DIP_HW Q8-Q10 資管四 蘇柏叡

Code link:

https://colab.research.google.com/drive/1M11jAaIqpvMGRrZG16LFGnCvID_VT2Zu?usp=sharing

---

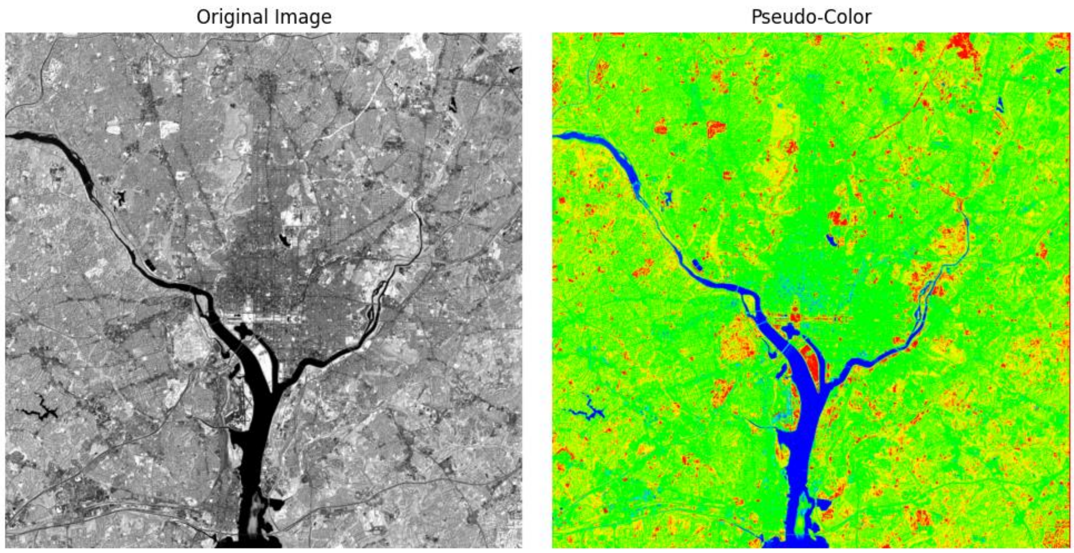
Q8

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load the image
img = cv2.imread('/content/Fig0110(4)(WashingtonDC Band4).TIF', cv2.IMREAD_GRAYSCALE)

# Display the original image
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(img, cmap='gray')
plt.title('Original Image')
plt.axis('off')

# Convert the image to double precision
img = img.astype(np.float64)
M, N = img.shape
gray_level = 256
R = np.zeros((M, N))
G = np.zeros((M, N))
B = np.zeros((M, N))

```

```python
# Apply the pseudo-coloring
for i in range(M):
    for j in range(N):
        if img[i, j] < gray_level / 4:
            R[i, j] = 0
            G[i, j] = 4 * img[i, j]
            B[i, j] = gray_level
        elif img[i, j] < gray_level / 2:
            R[i, j] = 0
            G[i, j] = gray_level
            B[i, j] = gray_level / 2 - 4 * img[i, j]
        elif img[i, j] < 3 * gray_level / 4:
            R[i, j] = 4 * img[i, j] - gray_level * 2
            G[i, j] = gray_level
            B[i, j] = 0
        else:
            R[i, j] = gray_level
            G[i, j] = 4 * gray_level - 4 * img[i, j]
            B[i, j] = 0

```

```python
# Combine the R, G, and B channels
img1 = np.zeros((M, N, 3))
img1[:, :, 0] = R
img1[:, :, 1] = G
img1[:, :, 2] = B

