МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет електроніки і комп’ютерних технологій

Кафедра системного проєктування

**Звіт**

про виконання лабораторної роботи № 4

з дисципліни «Цифрова обробка зображень»

«Колірна обробка»

**Виконав:**

студент групи Феп-31

Линва Віталій

**Перевірив:**

Проф. Половинко І. І.

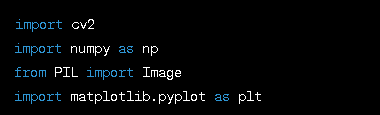
**Львів – 2023**

**Хід роботи**

Частина 1:

У даній лабораторній роботі я використав бібліотеки NumPy, OpenCV (cv2), PIL (Image) і matplotlib для обробки та візуалізації зображення.

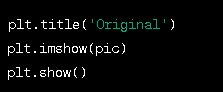
Спочатку, я імпортував необхідні бібліотеки:



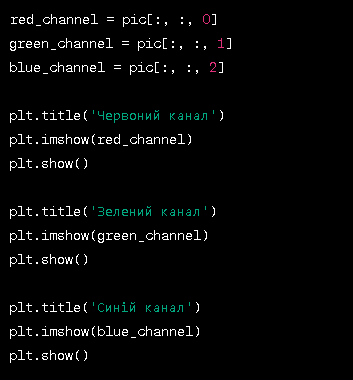
Після цього, я завантажив зображення за допомогою функції `Image.open` з бібліотеки PIL. Зображення було перетворено в масив NumPy за допомогою `np.array`.



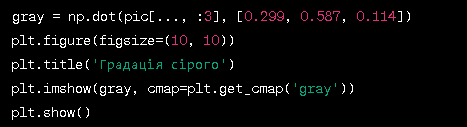
Далі, я відобразив оригінальне зображення за допомогою `plt.imshow`. Функція `plt.show()` була використана для відображення зображення.



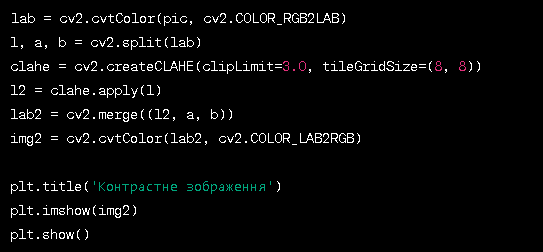
Після цього, я виділивокремі кольорові канали зображення (`червоний`, `зелений` і `синій`) за допомогою індексування масиву `pic`. Кожен канал був відображений окремо за допомогою `plt.imshow`.



Далі, я перетворив зображення в градацію сірого шляхом обчислення зважених сум кольорових каналів. Результат був відображений за допомогою `plt.imshow` з параметром `cmap='gray'` для відображення у відтінках сірого.



На останок, я застосував покращення контрастності до зображення за допомогою методу CLAHE (Contrast Limited Adaptive Histogram Equalization) з використанням бібліотеки OpenCV. Покращене зображення було відображене за допомогою `plt.imshow`.



Вивід:

|  |
| --- |
|  |
|  |
|  |

**Висновок:** Метою лабораторної роботи було ознайомлення з обробкою та аналізом зображень. Я успішно завантажив зображення, відобразив його та його окремі канали, перетворив його в градацію сірого та застосував покращення контрастності.

Частина 2:

\*\*Лабораторна робота з використанням бібліотеки Tkinter для створення графічного інтерфейсу\*\*

У даній лабораторній роботі було створено графічний інтерфейс за допомогою бібліотеки Tkinter у мові програмування Python. Інтерфейс дозволяє перетворювати кольори між різними моделями: RGB, CMYK, HSV та отримувати шістнадцятковий код кольору (HEX).

Інтерфейс містить наступні елементи:

1. Три поля для введення значень червоного (R), зеленого (G) та синього (B) каналів кольору в моделі RGB.

