Assignment 1 - Histogram Equalization | Rohin Kumar Maheswaran

Source Code :

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

# Path for dataset
path = '/Users/rohinkumar/Desktop/Dataset'


# RGB Split Histogram Equalization


# Split into different channels and plot histogram
def analyze(data,nameoffile) :
    blue, green, red = cv2.split(data)
    hist_blue = cv2.calcHist([blue], [0], None, [256], [0, 256])
    hist_green = cv2.calcHist([green], [0], None, [256], [0, 256])
    hist_red = cv2.calcHist([red], [0], None, [256], [0, 256])

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

    # Display the image
    ax1.imshow(cv2.cvtColor(data, cv2.COLOR_BGR2RGB))
    ax1.set_title(f'{nameoffile}')
    ax1.axis('off')

    # Display the histograms
    ax2.set_title(f'Histogram of {nameoffile}')
    ax2.plot(hist_blue, color='blue', label='Blue Channel')
    ax2.plot(hist_green, color='green', label='Green Channel')
    ax2.plot(hist_red, color='red', label='Red Channel')
    ax2.set_xlim([0, 256])
    ax2.legend()

    plt.show()
```

Assignment 1 - Histogram Equalization | Rohin Kumar Maheswaran

```
def Equalizeimg(data, nameoffile):
    # Convert input image to 8-bit depth per channel if not already
    if data.dtype != np.uint8:
        data = cv2.convertScaleAbs(data)

    b, g, r = cv2.split(data)

    # Calculate histograms for each channel
    hist_blue, _ = np.histogram(b, bins=256, range=(0, 256))
    hist_green, _ = np.histogram(g, bins=256, range=(0, 256))
    hist_red, _ = np.histogram(r, bins=256, range=(0, 256))

    # Calculate cumulative distribution functions (CDF) for each channel
    cdf_blue = hist_blue.cumsum()
    cdf_green = hist_green.cumsum()
    cdf_red = hist_red.cumsum()

    # Normalize the CDF to [0, 255]
    cdf_blue = (cdf_blue / cdf_blue.max()) * 255
    cdf_green = (cdf_green / cdf_green.max()) * 255
    cdf_red = (cdf_red / cdf_red.max()) * 255

    # Use the CDF to equalize the pixel values for each channel
    equalized_b = cdf_blue[b]
    equalized_g = cdf_green[g]
    equalized_r = cdf_red[r]

    # Merge the equalized channels back into an image
    equalized_image = cv2.merge([equalized_b, equalized_g, equalized_r])

    # Convert the equalized image to 8-bit format for display
    equalized_image_8bit = cv2.convertScaleAbs(equalized_image)

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

    # Display the equalized image using plt.imshow
    ax1.imshow(cv2.cvtColor(equalized_image_8bit, cv2.COLOR_BGR2RGB))
    ax1.set_title(f'RGB Split Histogram Equalized Image ({nameoffile})')
    ax1.axis('off')

    # Display the equalized histograms
    ax2.set_title(f'RGB Split Equalized Histogram of {nameoffile}')
    ax2.plot(hist_blue, color='blue', label='Blue Channel')
    ax2.plot(hist_green, color='green', label='Green Channel')
    ax2.plot(hist_red, color='red', label='Red Channel')
    ax2.set_xlim([0, 256])
    ax2.legend()

    plt.show()



def Displayimg(image_path):
    data = cv2.imread(image_path)
    if data is None:
        print("ERROR! Couldnt load the images from the path")
        return

    filename = os.path.basename(image_path)

    analyze(data, filename)
    Equalizeimg(data, filename)


for nameoffile in os.listdir(path):
    if nameoffile.endswith((".jpg", ".jpeg")):
        image_path = os.path.join(path, nameoffile)
        Displayimg(image_path)




 #HSV colorspace Histogram equalization

def hsv_equalize(image_path):
    img = cv2.imread(image_path)
    rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    hsv_img = cv2.cvtColor(rgb_img, cv2.COLOR_RGB2HSV)

    hue, saturation, value = cv2.split(hsv_img)

    value_equalized = cv2.equalizeHist(value)
    img_hsv_equalized = cv2.merge((hue, saturation, value_equalized))
    equalized_image_rgb = cv2.cvtColor(img_hsv_equalized, cv2.COLOR_HSV2RGB)

    e_blue, e_green, e_red = cv2.split(equalized_image_rgb)
    e_hist_blue = cv2.calcHist([e_blue], [0], None, [256], [0, 256])
    e_hist_green = cv2.calcHist([e_green], [0], None, [256], [0, 256])
    e_hist_red = cv2.calcHist([e_red], [0], None, [256], [0, 256])

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

    # Display the histograms
    ax1.set_title(f'Histogram of {filename} after HSV colorspace equalization')
    ax1.plot(e_hist_blue, color='blue', label='Blue Channel')
    ax1.plot(e_hist_green, color='green', label='Green Channel')
    ax1.plot(e_hist_red, color='red', label='Red Channel')
    ax1.set_xlim([0, 256])
    ax1.legend()

    # Display the equalized RGB image
    ax2.imshow(equalized_image_rgb)
    ax2.set_title(f'Equalized image over HSV')
    ax2.axis('off')

    plt.show()


for filename in os.listdir(path):
     if filename.endswith((".jpg", ".jpeg")):
         img_path = os.path.join(path, filename)
         hsv_equalize(img_path)


