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.

Хід роботи

1) Сформувати тестову вибірку зображень з вихідного пакета;

"from os import listdir\n",

"from matplotlib import image\n",

"import random\n",

"import numpy as np \n",

"from dataclasses import dataclass\n",

"\n",

"# load images in a directory \n",

"random.seed(8)\n",

"random\_indexes = random.sample(range(25000), 250)\n",

"loaded\_images = list()\n",

"for i in range(250):\n",

"\t# load image\n",

"\tfilename = 'im' + str(random\_indexes[i]) + '.jpg'\n",

"\timg\_data = image.imread('Q:/mirflickr25k/mirflickr/' + filename)\n",

"\t# store loaded image\n",

"\tloaded\_images.append(img\_data)\n",

"\tprint('> loaded %s %s' % (filename, img\_data.shape))"

2) Для кожного каналу кольору кожного зображення з тестового пакета обчислити наступні характеристики: Математичне сподівання і дисперсію; Коефіцієнти асиметрії та ексцесу (нормалізований);

"RED = 0\n",

"GREEN = 1\n",

"BLUE = 2\n",

"# Expected value & Dispersiya\n",

"#RED\n",

"sum\_val = sum(values[RED])\n",

"M\_red = 0\n",

"for index in range(len(values[RED])):\n",

" p = (values[RED][index] / sum\_val)\n",

" M\_red += p \* index\n",

"D\_red = 0\n",

"for index in range(len(values[RED])):\n",

" p = (values[RED][index] / sum\_val)\n",

" D\_red += p \* ((index - M\_red) \*\* 2)\n",

"print(\"Red: Expected value - {0:.2f}, Dispersiya - {1:.2f}\"\\\n",

".format(M\_red, D\_red))\n",

"\n",

"#GREEN\n",

"sum\_val = sum(values[GREEN])\n",

"M\_green = 0\n",

"for index in range(len(values[GREEN])):\n",

" p = (values[GREEN][index] / sum\_val)\n",

" M\_green += p \* index\n",

"D\_green = 0\n",

"for index in range(len(values[GREEN])):\n",

" p = (values[GREEN][index] / sum\_val)\n",

" D\_green += p \* ((index - M\_green) \*\* 2)\n",

"print(\"Green: Expected value - {0:.2f}, Dispersiya - {1:.2f}\"\\\n",

".format(M\_green, D\_green))\n",

" \n",

"#BLUE\n",

"sum\_val = sum(values[BLUE])\n",

"M\_blue = 0\n",

"for index in range(len(values[BLUE])):\n",

" p = (values[BLUE][index] / sum\_val)\n",

" M\_blue += p \* index\n",

"D\_blue = 0\n",

"for index in range(len(values[BLUE])):\n",

" p = (values[BLUE][index] / sum\_val)\n",

" D\_blue += p \* ((index - M\_blue) \*\* 2)\n",

"print(\"Blue: Expected value - {0:.2f}, Dispersiya - {1:.2f}\"\\\n",

".format(M\_blue, D\_blue))"

Red: Expected value - 112.49, Dispersiya - 6353.01,

Green: Expected value - 104.34, Dispersiya - 5756.84,

Blue: Expected value - 93.01, Dispersiya - 5873.88

"#Asymmetry and excess\n",

"Asym\_red = E\_operator(values[RED], M\_red, 3) / (D\_red \*\* (3 / 2))\n",

"Asym\_green = E\_operator(values[GREEN], M\_green, 3) / (D\_green \*\* (3 / 2))\n",

"Asym\_blue = E\_operator(values[BLUE], M\_blue, 3) / (D\_blue \*\* (3 / 2))\n",

"\n",

"Excess\_red = E\_operator(values[RED], M\_red, 4) / (D\_red \*\* 2)\n",

"Excess\_green = E\_operator(values[GREEN], M\_green, 4) / (D\_green \*\* 2)\n",

"Excess\_blue = E\_operator(values[BLUE], M\_blue, 4) / (D\_blue \*\* 2)\n",

"print('Red: Asymmetry - {0:.3f}, Excess - {1:.3f}'\\\n",

".format(Asym\_red, Excess\_red))\n",

"print('Green: Asymmetry - {0:.3f}, Excess - {1:.3f}'\\\n",

".format(Asym\_green, Excess\_green))\n",

"print('Blue: Asymmetry - {0:.3f}, Excess - {1:.3f}'\\\n",

".format(Asym\_blue, Excess\_blue))"

Red: Asymmetry - 0.206, Excess - 1.803,

Green: Asymmetry - 0.330, Excess - 1.964,

Blue: Asymmetry - 0.549, Excess - 2.135

3) Побудувати вектори параметрів зображень, що складаються з:

* + Математичних очікувань значень яскравості для кожного каналу кольору;
  + Математичних очікувань і дисперсії значень яскравості для кожного каналу кольору;
  + Математичних очікувань, дисперсії і коефіцієнта асиметрії значень яскравості для кожного каналу кольору;
  + Математичних очікувань, дисперсії, коефіцієнтів асиметрії та ексцесу значень яскравості для кожного каналу кольору;

Код програми:

Vector\_A = np.array([np.array([M\_red, D\_red, Asym\_red, Ekscess\_red]),

np.array([M\_green, D\_green, Asym\_green, Ekscess\_green]),

np.array([M\_blue, D\_blue, Asym\_blue, Ekscess\_blue])])

print("Vector\_A:\n" + str(Vector\_A))

Vector\_All\_DATA = np.copy(Vector\_A)

for image in loaded\_images:

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

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

Vector\_All\_DATA = np.concatenate((Vector\_All\_DATA,image),axis=1)

Vector\_All\_DATA – містить дані трьох каналів

Vector\_A:

[[1.12491758e+02 6.35301040e+03 2.06077204e-01 1.80325517e+00],

[1.04342507e+02 5.75684362e+03 3.30208537e-01 1.96430324e+00],

[9.30133830e+01 5.87388153e+03 5.48968596e-01 2.13480087e+00]],

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

"#4. Gaussian models\n",

"\n",

"#a Expected value\n",

"P\_x1 = np.random.normal(M\_red, D\_red, 1)\n",

"print(\"Expected value + colors:\\n\" + str(P\_x1))\n",

"\n",

"#b Expected value and dispersion\n",

"P\_x2 = np.cov(Vector\_All\_DATA)\n",

"print(\"Expected value + dispersion + colors:\\n\" + str(P\_x2[:2, :2]))\n",

"\n",

"#c Expected value, dispersion and asymetry\n",

"print(\"Expected value + dispersion + asymetry + colors:\\n\" + str(P\_x2[:3, :3]))\n",

"\n",

"#d Expected value, dispersion, asymetry and excess\n",

"print(\"Expected value + dispersion + asymetry + excess + colors:\\n\" + str(P\_x2[:4, :4]))"

Expected value + colors:

[-3409.32788175],

Expected value + dispersion + colors:

[[6353.86906855 5313.95741987]

[5313.95741987 5757.54808175]]

Expected value + dispersion + asymetry + colors:

[[6353.86906855 5313.95741987 4726.35131306]

[5313.95741987 5757.54808175 5238.69940006]

[4726.35131306 5238.69940006 5874.61823923]]

Expected value + dispersion + asymetry + excess + colors:

[[6353.86906855 5313.95741987 4726.35131306]

[5313.95741987 5757.54808175 5238.69940006]

[4726.35131306 5238.69940006 5874.61823923]]

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

* + Варіюючи кількість компонент, провести реконструкцію окремих каналів кольору зображень (від компонент з найбільшою енергією поступово переходячи до компонентів з мінімальною енергією).
  + Побудувати залежність помилки відновлення (середнє відхилення вихідного зображення відреконструйованого, MSE) від кількості використаних компонент.