# Normalize the image
img1 = img1 / 256.0

# Display the pseudo-colored image
plt.subplot(1, 2, 2)
plt.imshow(img1)
plt.title('Pseudo-Color')
plt.axis('off')

plt.tight_layout()
plt.show()

```

Result:

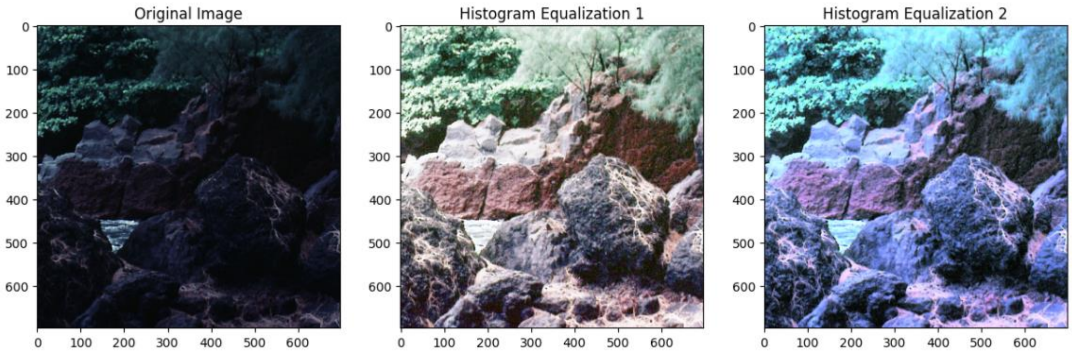


Q9

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

def rgb2hsi(img):
    # Convert from RGB to HSI
    img = img.astype(np.float32) / 255.0
    r, g, b = cv2.split(img)
    numerator = 0.5 * ((r - g) + (r - b))
    denominator = np.sqrt((r - g)**2 + (r - b) * (g - b))
    theta = np.arccos(numerator / (denominator + 1e-6))
    H = theta
    H[b > g] = 2 * np.pi - H[b > g]
    H = H / (2 * np.pi)

    S = 1 - (3 / (r + g + b + 1e-6)) * np.minimum(np.minimum(r, g), b)
    I = (r + g + b) / 3.0
    hsi = cv2.merge([H, S, I])
    return hsi

def hsi2rgb(hsi):
    # Convert from HSI to RGB
    hsi = hsi.astype(np.float32)
    H, S, I = cv2.split(hsi)
    H = H * 2 * np.pi

    R = np.zeros_like(H)
    G = np.zeros_like(H)
    B = np.zeros_like(H)

    # RG Sector (0 <= H < 2*pi/3)
    idx = (H >= 0) & (H < 2 * np.pi / 3)
    B[idx] = I[idx] * (1 - S[idx])
    R[idx] = I[idx] * (1 + (S[idx] * np.cos(H[idx])) / (np.cos(np.pi / 3 - H[idx]) + 1e-6))
    G[idx] = 3 * I[idx] - (R[idx] + B[idx])

    # GB Sector (2*pi/3 <= H < 4*pi/3)
    idx = (H >= 2 * np.pi / 3) & (H < 4 * np.pi / 3)
    H[idx] = H[idx] - 2 * np.pi / 3
    R[idx] = I[idx] * (1 - S[idx])
    G[idx] = I[idx] * (1 + (S[idx] * np.cos(H[idx])) / (np.cos(np.pi / 3 - H[idx]) + 1e-6))
    B[idx] = 3 * I[idx] - (R[idx] + G[idx])

    # BR Sector (4*pi/3 <= H < 2*pi)
    idx = (H >= 4 * np.pi / 3) & (H < 2 * np.pi)
    H[idx] = H[idx] - 4 * np.pi / 3
    G[idx] = I[idx] * (1 - S[idx])
    B[idx] = I[idx] * (1 + (S[idx] * np.cos(H[idx])) / (np.cos(np.pi / 3 - H[idx]) + 1e-6))
    R[idx] = 3 * I[idx] - (G[idx] + B[idx])

    rgb = cv2.merge([R, G, B])
    rgb = np.clip(rgb, 0, 1)
    return (rgb * 255).astype(np.uint8)
```

```python
# Load the image
img = cv2.imread('/content/Fig0635(bottom_left_stream).tif')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Plot original image
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
plt.imshow(img_rgb)
plt.title('Original Image')

# Histogram equalization on each RGB channel separately
R, G, B = cv2.split(img_rgb)
R_eq = cv2.equalizeHist(R)
G_eq = cv2.equalizeHist(G)
B_eq = cv2.equalizeHist(B)
img_eq1 = cv2.merge([R_eq, G_eq, B_eq])

plt.subplot(1, 3, 2)
plt.imshow(img_eq1)
plt.title('Histogram Equalization 1')

# Convert RGB to HSI, equalize the intensity channel, and convert back to RGB
img_hsi = rgb2hsi(img_rgb)
I = img_hsi[:, :, 2]
I_eq = cv2.equalizeHist((I * 255).astype(np.uint8)) / 255.0
img_hsi[:, :, 2] = I_eq
img_eq2 = hsi2rgb(img_hsi)

plt.subplot(1, 3, 3)
plt.imshow(img_eq2)
plt.title('Histogram Equalization 2')

plt.show()
```

Result:



Q10

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load the image
img = cv2.imread('/content/Fig0628(b)(jupiter-Io-closeup).tif')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Display the original image
plt.figure(figsize=(10, 8))
plt.subplot(2, 2, 1)
plt.imshow(img_rgb)
plt.title('Original Image')

# Convert the image to double precision (float)
img1 = img_rgb.astype(np.float64) / 255.0
R = img1[:, :, 0]
G = img1[:, :, 1]
B = img1[:, :, 2]

# Display the RGB channels
plt.subplot(2, 3, 4)
plt.imshow(R, cmap='gray')
plt.title('Red')
plt.subplot(2, 3, 5)
plt.imshow(G, cmap='gray')
plt.title('Green')
plt.subplot(2, 3, 6)
plt.imshow(B, cmap='gray')
plt.title('Blue')

```

```
# Define the region of interest in the red channel
R1 = R[128:256, 85:170]   # MATLAB indexing starts from 1, Python from 0
R1_ave = np.mean(R1)

# Calculate the standard deviation of the selected region
R1_d = np.std(R1)

# Segment the red channel based on the mean and standard deviation
R2 = np.zeros_like(R)
mask = (R > R1_ave - 1.25 * R1_d) & (R < R1_ave + 1.25 * R1_d)
R2[mask] = 1

# Display the segmented red channel
plt.subplot(2, 2, 2)
plt.imshow(R2, cmap='gray')
plt.title('Segmented Red')

plt.tight_layout()
plt.show()

```

Result: