Elaborazione di Segnali Multimediali

Elaborazioni nel dominio spaziale
Soluzioni

1 Operazioni puntuali

1. *Negativo di un'immagine.*

```
x = np.float32(io.imread('mammografia.jpg'))
y = 255-x
plt.imshow(y, clim=[0,255], cmap='gray')
```

2. *Full-Scale Histogram Stretch.*

```
def fshs(x):
    """
    effettua il Full-scale Histogram Stretch
    """

    m = np.min(x)
    M = np.max(x)
    y = 255*(x - m)/(M - m)
    return y
```

3. *Equalizzazione.*

```
x = io.imread('granelli.jpg')
plt.figure(1); plt.imshow(x, clim=[0,255], cmap='gray');
plt.figure(2); plt.hist(x.flatten(), 256);


from skimage.exposure import equalize_hist
y = equalize_hist(x) # output e' nel range [0,1]
y = 255*y            # lo convertiamo nel range [0, 255]

plt.figure(3); plt.imshow(y, clim=[0,255], cmap='gray');
plt.figure(4); plt.hist(y.flatten(), 256);
```

4. *Uso delle statistiche locali*.

```
x = io.imread('filamento.jpg')
MED = ndi.generic_filter(x, np.mean, (3,3))
DEV = ndi.generic_filter(x, np.std , (3,3))
med = np.mean(x)
dev = np.std(x)
mask = (MED<=0.4*med) & (DEV<=0.4*dev) & (DEV>=0.02*dev)
y = x+3*x*mask

plt.figure(3); plt.imshow(y, clim=[0,255], cmap='gray');
```

2 Bit-plane slicing

(a) per generare tutti i bit-plane si può scrivere il seguente codice:

```
from bitop import bitget

M, N = x.shape
bitplane = np.zeros((M,N,8), dtype=np.bool)  # matrice 3D
for i in range(8):
    bitplane[:,:,i] = bitget(x, i)
    plt.figure()
    plt.subplot(2,4,i+1)
plt.imshow(bitplane[:,:,i], clim=[0,1], cmap='gray')
    plt.title('Bitplane %d' % i)
```

(b) *Ricostruzione mediante bit-plane*. Proviamo adesso ad annullare i bit-plane meno significativi e poi a ricostruire l'immagine:

```
from bitop import bitset

n = 3 # Numero di bit-plane da annullare
y = np.copy(x)
for i in range(n):
    y = bitset(y, i, 0)

plt.figure()
plt.imshow(y, clim=[0,255],cmap='gray')
plt.title('Rappresentazione con i primi %d bitplane' % (8-n))
```

(c) *Esempio di Watermarking*. Inseriamo una firma digitale nel bit-plane meno significativo nell'ipotesi in cui x sia l'immagine originale e f la firma:

```
from bitop import bitset
y = bitset(x, 0, f)
plt.figure(); plt.imshow(y, clim=[0,255], cmap='gray');
% plt.title('Immagine marcata')
```

3 Operazioni geometriche

3.1 Rotazione

```
def ruota(x, theta):
    M,N = x.shape

    A1 = np.array([[1,0,N/2],[0,1,M/2],[0,0,1]], dtype=np.float32)
    A2 = np.array([[np.cos(theta),np.sin(theta),0],
     [-np.sin(theta),np.cos(theta),0],
     [0,0,1]], dtype=np.float32)
    A3 = np.array([[1,0,-N/2],[0,1,-M/2],[0,0,1]], dtype=np.float32)

    A = A1 @ A2 @ A3
    y = warp(x, A, order = 1)
    return y
```

3.2 Combinazione di operazioni geometriche

```
def rot_shear(x, theta, c):
    M,N = x.shape

    T1 = np.array([[1,0,0],[0,1,0],[M/2,N/2,1]], dtype=np.float32)
    T2 = np.array([[np.cos(theta),np.sin(theta),0],
     [-np.sin(theta),np.cos(theta),0], [0,0,1]], dtype=np.float32)
    T3 = np.array([[1,0,0],[c,1,0],[0,0,1]], dtype=np.float32)
    T4 = np.array([[1,0,0],[0,1,0],[-M/2,-N/2,1]], dtype=np.float32)

    T = T4 @ T3 @ T2 @ T1
    A = T[[1,0,2],:][:,[1,0,2]].T
    y = warp(x, A, order = 1)
    return y
```