"import scipy\n",

"import scipy.ndimage\n",

"import matplotlib.pyplot as plt \n",

"import numpy as np \n",

"from PIL import Image\n",

"\n",

"test\_img = loaded\_images[8]\n",

"a\_np = np.array(test\_img)\n",

"a\_r = a\_np[:,:,0]\n",

"a\_g = a\_np[:,:,1]\n",

"a\_b = a\_np[:,:,2]\n",

"\n",

"def PCA\_2d(image\_2d, numpc):\n",

"\tcov\_mat = image\_2d - np.mean(image\_2d)\n",

"\teig\_val, eig\_vec = np.linalg.eigh(np.cov(cov\_mat))\n",

"\tp = np.size(eig\_vec, axis =1)\n",

"\tidx = np.argsort(eig\_val)\n",

"\tidx = idx[::-1]\n",

"\teig\_vec = eig\_vec[:,idx]\n",

"\teig\_val = eig\_val[idx]\n",

" \n",

"\tif numpc <p or numpc >0:\n",

"\t\teig\_vec = eig\_vec[:, range(numpc)]\n",

"\tscore = np.dot(eig\_vec.T, cov\_mat)\n",

"\trecon = np.dot(eig\_vec, score) + np.mean(image\_2d).T\n",

"\trecon\_img\_mat = np.uint8(np.absolute(recon))\n",

"\treturn recon\_img\_mat\n",

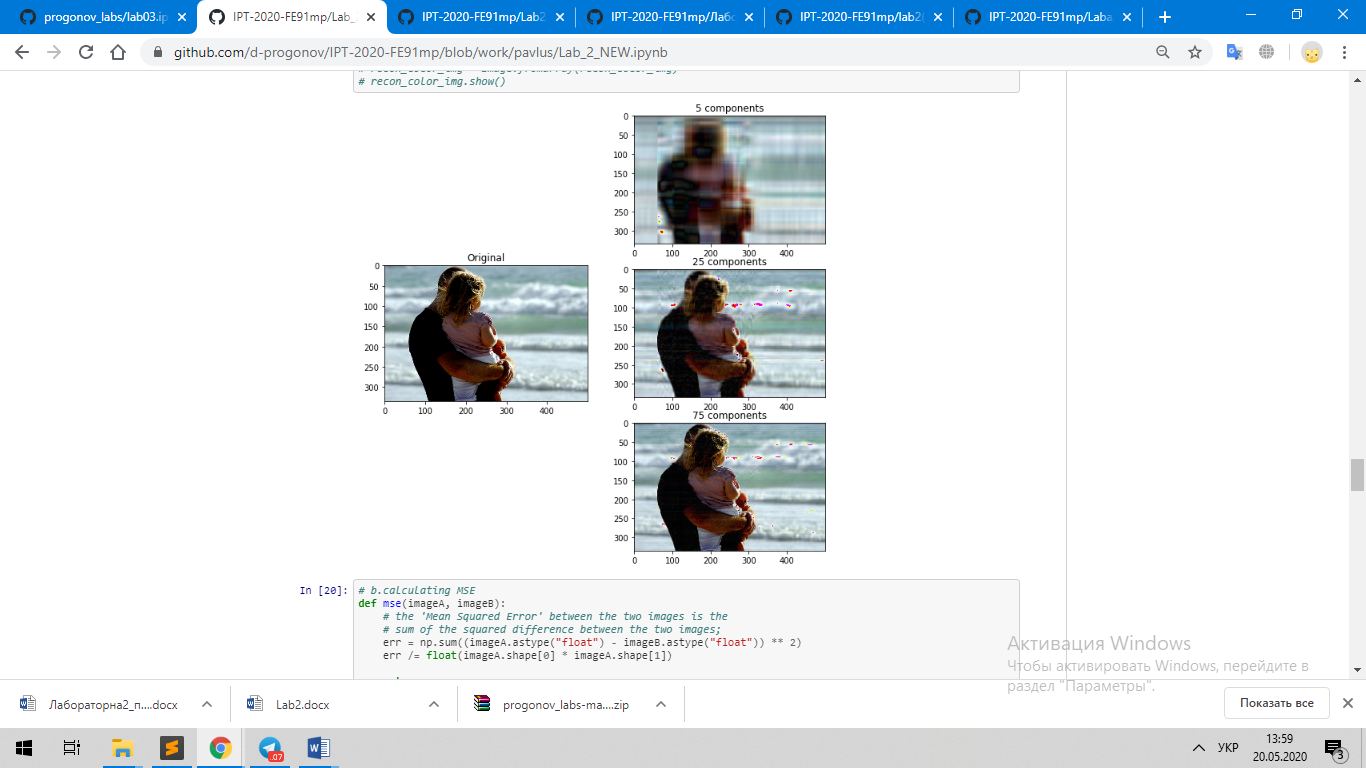


Рисунок 1 — Зображення для аналізу методу головних компонент

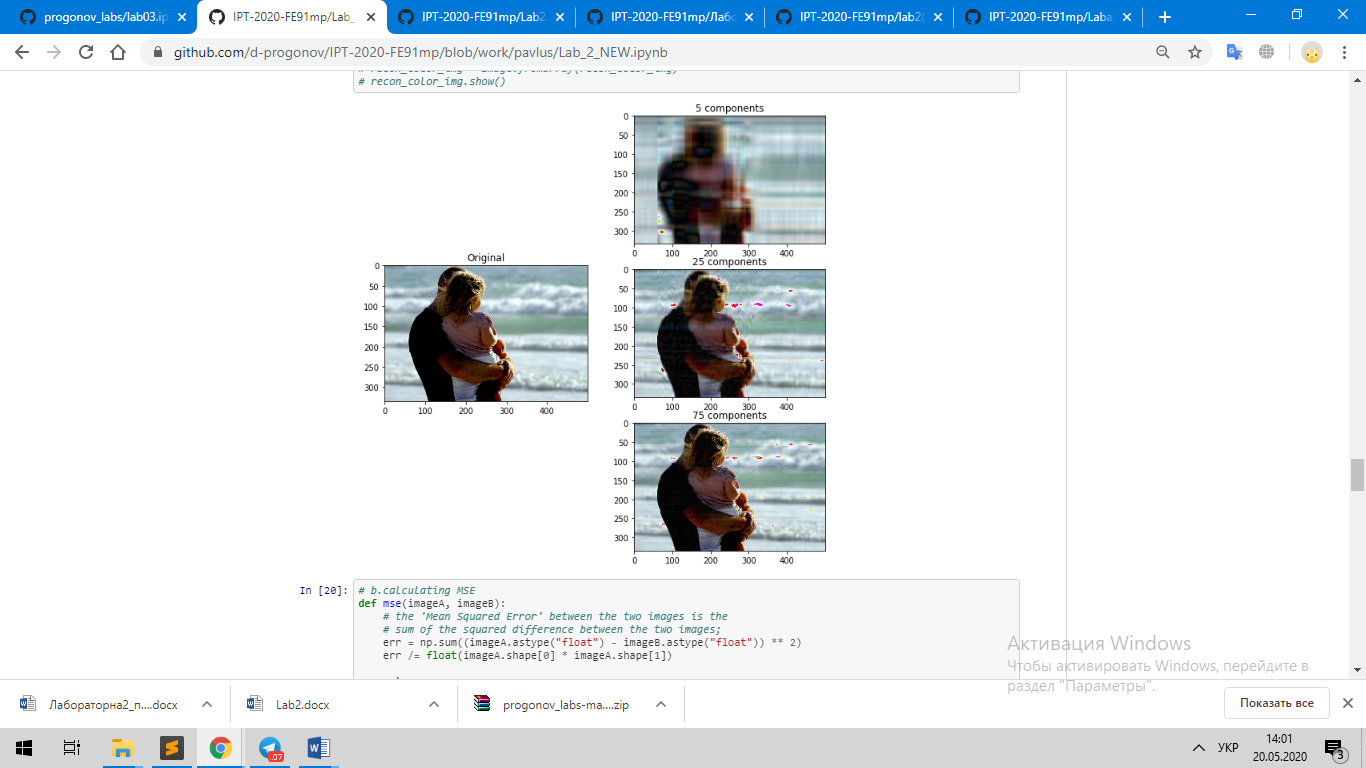
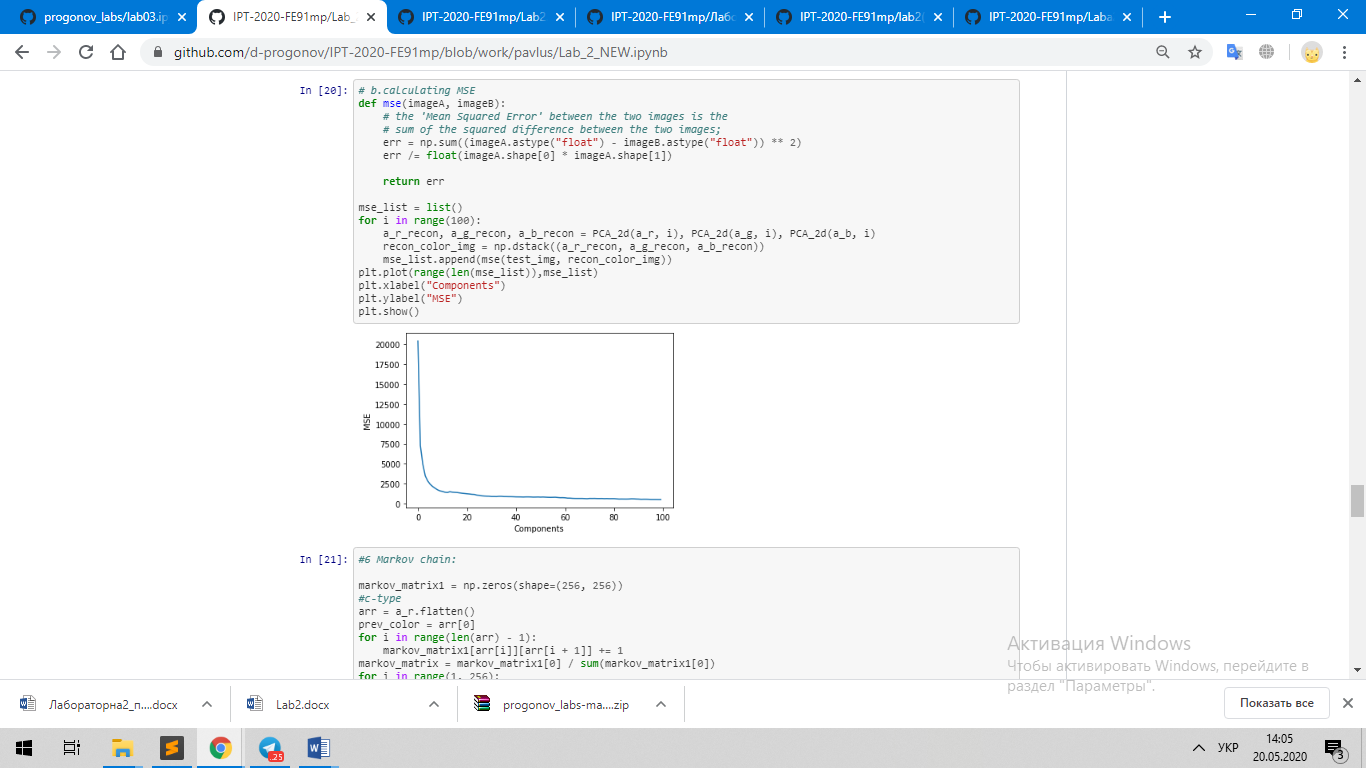


