HW 2 report

Group 5

0516222 許芳瑀 0516209 呂淇安 0516326 陳廷達

1. Introduction

   In task 1, there are many similar images pairs. Our goal is using filter to get the useful data, and blend two images to a reasonable image.

   In task 2, we need to construct Gaussian and Laplacian 5-layer image pyramids. This work help us observe the residual between two pictures within different resolutions.

   In task 3, we have some Prokudin-Gorskii photo. These photos are took by special method and records negative values of R, G, B respectively. The goal is to recover the original image by aligning the red channel and the green channel to the blue channel.

2. Implementation procedure

   Task1 :

   First, obtain the images and check whether the images are the same size. If they have different sizes, resize the smaller one to the bigger one.

```
## our resize
First_B, Second_B = My_resize(Img1 = First_B, Img2 = Second_B, Diff_x = Diff_x, Diff_y = Diff_y)
First_G, Second_G = My_resize(Img1 = First_G, Img2 = Second_G, Diff_x = Diff_x, Diff_y = Diff_y)
First_R, Second_R = My_resize(Img1 = First_R, Img2 = Second_R, Diff_x = Diff_x, Diff_y = Diff_y)
## our resize done
```

   The idea for image hybridization is to combine a low-pass filtered image with another high-pass filtered image.

   First, transform the original image to the low-pass filtered image by using the gaussian_filter function made by ourselves. There is a

parameter “D0”, which is called the “cutoff frequency”, controls how much frequency to leave in the transformed image and affects the blurriness of the low-pass filtered image.

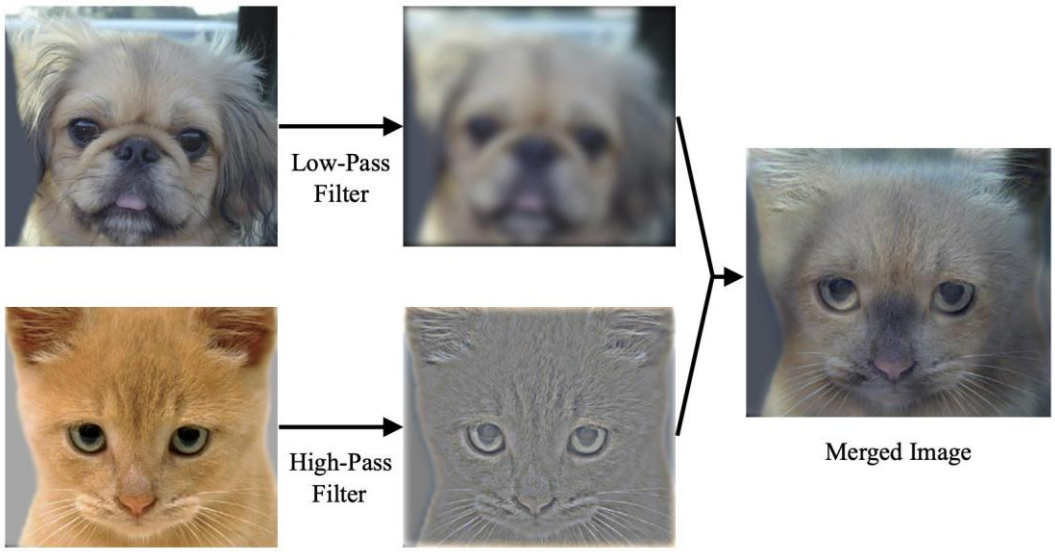
```
### Low
L_B = HFilter(FourierShift_2_B, D0 = 15000, H1L0 = 0)
L_G = HFilter(FourierShift_2_G, D0 = 15000, H1L0 = 0)
L_R = HFilter(FourierShift_2_R, D0 = 15000, H1L0 = 0)
```

Then generate a high-pass filtered version of another image by subtracting a blurred version of the image itself.

```
### High
H_B = HFilter(FourierShift_1_B, D0 = 70000, H1L0 = 1)
H_G = HFilter(FourierShift_1_G, D0 = 70000, H1L0 = 1)
H_R = HFilter(FourierShift_1_R, D0 = 70000, H1L0 = 1)
```

