ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

канд. техн. наук, доцент		А. В. Аграновский
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 7

ИССЛЕДОВАНИЕ АРТЕФАКТОВ АФФИННЫХ ПРЕОБРАЗОВАНИЙ

по курсу:

КОМПЬЮТЕРНАЯ ГРАФИКА

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4326		Г. С. Томчук
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

1 Цель работы

Цель работы: исследование особенностей и артефактов, возникающих при выполнении аффинных преобразований изображений.

2 Формулировка задания

Необходимо написать программу, выполняющую преобразования фотографии в соответствии с вариантом (№ 20).

1. Осуществить поворот изображения на различные углы с использованием стандартных аффинных матриц и матриц Оуэна и Македона.
2. Проанализировать и сравнить возникшие артефакты при выполнении преобразований обоими методами.
3. Визуализировать результаты преобразований и выявить возможные проблемы, связанные с потерей качества или появлением нежелательных эффектов.
4. Исследовать влияние различных углов поворота на степень проявления артефактов.

3 Теоретические сведения

Аффинные преобразования — это линейные преобразования, которые сохраняют параллельность прямых и пропорции длин отрезков, но могут изменять углы между прямыми и величины длин. Они могут быть представлены матрицей, которая действует на координаты точек в пространстве, преобразуя их в новые координаты. В двумерном пространстве общее аффинное преобразование можно записать в виде:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

где x, y — координаты исходной точки, x', y' — координаты преобразованной точки, a_{ij} — элементы матрицы преобразования, а b_1, b_2 — сдвиги по осям.

Примеры аффинных преобразований:

- Поворот: сохраняет форму объектов, но изменяет их ориентацию.
- Масштабирование: изменяет размер объекта, сохраняя пропорции.
- Перенос: сдвигает объект на заданное расстояние в любом направлении.
- Наклон (или сдвиг): искажает фигуру, но сохраняет параллельность прямых.

Поворот изображения вокруг его центра можно описать с использованием матрицы поворота. Для поворота на угол θ матрица будет иметь вид:

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Когда мы умножаем координаты точки на эту матрицу, точка поворачивается на угол θ против часовой стрелки.

Методы Оуэна и Македона используют комбинацию двух матриц для выполнения поворота. Они были предложены для решения проблемы артефактов, возникающих при применении стандартных матриц поворота в цифровых изображениях.

- Метод Оуэна. Это метод наклона, при котором используются два шага: сначала наклон в одном направлении (например, по оси x), а затем в другом направлении (например, по оси y). Это позволяет минимизировать некоторые виды искажений, часто встречающиеся при поворотах.
- Метод Македона комбинирует два шага: векторное поведение с коррекцией искажений, что позволяет более точно интерполировать значения пикселей, получая изображения с меньшими потерями качества.

Комбинированное использование матриц Оуэна и Македона позволяет

уменьшить количество артефактов, таких как пиксельные разрывы и неровности, которые могут возникать при стандартных аффинных преобразованиях.

При применении аффинных преобразований к изображениям могут возникать различные артефакты, такие как:

- Точечные артефакты: Появляются в случае неправильного округления координат после преобразования, что может привести к "потерянным" пикселям.
- Искажение текстур: При поворотах или других трансформациях может наблюдаться искажение текстуры изображения, особенно при больших углах поворота.
- Шум и разрывы: Ошибки в интерполяции и отсутствии плавности между пикселями, особенно при использовании методик, не учитывающих смежные пиксели.

Методы борьбы с артефактами:

- Интерполяция: Использование различных методов интерполяции, таких как билинейная или бикубическая интерполяция, может значительно снизить артефакты, улучшая качество изображения после преобразования.
- Применение более сложных матриц: Использование таких методов, как комбинация матриц Оуэна и Македона, помогает минимизировать искажения и артефакты, связанные с традиционными аффинными преобразованиями.
- Увеличение разрешения: Увеличение исходного разрешения изображения перед преобразованием может уменьшить количество артефактов, так как большое количество пикселей позволяет более точно отображать детали.

