МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Кафедра систем управління літальними апаратами

Лабораторна робота № 6

з дисципліни «Об'єктно-орієнтоване проектування СУ»

Тема: «Розробка віконних додатків для завантаження і обробки растрових зображень»

ХАІ.301 .174. 322 .6 ЛР

Виконав студент гр. _____322_____

_____Безпалова С.В.___
(підпис, дата) (П.І.Б.)

Перевірив
_____ к.т.н., доц. О. В. Гавриленко
_____ ас. В. О. Білозерський
(підпис, дата) (П.І.Б.)

2024

МЕТА РОБОТИ

Отримати досвід роботи з навчальними матеріалами та документацією до бібліотек Pillow і OpenCV, і навчитися розробляти віконні додатки для завантаження з файлу, обробки різними способами, збереження і відображення у вікні фото-зображень.

ПОСТАНОВКА ЗАДАЧІ

Завдання 1. Вивчити документацію до бібліотеки Pillow і написати скрипт з визначенням класу, що реалізує користувальницький інтерфейс для виконання наступних функцій:

1) відкриття файлу із зображенням будь-якого допустимого графічного формату;

2) відображення зображення та інформації про формат;

3) * Установка значень для виконання функцій 4-5;

4) створення зменшеної копії вихідного зображення;

5) геометричні перетворення мініатюри, фільтрація, перетворення формату і вставка в вихідне зображення відповідно до варіанту (див. табл.1);

6) збереження зміненого зображення в фай і реалізацією роботи з об'єктом цього класу для запуску віконного програми.

Завдання 2. Вивчити документацію до бібліотеки OpenCV і написати скрипт з визначенням і роботою об'єктів класу, що реалізує користувальницький інтерфейс для виконання наступних функцій:

1) відкриття файлу із зображенням будь-якого допустимого графічного формату;

2) * Установка значень для виконання функцій 3-4;

3) зміна розмірів зображення;

4) геометричні перетворення зображення, зміна колірного простору, фільтрація і виконання операцій із зображенням відповідно до варіанту (див.

табл.2);

5) відображення вихідного зображення і після кожної зміни;

6) збереження змінених зображень у файли

і реалізацією роботи з об'єктом цього класу для запуску віконного програми.

ВИКОНАННЯ РОБОТИ

Завдання 1

Опис роботи програми

1. Імпорт бібліотек:

- tkinter.filedialog: використовується для роботи з діалоговими вікнами відкриття та збереження файлів.
- Pillow (PIL): для обробки зображень (відкриття, зміна, збереження).

2. Функціональні можливості програми:

- Відкриття зображення через діалогове вікно (підтримуються формати .png, .jpg, .jpeg, .bmp).
- Поворот зображення на кут, заданий користувачем.
- Застосування фільтра рельєфу (EMBOSS) для створення текстурного ефекту.
- Створення мініатюри (розміром 100x100 пікселів) та вставка її у правий нижній кут обробленого зображення.
- Збереження результату у форматі .png через діалогове вікно.

3. Ключові етапи виконання програми:

- Відкриття зображення: використовується діалогове вікно askopenfilename, у якому користувач вибирає початкове зображення.
- Обробка зображення:
- Поворот зображення здійснюється методом rotate() з урахуванням заданого кута.
- Фільтр рельєфу застосовується за допомогою методу filter() і фільтра ImageFilter.EMBOSS.
- Мініатюра створюється методом thumbnail(), який зменшує копію зображення до заданого розміру.
- Вставка мініатюри реалізована методом paste() з підтримкою прозорості (альфа-каналу).
- Збереження зображення: виконується через asksaveasfilename з можливістю вибору шляху та імені файлу.

4. Обробка помилок:

- Якщо файл не вибрано або обробка завершилася помилкою, програма виводить відповідне повідомлення.

Приклад роботи програми

- Користувач запускає програму.
- Вибирає зображення через діалогове вікно.
- Вводить кут повороту (наприклад, 45 градусів).
- Програма обробляє зображення:
- Повертає його.
- Застосовує фільтр рельєфу.
- Додає мініатюру у правий нижній кут.
- Користувач зберігає оброблене зображення через діалогове вікно.

Лістинг програми

```
    from tkinter.filedialog import askopenfilename, asksaveasfilename
    from PIL import Image, ImageFilter

    def process_image():
        # Открываем файл через диалоговое окно
        file_path = askopenfilename(filetypes=[("Image files",
"*.png;*.jpg;*.jpeg;*.bmp"), ("All files", "*.*")])
        if not file_path:
            print("Файл не выбран")
            return

        try:
            # Открытие изображения
            img = Image.open(file_path).convert("RGBA")  # Режим 32 бита с
альфа-каналом
            print(f"Размер: {img.size}, Формат: {img.format}")

            # Поворот изображения на произвольный угол
            angle = float(input("Введите угол поворота: "))
            img_rotated = img.rotate(angle, expand=True)

            # Применение фильтра EMBOSS
            img_filtered = img_rotated.filter(ImageFilter.EMBOSS)

            # Создание миниатюры
            thumbnail_size = (100, 100)
            thumbnail = img_filtered.copy()
            thumbnail.thumbnail(thumbnail_size)

            # Вставка миниатюры в правый нижний угол
            img_filtered.paste(thumbnail, (img_filtered.width -
thumbnail.width, img_filtered.height - thumbnail.height), thumbnail)
```

```
            # Сохранение результата
            save_path = asksaveasfilename(defaultextension=".png",
filetypes=[("PNG files", "*.png"), ("All files", "*.*")])
            if save_path:
                img_filtered.save(save_path)
                print(f"Изображение сохранено по пути: {save_path}")
            else:
                print("Сохранение отменено")

        except Exception as e:
            print(f"Ошибка обработки изображения: {e}")

    if __name__ == "__main__":
        process_image()
```

