МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. І.СІКОРСЬКОГО»

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра фізико-технічних засобів захисту інформації

Лабораторна робота № 1

з дисципліни: «Автоматизація обробки ІзОД»

Варіант №1

Керівник: Виконав:

Прогонов Дмитро Олександрович студент 5 курсу

групи ФЕ-91мп

Захищено з оцінкою Баліцький Олег Анатолійович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, підпис

Київ – 2020 р.

**Хід роботи**

1. Сформувати тестову вибірку зображень з вихідного пакета;

Блок коду програми мовою високого рівня:

from os import listdir

from matplotlib import image

import random

import numpy as np

from dataclasses import dataclass

# load images in a directory

random.seed(1)

random\_indexes = random.sample(range(25000), 250)

loaded\_images = list()

for i in range(250):

# load image

filename = 'im' + str(random\_indexes[i]) + '.jpg'

img\_data = image.imread('Q:/mirflickr25k/mirflickr/' + filename)

# store loaded image

loaded\_images.append(img\_data)

print('> loaded %s %s' % (filename, img\_data.shape))

2. Для кожного каналу кольору кожного зображення з тестового пакета обчислити наступні характеристики:

a. Максимальне / мінімальне значення;

RED = 0

GREEN = 1

BLUE = 2

#max and min finding 1 task

print('Red: ', end='')

for i in range(255, 0, -1):

if (values[RED][i] != 0):

print('Max - {}, '.format(i), end='')

break

for i in range(0, 255):

if (values[RED][i] != 0):

print('Min - {}'.format(i))

break

print('Green: ', end='')

for i in range(255, 0, -1):

if (values[GREEN][i] != 0):

print('Max - {}, '.format(i), end='')

break

for i in range(0, 255):

if (values[GREEN][i] != 0):

print('Min - {}'.format(i))

break

print('Blue: ', end='')

for i in range(255, 0, -1):

if (values[BLUE][i] != 0):

print('Max - {}, '.format(i), end='')

break

for i in range(0, 255):

if (values[BLUE][i] != 0):

print('Min - {}'.format(i))

break

b. Математичне сподівання і дисперсію;

# Expected value & Variance

#RED

sum\_val = sum(values[RED])

M\_red = 0

for index in range(len(values[RED])):

p = (values[RED][index] / sum\_val)

M\_red += p \* index

D\_red = 0

for index in range(len(values[RED])):

p = (values[RED][index] / sum\_val)

D\_red += p \* ((index - M\_red) \*\* 2)

print("Red: Expected value - {0:.2f}, Variance - {1:.2f}"\

.format(M\_red, D\_red))

#GREEN

sum\_val = sum(values[GREEN])

M\_green = 0

for index in range(len(values[GREEN])):

p = (values[GREEN][index] / sum\_val)

M\_green += p \* index

D\_green = 0

for index in range(len(values[GREEN])):

p = (values[GREEN][index] / sum\_val)

D\_green += p \* ((index - M\_green) \*\* 2)

print("Green: Expected value - {0:.2f}, Variance - {1:.2f}"\

.format(M\_green, D\_green))

#BLUE

sum\_val = sum(values[BLUE])

M\_blue = 0

for index in range(len(values[BLUE])):

p = (values[BLUE][index] / sum\_val)

M\_blue += p \* index

D\_blue = 0

for index in range(len(values[BLUE])):

p = (values[BLUE][index] / sum\_val)

D\_blue += p \* ((index - M\_blue) \*\* 2)

print("Blue: Expected value - {0:.2f}, Variance - {1:.2f}"\

.format(M\_blue, D\_blue))

Вихідні дані:

Red: Expected value - 113.95, Variance - 5852.65

Green: Expected value - 107.31, Variance - 5392.78

Blue: Expected value - 99.49, Variance - 5774.33

c. Медіану значень, інтерквартільний розмах;