# Histogram Equalization LAB Color space

def LAB_equalize(j):
    imgforLAB = cv2.imread(j)
    rgb_imgforLAB = cv2.cvtColor(imgforLAB, cv2.COLOR_BGR2RGB)
    LABimg = cv2.cvtColor(rgb_imgforLAB, cv2.COLOR_BGR2LAB)
    l_channel, a_channel, b_channel = cv2.split(LABimg)
    equalized_l_channel = cv2.equalizeHist(l_channel)
    equalized_lab_img_LAB = cv2.merge((equalized_l_channel, a_channel, b_channel))
    equalized_rgb_image = cv2.cvtColor(equalized_lab_img_LAB, cv2.COLOR_LAB2BGR)

    LAB_blue, LAB_green, LAB_red = cv2.split(equalized_rgb_image)
    l_hist_blue = cv2.calcHist([LAB_blue], [0], None, [256], [0, 256])
    l_hist_green = cv2.calcHist([LAB_green], [0], None, [256], [0, 256])
    l_hist_red = cv2.calcHist([LAB_red], [0], None, [256], [0, 256])

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))
    # Display the histograms
    ax1.set_title(f'Histogram of {filename} after LAB colorspace equalization')
    ax1.plot(l_hist_blue, color='blue', label='Blue Channel')
    ax1.plot(l_hist_green, color='green', label='Green Channel')
    ax1.plot(l_hist_red, color='red', label='Red Channel')
    ax1.set_xlim([0, 256])
    ax1.legend()

    # Display the equalized RGB image
    ax2.imshow(equalized_rgb_image)
    ax2.set_title(f'Equalized image over LAB {filename}')
    ax2.axis('off')

    plt.show()




for filename in os.listdir(path):
    if filename.endswith((".jpg", ".jpeg")):
        j = os.path.join(path, filename)
        LAB_equalize(j)



# Improvement experiments


# 1st Method - CLAHE (Contrast Limited Adaptive Histogram Equalization)


def CLAHEhist(img_path) :
    imgforCLAHE = cv2.imread(img_path)
    convertLAB = cv2.cvtColor(imgforCLAHE, cv2.COLOR_BGR2LAB)
    L, A, B = cv2.split(convertLAB)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    clahe_l = clahe.apply(L)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    clahe_l_channel = clahe.apply(L)
    enhanced_lab_image = cv2.merge((clahe_l_channel, A, B))
    enhanced_rgb_image = cv2.cvtColor(enhanced_lab_image, cv2.COLOR_LAB2BGR)


    cLAB_blue, cLAB_green, cLAB_red = cv2.split(enhanced_rgb_image)
    cl_hist_blue = cv2.calcHist([cLAB_blue], [0], None, [256], [0, 256])
    cl_hist_green = cv2.calcHist([cLAB_green], [0], None, [256], [0, 256])
    cl_hist_red = cv2.calcHist([cLAB_red], [0], None, [256], [0, 256])

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

    # Display the histograms
    ax1.set_title(f'Histogram of {filename} after CLAHE')
    ax1.plot(cl_hist_blue, color='blue', label='Blue Channel')
    ax1.plot(cl_hist_green, color='green', label='Green Channel')
    ax1.plot(cl_hist_red, color='red', label='Red Channel')
    ax1.set_xlim([0, 256])
    ax1.legend()

    # Display the equalized RGB image
    ax2.imshow(enhanced_rgb_image)
    ax2.set_title(f'Equalized image over CLAHE {filename}')
    ax2.axis('off')

    plt.show()



for filename in os.listdir(path):
    if filename.endswith((".jpg", ".jpeg")):
        img_path = os.path.join(path, filename)
        CLAHEhist(img_path)


# 2nd Method - AHE (Adaptive Histogram Equalization)

def AHE(img_path, filename):
    imgforahe = cv2.imread(img_path)
    imgforahe = cv2.cvtColor(imgforahe, cv2.COLOR_BGR2RGB)

    ar, ag, ab = cv2.split(imgforahe)

    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    r_equalized = clahe.apply(ar)
    g_equalized = clahe.apply(ag)
    b_equalized = clahe.apply(ab)

    ahe_equalized_image = cv2.merge([r_equalized, g_equalized, b_equalized])
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

    # Calculate and plot histograms
    ax1.set_title(f'Histogram of {filename} after AHE')
    ax1.hist(r_equalized.ravel(), bins=256, color='blue', alpha=0.5, label='Blue Channel')
    ax1.hist(g_equalized.ravel(), bins=256, color='green', alpha=0.5, label='Green Channel')
    ax1.hist(b_equalized.ravel(), bins=256, color='red', alpha=0.5, label='Red Channel')
    ax1.set_xlim([0, 256])
    ax1.legend()

    # Display the equalized RGB image
    ax2.imshow(ahe_equalized_image)
    ax2.set_title(f'Equalized image over AHE {filename}')
    ax2.axis('off')

    plt.show()


for filename in os.listdir(path):
    if filename.endswith((".jpg", ".jpeg")):
        img_path = os.path.join(path, filename)
        AHE(img_path, filename)
```