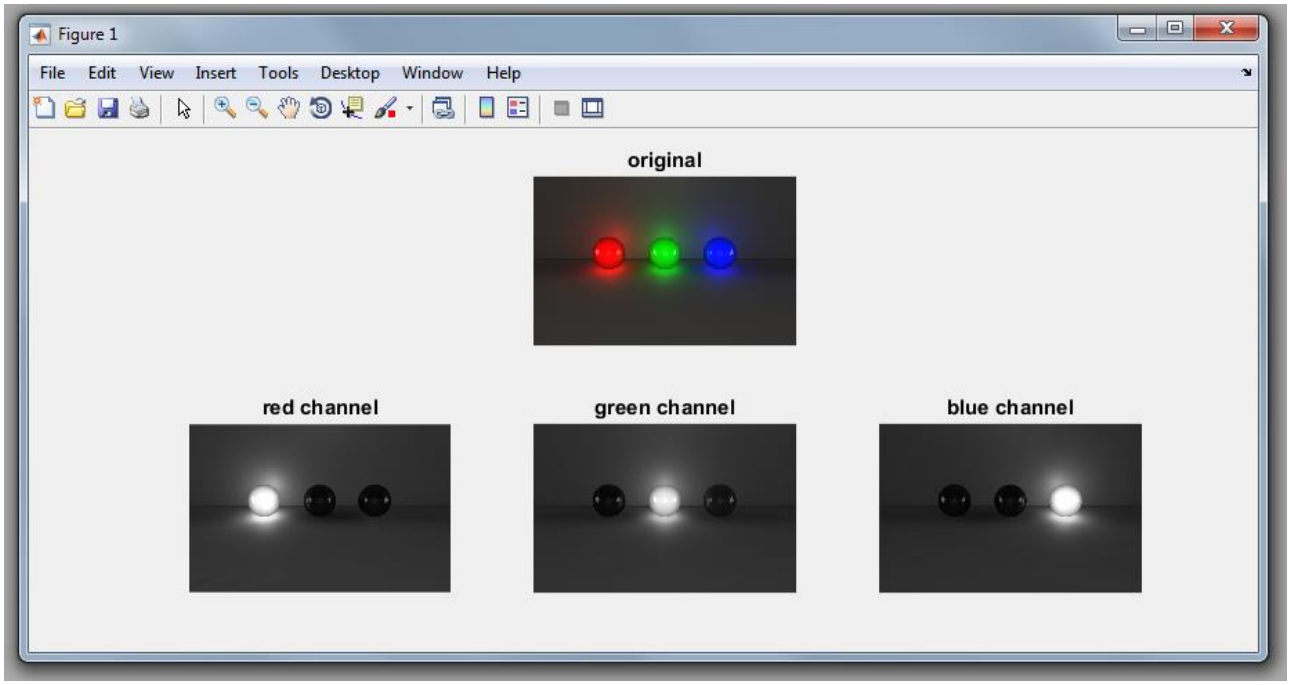
a2. Caricate l'immagine '**balls.jpg**' in una variabile **balls**.

b2. L'immagine è in RGB. Estraete i tre canali colore in tre variabili **R**, **G** e **B**.

c2. Visualizzate in un figura di 2x3 celle, l'immagine originale nella cella di mezzo della prima riga e nelle tre celle della seconda riga le tre immagini dei canali colore. Cosa notate?

d2. Caricate l'immagine '**wedding.jpg**' in una variabile **wedding.**

e2. Ripetete il procedimento del punto c2. Cosa notate? Osservate attentamente il motivo del tovagliolo.



a3. Caricate l'immagine **building.jpg'** in una variabile **building.**

b3. Visualizzate in un figura di 1x4 celle, l'immagine originale (building) e le immagini dei canali colore R, G e B.

