Міністерство освіти і науки України
Національний технічний університет України «КПІ» імені Ігоря Сікорського
Кафедра обчислювальної техніки ФІОТ

ЗВІТ
з лабораторної роботи №2
з навчальної дисципліни «Computer Vision»

Тема:

ДОСЛІДЖЕННЯ АЛГОРИТМІВ ФОРМУВАННЯ ТА ОБРОБКИ РАСТРОВИХ ЦИФРОВИХ ЗОБРАЖЕНЬ

Виконав:
Студент 3 курсу кафедри ІПІ ФІОТ,
Навчальної групи ІП-11
Панченко С.В.

Перевірив:
Професор кафедри ОТ ФІОТ
Писарчук О.О.

Київ 2024

I. Мета роботи:

Виявити дослідити та узагальнити особливості реалізації алгоритмів растрової цифрових зображень на прикладі застосування алгоритмів растеризації, побудови складних 3D растрових об’єктів та застосування технологій корекції характеристик кольору окремих растрів цифрових зображень.

II. Завдання

Лабораторія провідної IT-компанії реалізує масштабний проект розробки універсальної платформи з цифрової обробки зображень для задач Computer Vision. Платформа передбачає розташування back-end компоненти на власному хмарному сервері з наданням повноважень користувачам заздалегідь адаптованого front-end функціоналу універсальної платформи. Цим формується унікальна для потреб замовника ERP систем з технологіями Computer Vision. Змовниками ресурсів платформи є: державні та комерційні компанії, що розробляють медичне обладнання з діагностування захворювань за візуальною інформацією; автоматизації аграрного бізнесу в аспекті обліку посівних територій за даними з БПЛА; візуального контролю безпекових заходів на об’єктах критичної інфраструктури: аеропорти, торгівельно-розважальні центри, житлові комплекси тощо.

Вам, як Computer Vision поставлено завдання.

Завдання III рівня – максимально 9 балів.

Реалізувати розробку програмного скрипта, що реалізує корекцію кольору цифрового
растрового зображення з переліку: зміна яскравості, відтінки сірого, негатив, серпія – в градієнтах: діагональ (будь-який напрям); від центру; до центра. Обробку реалізувати на рівні матриці растра. Зображення обрати самостійно.

III. Результати виконання лабораторної роботи.

3.1. Синтезована математична модель перетворень графічних об’єктів відповідно до індивідуального завдання.

Створено математичну модель, яка відтворює операції, зазначені в умові завдання, над вхідним графічним об'єктом. Ця модель здійснює такі дії як додавання шуму, розмиття, зменшення і збільшення яскравості, а також створення негативу для растрового об'єкту.

Додання шуму $R/G/B(x,y)$ (де x, y – це положення пікселя на растрі) досягається шляхом додавання до початкових значень RGB пікселів $R/G/B(x,y)$ випадкових величин, згенерованих з використанням Гаусівського шуму. $N(0,\sigma^2)$, Тутσ^2 – відповідає за середнє квадратичне відхилення, а 0 – за середнє значення:

$$\begin{bmatrix} R_{noisy}(x,y) \\ G_{noisy}(x,y) \\ B_{noisy}(x,y) \end{bmatrix} = \begin{bmatrix} R(x,y) \\ G(x,y) \\ B(x,y) \end{bmatrix} + N(0,\sigma^2)$$

Розмиття растру $R_{blurred}/G_{blurred}/B_{blurred}(x,y)$ здійснюється шляхом застосування формули Гаусівського розмиття до кожного значення RGB в растрі:

$$\begin{bmatrix} R_{blurred}(x,y) \\ G_{blurred}(x,y) \\ B_{blurred}(x,y) \end{bmatrix} = \frac{1}{2\pi\sigma^2} \begin{bmatrix} R(x,y)\exp\left(\frac{-x^2+y^2}{2\sigma^2}\right) \\ G(x,y)\exp\left(\frac{-x^2+y^2}{2\sigma^2}\right) \\ B(x,y)\exp\left(\frac{-x^2+y^2}{2\sigma^2}\right) \end{bmatrix}$$

