МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НТУУ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Лабораторна робота №2

з дисципліни

«*Автоматизація обробки ІзОД*»

Варіант 4

**Виконав:**

студент 5 курсу ФТІ

групи ФЕ-91мп

Карнаух М.Ю.

**Перевірив:**

Прогонов Д. О.

КИЇВ 2020

**І. 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:
   1. Математичне сподівання і дисперсію;
   2. Коефіцієнти асиметрії та ексцесу (нормалізований);
3. Побудувати вектори параметрів зображень, що складаються з:
4. Математичних очікувань значень яскравості для кожного каналу кольору;
5. Математичних очікувань і дисперсії значень яскравості для кожного каналу кольору;
6. Математичних очікувань, дисперсії і коефіцієнта асиметрії значень яскравості для кожного каналу кольору;
7. Математичних очікувань, дисперсії, коефіцієнтів асиметрії та ексцесу значень яскравості для кожного каналу кольору;
8. Побудувати гаусові моделі зображень з використанням розрахованих раніше параметрів.
9. Провести декомпозицію кожного каналу кольору кожного зображення з застосуванням методу головних компонент (PCA):
   1. Варіюючи кількість компонент, провести реконструкцію окремих каналів кольору зображень (від компонент з найбільшою енергією поступово переходячи до компонентів з мінімальною енергією).
   2. Побудувати залежність помилки відновлення (середнє відхилення вихідного зображення відреконструйованого, MSE) від кількості використаних компонент.
10. Провести моделювання окремих каналів кольору зображень з використанням марковських ланцюгів:
11. 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;
12. Перевірити властивість регулярності, реккурентное і незворотності (irreducible) для отриманих марковских моделей для 5 ітерацій.

**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

1. **Знаходження статистичних даних**
2. **Математичне сподівання і дисперсія**

**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 115.07050075657062

Dispersia 6239.145360537583

green

Matematuchne spodivannia 106.90361208116828

Dispersia 5658.206235152412

blue

Matematuchne spodivannia 99.34551353618006

Dispersia 6078.493979325842

**b. Коефіцієнти асиметрії та 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

asymmetries[kanal] = asymm

excesses[kanal] = excess

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

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

Результат:

red

Asymmetry 0.17200205132957633

Excess -1.1999833213649318

green

Asymmetry 0.3091704854987942

Excess -1.0475381779994972

blue

Asymmetry 0.43800578130010975

Excess -1.039857267449018

**3-4. Побудувати вектори параметрів зображень. Побудувати гаусові моделі зображень з використанням розрахованих раніше параметрів.**

**def** get\_vector(kanal):

vector = np.array([matspodivannia[kanal], dispersion[kanal], asymmetries[kanal], excesses[kanal]])

**return** vector

**def** get\_all\_vectors(vector, kanal):

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

img = np.reshape(img, (-1, 3))

img = np.swapaxes(img, 0, 1)

vector = np.concatenate((np.copy(vector),img),axis=1)

x = np.random.normal(matspodivannia[kanal], dispersion[kanal], 1)

print(kanal + '**\n**')

print('Matematuchne spodivannia**\n**' + str(x))

print('Matematuchne spodivannia + dispersia**\n**' + str(np.cov(vector)[:2, :2]))

print('Matematuchne spodivannia + dispersia + asymmetry**\n**' + str(np.cov(vector)[:3, :3]))

print('Matematuchne spodivannia + dispersia + asymmetry + excess**\n**' + str(np.cov(vector)[:4, :4]) + '**\n**')

Результат:

red

Matematuchne spodivannia

[3103.64015788]

Matematuchne spodivannia + dispersia

[[6239.97763909 5110.67693039]

[5110.67693039 5658.89009955]]

Matematuchne spodivannia + dispersia + asymmetry

[[6239.97763909 5110.67693039 4341.68392504]

[5110.67693039 5658.89009955 5226.26531538]

[4341.68392504 5226.26531538 6079.28719898]]

Matematuchne spodivannia + dispersia + asymmetry + excess

[[6239.97763909 5110.67693039 4341.68392504]

[5110.67693039 5658.89009955 5226.26531538]

[4341.68392504 5226.26531538 6079.28719898]]

green

Matematuchne spodivannia

[-8690.28703728]

Matematuchne spodivannia + dispersia

[[6239.97763909 5110.67693039]

[5110.67693039 5658.89009955]]

Matematuchne spodivannia + dispersia + asymmetry

[[6239.97763909 5110.67693039 4341.68392504]

[5110.67693039 5658.89009955 5226.26531538]

[4341.68392504 5226.26531538 6079.28719898]]

Matematuchne spodivannia + dispersia + asymmetry + excess

[[6239.97763909 5110.67693039 4341.68392504]

