Laboratorijske vježbe iz

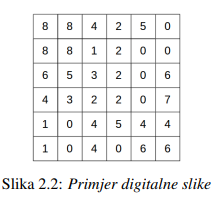
digitalne obrada i analiza slike

**Vježba 2:**

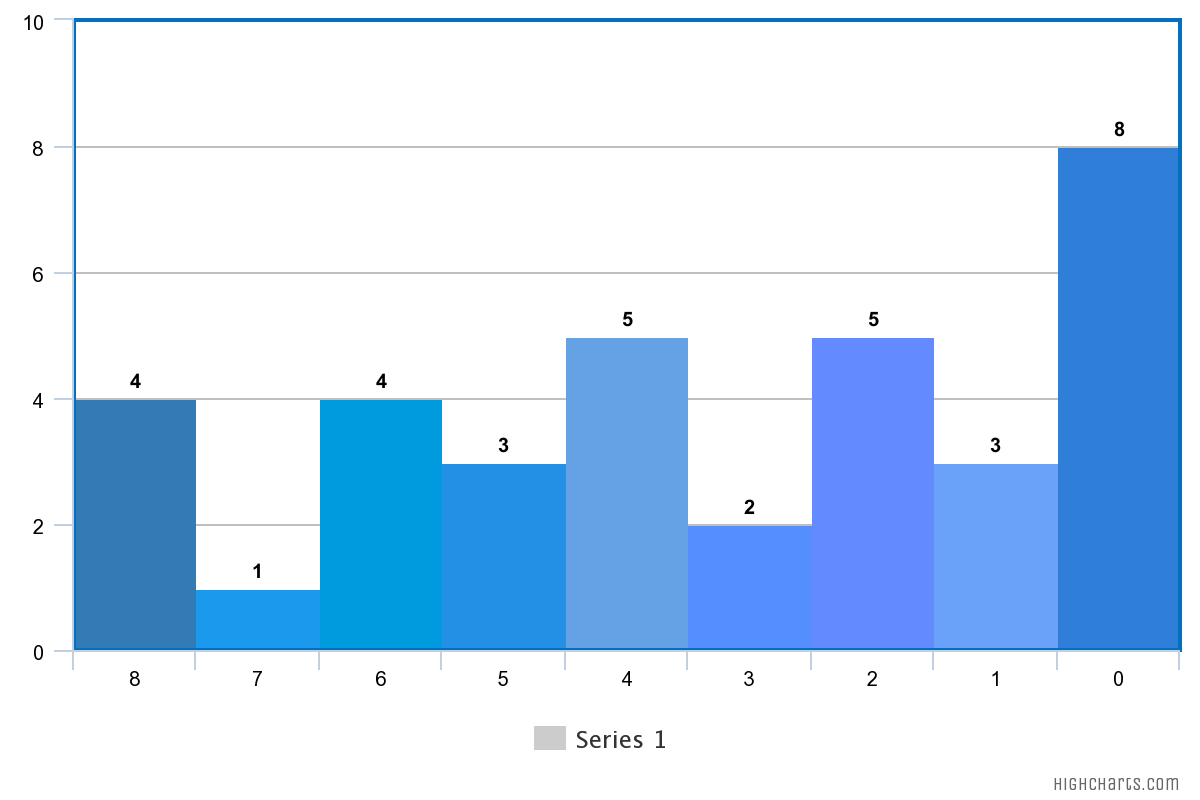
**Histogrami**

**Zadatak 1.**

Nacrtajte histogram za sliku prikazanu na slici 2.2. Histogram možete skicirati u proizvoljnom programu.

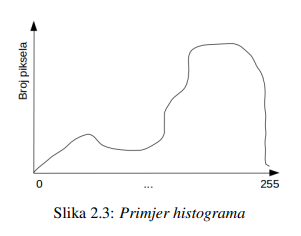


Histogram:



**Zadatak 2.**

Imajući na umu da 0 označava crnu boju a 255 bijelu boju, što možete zaključiti o slici čiji je histogram prikazan na slici 2.3?