Негатив растру $R_{\neg/G_{\neg/B_{\neg,x,y}}}$ досягається шляхом обернення кожного значення RGB для кожного пікселя, що означає вибір значення з протилежного кінця простору RGB. Наприклад, якщо всі значення RGB знаходяться в межах [0; 255], то негативом растру буде:

$$\begin{bmatrix} R_{neg}(x,y) \\ G_{neg}(x,y) \\ B_{neg}(x,y) \end{bmatrix} = \begin{bmatrix} 255 - R(x,y) \\ 255 - G(x,y) \\ 255 - B(x,y) \end{bmatrix}$$

Зміною яскравості $R_{br}/G_{br}/B_{br}(x,y)$ Це включає множення кожного значення в растрі на певне число S. Зменшення яскравості або затемнення відбувається, коли число S знаходиться у діапазоні (0; 1). Збільшення яскравості або освітлення відбувається, коли число S знаходиться у діапазоні (1; +∞).

$$\begin{bmatrix} R_{br}(x,y) \\ G_{br}(x,y) \\ B_{br}(x,y) \end{bmatrix} = S \begin{bmatrix} R(x,y) \\ G(x,y) \\ B(x,y) \end{bmatrix}$$

3.2. Блок схема алгоритму та її опис.

Блок-схема розв’язку матиме вигляд на рис. 1.

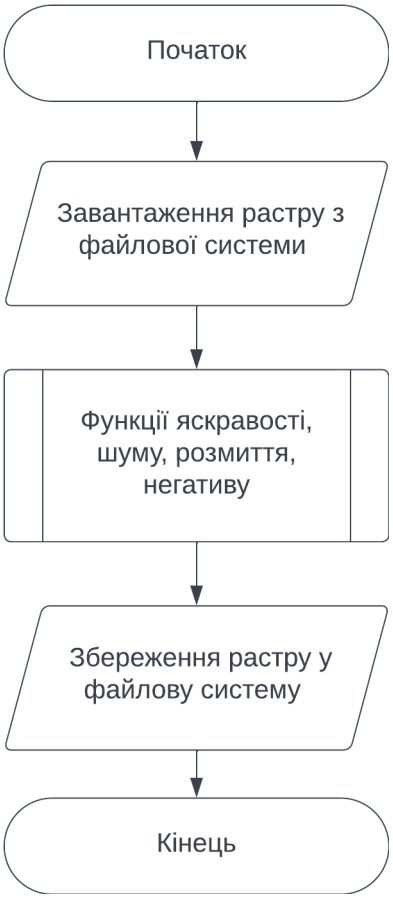

Рис. 1. Блок-схема алгоритму

У вікні десктопного інтерфейсу робота програми організована через кнопки, які виконують такі функції:

1. Розмиття
2. Збереження відредагованого растру
3. Зменшення яскравості
4. Завантаження растру
5. Негатив
6. Додання шуму
7. Підвищення яскравості

3.3. Опис структури проекту програми в середовищі PyCharm.

Для втілення розробленого алгоритму у мові програмування Python за допомогою можливостей інтегрованого середовища PyCharm був створений проект. Цей проект ґрунтується на принципах лінійної бізнес-логіки функціонального програмування та має наступну структуру.

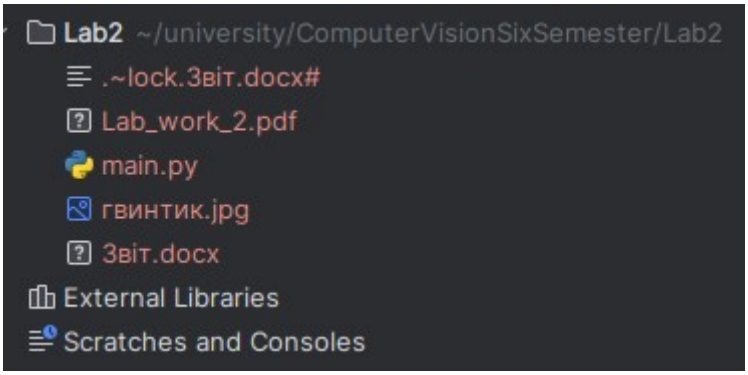

Рис.2. Структура проекту.

