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: Виконав:

Прогонов Дмитро Олександрович студент 5 курсу групи ФЕ-91мп

Захищено з оцінкою Павлусь Олександр Сергійович

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

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

дата, підпис

Київ – 2020 р.

Завдання лабораторної роботи:

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:
   1. Математичних очікувань значень яскравості для кожного каналу кольору;
   2. Математичних очікувань і дисперсії значень яскравості для кожного каналу кольору;
   3. Математичних очікувань, дисперсії і коефіцієнта асиметрії значень яскравості для кожного каналу кольору;
   4. 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):
   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.
6. Провести моделювання окремих каналів кольору зображень з використанням марковських ланцюгів:
   1. 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;
   2. Перевірити властивість регулярності, реккурентное і незворотності (irreducible) для отриманих марковских моделей для 5 ітерацій.

1) Для кожного каналу сформувати вектори параметрів зображень, що складаються з:

1. Математичних очікувань значень яскравості для кожного каналу кольору;  
 2. Математичних очікувань і дисперсії значень яскравості для кожного каналу кольору;  
 3. Математичних очікувань, дисперсії і коефіцієнта асиметрії значень яскравості для кожного каналу кольору;  
 4. Математичних очікувань, дисперсії, коефіцієнтів асиметрії та ексцесу значень яскравості для кожного каналу кольору;

Та побудувати гаусові моделі зображень з використанням розрахованих раніше параметрів.

**Матрица ковариации для векторов М (Розмірність 3х3):**

[[ 1558.8314113 1217.98109914 867.39932783]

[ 1217.98109914 1348.35620224 1343.21979526]

[ 867.39932783 1343.21979526 2153.74912621]]

**Матрица ковариации для векторов М и D (Розмірність 6х6):**

[[ 1.55883141e+03 1.21798110e+03 8.67399328e+02 1.69804436e+04

3.22434010e+04 3.29201492e+04]

[ 1.21798110e+03 1.34835620e+03 1.34321980e+03 -5.97887225e+02

2.53843959e+04 3.35534374e+04]

[ 8.67399328e+02 1.34321980e+03 2.15374913e+03 -1.04468525e+03

3.16366091e+04 5.22991471e+04]

[ 1.69804436e+04 -5.97887225e+02 -1.04468525e+03 6.91492773e+06

4.94826144e+06 4.09353161e+06]

[ 3.22434010e+04 2.53843959e+04 3.16366091e+04 4.94826144e+06

5.46480221e+06 5.53385021e+06]

[ 3.29201492e+04 3.35534374e+04 5.22991471e+04 4.09353161e+06 5.53385021e+06 6.32601606e+06]]

**Матрица ковариации для векторов М, D, Skew (Розмірність 9х9):**

[[ 1.55883141e+03 1.21798110e+03 8.67399328e+02 1.69804436e+04

3.22434010e+04 3.29201492e+04 -2.87986650e+01 -2.10904263e+01

-1.92310422e+01]

[ 1.21798110e+03 1.34835620e+03 1.34321980e+03 -5.97887225e+02

2.53843959e+04 3.35534374e+04 -2.36753292e+01 -2.48948808e+01

-3.41243430e+01]

[ 8.67399328e+02 1.34321980e+03 2.15374913e+03 -1.04468525e+03

3.16366091e+04 5.22991471e+04 -1.95043129e+01 -3.19628811e+01

-5.26524767e+01]

[ 1.69804436e+04 -5.97887225e+02 -1.04468525e+03 6.91492773e+06

4.94826144e+06 4.09353161e+06 -5.62529238e+02 -1.86398464e+02

1.17617256e+02]

[ 3.22434010e+04 2.53843959e+04 3.16366091e+04 4.94826144e+06

5.46480221e+06 5.53385021e+06 -7.39328905e+02 -6.13803900e+02

-8.83026013e+02]

