****

Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №2

**З дисципліни «Технології Computer Vision»**

*ДОСЛІДЖЕННЯ АЛГОРИТМІВ ФОРМУВАННЯ ТА ОБРОБКИ РАСТРОВИХ ЦИФРОВИХ ЗОБРАЖЕНЬ*

| Виконала  студент кафедри ІСТ ФІОТ, групи ІА-12: |  | Перевірив:  пос. Баран Д. Р. |
| --- | --- | --- |
| Яковенко Д. О. |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Київ 2024

**І. Мета:**

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. Формування та перетворення координат площинних (2d) та просторових (3d) об’єктів.

**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 система з технологіями Computer Vision Замовниками 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.*

**Здійснити синтез математичних моделей та розробити програмний скрипт, що забезпечує реалізацію растрових алгоритмів над 3D графічними примітивами. Технічні умови реалізації завдання наведені у таблиці додатку.**

**Завдання І рівня – максимально 7 балів.**

Здійснити виконання завдання лабораторної роботи зі зміною кольору контурної заливки від растра до растра.

**Завдання ІІ рівня – максимально 8 балів.**

Здійснити виконання завдання лабораторної роботи із заливкою обраних граней 3D фігури зі зміною кольору заливки від растра до растру.

**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.**

Відповідно до умов задачі синтезовано математичну модель операцій над структурою вхідного графічного об’єкту. Кожна з наступних моделей використовує певний метод для обчислення нового значення кожного пікселя у вихідному зображенні на основі значень пікселів у вихідному зображенні та додаткових параметрів, таких як фактор зміни яскравості чи напрямок градієнту.

**1. Зміна яскравості (brightness\_correction):**

- Модель додає певний фактор до значення яскравості кожного пікселя зображення. Це досягається шляхом додавання числового значення (фактора) до інтенсивності кольору кожного пікселя.

**2. Відтінки сірого (grayscale\_correction):**

- Модель перетворює кольорове зображення в зображення у відтінках сірого, де кожен піксель зображення представляється одним числовим значенням, що відповідає його інтенсивності.

**3. Негатив (negative\_correction):**

- Модель обчислює негатив кожного пікселя, використовуючи логічне заперечення (bitwise\_not) для значення кольору кожного пікселя. Це дозволяє отримати зображення зі зміненими кольорами, які є оберненими до оригінального.

**4. Градієнти:**

*- Діагональ (Зверзу та знизу у напрямах зліва та з правого боку):* Градієнт обчислюється вздовж діагоналей зображення. Використовуються різні формули для різних напрямків діагоналей.

*- Від центру:* Градієнт обчислюється від центру зображення до його країв. Використовуються відстані від центру зображення до кожного пікселя для обчислення градієнта.

*- До центру:* Градієнт обчислюється від країв зображення до його центру. Використовуються відстані від кожного пікселя до центру зображення для обчислення градієнта.

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

Застосування синтезованих моделей здійснюється у порядку, що відображає суть алгоритму реалізації завдань лабораторної роботи.

|  |
| --- |
| Рис.1. Блок-схема алгоритму програми для завдання третьої складності. |

Розглянемо даний алгоритм. Робота розпочинається із формування функцій, що будуть обробляти оригінальне зобрадження. У програмі буде визначено 4 таких функції:

**brightness** – ця функція приймає вхідне зображення image у форматі numpy.ndarray та фактор корекції яскравості factor. Вона додає цей фактор до інтенсивності кожного пікселя у зображенні, переводячи піксельні значення до типу float64 для точності обчислень. Після додавання фактору інтенсивності кожного пікселя обмежуються в діапазоні [0, 255] за допомогою функції np.clip, щоб уникнути переповнення або від'ємних значень. Нарешті, коректоване зображення повертається у форматі uint8;

**grayscale** –використовує функцію cv2.cvtColor для конвертації зображення у відтінки сірого, за допомогою параметра cv2.COLOR\_BGR2GRAY;

**negative** – використовує функцію cv2.bitwise\_not, щоб обернути всі біти кольорів у зображенні, створюючи ефект негативу. Після цього негативне зображення повертається як вихідний результат;

**gradient** – функція приймає вхідне зображення image у форматі numpy.ndarray та параметр direction, що вказує напрямок градієнту. Для кожного напрямку вона виконує відповідні обчислення градієнту та застосовує їх до пікселів зображення. Нарешті, вона обмежує значення пікселів у діапазоні [0, 255] та повертає результат у вигляді зображення.

