ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(МОСКОВСКИЙ ПОЛИТЕХ)**

Кафедра «Инфокогнитивные технологии»

Практические и лабораторные занятия по дисциплине

«Проектирование интеллектуальных систем»

Лабораторная работа № 1

**«Решение оптимизационных задач с помощью генетических** **алгоритмов»**

Группа 224-322

Студент Леонов Владислав Денисович

Преподаватель Кружалов Алексей Сергеевич

**Москва 2023**

**Цель**

Ознакомиться с подходом и приобрести практический навык решения оптимизационных задач с помощью генетических алгоритмов (ГА).

**Теория**

Показатели качества бывают субъективные и объективные.

Субъективные строятся на опросах и чаще всего рассчитываются через среднюю экспертную оценку. При проведении опроса эксперту могут показываться оба изображения исходное и измененное или только измененное. На субъективную оценку сильно влияют условия просмотра изображения.

Объективные методы используют алгоритмы, которые оценивают изменение качества изображения или внесенные преобразования. Объективные методы делятся на:

• методы полного сопоставления (с эталоном);

• методы частичного сопоставления (есть какая-либо информация об

эталоне);

• методы без сопоставления (без эталона);

В первом случае у нас есть эталонное изображение и обработанное

системой, которые мы можем сравнивать напрямую. Во втором — у нас есть изображение, обработанное системой и извлеченные признаки из эталонного, в третьем — только изображение, обработанное системой.

В этой работе будут рассмотрены методы полного сопоставления:

1. среднеквадратическая ошибка (MSE);

2. пиковое отношение сигнал-шум (PSNR);

3. мера структурного подобия (SSIM).

Среднеквадратическая ошибка (MSE)

Показатель определяет среднеквадратичное отклонение светлоты эталонного и обработанного изображений.

Пиковое отношение сигнал-шум (PSNR)

Пиковое отношение сигнала к шуму PSNR (англ. Peak Signal-to-Noise

Ratio) Данная метрика характеризует соотношение между максимумом возможного значения сигнала и мощностью шума, искажающего значения сигнала. Поскольку многие сигналы имеют широкий динамический диапазон, PSNR обычно измеряется в логарифмической шкале в децибелах.

Мера структурного подобия (SSIM)

Мера структурного подобия является развитием традиционных методов, таких как PSNR и MSE. Он учитывает изменение яркости и контраста, а также степень коррелированности между двумя изображениями.

**Выполнение работы**

1. Подобрать 9 изображений: с малым количеством деталей, со средним и большим. По 3 в каждой группе. Выполнение данного пункта показано на рисунке 1.

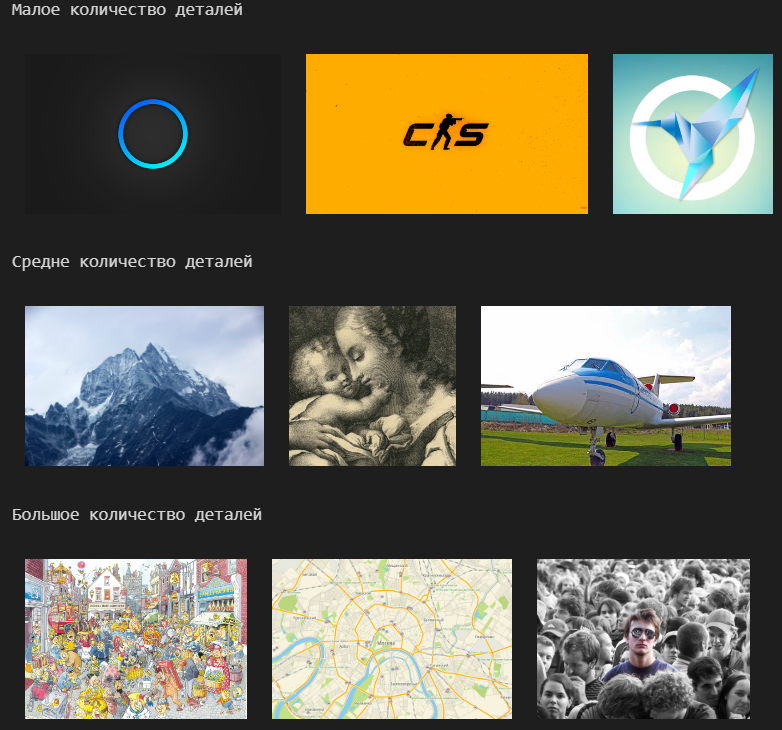


Рисунок 1 – Подобранные 9 изображений

1. В исходные изображения внести искажения: сохранить с потерями, размыть изображение, например, с помощью фильтра, добавить импульсный или детерминированный шум, или какие-либо другие искажения. Искажения могут применяться как по отдельности, так и вместе.