[5110.67693039 5658.89009955 5226.26531538]

[4341.68392504 5226.26531538 6079.28719898]]

blue

Matematuchne spodivannia

[8428.29904076]

Matematuchne spodivannia + dispersia

[[6239.97763909 5110.67693039]

[5110.67693039 5658.89009955]]

Matematuchne spodivannia + dispersia + asymmetry

[[6239.97763909 5110.67693039 4341.68392504]

[5110.67693039 5658.89009955 5226.26531538]

[4341.68392504 5226.26531538 6079.28719898]]

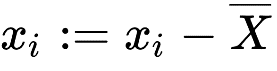
Matematuchne spodivannia + dispersia + asymmetry + excess

[[6239.97763909 5110.67693039 4341.68392504]

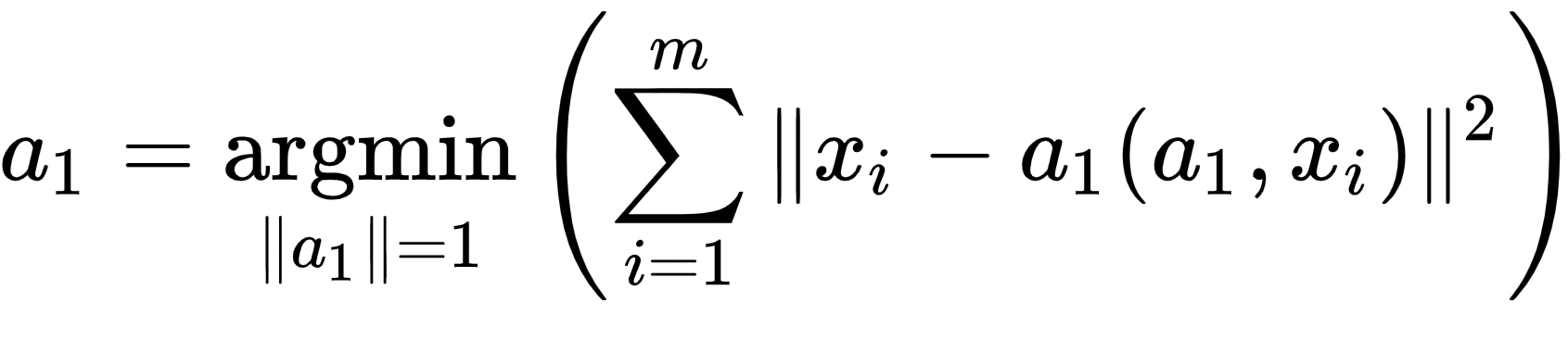
[5110.67693039 5658.89009955 5226.26531538]

[4341.68392504 5226.26531538 6079.28719898]]

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

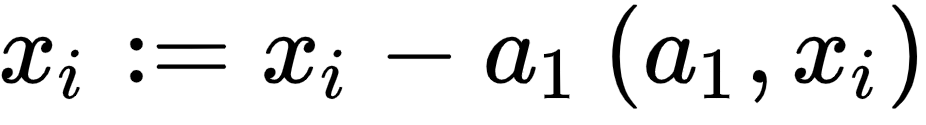
Для проведення декомпозиції каналів кольору сформуємо функцію що працюваниме по методу головних компонент (PCA). Алгоритм даної функції буде наступним:

1. Централізувати дані (відніманням середнього)

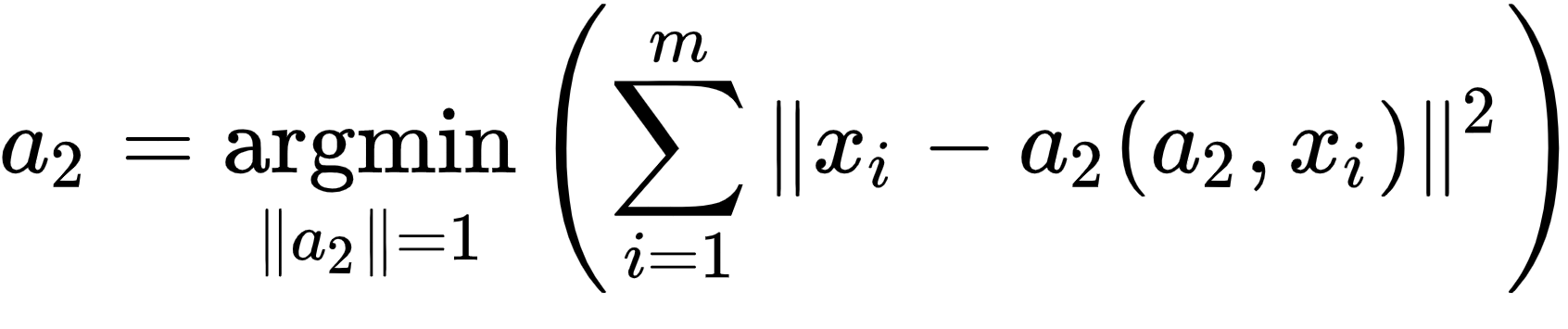
2. Знайти першу головну компоненту як рішення задачі: 

якщо рішення не одне, то здійснюється вибір одного з них.