Zaključak o slici bi bio da je većinski u svijetlim tonovima sive, tj. da je većinski svijetla (moguće je da joj je *brightness* povećan).

**Zadatak 3.**

U Pythonu učitajte sliku u boji i prikažite histogram njezinog zelenog kanala.

**Korištena slika:**



**Kod:**

import cv2

from matplotlib import pyplot as plt

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

histogram = cv2.calcHist([slika\_bgr], [1], None, [256], [0, 256])

plt.hist(slika\_bgr.ravel(), 256, [0,256])

plt.title('Histogram slike')

plt.xlabel('Intenziteti sive boje')

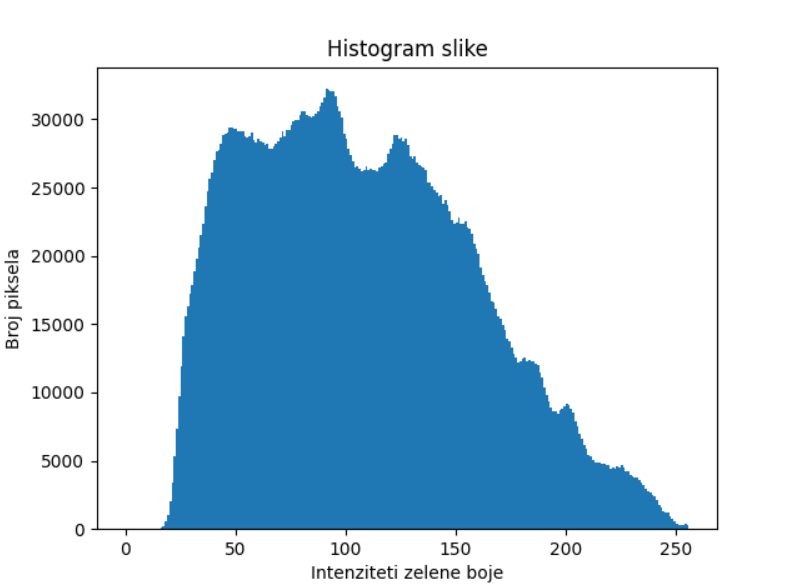
plt.ylabel('Broj piksela')

plt.show(*block* = False)

plt.pause(30)

plt.close()

**Histogram (zeleni kanal):**



**Zadatak 4.**

U Pythonu sami napišite funkciju za računanje histograma slike. Usporedite dobiveni rezultat s onim koji se dobije pomoću Pythonove ugrađene funkcije.

**Kod:**

import cv2

from matplotlib import pyplot as plt

img\_bgr = cv2.imread('../img/img.jpg')

row = img\_bgr.shape[0]

column = img\_bgr.shape[1]

channel = img\_bgr.shape[2]

histogram = []

for c in range(0, channel):

    channelValues = []

    for i in range(0, row):

        for j in range(0, column):

*#print(img\_bgr[i, j, c])*

            channelValues.append(img\_bgr[i, j, c]);

    histogram.append(channelValues)

plt.hist(histogram[0], *bins*=256)

plt.show()

U predzadnjoj liniji se poziva plt.hist() kojoj se prosljeđuje element niza histogram:, indeksi: 0 – plava, 1 – zelena, 2 – crvena.

