Univerza v Ljubljani, Fakulteta za elektrotehniko **REŠITVE**
Modul A: Slikovna informatika (64135) - Laboratorijske vaje

Vaja 8: Morfološka obdelava slik

Podrobna navodila za reševanje

S spletne učilnice naložimo sliki real-256x256-08bit.raw in test-128x256-08bit.raw, ter ju shranimo v novo mapo `Vaja08`. Zaženimo razvojno okolje Spyder, delovno mapo nastavimo na ustvarjeno mapo `Vaja08` in ponovno zaženemo konzolo IPython. V mapi ustvarimo tudi novo `.py` datoteko `vaja08.py`.

Kot je že v navadi, bomo pri tej nalogi uporabljali funkciji `displayImage()` in `loadImage()`, ki ju lahko skopiramo v novo datoteko `vaja08.py` iz Vaje 7: Prostorsko filtriranje slik. V testni blok `if __name__ == '__main__'` zapišemo ukaz za nalaganje in prikaz slike test-128x256-08bit.raw. Do tega koraka naj bo struktura datoteke `vaja08.py` sledeča:

```
import numpy as np
import matplotlib.pyplot as pp

def displayImage(iImage, iTitle, gridX=None, gridY=None):

  pp.figure()
  pp.title(iTitle)
  if gridX is not None and gridY is not None:
    stepX = gridX[1] - gridX[0]
    stepY = gridY[1] - gridY[0]
    extent = (gridX[0] - 0.5*stepX, gridX[-1] + 0.5*stepX,
      gridY[-1] + 0.5*stepY, gridY[0] - 0.5*stepY)
    pp.imshow(iImage, cmap=pp.cm.gray, vmin=0, vmax=255, extent=extent)
  else:
    pp.imshow(iImage, cmap=pp.cm.gray, vmin=0, vmax=255)
  pp.show()

def loadImage(iPath, iSize, iType):

  fid = open(iPath, 'rb')

  oImage = np.ndarray(
    (iSize[1],iSize[0]),
    dtype=iType,
    buffer=fid.read()
    )

  fid.close()

  return oImage

if __name__ == '__main__':

  # naloži in prikaži sliko
  I = loadImage('test-128x256-08bit.raw', [128, 256], np.uint8)
  displayImage(I, 'Originalna slika')
```

Naloga 1

V tej nalogi moramo zapisati funkcijo za morfološko erozijo slike, ki jo podaja sledeča enačba:

$$g(x,y) = (f \ominus b)(x,y) = \min_{\forall i,j} \Big\{ f(x-i, y-j) \cdot b(i,j) \Big\} \,, \tag{1}$$

kjer je $f(x,y)$ sivinska vrednost vhodne slike na položaju (x,y) in $b(i,j)$ strukturni element, ki določa pravilo s katerim priredimo vrednost izbranega slikovnega elementa z vrednostmi slikovnih elementov v njegovi okolici. Strukturni element $b(i,j)$ je sestavljen iz binarnih vrednosti, kjer vrednost 1 pomeni, da slikovni element vključimo v operacijo, 0 pa da vrednosti ne vključimo. Morfološka erozija je operacija, ki vrednosti izbranega slikovnega elementa $f(x,y)$ priredi najmanjšo vrednost v okolici izbranega slikovnega elementa, ki jo določa strukturni element $b(i,j)$.

Funkcija bo torej sprejela vhodno sliko `iImage` in morfološki element `iStruct`:

```
def morphErosion(iImage, iStruct):

  '''
  Erozija slike s strukturnim elementom.
  '''

  # inicializacija izhodne slike in dimenzije
  oImage = np.zeros_like(iImage)

  # dimenzije strukturnega elementa
  m = int((iStruct.shape[1]-1)/2)
  n = int((iStruct.shape[0]-1)/2)

  # priredi strukturni element glede na definicijo
  iStruct = np.rot90(iStruct, 2)

  # razširitev slikovne domene (max=255)
  iImage_padded = np.pad(iImage, ((n,n), (m,m)),
                         mode='constant', constant_values=255)

  # erozija dane slike (min)
  for y in range(iImage.shape[0]):
    for x in range(iImage.shape[1]):

      iArea = iImage_padded[y:y+2*n+1, x:x+2*m+1] * iStruct

      oImage[y, x] = np.min(iArea[iStruct != 0])

  return oImage
```

V funkciji sprva definiramo izhodno sliko oziroma večdimenzionalno podatkovno polje `oImage`, ki vsebuje same ničle in je iste dimenzije kot vhodna slika `iImage`. Sledi definicija celih števil m in n, ki sta povezani z velikostjo pravokotnega strukturnega elementa $M \times N = (2m+1) \times (2n+1)$ (pozor, dimenzije strukturnega elementa podajamo podobno kot dimenzije slike $X \times Y$). Celi števili m in n nam podata mejne vrednosti i in j, torej $i = -m \ldots m$ v smeri x in $j = -n \ldots n$ v smeri y, znotraj katerih je definirana okolica izbranega slikovnega elementa na mestu (x,y). Ker bomo z m in n indeksirali podatkovna polja, uporabimo funkcijo `int()` za pretvorbo v celoštevilski podatkovni tip.