Finally, blend two images by obtaining the real part.

```
## Obtain the real part
### mix
MixB = np.fft.ifftshift(H_B + L_B)
MixG = np.fft.ifftshift(H_G + L_G)
MixR = np.fft.ifftshift(H_R + L_R)

### inverse
RealPart_B = np.fft.ifft2(MixB)
RealPart_G = np.fft.ifft2(MixG)
RealPart_R = np.fft.ifft2(MixR)

### REAL PART??????
Pre_FinalImg = np.dstack((RealPart_R, RealPart_G, RealPart_B))
Pre_FinalImg = UnderBound(InputImg = Pre_FinalImg)
FinalImg = np.absolute(Pre_FinalImg)

## Obtain done

## show
OriginImg1 = cv2.merge([First_R, First_G, First_B])
OriginImg2 = cv2.merge([Second_R, Second_G, Second_B])
```



Task2 :

First, read the picture and append this full-sized picture in ‘gau_pyramid’, which will store 5-layer gaussian pyramid.

```
img = cv2.imread('task1and2_hybrid_pyramid/0_Afghan_girl_after.jpg')
shape = img.shape
ori_w = shape[0]
ori_h = shape[1]
dir = '0'
```

```
gau_pyramid = list()
gau_pyramid.append(img)
# create gaussian matrix
gaussian_kernel = gau()
```

Second, use for loop to do the gaussian blur and downsampling to get other 4 layers. Also, collect magnitude spectrum of each pictures.

```
index=[2,4,8,16]
for sub_size in index:
    w = math.floor(ori_w / sub_size)
    h = math.floor(ori_h / sub_size)
    sub_img = np.zeros((w, h, 3))
    ff = np.zeros((w, h,3))
    for i in range(0,3):
        temp = gau_pyramid[-1][:,:,i]
        new = np.zeros((temp.shape[0], temp.shape[1]))
        pad = np.pad(temp,((1,1),(1,1)),'constant',constant_values = (0,0))
        for x in range(1, temp.shape[0] + 1):
            for y in range(1, temp.shape[1] + 1):
                new[x-1,y-1] = gaussian_kernel[0,0]*pad[x-1,y-1]+gaussian_kernel[1,0]*pad[x,y-1]+gaussian_kernel[2,0]*pad[x+1,y-1]
        sub_img[:,:,i] = subsampling(new.astype('uint8'), 2)
```

```
ff[:,:,i] = 15*np.log(np.abs(np.fft.fftshift(np.fft.fft2(sub_img[:,:,i]))))
```

```
def subsampling(img, sub_size):
    size_w = img.shape[0]
    size_h = img.shape[1]
    new_w = math.floor(size_w / sub_size)
    new_h = math.floor(size_h / sub_size)
    new = np.zeros((new_w, new_h))
    for x in range(0, new_w):
        for y in range(0, new_h):
            new[x, y] = img[x*sub_size, y*sub_size]
    return new
```

Third, create laplacian pyramid (different from step2, doing upsampling) and corresponding spectrums.

```
lap_pyramid = list()
lap_pyramid.append(gau_pyramid[-1])
for index in range(3, -1, -1):
    w = gau_pyramid[index].shape[0]
    h = gau_pyramid[index].shape[1]
    up_img = np.zeros((w, h, 3))
    ff = np.zeros((w, h, 3))
    for i in range(0, 3):
        temp = upsampling(lap_pyramid[-1][:,:,i], w, h)
        new = np.zeros((temp.shape[0], temp.shape[1]))
        pad = np.pad(temp,((1,1),(1,1)),'constant',constant_values = (0,0))
        for x in range(1, temp.shape[0] + 1):
            for y in range(1, temp.shape[1] + 1):
                new[x-1,y-1] = gaussian_kernel[0,0]*pad[x-1,y-1]+gaussian_kernel[1,0]*pad[x,y-1]+gaussian_kernel[2,0]*pad[x+1,y-1
        up_img[:,:,i] = new
        up_img[:,:,i] = new.astype('uint8')
        # get the spectrum
        ff[:,:,i] = 15*np.log(np.abs(np.fft.fftshift(np.fft.fft2(up_img[:,:,i]))))
```

```
def upsampling(img, new_w, new_h):
    size_w = img.shape[0]
    size_h = img.shape[1]
    new = np.zeros((new_w, new_h))
    for x in range(0, size_w):
        for y in range(0, size_h):
            new[2*x, 2*y] = img[x,y]
    return new
```

