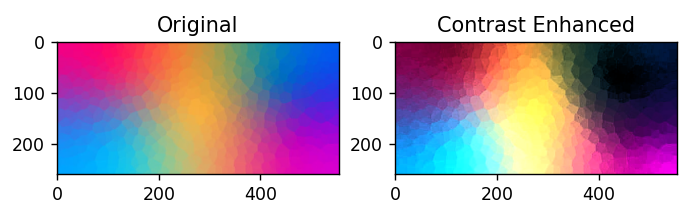
**Comment**

這次作業是第一次實作 RGB 跟 HSI 之間的轉換，過程中比較有挑戰的是理解 HSI 的公式，尤其是 H 的計算邏輯。用直方圖均衡化處理 I（亮度）通道後，能明顯看出影像對比變得更清楚，而且不會影響原本的色彩，覺得這種分離亮度處理的方式蠻實用的。整體流程從讀圖、轉換、增強到顯示都滿有成就感，也讓我更熟悉色彩空間的概念。

**Photo**



**Program**

import cv2

import numpy as np

import matplotlib.pyplot as plt

from skimage import exposure

# Step 1: 讀取 RGB 圖片

img\_rgb = cv2.imread('colorful.jpg')  # BGR

img\_rgb = cv2.cvtColor(img\_rgb, cv2.COLOR\_BGR2RGB)

# Step 2: 歸一化

rgb = img\_rgb.astype(np.float32) / 255.0

# Step 3: RGB to HSI

def rgb\_to\_hsi(rgb):

    r, g, b = rgb[..., 0], rgb[..., 1], rgb[..., 2]

    num = 0.5 \* ((r - g) + (r - b))

    den = np.sqrt((r - g)\*\*2 + (r - b)\*(g - b)) + 1e-6

    theta = np.arccos(num / den)

    H = np.where(b <= g, theta, 2\*np.pi - theta)

    H = H / (2\*np.pi)

    min\_rgb = np.minimum(np.minimum(r, g), b)

    S = 1 - 3 \* min\_rgb / (r + g + b + 1e-6)

    I = (r + g + b) / 3

    HSI = np.stack([H, S, I], axis=-1)

    return HSI

hsi = rgb\_to\_hsi(rgb)

# Step 4: Histogram Equalization on I channel

h, s, i = hsi[..., 0], hsi[..., 1], hsi[..., 2]

i\_eq = exposure.equalize\_hist(i)

# Step 5: HSI to RGB

def hsi\_to\_rgb(hsi):

    H, S, I = hsi[..., 0]\*2\*np.pi, hsi[..., 1], hsi[..., 2]

    R = np.zeros\_like(H)

    G = np.zeros\_like(H)

    B = np.zeros\_like(H)

    # Sector 0 to 2π/3

    mask1 = (H >= 0) & (H < 2\*np.pi/3)

    B[mask1] = I[mask1] \* (1 - S[mask1])

    R[mask1] = I[mask1] \* (1 + S[mask1] \* np.cos(H[mask1]) / np.cos(np.pi/3 - H[mask1]))

    G[mask1] = 3\*I[mask1] - (R[mask1] + B[mask1])

    # Sector 2π/3 to 4π/3

    mask2 = (H >= 2\*np.pi/3) & (H < 4\*np.pi/3)

    H2 = H[mask2] - 2\*np.pi/3

    R[mask2] = I[mask2] \* (1 - S[mask2])

    G[mask2] = I[mask2] \* (1 + S[mask2] \* np.cos(H2) / np.cos(np.pi/3 - H2))

    B[mask2] = 3\*I[mask2] - (R[mask2] + G[mask2])

    # Sector 4π/3 to 2π

    mask3 = (H >= 4\*np.pi/3)

    H3 = H[mask3] - 4\*np.pi/3

    G[mask3] = I[mask3] \* (1 - S[mask3])

    B[mask3] = I[mask3] \* (1 + S[mask3] \* np.cos(H3) / np.cos(np.pi/3 - H3))

    R[mask3] = 3\*I[mask3] - (G[mask3] + B[mask3])

    rgb\_out = np.stack([R, G, B], axis=-1)

    return np.clip(rgb\_out, 0, 1)

hsi\_eq = np.stack([h, s, i\_eq], axis=-1)

rgb\_eq = hsi\_to\_rgb(hsi\_eq)

# Step 6: 轉換成 [0,255] 並輸出

rgb\_out = (rgb\_eq \* 255).astype(np.uint8)

cv2.imwrite('output.jpg', cv2.cvtColor(rgb\_out, cv2.COLOR\_RGB2BGR))

# Step 7: 顯示前後影像

plt.subplot(1, 2, 1)

plt.title("Original")

plt.imshow(img\_rgb)

plt.subplot(1, 2, 2)

plt.title("Contrast Enhanced")

plt.imshow(rgb\_out)

plt.show()