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 програмування авіаційно-транспортних систем»

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

ХАІ.301.173.320.03 ЛР

Виконав студент гр. _____320_____

_____Полякова Софія___
(підпис, дата) (П.І.Б.)

Перевірив
_____ к.т.н., доц. О. В. Гавриленко
_____ ас. Є. В. Пявка
(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;

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

3.	Вертикальний дзеркальний образ	DETAIL	8 бітів, (256 кольорів)	У нижнього краю по центру

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;

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.

24.	зсув (Вертикальний)	з підкресленням кордонів	XYZ	розмивання

ВИКОНАННЯ РОБОТИ

Завдання 1. Вирішення задачі Pillow_3

Вхідні дані (ім’я, опис, тип, обмеження):

image_path — основне зображення, формат .jpg.

Вихідні дані (ім’я, опис, тип):

save_path — збережене перетворене зображення, формат .jpg або .png.

Алгоритм вирішення показано на рис.1-2.

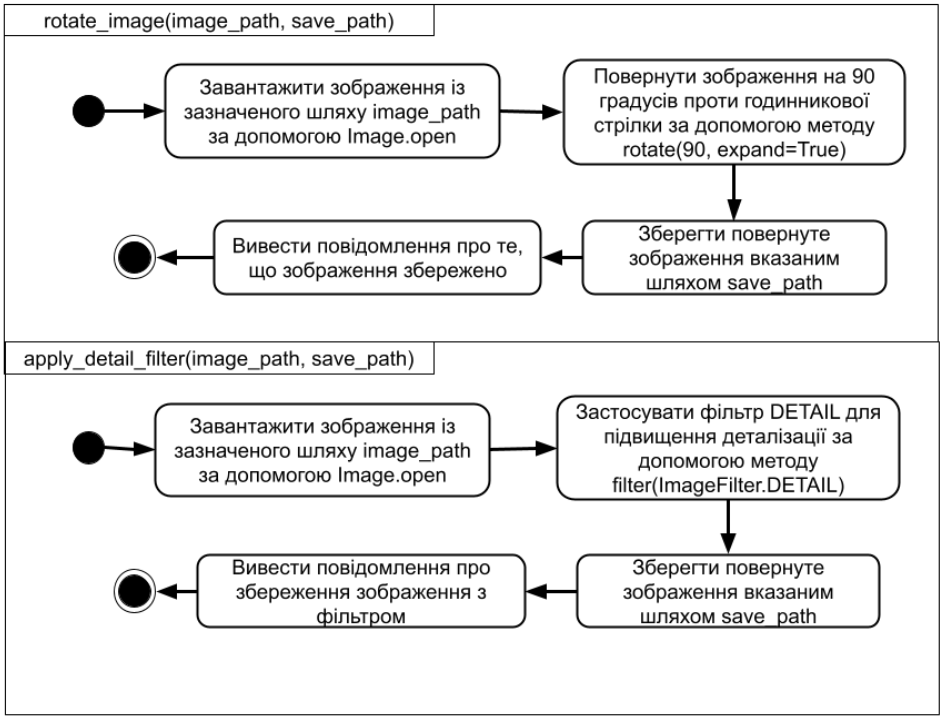


Рисунок 1 – Алгоритм вирішення задачі Pillow_3
(два завдання: повернення на 90 градусів проти годинникової стрілки та фільтр)

