Elaborazione di Segnali Multimediali

Rappresentazione delle immagini
Soluzioni

1 Immagini su scala di grigio e a colori

1. *Lettura e visualizzazione di un'immagine JPEG.*

```
import numpy as np               # importa Numpy
import matplotlib.pyplot as plt  # importa Matplotlib
import skimage.io as io          # importa il modulo Input/Output di SK-Image
from os.path import isfile       # importa la funzione "isfile" da "os.path"

def vediJPG(nomefile):
    """
    carica un'immagine memorizzata in formato JPEG e la visualizza.
    x = vediJPG(nomefile) carica l'immagine contenuta nel file 'nomefile'
    """

    if not isfile(nomefile):
        print('Il file %s non esiste!' % nomefile);
    else:
        x = io.imread(nomefile)
        x = np.float32(x)
        plt.figure(); plt.imshow(x, clim=(0,255), cmap='gray')
        plt.show()
        return x
```

Notate che in ogni script e modulo Python c'è sempre un prima sezione per importare i moduli che verranno utilizzati. Per leggibilità di questo documento ometteremo questa sezione nelle soluzioni di seguito riportate.

2. *Lettura e visualizzazione di un'immagine grezza*.

```
def vediRAW(nomefile,M,N,dtype):
    """
    carica un'immagine in formato grezzo e la visualizza.
    x = vediRAW(nomefile,M,N,dtype) carica l'immagine
    di dimensioni MxN contenuta in 'nomefile';
    tipo e' il formato di dato (np.uint8, np.float32, ecc.)
    """

    if not isfile(nomefile):
        print('Il file %s non esiste!' % nomefile)
    else:
        x = np.fromfile(nomefile, dtype)
        x = np.reshape(x, (M, N)) # reshape permette di ottenere la matrice
        x = np.float32(x)
        plt.figure(); plt.imshow(x, clim=(0,255), cmap='gray')
        plt.show()
        return x
```

Se si inseriscono commenti con le tre virgolette subito dopo la definizione della funzione, essi compariranno nell'help in linea del Python. Importate la funzione `vediRAW` in console e provate per esempio a digitare da linea di comando `help(vediRAW)`, vedrete che verranno visualizzate le prime righe di commento presenti nella funzione `def vediRAW`. E' molto importante commentare bene ogni funzione riportando un esempio di chiamata della funzione.

Una volta importata la funzione basterà digitare i comandi:

```
x = vediRAW('lena.y',512,512,np.uint8)
```

per caricare in `x` e visualizzare l'immagine.

3. *Rappresentazione in falsi colori*.

```
R = io.imread('Washington_red.tif')
G = io.imread('Washington_green.tif')
B = io.imread('Washington_blue.tif')
I = io.imread('Washington_infrared.tif')

x = np.stack((R,G,B), 2)
plt.figure(1); plt.imshow(x); plt.show();

y = np.stack((I,G,B), 2)
plt.figure(2); plt.imshow(y); plt.show();
```

2 Caratteristiche delle immagini

1. Utilizziamo il codice già visto che calcola le medie locali per scrivere la funzione `medie(x,K)`:

```
def medie(x, K):
    """
    calcola l'immagine delle medie
    """

    M = x.shape[0]
    N = x.shape[1]
    MED = np.zeros((M-K+1,N-K+1))
    for i in range(M-K+1):
        for j in range(N-K+1):
            MED[i,j] = np.mean(x[i:i+K,j:j+K])
    return MED
```

Questo codice non prevede di gestire i bordi, infatti produce un'immagine delle medie più piccola di quella originale. Per gestire il problema dei bordi è necessario estendere l'immagine di partenza mediante riempimento con zeri (*zero padding*) o estensione periodica o simmetrica (in base alle esigenze dell'applicazione). Per realizzare questa operazione si può usare il comando `np.pad`:

```
xext = np.pad(x, ((h,h), (h,h)), constant_values=0)
```

In questo caso è stata aggiunta una cornice esterna di h zeri all'immagine.

Usando le opzioni `mode='reflect'` e `mode='wrap'` è possibile effettuare un'estensione simmetrica e periodica, rispettivamente. Modifichiamo allora il codice usando l'estensione ai bordi:

```
def medie(x, K):
    """
    calcola l'immagine delle medie
    """

    M = x.shape[0]
    N = x.shape[1]
    MED = np.zeros((M,N))
    h = (K-1) // 2
    xext = np.pad(x, ((h,h), (h,h)), constant_values=0)
    for i in range(M):
        for j in range(N):
            MED[i,j] = np.mean(xext[i:i+K,j:j+K])
    return MED
```

Tenete presente che potete anche usare la funzione `ndi.generic_filter` che prevede il parametro `mode` per scegliere il tipo di estensione ai bordi. Di default la funzione `ndi.generic_filter` effettua

un'estensione simmetrica ai bordi (`mode='reflect'`). A questo punto è semplice scrivere la funzione per il calcolo delle medie:

```
import scipy.ndimage as ndi

def medie(x, K):
    """
    calcola l'immagine delle medie
    """

    MED = ndi.generic_filter(x, np.mean, (K,K), mode='constant')
    return MED
```

2. Per il calcolo dell'immagine delle varianze su blocchi $K \times K$, le possibili soluzioni sono le seguenti:

```
def varianze(x, K):
    """
    calcola l'immagine delle varianze
    """

    M = x.shape[0]
    N = x.shape[1]
    VAR = np.zeros((M-K+1,N-K+1))
    for i in range(M-K+1):
        for j in range(N-K+1):
            VAR[i,j] = np.var(x[i:i+K,j:j+K])
    return VAR

def varianze_v2(x, K):
    """
    calcola l'immagine delle varianze
    """

    return ndi.generic_filter(x, np.var, (K,K))
```