МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НТУУ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Лабораторна робота №1

з дисципліни

«*Автоматизація обробки ІзОД*»

Варіант 4

**Виконав:**

студент 5 курсу ФТІ

групи ФЕ-91мп

Карнаух М.Ю.

**Перевірив:**

Прогонов Д. О.

КИЇВ 2020

**ПІДГОТОВКА**

**Вхідні дані**

Тестовий пакет – MIRFlickr-20k (https://press.liacs.nl/mirflickr/#sec\_download)

Вибірка зображень – 250 зображень;

Формування вибірки зображень – псевдовипадкове, з використанням генератора Мерсена (стартове значення співпадає з номером студента в загальному списку групи) за модулем кількості зображень в тестовому пакеті.

**Завдання**

1. Сформувати тестову вибірку зображень з вихідного пакета;

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

a. Максимальна / мінімальне значення;

b. Математичне сподівання і дисперсію;

c. Медіану значень, інтерквартільний розмах;

d. Коефіцієнти асиметрії та ексцесу (нормалізований);

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

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

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

**ХІД РОБОТИ**

Роботу виконуватимемо мовою Python за допомогою блокового інтерпретатора Jupyter.

**1. Формування тестової вибірки зображень з вихідного пакета**

**def** get\_test\_sample(path, quantity, random):

np.random.RandomState(random)

ind = np.random.choice(range(25000), quantity)

sample = list()

**for** i **in** range(quantity):

file = 'im' + str(ind[i]) + '.jpg'

sample.append(image.imread(path + file))

val = np.zeros((3, 256))

**for** img **in** sample:

**for** i **in** range(img.shape[0]):

**for** j **in** range(img.shape[1]):

val[0][img[i][j][0]] += 1

val[1][img[i][j][1]] += 1

val[2][img[i][j][2]] += 1

print('The sample is formed!')

**return** val, sample

**2. Знаходження статистичних даних**

1. **Максимальна / мінімальне значення**

Маючи вихідний масив з кількістю пікселів відповідної яскравості для знаходження максимального значення потрібно йти з кінця масиву до першого ненульового значення. Його індекс і казатиме про наявність пікселів відповідної яскравості. Для мінімального потрібно проробити те саме але з початку. Після виконання коду отримаємо наступні значення: red

Max 255,

Min 0

green

Max 255,

Min 0

blue

Max 255,

Min 0

****

1. **Математичне сподівання і дисперсія**

Для знаходження скористаємось відповідними формулами:

Де xi наше значення яскравості, а pi – ймовірність її появи. pi можна знайти як кількість пікселів даної яскравості поділену на всю кількість пікселів

**def** get\_matspod\_and\_disp(val, RGB, kanal):

print(kanal)

sum\_val = sum(val[RGB[kanal]])

matspod = 0

**for** ind **in** range(len(val[RGB[kanal]])):

k = (val[RGB[kanal]][ind] / sum\_val)

matspod += k \* ind

disp = 0

**for** ind **in** range(len(val[RGB[kanal]])):

k = (val[RGB[kanal]][ind] / sum\_val)

disp += k \* ((ind - matspod) \*\* 2)

matspodivannia[kanal] = matspod

dispersion[kanal] = disp

print(f'Matematuchne spodivannia **{matspod}**')

print(f'Dispersia **{disp}\n**')

Результат:

red   
Matematuchne spodivannia 118.37678442744522   
Dispersia 6141.082144840509

green   
Matematuchne spodivannia 110.31837030597202 Dispersia 5755.174281531292

blue   
Matematuchne spodivannia 99.78313890820282 Dispersia 6011.309896281168

**с. Медіана значень та інтерквартільний розмах.**

Для медіани будемо сумувати всі пікселі в масиві. Коли сума перевищить половину від кількості пікселів, буде визначена медіана.

Для інтерквартільного розмаху скористаємось тією ж логікою але будемо шукати вже сумарну імовірність пройдених пікселів.