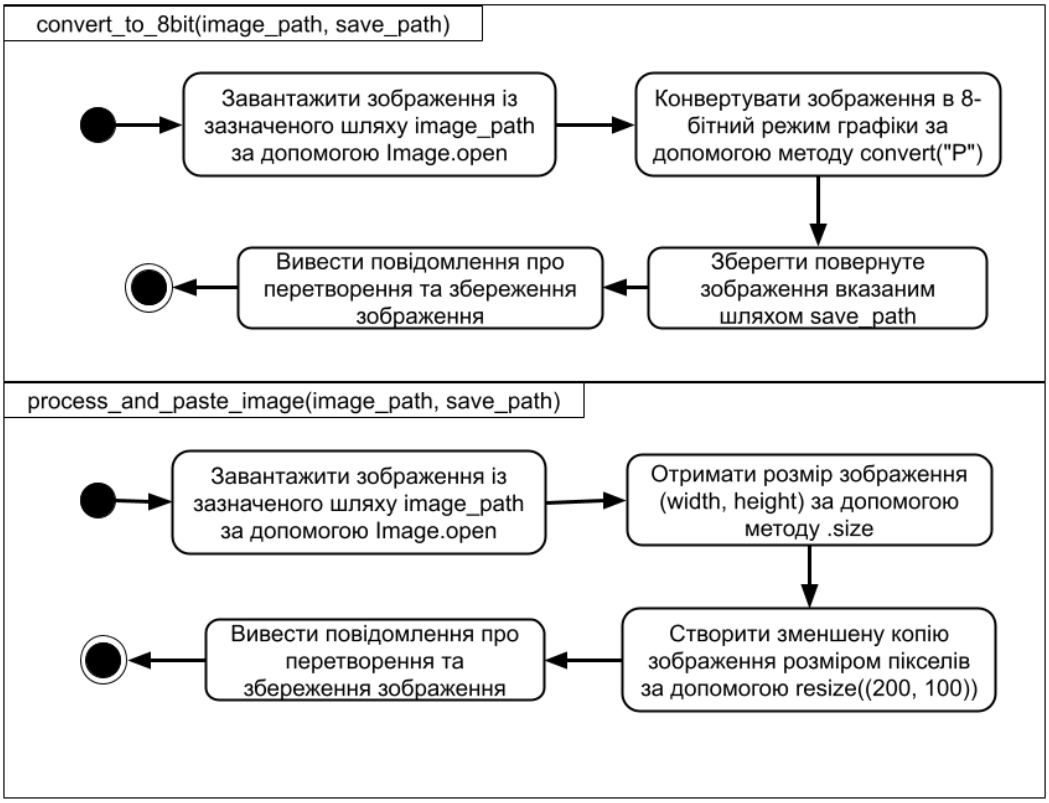


Рисунок 2 – Алгоритм вирішення задачі Pillow_3
(два завдання: переведення в 8-біт та вставка мініатюри)

Лістинг коду вирішення задачі наведено в дод. А (стор. 8).
Екран роботи програми показаний на рис. В.1-4.

Завдання 2. Вирішення задачі OpenCV_23

Вхідні дані (ім’я, опис, тип, обмеження):

image — основне зображення, формат .jpg.

Вихідні дані (ім’я, опис, тип):

1. shifted_image — зміщення зображення по вертикалі, формат .jpg;
2. filtered_image — зображення з фільтром верхніх частот, формат .jpg;
3. xyz_image — зображення з конвертацією до системи XYZ, формат .jpg;
4. roberts_image — зображення з фільтром Робертса, формат .jpg.

Алгоритм вирішення показано на рис.3-5.

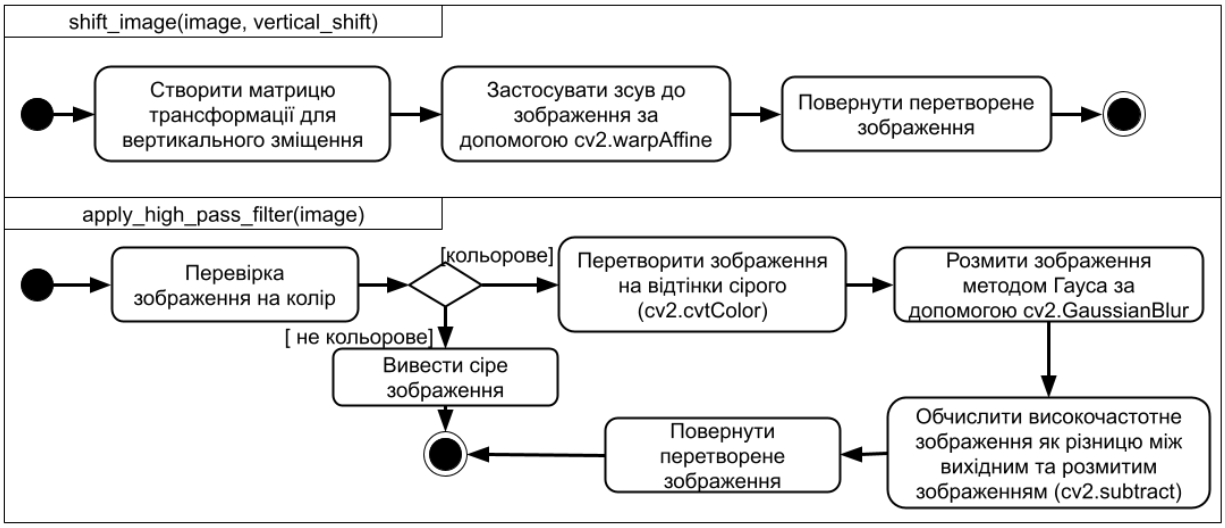


Рисунок 3 – Алгоритм вирішення задачі OpenCV_23
(два завдання: вертикальне зміщення та з фільтром верхніх частот)

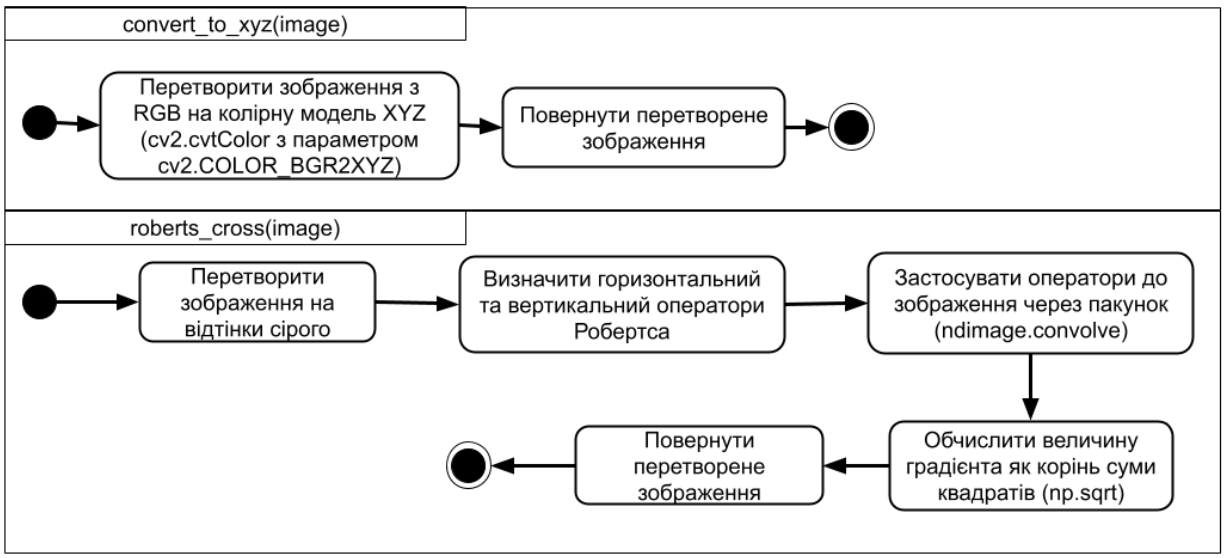


Рисунок 4 – Алгоритм вирішення задачі OpenCV_23
(два завдання: конвертування у XYZ та перетворення Робертса)

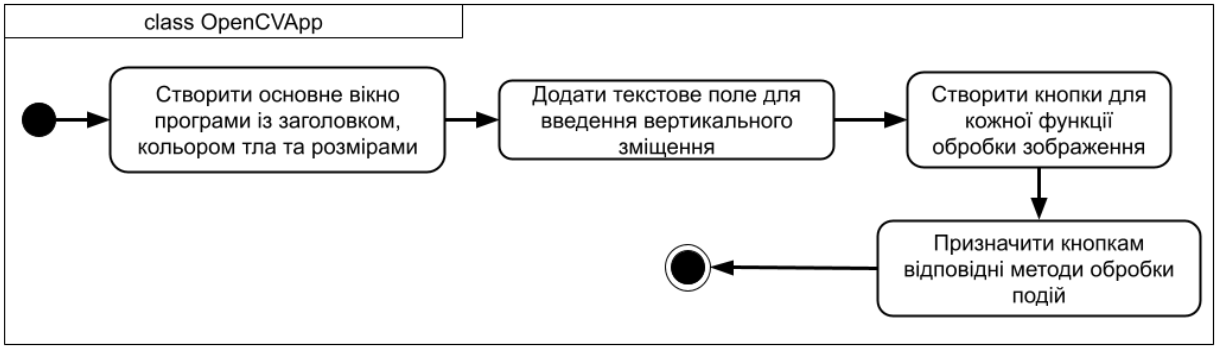


Рисунок 5 – Алгоритм вирішення задачі OpenCV_23 (створення інтерфейсу)

Лістинг коду вирішення задачі наведено в дод. Б (стор. 10).

Екран роботи програми показаний на рис. В.5-11.