4 Описание алгоритма решения

1. Загрузить исходное изображение в формате BMP.
2. Получить размер изображения и подготовить пустое полотно для хранения преобразованных изображений.
3. Для каждого угла поворота (2°, 6° и 10°): применить стандартную матрицу поворота для аффинного преобразования; применить методы Оуэна и Македона для вычисления преобразования с минимальными артефактами.
4. Используя соответствующую матрицу для каждого угла поворота, вычислить новые координаты для каждого пикселя.
5. Для каждой трансформации (стандартное аффинное и метод Оуэна-Македона) выполнить интерполяцию и округление координат пикселей.
6. Отобразить исходное изображение и три поворота, выполненные стандартным способом.
7. Отобразить те же повороты, выполненные с использованием методов Оуэна и Македона.
8. Сравнить результаты и выявить артефакты, такие как искажения, потеря пикселей или шум.
9. Сохранить преобразованные изображения в формате BMP.
10. Вывести на экран изображения для анализа.

5 Выбор языка программирования и используемых библиотек

Для написания программы я выбрал язык программирования Python. Данный язык имеет богатую экосистему библиотек, в том числе и необходимых для работы с отрисовкой и графикой. Помимо этого, я уже имел опыт работы с данным языком, что ускорило и облегчило рабочий процесс.

Используемые библиотеки:

– NumPy — используется для работы с матрицами и векторами, что упрощает вычисления аффинных преобразований, таких как поворот, сдвиг и масштабирование. Это стандартная библиотека

для эффективных математических операций в Python.

- Pillow – для загрузки, изменения и сохранения изображений.
- Tkinter – для интерфейса.

6 Описание разработанной программы

Программа (см. Приложение А) выполняет следующие основные шаги:

1. Используя библиотеку Pillow, программа загружает исходное изображение в формате BMP и извлекает его размеры.
2. Программа реализует поворот изображения на три заданных угла (например, 2°, 6° и 10°) с использованием стандартной матрицы поворота.
3. Для каждого угла выполняется также поворот с использованием методов Оуэна и Македона для минимизации артефактов.
4. Для каждого метода программа вычисляет новые координаты пикселей, применяя соответствующие матрицы преобразования. Для точности используется интерполяция (например, билинейная), чтобы избежать появления "пустых" пикселей.
5. С помощью tkinter программа отображает исходное изображение, а также три преобразованных изображения для каждого метода.
6. Программа сохраняет преобразованные изображения в формате BMP для дальнейшего анализа.

Цель программы — продемонстрировать артефакты, возникающие при различных способах выполнения аффинных преобразований, и оценить влияние методов Оуэна и Македона.

7 Скриншоты, иллюстрирующие результаты работы программы

На рис. 1 представлен скриншот окна программы. Можно наблюдать, что при использовании обоих методов преобразования так или иначе видны артефакты в виде черных точек. Это может быть связано в первую очередь с тем, что исходное изображение имеет довольно низкое разрешение.

