**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**

**І. 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.

**ІІ. Завдання**

Лабораторія провідної ІТ-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 ІІІ 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.

**ІІІ. 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** (де x, y – це положення пікселя на раcтрі) досягається шляхом додавання до початкових значень RGB пікселів випадкових величин, згенерованих з використанням Гаусівського шуму. , Тут – відповідає за середнє квадратичне відхилення, а 0 – за середнє значення:

**Розмиття растру** здійснюється шляхом застосування формули Гаусівського розмиття до кожного значення RGB в растрі:

**Негатив растру**  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:

**Зміною яскравості**  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; +).

**3.2. Блок схема алгоритму та її опис.**

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

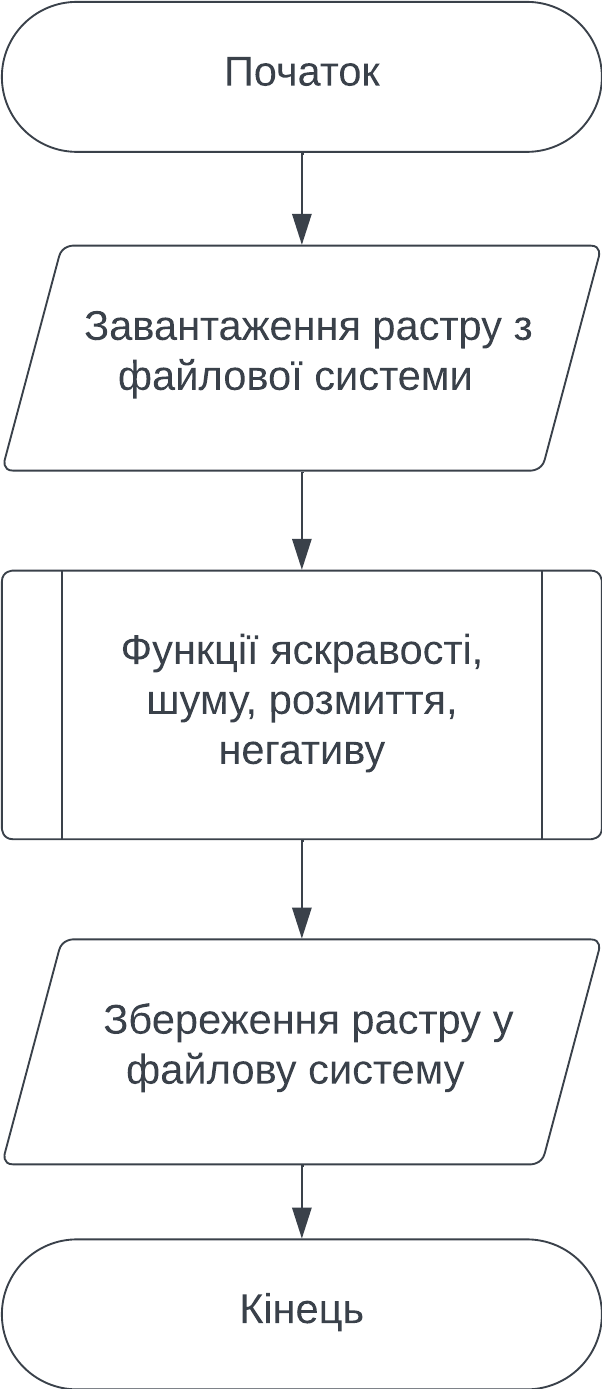


Рис. 1. Блок-схема алгоритму

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:

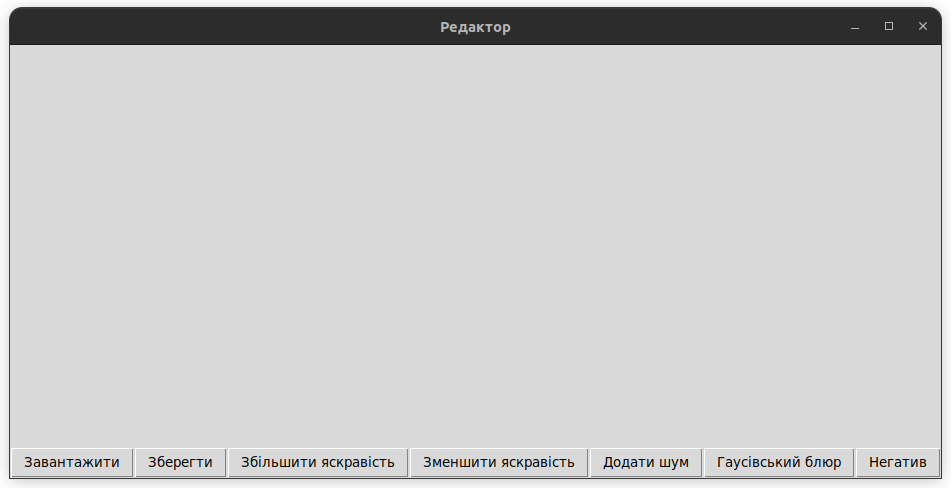


Рис. 3. Початковий екран

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

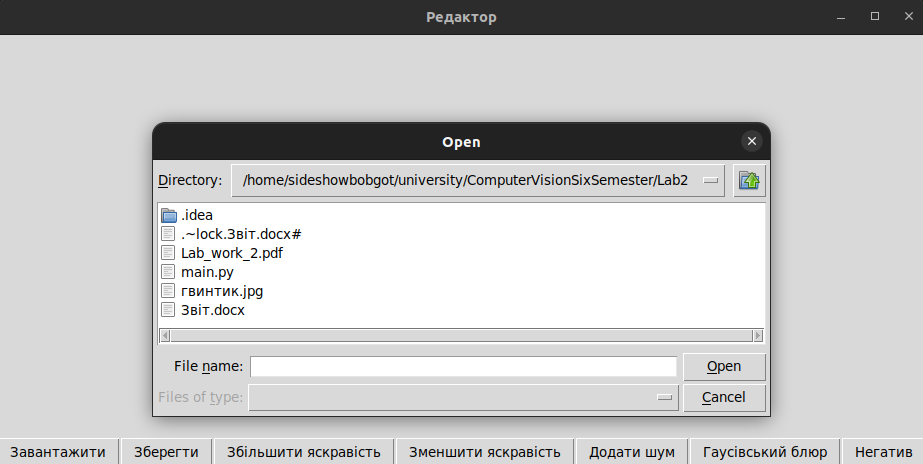


Рис. 4. Обирання растру з файлової системи.

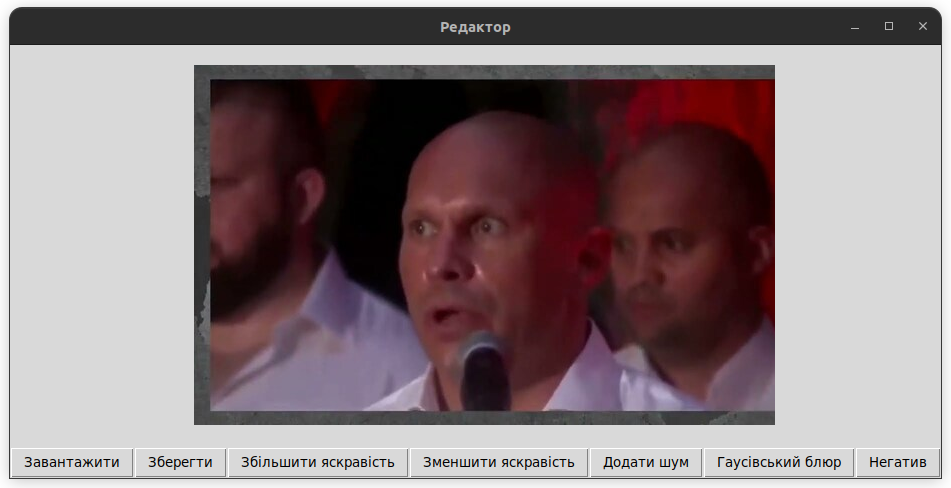


Рис. 5. Редактор після обрання картинки

1. При обранні кнопок:

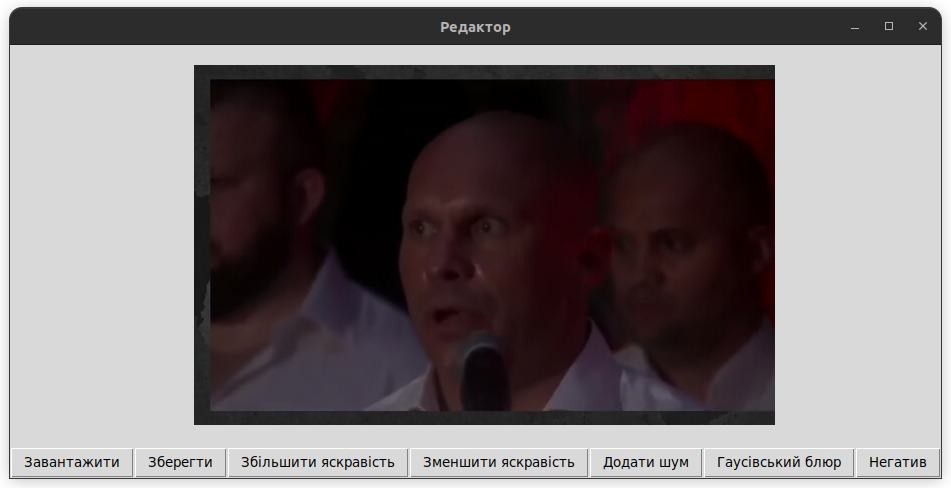


Рис. 6. Зменшення яскравості

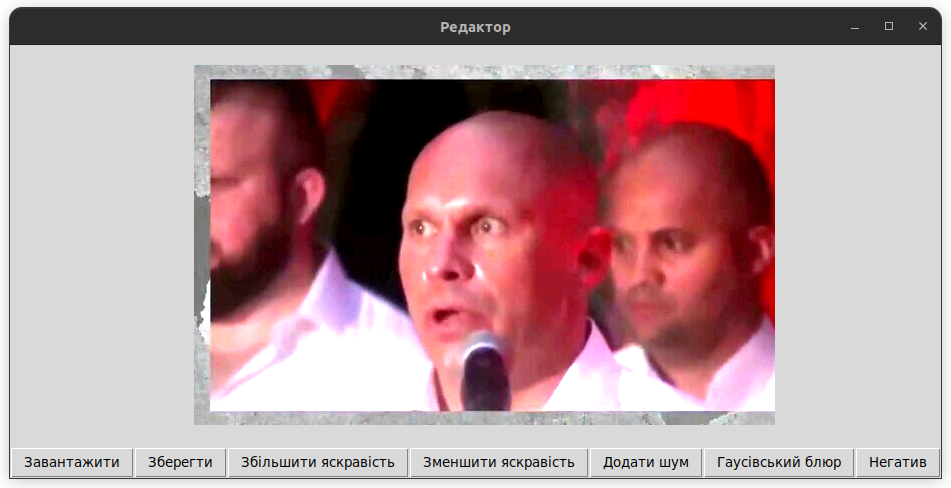


Рис. 7. Збільшення яскравості

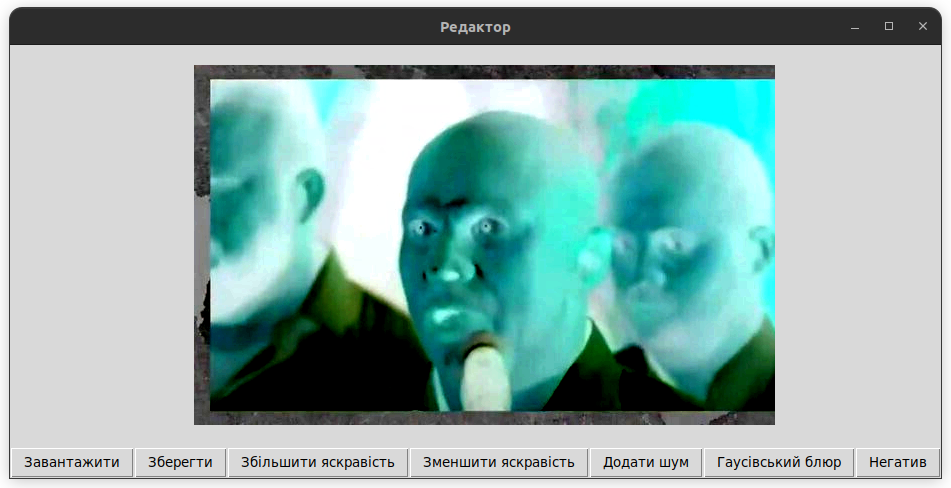


Рис. 8. Негативна фотографія

