機器視覺作業報告

題目

姓名:岳紀伶
學號:01057113
日期:2024/04/16

方法

做法:
根據需要的 filter 製作 kernel，Version1 直接做高斯模糊，所以只需要做 Gaussian filter 就好。Version2 需要先轉換到 YUV 座標再做高斯模糊，最後轉回 RGB，所以需要增加一個轉換到 YUV 座標的矩陣，並透過將這個矩陣做 inverse 得到轉回 RGB 的矩陣。

重要程式片段:
Version1

```
def gaussian_kernel(size, sigma):
    kernel = np.fromfunction(lambda x, y: (1/(2*np.pi*sigma**2)) * np.exp(-((x - size//2)**2 + (y - size//2)**2)/(2*sigma**2)), (size, size))
    print(kernel)
    return kernel / np.sum(kernel)
```

藉由 np.fromfunction 從 0 到 size 把值帶入 x、y，形成一個 size*size(5*5)的矩陣，再把這個矩陣正規化得到 Gaussian kernel

```
kernel = gaussian_kernel(5, sigma=1)
kernel = kernel[:, :, np.newaxis, np.newaxis]
```

為了符合 DepthwiseConv2D 的格式，要把 kernel 擴充成四維，增加 in_channels 和 out_channels

```
plt.figure(figsize=(12, 8))
plt.suptitle('Version1')
for j in range(1, 4):
    img = cv2.imread("TestImage{}.jpg".format(j))
    plt.subplot(4, 3, j)
    plt.imshow(img[:,:,[2,1,0]])
    plt.axis(False)
    plt.title('Original')

    a = model.predict([img[np.newaxis,:,:,[2,1,0]],np.array([[1.0]])])
    plt.subplot(4, 3, j + 3)
    plt.imshow(np.clip(a,0,255).astype(np.uint8)[0,...])
    plt.axis(False)
    plt.title('k=1')

    for i in range(1, 3):
        a = model.predict([img[np.newaxis,:,:,[2,1,0]],np.array([[float(5*i)]])])
        plt.subplot(4, 3, 3*(i+1) + j)
        plt.imshow(np.clip(a,0,255).astype(np.uint8)[0,...])
        plt.axis(False)
        plt.title('k={}'.format(float(5*i)))

plt.tight_layout()
plt.show()
```

透過 for 迴圈依序讀入 TestImage1、TestImage2、TestImage3，由於有三張測試照片，且每張測試照片要產生三種變化，加上原圖後共 12 張，形成 4 列三行的格式，因此用 plt.subplot(4, 3, j)來表示圖片位置，j 為原圖的位置，下一種變化的位置需要加三。

Version2

```
yuv_conversion_matrix = np.array([[0.299, 0.587, 0.114],
                                  [-0.14713, -0.28886, 0.436],
                                  [0.615, -0.51499, -0.10001]]).T
rgb_conversion_matrix = np.linalg.inv(yuv_conversion_matrix)
```

Version2 在 Version1 的基礎上增加了 RGB 到 YUV 的轉換和 YUV 到 RGB 的轉換，YUV 到 RGB 可以透過將 RGB 到 YUV 的轉換矩陣取 inverse 得到