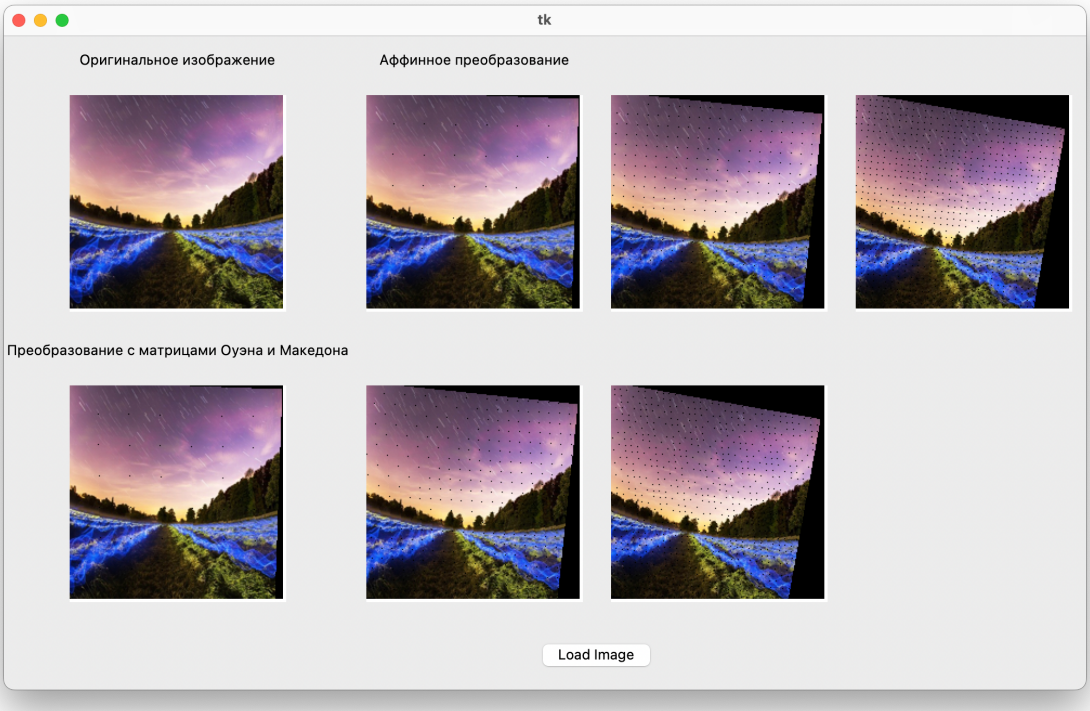

Рисунок 1

8 Вывод

В ходе выполнения лабораторной работы был проведен детальный анализ артефактов, возникающих при аффинных преобразованиях изображений, а также сравнение двух методов поворота: стандартной матрицы поворота и комбинированных методов Оуэна и Македона.

Теоретические знания и навыки, полученные в процессе работы:

1. Были углубленно изучены основы аффинных преобразований, таких как повороты, масштабирования и сдвиги, а также их применение в компьютерной графике для обработки изображений. Понимание теории этих преобразований позволило лучше понять, как они

воздействуют на пиксельные данные и приводят к искажениям.

2. Важной частью работы стало освоение применения матриц для выполнения поворотов. Было рассмотрено использование стандартных матриц поворота и более сложных методов Оуэна и Македона для минимизации артефактов, связанных с потерей качества и текстурными искажениями.
3. Были получены навыки применения различных методов интерполяции, таких как билинейная интерполяция, для вычисления новых значений пикселей после преобразования. Это важно для устранения артефактов и улучшения качества изображения.
4. Работа с библиотеками Pillow и matplotlib для визуализации результатов преобразований помогла лучше понять, как именно аффинные преобразования влияют на изображение и как можно увидеть их влияние на практических примерах.
5. При использовании стандартной матрицы поворота часто возникали точечные артефакты, такие как разрывы и потеря пикселей, особенно при поворотах на большие углы. Эти артефакты возникали из-за недостаточной точности интерполяции и округления координат пикселей.
6. При использовании этих методов артефакты проявлялись в меньшей степени. Комбинированное использование наклонов в разных направлениях помогло снизить искажения и улучшить точность преобразования, однако полностью избавиться от артефактов не удалось, особенно при больших углах поворота.
7. Проблемы, связанные с интерполяцией, также играли важную роль в появлении артефактов. Для повышения качества изображения можно было бы применить более сложные методы интерполяции, такие как бикубическая интерполяция, что могло бы уменьшить разрывы и

шум.

Возникшие проблемы и пути их решения:

1. Проблемы с точностью преобразования: Один из основных вызовов заключался в точности вычислений при применении матриц и округлении пикселей. Некоторые артефакты, такие как разрывы и пиксельные искажения, могли бы быть уменьшены при использовании более точных методов интерполяции и увеличении разрешения изображений перед преобразованием.
2. Появление артефактов при больших углах поворота: При больших углах поворота возникали проблемы с выравниванием пикселей, что приводило к появлению "пустых" или "искаженных" областей на изображении. Для решения этой проблемы можно было бы увеличить размер изображения или использовать более продвинутые методы обработки, такие как коррекция сдвига пикселей или многократное преобразование с корректировкой.
3. Ограничения матриц преобразования: Несмотря на использование методов Оуэна и Македона, полный контроль над артефактами оставался ограниченным. Возможным улучшением было бы использование более сложных комбинированных методов, включающих дополнительные шаги для коррекции и восстановления данных в процессе преобразования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Аграновский, А. В. Использование методов преобразования координат для формирования растровых изображений. Учебно-методическое пособие / А. В. Аграновский. — СПб.: Редакционно-издательский центр ГУАП, 2024. — 40 с.
2. Петров, А. Н., Смирнов, И. В. Основы программирования на Python: Учебное пособие. – Санкт-Петербург: Издательство СПб ГУАП, 2022. – 320 с.
3. Бобылев, С. И. Основы компьютерной графики: учебное пособие / С. И. Бобылев. — М.: Издательство МГТУ им. Н. Э. Баумана, 2010. — 224 с.
4. Owen Ch.B., Makedon F. High quality alias free image rotation // Proceeding of 30th Asilomar Conference on Signals, Systems, and Computers Pacific Grove, California, November 2—6, 1996. URL: https://digitalcommons.dartmouth.edu/cgi/viewcontent.cgi?article=5030&context=facoa (дата посещения 01.12.2024).
5. NumPy. Documentation. URL: https://numpy.org/doc/stable/ (дата обращения: 27.10.2024).
6. Matplotlib. Documentation. URL: https://matplotlib.org/stable/contents.html (дата обращения: 27.10.2024).