**Histogrami (redom B, G, R) s gotovom Python funkcijom:**

|  |  |  |
| --- | --- | --- |
|  |  |  |

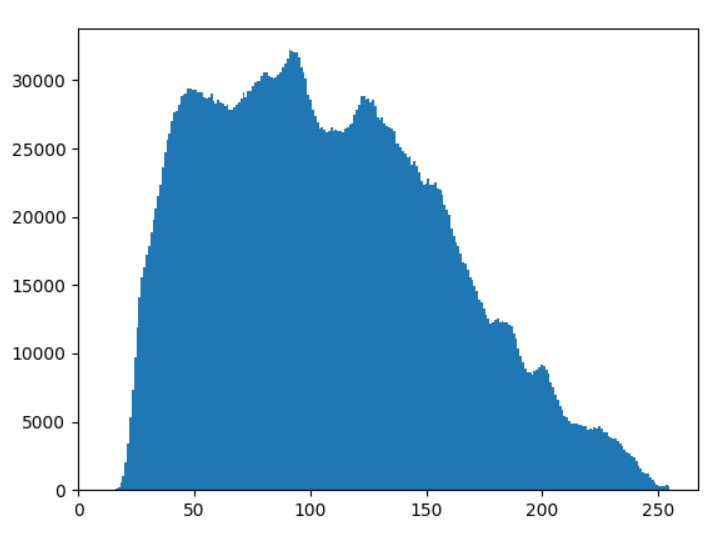
**Histogrami (redom B, G, R) s mojom funkcijom:**

|  |  |  |
| --- | --- | --- |
|  |  |  |

Primjetno je da se u gotovoj Python funkciji oni intenziteti koji se ne pojavljuju na slici odbacuju kako bi graf bio neprekidan, dok se kod moje funkcije vide praznine za vrijednosti intenziteta od 0 do 255 koje se nisu pojavile (primjer zeleni G kanal). Vidimo da je točno 12 prekida (tj. 12 vrijednosti intenziteta se ne pojavljuje), ako u kodu modificiramo liniju na sljedeći način:

plt.hist(histogram[0], *bins*=244)

vidimo da će graf biti bez tih prekida:



**Zadatak 5.**

U Pythonu učitajte sliku u razinama sive boje te na nju primijenite operaciju rastezanja histograma. Usporedite dobivenu sliku i njezin histogram s originalnom slikom i njenim histogramom. Što se dogodi s slikom ako promijenite cutoff vrijednost?

**Crno-bijela slika:**

|  |  |
| --- | --- |
| Prije rastezanja histograma | Poslije rastezanja histograma |
|  |  |

**Kod:**

import cv2

from matplotlib import pyplot as plt

from PIL import Image, ImageOps

img\_greyscale = Image.open("../img/img\_greyscale.jpg")

new\_img = ImageOps.autocontrast(img\_greyscale, *cutoff* = 0)

new\_img.save("../img/new\_img\_greyscale.jpg")

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

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

plt.hist(new\_img\_greyscale[:, :, 0].ravel(), 256, [0,256])

plt.show(*block* = False)

plt.pause(5)

plt.close()

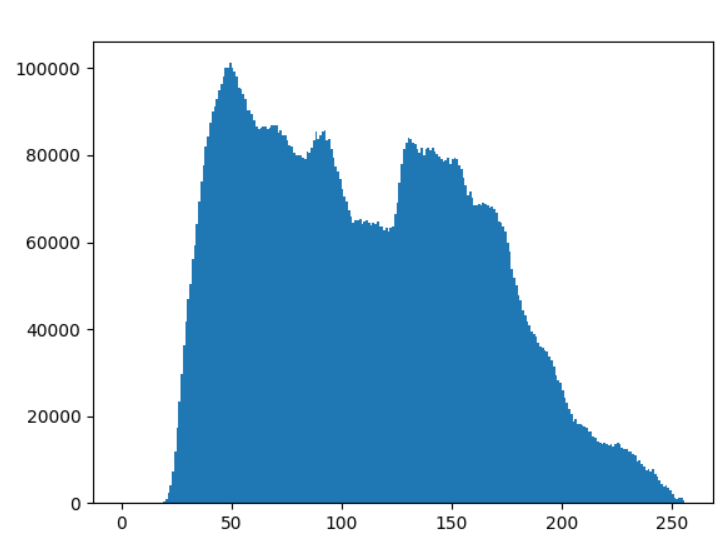
plt.hist(old\_img\_greyscale[:, :, 0].ravel(), 256, [0,256])

plt.show(*block* = False)