Lab_work_2.pdf – завдання лабораторної роботи
Звіт.docx – звіт лабораторної роботи
гвинтик.jpg — приклад рисунку
main.py – файл з математичним функціоналом у TKinter

3.4. Результати роботи програми відповідно до завдання.

1. Програма генерує десктопний застосунок з функціями, які можна виконувати у будь-якому порядку та навіть комбінувати. Нижче наведено послідовність виконання основних операцій над растром.

2. Початковий екран на рис. 3:

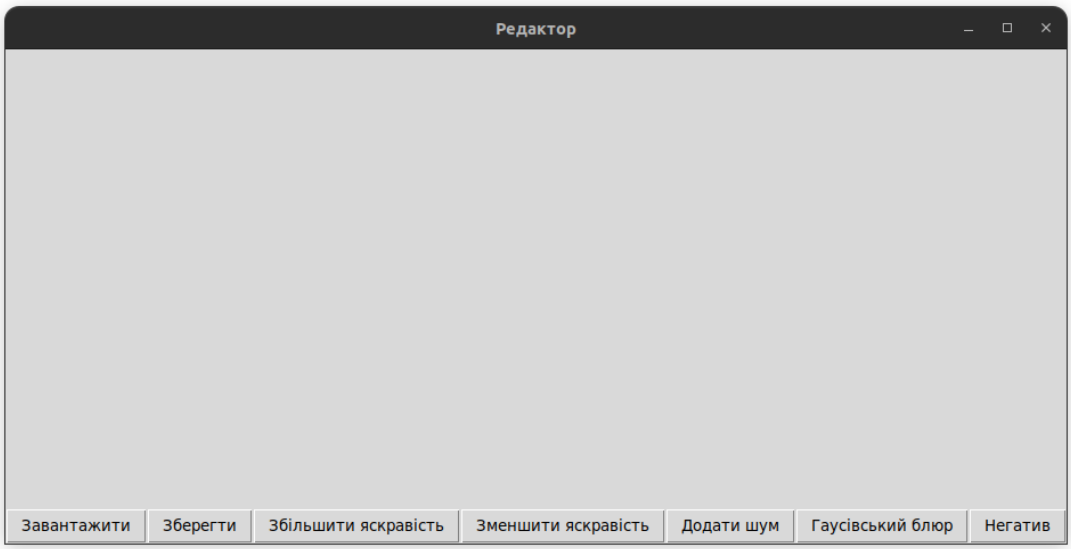

Рис. 3. Початковий екран

3. При обиранні кнопки “Завантажити”, обираємо картинку з файлу:

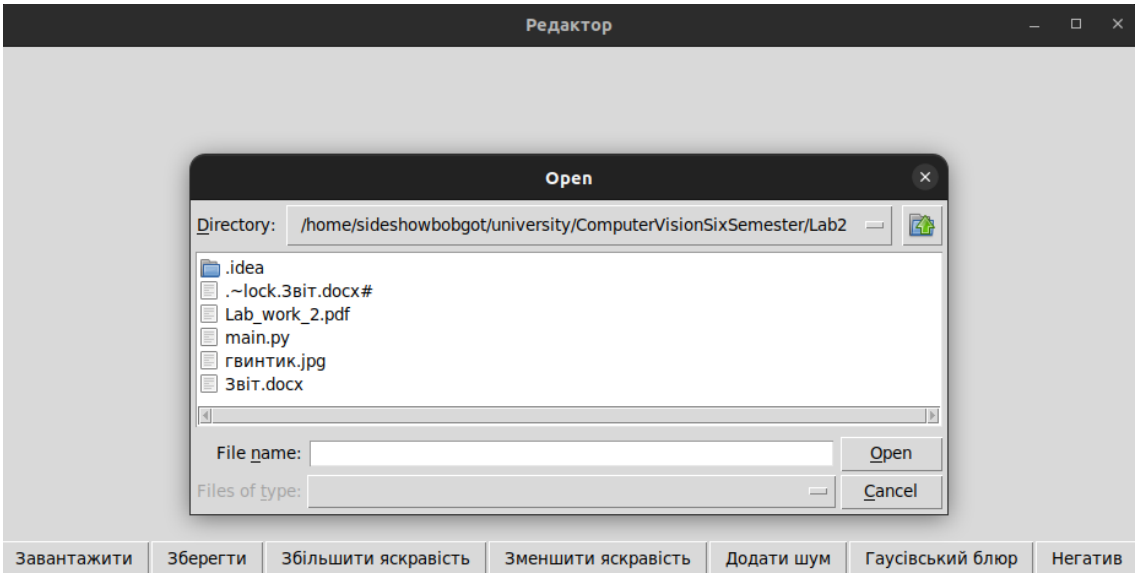

Рис. 4. Обирання растру з файлової системи.

