**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 ТЕХНОЛОГІЙ ПОБУДОВИ ТА ПЕРЕТВОРЕННЯ КООРДИНАТ**

**ПЛОЩИННИХ (2D) ОБ’ЄКТІВ**

**Виконав:**

Студент 3 курсу кафедри ІПІ ФІОТ,

Навчальної групи ІП-11

Головня О.Р.

**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**

**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 завданні, над вхідним растровим графічним об'єктом. Ця модель здійснює дії, такі як додавання шуму, розмиття, збільшення яскравості та створення негативу.

Для зміни яскравості кожного пікселя (x, y) на зображенні застосовується множення значення RGB кожного пікселя на певний коефіцієнт S.

Для додавання шуму до зображення застосовується метод, що полягає в додаванні до кожного пікселя (x, y) значень шуму, які генеруються з використанням Гаусівського розподілу. Це реалізується шляхом додавання випадкової величини, згенерованої з розподілу N(0, σ^2), до початкових значень RGB пікселя (x, y). Тут σ^2 відповідає за дисперсію, що визначає ступінь відхилення значень шуму, а 0 вказує на середнє значення шуму (яке в цьому випадку рівне 0).

Для розмиття кожного пікселя (x, y) на зображенні застосовується гаусівське розмиття, яке обчислює нове значення RGB пікселя з урахуванням значень сусідніх пікселів. Гаусівське розмиття розглядає кожен піксель як центральний у вагованій матриці, де вага кожного сусіднього пікселя залежить від його відстані від центрального пікселя. Цей процес призводить до згладжування граней та зменшення різкості контурів на зображенні.

Для створення негативу кожного пікселя (x, y) на зображенні застосовується обернення значення RGB, що означає вибір значення з протилежного кінця діапазону значень RGB.

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

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

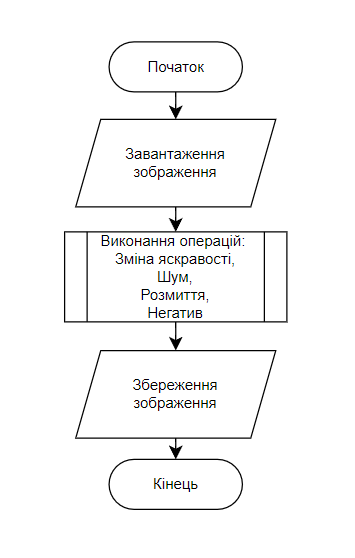


Рис. 1. Блок-схема програми

**3.3. Опис структури проекту програми**

|  |
| --- |
|  |
| Рис.2. Структура проекту. |

file.png – зображення для зміни

Holovnia\_lr\_2.docx – звіт

Holovnia\_lr\_2.py - файл програми

saved.jpg – Збережене зображення

**3.4. Результати роботи програми відповідно до завдання.**



Рис. 3. Зображення перед змінами



Рис. 4. Зображення після змін

**3.5. Програмний код.**

from PIL import Image, ImageFilter

import numpy as np

def load\_image(file\_path):

return Image.open(file\_path).convert('RGB')

def save\_image(img, file\_path):

img.save(file\_path)

def change\_brightness(img, factor):

img\_array = np.array(img, dtype=np.float32)

img\_array \*= factor

img\_array = np.clip(img\_array, 0, 255).astype(np.uint8)

return Image.fromarray(img\_array)

def add\_noise(img, 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)

return Image.fromarray(noisy\_img\_array)

def gaussian\_blur(img, radius):

return img.filter(ImageFilter.GaussianBlur(radius))

def make\_negative(img):

img\_array = np.array(img)

negative\_img\_array = 255 - img\_array

return Image.fromarray(negative\_img\_array)

def main():

# Load image

file\_path = "file.png"

img = load\_image(file\_path)

# Apply operations

img = change\_brightness(img, 1.5)

img = add\_noise(img, 25)

img = gaussian\_blur(img, 2)

img = make\_negative(img)

# Save image

save\_path = "saved.jpg"

save\_image(img, save\_path)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**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 програму, яка змінює яскравость, перетворює на негатив, додає шуму та розмиття.