Sledi priredba strukturnega elementa `iStruct`, da bo ustrezal definiciji enačbe (1). Podobno, kot smo to ugotovili že pri konvoluciji pri Vaji 8: Prostorsko filtriranje slike, se bomo z naraščanjem i in j po slikovnih elementih vhodne slike $f(x-i, y-j)$ sprehajali v nasprotni smeri kot po strukturnem elementu $b(i,j)$. Enačbo 1 pa lahko z ustrezno substitucijo $i \rightarrow -i$ in $j \rightarrow -j$ preoblikujemo v:

$$g(x,y) = (f \ominus b)(x,y) = \underset{\forall i,j}{min}\Big\{ f(x+i, y+j) \cdot b(-i,-j) \Big\} \,, \tag{2}$$

kjer sedaj področje vhodne slike $f(x+i, y+j)$ množimo s strukturnim elementom $b(-i,-j)$. Sledi torej, da moramo strukturni element horizontalno in vertikalno prezrcaliti, da ga lahko neposredno množimo z istoležnimi elementi področja vhodne slike. Dvakratno zrcaljenje dosežemo z rotacijo strukturnega elementa za 180°, pri čemer uporabimo funkcijo np.rot90(), kjer prvi vhodni parameter ustreza strukturnemu elementu `iStruct`, drugi vhodni parameter pa številu rotacij za 90°, torej 2.

Podobno kot že pri prostorskem filtriranju slik z nekim jedrom filtra, se nam v primeru morfološke obdelave slik lahko zgodi, da na robu slike za neke vrednosti i in j dobimo indekse, s katerimi naslavljamo vhodno sliko $f(x,y)$ izven njene domene. Temu se izognemo tako, da razširimo domeno vhodne slike $f(x,y)$ oziroma `iImage` z okvirjem debeline n v smeri y (vrstice) in debeline m v smeri x (stolpci). To storimo z uporabo funkcije `np.pad()`, ki smo jo že spoznali pri Vaji 7: Prostorsko filtriranje slik. Funkcija sprejme sliko oziroma podatkovno polje `iImage`, ki ga želimo okviriti, seznam `tuple`, kjer za vsako stranico posebej definiramo debelino okvirja `((rob_y_zgoraj,rob_y_spodaj),(rob_x_levo,rob_x_desno))`, način izvedbe okvirjanja `mode='constant'` in parameter `constant_values=255`. Za debelino zgornjega in spodnjega dela okvirja smo izbrali n, saj slednji določa razpon delovanja strukturnega elementa v smeri y, torej v smeri vrstic. Za debelino levega in desnega dela okvirja smo izbrali m, saj slednji določa razpon delovanja strukturnega elementa v smeri x, torej v smeri stolpcev. Za način izvedbe okvirja smo izbrali konstantne vrednosti `'constant'`, ki znašajo `constant_values=255`. S tem smo domeno vhodne slike `iImage` razširili z največjo vrednostjo dinamičnega območja sivinskih vrednosti, tako erozija na robovih vrne ustrezne vrednosti in slikovnih elementov razširjenega okvirja ne upošteva (saj pri eroziji iščemo najmanjše vrednosti)!

Sledi sprehod po celotni sliki z dvema zankama `for`, pri čemer definiramo razpon v smeri y s številom vrstic `iImage.shape[0]` in v smeri x s številom stolpcev `iImage.shape[1]`, ki ju kot vhodna parametra podamo v funkcijo za ustvarjanje zaporednih števil `range()`.