На каждое изображение по отдельности были добавлены 3 вида искажений:

1. Сохрание изображения с потерями
2. Нанесение размытия (blur)
3. Добавление импульсного шума

Результат искажений для 1-й группы показан на рисунке 2.

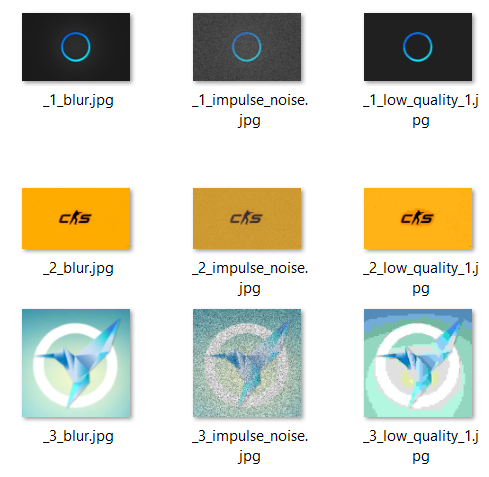


Рисунок 2 – Результат искажений для 1-й группы

Результат искажений для 2-й группы показан на рисунке 3.

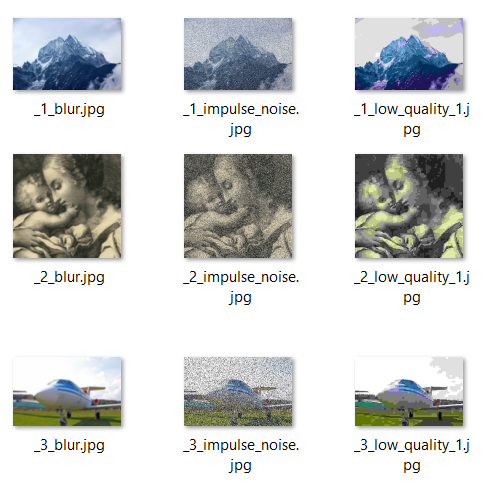


Рисунок 3 – Результат искажений для 2-й группы

Результат искажений для 3-й группы показан на рисунке 4.

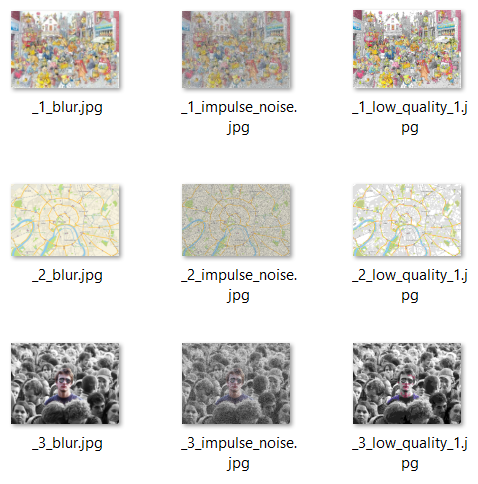


Рисунок 4 – Результат искажений для 3-й группы

1. Написать код для расчета трех показателей: MSE, PSNR и SSIM.

Код для расчет показателей приведен в листинге 1.