ВИСНОВКИ

В ході лабораторної роботи було вивчено навчальні матеріали та документації до бібліотек Pillow і OpenCV. Отримано навички обробки файлів різними способами та збереження зображень у файли. Відпрацьовано на коді різні фільтри для зображень, створення мініатюр, повернення та зміщення зображень. Закріплено на практиці створення інтерфейсу з кнопками та полем для вводу даних.

ДОДАТОК А
Лістинг коду програми до задач Pillow_3

```
from PIL import Image, ImageFilter, ImageDraw, ImageFont

def rotate_image(image_path, save_path):
    # uploading an image
    image = Image.open(image_path)

    # turn 90 degrees counter-clockwise
    rotated_image = image.rotate(90, expand=True)

    # saving the rotated image
    rotated_image.save(save_path)
    print(f"The returned image is saved to: {save_path}")

def apply_detail_filter(image_path, save_path):
    # uploading an image
    image = Image.open(image_path)

    # applying the DETAIL filter
    detailed_image = image.filter(ImageFilter.DETAIL)

    # saving an image with a filter
    detailed_image.save(save_path)
    print(f"Image with DETAIL filter saved at: {save_path}")


def convert_to_8bit(image_path, save_path):
    # uploading an image
    image = Image.open(image_path)

    # convert to 8-bit graphics mode
    image_8bit = image.convert("P")

    # save the new image
    image_8bit.save(save_path)
    print(f"Image converted to 8-bit mode and saved at: {save_path}")

def process_and_paste_image(image_path, save_path):
            try:
                # uploading an image
                image = Image.open(image_path)
                print("Main Image Size:", image.size)
                width, height = image.size
                # creating mini image
                img1 = image.resize((200, 100))

                x_offset = (width - 200) // 2
                y_offset = height - 100
```

```
                image.paste(img1, (x_offset, y_offset))

                image.save(save_path)
                print(f"The processed image is saved as{save_path}")
            except FileNotFoundError:
                print(f"Error: File {image_path} is not found.")
            except Exception as e:
                print(f"An error: {e}")


# using functions
rotate_image('plane.jpg', 'rotated_image.jpg') # returned image
apply_detail_filter('plane.jpg', 'detailed_image.jpg') # Image with filter
convert_to_8bit('plane.jpg', '8bit_image.png') # imagen coverted to 8-bit mode
process_and_paste_image('plane.jpg', 'processed_image.jpg') # Image resized and
inserted
```

ДОДАТОК Б
Лістинг коду програми до задач OpenCV_23

```
import tkinter as tk
from tkinter import messagebox
import cv2
import numpy as np
from scipy import ndimage
# vertical shift
def shift_image(image, vertical_shift):
        translation_matrix = np.float32([[1, 0, 0], [0, 1, vertical_shift]])
        shifted_image = cv2.warpAffine(image, translation_matrix,
(image.shape[1], image.shape[0]))
        return shifted_image
# high pass filter
def apply_high_pass_filter(image):
        if len(image.shape) == 3:
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        else:
            gray = image

        blurred = cv2.GaussianBlur(gray, (15, 15), 0)
        high_pass = cv2.subtract(gray, blurred)
        return high_pass
# convert to xyz
def convert_to_xyz(image):
        xyz_image = cv2.cvtColor(image, cv2.COLOR_BGR2XYZ)
        return xyz_image
# roberts cross
def roberts_cross(image):
        img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        gx = np.array([[1, 0], [0, -1]])
        gy = np.array([[0, 1], [-1, 0]])

        gradient_x = ndimage.convolve(img, gx)
        gradient_y = ndimage.convolve(img, gy)

        magnitude = np.sqrt(gradient_x**2 + gradient_y**2)
        return magnitude

class OpenCVApp:
        def __init__(self, root, image):
            self.root = root
            self.root.title("Image Processor - OpenCV")
            self.root.geometry("600x600")
            self.root.configure(bg="light blue")
            self.image = image
            # Data entry field for vertical offset
            self.shift_label = tk.Label(root, text="Enter the offset (in
pixels):", bg = "light blue")
```

```
            self.shift_label.pack()

            self.shift_entry = tk.Entry(root, width=10)
            self.shift_entry.pack()

            # Buttons
            self.shift_button = tk.Button(root, text="Move Image", bg="pink",
command=self.on_shift_button_click)
            self.shift_button.pack()

            self.high_pass_button = tk.Button(root, text="High Pass Filter",
bg="pink", command=self.on_high_pass_button_click)
            self.high_pass_button.pack()

            self.xyz_button = tk.Button(root, text="Convert to XYZ",  bg="pink",
command=self.on_xyz_button_click)
            self.xyz_button.pack()

            self.roberts_button = tk.Button(root, text="Roberts Transform",
bg="pink", command=self.on_roberts_button_click)
            self.roberts_button.pack()

        def on_shift_button_click(self):
            try:
                vertical_shift = int(self.shift_entry.get())
            except ValueError:
                messagebox.showerror("Error", "Enter a valid numeric value for
the offset")
                return

            shifted_image = shift_image(self.image, vertical_shift)
            cv2.imwrite('shifted.jpg', shifted_image)
            messagebox.showinfo("Success", "The shifted image is saved as
'shifted.jpg'.")

        def on_high_pass_button_click(self):
            filtered_image = apply_high_pass_filter(self.image)
            cv2.imwrite('filtered.jpg', filtered_image)
            messagebox.showinfo("Success", "The filtered image is saved as
'filtered.jpg'.")

        def on_xyz_button_click(self):
            xyz_image = convert_to_xyz(self.image)
            cv2.imwrite('xyz.jpg', xyz_image)
            messagebox.showinfo("Success", "An image in the XYZ color model is
saved as 'xyz.jpg'.")

        def on_roberts_button_click(self):
            roberts_image = roberts_cross(self.image)
            cv2.imwrite('roberts.jpg', roberts_image)
            messagebox.showinfo("Success", "The image after Roberts Transform is
```

```
saved as 'roberts.jpg'.")

if __name__ == "__main__":
        img = cv2.imread('Wall-e.jpg')

        if img is None:
            print("Error: Image not found. Check the file path.")
        else:
            root = tk.Tk()
            app = OpenCVApp(root, img)
            root.mainloop()
```

