Laboratorijske vježbe iz

digitalne obrada i analiza slike

**Vježba 6:**

**Segmentacija**

**digitalne slike**

**Zadatak 1.**

U Pythonu učitajte sliku u razinama sive boje i napišite funkciju koja će je segmentirati na slijedeći način:

* klasa 1 - svi pikseli u rangu [0, 85]
* klasa 2 - svi pikseli u rangu [86, 171]
* klasa 3 - svi pikseli u rangu [172, 255]

**Originalna slika:**

****

**Segmentirana slika:**

****

**Kod:**

import cv2

slika=cv2.imread('../img/img\_greyscale.jpg')

retci=slika.shape[0]

stupci=slika.shape[1]

kanali=slika.shape[2]

for k in range(0, kanali):

    for i in range(0, retci):

        for j in range(0, stupci):

            if slika[i, j, k] >= 172 and slika[i, j, k] <= 225:

                slika[i, j, k] = 255

            elif slika[i, j, k] >= 86 and slika[i, j, k] <= 171:

                slika[i, j, k] = 150

            else:

                slika[i, j, k] = 0

cv2.imwrite('../img/img\_z1.jpg', slika)

**Zadatak 2.**

U Pythonu učitajte sliku u boji i na nju primijenite algoritam k-sredina (ne trebate ga sami implementirati, možete koristiti postojeću funkciju). Kako se segmentirana slika i brzina algoritma mijenjaju kada se mijenja k?

**Odgovor:**

Što je k veći, proces je sporiji ali i s razlogom jer uvodimo više segmenata, odnosno, što je k manji, manje je segmenata (cjelina) pa će proces bit brži. U konačnici o veličini k ovisi hoćemo li imat jednobojnu sliku, ili „višebojnu“ tj. hoće li slika sličiti izvornoj s detaljima tj. sa jako puno segmenata ili će imati manji broj segmenata pa s time sličiti više na binarnu sliku ili na jednobojnu cjelinu.

**Originalna slika:**

****

**Segmentirana slika:**

****

**Kod:**

import numpy as np

import matplotlib.pyplot as plt

import cv2

image = cv2.imread("../img/img.jpg")

pixel\_vals = image.reshape((-1, 3))

pixel\_vals = np.float32(pixel\_vals)

criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, 100, 0.85)

k = 5

retval, labels, centers = cv2.kmeans(pixel\_vals, k, None, criteria, 10, cv2.KMEANS\_RANDOM\_CENTERS)

centers = np.uint8(centers)

segmented\_data = centers[labels.flatten()]

segmented\_image = segmented\_data.reshape((image.shape))

cv2.imwrite('../img/img\_z2.jpg', segmented\_image)

**Zadatak 3.**

U Pythonu učitajte sliku u boji i na nju primijenite algoritam razvodnjavanja (ne trebate ga sami implementirati, možete koristiti postojeću funkciju). Usporedite dobivenu segmentiranu sliku sa rezultatima iz prethodnog zadatka.

**Odgovor:**

Ako je kod napravio što je trebao, jako je loše odredio segmente te mi se čini da je postupak iz 2. zadatka učinkovitiji.

**Originalna slika:**

****

**Segmentirana slika:**

****

**Kod:**

import numpy as np

import cv2 as cv

from matplotlib import pyplot as plt

img = cv.imread('../img/img.jpg')

gray = cv.cvtColor(img, cv.COLOR\_BGR2GRAY)

ret, thresh = cv.threshold(gray, 0, 255, cv.THRESH\_BINARY\_INV + cv.THRESH\_OTSU)

*# noise removal*

kernel = np.ones((3,3), np.uint8)

opening = cv.morphologyEx(thresh, cv.MORPH\_OPEN, kernel, *iterations* = 2)

*# sure background area*

sure\_bg = cv.dilate(opening, kernel, *iterations*=3)

*# Finding sure foreground area*

dist\_transform = cv.distanceTransform(opening, cv.DIST\_L2, 5)

ret, sure\_fg = cv.threshold(dist\_transform, 0.7 \* dist\_transform.max(), 255, 0)

*# Finding unknown region*

sure\_fg = np.uint8(sure\_fg)

unknown = cv.subtract(sure\_bg, sure\_fg)

*# Marker labelling*

ret, markers = cv.connectedComponents(sure\_fg)

*# Add one to all labels so that sure background is not 0, but 1*

markers = markers + 1

*# Now, mark the region of unknown with zero*

markers[unknown == 255] = 0

markers = cv.watershed(img, markers)

img[markers == -1] = [255, 0, 0]

cv.imwrite('../img/img\_z3.jpg', img)