Опис роботи програми

1. **Основна ідея**:
   Програма дозволяє користувачеві обрати зображення, виконати його обробку (поворот, застосування фільтру рельєфу) і вставити мініатюру.
2. **Ключові етапи роботи**:
   - Вибір файлу через діалогове вікно (`askopenfilename`).
   - Введення кута повороту з клавіатури.
   - Обробка зображення (поворот, фільтр, мініатюра).
   - Збереження результату через діалогове вікно (`asksaveasfilename`).
3. **Особливості коду**:
   - Використовується бібліотека `Pillow` для роботи з графікою.
   - Підтримуються формати `.png`, `.jpg`, `.jpeg`, `.bmp`.
   - Мініатюра створюється методом `thumbnail()` і вставляється у правий нижній кут результату за допомогою `paste()`.
4. **Обробка помилок**:
   - Якщо користувач не вибрав файл, програма повідомить про це.
   - Усі помилки обробляються та відображаються у вигляді повідомлень.

Переваги програми

- Простий код, який легко розширити додатковими функціями.
- Зручний інтерфейс вибору файлів через діалогові вікна.
- Використання популярної бібліотеки `Pillow` для роботи з графікою.

Завдання 2.

Код програми

```
from PIL import Image, ImageFilter
import cv2
import numpy as np

def apply_transformations(input_image_path, output_image_path):
    # Відкриваємо зображення
    img = Image.open(input_image_path)

    # Перенос зображення
    numpy_image = np.array(img)
    rows, cols, _ = numpy_image.shape
    M_translation = np.float32([[1, 0, 50], [0, 1, 50]])  # Перенесення на 50
пікселів вправо і вниз
    translated_image = cv2.warpAffine(numpy_image, M_translation, (cols, rows))

    # Перетворення у LAB
    lab_image = cv2.cvtColor(translated_image, cv2.COLOR_RGB2LAB)

    # Застосування фільтра Превітта
    kernel_x = np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]])
    kernel_y = np.array([[1, 1, 1], [0, 0, 0], [-1, -1, -1]])
    prewitt_x = cv2.filter2D(lab_image, -1, kernel_x)
    prewitt_y = cv2.filter2D(lab_image, -1, kernel_y)
    prewitt_combined = cv2.add(prewitt_x, prewitt_y)

    # Конвертуємо назад для збереження
    final_image = Image.fromarray(cv2.cvtColor(prewitt_combined,
cv2.COLOR_LAB2RGB))

    # Зберігаємо результат
    final_image.save(output_image_path)

# Шлях до вхідного і вихідного зображень
input_image = "input_image.jpg"  # Вкажіть ім'я вашого файлу
output_image = "output_image.jpg"

# Виконуємо обробку
apply_transformations(input_image, output_image)

print("Зображення оброблено і збережено як", output_image)
```

Опис роботи програми

1. **Основна ідея**:
   Програма обробляє зображення з використанням бібліотек `OpenCV` та `Pillow`, виконуючи кілька етапів перетворення.
2. **Основні етапи обробки**:

- **Геометрична трансформація (перенос)**:
  Використовується матриця трансляції (`M_translation`), яка зміщує зображення на 50 пікселів вправо і вниз. Це досягається за допомогою функції `cv2.warpAffine()`.
- **Перетворення колірного простору**:
  Зображення з простору RGB перетворюється у LAB за допомогою `cv2.COLOR_RGB2LAB`. LAB простір розділяє інформацію про яскравість (L) і кольори (A, B), що спрощує подальшу обробку.
- **Фільтр Превітта**:
  Реалізується через згортку зображення з ядрами (`kernel_x` і `kernel_y`). Ці ядра обчислюють горизонтальні та вертикальні градієнти, що дозволяє виявити краї об'єктів. Результати по обох напрямках комбінуються у фінальний результат.
- **Збереження результату**:
  Оброблене зображення конвертується назад у формат RGB і зберігається у файлі.

3. **Особливості коду**:
   - Використання бібліотеки `OpenCV` для геометричних і колірних перетворень.
   - Використання бібліотеки `Pillow` для кінцевого збереження зображення.
   - Комплексний підхід до обробки країв через фільтр Превітта.
4. **Можливості розширення**:
   - Додавання інших фільтрів, таких як Собеля чи Лапласа.
   - Зміна параметрів трансляції (напрямок, величина).
   - Інтерактивний вибір параметрів через інтерфейс.

Висновок

У ході виконання лабораторної роботи було реалізовано програму для обробки зображень із використанням бібліотек `Pillow`, `OpenCV` та `NumPy`. Програма виконує такі основні операції:

1. Геометричну трансформацію (перенос зображення), що дозволяє змінювати положення об’єкта на зображенні.
2. Перетворення колірного простору з RGB у LAB, що спрощує аналіз та обробку кольорів.
3. Застосування фільтра Превітта для виявлення країв об'єктів, що є важливим етапом у задачах комп'ютерного зору.
4. Збереження обробленого зображення у файл для подальшого використання.

Програма продемонструвала можливості сучасних інструментів Python для обробки графічних даних. Було вивчено основи роботи з бібліотеками `OpenCV` та `Pillow`, зокрема операції згортки, колірні перетворення та геометричні трансформації.

Результати роботи демонструють, як за допомогою нескладного коду можна автоматизувати обробку зображень та покращити їх якість для подальшого аналізу чи використання.

У майбутньому дану програму можна розширити, додавши підтримку інших фільтрів, масштабування, обертання чи інтерактивний вибір параметрів. Це дозволить створити ще більш універсальний інструмент для обробки графічних даних.