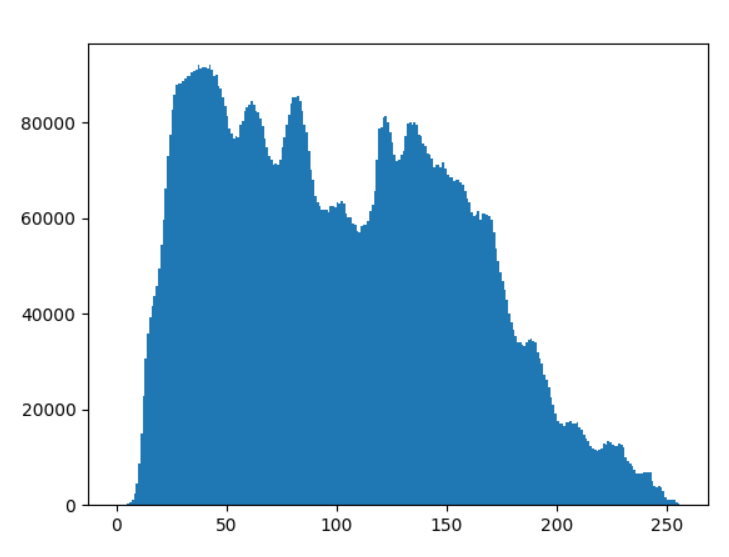
plt.pause(5)

plt.close()

**Histogram prije rastezanja:**



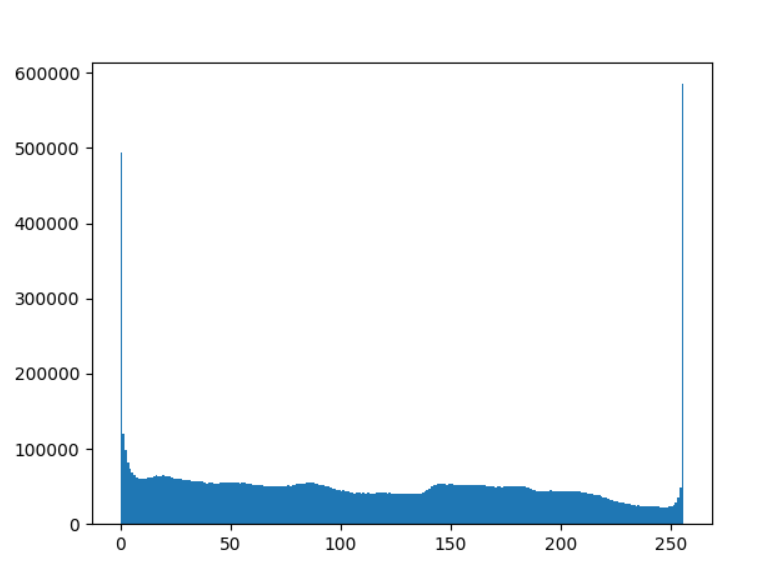
**Histogram poslije rastezanja:**



Vidimo da je novi histogram rastegnut s obzirom na x-os tako da ide od 0 do 255, za razliku od starog, ali je sada i valovitiji s obzirom na y-os.

Promjenom vrijednosti argumentu cutoff vidimo da se određeni postotak na histogramu reže, a samim time se kontrast pojačava.

Ako stavimo da je cutoff = 5, histogram je:



a slika:



**Zadatak 6.**

U Pythonu učitajte sliku u razinama sive boje te na nju primijenite operaciju ujednačavanja histograma. Usporedite dobivenu sliku i njezin histogram s originalnom slikom i njenim histogramom.