2. Чотири поля для відображення та введення значень гіана (C), мажента (M), жовтого (Y) та чорного (K) каналів кольору в моделі CMYK.

3. Три поля для відображення та введення значень тону (H), насиченості (S) та значення (V) каналів кольору в моделі HSV.

4. Одне поле для відображення шістнадцяткового коду кольору (HEX).

5. Кнопки "from RGB", "from CMYK", "from HSV" для конвертації значень кольору з однієї моделі в іншу.

6. Кнопка "Clear" для очищення всіх полів та повернення фону в білий колір.

При натисканні на кнопки "from RGB", "from CMYK" або "from HSV" відбувається обробка введених значень кольору та виведення результатів у відповідні поля. Також змінюється фоновий колір вікна на вибраний колір.

Кнопка "Clear" дозволяє очистити всі поля та повернути фон в білий колір.

Цей графічний інтерфейс дозволяє зручно перетворювати значення кольору між різними моделями та візуалізувати результат на екрані.



**Висновок: У даній лабораторній роботі було створено графічний інтерфейс за допомогою бібліотеки Tkinter у мові програмування Python. Інтерфейс дозволяє перетворювати кольори між різними моделями: RGB, CMYK, HSV та отримувати шістнадцятковий код кольору (HEX)**

import cv2

import numpy as np

from PIL import Image

import matplotlib.pyplot as plt

# Read the image

pic = np.array(Image.open('Lab3\pict.png'))

# Display the original image

plt.title('Original')

plt.imshow(pic)

plt.show()

# Extract color channels

red\_channel = pic[:, :, 0]

green\_channel = pic[:, :, 1]

blue\_channel = pic[:, :, 2]

# Display color channels separately

plt.title('Червоний канал')

plt.imshow(red\_channel)

plt.show()

plt.title('Зелений канал')

plt.imshow(green\_channel)

plt.show()

plt.title('Синій канал')

plt.imshow(blue\_channel)

plt.show()

# Display single-channel grayscale image

gray = np.dot(pic[..., :3], [0.299, 0.587, 0.114])

plt.figure(figsize=(10, 10))

plt.title('Градація сірого')

plt.imshow(gray, cmap=plt.get\_cmap('gray'))

plt.show()

# Apply contrast enhancement

lab = cv2.cvtColor(pic, cv2.COLOR\_RGB2LAB)

l, a, b = cv2.split(lab)

clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8, 8))

l2 = clahe.apply(l)

lab2 = cv2.merge((l2, a, b))

img2 = cv2.cvtColor(lab2, cv2.COLOR\_LAB2RGB)

# Display contrast-enhanced image

plt.title('Контрастне зображення')

plt.imshow(img2)

plt.show()

from tkinter import \*

cmyk\_scale = 100

def rgb\_to\_cmyk(r, g, b):

    if (r == 0) and (g == 0) and (b == 0):

        # black

        return 0, 0, 0, cmyk\_scale

    c = 1 - r / 255.

    m = 1 - g / 255.

    y = 1 - b / 255.

    min\_cmy = min(c, m, y)

    c = (c - min\_cmy) / (1 - min\_cmy)

    m = (m - min\_cmy) / (1 - min\_cmy)

    y = (y - min\_cmy) / (1 - min\_cmy)

    k = min\_cmy

    return (int)(c \* cmyk\_scale), (int)(m \* cmyk\_scale), (int)(y \* cmyk\_scale), (int)(k \* cmyk\_scale)

def cmyk\_to\_rgb(c, m, y, k, cmyk\_scale, rgb\_scale=255):

    r = rgb\_scale \* (1.0 - c / float(cmyk\_scale)) \* (1.0 - k / float(cmyk\_scale))

    g = rgb\_scale \* (1.0 - m / float(cmyk\_scale)) \* (1.0 - k / float(cmyk\_scale))

    b = rgb\_scale \* (1.0 - y / float(cmyk\_scale)) \* (1.0 - k / float(cmyk\_scale))