Рисунок 2 – Відновлені зображення при різній кількості головних компонент

Покажемо на графіку залежність помилки відновлення від кількості використаних компонент.



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

"markov\_matrix1 = np.zeros(shape=(256, 256))\n",

"#c-type\n",

"arr = a\_r.flatten()\n",

"prev\_color = arr[0]\n",

"for i in range(len(arr) - 1):\n",

" markov\_matrix1[arr[i]][arr[i + 1]] += 1\n",

"markov\_matrix = markov\_matrix1[0] / sum(markov\_matrix1[0])\n",

"for i in range(1, 256):\n",

" markov\_matrix = np.vstack((markov\_matrix, markov\_matrix1[i] / sum(markov\_matrix1[i])))\n",

"print(\"Red matrix 1st oder:\\n\", markov\_matrix)\n",

"print(\"\\nRed matrix 2nd order:\\n\", np.linalg.matrix\_power(markov\_matrix, 2))\n",

"\n",

"markov\_matrix1 = np.zeros(shape=(256, 256))\n",

"#Fortran-type\n",

"arr = a\_r.flatten('F')\n",

"prev\_color = arr[0]\n",

"for i in range(len(arr) - 1):\n",

" markov\_matrix1[arr[i]][arr[i + 1]] += 1\n",

"markov\_matrix = markov\_matrix1[0] / sum(markov\_matrix1[0])\n",

"for i in range(1, 256):\n",

" markov\_matrix = np.vstack((markov\_matrix, markov\_matrix1[i] / sum(markov\_matrix1[i])))\n",

