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 № 2

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. - Математичне сподівання і дисперсію;
    b. - Коефіцієнти асиметрії та ексцесу (нормалізований);
3. Побудувати вектори параметрів зображень, що складаються з:
    a. Математичних очікувань значень яскравості для кожного каналу кольору;
    b. Математичних очікувань і дисперсії значень яскравості для кожного каналу кольору;
    c. Математичних очікувань, дисперсії і коефіцієнта асиметрії значень яскравості для кожного каналу кольору;
    d. Математичних очікувань, дисперсії, коефіцієнтів асиметрії та ексцесу значень яскравості для кожного каналу кольору;
4. Побудувати гаусові моделі зображень з використанням розрахованих раніше параметрів.
5. Провести декомпозицію кожного каналу кольору кожного зображення з застосуванням методу головних компонент (PCA):
    a. Варіюючи кількість компонент, провести реконструкцію окремих каналів кольору зображень (від компонент з найбільшою енергією поступово переходячи до компонентів з мінімальною енергією).
    b. Побудувати залежність помилки відновлення (середнє відхилення вихідного зображення відреконструйованого, MSE) від кількості використаних компонент.
6. Провести моделювання окремих каналів кольору зображень з використанням марковських ланцюгів:
    a. Для кожного каналу кольору кожного зображення розрахувати стохастическую матрицю марковської ланцюга першого і другого порядків (обробка пікселів по горизонталі справа наліво і навпаки, а також по вертикалі зверху вниз і навпаки). У звіті привести явний вигляд однієї марковської ланцюга для одного з каналів кольору тестового зображення;
    b. Перевірити властивість регулярності, реккурентное і незворотності (irreducible) для отриманих марковских моделей для 5 ітерацій.

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. Побудувати вектори параметрів зображень:

Для цього сформуємо вектор всіх потрибних нам значень, та для розрахунків будемо брати окремі частини готових даних:

```
def get_images_vectorData(image_list):
    data = {}
    print('\n\n\n\n>>Get vector data of images')
    vectorData = {}
    for color_index, color_name in enumerate(RGB):
        vectorData[color_name] = {}
        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)):
            image_info = get_image_info_vectors(
                image_index, image_name, color_index)
            for parameter in image_info:
                if not parameter in vectorData[color_name]:
                    vectorData[color_name][parameter] = []

vectorData[color_name][parameter].append(image_info[parameter])

        output_file_path =
'./output/'+output_dir+'/vectorData/'+color_name+'.csv'
        output_file = open(output_file_path, "w")
        writer = csv.writer(output_file)
        vectorDataParameters = vectorData[color_name].keys()
        writer.writerow(vectorDataParameters)
        for i in
tqdm(range(len(vectorData[color_name][vectorDataParameters[0]]))):
            row = []
            for parameter in vectorDataParameters:
                row.append(vectorData[color_name][parameter][i])
            writer.writerow(row)
        print('\t\t>>Write data to '+output_file_path+'\n\n')
    return vectorData
```

Для синього кольору обробка двох перших картинок:

```
var,skewness,kurtosis,mean
```

```
1524.504590676763,-0.5088496333246547,-3.743783050956351,101.58758758758759
32.255472287374815,-0.2473624533422034,-3.830023036602758,246.2867540029112
```