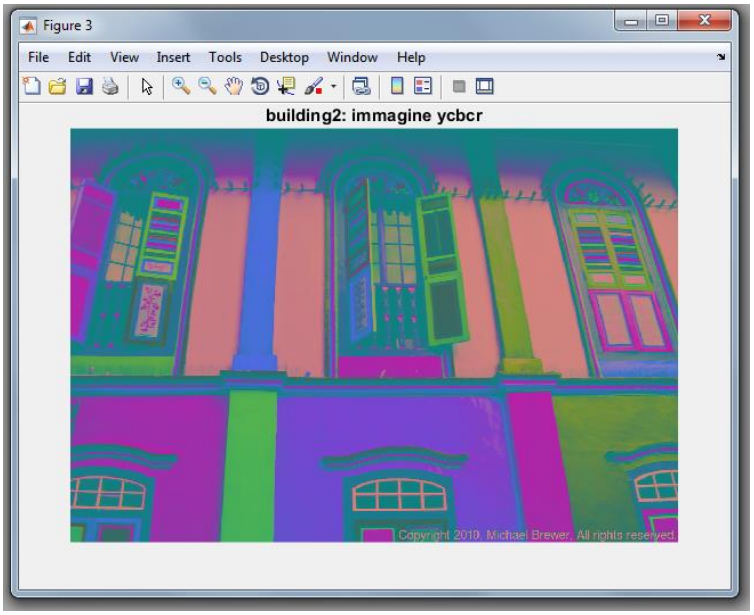
c3. Utilizzate il comando **rgb2ycbcr** per trasformare l'immagine dallo spazio color RGB a quello YCbCr. Chiamate l'immagine trasformata **building2.**

d3. Visualizzate in un figura di 1x4 celle, l'immagine originale (building) e le tre immagini dei canali colore Y, Cb e Cr. Studiate le caratteristiche delle immagini dei canali colore.

e3. Visualizzate l'immagine **building2.** Cosa notate? Perchè secondo voi i colori sono così?

f3. Utilizzate il comando **rgb2hsv** per trasformare l'immagine dallo spazio color RGB a quello HSV. Chiamate l'immagine trasformata **building3.** .

g3. Visualizzate in un figura di 1x4 celle, l'immagine originale (building) e le tre immagini dei canali colore H, S e V. Studiate le caratteristiche delle immagini dei canali colore.



La trasformazione dallo spazio colore RGB a quello YCbCr è una trasformazione lineare. Se l'immagine di input ha range di valori tra 0 e 255, la trasformazione è la seguente:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad \text{(EQ)}$$

Il primo termine della sommatoria (offset) serve per portare i valori di output in un range positivo. Se l'immagine di input ha valori tra 0 e 1, l'offset è moltiplicato per 1/255.

(4)

a4. Caricate l'immagine **face-noisy.png**' in una variabile **face**.

b4. Separate i tre canali colore in tre variabili **R**, **G** e **B**.

c4. Create un filtro Gaussiano **GF** di dimensione 7x7 con la Sigma adatta.

d4. Applicando il filtro Gaussiano (con **imfilter**) ai tre canali colore create le tre immagini **R2**, **G2** e **B2**.

e4. Usando le immagini **R2**, **G2** e **B2** come piani colore, ricreate una immagine RGB chiamata **face2**

e4. Visualizzate le immagini **face** e **face2**

f4. Create una immagine **face3** filtrando direttamente l'immagine **face** con il filtro gaussiano e **imfilter**

g4. Confrontate i risultati di face2 e face3. Cosa notate?

Le operazioni su immagini a colori solitamente vengono eseguite applicando le operazioni sui singoli canali colore che sono visti come immagini a livelli di grigio. I risultati poi sono combinati nell'immagine di output.

---

a5. Caricate l'immagine '**balloons_noisy.png**' in **balloons**.

b5. Visualizzate l'immagine caricata.

b5. Filtrate l'immagine con un filtro mediano 5x5.
Il filtro mediano si applica solo a immagini a livelli di grigio, quindi applicate il filtro ai tre canali colore singolarmente.

c5. Visualizzate l'immagine risultato

d5. Provate a filtrare con il filtro mediano solo uno dei piani colore e ricomponete l'immagine RGB con il piano colore filtrato.

e5. Visualizzate i risultati delle vostre prove.

