B09611007 陳柏霖

Part 1: Histogram of an Image

建立對照.64 檔案中每個像素的數字或字母到 0-31 的色彩深度的 dictionary

```
# Define the mapping of characters '0' to '9' and 'A' to 'V' to numerical values 0 to 31
char_to_value = {chr(48 + i): i for i in range(10)}
char_to_value.update({chr(65 + i): i + 10 for i in range(22)})
```

將檔案名稱宣告在 filenames 中

```
filenames = ['LINCOLN', 'JET', 'LISA', 'LIBERTY']
```

依序開啟檔案，將檔案存入 data，並且解決檔案末有SUB符號的問題。

```
for filename in filenames:
    # Read the .64 file and convert it to a 2D array
    file_path = filename + '.64'
    with open(file_path, 'r') as file:
        lines = file.read().splitlines()
        # Remove the last line if it contains SUB
        if lines[-1] == 'SUB': lines.pop()
        # Remove the last character if it is a SUB
        data = [[char_to_value[char] for char in line if char != 'SUB'] for line in lines]
```

將 data 存入 Numpy 陣列，成為 image_array

```
# Convert the 2D array to a NumPy array
image_array = np.array(data, dtype=np.uint8)
```

透過 cv2 將 image_array 轉成圖片，成為 image

```
# Create an image from the NumPy array
image = cv2.resize(image_array, (64, 64), interpolation=cv2.INTER_NEAREST)
```

儲存圖片，並完成 HISTOGRAM 的輸出

```
# Save the image
cv2.imwrite(filename + '.png', image)
# generate a histogram of the image
# Make the x ticks integers line up with the bin centers
bins = np.arange(33) - 0.5
plt.hist(image.ravel(), bins, color='red', alpha=0.5, ec='black')
# plt.legend(loc='best')
plt.title(filename)
plt.xlabel('Value')
plt.ylabel('Frequency')
# save the histogram
plt.savefig(filename + '_histogram.png')
#clear the plot
plt.clf()
```

Part 2: Arithmetic Operations of an Image Array

請使用者輸入要執行不同影像處理結果比較的兩個檔名

```
# Ask the user to enter the path to the image
image_name = input("Enter the file name of the first image: ")
image_path = image_name+ '.png'
# image_path = 'LISA.png'  # Replace with the path to your .64 image
image2_name = input("Enter the file name of the second image: ")
image2_path = image2_name + '.png'
```

讀取檔案並確定無問題後，創建圖表輸出的 Figure

```
# Load the image and check if it was loaded successfully
original_image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
if original_image is None:
    print("Error: Could not load the image.")
else:
    # Create a single figure for all histograms
    plt.figure(figsize=(16, 12))
```

定義函數，用來將圖片數據輸出為在 Figure 上指定位置的 Histogram

```
def plot_histogram(image, title, position, custom_color):
    plt.subplot(3, 3, position)
    # Make the x ticks integers line up with the bin centers
    bins = np.arange(33) - 0.5
    # Plot the histogram
    plt.hist(image.ravel(), bins=bins, range=(0, 32), color=custom_color, alpha=0.5, ec='black')
    plt.title(title)
    plt.xlabel("Pixel Value")
    plt.ylabel("Frequency")
    # make the bars of the histogram to align with the grid lines
    plt.grid(axis='y', alpha=0.75)
```

將原圖片的 Histogram 先置於第一個位置

```
# Plot the histogram of the original image
plot_histogram(original_image, image_name +" Original Image", 1, 'blue')
```

確認像素數據大小在範圍內後，請使用者輸入欲加減以及乘除的數值，並生成 Histogram 以供比較

```
if original_image.min() >= 0 and original_image.max() <= 32:
    # 1. Add or Subtract a Constant Value
    # Ask the user to enter a constant value
    constant_value = int(input("Enter a constant value to add: "))
    # constant_value = 10
    addition_result = np.clip(original_image + constant_value, 0, 32)  # Ensure values stay in the [0, 32] range
    subtraction_result = np.clip(original_image - constant_value, 0, 32)

    # plot_histograms(original_image, addition_result, "Original Image", "Addition Result")
    # plot_histograms(original_image, subtraction_result, "Original Image", "Subtraction Result")
    plot_histogram(addition_result, "Addition Result: +" + str(constant_value), 2, 'red')

    # 2. Multiply a Constant
    # Ask the user to enter a constant multiplier
    constant_multiplier = int(input("Enter a constant multiplier: "))
    # constant_multiplier = 2
    multiplication_result = np.clip(original_image * constant_multiplier, 0, 32)

    # plot_histograms(original_image, multiplication_result, "Original Image", "Multiplication Result")
    plot_histogram(multiplication_result, "Multiplication Result: *" + str(constant_multiplier), 3, 'red')
```

讀取第二張照片，並輸出 Average 與 Difference: $g(x,y) = f(x,y) - f(x-1,y)$ 的 Histogram 結果

```
    # 3. Average of Two Images
    # Load another .64 image for averaging (replace with the path to the second image)
    #
    image2 = cv2.imread(image2_path, cv2.IMREAD_GRAYSCALE)
    if image2 is not None:
        average_result = np.clip((original_image + image2) // 2, 0, 32)

        # plot_histograms(original_image, average_result, "Original Image", "Average Result")
        plot_histogram(average_result, " Average Result of " + image_name + " and " + image2_name, 4, 'green')

        # 4. Pixel-wise Difference
        diff_result = np.zeros_like(original_image)
        diff_result[1:, :] = original_image[1:, :] - original_image[:-1, :]

        # plot_histograms(original_image, diff_result, "Original Image", "Difference Result")
        plot_histogram(diff_result, "Difference Result of " + image_name + " and " + image2_name, 5, 'green')
    else:
        print("Error: Could not load the second image.")
else:
    print("Error: Image pixel values are not in the range [0, 32].")
```

設置 figure 的標題與格式

```
# Adjust layout and display the combined figure
plt.subplots_adjust(top=0.9, bottom=0.2, hspace=0.5)
plt.savefig(image_name + " and " + image2_name + 'process_comparison_histogram.png')
plt.show()
```