HW1 Report

109550156 曾偉杰

1. Method

(1) Exchange Position

將原圖複製一份，然後將要交換的區域直接從新複製的圖中的 RGB 值直接 assign 給原本的圖片就好。

(2) Gray Scale

迭代原本圖中的每個 pixel，然後將每個 pixel 的 RGB 值相加取平均，然後將 3 個通道都設為該平均值，就可以把圖片變為灰階。

(3) Intensity Resolution

先把圖片轉成灰階，再把灰階圖片的 3 個通道值除以 64 且只取整數部分，最後將 3 個通道乘以 64 即可。

(4) Color Filter – Red

```
def color_filter_red(img):
    for row in range(img.shape[0]):
        for col in range(img.shape[1]):
            if not (img[row, col, 2] > 150
                    and img[row, col, 2] * 0.6 > img[row, col, 0]
                    and img[row, col, 2] * 0.6 > img[row, col, 1]):

                gray_value = np.sum(img[row, col]) / 3
                for i in range(3):
                    img[row, col, i] = gray_value
    return img
```

對於每個 pixel，判斷 R 值是否大於 150 和 R * 0.6 > B 和 R * 0.6 > G，只要有一個條件不成立，就把該 pixel 轉成灰階。

(5) Color Filter – Yellow

類似於(4)，對於每個 pixel，判斷條件變成 (G + R) * 0.3 是否大於 B 和 G 減 R 的絕對值是否 50 小於 50。

(6) Channel Operation

直接把圖像的 G 值通道乘 2 即可，因為圖像的型別是 np.uint8，為了避免 overflow，需先轉換成更大的型別，此部分這次是轉成 np.int32，最後再轉回 np.uint8。

(7) Bilinear Interpolation

```
def bilinear_interpolation(img, scale):
    height, width = img.shape[0], img.shape[1]
    blank_img = np.zeros((img.shape[0], img.shape[1], 3), np.uint8)
    for row in range(height):
        for col in range(width):
            y = row/scale
            x = col/scale
            y1, y2 = int(y), min(int(y + 1), int(height/2))
            x1, x2 = int(x), min(int(x + 1), int(width/2))

            f11, f12 = img[y1, x1], img[y1, x2]
            f21, f22 = img[y2, x1], img[y2, x2]

            fx1 = (x2-x)/(x2-x1)*f11 + (x-x1)/(x2-x1)*f12
            fx2 = (x2-x)/(x2-x1)*f21 + (x-x1)/(x2-x1)*f22

            fxy = (y2-y)/(y2-y1)*fx1 + (y-y1)/(y2-y1)*fx2

            blank_img[row, col] = fxy.astype(np.uint8)

    return blank_img
```

迭代輸出圖的每個 pixel，然後對該 pixel 的座標除以 scale 的值，即得到原圖座標 x, y，然後在原圖座標找到最相近的四個點 (圖中的 x1, x2, y1, y2 就是相近四點的座標)，然後就利用 x, y, x1, y1, x2, y2 做雙線性插值即可。

(8) Bicubic Interpolation

```
def cubic_polynomial(row, x):
    p0, p1, p2, p3 = row.astype(float)
    a = (-1/2) * p0 + (3/2) * p1 + (-3/2) * p2 + (1/2) * p3
    b = p0 + (-5/2) * p1 + 2 * p2 + (-1/2) * p3
    c = (-1/2) * p0 + (1/2) * p2
    d = p1
    f = a * math.pow(x, 3) + b * math.pow(x, 2) + c * x + d
    for i in range(3):
        if f[i] > 255:
            f[i] = 255
        elif f[i] < 0:
            f[i] = 0
    return f
```

方法跟雙線性插值差不多，對每個 pixel 的座標除以 scale 的值得到原圖座標 x, y 後，變成在原圖找到相近的 16 個點，然後把水平的 4 個點的值帶入由上而下帶入 cubuic_polynomial() 得到 4 個垂直的單三次插值，然後

對這 4 個垂直點的值再帶入一次 cubuic_polynomial()，並得到雙三次插值的結果。

2. Result

3. Feedback

對於這次的作業，讓我更了解到了如何使用 opencv 對圖像處理做基本操作，在這次的作業中，我覺得比較困難的是倒數兩個插值法，需要先花時間想要怎麼做，除此之外，需要特別注意在處理 RGB 通道時會不會 overflow，因握 cv2.imread()讀出來的型別是 np.uint8，最多只到 255，如果 overflow，會造成印出來的圖片顏色無法預期。