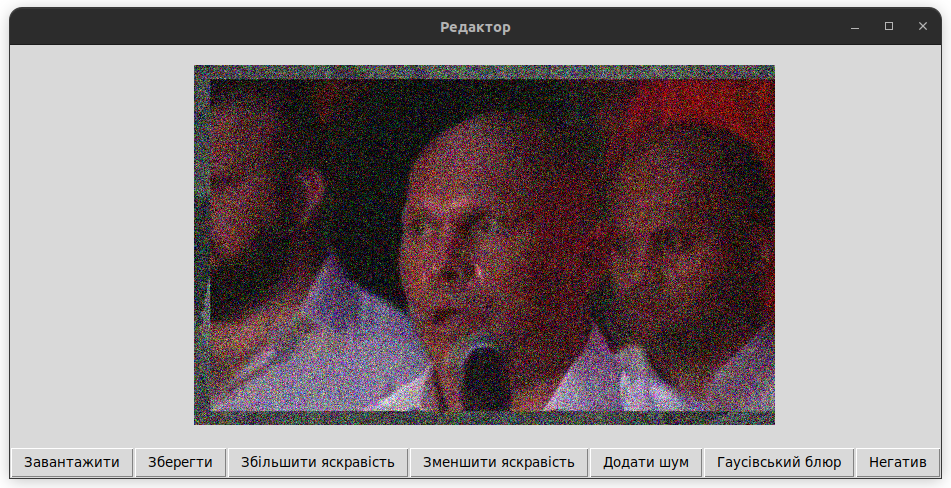


Рис. 9. Додання шуму до фотографії

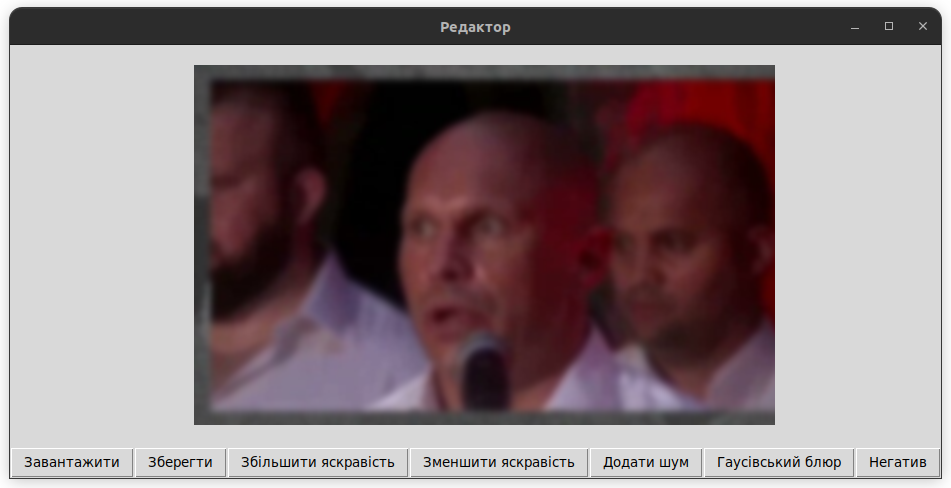


Рис. 10. Додання блюру до растру

**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.