---

(6)

a6. Caricate in una variabile **im** l'immagine '**lanterns.png**'.

b6. Visualizzate l'immagine caricata.

c6. In due variabili **im1 e im2**, trasformate l'immagine **im** nello spazio colore ycbcr.

d6. Filtrate il canale y di **im1** con un filtro gaussiano di dimensione 33x33 e sigma 6.

e6. Filtrate il canale cr di **im2** con un filtro gaussiano di dimensione 33x33 e sigma 6.

f6. In una variabile **out1** ricreate una immagine RGB da **im1** con il comando **ycbcr2rgb**

g6. In una variabile **out2** ricreate una immagine RGB da **im2** con il comando **ycbcr2rgb**

h6. Visualizzate le immagini **out1 e ou2.** Cosa notate?

(7)

a7. Caricate in una variabile **im** l'immagine '**mondrian.jpg**'.

b7. Convertite l'immagine a vaolori tra 0 e 1 con **im2double** e poi a livelli di grigio con **rgb2gray** e mettete il risultato in una variabile **gray**.

c7. Usate la funzione **edge** per trovare i bordi delle regioni presenti nell'immagine. Testate gli algoritmi 'prewitt', 'sobel', 'roberts' con il valore di soglia automatico.

d7. Visualizzate e confrontate i risultati dei tre diversi algoritmi. Cosa notate?

e7. Provate a settare delle soglie manualmente e analizzate i risultati.

f7. Ripetete l'esperimento con l'immagine '**signs.jpg**'

La funzione **edge** ha diversi modi per trovare i bordi. I metodi più semplici usano dei filtri lineari di edge (prewitt, sobel, roberts). Questi filtri, per ogni pixel, calcolano la magnitudine del gradiente 2D che è un indicatore della "forza" degli edge. I valori poi sono sogliati per identificare i soli pixel corrispondenti a edge "forti". Di default, **edge** determinata automaticamente la soglia. E' possibile però specificarla.

I filtri lineari usati internamente per calcolare le derivate parziali (e quindi i gradienti) sono questi:

Prewitt: $Gy = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$ $Gx = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$

Sobel: $Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$ $Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$

Roberts: $Gx = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix}$ $Gy = \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}$

Il calcolo della magnitudine del gradiente può avere diverse formulazioni:

$$|\nabla| = \sqrt{\left(Gx^2 + Gy^2\right)}$$

$$|\nabla| = \frac{|Gx| + |Gy|}{2}$$

Compiti a casa - Lab3

Scrivete gli script o le funzioni Matlab richieste e consegnatele in un file zip tramite l'apposito link sul sito del corso. NON potete usare le rispettive funzioni Matlab!

Scrivete una funzione **myrgb2gray** per trasformare una immagine RGB in una immagine a livelli di grigio

function out_gray = myrgb2gray(image_rgb)

…

end

Scrivete una funzione **myedge** per determinare i pixel di edge usando uno dei tre metodi: 'prewitt', 'sobel' o 'roberts'. La funzione ha in input l'immagine e una soglia. L'immagine può essere a livelli di grigio o a colori RGB e quindi dovete gestire i due casi. L'output è una immagine binaria (logical) che contiene un 1 se il pixel è di edge, 0 altrimenti.

function out_edges = myedge(image, threshold)

…

end

Scrivete una funzione **myRGB2YCbCr** per trasformare una immagine RGB in una immagine nello spazio YCbCr.

function out_ycbcr = myRGB2YCbCr(image_rgb)

…

end

SUGGERIMENTO: l'idea è quella di applicare la formula dell'esercizio 3 (EQ) a TUTTI i pixel dell'immagine di input contemporaneamente. Separate la trasformazione in due step: prodotto matriciale e somma dell'offset.

Ricordate le nozioni di algebra lineare.

Ricordate che Matlab esegue nativamente operazioni matriciali.

Dovete assicurarvi di moltiplicare matrici trasformate nel modo corretto.

L'input e l'output della funzione sono array 3D RxCx3. Internamente potete usare array nelle forme che vi sono più utili…

Studiate le funzioni **reshape** e **repmat**.