Task3 :

First, read in the picture data and split it to three part R, G, B

for some pictures whose height cannot divided by three, we cut the border off, this will not affect the result since the border doesn't contain useful informations.

```
#cut boarder
w, h = img.shape
print('origin:', w, h)
img = img[int(w*0.01) : int(w-w*0.02), int(h*0.05) : int(h-h*0.05)]
w, h = img.shape
print('small:', w, h)
```

```
#spilt RGB
height = int(w/3)
blue = img[0 : height, :]
green = img[height : 2*height, :]
red = img[2*height : 3*height, :]
print('origin size rgb', red.shape, green.shape, blue.shape)
```

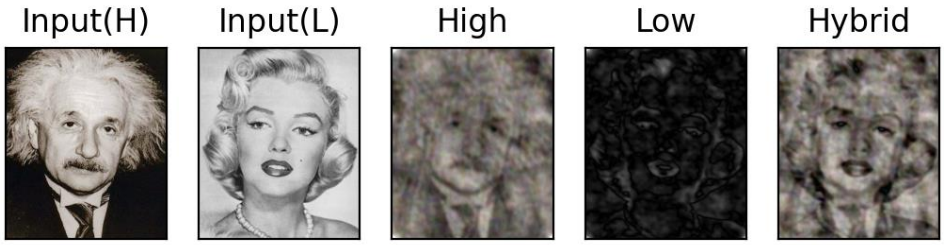
Then we have to align the images, we use normalized cross-correlation here, and the window size is 20. Function *NccAlign* will give the best start index of R and G aligning to B.

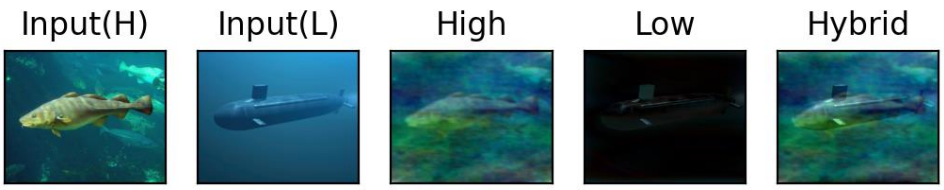
```
def Ncc(a, b):
    a = a-a.mean(axis=0)
    b = b-b.mean(axis=0)
    return np.sum(((a/np.linalg.norm(a)) * (b/np.linalg.norm(b))))

def NccAlign(a, b, window_size):
    min_ncc = -1
    ivalue = np.linspace(-window_size, window_size, 2*window_size, dtype=int)
    jvalue = np.linspace(-window_size, window_size, 2*window_size, dtype=int)
    for i in ivalue:
        for j in jvalue:
            nccDiff = Ncc(a, np.roll(b, [i, j], axis=(0, 1)))
            if nccDiff > min_ncc:
                min_ncc = nccDiff
                output = [i, j]
    return output
```

In the end, we use Image.fromarray() turn the nparray colored into an image.

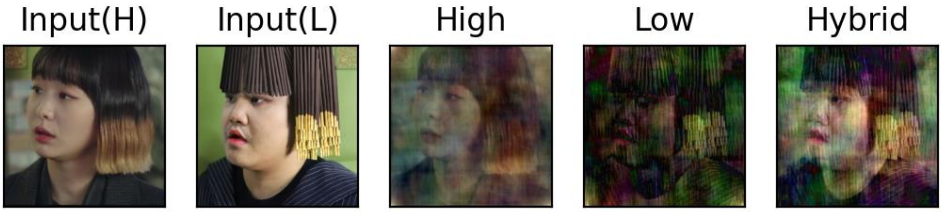
3. Experimental results (of course you should also try your own images)