V jedru obeh zank z ukazom `iImage_padded[y:y+2*n+1, x:x+2*m+1]` naslovimo področje vhodne slike z razširjeno domeno `iImage_padded`. Središče tega področja je slikovni element vhodne slike `iImage` na mestu (x,y). Dobljeno področje nato pomnožimo z istoležnimi elementi strukturnega elementa `iStruct` ter rezultat shranimo v spremenljivko `iArea`. Na novem področju `iArea` s funkcijo `np.min()` sedaj poiščemo najmanjšo vrednost slikovnega elementa, pri čemer pa upoštevamo le tiste, kjer je strukturni element različen od nič. To storimo z logičnim naslavljanjem `iArea[iStruct != 0]`. Rezultate operacije morfološke erozije nato shranimo na mestu (x,y) izhodne slike `oImage`. Po izvedbi obeh zank izhodno sliko tudi vrnemo.

Naloga 2

V tej nalogi moramo zapisati funkcijo za morfološko dilacijo slike, ki jo podaja sledeča enačba:

$$g(x,y) = (f \oplus b)(x,y) = \underset{\forall i,j}{max}\Big\{ f(x-i, y-j) \cdot b(i,j) \Big\} \,,$$

kjer je $f(x,y)$ sivinska vrednost vhodne slike na položaju (x,y) in $b(i,j)$ strukturni element. Morfološka dilacija je operacija, ki vrednosti izbranega slikovnega elementa $f(x,y)$ priredi največjo vrednost v okolici izbranega slikovnega elementa, ki jo določa strukturni element $b(i,j)$.

Podobno kot v nalogi 1 bo torej funkcija sprejela vhodno sliko `iImage` in morfološki element `iStruct`:

```
def morphDilation(iImage, iStruct):

  '''
  Dilacija slike s strukturnim elementom
  '''

  # inicializacija izhodne slike in dimenzije
  oImage = np.zeros_like(iImage)

  # dimenzije strukturnega elementa
  m = int((iStruct.shape[1]-1)/2)
  n = int((iStruct.shape[0]-1)/2)

  # priredi strukturni element glede na definicijo
  iStruct = np.rot90(iStruct, 2)

  # razširitev slikovne domene (min=0)
  iImage_padded = np.pad(iImage, ((n, n), (m, m)),
                         mode='constant', constant_values=0)

  # dilacija dane slike (max)
  for y in range(iImage.shape[0]):
    for x in range(iImage.shape[1]):

      iArea = iImage_padded[y:y+2*n+1, x:x+2*m+1] * iStruct

      oImage[y, x] = np.max(iArea[iStruct != 0])

  return oImage
```

Funkcija morfološke dilacije morphDilation() se od funkcije morfološke erozije morphErosion(), ki smo jo že napisali, razlikuje le na dveh mestih.

Prva razlika je razširitev slikovne domene, kjer pri morfološki dilaciji uporabimo `constant_values=0`. S tem domeno vhodne sliko `iImage` razširimo z najmanjšo vrednostjo dinamičnega območja sivinskih vrednosti, tako dilacija na robovih vrne ustrezne vrednosti in slikovnih elementov rezširjenega okvirja ne upošteva (pri dilaciji iščemo maksimalne vrednosti)!

Druga razlika je v jedru obeh zank `for`, kjer na področju `iArea` s funkcijo `np.max()` sedaj poiščemo maksimalno vrednost slikovnega elementa, pri čemer pa upoštevamo le tiste, kjer je strukturni element različen od nič (zato logični izraz `iStruct != 0`).

Naloga 3

V tej nalogi bomo napisali funkcijo za morfološko odpiranje slike, ki je zaporedje operacij erozije in nato dilacije na sliki $f(x, y)$:

$$g(x, y) = (f \circ b)(x, y) = \Big((f \ominus b)(x, y) \oplus b\Big)(x, y) \quad \text{oz.} \quad f \circ b = (f \ominus b) \oplus b\,.$$

Na vhodni sliki `iImage` moramo torej s funkcijo `morphErosion()` izvesti morfološko erozijo slike, nato pa na rezultatu še s funkcijo `morphDilation()` izvesti morfološko dilacijo slike. Strukturni element je v obeh primerih isti, funkcija `morphOpening()` pa izgleda takole:

```
def morphOpening(iImage, iStruct):

  '''
  Odpiranje slike s strukturnim elementom.
  '''

  # odpiranje = dilacija erozije
  oImage = morphDilation(morphErosion(iImage, iStruct), iStruct)

  return oImage
```

Naloga 4

Ostane nam še zapis funkcije `morphClosing()` za morfološko zapiranje slike, ki je zaporedje operacij dilacije in nato erozije na sliki $f(x, y)$:

$$(f \bullet b)(x, y) = \Big((f \oplus b)(x, y) \ominus b\Big)(x, y) \quad \text{oz.} \quad f \bullet b = (f \oplus b) \ominus b\,.$$

