

# CV 2021 HW2 Report

Group 24

0610122 李思賢, 309553049 陳怡安

## Introduction

This assignment contains three parts. First, we are asked to implement a work that can produce hybrid images. Hybrid images are images that seem different when people look at them from different distances. For example, when you look at the image closely, it may look like a cat. However, when you take a step back, suddenly it changes to a man.

In the second part of our work, we tried to implement the image pyramids. Image pyramid is a collection of representations of an image. In other words, an image in different scales with different features.

Last, we take digitized Prokudin-Gorskii glass plate images as inputs. Each of these images is composed of three duplicated photos, filtered by blue, green, red glass plate respectively. Our goal is to find out the displacement between the three channels of each image, align them, and reconstruct the full-color photo.

## Implementation

### 1. Hybrid images

A hybrid image is generated by the low spatial frequencies of one image and the high spatial frequencies of another image. In this task, we do the convolution part in the frequency domain.

#### a. Generate filters

We implement two kinds of pass filters, Ideal filter and Gaussian filter.

Gaussian low-pass filter is calculated according to the formula:

$$H(u, v) = 1 - e^{-D^2(u, v) / 2D_0^2}$$

Ideal low-pass filter is calculated by the conditional formula:

$$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \leq D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases}$$

where  $D_0$  is a cutoff frequency,  $D(u, v) = (u^2 + v^2)^{1/2}$ .

After getting a low-pass filter, we get the high-pass filter by  $1 - \text{low-pass filter}$ .

#### b. Filter input images

Multiply two input images by  $(-1)^{x+y}$  and compute Fourier transformation of input images by `numpy.fft.fft2`. Then, multiply the Fourier transformation images by low-pass filter and high-pass filter, respectively, compute the inverse Fourier transformation by `numpy` and multiply two images by  $(-1)^{x+y}$ .

#### c. Hybrid images

Finally, add two images together and clip overflow values to get the result.

## 2. Image pyramid

We implement image pyramids in spatial domain and frequency domain by recursion.

### a. Smoothing the image by 5x5 Gaussian filter

#### **-Spatial domain**

We scratch the convolution function by ourselves and use padding to keep the image at its original size.

#### **-Frequency domain**

The filter size must be the same as the input image, so we pad the 5x5 Gaussian filter with 0 to the image size. Next, compute Fourier transformation of input image and filter, and multiply them. Then, we compute the inverse Fourier transformation and use `np.fft.fftshift` to center the transform.

### b. Create pyramids

In the previous step, we can get the Gaussian image and then get the Laplacian image from the difference between input image and Gaussian image. The difference must belong to  $[-255, 255]$ , so we cannot show the image directly. Therefore, we add 255 to the difference and divide them by 2 to show the image. For showing magnitude spectrums, we convert color images to grayscale. Then use `numpy.abs`, `numpy.log`, and normalization to show the Fourier transformation images.

After that, we input the new image by subsampling the Gaussian image to create the next layer.

## 3. Colorizing the Russian Empire

### a. Preparing the image

To obtain the three channels of a picture, we divide the input image into three parts equally by its heights. The top one is the blue channel, following is the green channel, and the bottom one is the red channel.

### b. Alignment

#### **-Simple align**

As the assignment mentioned, we fix the blue channel as the base and try to find out the displacement of the other two ones respectively. To achieve this, we slide the green or red channel vertically and horizontally within a  $[-15, 15]$  window using `np.roll`. Then, we calculate the similarity between the displaced channel and the blue channel. The matrix we use is SSD(Sum of Squared Differences), which simply computes the square error of every pixel and sums them up. After the exhausting searching, we take the displacement associated with the smallest SSD as the result.

#### **-Pyramid align**

Although the simple align method is sufficient for most small images, it not only takes too much time for large images, but also produces terrible results. Thus, we utilize the image pyramid to improve our work. We filter the image using a gaussian filter, then subsample it to half of its original size. We downsize the image repeatedly until it is smaller than 400x400 pixels, then we

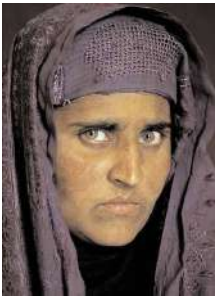















do the simple alignment on this coarsest image. The result is passed to the upper layer as an offset of doing alignment on the second coarsest image. Furthermore, the window size is reduced to  $[-3,3]$  because we now have the prediction from the coarser layer. We recursively do the alignment until doing on the finest image.

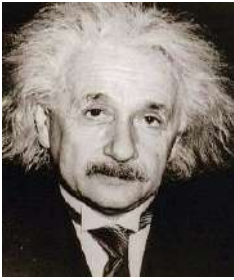















c. Showing the result

Finally, we align the red and green channel to the blue one using the displacement we found out previously. We use `np.roll` to slide the channels, then use `np.dstack` to combine them into a RGB image.