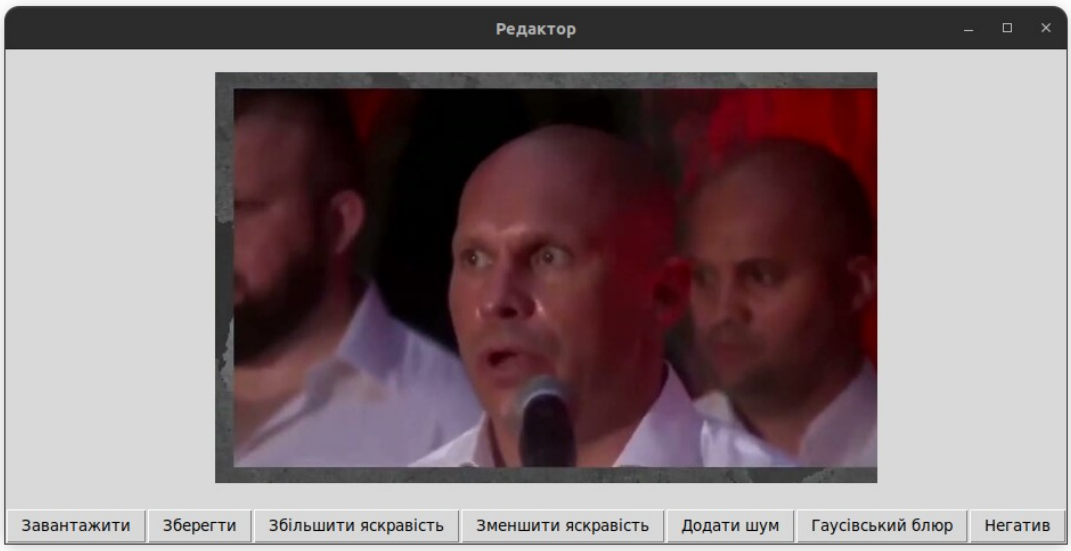

Рис. 5. Редактор після обрання картинки

4. При обранні кнопок:

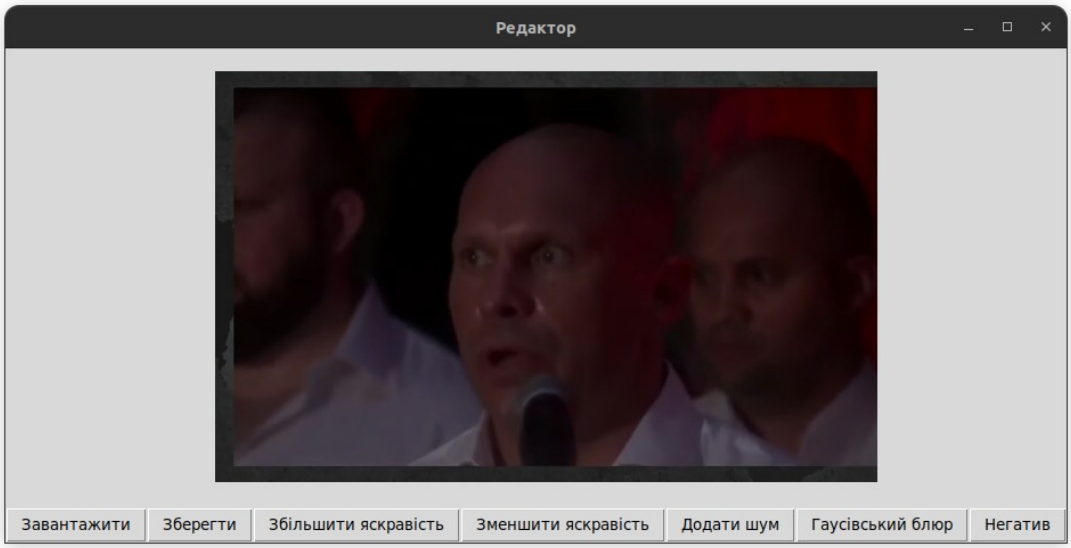

Рис. 6. Зменшення яскравості

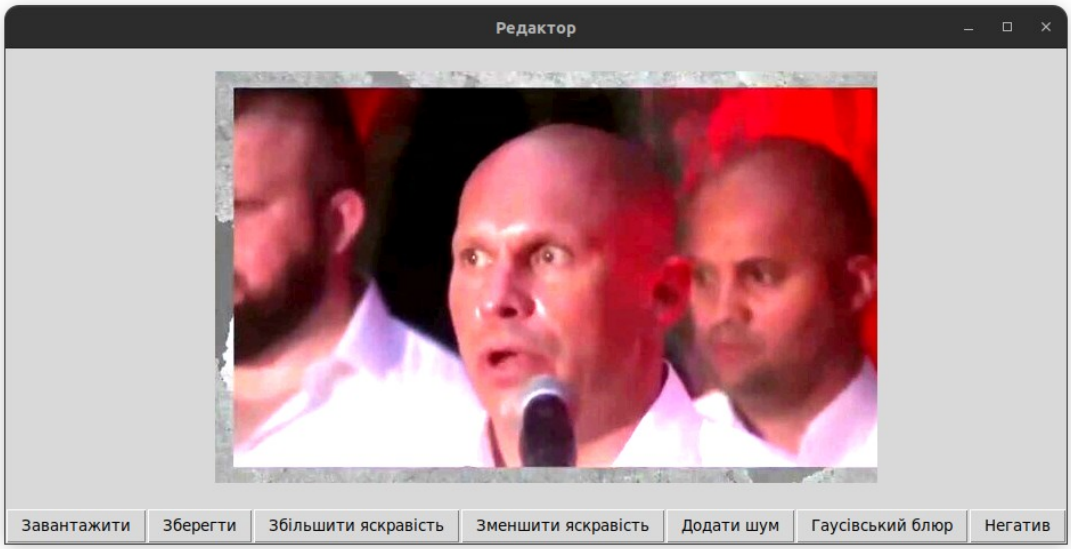

Рис. 7. Збільшення яскравості

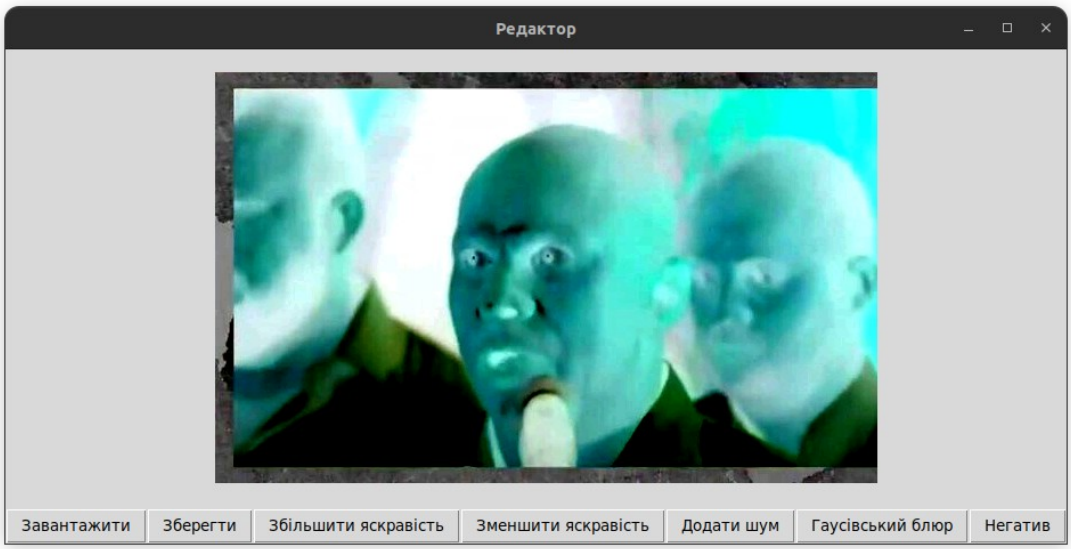