Листинг 1 – Код для расчета трех показателей

|  |
| --- |
| import cv2  import os  import numpy as np  import pandas as pd  from skimage.metrics import peak\_signal\_noise\_ratio as psnr  from skimage.metrics import structural\_similarity as ssim  from skimage.metrics import mean\_squared\_error as mse  \_group\_1\_paths = ['\_images/group\_1/\_1.jpg', '\_images/group\_1/\_2.jpg', '\_images/group\_1/\_3.jpg']  \_group\_2\_paths = ['\_images/group\_2/\_1.jpg', '\_images/group\_2/\_2.jpg', '\_images/group\_2/\_3.jpg']  \_group\_3\_paths = ['\_images/group\_3/\_1.jpg', '\_images/group\_3/\_2.jpg', '\_images/group\_3/\_3.jpg']  df = pd.DataFrame()  def Calc\_MSE\_PSNR\_SSIM(orig\_img\_path, name\_group):      global df      orig\_img = cv2.imread(orig\_img\_path)      \_basename = os.path.basename(orig\_img\_path).replace(".jpg", "")      \_dirname = os.path.dirname(orig\_img\_path)      \_result\_dirname = os.path.join(\_dirname, "results")      for root, dirs, files in os.walk(\_result\_dirname):          for file in files:              if \_basename not in os.path.basename(file)[:5]: continue;              dif\_img  = cv2.imread(os.path.join(root, file))              \_MSE = mse(orig\_img, dif\_img)              \_PSNR = psnr(orig\_img, dif\_img)              \_SSIM = ssim(orig\_img, dif\_img, channel\_axis = -1)              df = df.append({'Group': name\_group, 'BaseFile': os.path.basename(orig\_img\_path), 'DifFile': os.path.basename(file), 'MSE': round(\_MSE, 3), 'PSNR': round(\_PSNR, 3), 'SSIM': round(\_SSIM, 3)}, ignore\_index=True)  print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  [Calc\_MSE\_PSNR\_SSIM(x, "Small") for x in \_group\_1\_paths]  print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  [Calc\_MSE\_PSNR\_SSIM(x, "Middle") for x in \_group\_2\_paths]  print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  [Calc\_MSE\_PSNR\_SSIM(x, "Large") for x in \_group\_3\_paths]  print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  print(df)  df.to\_excel("\_images/output\_Calc\_MSE\_PSNR\_SSIM.xlsx") |

1. Провести расчет трех показателей для полученных изображений.

Расчет показателей MSE, PSNR и SSIM приведет в таблице 1.

Таблица 1 - Расчет трех показателей для полученных изображений

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Group** | **BaseFile** | **DifFile** | **MSE** | **PSNR** | **SSIM** |
| **0** | Small | \_1.jpg | \_1\_blur.jpg | 9,346 | 38,425 | 0,991 |
| **1** | Small | \_1.jpg | \_1\_impulse\_noise.jpg | 9740,184 | 8,245 | 0,008 |
| **2** | Small | \_1.jpg | \_1\_low\_quality\_1.jpg | 46,801 | 31,428 | 0,97 |
| **3** | Small | \_2.jpg | \_2\_blur.jpg | 5,63 | 40,626 | 0,989 |
| **4** | Small | \_2.jpg | \_2\_impulse\_noise.jpg | 8061,961 | 9,066 | 0,008 |
| **5** | Small | \_2.jpg | \_2\_low\_quality\_1.jpg | 177,739 | 25,633 | 0,679 |
| **6** | Small | \_3.jpg | \_3\_blur.jpg | 23,968 | 34,334 | 0,95 |
| **7** | Small | \_3.jpg | \_3\_impulse\_noise.jpg | 8726,489 | 8,722 | 0,017 |
| **8** | Small | \_3.jpg | \_3\_low\_quality\_1.jpg | 274,248 | 23,749 | 0,816 |
| **9** | Middle | \_1.jpg | \_1\_blur.jpg | 142,259 | 26,6 | 0,766 |
| **10** | Middle | \_1.jpg | \_1\_impulse\_noise.jpg | 8353,565 | 8,912 | 0,04 |
| **11** | Middle | \_1.jpg | \_1\_low\_quality\_1.jpg | 459,93 | 21,504 | 0,655 |
| **12** | Middle | \_2.jpg | \_2\_blur.jpg | 503,598 | 21,11 | 0,317 |
| **13** | Middle | \_2.jpg | \_2\_impulse\_noise.jpg | 7504,641 | 9,378 | 0,103 |
| **14** | Middle | \_2.jpg | \_2\_low\_quality\_1.jpg | 772,703 | 19,251 | 0,273 |
| **15** | Middle | \_3.jpg | \_3\_blur.jpg | 204,719 | 25,019 | 0,751 |
| **16** | Middle | \_3.jpg | \_3\_impulse\_noise.jpg | 9317,759 | 8,438 | 0,049 |
| **17** | Middle | \_3.jpg | \_3\_low\_quality\_1.jpg | 346,963 | 22,728 | 0,676 |
| **18** | Large | \_1.jpg | \_1\_blur.jpg | 1005,018 | 18,109 | 0,402 |
| **19** | Large | \_1.jpg | \_1\_impulse\_noise.jpg | 8172,905 | 9,007 | 0,165 |
| **20** | Large | \_1.jpg | \_1\_low\_quality\_1.jpg | 1121,323 | 17,633 | 0,528 |
| **21** | Large | \_2.jpg | \_2\_blur.jpg | 611,844 | 20,264 | 0,564 |
| **22** | Large | \_2.jpg | \_2\_impulse\_noise.jpg | 10010,29 | 8,126 | 0,076 |
| **23** | Large | \_2.jpg | \_2\_low\_quality\_1.jpg | 833,345 | 18,923 | 0,495 |
| **24** | Large | \_3.jpg | \_3\_blur.jpg | 103,675 | 27,974 | 0,836 |
| **25** | Large | \_3.jpg | \_3\_impulse\_noise.jpg | 7835,349 | 9,19 | 0,053 |
| **26** | Large | \_3.jpg | \_3\_low\_quality\_1.jpg | 286,095 | 23,566 | 0,583 |