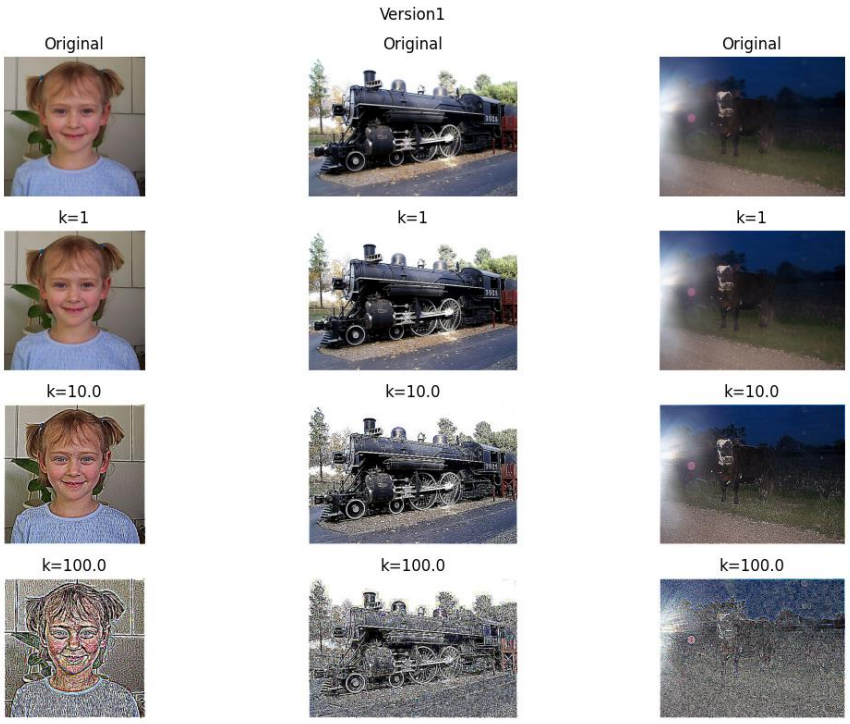
結果

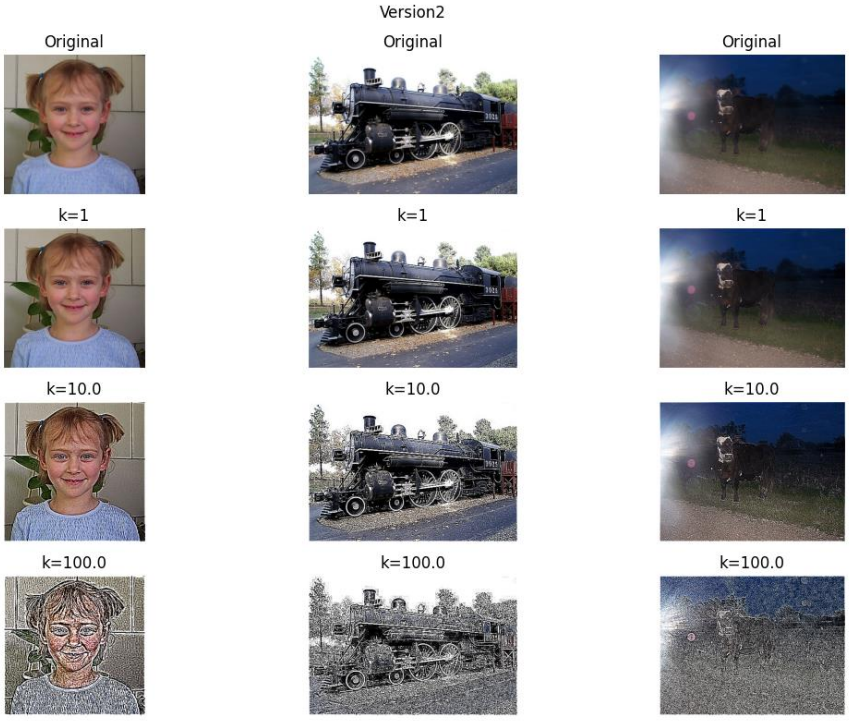
K=0.5、1.0、5.0、10.0

Version1

Version2

K=0.5、1.0、10.0、100.0





在 k=0.5、1.0、5.0、10.0 的情況下，兩個版本無法直接分辨差異。但若將 k 調大，可以發現在 version1 相較於 version2 顏色的飽和度更高

結論

在經過這次作業後，對於 kernel 的運作方式更加清楚，也發現如果要對 RGB 進行座標轉換需要將矩陣轉置才能得到正確的結果。另外在寫作業的過程中，我也發現我的寫法需要在 keras 3 中才能順利執行，否則會有 TypeError。

參考文獻

Chatgpt

https://keras.io/api/layers/convolution_layers/depthwise_convolution2d/

https://medium.com/@RiwajNeupane/convolutions-blurring-and-sharpening-images-44559460977d

https://antonmilev.medium.com/implement-image-color-filters-using-keras-conv2d-layers-3f2682105b7c