ДОДАТОК В

Скрін-шоти вікна виконання програми

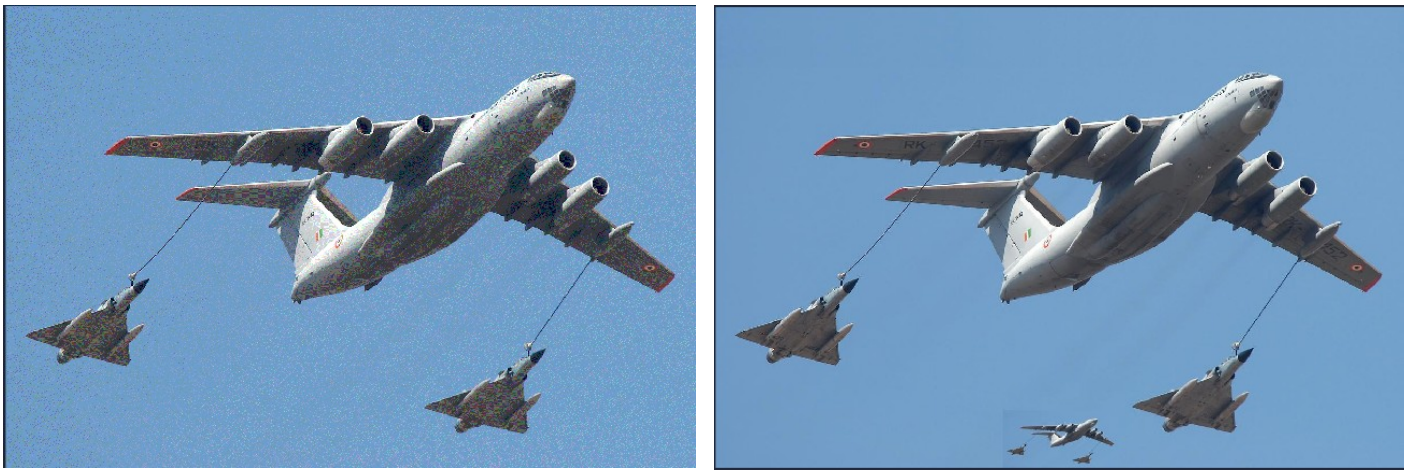
```
The returned image is saved to: rotated_image.jpg
Image with DETAIL filter saved at: detailed_image.jpg
Image converted to 8-bit mode and saved at: 8bit_image.png
Main Image Size: (1200, 800)
The processed image is saved asprocessed_image.jpg
```