**def** get\_median(m):

sum\_val = sum(m)

summa = 0

**for** ind **in** range(len(m)):

summa += m[ind]

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

**return** ind - 1

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

**return** (2 \* ind + 1) / 2

**def** get\_quartile(m, quartile):

summa\_values = sum(m)

summa = 0

**for** i **in** range(len(m)):

prev\_sum = summa

p = m[i] / summa\_values

summa += p

**if** (summa > quartile):

**if** (quartile - prev\_sum < summa - quartile):

**return** i - 1

**elif** (quartile - prev\_sum > summa - quartile):

**return** i

**else**:

**return** (2 \* i - 1) / 2

**def** get\_median\_quartile(val, RGB, kanal):

print(kanal)

print(f'Mediana {get\_median(val[RGB[kanal]])}')

print(f'Quartile {get\_quartile(val[RGB[kanal]], 0.75) - get\_quartile(val[RGB[kanal]], 0.25)}**\n**')

Результат:

red

Mediana 115

Quartile 137

green

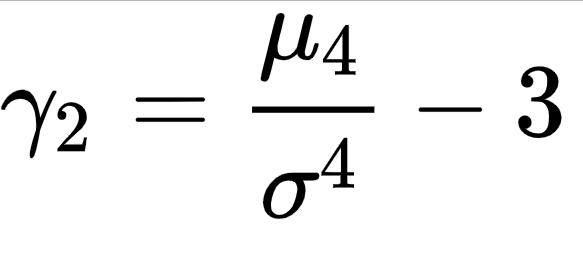
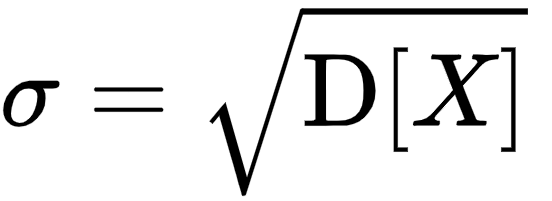
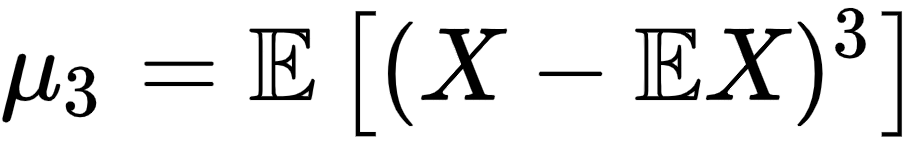
Mediana 106

Quartile 131

blue

Mediana 86

Quartile 134

**d. Коефіцієнти асиметрії та ексцесу**

Для цього також використаємо відповідні формули коефіцієнту асиметрії та коефіцієнту ексцесу:



**def** fi(m, matspod, p):

sum\_val = sum(m)

f = 0

**for** ind **in** range(len(m)):

k = (m[ind] / sum\_val)

f += k \* ((ind - matspod) \*\* p)

**return** f

**def** get\_asymm\_excess(val, RGB, kanal):

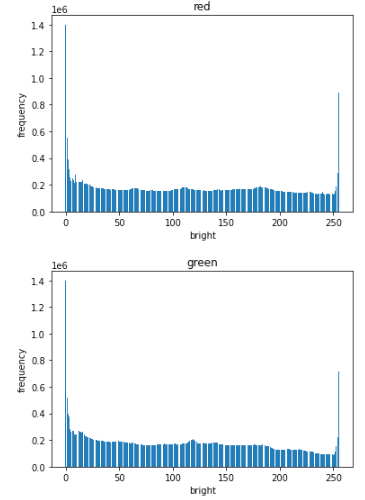
print(kanal)

asymm = fi(val[RGB[kanal]], matspodivannia[kanal], 3) / (dispersion[kanal] \*\* (3 / 2))

excess = fi(val[RGB[kanal]], matspodivannia[kanal], 4) / (dispersion[kanal] \*\* 2) - 3

print(f'Asymmetry **{asymm}**')

print(f'Excess **{excess}\n**')

Результат:

red

Asymmetry 0.08929185045460528

Excess -1.2254031382987443

green

Asymmetry 0.2098846887141036

Excess -1.1277698056973646

blue

Asymmetry 0.411238169724522

Excess -1.059775556723196

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

**def** get\_histogram(val, RGB, kanal):

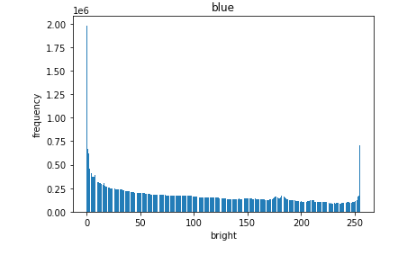
plt.bar(range(256), val[RGB[kanal]])

plt.title(kanal)

plt.xlabel('bright')

plt.ylabel('frequency')

plt.show()



**4-5. Провести апроксимацію отриманих гістограм з використанням відомих ймовірнісних розподілів, визначити найкращу апроксимацію. Побудувати розподіл типів використаних імовірнісних розподілів для яких досягається мінімальне значення помилки апроксимації**

**def** get\_apr(val, RGB, kanal):

plt.hist(val[RGB[kanal]], bins=1000)

plt.title(kanal)

plt.show()

X, Y = seaborn.distplot(np.ravel(val[RGB[kanal]])).get\_lines()[0].get\_data()

np.ravel(val[RGB[kanal]])

**return** X, Y

X, Y = get\_apr(val, RGB, 'red')

green\_apr = get\_apr(val, RGB, 'green')

blue\_apr = get\_apr(val, RGB, 'blue')

x\_min, x\_max = min(plt.xticks()[0]), max(plt.xticks()[0])

m, s = stats.norm.fit((X, Y))

pg = stats.norm.pdf(np.linspace(0, 255, len(X)), m, s)

plt.plot(np.linspace(0, 255, len(X)), pg, label='norm', color='black')

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

plt.plot(np.linspace(0, 255, len(X)), pg, label='norm')

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

kgamma = stats.gamma.pdf(np.linspace(0, 255, len(X)), ag, bg, cg)

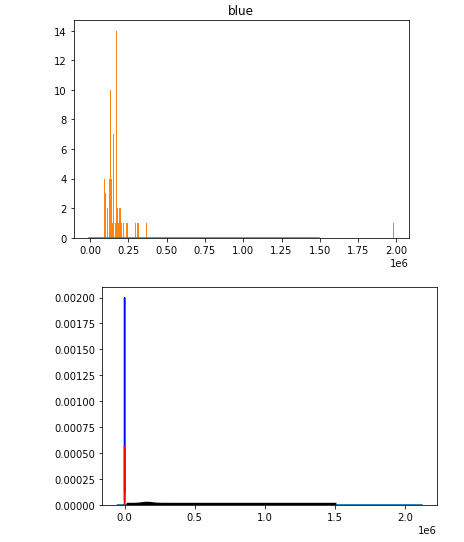
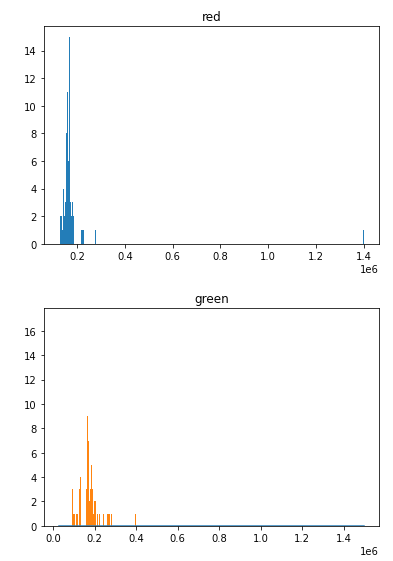
plt.plot(np.linspace(0, 255, len(X)), kgamma, label='gamma', color='blue')

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

kbeta = stats.beta.pdf(np.linspace(0, 255, len(X)), ab, bb, cb, db)

plt.plot(np.linspace(0, 255, len(X)), kbeta, label='beta', color='red')

plt.show()

 Результати:

**ВИСНОВКИ**

В даній лабораторній роботі було проаналізовано вибірку з 250 зображень. Було знайдено що всі канали охоплюють увесь спектр значень. Було знайдено мат. очікування і дисперсію для трьох каналів, медіану значень та інтерквартільний розподіл, коефіцієнти асиметрії та ексцесу.

Побудовано відповідні графіки, які повністю відповідають знайденим значенням.