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 №3

Керівник: Виконав:

Прогонов Дмитро Олександрович студент 5 курсу групи ФЕ-91мп

Захищено з оцінкою Євдокимов О.О.

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

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

дата, підпис

Київ – 2020 р.Завдання на лабораторну роботу:

1. Сформувати тестову вибірку зображень з вихідного пакета;
2. Для кожного каналу кольору кожного зображення з тестового пакета обчислити наступні характеристики:
   1. Максимальна / мінімальне значення;
   2. Математичне сподівання і дисперсію;
   3. Медіану значень, інтерквартільний розмах;
   4. 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**

**1) Формування тестової вибірки зображеннь**

Лабораторна робота буде виконана мовою Python за допомогою блокового інтерпритатора Jupyter. В данній лабораторній роботі використовувались наступні бібліотеки:

* Scipy
* Matplotlib
* Matplotlib.pyplot
* Os
* Numpy
* Seaborn
* Random

Зробимо вибірку псевдовипадкової кількості 250 кольорових зображень із пакету mirflick та сформуємо матрицю для збору статистичних даних про зобьраження. Для цього створимо двомірний Numpy масив на три рядки для кожного каналу кольору та на 256 стовпчиків, що відповідатиме кількості пікселів відповідної яскравості

*# 1.generate a test sample of images from the source package*

np.random.RandomState(3)

sample\_size = 250

indexes = np.random.choice(range(25000), sample\_size)

images = list()

**for** i **in** range(sample\_size):

filename = 'im' + str(indexes[i]) + '.jpg'

img\_data = img.imread('.../downloads/mirflickr/' + filename)

images.append(img\_data)

print(f'**{sample\_size}** images uploaded, processing begins...')

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

**for** image **in** images:

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

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

values[0][image[i][j][0]] += 1

values[1][image[i][j][1]] += 1

values[2][image[i][j][2]] += 1

print('all images processed')

**2) Обчислення характеристик для кожного каналу зображень**

1. Максимальна / мінімальне значення

Після виконання коду:

colors = {

'RED': 0,

'GREEN': 1,

'BLUE': 2

}

**for** color **in** colors:

print(color + ':')

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

**if** (values[colors[str(color)]][i] != 0):

print(f'maximum value - **{i}**, ')

**break**

**for** i **in** range(0, 255):

**if** (values[colors[str(color)]][i] != 0):

print(f'minimum value - **{i}**')

**break**

отимаємо наступні значення :

RED:

Max:255, Min: 0

BLUE:

Max:255, Min: 0

GREEN:

Max:255, Min: 0

1. Математичне сподівання і дисперсія

Розрахунки будуть проводитись за наступними формулами:



(1),



(2),

Де (1) - математичне очікування, а (2) – дисперсія, xi - значення яскравості, pi – ймовірність її появи. pi можна знайти як кількість пікселів даної яскравості поділену на всю кількість пікселів

expectations = {

'RED': 0,

'GREEN': 0,

'BLUE': 0

}

variances = {

'RED': 0,

'GREEN': 0,

'BLUE': 0

}

**for** color **in** colors:

print(color + ':')

sum\_values = sum(values[colors[str(color)]])

expectation = 0

**for** i **in** range(len(values[colors[str(color)]])):

p = (values[colors[str(color)]][i] / sum\_values)

expectation += p \* i

variance = 0

**for** i **in** range(len(values[colors[str(color)]])):

p = (values[colors[str(color)]][i] / sum\_values)

variance += p \* ((i - expectation) \*\* 2)

expectations[str(color)] = expectation

variances[str(color)] = variance

print(f'expectation - **{expectation}**')

print(f'variance - **{variance}**')

Таким чином отримуэмо значення Мат. очікування та дисперсії:

RED:

expectation - 113.60443421273365

variance - 6248.877506995998

GREEN:

expectation - 107.30682031381062

variance - 5613.174370057347

BLUE:

expectation - 98.59806754867645

variance - 5781.50450273308

с. Медіана значень та інтерквартальний розмах.