Рисунок В.1 – Екран виконання програми для вирішення завдання Pillow_3

Рисунок В.2 – Основне зображення програми для вирішення завдання Pillow_3

а б

Рисунок В.3 – Перетворені зображення програми Pillow_3
(а — зображення з фільтром, б — перевернуте зображення)



а б

Рисунок В.4 – Перетворені зображення програми Pillow_3
(а — 8-бітне зображення, б — зображення з мініатюрою)

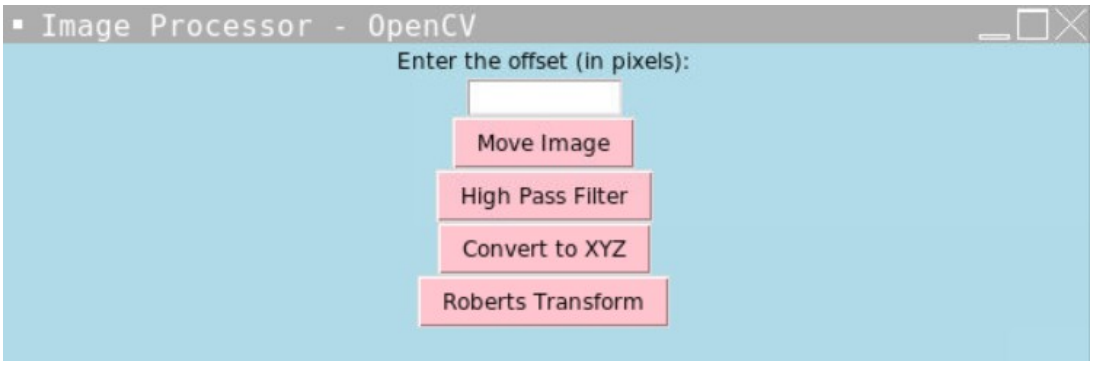
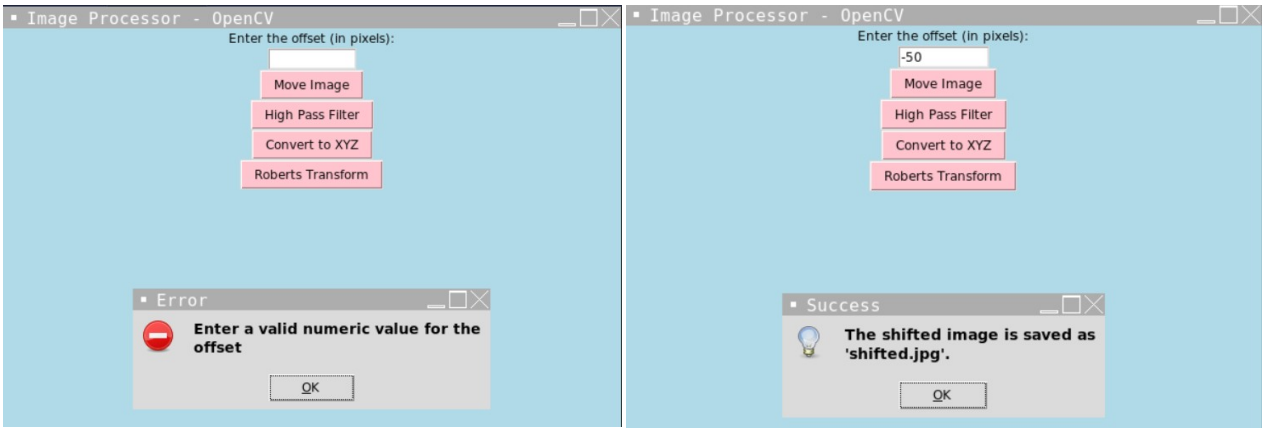


Рисунок В.5 – Екран інтерфейсу програми для вирішення завдання OpenCV_23

Рисунок В.6 – Основне зображення програми для вирішення завдання OpenCV_23



а б

Рисунок В.7 – Робота інтерфейсу програми OpenCV_23
(а — виведення помилки, б — збереження зміщеного зображення)

