**Comment**

灰階圖的直方圖均衡化相對簡單，可以直接使用 OpenCV 提供的 cv2.equalizeHist() 來處理，因為它僅針對單通道的亮度資訊進行調整。

然而，對於彩色圖像，不能直接對每個通道獨立應用直方圖均衡化，否則可能會導致色彩失真。因此，一種更合適的方法是 **先將彩色圖像轉換為灰階圖，並在灰階圖上應用直方圖均衡化，以增強亮度與對比度**。接著，將均衡化後的灰階影像視為新的亮度參考，並根據其與原始灰階影像的比例來調整原始彩色圖像的 RGB 值。這樣可以在提升影像細節與對比的同時，保留原始的色彩關係，使影像看起來更自然且不會有色偏的問題。

**Photos**

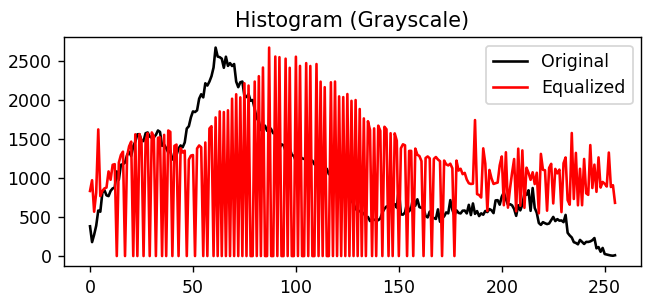
Origin gray image



Histogram gray image



Histogram (gray)



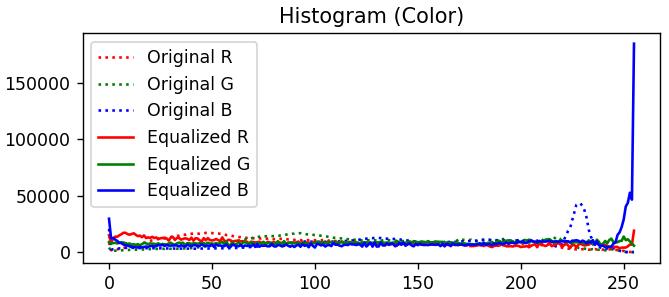
Origin color image



Histogram color image



Histogram (color)



**Program**

import cv2

import numpy as np

import matplotlib.pyplot as plt

gray\_image = cv2.imread('cat.jpg', cv2.IMREAD\_GRAYSCALE)

color\_image = cv2.imread('landscape.jpg')

if gray\_image is None or color\_image is None:

    print("Error: Image not found!")

    exit()

### 灰階圖

equalized\_gray = cv2.equalizeHist(gray\_image)

hist\_gray\_original = cv2.calcHist([gray\_image], [0], None, [256], [0, 256])

hist\_gray\_equalized = cv2.calcHist([equalized\_gray], [0], None, [256], [0, 256])

### 彩圖

gray\_from\_color = cv2.cvtColor(color\_image, cv2.COLOR\_BGR2GRAY)

equalized\_gray\_from\_color = cv2.equalizeHist(gray\_from\_color)

# 避免除0

gray\_from\_color = np.maximum(gray\_from\_color, 1)

# 計算新的 RGB 值 (r’, g’, b’) = (r, g, b) \* G’ / G

color\_image\_float = color\_image.astype(np.float32)

equalized\_gray\_from\_color\_float = equalized\_gray\_from\_color.astype(np.float32)

gray\_from\_color\_float = gray\_from\_color.astype(np.float32)

new\_color\_image = (color\_image\_float \* (equalized\_gray\_from\_color\_float[:, :, None] / gray\_from\_color\_float[:, :, None]))

new\_color\_image = np.clip(new\_color\_image, 0, 255).astype(np.uint8)

hist\_color\_original\_r = cv2.calcHist([color\_image], [2], None, [256], [0, 256])

hist\_color\_original\_g = cv2.calcHist([color\_image], [1], None, [256], [0, 256])

hist\_color\_original\_b = cv2.calcHist([color\_image], [0], None, [256], [0, 256])

hist\_color\_equalized\_r = cv2.calcHist([new\_color\_image], [2], None, [256], [0, 256])

hist\_color\_equalized\_g = cv2.calcHist([new\_color\_image], [1], None, [256], [0, 256])

hist\_color\_equalized\_b = cv2.calcHist([new\_color\_image], [0], None, [256], [0, 256])

plt.figure(figsize=(15, 5))

plt.subplot(2, 3, 1)

plt.title("Original Grayscale")

plt.imshow(gray\_image, cmap='gray')

plt.subplot(2, 3, 2)

plt.title("Equalized Grayscale")

plt.imshow(equalized\_gray, cmap='gray')

plt.subplot(2, 3, 3)

plt.title("Histogram (Grayscale)")

plt.plot(hist\_gray\_original, color='black', label='Original')

plt.plot(hist\_gray\_equalized, color='red', label='Equalized')

plt.legend()

plt.subplot(2, 3, 4)

plt.title("Original Color Image")

plt.imshow(cv2.cvtColor(color\_image, cv2.COLOR\_BGR2RGB))

plt.subplot(2, 3, 5)

plt.title("Equalized Color Image")

plt.imshow(cv2.cvtColor(new\_color\_image, cv2.COLOR\_BGR2RGB))

plt.subplot(2, 3, 6)

plt.title("Histogram (Color)")

plt.plot(hist\_color\_original\_r, color='r', linestyle='dotted', label='Original R')

plt.plot(hist\_color\_original\_g, color='g', linestyle='dotted', label='Original G')

plt.plot(hist\_color\_original\_b, color='b', linestyle='dotted', label='Original B')

plt.plot(hist\_color\_equalized\_r, color='r', label='Equalized R')

plt.plot(hist\_color\_equalized\_g, color='g', label='Equalized G')

plt.plot(hist\_color\_equalized\_b, color='b', label='Equalized B')

plt.legend()

plt.tight\_layout()

plt.show()

cv2.imwrite('equalized\_cat.jpg', equalized\_gray)

cv2.imwrite('equalized\_landscape.jpg', new\_color\_image)