[ 3.29201492e+04 3.35534374e+04 5.22991471e+04 4.09353161e+06

5.53385021e+06 6.32601606e+06 -7.53050098e+02 -7.61215065e+02

-1.42214591e+03]

[ -2.87986650e+01 -2.36753292e+01 -1.95043129e+01 -5.62529238e+02

-7.39328905e+02 -7.53050098e+02 6.13547682e-01 4.28316735e-01

3.97176996e-01]

[ -2.10904263e+01 -2.48948808e+01 -3.19628811e+01 -1.86398464e+02

-6.13803900e+02 -7.61215065e+02 4.28316735e-01 5.94335154e-01

8.13657421e-01]

[ -1.92310422e+01 -3.41243430e+01 -5.26524767e+01 1.17617256e+02

-8.83026013e+02 -1.42214591e+03 3.97176996e-01 8.13657421e-01

1.55961769e+00]]

2) Провести декомпозицію кожного каналу кольору кожного зображення з застосуванням методу головних компонент.

Для аналізу методу головних компонент було обрано таке зображення Im15.jpg:



Рисунок 1 — Зображення для аналізу методу головних компонент

При відновленні даного зображення із 1 головною компонентою, тобто матриця сингулярних чисел має лише 1 сингулярне число отримуємо таку похибку відновлення зображення:

**1 компонента - Ошибка исходной матрици и полученой: 1029.721 %**

**10 компонент - Ошибка исходной матрици и полученой: 431.462 %**

**60 компонент - Ошибка исходной матрици и полученой: 182.963 %**

**150 компонент - Ошибка исходной матрици и полученой: 63.531 %**

**250 компонент - Ошибка исходной матрици и полученой: 14.567 %**

**333(максимум) - Ошибка исходной матрици и полученой 5.62e-11 %**

Відновлене зображення для 3 головних компонент зображене на рисунку 2.



Рисунок 2 — Відновлене зображення при для 3 головних компонент

Покажемо яке буде зображення при нульовій кількості головних компонент:



Рисунок 3 — Відновлене зображення при для 0 головних компонент

3) Провести моделювання окремих каналів кольору зображень з використанням марковських ланцюгів

Для аналізу марковських ланцюгів було обрано Red-канал зображення Im15.jpg

Стохастична матриця L -> R (при проходженні зліва на право) має наступний вигляд:

Розмірність матриці (256х256)

**[[ 0.74889135 0.23059867 0.01552106 ..., 0. 0. 0. ]**

**[ 0.20067883 0.59015698 0.1718286 ..., 0. 0. 0. ]**

**[ 0.01851852 0.21560847 0.48677249 ..., 0. 0. 0. ]**

**...,**

**[ 0. 0. 0. ..., 0.2373247 0.17583603 0.16073355]**

**[ 0. 0. 0. ..., 0.17106549 0.29423265 0.17986315]**

**[ 0. 0. 0. ..., 0.08142494 0.1278626 0.22964377]]**

Сума елементів матриці разділена на кількість рядів = 1.0

ВИСНОВКИ

Отже, в роботі було отримано параметри Гаусової моделі, а саме матриці коваріації для 4 випадків, коли вектор моделі складався із векторів:

1. Математичних очікувань значень яскравості для кожного каналу кольору;  
 2. Математичних очікувань і дисперсії значень яскравості для кожного каналу кольору;  
 3. Математичних очікувань, дисперсії і коефіцієнта асиметрії значень яскравості для кожного каналу кольору;  
 4. Математичних очікувань, дисперсії, коефіцієнтів асиметрії та ексцесу значень яскравості для кожного каналу кольору;

Було проведено декомпозицію зображення за допомогою метода головних компонент (РСА) для різної кількості компонент. При нульовій кількості головних компонент можемо спостерігати повну втрату Red-каналу, в результаті чого зображення набуває зеленого відтінку. Також було розраховано похибки відтворення вихідного зображення при різній кількості головних компонент.

