

**Prova del dd-Mmm-yyyy (90 minuti)**

**Cognome** \_\_\_\_\_ **Nome:** \_\_\_\_\_

[illegible][illegible][illegible]

4) Utilizzando NumPy e OpenCV, implementare in Python la funzione *esercizio(img)* che riceve un'immagine grayscale *img* (con un byte per pixel) e deve eseguire le seguenti operazioni:

- 1) Calcolare, per ciascuna riga dell'immagine, la somma dei valori dei pixel: sia  $Y_m$  la coordinata  $y$  della riga dell'immagine con la somma minore.
- 2) Binarizzare *img*, utilizzando come unica soglia la media dei livelli di grigio dei pixel con coordinata  $y$  minore o uguale a  $Y_m$ .
- 3) Applicare, al risultato del passo precedente, un'operazione morfologica di dilatazione con un cerchio di diametro 3 pixel come elemento strutturante: sia *img3* il risultato.
- 4) Costruire un'immagine contenente solo i bordi (con uno spessore di 2 pixel) delle componenti connesse dell'immagine ottenuta al punto precedente. Suggerimento: questo risultato può essere ottenuto con la differenza fra l'immagine e il risultato dell'erosione con un cerchio di diametro  $2*2+1$  pixel.
- 5) Determinare tutti i pixel di background di *img3* con distanza maggiore di 4 pixel (secondo la metrica  $d_8$ ) dal foreground.
- 6) Restituire un'immagine a colori in formato BGR in cui i pixel di bordo individuati al passo 4 sono blu, i pixel individuati al punto 5 verdi e i restanti pixel hanno un valore, nel solo canale R, pari alla metà (arrotondata all'intero inferiore) della corrispondente luminosità in *img*.

```
import numpy as np
```

```
import cv2 as cv
```

```
def esercizio(img):
```

*Promemoria: alcune funzioni che potrebbero essere utili*

`cv.threshold(src, thresh, maxval, type) -> retval, dst`

`cv.morphologyEx(src, op, kernel) -> dst`

`cv.getStructuringElement(shape, ksize) -> retval`

`cv.distanceTransform(src, distanceType, maskSize) -> dst`

*Promemoria: alcune costanti che potrebbero essere utili*

`cv.THRESH_BINARY`

`cv.MORPH_ELLIPSE`

`cv.MORPH_ERODE, cv.MORPH_DILATE`

`cv.DIST_C`