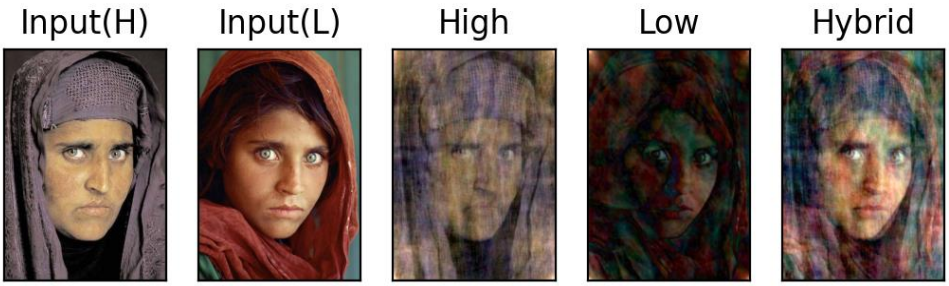
   Task1:

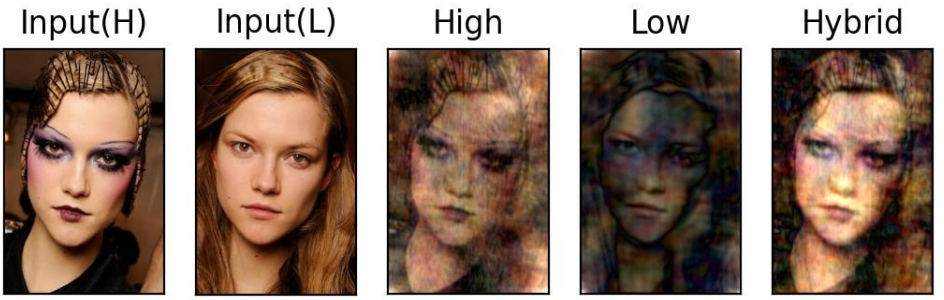



Input(H)
Input(L)
High
Low
Hybrid

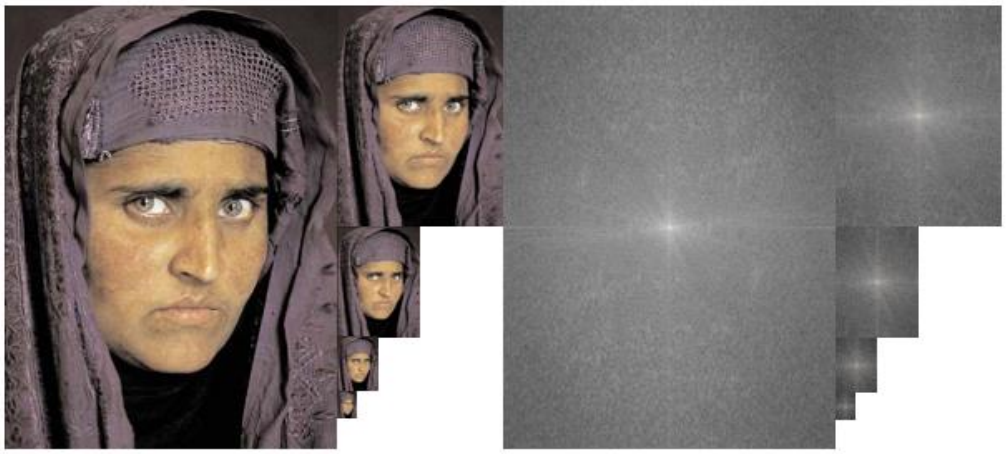

Input(H)
Input(L)
High
Low
Hybrid

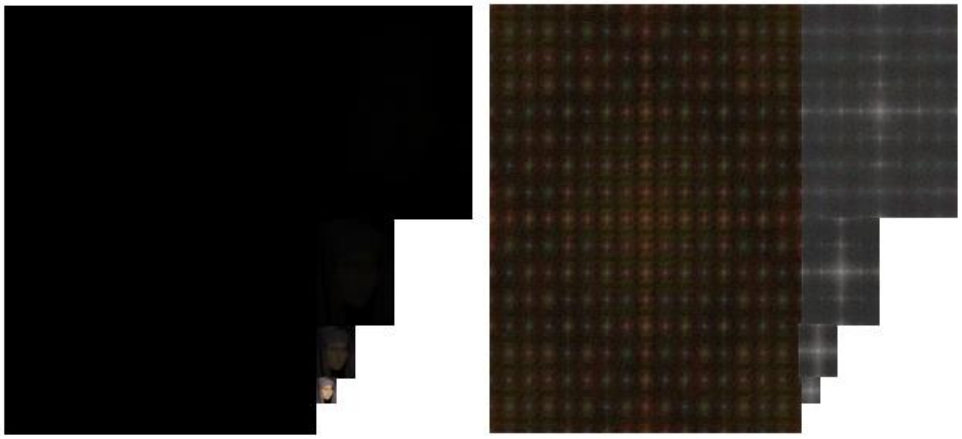

Input(H)
Input(L)
High
Low
Hybrid


Input(H)
Input(L)
High
Low
Hybrid

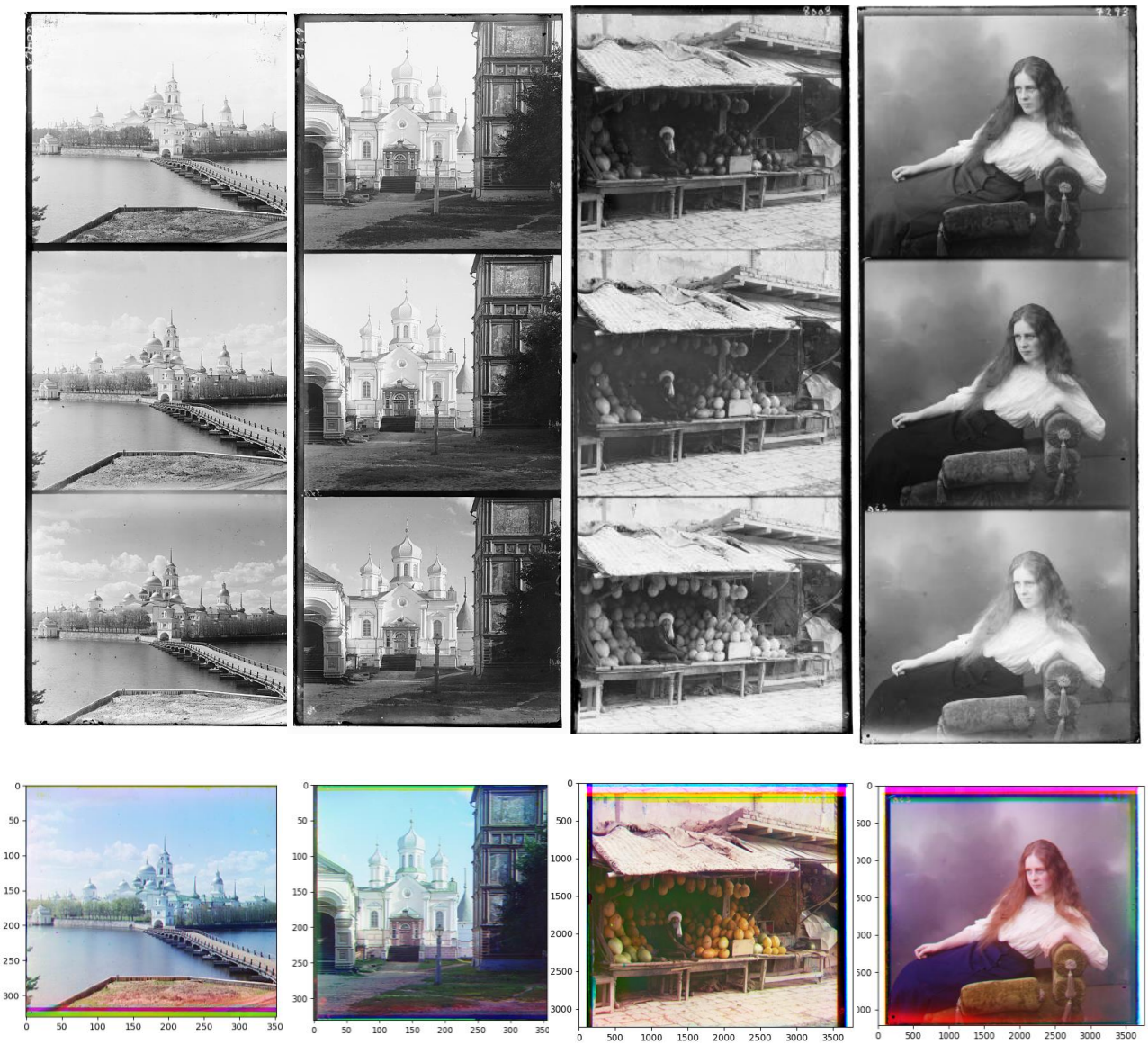

Input(H)
Input(L)
High
Low
Hybrid

Task2:





Task 3:



4. Discussion (what difficulties you have met? how you resolve them?)

Task 1:

In the high filter and low filter, parameter “D0” is too hard to determine. When “D0” is much higher, the blending image would be weird. When “D0” is much lower, there is nothing in the low pass image. In the end, we try and error many combinations, and get a better parameter.

```
def HFilter(InputImg, D0, H1L0):
    process_img = InputImg
    Dim_x, Dim_y = InputImg.shape
    for x in range(Dim_x):
        for y in range(Dim_y):
            if H1L0 == 1:
                if InputImg[x][y] < D0:
                    InputImg[x][y] = 0
                else:
                    InputImg[x][y] = InputImg[x][y] * (1 - GaussianValue(InputImg[x][y], D0))