Для створення векторів скористаемось функцією:

```
def create_vectors(vectorData):
   print('>>Creating vectors')
   vector = []
   for color_name in vectorData:
       v = np.array([vectorData[color_name]['mean'],
vectorData[color_name]['var'],
                     vectorData[color_name]['skewness'],
vectorData[color_name]['kurtosis']])
       vector.append(v)

   vector = np.array(vector)

   output_file_path = './output/'+output_dir+'/vectors/vector.txt'
   with open(output_file_path, 'w') as f:
       csv.writer(f, delimiter=' ').writerows(vector)

   matrix = []
   vectorDataParameters = vectorData[color_name].keys()
   for parameter in vectorDataParameters:
       m = np.array((vectorData['red'][parameter],
vectorData['green']
                     [parameter], vectorData['blue'][parameter]))
       matrix.append(m)

   output_file_path =
'./output/'+output_dir+'/vectors/mean_matrix.txt'
   with open(output_file_path, 'w') as f:
       csv.writer(f, delimiter=' ').writerows(matrix)

   mean_var_vector = ((vectorData['red']['mean'],
vectorData['green']['mean'], vectorData['blue']['mean'],
                       vectorData['red']['var'],
vectorData['green']['var'], vectorData['blue']['var']))
```

```
   output_file_path =
'./output/'+output_dir+'/vectors/mean_var_vector.txt'
   with open(output_file_path, 'w') as f:
       csv.writer(f, delimiter=' ').writerows(mean_var_vector)

   mean_var_skew_vector = ((vectorData['red']['mean'],
vectorData['green']['mean'], vectorData['blue']['mean'],
                            vectorData['red']['var'],
vectorData['green']['var'], vectorData['blue']['var'],
                            vectorData['red']['skewness'],
vectorData['green']['skewness'], vectorData['blue']['skewness']))

   output_file_path =
'./output/'+output_dir+'/vectors/mean_var_skew_vector.txt'
   with open(output_file_path, 'w') as f:
       csv.writer(f, delimiter='
').writerows(mean_var_skew_vector)

   mean_var_skew_kurt_vector = ((vectorData['red']['mean'],
vectorData['green']['mean'], vectorData['blue']['mean'],
                                vectorData['red']['var'],
vectorData['green']['var'], vectorData['blue']['var'],
                                vectorData['red']['skewness'],
vectorData['green']['skewness'], vectorData['blue']['skewness'],
                                vectorData['red']['kurtosis'],
vectorData['green']['kurtosis'], vectorData['blue']['kurtosis']))
   output_file_path = './output/'+output_dir + \
       '/vectors/mean_var_skew_kurt_vector.txt'
   with open(output_file_path, 'w') as f:
       csv.writer(f, delimiter='
').writerows(mean_var_skew_kurt_vector)
```

Отримаемо такі результати:

mean_matrix.txt

```
"[1325.15570425   32.24833411]" "[1440.63898333   32.24985539]"
"[1524.50459068   32.25547229]"
"[-0.4038645  -0.24893708]" "[-0.48883685 -0.24668549]"
"[-0.50884963 -0.24736245]"
"[-3.7608708  -3.81938603]" "[-3.6985433  -3.82998722]"
"[-3.74378305 -3.83002304]"
"[102.63863864 246.29403202]" "[102.17517518 246.28384279]"
"[101.58758759 246.286754  ]"
```

mean_var_skew_kurt_vector.txt

```
102.63863863863864 246.29403202328967
102.17517517517517 246.28384279475983
101.58758758758759 246.2867540029112
1325.1557042528013 32.248334106689214
1440.6389833276721 32.24985539304488
1524.504590676763 32.255472287374815
-0.40386449796999685 -0.24893707922666128
-0.48883684833443136 -0.24668548601772095
-0.5088496333246547 -0.2473624533422034
-3.7608707995362565 -3.8193860347586206
-3.6985432994119356 -3.82998722122842
-3.743783050956351 -3.830023036602758
```

mean_var_skew_vector.txt

```
102.63863863863864 246.29403202328967
102.17517517517517 246.28384279475983
101.58758758758759 246.2867540029112
1325.1557042528013 32.248334106689214
1440.6389833276721 32.24985539304488
1524.504590676763 32.255472287374815
-0.40386449796999685 -0.24893707922666128
-0.48883684833443136 -0.24668548601772095
-0.5088496333246547 -0.2473624533422034
```

mean_var_vector.txt

```
102.63863863863864 246.29403202328967
102.17517517517517 246.28384279475983
101.58758758758759 246.2867540029112
1325.1557042528013 32.248334106689214
1440.6389833276721 32.24985539304488
1524.504590676763 32.255472287374815
```

vector.txt

```
"[101.58758759 246.286754  ]" "[1524.50459068   32.25547229]"
"[-0.50884963 -0.24736245]" "[-3.74378305 -3.83002304]"
"[102.17517518 246.28384279]" "[1440.63898333   32.24985539]"
"[-0.48883685 -0.24668549]" "[-3.6985433  -3.82998722]"
"[102.63863864 246.29403202]" "[1325.15570425   32.24833411]"
"[-0.4038645  -0.24893708]" "[-3.7608708  -3.81938603]"
```

4. Побудувати гаусові моделі зображень з використанням розрахованих раніше параметрів.