Рис. 8. Негативна фотографія

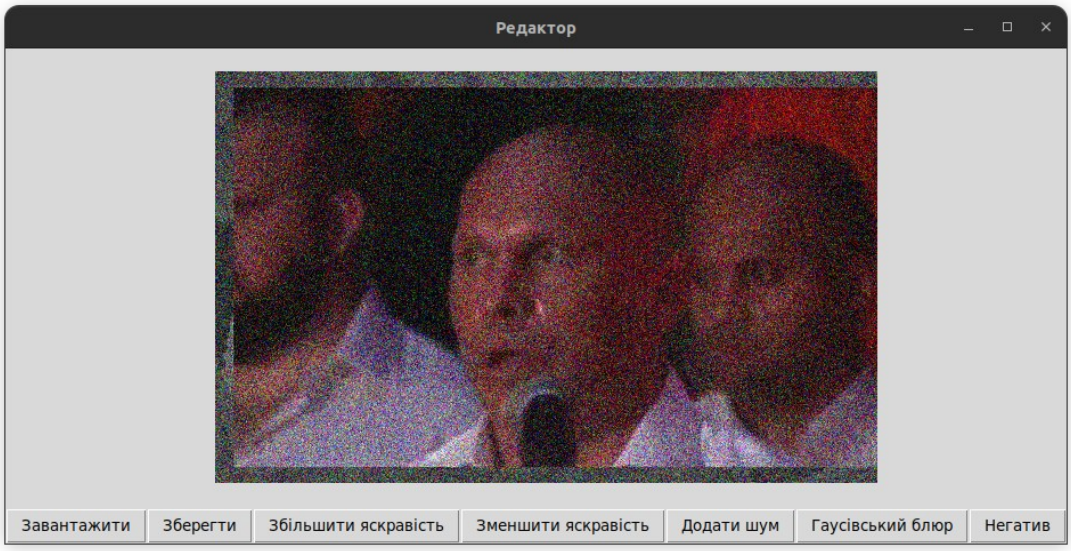

Рис. 9. Додання шуму до фотографії

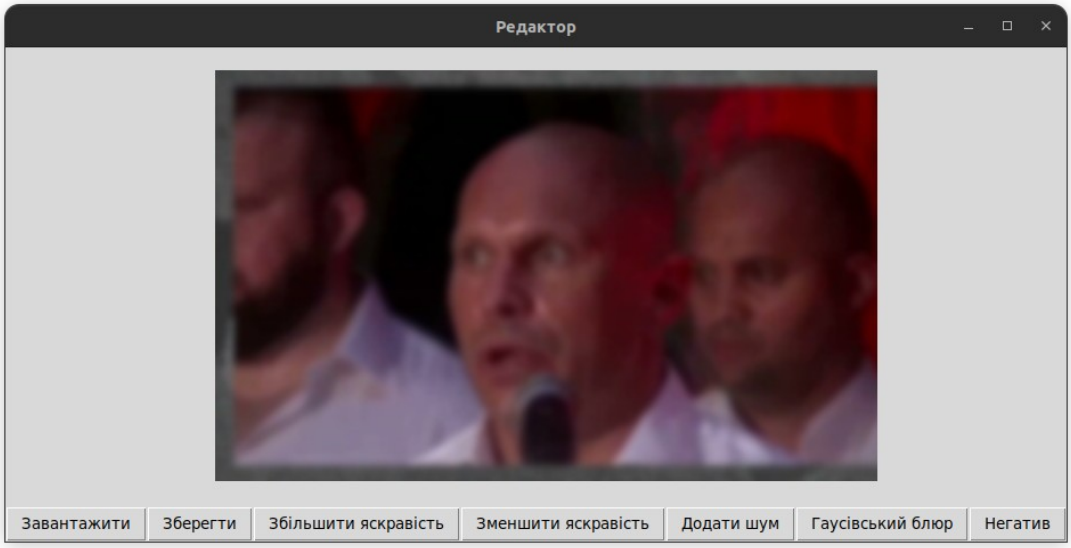

Рис. 10. Додання блюру до растру

3.5. Програмний код.

