**Минобрнауки РФ**

**ЮЗГУ**

**ФФиПИ**

Кафедра программной инженерии

**ЛАБОРАТОРНАЯ РАБОТА №4**

**ГЕОМЕТРИЧЕСКИЕ ПРЕОБРАЗОВАНИЯ**

Выполнил: ст. гр. ПО-22б

Алешин К.Р.

Проверил: Ефремов В. В.

Курск, 2024 г

**Задание:**  
****

**Листинг программы:**import numpy as np

from PIL import Image

def affine\_transform(image, scale=(1, 1), translate=(0, 0)):

"""Аффинное преобразование: масштабирование и перенос."""

width, height = image.size

new\_width = int(width \* scale[0])

new\_height = int(height \* scale[1])

# Создаём пустое изображение

transformed\_image = Image.new("RGB", (new\_width, new\_height))

pixels\_original = image.load()

pixels\_transformed = transformed\_image.load()

# Преобразование координат

for x in range(new\_width):

for y in range(new\_height):

src\_x = int((x - translate[0]) / scale[0])

src\_y = int((y - translate[1]) / scale[1])

if 0 <= src\_x < width and 0 <= src\_y < height:

pixels\_transformed[x, y] = pixels\_original[src\_x, src\_y]

return transformed\_image

def inverse\_affine\_transform(image, scale=(1, 1), translate=(0, 0), original\_size=(0, 0)):

"""Обратное аффинное преобразование."""

width, height = original\_size

transformed\_width, transformed\_height = image.size

# Создаём пустое изображение

restored\_image = Image.new("RGB", (width, height))

pixels\_transformed = image.load()

pixels\_restored = restored\_image.load()

# Преобразование координат

for x in range(width):

for y in range(height):

src\_x = int(x \* scale[0] + translate[0])

src\_y = int(y \* scale[1] + translate[1])

if 0 <= src\_x < transformed\_width and 0 <= src\_y < transformed\_height:

pixels\_restored[x, y] = pixels\_transformed[src\_x, src\_y]

return restored\_image

def nonlinear\_transform(image):

"""Нелинейное преобразование: i = 2^x', j = y'."""

width, height = image.size

new\_width = int(2 \*\* (width - 1).bit\_length()) # Определяем минимальную степень 2

new\_height = height

# Создаём пустое изображение

transformed\_image = Image.new("RGB", (new\_width, new\_height))

pixels\_original = image.load()

pixels\_transformed = transformed\_image.load()

# Преобразование координат

for x in range(width):

for y in range(height):

transformed\_x = int(2 \*\* x) if 2 \*\* x < new\_width else new\_width - 1

transformed\_y = y

pixels\_transformed[transformed\_x, transformed\_y] = pixels\_original[x, y]

return transformed\_image

def main():

# Пути к файлам

input\_image\_path = "input.jpg"

affine\_transformed\_path = "affine\_transformed.jpg"

restored\_image\_path = "restored\_image.jpg"

nonlinear\_transformed\_path = "nonlinear\_transformed.jpg"

# Чтение изображения

image = Image.open(input\_image\_path).convert("RGB")

# Аффинные преобразования (масштабирование и перенос)

scale = (1.5, 1.5) # Масштабирование по X и Y

translate = (50, 50) # Перенос на 50 пикселей вправо и вниз

affine\_transformed\_image = affine\_transform(image, scale, translate)

affine\_transformed\_image.save(affine\_transformed\_path)

print(f"Аффинное преобразование выполнено, результат сохранён в {affine\_transformed\_path}")

# Восстановление изображения

restored\_image = inverse\_affine\_transform(affine\_transformed\_image, scale, translate, image.size)

restored\_image.save(restored\_image\_path)

print(f"Исходное изображение восстановлено, результат сохранён в {restored\_image\_path}")

# Нелинейное преобразование

nonlinear\_transformed\_image = nonlinear\_transform(image)

nonlinear\_transformed\_image.save(nonlinear\_transformed\_path)

print(f"Нелинейное преобразование выполнено, результат сохранён в {nonlinear\_transformed\_path}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Пример работы программы:  
Изображение выглядит как снимок экрана, дерево, небо, на открытом воздухе

Автоматически созданное описание**

**Изображение выглядит как снимок экрана, Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание**