Elaborazione delle Immagini

Laboratorio 2

Obiettivi:

- Applicare la gamma correction "adattativa"
- Filtro di Media
- Filtro Gaussiano
- Applicare i filtri lineari

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, ...)

---

(1)

a1. Caricate l'immagine '**contrast.jpg**' in una variabile **im** e scalataela tra 0 e 1.

b1. Create un ritaglio (**crop**) dell'immagine prendendo solo la parte inferiore più scura corrispondente alla regione (top,left)-(bottom,right)=(360,1)-(419,505)

c1. Visualizzate (con **imhist**) l'istogramma dell'immagine **crop** e determinate una soglia **T** che identifichi i valori dei pixel scuri.

d1. Usando la soglia **T** e la tecnica delle maschere create due maschere **light** e **dark** che contengono rispettivamente le regioni chiare e scure di **im**.

e1. Separatamente, applicate una gamma correction ai soli pixel di **im** corrispondenti alla maschera **light** e a quelli corrispondenti alla maschera **dark**. Sarà necessario usare due valori di gamma diversi.

f1. Dovete ottenete una immagine **out** complessivamente meglio contrastata

(2)

a2. Caricate l'immagine **cat.jpg**' in una variabile **cat0** e scalatela tra 0 e 1.

b2. Usando la funzione **fspecial,** create un filtro di media (average) di dimensione 5x5. Mettete il filtro in una variablile **F5.** Stampate a schermo il contenuto del filtro. Perchè il filtro ha questi valori?

c2. Usando la funzione **imfilter**, eseguite la convoluzione dell'immagine **cat0** con il filtro **F5**. Mettete il risultato in una variabile **cat5.**

d2. Ripetete il filtraggio creando dei filtri a dimensione 11x11 (**F11**) e 21x21 (**F21**) creando le immagini filtrate **cat11** e **cat21.**

e2. Create tre immagini **d5**, **d11** e **d21** per contenere il valore assoluto della differenza tra **cat0** e **cat5**, **cat11** e **cat21** rispettivamente. Usate **abs** per il valore assoluto.

f2. Visualizzate le immagini **cat0**, **cat5**, **cat11** e **cat21** e confrontatele. Che cosa notate?

g2. Visualizzate le immagini differenza **d5**, **d11** e **d21** con **imagesc**. Includete i comandi "axis image, colorbar" quando usate imagesc. Cosa notate?

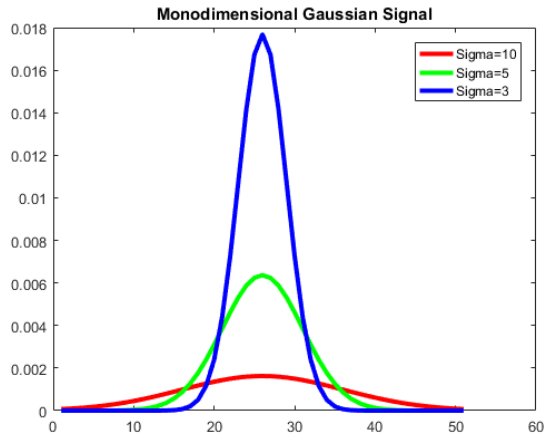
---

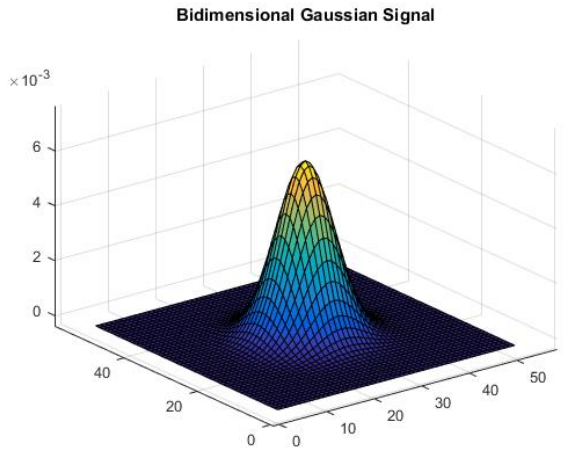
a3. Usando la funzione **fspecial,** analizzate nella command window il contenuto di un filtro di smooth Gaussiano (gaussian) di dimensione 3x3 e con Sigma=0.5. Potete anche visualizzare il filtro con **imagesc**.

b3. Nello stesso modo e senza cambiare Sigma, analizzate i filtri Gaussiani di dimensione 5x5, 7x7, 9x9. Cosa notate? Ha senso avere un filtro di dimensione maggiore di 5x5? Perchè?

c3. Mantenendo la dimensione del filtro Gaussiano 5x5, cambiate il valore di Sigma in 1, 2, 3 e 10 e analizzate i filtri ottenuti. Cosa notate?

d3. Provate ad applicate un Filtro Gaussiano di dimensione 5x5 sulla immagine '**cat.jpg**'.