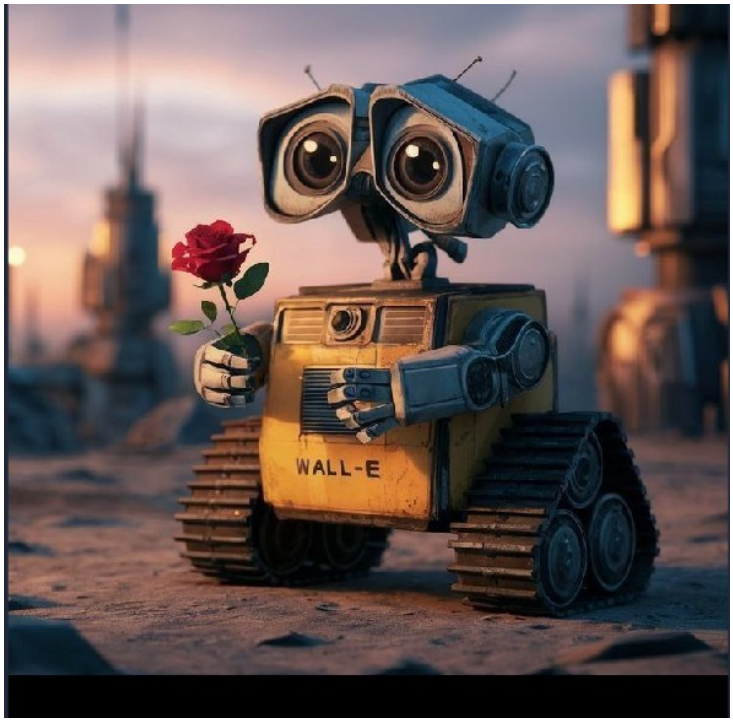
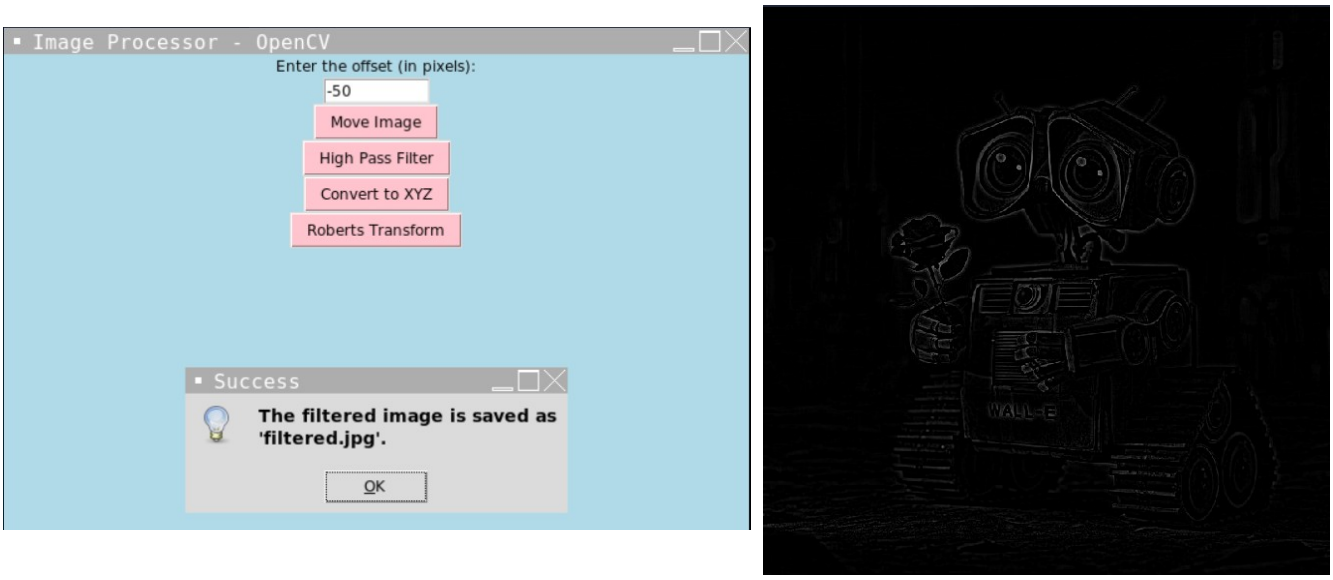
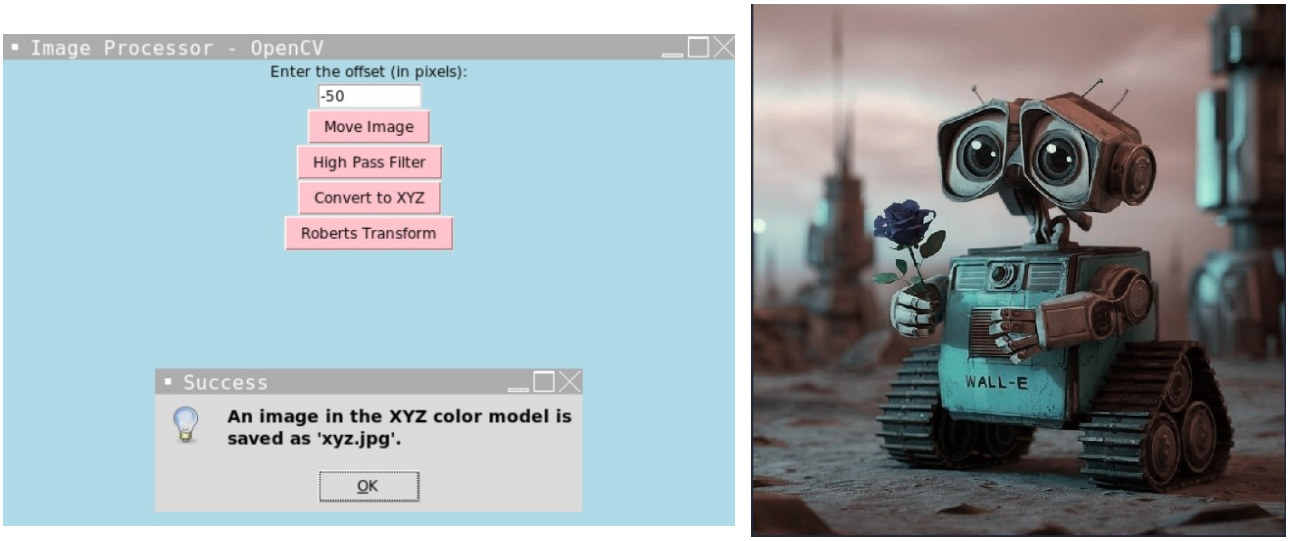


