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»

Керівник:
Прогонов Дмитро Олександрович

Виконав:
студент 5 курсу
групи ФЕ-91мп
Соколовський Владислав

Захищено з оцінкою

_____

_____

дата, підпис

Київ – 2020 р.

I. Вступ

Вихідні 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 повинно збігатися 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.

II. Хід роботи

Роботу виконуватимемо мовою Python.Також в роботі будуть використані такі бібліотеки як:

- Os
- Matplotlib
- Numpy
- Scipy
- Pandas
- та інші

1. Формування тестової вибірки зображень з вихідного пакета

Для цього скористаємося функцією numpy.random() що обирає випадкові числа з переданого масиву за допомогою генератора Мерсена.

```
def create_index_list(file_list,count):
    print('\n>>Creating list of indexes')
    index_list = np.random.random_integers(0,len(file_list)-1,count)
    filtered_file_list=list()
    for index in tqdm(index_list):
        filtered_file_list.append(file_list[index])
    return index_list,filtered_file_list
```

Також задамо початкове значення варіанту за допомогою функції numpy.random.seed()

```
def config_random_generator(seed=14):
    print('>>Configure generator')
    np.random.seed(int(seed))
    generator_info = np.random.get_state()
    return
```

Після цього отриманий масив зображень буде знаходитись в loaded_images в виді двомірного масиву з трьома значеннями яскравості в кожній комірці.

```
files = lib.create_list_files()
index_list,files = lib.create_index_list(files,config['countImages'])
```

Тепер сформуємо матрицю для збору статистичних даних.

```
        def get_images_info(image_list):
            data = {}
            print('\n\n\n\n>>Analyzing images')
            for color_index,color_name in enumerate(RGB):
                print('\t\t>>Get info about '+str(color_name)+' color')
                data[color_name] = pd.DataFrame()
                for image_index,image_name in tqdm(enumerate(image_list)):
                    data[color_name] = pd.concat([data[color_name],
pd.DataFrame(pd.DataFrame(get_image_info(image_index,image_name,color_index),
index=[0, ]))], ignore_index=True)

data[color_name].to_csv(path_or_buf='./output/lab1/'+color_name+'.csv',index=False)
                print('\t\t>>Write data to ./output/lab1/'+color_name+'.csv\n\n')
```

2. Знаходження статистичних даних

a. 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.

b. Математичне сподівання і дисперсія

Для знаходження скористаємось відповідними формулами:

$$D[X] = \sum_{i=1}^{n} p_i (x_i - M[X])^2,$$

$$M\left[g(X)\right] = \sum_{i=1}^{\infty} g(x_i) p_i,$$

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

c. Медіана значень та інтерквартальний розмах.

Для пошуку медіани та інтерквартального розмаху скористаємось функціями макету numpy:

```
np.nanmedian(a) # медиана
sp.stats.iqr(a) # интерквартильный размах
```

d. Коефіцієнти асиметрії та ексцесу

Для пошуку медіани та інтерквартального розмаху скористаємось функціями макету scipy:

```
sp.stats.skew(a), # коэффициент асимметрии
sp.stats.kurtosis(a), # коэффициéнт эксцéсса
```

Після виконання коду:

```
def get_image_info(image_index,image_name,color_index):
    image = np.array(Image.open(image_name))
    a = image[ :, color_index].ravel()
    d = {
        'name': image_name,  # название файла
        'min': np.nanmin(a),  # минимум
        'max': np.nanmax(a),  # максимум
        'mean': np.nanmean(a),   # среднеарифметическое
        'var': np.nanvar(a), # дисперсия
        'median': np.nanmedian(a),  # медиана
        'average': np.average(a), # средневзвешенное(мат ожидание)
        'std': np.nanstd(a), # среднеквадратичное (стандартное) отклонение
        'skewness': sp.stats.skew(a), # коэффициент асимметрии
        'kurtosis': sp.stats.kurtosis(a), # коэффициéнт эксцéсса
        'interquartile range': sp.stats.iqr(a), # интерквартильный размах
        'best distribution': get_image_histogram(image_index,image_name)
        }
    return d
```

отримаємо наступні значення для кожного кольру кожного сображення:

```
average,best distribution,interquartile
range,kurtosis,max,mean,median,min,name,skewness,std,var
101.58758758758759,beta,60.0,-0.7437830509563512,172,101.58758758758759,110.0,9,./input/
mirflickr/im19052.jpg,-0.5088496333246547,39.04490479789601,1524.504590676763
246.2867540029112,beta,9.0,-0.8300230366027579,255,246.2867540029112,247.0,233,./input/m
irflickr/im13657.jpg,-0.2473624533422034,5.679390133401192,32.255472287374815
```