I filtri spaziali sono creati andando a campionare i segnali mono o bidimensionali ad intervalli regolari.

Un filtro Gaussiano ha due parametri, la dimensione del filtro e la Sigma da usare. Come avete sperimentato, questi due parametri non sono indipendenti ma sono legati tra loro. Di solito, si la dimensione di un filtro Gaussiano si ricava dalla seguente formula che garantisce di avere tutti i valori utili del filtro:

$$N = 1 + 2 \times \lfloor 2.5 \times sigma \rfloor$$

A differenza di un filtro di media, il filtro gaussiano pesa di più i valori dei pixel vicini al centro del filtro. Questo permette di avere un filtro che, a parità di dimensioni, è meno "forte" di un filtro di media. La possibilità di variare la sigma inoltre permette un maggiore controllo sul risultato finale. Sigma molto basse danno un filtro molto concentrato (al limite si ha un filtro identità). Sigma molto alte danno un filtro piatto (al limite di media).

a4. Caricate l'immagine '**glamor.jpg**' in una variabile **glamor** e convertitela in valori tra 0 e 1.

b4. Create un filtro Gaussiano **G** di dimensione 21x21 con la Sigma adatta.

c4. Filtrate **glamor** per ottenere l'immagine **filtered.**

d4. Create una immagine **out** ottenuta mediando la somma tra **glamor** e **filtered**.

e4. Visualizzate le immagini **glamor, filtered** e **out**.

Quello che è stato creato è un procedimento per ottenere delle immagine con effetto 'flou'.

---

(5)

a5. Caricate l'immagine '**lawrence1.jpg**' in una variabile **im1**.

b5. Caricate l'immagine '**lawrence2.jpg**' in una variabile **im2**.

c5. Visualizzate le immagini in due finestre distinte e, utilizzando gli strumenti della finestra, ingrandite una regione dell'immagine osservando come è fatto il segnale di rumore. Cosa notate?

d5. Provate ad applicare dei filtri di media o Gaussiani sulle due immagini per tentare di ridurre il rumore presente. Quale filtro funziona meglio su quale immagine? Perchè?

Diverse tipologie di rumore, richiedono diverse tecniche di rimozione e filtraggio.

Come abbiamo visto nell'esercizio 4, i filtri lineari si possono usare anche per ottenere degli "effetti speciali" sulle immagini. Provate a dedurre cosa fanno i seguenti filtri se applicati ad una immagine:

$$F1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad F2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \qquad F3 = \begin{bmatrix} 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 \end{bmatrix}$$

a6. Verificate se la vostra intuizione è giusta creando i filtri sopra indicati e applicandoli all'immagine '**running.png**'.

a7. A partire dalle istruzioni dell'esercizio 4, pensate a come poter creare un singolo filtro lineare che permette di ottenere l'effetto flou.

Guardate la soluzione riportata sotto solo dopo averci provato

Suggerimenti:

partite dalla formula finale usata per ottenere out
dovete vedere il risultato finale come una combinazione di convoluzioni tra immagini e filtri
quale è il filtro equivalente a non fare nulla?
si possono combinare i filtri lineari se hanno la stessa dimensione
considerate che l'operazione di convoluzione è una operazione lineare
usate un pò di fantasia

Identifichiamo una convoluzione con il simbolo ⊗.
out = (glamor + filtered) * 0.5
out = (glamor + glamor ⊗ G) * 0.5
out = (glamor ⊗ I + glamor ⊗ G) * 0.5
I è il filtro identità (ha solo un 1 nel punto centrale) delle stesse dimensioni di G
out = glamor ⊗ (I + G) * 0.5
Il filtro richiesto è F = (I+G)*0.5
out = glamor ⊗ F

Compiti a casa - Lab2

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 voi una funzione **myresize** che data una immagine in input e un fattore di scala, ritorna l'immagine ridimensionata usando il metodo nearest neighbor:

```
function out_image = myresize(image, scale)
...
end
```

Utilizzate le nozioni sulle trasformazioni geometriche che avete visto a lezione. Suggerimento: dovete determinare, per ogni coordinata dei pixel dell'immagine di output, le coordinate del pixel dell'immagine di input da cui prelevare il valore. Arrotondate le coordinate all'intero più vicino.

Scrivete voi una funzione **myfiltering** che calcola la convoluzione tra una immagine e un filtro:

```
function out_filtered = myfiltering(image,filter)
...
end
```

L'immagine di output deve avere la stessa dimensione di quella di input. Potete ignorare il calcolo della convoluzione quando il filtro "esce" dall'immagine. In questi casi il risultato nell'immagine di output sarà zero. Usate tutti i cicli for che sono necessari.

Scrivete voi una funzione **mymaxfilter** che data una immagine e la dimensione di un intorno (dispari), ritorna, per ogni pixel, il valore massimo contenuto nell'intorno di tale pixel:

```
function out_filtered = mymaxfilter(image,neighborhood)
...
end
```

L'immagine di output deve avere la stessa dimensione di quella di input.