Рисунок В.8 – Перетворене зображення програми OpenCV_23
(зміщення по вертикалі)



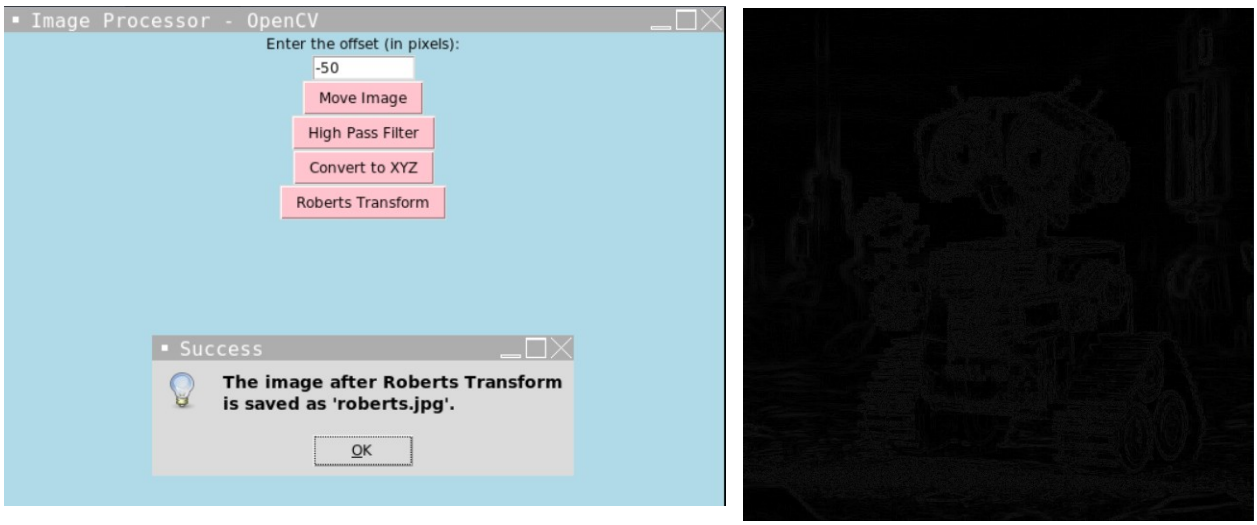
а б

Рисунок В.9 – Робота інтерфейсу та перетворене зображення програми OpenCV_23
(а — екран інтерфейсу, б — фільтроване зображення)



а б

Рисунок В.10 – Робота інтерфейсу та перетворене зображення програми OpenCV_23
(а — екран інтерфейсу, б — зображення в XYZ)



а б

Рисунок В.11 – Робота інтерфейсу та перетворене зображення програми OpenCV_23

(а — екран інтерфейсу, б — зображення з фільтром Робертса)