"print(\"Red matrix 2-nd type 1st oder:\\n\", markov\_matrix)\n",

"print(\"\\nRed matrix 2-nd type 2nd order:\\n\", np.linalg.matrix\_power(markov\_matrix, 2))\n",

"\n",

"markov\_matrix2 = np.zeros(shape=(256, 256))\n",

"arr = a\_g.flatten()\n",

"prev\_color = arr[0]\n",

"for i in range(len(arr) - 1):\n",

" markov\_matrix2[arr[i]][arr[i + 1]] += 1\n",

"markov\_matrix = markov\_matrix2[0] / sum(markov\_matrix2[0])\n",

"for i in range(1, 256):\n",

" markov\_matrix = np.vstack((markov\_matrix, markov\_matrix2[i] / sum(markov\_matrix2[i])))\n",

"print(\"\\n\\nGreen matrix 1st oder:\\n\", markov\_matrix)\n",

"print(\"\\nGreen matrix 2nd order:\\n\", np.linalg.matrix\_power(markov\_matrix, 2))\n",

"\n",

"markov\_matrix3 = np.zeros(shape=(256, 256))\n",

"arr = a\_g.flatten()\n",

"prev\_color = arr[0]\n",

"for i in range(len(arr) - 1):\n",

" markov\_matrix3[arr[i]][arr[i + 1]] += 1\n",

"markov\_matrix = markov\_matrix3[0] / sum(markov\_matrix3[0])\n",

"for i in range(1, 256):\n",

" markov\_matrix = np.vstack((markov\_matrix, markov\_matrix3[i] / sum(markov\_matrix3[i])))\n",

"print(\"\\n\\nBlue matrix 1st oder:\\n\", markov\_matrix)\n",

"print(\"\\nBlue matrix 2nd order:\\n\", np.linalg.matrix\_power(markov\_matrix, 2))"

Отримуємо такі дані:  
  
 Red matrix 1st oder:

[[0.91670631 0.03472332 0.01231436 ... 0. 0. 0. ]

[0.42098446 0.23898964 0.08290155 ... 0. 0. 0. ]

[0.41638225 0.17406143 0.07679181 ... 0. 0. 0. ]

…

[0. 0. 0. ... 0.02083333 0. 0.04166667]

[0. 0. 0. ... 0.03174603 0. 0.03174603]

[0. 0. 0. ... 0.00490196 0.00980392 0.04901961]]

----------------------------------

Red matrix 2nd order:

[[8.65987664e-01 4.53936521e-02 1.66383106e-02 ... 0.00000000e+00

1.19573143e-07 8.90266238e-07]

[5.63330679e-01 1.08696824e-01 4.23282283e-02 ... 8.65288435e-07

4.28068998e-07 4.48222037e-06]

[5.35620762e-01 9.60347518e-02 3.85810058e-02 ... 0.00000000e+00

0.00000000e+00 1.93143037e-05]

…

[1.09104200e-04 3.00625301e-05 0.00000000e+00 ... 6.84857421e-03

1.01985386e-02 2.77937857e-02]

[3.05190128e-04 2.40466451e-04 2.16548648e-05 ... 8.96441187e-03

5.76031961e-03 2.53005425e-02]

[3.57038312e-04 3.36751454e-04 1.25844081e-04 ... 5.97401741e-03

5.89606279e-03 2.00217347e-02]]

-------------------------------------

Red matrix 2-nd type 1st oder:

[[0.92088156 0.03218646 0.01141589 ... 0. 0. 0. ]

[0.41126943 0.26101036 0.07253886 ... 0. 0. 0. ]

[0.37713311 0.21331058 0.09726962 ... 0. 0. 0. ]

…

[0. 0. 0. ... 0.0625 0.02083333 0.08333333]

[0. 0. 0. ... 0.04761905 0.06349206 0.15873016]

[0. 0. 0. ... 0.01960784 0.04901961 0.16176471]]

------------------------

Red matrix 2-nd type 2nd order:

[[8.70525462e-01 4.34622629e-02 1.53622682e-02 ... 1.33546333e-06

9.26774795e-07 6.52730516e-06]

[5.51966110e-01 1.20266388e-01 4.17396057e-02 ... 2.34277529e-06

4.53740341e-06 1.50002999e-05]

[5.19949600e-01 1.18270865e-01 4.31814234e-02 ... 0.00000000e+00

0.00000000e+00 1.43907165e-05]

…

[0.00000000e+00 4.86760125e-05 0.00000000e+00 ... 1.58592559e-02 2.12752593e-02 5.10268350e-02]

[1.19977444e-05 0.00000000e+00 0.00000000e+00 ... 1.68056272e-02

2.47214881e-02 7.16076889e-02]

[1.32464441e-04 3.81088572e-05 5.60224090e-06 ... 1.28758567e-02

2.04869527e-02 5.64856483e-02]]

-------------------------------

Green matrix 1st oder:

[[0.86210575 0.0512987 0.02639147 ... 0. 0. 0. ]

[0.44171293 0.23473434 0.10666138 ... 0. 0. 0. ]

[0.39336158 0.1970339 0.11935028 ... 0. 0. 0. ]

...

[0. 0. 0. ... 0.1375 0.19166667 0.19583333]

[0. 0. 0. ... 0.12394366 0.27605634 0.32112676]

[0. 0. 0. ... 0.07407407 0.17874396 0.38164251]]

---------------------------

Green matrix 2nd order:

[[7.86438780e-01 6.60140412e-02 3.47544873e-02 ... 9.52994374e-07

1.91216163e-06 1.95357463e-06]

[5.61022701e-01 1.14123162e-01 6.09868526e-02 ... 7.62520588e-06

0.00000000e+00 5.55325188e-06]

[5.17709808e-01 1.09798935e-01 6.07084081e-02 ... 5.90974635e-06

2.23776553e-05 7.51292223e-06]

…

[2.46802053e-04 1.07963301e-04 5.41664912e-05 ... 8.74000761e-02

1.44488307e-01 1.93565891e-01]

[1.09814479e-04 3.00065438e-05 4.14479841e-05 ... 8.92581329e-02

1.71203706e-01 2.49976893e-01]

[4.62739785e-04 1.78536624e-04 1.12670925e-04 ... 7.11831287e-02

1.41899633e-01 2.29455015e-01]]

---------------------------

Blue matrix 1st oder:

[[0.86210575 0.0512987 0.02639147 ... 0. 0. 0. ]

[0.44171293 0.23473434 0.10666138 ... 0. 0. 0. ]

[0.39336158 0.1970339 0.11935028 ... 0. 0. 0. ]

…

[0. 0. 0. ... 0.1375 0.19166667 0.19583333]

[0. 0. 0. ... 0.12394366 0.27605634 0.32112676]

[0. 0. 0. ... 0.07407407 0.17874396 0.38164251]]

-------------------

Blue matrix 2nd order:

[[7.86438780e-01 6.60140412e-02 3.47544873e-02 ... 9.52994374e-07

1.91216163e-06 1.95357463e-06]

[5.61022701e-01 1.14123162e-01 6.09868526e-02 ... 7.62520588e-06

0.00000000e+00 5.55325188e-06]

[5.17709808e-01 1.09798935e-01 6.07084081e-02 ... 5.90974635e-06

2.23776553e-05 7.51292223e-06]

…

[2.46802053e-04 1.07963301e-04 5.41664912e-05 ... 8.74000761e-02

1.44488307e-01 1.93565891e-01]

[1.09814479e-04 3.00065438e-05 4.14479841e-05 ... 8.92581329e-02

1.71203706e-01 2.49976893e-01]

[4.62739785e-04 1.78536624e-04 1.12670925e-04 ... 7.11831287e-02

1.41899633e-01 2.29455015e-01]]

Графічне представлення ланцюга:

"import networkx as nx\n",

"import pandas as pd\n",

"\n",

"data = markov\_matrix\n",

"data = np.triu(data) + np.triu(data).T\n",

"ind = [str(i) for i in range(data.shape[0])]\n",

"df2 = pd.DataFrame(data, index=ind, columns=ind)\n",

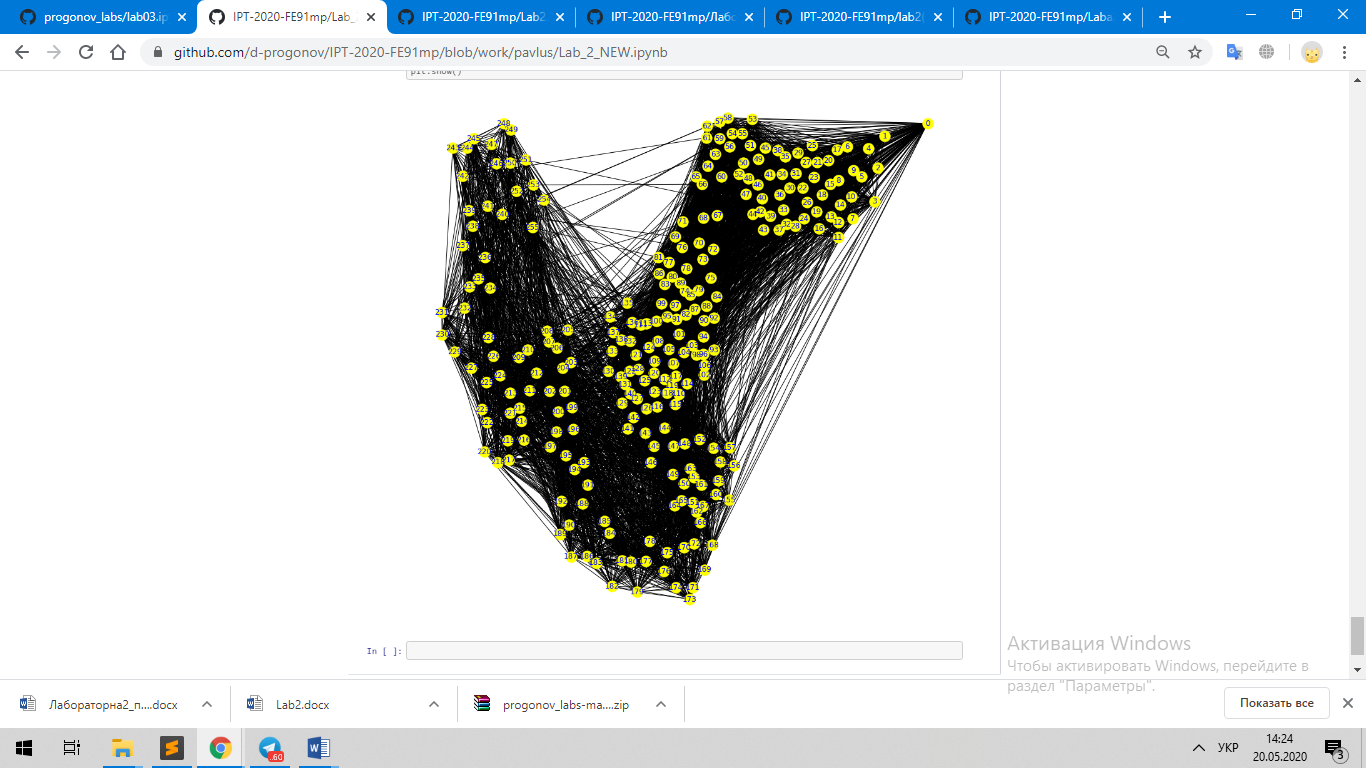
"plt.figure(1,figsize=(12,12)) \n",

"G2 = nx.from\_pandas\_adjacency(df2)\n",

"\n",

"nx.draw(G2, with\_labels=True, node\_color='yellow', font\_color='blue')\n",

"plt.show()"



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 збільшенні кількості компонент зростає якість відновлення. Також було помічено та проаналізовано помилки відновлення. Зібравши дані, було побуловано графік залежності середньої квадратичної похибки відновлених дображень від кількості компонент. Було помічено експоненціальну залежність, що свідчить про значні зміни при невеликих кількостях компонент (до 20) та майже непомітні при великих значеннях (більше 100).

Було проведуно моделювання окремих каналів кольору зображень з використанням марковських ланцюгів, та сформовано стохастичні матриці за різними типами обходів. Також за даними було побуловано графічну можель марківського ланцюга.