Тепер сформуємо гаусові моделі. Для першого випадку маємо одномірний варіант лише з мат очікуванням:

```
def get_gauss_model(vectorData):
   print('>>Creating gauss models')
   mean_g_v = np.cov(np.vstack(
       (vectorData['red']['mean'], vectorData['green']['mean'],
vectorData['blue']['mean'])))
   output_file_path =
'./output/'+output_dir+'/gaussModels/mean_g_v.txt'
   with open(output_file_path, 'w') as f:
       csv.writer(f, delimiter=' ').writerows(mean_g_v)

   mean_var_g_v = np.cov(np.vstack((vectorData['red']['mean'],
vectorData['green']['mean'], vectorData['blue']['mean'],
                                    vectorData['red']['var'],
vectorData['green']['var'], vectorData['blue']['var'])))
   output_file_path =
'./output/'+output_dir+'/gaussModels/mean_var_g_v.txt'
   with open(output_file_path, 'w') as f:
       csv.writer(f, delimiter=' ').writerows(mean_var_g_v)

   mean_var_skew_g_v =
np.cov(np.vstack((vectorData['red']['mean'],
vectorData['green']['mean'], vectorData['blue']['mean'],
                                         vectorData['red']['var'],
vectorData['green']['var'], vectorData['blue']['var'],

vectorData['red']['skewness'], vectorData['green']['skewness'],
vectorData['blue']['skewness'])))
   output_file_path =
'./output/'+output_dir+'/gaussModels/mean_var_skew_g_v.txt'
   with open(output_file_path, 'w') as f:
       csv.writer(f, delimiter=' ').writerows(mean_var_skew_g_v)

   mean_var_skew_kurt_g_v =
np.cov(np.vstack((vectorData['red']['mean'],
vectorData['green']['mean'], vectorData['blue']['mean'],
```

```
vectorData['red']['var'], vectorData['green']['var'],
vectorData['blue']['var'],

vectorData['red']['skewness'], vectorData['green']['skewness'],
vectorData['blue']['skewness'],

vectorData['red']['kurtosis'], vectorData['green']['kurtosis'],
vectorData['blue']['kurtosis'])))
   output_file_path = './output/'+output_dir + \
       '/gaussModels/mean_var_skew_kurt_g_v.txt'
   with open(output_file_path, 'w') as f:
       csv.writer(f, delimiter='
').writerows(mean_var_skew_kurt_g_v)
```

Отримаємо такі результати

mean_g_v.txt

```
10318.43602424942 10350.993668514679 10393.4078369122
10350.993668514679 10383.654041545966 10426.20203888842
10393.4078369122 10426.20203888842 10468.92438064476
```

mean_var_g_v.txt

```
10318.43602424942 10350.993668514679 10393.4078369122
-92866.55843412718 -101161.34710605725 -107184.81706506312
10350.993668514679 10383.654041545966 10426.20203888842
-93159.5792336487 -101480.54035828395 -107523.0161037973
10393.4078369122 10426.20203888842 10468.92438064476
-93541.30935618535 -101896.36640027257 -107963.60175747302
-92866.55843412718 -93159.5792336487 -93541.30935618535
835804.7338890679 910458.3417701676 964669.9416298391
-101161.34710605725 -101480.54035828395 -101896.36640027257
910458.3417701676 991779.9678422299 1050833.7172548235
-107184.81706506312 -107523.0161037973 -107963.60175747302
964669.9416298391 1050833.7172548235 1113403.7156669532
```

