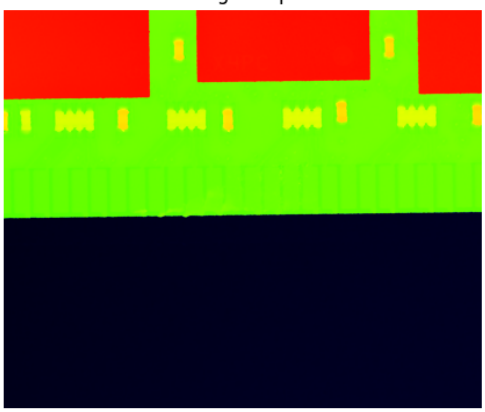
功課要求

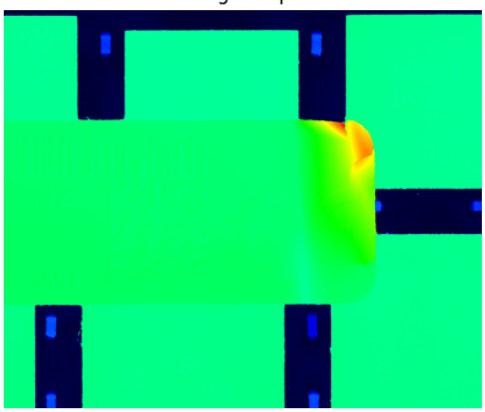
附件中為三張利用將晶片高度以色彩視覺化後的圖片。
請設計一個基於Run-Length 的壓縮法方，對圖檔作無失真壓縮後儲存成新檔案。

部落格上應敘述你的壓縮方法，提供壓縮檔之格式，並計算三張圖的平均壓縮率(compression ratio)。

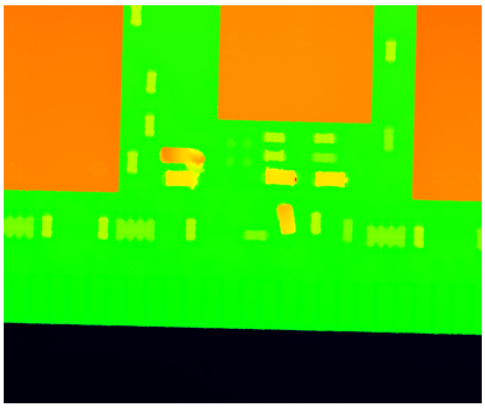
img1.bmp



img2.bmp



img3.bmp



{: w="650", h="650"}

成果

Image Name	Original Size	Compress Size	Compression Ratio
img1.dat	14665254 bytes	8074086 bytes	1.81634
img2.dat	14665254 bytes	13789338 bytes	1.06352
img3.dat	14665254 bytes	7488783 bytes	1.9583

Average compression ratio: 1.6127177979475829.

{: w="500", h="500"}
得到檔案的壓縮率與平均壓縮率

開發環境

OS	Editor	Language	OpenCV
Windows 10	Visual Studio Code	Python 3.9.16	OpenCV 4.5.4

實作

> 本次程式碼

使用的 libraries 如下：

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

1/ 利用迴圈讀入三張圖片

建立一個儲存三張圖片路徑的 list ，使用迴圈搭配讀入圖片並送至 `compress(original_img, compress_file)` 進行壓縮，取得壓縮率並儲存在 `compress_ratio[i]` 中。

```
def main():
    img_path = ["img1.bmp", "img2.bmp", "img3.bmp"]
    compress_path = ["img1.dat", "img2.dat", "img3.dat"]
    # 跑過三張圖片
    compress_ratio = [0, 0, 0]
    print("| Image Name | Original Size  | Compress Size  | Compression Ratio |")
    print("| ---------- | -------------- | -------------- | ----------------- |")
    for i in range(0, len(img_path)):
        original_img = img_path[i]
        compress_file = compress_path[i]

        compress_ratio[i] =  compress(original_img, compress_file)
```

2/ 壓縮圖片

2.1/ 計算圖片大小

使用 `os.path.getsize(filename)` 取得圖片大小，並使用 `with open(compress_file, "w") as file` 寫入壓縮檔案中，做為解壓縮時建立 array 的參考。

```
img = cv2.imread(img_path)
    with open(compress_file, "w") as file:
        # 記下圖片大小
        img_size = [img.shape[0], img.shape[1]]
        file.write(str(img_size[0]) + ", " + str(img_size[1]) + "\n")
```

2.2/ 將二維的圖片轉為一維

三個 channel 的值不會一樣，所以三個通道必須分開儲存。

一維的圖片比較好操控，而且能夠增加壓縮的效率，所以使用 `flatten()` 將二維的矩陣轉換至一維。

```
# 三個 channel 分開跑
for channel in range(0, 3):
    img_channel = img[:, :, channel]
    img_flat = img_channel.flatten()
```