**Вывод:**

В данной работе наблюдается зависимость между полученными показателями: чем выше показатель MSE, тем ниже показатели PSNR и SSIM.

По показателю MSE в итоговой таблице можно сказать, что шум наибольшим образом влияет на разность изображений, так как значение становится очень большим, а с увеличением количества деталей на фото при искажениях увеличивается и значение MSE.

По показателю PSNR в итоговой таблице можно сказать, что с увеличением деталей на фотографиях данный параметр уменьшается, также на данный параметр сильно влияет шум, что приводит к резкому спаду значений.

По показателю SSIM в итоговой таблице можно сказать, что размытие и сохранение с потерей качества наименьшем образом влияет на разность с оригинальным изображением, так как значение близко к 1, но с увеличением деталей эта разность становится все больше и больше. Шум же максимально влияет на разность изображений.

**Программный код:**

from IPython.display import Image, display, HTML

def make\_html(img\_path):

     return '<img src="{}" style="display:inline;margin:10px;" height="128"/>'.format(img\_path)

\_group\_1\_paths = ['\_images/group\_1/\_1.jpg', '\_images/group\_1/\_2.jpg', '\_images/group\_1/\_3.jpg']

\_group\_2\_paths = ['\_images/group\_2/\_1.jpg', '\_images/group\_2/\_2.jpg', '\_images/group\_2/\_3.jpg']

\_group\_3\_paths = ['\_images/group\_3/\_1.jpg', '\_images/group\_3/\_2.jpg', '\_images/group\_3/\_3.jpg']

print("Малое количество деталей")

display(HTML(''.join([make\_html(x) for x in \_group\_1\_paths])))

print("Средне количество деталей")

display(HTML(''.join([make\_html(x) for x in \_group\_2\_paths])))

print("Большое количество деталей")

display(HTML(''.join([make\_html(x) for x in \_group\_3\_paths])))

import cv2

import os

import numpy as np

# Сохраняем изображения с потерями

def save\_low\_quality(in\_path):

    \_QUALITY = 1

    \_dirname = os.path.dirname(in\_path)

    \_filename = os.path.basename(in\_path).replace(".jpg", "")

    \_result\_dirname = os.path.join(\_dirname, "results")

    \_result\_path = os.path.join(\_result\_dirname, \_filename + f"\_low\_quality\_{\_QUALITY}" + ".jpg")

    img = cv2.imread(in\_path)

    cv2.imwrite(\_result\_path, img, [cv2.IMWRITE\_JPEG\_QUALITY, \_QUALITY])

print("Сохраняем изображения с потерями")

[save\_low\_quality(x) for x in \_group\_1\_paths]

[save\_low\_quality(x) for x in \_group\_2\_paths]

[save\_low\_quality(x) for x in \_group\_3\_paths]

# Нанесение размытия (blur)

def save\_blur(in\_path):

    \_dirname = os.path.dirname(in\_path)

    \_filename = os.path.basename(in\_path).replace(".jpg", "")

    \_result\_dirname = os.path.join(\_dirname, "results")