mean_var_skew_g_v.txt

```
10318.43602424942 10350.993668514679 10393.4078369122
-92866.55843412718 -101161.34710605725 -107184.81706506312
11.128079642821215 17.393174606118098 18.782021852711047
10350.993668514679 10383.654041545966 10426.20203888842
-93159.5792336487 -101480.54035828395 -107523.0161037973
11.163191946421778 17.44805509286422 18.841284553436797
10393.4078369122 10426.20203888842 10468.92438064476
-93541.30935618535 -101896.36640027257 -107963.60175747302
11.208934173519221 17.519550136781504 18.9184884858772
-92866.55843412718 -93159.5792336487 -93541.30935618535
835804.7338890679 910458.3417701676 964669.9416298391
-100.15340076548574 -156.53964051509823 -169.03935109901713
-101161.34710605725 -101480.54035828395 -101896.36640027257
910458.3417701676 991779.9678422299 1050833.7172548235
-109.0990460885446 -170.52167300070687 -184.13785069078477
-107184.81706506312 -107523.0161037973 -107963.60175747302
964669.9416298391 1050833.7172548235 1113403.7156669532
-115.59515201704303 -180.67507846695023 -195.10200689947013
11.128079642821215 11.163191946421778 11.208934173519221
-100.15340076548574 -109.0990460885446 -115.59515201704303
0.012001252539236423 0.01875794275445508 0.020255766914577596
17.393174606118098 17.44805509286422 17.519550136781504
-156.53964051509823 -170.52167300070687 -180.67507846695023
0.01875794275445508 0.02931864113591938 0.03165973843055272
18.782021852711047 18.841284553436797 18.9184884858772
-169.03935109901713 -184.13785069078477 -195.10200689947013
0.020255766914577596 0.03165973843055272 0.03418777264758745
```

mean_var_skew_kurt_g_v.txt

```
10318.43602424942 10350.993668514679 10393.4078369122
-92866.55843412718 -101161.34710605725 -107184.81706506312
11.128079642821215 17.393174606118098 18.782021852711047
-4.203014567432053 -9.441314148284194 -6.194419531760604
10350.993668514679 10383.654041545966 10426.20203888842
-93159.5792336487 -101480.54035828395 -107523.0161037973
11.163191946421778 17.44805509286422 18.841284553436797
-4.216276291670742 -9.471104219833215 -6.21396471351788855
10393.4078369122 10426.20203888842 10468.92438064476
-93541.30935618535 -101896.36640027257 -107963.60175747302
```

```
11.208934173519221 17.519550136781504 18.9184884858772
-4.233552879636337 -9.509912958603135 -6.239427017352259
-92866.55843412718 -93159.5792336487 -93541.30935618535
835804.7338890679 910458.3417701676 964669.9416298391
-100.15340076548574 -156.53964051509823 -169.03935109901713
37.827389442413974 84.97240763872105 55.75015652176705
-101161.34710605725 -101480.54035828395 -101896.36640027257
910458.3417701676 991779.9678422299 1050833.7172548235
-109.0990460885446 -170.52167300070687 -184.13785069078477
41.206110552857496 92.56209520971294 60.72972908881873
-107184.81706506312 -107523.0161037973 -107963.60175747302
964669.9416298391 1050833.7172548235 1113403.7156669532
-115.59515201704303 -180.67507846695023 -195.10200689947013
43.65965408646027 98.0735382241463 64.34577127538192
11.128079642821215 11.163191946421778 11.208934173519221
-100.15340076548574 -109.0990460885446 -115.59515201704303
0.012001252539236423 0.01875794275445508 0.020255766914577596
-0.004532807175079993 -0.010182133758264374 -0.006680469184330047
17.393174606118098 17.44805509286422 17.519550136781504
-156.53964051509823 -170.52167300070687 -180.67507846695023
0.01875794275445508 0.02931864113591938 0.03165973843055272
-0.007084771962689115 -0.015914662368056443 -0.010441565005225464
18.782021852711047 18.841284553436797 18.9184884858772
-169.03935109901713 -184.13785069078477 -195.10200689947013
0.020255766914577596 0.03165973843055272 0.03418777264758745
-0.0076504919221529015 -0.017185450220813164 -0.01127532532420298
-4.203014567432053 -4.216276291670742 -4.233552879636337
37.827389442413974 41.206110552857496 43.65965408646027
-0.004532807175079993 -0.007084771962689115 -0.0076504919221529015
0.0017120163765643017 0.0038457360018208137 0.002523176522836396
-9.441314148284194 -9.471104219833215 -9.509912958603135
84.97240763872105 92.56209520971294 98.0735382241463
-0.010182133758264374 -0.015914662368056443 -0.017185450220813164
0.0038457360018208137 0.008638752291249038 0.00566786096538051
-6.194419531760604 -6.2139647135178855 -6.239427017352259
55.75015652176705 60.72972908881873 64.34577127538192
-0.006680469184330047 -0.010441565005225464 -0.01127532532420298
0.002523176522836396 0.00566786096538051 0.0037186675621462143
```