Na vhodni sliki `iImage` moramo torej s funkcijo `morphDilation()` izvesti morfološko dilacijo slike, nato pa na rezultatu še s funkcijo `morphErosion()` izvesti morfološko erozijo slike. Strukturni element je v obeh primerih isti, funkcija `morphClosing()` pa izgleda takole:

```
def morphClosing(iImage, iStruct):

  '''
  Zapiranje slike s strukturnim elementom.
  '''

  # zapiranje = erozija dilacije
  oImage = morphErosion(morphDilation(iImage, iStruct), iStruct)

  return oImage
```

Naloga 5

Oglejmo si sedaj delovanje in rezultate operacij morfološke obdelave na primeru slike test-128x256-08bit.raw, pri čemer uporabimo sledeči strukturni element:

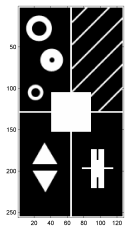
$$M \times N = 5 \times 5$$

0	0	1	0	0
0	1	1	1	0
1	1	1	1	1
0	1	1	1	0
0	0	1	0	0

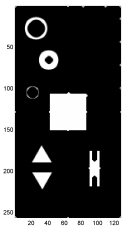
Ukaze za izvedbo morfološke erozije, dilacije, odpiranja in zapiranja slike lahko zapišemo v testni blok:

```
if __name__ == '__main__':
  # naloži in prikaži sliko
  I = loadImage('test-128x256-08bit.raw', [128, 256], np.uint8)
  # I = loadImage('real-256x256-08bit.raw', [256, 256], np.uint8)
  displayImage(I, 'Originalna slika')

  # strukturni element
  SE = np.array([[0, 0, 1, 0, 0],
                 [0, 1, 1, 1, 0],
                 [1, 1, 1, 1, 1],
                 [0, 1, 1, 1, 0],
                 [0, 0, 1, 0, 0]])
  # SE = np.ones((3,1))  # MxN=1x3  (stolpec)
  # SE = np.ones((1,3))  # MxN=3x1  (vrstica)
  # SE = np.ones((5,5))
  # SE = np.ones((3,3))
  # SE = ... (za občutek preizkusite več možnosti)

  # erozija in dilacija slike
  eI = morphErosion(I, SE)
  displayImage(eI, 'Erozija slike')
  dI = morphDilation(I, SE)
  displayImage(dI, 'Dilacija slike')

  # odpiranje (dilacija erozije) in zapiranje (erozija dilacije) slike
  oI = morphOpening(I, SE)
  displayImage(oI, 'Odpiranje slike')
  cI = morphClosing(I, SE)
  displayImage(cI, 'Zapiranje slike')
```

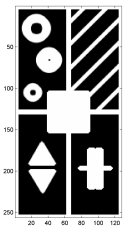
Delovanje preostalih strukturnih elementov na sliki test-128x256-08bit.raw ali real-256x256-08bit.raw lahko dosežemo z ustreznim od-komentiranjem ukazov v zgornjem bloku kode.



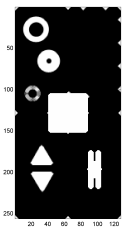
(a) Originalna slika



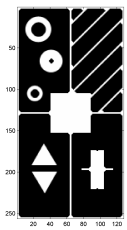
(b) Erozija



(c) Dilacija



(d) Odpiranje



(e) Zapiranje

Za izbran strukturni element velikosti 5×5, ki ga uporabimo v testnem bloku na sliki test-128x256-08bit.raw, so rezultati podani zgoraj.

Kot lahko vidimo, so učinki morfološke erozije (b) zmanjševanje struktur po velikosti ter povečevanje praznih prostorov znotraj struktur. V tem primeru smo vodoravne, navpične in poševne črte povsem izničili. Učinki morfološke dilacije (c) pa so povečevanje struktur po velikosti ter zmanjševanje praznih prostorov znotraj struktur. Tako smo na primer črno špranjo spodnjega desnega objekta povsem zapolnili.

Če se osredotočimo na morfološki obdelavi z odpiranjem (d) in zapiranjem (e), lahko vidimo, da ima odpiranje podoben učinek kot erozija, le da je manj destruktivno v smislu osnovne oblike objektov. Podobno ima zapiranje podoben učinek kot dilacija, le da je spet manj destruktivno za oblike osnovnih objektov.

Z ustrezno izbiro strukturnega elementa lahko dosežemo vrsto zanimivih učinkov. Na primer, z vrstičnim strukturnim elementom se lahko znebimo samo navpičnih struktur, medtem ko vodoravne obdržimo.