Для обраного каналу кольору та обраного зображення був отриманий марковський ланцюг 1 порядку. Розмірність стохастичної матриці становить 256х256.

Лістинг програми на мові програмування Python

from math import sqrt, pi, exp

import numpy as np

import scipy.stats as st

from scipy import linalg

from pylab import \*

from PIL import Image, ImageDraw

MEAN\_VECTOR\_R = []

MEAN\_VECTOR\_G = []

MEAN\_VECTOR\_B = []

VAR\_VECTOR\_R = []

VAR\_VECTOR\_G = []

VAR\_VECTOR\_B = []

SKEW\_VECTOR\_R = []

SKEW\_VECTOR\_G = []

SKEW\_VECTOR\_B = []

KURT\_VECTOR\_R = []

KURT\_VECTOR\_G = []

KURT\_VECTOR\_B = []

def get\_array(n):

array = [[ 0 for i in range(n) ] for i in range(n) ]

return array

def get\_RGB(image):

width = image.size[0] #Определяем ширину.

height = image.size[1] #Определяем высоту.

pix = image.load() #Выгружаем значения пикселей.

RGB = [[], [], [] ]

for i in range(width):

for j in range(height):

a = pix[i, j][0]

b = pix[i, j][1]

c = pix[i, j][2]

RGB[0].append(a)

RGB[1].append(b)

RGB[2].append(c)

return RGB

def descript(x):

result = []

result.append(np.mean(x)) # среднее

result.append(np.var(x)) # дисперсия

result.append(st.skew(x)) # асимметрия

result.append(st.kurtosis(x)) # эксцесс

return tuple(result)

def main():

# --------Gauss------------

print('------------------Task ------------------')

for j in range(1,21):

image = Image.open('Pictures/im{}.jpg'.format(j))

RGB = get\_RGB(image)

mean0, var0, skew0, kurt0 = descript(RGB[0])

mean1, var1, skew1, kurt1 = descript(RGB[1])

mean2, var2, skew2, kurt2 = descript(RGB[2])

MEAN\_VECTOR\_R.append(mean0)

MEAN\_VECTOR\_G.append(mean1)

MEAN\_VECTOR\_B.append(mean2)

VAR\_VECTOR\_R.append(var0)

VAR\_VECTOR\_G.append(var1)

VAR\_VECTOR\_B.append(var2)

SKEW\_VECTOR\_R.append(skew0)

SKEW\_VECTOR\_G.append(skew1)

SKEW\_VECTOR\_B.append(skew2)

KURT\_VECTOR\_R.append(kurt0)

KURT\_VECTOR\_G.append(kurt1)

KURT\_VECTOR\_B.append(kurt2)

M = np.cov(np.vstack((MEAN\_VECTOR\_R, MEAN\_VECTOR\_G, MEAN\_VECTOR\_B)))

M\_D = np.cov(np.vstack((MEAN\_VECTOR\_R, MEAN\_VECTOR\_G, MEAN\_VECTOR\_B, VAR\_VECTOR\_R, VAR\_VECTOR\_G, VAR\_VECTOR\_B)))

M\_D\_Skew = np.cov(np.vstack((MEAN\_VECTOR\_R, MEAN\_VECTOR\_G, MEAN\_VECTOR\_B, VAR\_VECTOR\_R, VAR\_VECTOR\_G, VAR\_VECTOR\_B, SKEW\_VECTOR\_R, SKEW\_VECTOR\_G, SKEW\_VECTOR\_B)))