5. Провести декомпозицію кожного каналу кольору кожного зображення з застосуванням методу головних компонент (PCA):

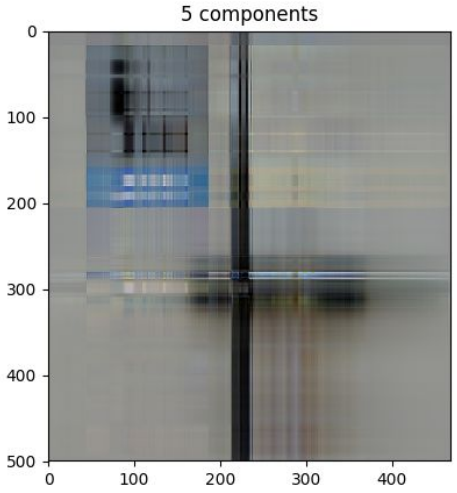
Для проведення декомпозиції скористаємось

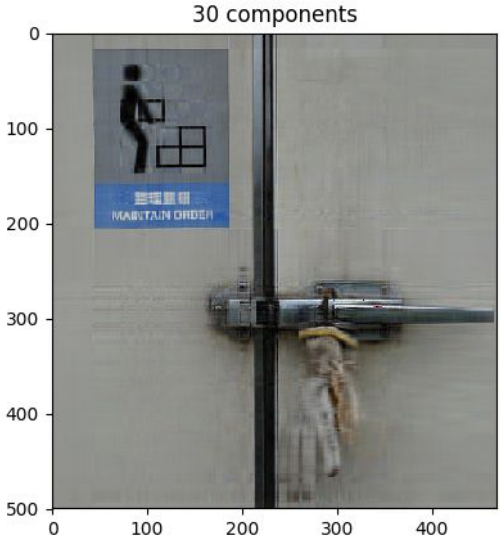
```
def PCA(img_2d, nmpc):
    cov_mat = img_2d - np.mean(img_2d)
    val, vec = np.linalg.eigh(np.cov(cov_mat))
    p = np.size(vec, axis=1)
    i = np.argsort(val)
    i = i[::-1]
    vec = vec[:, i]
    val = val[i]

    if (nmpc < p) or (nmpc > 0):
        vec = vec[:, range(nmpc)]

    score = np.dot(vec.T, cov_mat)
    rcn = np.dot(vec, score) + np.mean(img_2d).T
    rcn_img_mat = np.uint8(np.absolute(rcn))
    return rcn_img_mat

def decompose(files):
    print('>>Decomposing images')
    decomposed_data = {}
    for file_path in tqdm(files):
        img_data = img.imread(file_path)
        values = np.zeros((3, 256))
        for i in range(img_data.shape[0]):
            for j in range(img_data.shape[1]):
                values[0][img_data[i][j][0]] += 1
                values[1][img_data[i][j][1]] += 1
                values[2][img_data[i][j][2]] += 1

        test = img_data
        test_np = np.array(test)
        red = test_np[:, :, 0]
```

```
        green = test_np[:, :, 1]
        blue = test_np[:, :, 2]

        number_of_comp = [5, 30, 80]
        filename = file_path.split('.')[1].split('/')[-1]

        if not
os.path.exists('./output/'+output_dir+'/PCA/'+str(filename)):

os.makedirs('./output/'+output_dir+'/PCA/'+str(filename))

        plt.figure(figsize=(20, 20))
        plt.title('original')
        plt.imshow(img_data)
        plt.savefig('./output/'+output_dir+'/PCA/' +
                    str(filename)+'/original.png')

        for number in number_of_comp:
            red_rcn, green_rcn, blue_rcn = PCA(red, number), PCA(
                green, number), PCA(blue, number)
            rcn_img = np.dstack((red_rcn, green_rcn, blue_rcn))
            plt.figure()
            plt.title(str(number) + ' components')
            plt.imshow(rcn_img)
            plt.savefig('./output/'+output_dir+'/PCA/' +

str(filename)+'/'+str(number)+'_components.png')

        all_mse = list()
        for i in range(100):
            test_r_recon, test_g_recon, test_b_recon = PCA(
                red, i), PCA(green, i), PCA(blue, i)
            rcn_img = np.dstack((test_r_recon, test_g_recon,
test_b_recon))
            all_mse.append(get_mse(test, rcn_img))

        plt.figure()
        plt.plot(range(len(all_mse)), all_mse)
        plt.xlabel("components")
        plt.ylabel("MSE")
```

```
plt.savefig('./output/'+output_dir+'/PCA/'+str(filename)+'/mse.png
')
        decomposed_data[filename] = {'red': red, 'green': green,
'blue': blue}
    return decomposed_data
```