Програмний код систематично втілює алгоритм, який показаний на рисунку 1 та спрямований на досягнення результатів, зображених на рисунках 2-10. Для спрощення програмного коду і оптимізації обчислень використано функціональні механізми створення підпрограм. Обчислення проводились з розширеною матрицею координат паралелепіпеду. При цьому використано можливості бібліотек Python: numpy, tkinter, PIL.

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

def load_image():
```

```
    global img, tk_img
    file_path = filedialog.askopenfilename()
    if file_path:
        img = Image.open(file_path).convert('RGB')
        tk_img = ImageTk.PhotoImage(img)
        canvas.itemconfig(image_container, image=tk_img)

def save_image():
    file_path = filedialog.asksaveasfilename(defaultextension=".png")
    if file_path:
        img.save(file_path)

def update_image(new_img):
    global img, tk_img
    img = new_img
    tk_img = ImageTk.PhotoImage(new_img)
    canvas.itemconfig(image_container, image=tk_img)

def change_brightness(factor):
    # Ensure img is a numpy array
    img_array = np.array(img, dtype=np.float32)
    img_array *= factor
    img_array = np.clip(img_array, 0, 255).astype(np.uint8)
    update_image(Image.fromarray(img_array))

def add_noise(sigma):
    img_array = np.array(img)
    noise = np.random.normal(0, sigma, img_array.shape)
    noisy_img_array = img_array + noise
    noisy_img_array = np.clip(noisy_img_array, 0, 255).astype(np.uint8)
    update_image(Image.fromarray(noisy_img_array))

def gaussian_blur(radius):
    blurred_img = img.filter(ImageFilter.GaussianBlur(radius))
    update_image(blurred_img)

def make_negative():
    img_array = np.array(img)
    negative_img_array = 255 - img_array
    update_image(Image.fromarray(negative_img_array))

app = tk.Tk()
app.title("Редактор")

canvas = tk.Canvas(app, width=600, height=400)
canvas.pack()
image_container = canvas.create_image(20, 20, anchor="nw")

# Buttons for Image Operations
button_frame = tk.Frame(app)
button_frame.pack()

load_button = tk.Button(button_frame, text="Завантажити", command=load_image)
load_button.pack(side=tk.LEFT)

save_button = tk.Button(button_frame, text="Зберегти", command=save_image)
save_button.pack(side=tk.LEFT)
```

```
brightness_inc_button = tk.Button(button_frame, text="Збільшити яскравість",
command=lambda: change_brightness(1.5))
brightness_inc_button.pack(side=tk.LEFT)

brightness_dec_button = tk.Button(button_frame, text="Зменшити яскравість",
command=lambda: change_brightness(0.5))
brightness_dec_button.pack(side=tk.LEFT)

noise_button = tk.Button(button_frame, text="Додати шум", command=lambda:
add_noise(25))
noise_button.pack(side=tk.LEFT)

blur_button = tk.Button(button_frame, text="Гаусівський блюр", command=lambda:
gaussian_blur(2))
blur_button.pack(side=tk.LEFT)

negative_button = tk.Button(button_frame, text="Негатив", command=make_negative)
negative_button.pack(side=tk.LEFT)

app.mainloop()
```

3.6. Аналіз результатів відлагодження та верифікації результатів роботи програми.

Результати відладки та тестування підтвердили працездатність розробленого коду. Порівняння отриманих результатів з технічними умовами завдання на лабораторну роботу та верифікація функціоналу програмного коду показали, що всі вимоги були виконані у повному обсязі.

IV. Висновки.

Під час виконання лабораторної роботи було проведено дослідження методів обробки растрових зображень. Було реалізовано функціонал, який включає в себе зменшення і збільшення яскравості, перетворення растру на негатив, додавання шуму та розмиття. Також було проведено аналіз та синтез математичної моделі обробки растру.

Виконав: студент Панченко С.В.