3. З даних відняти проекція на першу головну компоненту:



4. Відшукати другу головну компоненту як рішення задачі:



якщо рішення не одне, то здійснюється вибір одного з них.

**def** get\_pca(img\_2d, nmpc):

cov\_mat = img\_2d - np.mean(img\_2d)

value, vec = np.linalg.eigh(np.cov(cov\_mat))

p = np.size(vec, axis =1)

ind = np.argsort(value)

ind = ind[::-1]

vec = vec[:, ind]

value = value[ind]

**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** show\_pca():

test = np.array(sample[3])

red = test[:,:,0]

green = test[:,:,1]

blue = test[:,:,2]

numbers = [3, 30,90]

fig = plt.figure(figsize=(20,20))

fig.add\_subplot(1, 2, 1)

plt.imshow(sample[3])

m = 2

**for** num **in** numbers:

red\_rcn, green\_rcn, blue\_rcn = get\_pca(red, num), get\_pca(green, num), get\_pca(blue, num)

test\_rcn = np.dstack((red\_rcn, green\_rcn, blue\_rcn))

fig.add\_subplot(3, 2, m)

plt.imshow(test\_rcn)

m += 2

**def** my\_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

**def** show

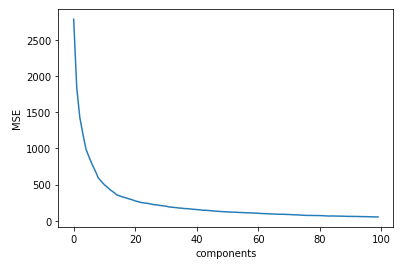
Передавши параметри тестової фотографії отримаємо наступний результат, який зображено на Рисунку 1.



Рисунок 1 **–** відновлені фото з різної кількості компонентів

Як бачимо, зі збільшенням кількості компонентів росте якість відновлення зображення. Також можемо спостерігати «артефакти» відновлення на частинах з різким кольоровим зсувом та контрастом. Це спричинено високою кількістю інформації, що потрібна для відновлення таких частин.

Після цього виконаємо функцію для багатьох кроків та порівняємо початкове фото з відновленим за допомогою функції середньої квадратичної похибки.

**def** get\_mse(img1, img2):

e = np.sum((img1.astype("float") - img2.astype("float")) \*\* 2)

e /= float(img1.shape[0] \* img1.shape[1])

**return** e

**def** show\_mse():

test = np.array(sample[4])

red = test[:,:,0]

green = test[:,:,1]

blue = test[:,:,2]

mse = list()

**for** i **in** range(100):

red\_rcn, green\_rcn, blue\_rcn = get\_pca(red, i), get\_pca(green, i), get\_pca(blue, i)

rcn\_img = np.dstack((red\_rcn, green\_rcn, blue\_rcn))

mse.append(get\_mse(sample[4], rcn\_img))

plt.plot(range(len(mse)), mse)

plt.xlabel('components')

plt.ylabel('MSE')

plt.show()

Отримаємо наступний графік:

Рисунок 2 – залежність MSE відновлених фото від кількості компонент

З даного графіка видно що залежність має експоненціальний характер та похибка дуже значно зменшується при збільшенні компонентів.

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

Для кожного каналу кольору кожного зображення розрахувати стохастичну матрицю марковського ланцюга першого і другого порядків;

Для побудови марківського ланцюга сформуємо матрицю 256 на 256 та пройшовши по всіх пікселях картинки запишемо кількість переходів між ними.

**def** markov\_matrix():

test = np.array(sample[4])

red = test[:,:,0]

green = test[:,:,1]

blue = test[:,:,2]

kanals\_list = [red, green, blue]

kanals = ['red', 'green', 'blue']

k = 0

**for** kanal **in** kanals\_list:

matrix1 = np.zeros(shape=(256, 256))

massif = kanal.flatten('F')

prev\_color = massif[0]

**for** i **in** range(len(massif) - 1):

matrix1[massif[i]][massif[i + 1]] += 1

matrix = matrix1[0] / sum(matrix1[0])

**for** i **in** range(1, 256):

matrix = np.vstack((matrix, matrix1[i] / sum(matrix1[i])))

print(f'matrix **{kanals[k]}** 1st**\n**', matrix)

print(f'**\n**matrix for **{kanals[k]}** 2d**\n**', np.linalg.matrix\_power(matrix, 2))

k +=1

Результат:  
matrix red 1st

[[0.94040068 0.03075755 0.00978915 ... 0. 0. 0. ]

[0.36006031 0.359375 0.12870066 ... 0. 0. 0. ]

[0.21575758 0.20848485 0.23587879 ... 0. 0. 0. ]

...