- Варіюючи кількість компонент, провести реконструкцію окремих каналів кольору зображень (від компонент з найбільшою енергією поступово переходячи до компонентів з мінімальною енергією).



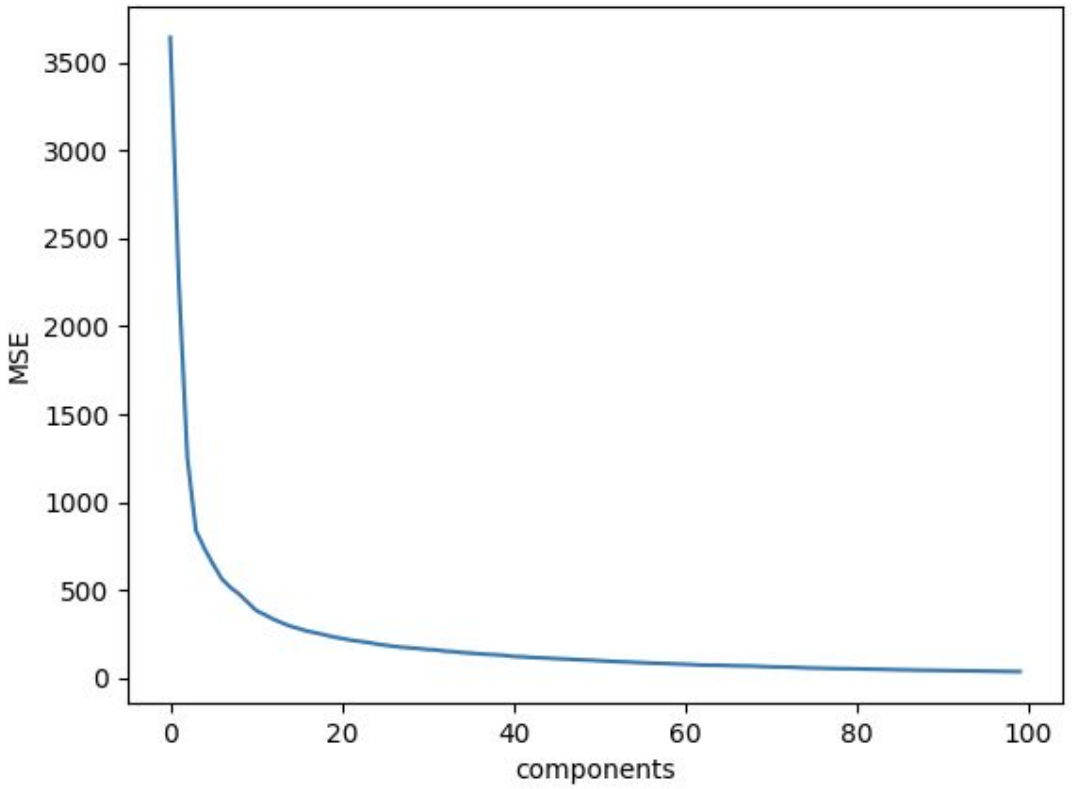



80 components
0
100
200
300
400
500
0
100
200
300
400
整理整頓
MAINTAIN ORDER


original
整理整頓
MAINTAIN ORDER

- Побудувати залежність помилки відновлення (середнє відхилення вихідного зображення відреконструйованого, MSE) від кількості використаних компонент.

```
def get_mse(first_img, second_img):
    err = np.sum((first_img.astype("float") -
second_img.astype("float")) ** 2)
    err /= float(first_img.shape[0] * first_img.shape[1])

    return err
```



6. Провести моделювання окремих каналів кольору зображень з використанням марковських ланцюгів:

- Для кожного каналу кольору кожного зображення розрахувати стохастическую матрицю марковської ланцюга першого і другого порядків (обробка пікселів по горизонталі справа наліво і навпаки, а також по вертикалі зверху вниз і навпаки). У звіті привести явний вигляд однієї марковської ланцюга для одного з каналів кольору тестового зображення;