    return (int)(r), (int)(g), (int)(b)

def retmax(a, b, c):

    max = 0

    max = a

    if (max < b):

        max = b

    if (max < c):

        max = c

    return max

def retmin(a, b, c):

    min = 0

    min = a

    if (min > b):

        min = b

    if (min > c):

        min = c

    return min

def rgb\_to\_hsv(R, G, B):

    max = 0

    min = 0

    R = R / 100

    G = G / 100

    B = B / 100

    max = retmax(R, G, B)

    min = retmin(R, G, B)

    v = max

    if (max == 0):

        s = 0

    else:

        s = 1 - (min / max)

    if (max == min):

        h = 0

    elif (max == R and G >= B):

        h = 60 \* ((G - B) / (max - min))

    elif (max == R and G < B):

        h = 60 \* ((G - B) / (max - min)) + 360

    elif (max == G):

        h = 60 \* ((B - R) / (max - min)) + 120

    elif (max == B):

        h = 60 \* ((R - G) / (max - min)) + 240

    v = v \* 39.5

    s = s \* 100

    return (int)(h), (int)(s), (int)(v)

def hsv\_to\_rgb(h, s, v):

    H = (float)(h)

    S = (float)(s) / 100.0

    V = (float)(v) / 39.5

    if (S == 0):

        R = G = B = V

    else:

        H = H / 60

        i = (int)(H)

        C = H - i

        X = V \* (1 - S)

        Y = V \* (1 - S \* C)

        Z = V \* (1 - S \* (1 - C))

        if (i == 0):

            R = V

            G = Z

            B = X

        if (i == 1):

            R = Y

            G = V

            B = X

        if (i == 2):

            R = X

            G = V

            B = Z

        if (i == 3):

            R = X

            G = Y

            B = V

        if (i == 4):

            R = Z

            G = X

            B = V

        if (i == 5):

            R = V

            G = X

            B = Y

    R = R \* 100

    G = G \* 100

    B = B \* 100

    return (int)(R), (int)(G), (int)(B)

window = Tk()

window.title("Color Models")

window.geometry("1000x300")

# Labels

lb1 = Label(window, text="R =", font=("Arial", 14))

lb2 = Label(window, text="G =", font=("Arial", 14))

lb3 = Label(window, text="B =", font=("Arial", 14))

lb1.grid(column=0, row=0)

lb2.grid(column=0, row=1)

lb3.grid(column=0, row=2)

lb4 = Label(window, text="C =", font=("Arial", 14))

lb5 = Label(window, text="M =", font=("Arial", 14))

lb6 = Label(window, text="Y =", font=("Arial", 14))

lb7 = Label(window, text="K =", font=("Arial", 14))

lb4.grid(column=5, row=0)

lb5.grid(column=5, row=1)

lb6.grid(column=5, row=2)

lb7.grid(column=5, row=3)

lb8 = Label(window, text="H =", font=("Arial", 14))

lb9 = Label(window, text="S =", font=("Arial", 14))

lb10 = Label(window, text="V =", font=("Arial", 14))

lb8.grid(column=10, row=0)

lb9.grid(column=10, row=1)

lb10.grid(column=10, row=2)

lb11 = Label(window, text="HEX =", font=("Arial", 14))

lb11.grid(column=0, row=5)

# Textboxes

tb1 = Entry(window, width=10)

tb1.grid(column=1, row=0)

tb2 = Entry(window, width=10)

tb2.grid(column=1, row=1)

tb3 = Entry(window, width=10)

tb3.grid(column=1, row=2)

tb4 = Entry(window, width=10)

tb4.grid(column=6, row=0)

tb5 = Entry(window, width=10)

tb5.grid(column=6, row=1)

tb6 = Entry(window, width=10)

tb6.grid(column=6, row=2)

tb7 = Entry(window, width=10)

tb7.grid(column=6, row=3)

tb8 = Entry(window, width=10)

tb8.grid(column=16, row=0)

tb9 = Entry(window, width=10)

tb9.grid(column=16, row=1)

tb10 = Entry(window, width=10)

tb10.grid(column=16, row=2)

tb11 = Entry(window, width=10)

tb11.grid(column=1, row=5)

def click\_button1():