2.3/ Run-Length Encoding

遍歷矩陣中所有的 pixel ，如果當前 pixel 和上一個 pixel 相等，數量就 + 1，不相等就將當前 pixel 的值和數量寫入 .dat 檔中。

並且利用 `\n` 來分隔每個 channel 的資料。

```
last_pixel = 0
pixel_count = 0

# 遍歷每個 pixel
for pixel in img_flat:
    if pixel == last_pixel:
        pixel_count += 1
    else:
        file.write(str(last_pixel) + ", " + str(pixel_count) + ", ")
        pixel_count = 1
        last_pixel = pixel

# 存入最後一筆資料
file.write(str(last_pixel) + ", " + str(pixel_count) + "\n")
```

2.4/ 計算壓縮率

首先，利用 `os.path.getsize(filename)` 取得檔案大小。

壓縮率的計算為 原檔案大小 / 壓縮檔案大小，相除之後使用 formatted string 顯示在表格中。

```
original_size = os.path.getsize(img_path)
compress_size = os.path.getsize(compress_file)
compress_ratio = original_size / compress_size

print(f"| {compress_file:10} | {str(original_size):7} bytes | {str(compress_size):8} bytes | {st

return compress_ratio
```

3/ 解壓縮圖片

在 `main()` 中呼叫 `decompress(compress_file)` 來將剛才產生的 dat 檔轉換回圖片並使用 plt 顯示。

```
decompress_img = decompress(compress_file)
plt.subplot(2, 2, i + 1)
plt.imshow(decompress_img)
plt.title(img_path[i])
plt.axis("off")
```

3.1/ 創建畫布

使用 `img_size = file.readline().rstrip('\n').split(", ")` 取得圖片的大小之後，建立一個擁有三個元素的二維 list ，每一個元素是儲存 B, G, R 單個通道的資料，每一個元素的大小為 `img_size[0] * img_size[1]` 。

同時，使用 `file.read().split("\n")` 讀取剩下的資料，每一個通道的值是使用 `\n` 做分割，其中每一個 pixel 的值用 `,` 分割。

```
with open(compress_file, "r") as file:
    # 讀取圖片大小
    img_size = file.readline().rstrip('\n').split(", ")
    # 讀取剩下的資料
    row_data = file.read().split("\n")
    data = [row_data[i].split(", ") for i in range(0, len(row_data))]
    # 創建畫布
    image_bgr = [np.zeros(int(img_size[0]) * int(img_size[1]), dtype=np.uint8), np.zeros(int(img
```

3.2/ 將 pixel 填上 array

`image_bgr` 中，依序填入 B, G, R 通道的資料。

```
for channel in range (0, 3):
    pixel_count = 0
    for index in range(0, len(data[channel]), 2):
        pixel_length = pixel_count + int(data[channel][index + 1])
        for i in range(pixel_count, pixel_length):
            image_bgr[channel][i] = np.uint8(data[channel][index])
        pixel_count = pixel_length
```

3.3/ Merge 三個通道成一張圖片

將填好值的一維 array 使用 `reshape()` 轉換成二維矩陣，並使用 `cv2.merge()` 合併三個通道，使其變成 BGR 的彩色圖片。

由於要在 plt 做顯示，所以使用 `cv2.cvtColor()` 將 BGR 轉換成 RGB 。

```
# 將三個通道合併
image_b = image_bgr[0].reshape(int(img_size[0]), int(img_size[1]))
image_g = image_bgr[1].reshape(int(img_size[0]), int(img_size[1]))
image_r = image_bgr[2].reshape(int(img_size[0]), int(img_size[1]))
result_img = cv2.merge([image_b, image_g, image_r])
# 將 BGR 改成 RGB
image = cv2.cvtColor(result_img, cv2.COLOR_BGR2RGB)
# return 解壓縮完的圖片
return image
```

4/ 計算平均壓縮率

將三張圖片的壓縮率加總後平均並輸出，最後使用 `plt.show()` 顯示圖片。

```
average_ratio = sum(compress_ratio) / len(compress_ratio)
print("Average compression ratio: " + str(average_ratio) + ".")

plt.show()
```

總結

Run-length encoding 可以用來壓縮相同顏色連續出現的圖片，不過如果圖片沒有大量相同的顏色相鄰，壓縮效果可能會不太好。

參考資料

- 图像处理（一）：基于行程编码的图像压缩python实现
- numpy.reshape
- 運行長度編碼 (RLE) 數據壓縮算法