Розраховуємо схоластичні матриці першого та другого порядків марківського ланцюга для кожного каналу кольору кожного зображення:

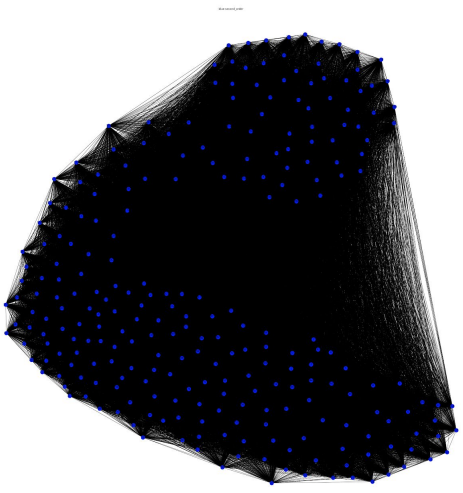
```
print('>>Creating markov chains')
for mean in tqdm(decomposed_data):
   colors = ['red', 'green', 'blue']
   m = 0
   for key in decomposed_data[mean]:
       matrix1 = np.zeros(shape=(256, 256))
       array = decomposed_data[mean][key].flatten('F')
       prev_color = array[0]
       for i in range(len(array) - 1):
           matrix1[array[i]][array[i + 1]] += 1
       matrix = matrix1[0] / sum(matrix1[0])
       for i in range(1, 256):
           matrix = np.vstack((matrix, matrix1[i] / sum(matrix1[i])))
       matrix_2 = np.linalg.matrix_power(matrix, 2)

       output_file_path = './output/' + output_dir + '/markovChain/' + str(mean) + '/' +
colors[m] + '_matrix.txt'
       with open(output_file_path, 'w') as f:
           csv.writer(f, delimiter=' ').writerows(matrix)

       output_file_path = './output/' + output_dir + '/markovChain/' + str(mean) + '/' +
colors[
           m] + '_matrix_2.txt'
       with open(output_file_path, 'w') as f:
           csv.writer(f, delimiter=' ').writerows(matrix_2)
```