[0. 0. 0. ... 0. 0.25 0.25 ]

[0. 0. 0. ... 0. 0. 0.125 ]

[0. 0. 0. ... 0. 0.08333333 0.16666667]]

matrix for red 2d

[[8.98927835e-01 4.31612271e-02 1.69429376e-02 ... 1.63760196e-07

2.20061340e-07 0.00000000e+00]

[5.09664058e-01 1.78654789e-01 9.51554689e-02 ... 0.00000000e+00

1.72269168e-06 9.01719760e-07]

[3.63304354e-01 1.59617210e-01 1.22394568e-01 ... 0.00000000e+00

4.32900433e-06 0.00000000e+00]

...

[0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00

3.75000000e-02 8.95833333e-02]

[5.88235294e-04 2.94117647e-04 2.94117647e-04 ... 0.00000000e+00

1.04166667e-02 3.33333333e-02]

[0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 6.94444444e-03

2.63888889e-02 5.76388889e-02]]

matrix green 1st

[[0.93865589 0.03229497 0.01072996 ... 0. 0. 0. ]

[0.37491634 0.35202784 0.1272922 ... 0. 0. 0. ]

[0.21562083 0.21139297 0.23631509 ... 0. 0. 0. ]

...

[0. 0. 0. ... 0. 0. 0.6 ]

[0. 0. 0. ... 0. 0. 0. ]

[0. 0. 0. ... 0.02941176 0. 0.20588235]]

matrix for green 2d

[[8.96666004e-01 4.49555843e-02 1.82069019e-02 ... 0.00000000e+00

1.78254899e-07 5.69704029e-06]

[5.22533662e-01 1.72692424e-01 9.34030805e-02 ... 0.00000000e+00

2.04352504e-06 3.02675789e-05]

[3.61096102e-01 1.56264299e-01 1.22298868e-01 ... 0.00000000e+00

0.00000000e+00 1.82692048e-05]

...

[0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 1.76470588e-02

0.00000000e+00 1.23529412e-01]

[9.50883498e-03 3.70370370e-03 0.00000000e+00 ... 0.00000000e+00

0.00000000e+00 0.00000000e+00]

[1.15688706e-03 2.34356691e-04 3.00630716e-04 ... 1.27080244e-02

2.10084034e-03 7.50954334e-02]]

matrix blue 1st

[[0.91584264 0.02661204 0.01744206 ... 0. 0. 0. ]

[0.38917062 0.27106167 0.12236218 ... 0. 0. 0. ]

[0.27533526 0.12103208 0.22627737 ... 0. 0. 0. ]

...

[0. 0. 0. ... 0.125 0. 0.125 ]

[0. 0. 0. ... 0.33333333 0. 0.33333333]

[0. 0. 0. ... 0.04347826 0. 0.2173913 ]]

matrix for blue 2d

[[8.58570688e-01 3.56845473e-02 2.65629644e-02 ... 1.10762905e-07

0.00000000e+00 8.34981966e-06]

[5.24089434e-01 1.11033435e-01 8.87958796e-02 ... 1.31216343e-05

0.00000000e+00 1.90951810e-05]

[4.10190340e-01 8.86154930e-02 1.06766408e-01 ... 0.00000000e+00

0.00000000e+00 4.62173912e-05]

...

[1.17535651e-02 0.00000000e+00 1.02654698e-02 ... 2.10597826e-02

0.00000000e+00 7.40489130e-02]

[0.00000000e+00 0.00000000e+00 2.56410256e-02 ... 5.61594203e-02

0.00000000e+00 1.14130435e-01]

[1.85137909e-03 0.00000000e+00 1.46886016e-04 ... 1.48865784e-02

8.30039526e-03 8.79177769e-02]]

**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.

Було побудовано вектори даних та гаусові моделі для одновимірного та багатовимірних варіантів в залежності від кількості даних.

За допомогою методу головних компонент було відновлено тестові зображення та показано, що при збільшенні кількості компонент зростає якість відновлення.

Зібравши дані, було побудовано графік залежності середньої квадратичної похибки відновлених зображень від кількості компонент. Було помічено експоненціальну залежність, що свідчить про значні зміни при невеликих кількостях компонент та майже непомітні при великих значеннях.

Було проведено моделювання окремих каналів кольору зображень з використанням марковських ланцюгів, та сформовано стохастичні матриці за різними типами обходів.