            else:
                if InputImg[x][y] > D0:
                    InputImg[x][y] = 0
                else:
                    InputImg[x][y] = InputImg[x][y] * (GaussianValue(InputImg[x][y], D0))
    return InputImg
```

Task 2:

When creating the pictures of spectrums, first I only use `np.abs(np.fft.fftshift(np.fft.fft2(up_img[:,:,i])))`

And then the picture show nothing(or something people can't identify). So I rescale the coefficient

```
15*np.log(np.abs(np.fft.fftshift(np.fft.fft2(up_img[:,:,i]))))
```

The picture successfully show obvious features .

Task 3:

There are totally 13 pictures in the dataset, 4 of them are jpg file and the others are tif file, which are so big that we can't just roll over all pixels(It costs too much time). So we down sampled the Image to 1/10 with function *subsampling* in Task 2. Then multiply the output of NccAlign by 10 to get our real index. It works good on most cases.

```
# resize to 1/10
new_w = math.floor(w/10)
new_h = math.floor(h/10)
#smallimg = img.resize((new_w, new_h), Image.NEAREST)
img = np.asarray(img)
#smallimg = np.asarray(smallimg)
smallimg = subsampling(img, 10)
print(img.shape)
print('w', w, 'h', h)
print('w_small', new_w, 'h', new_h)
```

```
#allign
alignGtoB = NccAlign(blue, green, 20)
alignRtoB = NccAlign(blue, red, 20)
print(alignGtoB, alignRtoB)
g=np.roll(green_, [alignGtoB[0]*10, alignGtoB[1]*10], axis=(0, 1))
r=np.roll(red_, [alignRtoB[0]*10, alignRtoB[1]*10], axis=(0, 1))
print('final rgb', r.shape, g.shape, blue_.shape)
```

Another point is the RGB value in tif file(0~65535) is different from jpg(0~255), so we have to deal with it or we will get a overexposed photo.

5. Conclusion
   In task 1, we know that how to blend two images together. Using two filters which are high and low, we figure out two images. We overlapped them and obtain the real part. Finally, the hybrid the is computed. In task 2, we complete gaussian blur and downsampling as well as laplacian pyramid and upsampling, they are useful in computer vision. Task three is a simple example of image aligning and matching, for large image, downsampling which we completed in task 2 is used.

6. Work assignment plan between team members.

   Task 1：0516326 陳廷達

   Task 2：0516222 許芳瑀

   Task 3：0516209 呂淇安

   report：陳廷達、許芳瑀、呂淇安