    R = int(tb1.get())

    G = int(tb2.get())

    B = int(tb3.get())

    C, M, Y, K = rgb\_to\_cmyk(R, G, B)

    H, S, V = rgb\_to\_hsv(R, G, B)

    color = '#%02x%02x%02x' % (R, G, B)

    tb4.delete(0, END)

    tb4.insert(0, C)

    tb5.delete(0, END)

    tb5.insert(0, M)

    tb6.delete(0, END)

    tb6.insert(0, Y)

    tb7.delete(0, END)

    tb7.insert(0, K)

    tb8.delete(0, END)

    tb8.insert(0, H)

    tb9.delete(0, END)

    tb9.insert(0, S)

    tb10.delete(0, END)

    tb10.insert(0, V)

    tb11.delete(0, END)

    tb11.insert(0, color)

    window['background'] = color

def click\_button2():

    C = (float)(tb4.get())

    M = (float)(tb5.get())

    Y = (float)(tb6.get())

    K = (float)(tb7.get())

    R, G, B = cmyk\_to\_rgb(C, M, Y, K, cmyk\_scale)

    color = '#%02x%02x%02x' % (R, G, B)

    tb1.delete(0, END)

    tb1.insert(0, R)

    tb2.delete(0, END)

    tb2.insert(0, G)

    tb3.delete(0, END)

    tb3.insert(0, B)

    tb11.delete(0, END)

    tb11.insert(0, color)

    window['background'] = color

    click\_button1()

def click\_button3():

    H = int(tb8.get())

    S = int(tb9.get())

    V = int(tb10.get())

    R, G, B = hsv\_to\_rgb(H, S, V)

    color = '#%02x%02x%02x' % (R, G, B)

    tb1.delete(0, END)

    tb1.insert(0, R)

    tb2.delete(0, END)

    tb2.insert(0, G)

    tb3.delete(0, END)

    tb3.insert(0, B)

    tb11.delete(0, END)

    tb11.insert(0, color)

    window['background'] = color

    C, M, Y, K = rgb\_to\_cmyk(R, G, B)

    tb4.delete(0, END)

    tb4.insert(0, C)

    tb5.delete(0, END)

    tb5.insert(0, M)

    tb6.delete(0, END)

    tb6.insert(0, Y)

    tb7.delete(0, END)

    tb7.insert(0, K)

def click\_button4():

    tb1.delete(0, END)

    tb2.delete(0, END)

    tb3.delete(0, END)

    tb4.delete(0, END)

    tb5.delete(0, END)

    tb6.delete(0, END)

    tb7.delete(0, END)

    tb8.delete(0, END)

    tb9.delete(0, END)

    tb10.delete(0, END)

    tb11.delete(0, END)

    window['background'] = '#%02x%02x%02x' % (255, 255, 255)

btn1 = Button(text="from RGB", background="#555", foreground="#ccc",

              padx="15", pady="6", font="15", command=click\_button1)

btn2 = Button(text="from CMYK", background="#555", foreground="#ccc",

              padx="15", pady="6", font="15", command=click\_button2)

btn3 = Button(text="from HSV", background="#555", foreground="#ccc",

              padx="15", pady="6", font="15", command=click\_button3)

btn4 = Button(text="Clear", background="#555", foreground="#ccc",

              padx="15", pady="6", font="15", command=click\_button4)

btn1.place(x=375, y=25)

btn2.place(x=625, y=25)

btn3.place(x=875, y=25)

btn4.place(x=615, y=100)

window.mainloop()