ПРИЛОЖЕНИЕ А

```
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk
import numpy as np
import math


def load_image():
    file_path = filedialog.askopenfilename(filetypes=[("BMP
files", "*.bmp")])
    if not file_path:
        return
    image = Image.open(file_path)
    image = image.resize((200, 200))  # Приводим все изображения к
единому размеру
    display_image(image, original_canvas)

    # Углы для поворота
    angles = [2, 6, 10]

    # Преобразования
    affine_images = [rotate_affine(image, angle) for angle in
angles]
    owen_makedon_images = [rotate_owen_makedon(image, angle) for
angle in angles]

    # Отображение изображений
    display_images(affine_images, affine_canvases,
affine_images_storage)
    display_images(owen_makedon_images, owen_canvases,
owen_images_storage)


def display_image(image, canvas):
    canvas.image = ImageTk.PhotoImage(image)
    canvas.create_image(0, 0, anchor="nw", image=canvas.image)


def display_images(images, canvases, storage):
    for i, (img, canvas) in enumerate(zip(images, canvases)):
        img_tk = ImageTk.PhotoImage(img)
        storage[i] = img_tk  # Сохраняем ссылку на изображение
        canvas.create_image(0, 0, anchor="nw", image=img_tk)


def rotate_affine(image, angle):
    radians = math.radians(angle)
    cos_a, sin_a = math.cos(radians), math.sin(radians)
    affine_matrix = np.array([
```

```
        [cos_a, -sin_a, 0],
        [sin_a, cos_a, 0],
        [0, 0, 1]
    ])
    return apply_transformation(image, affine_matrix)


def rotate_owen_makedon(image, angle):
    radians = math.radians(angle)
    cos_a, sin_a = math.cos(radians), math.sin(radians)
    owen_matrix = np.array([
        [1, -sin_a, 0],
        [0, cos_a, 0],
        [0, 0, 1]
    ])
    makedon_matrix = np.array([
        [cos_a, 0, 0],
        [sin_a, 1, 0],
        [0, 0, 1]
    ])
    combined_matrix = np.dot(makedon_matrix, owen_matrix)
    return apply_transformation(image, combined_matrix)


def apply_transformation(image, matrix):
    width, height = image.size
    src_coords = np.array([[x, y, 1] for y in range(height) for x
in range(width)])
    transformed_coords = np.dot(src_coords, matrix.T).astype(int)
    transformed_image = Image.new("RGB", (width, height))
    transformed_image_pixels = transformed_image.load()
    for (x, y, _), (tx, ty, _) in zip(src_coords,
transformed_coords):
        if 0 <= tx < width and 0 <= ty < height:
            transformed_image_pixels[tx, ty] = image.getpixel((x,
y))
    return transformed_image


# Интерфейс
root = tk.Tk()

original_label = tk.Label(root, text="Оригинальное изображение")
original_label.grid(row=0, column=0, pady=10)
original_canvas = tk.Canvas(root, width=200, height=200,
bg="white")
original_canvas.grid(row=1, column=0, padx=10, pady=10)

affine_label = tk.Label(root, text="Аффинное преобразование")
affine_label.grid(row=0, column=1, pady=10)
affine_canvases = [tk.Canvas(root, width=200, height=200,
```

```
bg="white") for _ in range(3)]
for i, canvas in enumerate(affine_canvases):
    canvas.grid(row=1, column=i + 1, padx=10, pady=10)

owen_label = tk.Label(root, text="Преобразование с матрицами Оуэна
и Македона")
owen_label.grid(row=2, column=0, pady=10)
owen_canvases = [tk.Canvas(root, width=200, height=200,
bg="white") for _ in range(3)]
for i, canvas in enumerate(owen_canvases):
    canvas.grid(row=3, column=i, padx=10, pady=10)

# Хранилище для изображений
affine_images_storage = [None] * 3
owen_images_storage = [None] * 3

load_button = tk.Button(root, text="Load Image",
command=load_image)
load_button.grid(row=4, column=1, columnspan=2, pady=20)

root.mainloop()
```