    \_result\_path = os.path.join(\_result\_dirname, \_filename + "\_blur" + ".jpg")

    img = cv2.imread(in\_path)

    blur\_img = cv2.blur(img, (7, 7))

    cv2.imwrite(\_result\_path, blur\_img)

print("Размываем изображения ядром 7х7 (blur)")

[save\_blur(x) for x in \_group\_1\_paths]

[save\_blur(x) for x in \_group\_2\_paths]

[save\_blur(x) for x in \_group\_3\_paths]

# Добавляем импульсный шум

def save\_impulse\_noise(in\_path):

    noise\_probability = 50

    \_dirname = os.path.dirname(in\_path)

    \_filename = os.path.basename(in\_path).replace(".jpg", "")

    \_result\_dirname = os.path.join(\_dirname, "results")

    \_result\_path = os.path.join(\_result\_dirname, \_filename + "\_impulse\_noise" + ".jpg")

    img = cv2.imread(in\_path)

    h, w = img.shape[:2]

    noise = np.zeros((h, w), np.uint8)

    cv2.randu(noise, 0, 255)

    image\_with\_noise = img.copy()

    image\_with\_noise[noise < noise\_probability] = 0

    image\_with\_noise[noise > 255 - noise\_probability] = 255

    cv2.imwrite(\_result\_path, image\_with\_noise)

print("Добавляем импульсный шум")

[save\_impulse\_noise(x) for x in \_group\_1\_paths]

[save\_impulse\_noise(x) for x in \_group\_2\_paths]

[save\_impulse\_noise(x) for x in \_group\_3\_paths]

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

# python -m pip install -U scikit-image

# pip install pandas

# pip install openpyxl

import cv2

import os

import numpy as np

import pandas as pd

from skimage.metrics import peak\_signal\_noise\_ratio as psnr

from skimage.metrics import structural\_similarity as ssim

from skimage.metrics import mean\_squared\_error as mse

\_group\_1\_paths = ['\_images/group\_1/\_1.jpg', '\_images/group\_1/\_2.jpg', '\_images/group\_1/\_3.jpg']

\_group\_2\_paths = ['\_images/group\_2/\_1.jpg', '\_images/group\_2/\_2.jpg', '\_images/group\_2/\_3.jpg']

\_group\_3\_paths = ['\_images/group\_3/\_1.jpg', '\_images/group\_3/\_2.jpg', '\_images/group\_3/\_3.jpg']

df = pd.DataFrame()

def Calc\_MSE\_PSNR\_SSIM(orig\_img\_path, name\_group):

    global df

    orig\_img = cv2.imread(orig\_img\_path)

    \_basename = os.path.basename(orig\_img\_path).replace(".jpg", "")

    \_dirname = os.path.dirname(orig\_img\_path)

    \_result\_dirname = os.path.join(\_dirname, "results")

    for root, dirs, files in os.walk(\_result\_dirname):

        for file in files:

            if \_basename not in os.path.basename(file)[:5]: continue;

            dif\_img  = cv2.imread(os.path.join(root, file))

            \_MSE = mse(orig\_img, dif\_img)

            \_PSNR = psnr(orig\_img, dif\_img)

            \_SSIM = ssim(orig\_img, dif\_img, channel\_axis = -1)

            #print(f"----- {\_dirname}: {os.path.basename(orig\_img\_path)} -> {os.path.basename(file)} -----")

            #print(f'MSE: {round(\_MSE, 2)}, PSNR: {round(\_PSNR, 2)}, SSIM: {round(\_SSIM, 2)}')

            df = df.append({'Group': name\_group, 'BaseFile': os.path.basename(orig\_img\_path), 'DifFile': os.path.basename(file), 'MSE': round(\_MSE, 3), 'PSNR': round(\_PSNR, 3), 'SSIM': round(\_SSIM, 3)}, ignore\_index=True)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

[Calc\_MSE\_PSNR\_SSIM(x, "Small") for x in \_group\_1\_paths]

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

[Calc\_MSE\_PSNR\_SSIM(x, "Middle") for x in \_group\_2\_paths]

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

[Calc\_MSE\_PSNR\_SSIM(x, "Large") for x in \_group\_3\_paths]

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print(df)

df.to\_excel("\_images/output\_Calc\_MSE\_PSNR\_SSIM.xlsx")