M\_D\_Skew\_Kurt = np.cov(np.vstack((MEAN\_VECTOR\_R, MEAN\_VECTOR\_G, MEAN\_VECTOR\_B, VAR\_VECTOR\_R, VAR\_VECTOR\_G, VAR\_VECTOR\_B, SKEW\_VECTOR\_R, SKEW\_VECTOR\_G, SKEW\_VECTOR\_B, KURT\_VECTOR\_R, KURT\_VECTOR\_G, KURT\_VECTOR\_B)))

print('Матрица ковариации для векторов М: ')

print(M)

print()

print('Матрица ковариации для векторов М и D: ')

print(M\_D)

print()

print('Матрица ковариации для векторов М, D, Skew: ')

print(M\_D\_Skew)

print()

print('Матрица ковариации для векторов М, D, Skew, Kurt: ')

print(M\_D\_Skew\_Kurt)

print()

print('------------------Task ------------------')

image = Image.open("Pictures/im15.jpg")

width = image.size[0] # 500

height = image.size[1] # 333

draw = ImageDraw.Draw(image)

pix = image.load()

R\_ch\_RGB = [[0 for i in range(height)] for i in range(width)] # Создаем пустой список нужного размера

for i in range(width):

for j in range(height):

R\_ch\_RGB[i][j] = pix[i,j][0]

U, S, Vt = linalg.svd(R\_ch\_RGB)

#restored\_matrix = np.dot(U, S, Vt)

S\_add = np.zeros((width,height)) # Создаем пустую матрицу для сингулярных чисел

for i in range(width):

for j in range(height):

if i==j:

S\_add[i][j] = S[i]

number = 0

for i in range(number,width):

for j in range(number,height):

S\_add[i][j] = 0

US = np.dot(U, S\_add)

restored\_matrix = np.dot(US, Vt)

# Восстанавливаем картинку

for i in range(width):

for j in range(height):

a = int(restored\_matrix[i][j])

b = pix[i, j][1]

c = pix[i, j][2]

draw.point((i, j), (a, b, c))

image.save("Restored.jpg", "JPEG")

del draw

#print( [restored\_matrix[i][j] for i in range(3) for j in range(3)] )

#print( [R\_ch\_RGB[i][j] for i in range(3) for j in range(3)] )

total\_error = 0

for i in range(width):

for j in range(height):

diff = abs(R\_ch\_RGB[i][j] - restored\_matrix[i][j])

total\_error += diff

EPS = total\_error/(width\*height)\*100

print('Ошибка исходной матрици и полученой: ', EPS, '%')

print()

print('------------------Task ------------------')

# Стохастическая матрица слева направо

Stochastic\_matrix = np.zeros((256,256))

suma\_row = 0

sum1 = 0

print('Стохастическая матрица L -> R: ')

for i in range(width):

for j in range(height-1):

k1 = R\_ch\_RGB[i][j]

k2 = R\_ch\_RGB[i][j+1]

Stochastic\_matrix[k1][k2] += 1

# Преобразование в нормальный вид

for i in range(256):

for j in range(256):

suma\_row += Stochastic\_matrix[i][j]

Stochastic\_matrix[i] /= suma\_row

suma\_row = 0

print(Stochastic\_matrix)

print()

# Сума элементов разделенная на количество элементов

for i in range(256):

for j in range(256):

sum1 += Stochastic\_matrix[i][j]

print('Сума элементов разделенная на количество строк = ', sum1/(256))

print()

# Стохастическая матрица с права на лево

Stochastic\_matrix = np.zeros((256,256))

suma\_row = 0

sum1 = 0

print('Стохастическая матрица R -> L: ')

for i in range(width):

for j in range(height-1):

k1 = R\_ch\_RGB[i][height-j-2]

k2 = R\_ch\_RGB[i][height-j-1]

Stochastic\_matrix[k1][k2] += 1

# Преобразование в нормальный вид

for i in range(256):

for j in range(256):

suma\_row += Stochastic\_matrix[i][j]

Stochastic\_matrix[i] /= suma\_row

suma\_row = 0

print(Stochastic\_matrix)

# Сума элементов разделенная на количество элементов

for i in range(256):

for j in range(256):

sum1 += Stochastic\_matrix[i][j]

print('Сума элементов разделенная на количество строк = ', sum1/(256))

if \_\_name\_\_ == '\_\_main\_\_':

main()