Для медіани будемо сумувати всі кількості пікселів в масиві і коли сума перевалить за половину кількості пікселів – ми будемо знати що це і є медіана

**def** median\_values(array):

summa\_values = sum(array)

summa = 0

**for** index **in** range(len(array)):

summa += array[index]

**if** (summa > (summa\_values / 2)):

**return** index - 1

**if** (summa == (summa\_values / 2)):

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

Для інтерквартального розмаху працюємо аналогічно, але будемо шукати вже сумарну імовірність пройдених пікселів.

**def** interquartile\_range(array, interquartile):

summa\_values = sum(array)

summa = 0

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

prev\_summa = summa

p = array[i] / summa\_values

summa += p

**if** (summa > interquartile):

**if** (interquartile - prev\_summa < summa - interquartile):

**return** i - 1

**elif** (interquartile - prev\_summa > summa - interquartile):

**return** i

**else**:

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

Виведемо оброблено інформацію:

**for** color **in** colors:

print(color + ':')

print(f'median values - {median\_values(values[colors[str(color)]])}')

print(f'interquartile range - {interquartile\_range(values[colors[str(color)]], 0.75) - interquartile\_range(values[colors[str(color)]], 0.25)}')

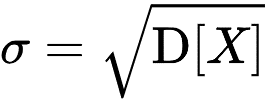
Таким чином отримаємо :

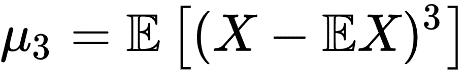
RED: median values - 110 interquartile range - 137 GREEN: median values - 102 interquartile range - 125 BLUE: median values - 89 interquartile range – 130

d. Коефіцієнти асиметрії та ексцесу

Використаємо відповідні формули коефіцієнту асиметрії:







Реалізуємо даний функціонал за допомогою такого коду:

**def** finder(array, expectation, power):

summa\_values = sum(array)

final = 0

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

p = (array[i] / summa\_values)

final += p \* ((i - expectation) \*\* power)

**return** final

**for** color **in** colors:

print(color + ':')

asymmetry = finder(values[colors[color]], expectations[color], 3) / (variances[color] \*\* (3 / 2))

excess = finder(values[colors[color]], expectations[color], 4) / (variances[color] \*\* 2) - 3

print(f'asymmetry - **{asymmetry}**')

print(f'excess - **{excess}**')

Отримаємо:

RED:

asymmetry - 0.1608408344525885

excess - -1.202426131141161

GREEN:

asymmetry - 0.2596345757144237

excess - -1.046134480329933

BLUE:

asymmetry - 0.3781474508728697

excess - -1.0105627217737896

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

Звернемося до раніше використанних бібліотек matplotlib та matplotlib.pyplot , таким чином отримаємо наступні значення:

x = range(256)

**for** color **in** colors:

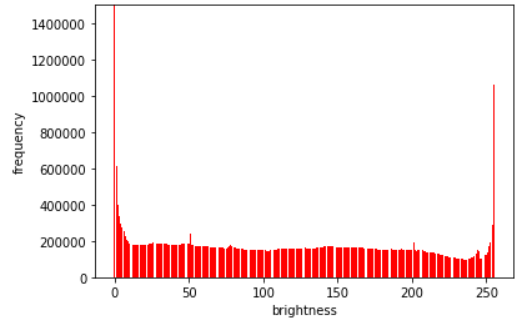
plt.bar(x, values[colors[color]], color=color.lower())

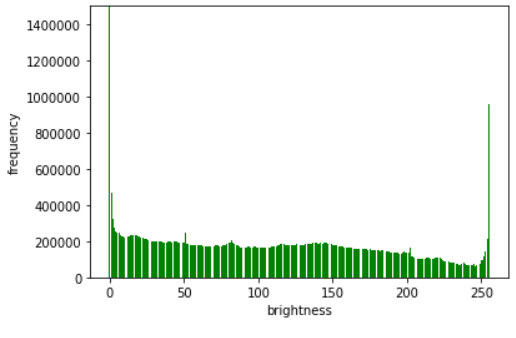
plt.xlabel('brightness')