#Median and interquartile range

def medianFinder(array\_values):

sum\_val = sum(array\_values)

summa = 0

for index in range(len(array\_values)):

summa += array\_values[index]

if (summa > (sum\_val / 2)):

return index - 1

if (summa == (sum\_val / 2)):

return (2 \* index + 1) / 2

def quartFinder(array\_values, quart):

sum\_val = sum(array\_values)

summa = 0

for index in range(len(array\_values)):

prev\_sum = summa

p = array\_values[index] / sum\_val

summa += p

if (summa > quart):

if (quart - prev\_sum < summa - quart):

return index - 1

elif (quart - prev\_sum > summa - quart):

return index

else:

return (2 \* index - 1) / 2

#RED

print('Red: Median - {0}, IQR - {1}'.format(medianFinder(values[RED]),\

quartFinder(values[RED], 0.75) - quartFinder(values[RED], 0.25)))

print('Green: Median - {0}, IQR - {1}'.format(medianFinder(values[GREEN]),\

quartFinder(values[GREEN], 0.75) - quartFinder(values[GREEN], 0.25)))

print('Blue: Median - {0}, IQR - {1}'.format(medianFinder(values[BLUE]),\

quartFinder(values[BLUE], 0.75) - quartFinder(values[BLUE], 0.25)))

Вихідні дані:

Red: Median - 110, IQR - 131

Green: Median - 101, IQR - 121

Blue: Median - 87, IQR - 126

d. Коефіцієнти асиметрії та ексцесу (нормалізований);

#Asymmetry and excess

Asym\_red = E\_operator(values[RED], M\_red, 3) / (D\_red \*\* (3 / 2))

Asym\_green = E\_operator(values[GREEN], M\_green, 3) / (D\_green \*\* (3 / 2))

Asym\_blue = E\_operator(values[BLUE], M\_blue, 3) / (D\_blue \*\* (3 / 2))

Excess\_red = E\_operator(values[RED], M\_red, 4) / (D\_red \*\* 2)

Excess\_green = E\_operator(values[GREEN], M\_green, 4) / (D\_green \*\* 2)

Excess\_blue = E\_operator(values[BLUE], M\_blue, 4) / (D\_blue \*\* 2)

print('Red: Asymmetry - {0:.3f}, Excess - {1:.3f}'\

.format(Asym\_red, Excess\_red))

print('Green: Asymmetry - {0:.3f}, Excess - {1:.3f}'\

.format(Asym\_green, Excess\_green))

print('Blue: Asymmetry - {0:.3f}, Excess - {1:.3f}'\

.format(Asym\_blue, Excess\_blue))

Вихідні дані:

Red: Asymmetry - 0.174, Excess - 1.853

Green: Asymmetry - 0.275, Excess - 1.994

Blue: Asymmetry - 0.425, Excess - 2.024

3. Для кожного каналу кольору кожного зображення з тестового пакета побудувати гістограму значень яскравості пікселів;

import matplotlib.pyplot as plt

x = range(256)

#RED

plt.bar(x,values[RED],color='red')

plt.xlabel('Brightness')

plt.ylabel('Frequency')

plt.ylim(top=1500000)

plt.show()

#GREEN

plt.bar(x,values[GREEN],color='green')

plt.xlabel('Brightness')

plt.ylabel('Frequency')

plt.ylim(top=1500000)

plt.show()

#BLUE

plt.bar(x,values[BLUE],color='blue')

plt.xlabel('Brightness')

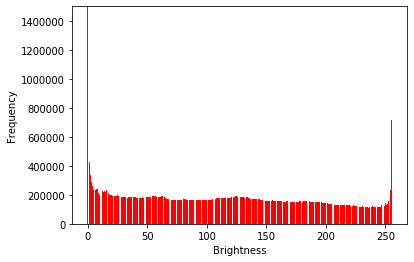
plt.ylabel('Frequency')

plt.ylim(top=1500000)

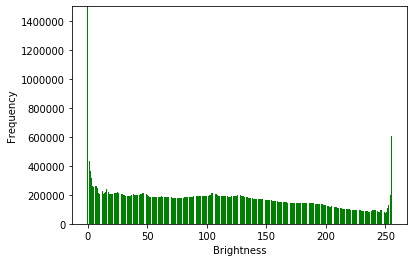
plt.show()

Вихідні дані:

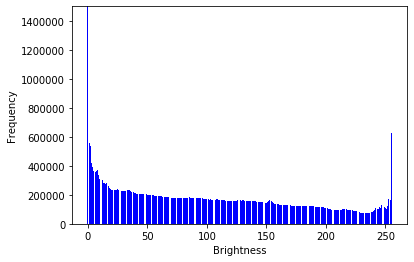
Гістограма значень яскравості червоних пікселів:



Гістограма значень яскравості зелених пікселів:



Гістограма значень яскравості синіх пікселів:



4. Провести апроксимацію отриманих гістограм з використанням відомих імовірнісних розподілів, визначити найкращу апроксимацію;

import seaborn as sns

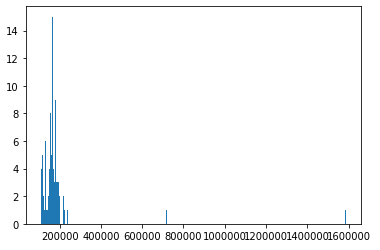
test = values[RED]

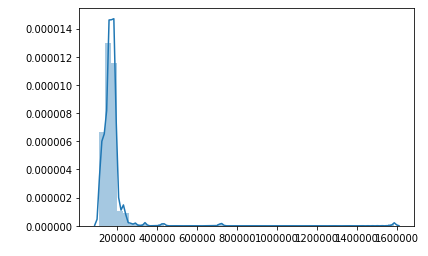
plt.hist(test,bins = 1000)

plt.show()

X,Y = sns.distplot(np.ravel(test)).get\_lines()[0].get\_data()

Вихідні дані:





5. Побудувати розподіл типів використаних імовірнісних розподілів для яких досягається мінімальне значення помилки апроксимації з п.4.

from scipy import stats

test = np.ravel(test)

xt = plt.xticks()[0]

xmin, xmax = min(xt), max(xt)

lnspc = np.linspace(0, 255, len(X))

m, s = stats.norm.fit((X,Y))#Standard, mean deviation

pdf\_g = stats.norm.pdf(lnspc, m, s) #Theoretical values

plt.plot(lnspc, pdf\_g, label="Norm",color = 'grey')

plt.plot(X,Y,color = 'black',linewidth = 4)

plt.plot(lnspc, pdf\_g, label="Norm")

ag,bg,cg = stats.gamma.fit((X,Y))

pdf\_gamma = stats.gamma.pdf(lnspc, ag, bg,cg)

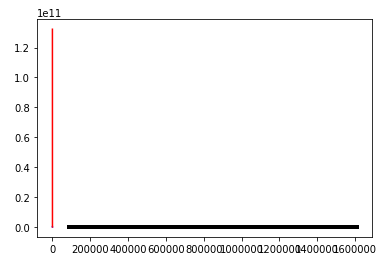
plt.plot(lnspc, pdf\_gamma, label="Gamma",color = 'blue')

ab,bb,cb,db = stats.beta.fit((X,Y))

pdf\_beta = stats.beta.pdf(lnspc, ab, bb,cb, db)

plt.plot(lnspc, pdf\_beta, label="Beta",color = 'red')

plt.show()



Як бачимо найближче до нашого розподілу апроксимує Beta.

**Висновки**

В даній лабораторній роботі було проаналізовано вибірку з 250 зображень датасету MIRFlickr-20k. Було знайдено що всі канали охоплюють увесь спектр значень. Було знайдено мат. очікування – 114 для чевоного каналу, 107 для зеленого і 99 для синього каналу відповідно і дисперсію 5852.65, 5392.78 та 5774.33 для цих же каналів. Медіану значень та інтерквартальний розподіл.

Також було знайдено коефіцієнти асиметрії. Оскільки вони були додатніми, то нахилений в бік спадання значень. Також коефіцієнт ексцесу був додатній для всіх розподілів.

Графіки приведено за допомогою бібліотеки matplotlib, вони повністю відповідають знайденим значенням.

При знаходженні апроксимації було показано топ відомих розподілів по схожості з нашим. Було отримано що бета-розподіл підходить краще інших.