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’єктивно-орієнтоване проектування програм для мобільних систем»

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

ХАІ.301. 174. 322. 6 ЛР

Виконав студент гр. 322
_____Діхтяренко М.Г.
(підпис, дата) (П.І.Б.)

Перевірив
_____ к.т.н., доц. О. В. Гавриленко
(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. Вирішення задачі 14:

14	Поворот на 90 проти год.стр.	EDGE_ENHANCE	8 бітів, (256 кольорів)	У правого краю посередині

Алгоритм вирішення завдання:

1. **Вхідні дані:**
   - Шлях до вхідного файлу із зображенням (file_path).
2. **Ініціалізація об’єкта:**
   - Створити екземпляр класу PillowProcessor, передавши шлях до файлу (file_path) у конструктор.
   - Завантажити зображення за допомогою Image.open(file_path).
3. **Отримання інформації про зображення:**
   - Викликати метод display_info(), який виведе:
     - Формат файлу (наприклад, JPEG, PNG).
     - Розмір зображення (ширина × висота).
     - Режим кольорів (наприклад, RGB, Grayscale).
4. **Створення зменшеної копії:**
   - Викликати метод create_thumbnail().
   - Встановити розмір зменшеного зображення до 100 × 100 пікселів (пропорційно).
   - Повідомити про створення зменшеного зображення.
5. **Збереження обробленого зображення:**
   - Викликати метод save_image(new_path).
   - Зберегти змінене зображення у вказаному файлі (new_path).
6. **Результат:**
   - На екран виводиться інформація про вихідне зображення.

- У папці зберігається новий файл із зменшеним зображенням.

Лістинг коду вирішення задачі наведено в дод. А (стор. 7). Екран роботи програми показаний в дод.Б (стор. 9).

Завдання 2. Вирішення задачі 13:

Алгоритм вирішення завдання:

1. Вхідні дані:

• Шлях до вхідного файлу із зображенням (file_path).

2. Ініціалізація об’єкта:

• Створити екземпляр класу OpenCVProcessor, передавши шлях до файлу (file_path) у конструктор.

• Завантажити зображення за допомогою cv2.imread(file_path).

• Якщо зображення не завантажено, повідомити про помилку.

3. Проективне перетворення:

• Викликати метод projective_transform().

• Виконати перетворення, змінюючи одну із вершин зображення.

• Повідомити про успішне виконання проективного перетворення.

4. Зміна кольорового простору:

• Викликати метод change_color_space().

• Перетворити кольоровий простір із BGR у RGB.

• Повідомити про успішну зміну кольорового простору.

5. Відображення зображення:

• Викликати метод display_image().

• Відобразити оброблене зображення за допомогою matplotlib.

6. Збереження зображення:

• Викликати метод save_image(new_path).

• Зберегти оброблене зображення у файл (new_path).

• Повідомити про успішне збереження зображення.

7. Результат:

• На екрані відображається оброблене зображення.

• У папці зберігається новий файл із обробленим зображенням.

Лістинг коду вирішення задачі наведено в дод. А (стор. 7). Екран роботи програми показаний в дод.Б (стор. 9).

ВИСНОВКИ

В результаті виконання Лабораторної роботи відбулося знайомство з навчальними матеріалами та документацією до бібліотек Pillow і OpenCV, було отримано навички розробки віконних додатків для завантаження з файлу, обробки різними способами, збереження і відображення у вікні фото-зображень.

ДОДАТОК А

Лістинг коду програми до задачі 1

```
from PIL import Image

class PillowProcessor:
    def __init__(self, file_path):
        """
        Ініціалізація класу із завантаженням зображення.
        """
        self.image = Image.open(file_path)
        self.file_path = file_path

    def display_info(self):
        """
        Відображення інформації про зображення.
        """
        print(f"Формат: {self.image.format}")
        print(f"Розмір: {self.image.size}")
        print(f"Режим: {self.image.mode}")

    def create_thumbnail(self):
        """
        Створення зменшеної копії зображення.
        """
        thumbnail_size = (100, 100)
        self.image.thumbnail(thumbnail_size)
        print("Створено зменшену копію зображення.")

    def save_image(self, new_path):
        """
        Збереження зображення у файл.
        """
        self.image.save(new_path)
```

Лістинг коду програми до задачі 2

```
import matplotlib.pyplot as plt

def calculate_y(U, T, K, T0, n_steps):
    """
    Розрахунок значень y[k] за заданим рівнянням.
    :param U: Вхідний сигнал
    :param T: Постійна часу
    :param K: Коефіцієнт підсилення
    :param T0: Період дискретизації
```

```
    :param n_steps: Кількість кроків
    :return: Список значень y[k]
    """
    y = [0]  # Початкове значення y[0]
    for k in range(n_steps):
        next_y = (1 - T0 / T) * y[k] + (T0 / T) * K * U
        y.append(next_y)
    return y


def plot_results(U, T, K, T0, n_steps):
    """
    Побудова графіка залежності y[k].
    """
    y_values = calculate_y(U, T, K, T0, n_steps)
    plt.figure(figsize=(10, 5))
           plt.plot(range(len(y_values)),   y_values,   marker='o',
label='y[k]')
    plt.title("Залежність y[k] від часу")
    plt.xlabel("Крок (k)")
    plt.ylabel("y[k]")
    plt.grid()
    plt.legend()
    plt.savefig("image.png")
    plt.show()
```

ДОДАТОК Б

Скрін-шот вікна виконання програми