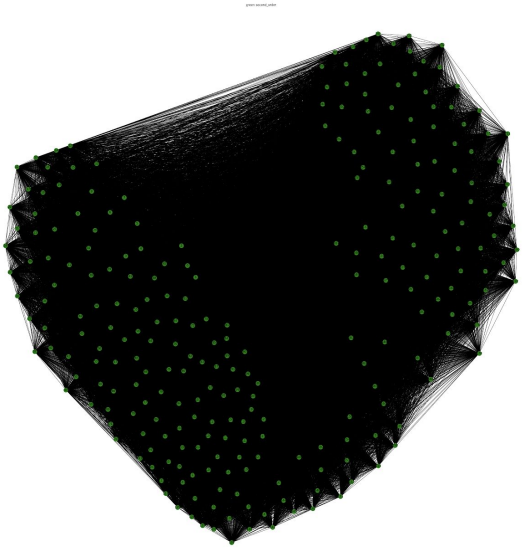
Отримано такі результати:

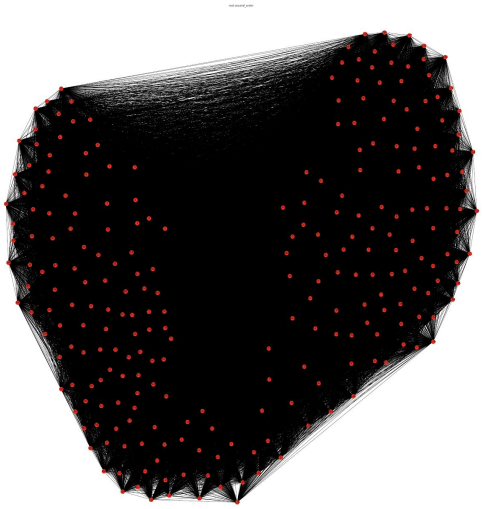
Отримуємо явний вигляд вигляд всіх марковських ланцюгів для каналів кольорів тестових зображень:

```
count = 0
for i in range(256):
   for j in range(256):
       if (mat[i][i] == 1) or (mat[i][j] == 0):
           count += 1
if count == 0:
   file_checks.write('Свойство необратимости выполняется\n')
else:
   file_checks.write('Свойство необратимости не
выполняется\n')
file_checks.close()
data = np.triu(mat) + np.triu(mat).T
index = [str(p) for p in range(data.shape[0])]
dataframe = pd.DataFrame(data, index=index, columns=index)
plt.figure(figsize=(40, 40))
plt.title(str(colors[m]) + ' ' + title)
g = nx.from_pandas_adjacency(dataframe)
nx.draw(g, with_labels=True)
if not os.path.exists('./output/' + output_dir + '/markovChain/' + str(mean)):
   os.makedirs('./output/' + output_dir + '/markovChain/' + str(mean))
plt.savefig('./output/' + output_dir + '/markovChain/' + str(mean) + '/' +
str(colors[m]) + '.png')
i += 1
```

Отримано такі результати:







- Перевірити властивість регулярності, реккурентное і незворотності (irreducible) для отриманих марковских моделей для 5 ітерацій.

Перевіряємо властивості регулярності, рекурентності, незворотності для отриманих матриць першого та другого порядків кожного каналу кольору кожного зображення:

```
i = 0
matrix_list = [matrix, matrix_2]
for mat in matrix_list:
    if i == 0:
        title = 'first_order'
    else:
        title = 'second_order'
    file_checks = open('./output/' + output_dir + '/markovChain/' + str(mean) + '/' +
colors[m] + '_' + str(
        title) + '.txt', "w+")
    file_checks.write('Проверка свойств матрицы ' + str(title) + ' ' +
colors[m] + '')
    count = 0
    for k in range(6):
        reg = matrix_power(mat, k)
        for i in range(256):
            if reg[i][i] == 0:
                count += 1
    if count == 0:
        file_checks.write('Свойство регулярности выполняется\n')
        file_checks.write('Свойство рекуррентности
выполняется\n')
    else:
        file_checks.write('Свойство регулярности не
выполняется\n')
        file_checks.write('Свойство рекуррентности не
выполняется\n')
```

Отримано такі результати:

```
Проверка свойств матрицы first_order red
Свойство регулярности не выполняется
Свойство рекуррентности не выполняется
Свойство необратимости не выполняется
```