**Crno-bijela slika:**

|  |  |
| --- | --- |
| Prije izjednačavanja histograma | Poslije izjednačavanja histograma |
|  |  |

**Kod:**

import cv2

from matplotlib import pyplot as plt

from PIL import Image, ImageOps

img\_greyscale = Image.open("../img/img\_greyscale.jpg")

new\_img = ImageOps.equalize(img\_greyscale, *mask* = None)

new\_img.save("../img/new\_img\_greyscale.jpg")

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

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

plt.hist(new\_img\_greyscale.ravel(), 256, [0,256])

plt.show(*block* = False)

plt.pause(5)

plt.close()

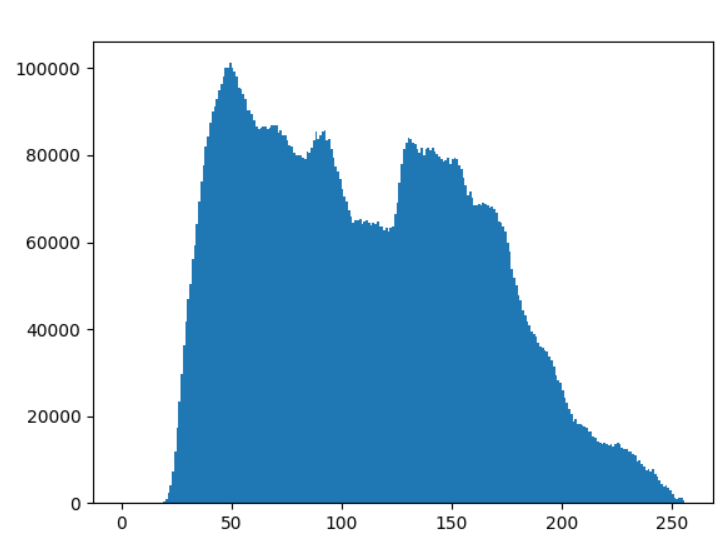
plt.hist(old\_img\_greyscale.ravel(), 256, [0,256])

plt.show(*block* = False)

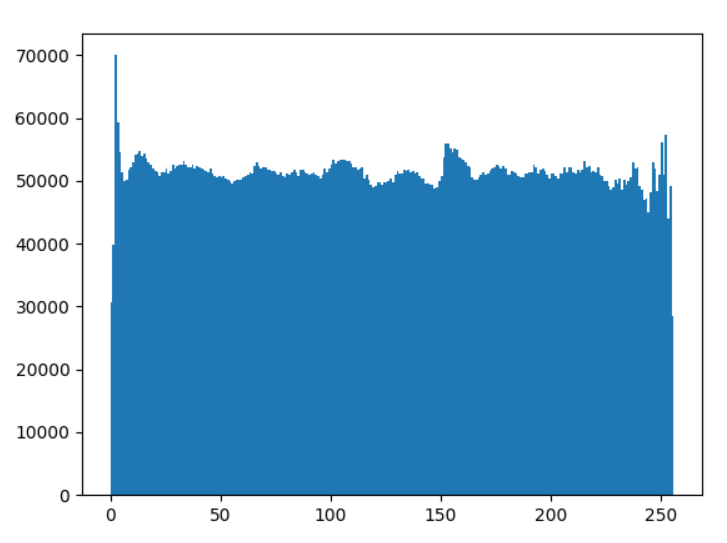
plt.pause(5)

plt.close()

**Histogram prije izjednačavanja:**



**Histogram poslije izjednačavanja:**



Vidimo da su vrijednosti intenziteta u histogramu sada gotovo izjednačene. U prošlom zadatku je cutoff gotovo sličnu stvar napravio ali je i za određeni postotak umanjio sve intenzitete. Gotovo su jednake slika iz ovog zadatka nakon izjednačavanja i nakon rastezanja s cutoff vrijednošću 5 (razlika je gotovo ne postojeća u kontrastu, ali je zato postojeća u brightnessu).