Описані дії реалізуються у блоці 1 блок-схеми алгоритму рис.1.

Відповідно для реалізації цих функцій необхідно завантажити зображення, яке буде застосовуватися при обробці. Завантажуємо наступним чином (блок 2):

image\_path = 'img.jpeg'

image = cv2.imread(image\_path, cv2.IMREAD\_COLOR)

В блоці 3 усі функції описані в блоці 1 застосовуються до завантаженого зображення.

Результат вищеописаних обрахунків відображається у графічному вікні. У ньому виводимо усі оброблені зображення.

На цьому робота алгоритму завершується.

**3.3. Опис структури проекту програми в середовищі PyCharm.**

Для реалізації розробленого алгоритму мовою програмування Python з використанням можливостей інтегрованого середовища PyCharm сформовано проект.

Проект базується на лінійній бізнес-логіці функціонального програмування та має таку структуру.

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

CV\_lab2 – головний каталог проекту

lab2.py – файл програмного коду лабораторної роботи;

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

Результатом роботи програми є програма, в якій при її запуску отримуємо такий результат:

|  |
| --- |
| Рис.3. Результат обробки зображення. |

Таким чином, представлені результати у повному обсязі відповідають завданню лабораторної роботи.

**3.5. Програмний код.**

Програмний код послідовно реалізує алгоритми рис.1 для виконання завдання третього рівня складності.

Для спрощення програмного коду і раціоналізації обчислень застосовано функціональні механізми створення підпрограм.

Обчислення проводились з розширеною матрицею координат паралелепіпеду.

При цьому використано можливості Python бібліотек: openCV; numpy; matplotlib.

import numpy as np

import cv2

import matplotlib.pyplot as plt

def brightness(image, factor):

corrected\_image = image.astype(np.float64) + factor

corrected\_image = np.clip(corrected\_image, 0, 255)

return corrected\_image.astype(np.uint8)

def grayscale(image):

grayscale\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

return grayscale\_image

def negative(image):

negative\_image = cv2.bitwise\_not(image)

return negative\_image

def gradient(image, direction='diagonal'):

rows, cols, \_ = image.shape

gradient\_image = np.zeros\_like(image, dtype=np.float64)

if direction == 'diagonal\_top\_left':

for i in range(rows):

for j in range(cols):

gradient = (i + j) / (rows + cols) \* 255

gradient\_image[i, j] = image[i, j] \* (gradient / 255)

elif direction == 'diagonal\_top\_right':

for i in range(rows):

for j in range(cols):

distance = np.sqrt((i) \*\* 2 + (cols - j) \*\* 2)

max\_distance = np.sqrt(rows \*\* 2 + cols \*\* 2)

gradient = distance / max\_distance \* 255

gradient\_image[i, j] = image[i, j] \* (gradient / 255)

elif direction == 'diagonal\_bottom\_left':

for i in range(rows):

for j in range(cols):

distance = np.sqrt((rows - i) \*\* 2 + (j) \*\* 2)

max\_distance = np.sqrt(rows \*\* 2 + cols \*\* 2)

gradient = distance / max\_distance \* 255

gradient\_image[i, j] = image[i, j] \* (gradient / 255)

elif direction == 'diagonal\_bottom\_right':

for i in range(rows):

for j in range(cols):

distance = np.sqrt((rows - i) \*\* 2 + (cols - j) \*\* 2)

max\_distance = np.sqrt(rows \*\* 2 + cols \*\* 2)

gradient = distance / max\_distance \* 255

gradient\_image[i, j] = image[i, j] \* (gradient / 255)

elif direction == 'from\_center':

center\_x, center\_y = rows / 2, cols / 2

for i in range(rows):

for j in range(cols):

distance = np.sqrt((center\_x - i) \*\* 2 + (center\_y - j) \*\* 2)

max\_distance = np.sqrt(center\_x \*\* 2 + center\_y \*\* 2)