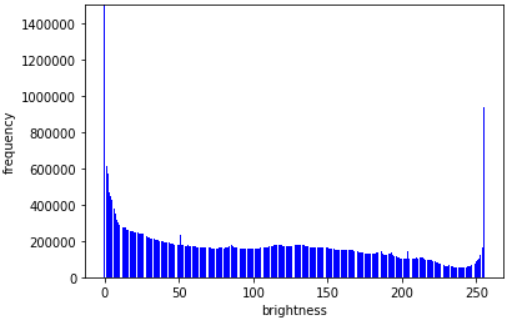
plt.ylabel('frequency')

plt.ylim(top=1500000)

plt.show()



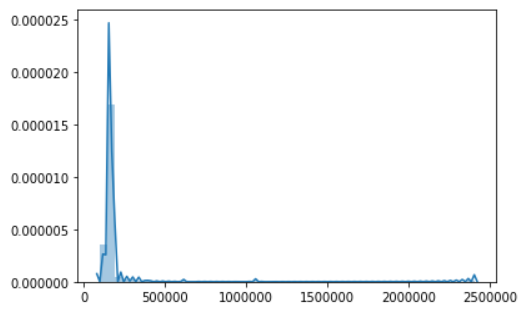




1. **Провести апроксимацію отриманих гістограм з використанням відомих імовірнісних розподілів, визначити найкращу апроксимацію**

Для даного завдання звернемося до раніше використанної бібліотеки Seaborn.

test = values[colors['RED']] plt.hist(test, bins=1000) plt.show() X, Y = sns.distplot(np.ravel(test)).get\_lines()[0].get\_data()



Побудована апроксимація описує наші дані, тепер використаемо відомі розподіли:

test = np.ravel(test)

xt = plt.xticks()[0]

x\_min, x\_max = min(xt), max(xt)

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

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

pg = stats.norm.pdf(l, m, s)

plt.plot(l, pg, label="norm", color='black')

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

plt.plot(l, pg, label="norm")

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

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

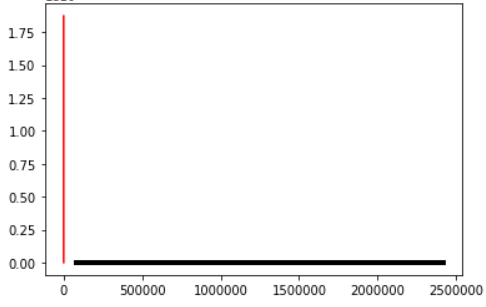
plt.plot(lnspc, pgamma, label="gamma", color='blue')

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

pbeta = stats.beta.pdf(l, ab, bb, cb, db)

plt.plot(l, pbeta, label="beta", color='red')

plt.show()



Таким чином, до нашого розподілу апроксимує Beta.

**ВИСНОВКИ**

У данній лабораторній роботі було розраховано значення статистичних характеристик вибірки 250 зображення для кожного каналу кольорів. Було знайдено що всі канали охоплюють увесь спектр значень.

Розраховано мат.очікування для кожного кольору:

RED:

expectation - 113.60443421273365

variance - 6248.877506995998

GREEN:

expectation - 107.30682031381062

variance - 5613.174370057347

BLUE:

expectation - 98.59806754867645

variance - 5781.50450273308

Медіану значень та інтерквартальний розподіл:

RED: median values - 110 interquartile range - 137 GREEN: median values - 102 interquartile range - 125 BLUE: median values - 89 interquartile range – 130

Також було знайдено коефіцієнти асиметрії та коефіцієнт ексцесу:

RED:

asymmetry - 0.1608408344525885

excess - -1.202426131141161

GREEN:

asymmetry - 0.2596345757144237

excess - -1.046134480329933

BLUE:

asymmetry - 0.3781474508728697

excess - -1.0105627217737896

Так, як коефіцієнт асиметрії додатній, то напрямок грайіка буде в бік спадання значень.

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

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