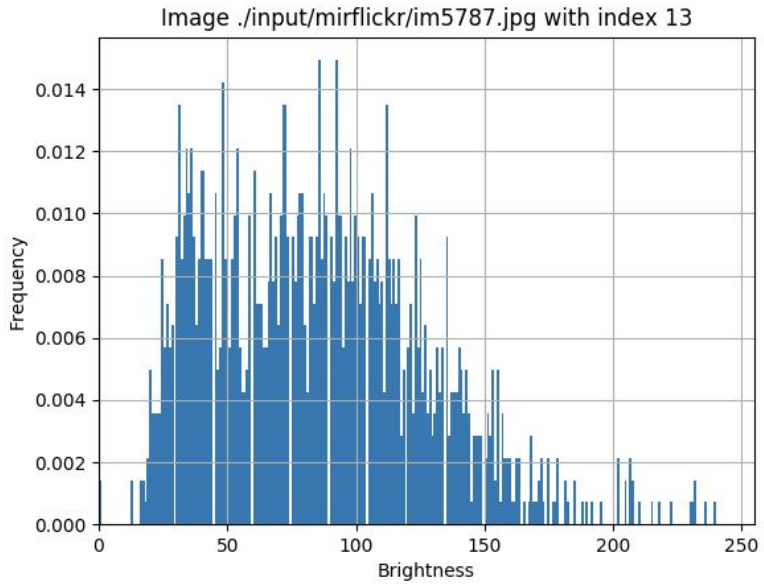
```
147.84733333333332,beta,4.0,-0.19336374177636895,155,147.84733333333332,148.0,137,./inpu
t/mirflickr/im9485.jpg,-0.3831508928596452,3.130925244857962,9.802692888888888
0.3774104683195592,beta,0.0,63.08458248026247,14,0.3774104683195592,0.0,0,./input/mirfli
ckr/im18839.jpg,6.621544503029918,1.0520801457914941,1.1068726331686514
177.44544544544544,laplace,97.0,-0.723479287578332,255,177.44544544544544,196.0,1,./inpu
t/mirflickr/im22856.jpg,-0.667994086672298,57.617693249141176,3319.798575352129
```

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

Для цього скористаємося бібліотекою matplotlib:

```
plt.figure()

plt.xlim(0,255)

plt.xlabel('Brightness')

plt.ylabel('Frequency')

plt.title('Image '+ str(image_name)+ ' with index '+ str(image_index))


plt.savefig('./output/lab1/'+str(color_name)+'_histograms/'+str(image_index)+'.p
ng')
```

Отримаємо:



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

Для цього скористаємося можливостями бібліотек scipy та pandas.

```
def get_image_histogram(image_index,image_name):
   print('\n\n\t>>Get histogram about:  '+ image_name)
   for color_index,color_name in enumerate(RGB):
       plt.figure()
       image = np.array(Image.open(os.path.join(image_name)))
       a = image[ :, color_index].ravel()
       f = Fitter(a, distributions=distributions.keys(), bins=256)
       f.fit()
       # pprint(f.summary())
       f.hist();
       best_distribution = f.get_best()
       print('\n\n\n*******************************')
       print(best_distribution)
       print('*******************************\n\n\n')
       plt.xlim(0,255)
       plt.xlabel('Brightness')
       plt.ylabel('Frequency')
       plt.title('Image '+ str(image_name)+ ' with index '+ str(image_index))


plt.savefig('./output/lab1/'+str(color_name)+'_histograms/'+str(image_index)+'.png'
)
   return best_distribution.keys()[0]
```

Порівняємо наші графіки з вбудованими розподілами, такими як:

```
'beta':0, 'gamma':0, 'laplace':0, 'norm':0, 'pareto':0
```

Отримаємо наступну статистику

```
Fitted beta distribution with error=14.9958106461)
Fitted norm distribution with error=15.0711828601)
Fitted pareto distribution with error=15.3335313629)
Fitted laplace distribution with error=15.1936141624)
Fitted gamma distribution with error=15.0715386534)
```

```
******************************
{'beta': (2.2974426739102585, 0.8893100095501101,
229.50095866454916, 25.499041335450844)}
******************************
```