gradient = distance / max\_distance \* 255

gradient\_image[i, j] = image[i, j] \* (gradient / 255)

elif direction == 'to\_center':

center\_x, center\_y = rows / 2, cols / 2

for i in range(rows):

for j in range(cols):

distance = np.sqrt((center\_x - i) \*\* 2 + (center\_y - j) \*\* 2)

max\_distance = np.sqrt(center\_x \*\* 2 + center\_y \*\* 2)

gradient = (1 - distance / max\_distance) \* 255

gradient\_image[i, j] = image[i, j] \* (gradient / 255)

return np.clip(gradient\_image, 0, 255).astype(np.uint8)

image\_path = 'img.jpeg'

image = cv2.imread(image\_path, cv2.IMREAD\_COLOR)

brightened\_image = brightness(image, 50)

grayscale\_image = grayscale(image)

negative\_image = negative(image)

diagonal\_gradient\_image\_tl = gradient(image, direction='diagonal\_top\_left')

diagonal\_gradient\_image\_tr = gradient(image, direction='diagonal\_top\_right')

diagonal\_gradient\_image\_bl = gradient(image, direction='diagonal\_bottom\_left')

diagonal\_gradient\_image\_br = gradient(image, direction='diagonal\_bottom\_right')

from\_center\_gradient\_image = gradient(image, direction='from\_center')

to\_center\_gradient\_image = gradient(image, direction='to\_center')

plt.figure(figsize=(15, 15))

plt.subplot(4, 3, (1, 3))

plt.imshow(cv2.cvtColor(image, cv2.COLOR\_BGR2RGB))

plt.title('Оригінальне зображення')

plt.subplot(4, 3, 4)

plt.imshow(cv2.cvtColor(brightened\_image, cv2.COLOR\_BGR2RGB))

plt.title('Збільшена яскравість')

plt.subplot(4, 3, 5)

plt.imshow(grayscale\_image, cmap='gray')

plt.title('Відтінки сірого')

plt.subplot(4, 3, 6)

plt.imshow(cv2.cvtColor(negative\_image, cv2.COLOR\_BGR2RGB))

plt.title('Негатив')

plt.subplot(4, 3, 7)

plt.imshow(cv2.cvtColor(diagonal\_gradient\_image\_tl, cv2.COLOR\_BGR2RGB))

plt.title('Градієнт: діагональ (зверху-ліворуч)')

plt.subplot(4, 3, 8)

plt.imshow(cv2.cvtColor(diagonal\_gradient\_image\_tr, cv2.COLOR\_BGR2RGB))

plt.title('Градієнт: діагональ (зверху-праворуч)')

plt.subplot(4, 3, 9)

plt.imshow(cv2.cvtColor(diagonal\_gradient\_image\_bl, cv2.COLOR\_BGR2RGB))

plt.title('Градієнт: діагональ (знизу-ліворуч)')

plt.subplot(4, 3, 10)

plt.imshow(cv2.cvtColor(diagonal\_gradient\_image\_br, cv2.COLOR\_BGR2RGB))

plt.title('Градієнт: діагональ (знизу-праворуч)')

plt.subplot(4, 3, 11)

plt.imshow(cv2.cvtColor(from\_center\_gradient\_image, cv2.COLOR\_BGR2RGB))

plt.title('Градієнт: від центру')

plt.subplot(4, 3, 12)

plt.imshow(cv2.cvtColor(to\_center\_gradient\_image, cv2.COLOR\_BGR2RGB))

plt.title('Градієнт: до центру')

plt.tight\_layout()

plt.show()

**3.6. Аналіз результатів відлагодження та верифікації результатів роботи програми.**

Результати відлагодження та тестування довели працездатність розробленого коду.

Верифікація функціоналу програмного коду, порівняння отриманих результатів з технічними умовами завдання на лабораторну роботу доводять, що завдання виконано у повному обсязі.

**IV. Висновки.**

У ході виконання лабораторної роботи проведено дослідження алгоритмів формування та обробки растрових цифрових зображень з використанням можливостей алгоритмічної мови високого рівня Python. В результаті виконання я здійснила обробку завантаженого зображення та дослідила алгоритми обробки.

Виконала: студент Яковенко Д.О.