## Experimental results

### 1. Hybrid images

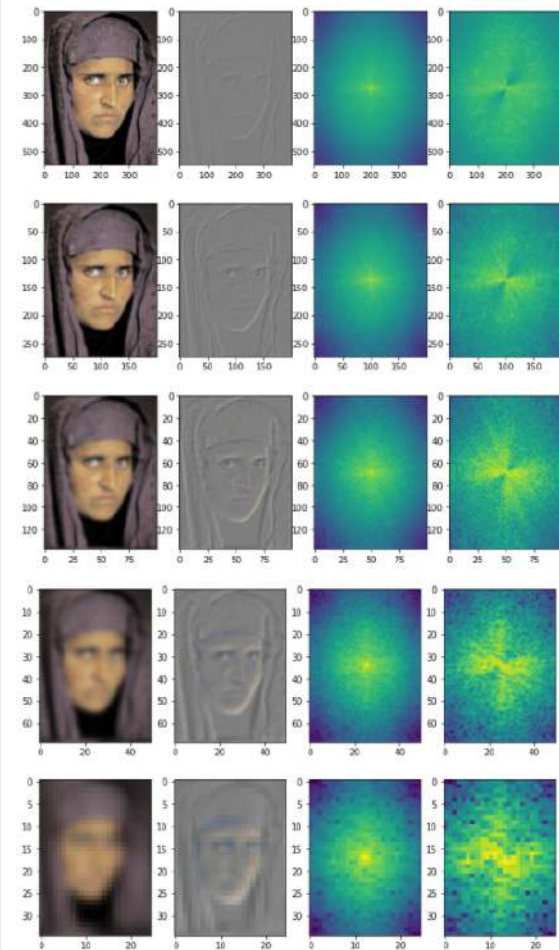
	Low	Hight	Ideal	Gaussian	$D_0$
0					5
1					5
2					10
3					3

4					7
5					4
6					5
7					4

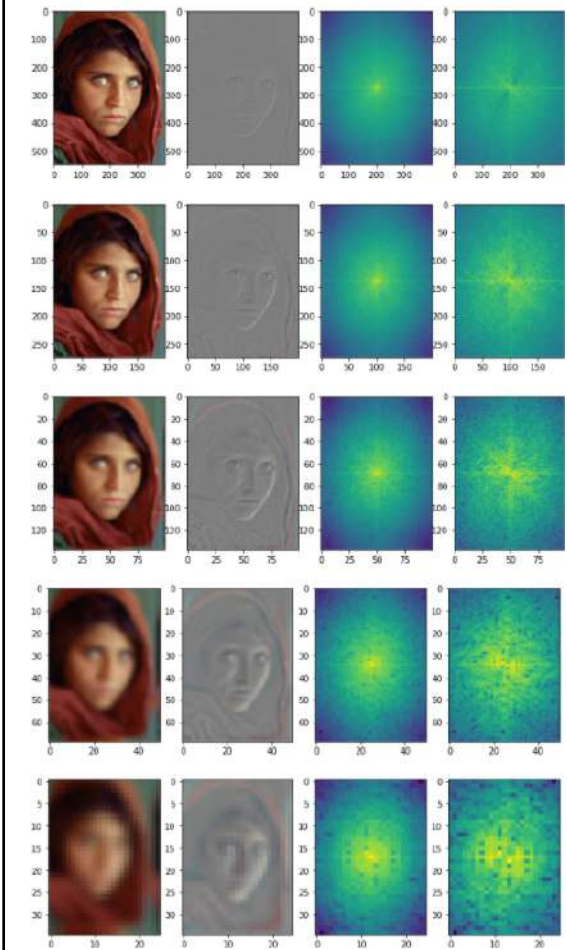


## 2. Image pyramid

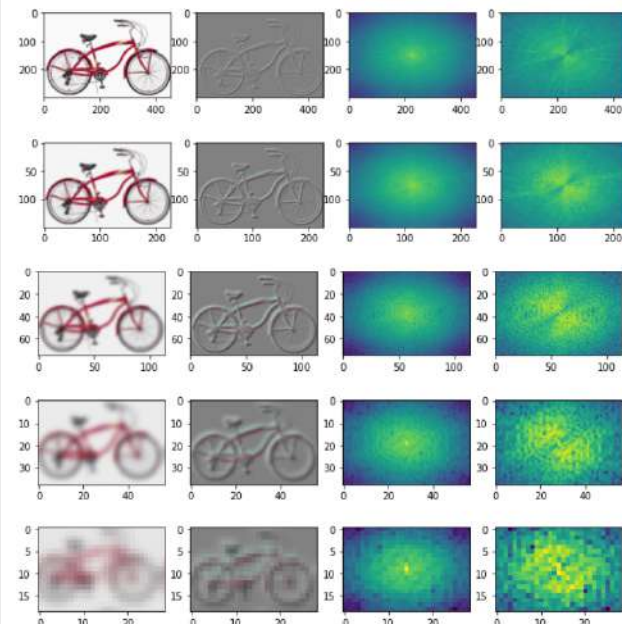
0\_Afghan\_girl\_after



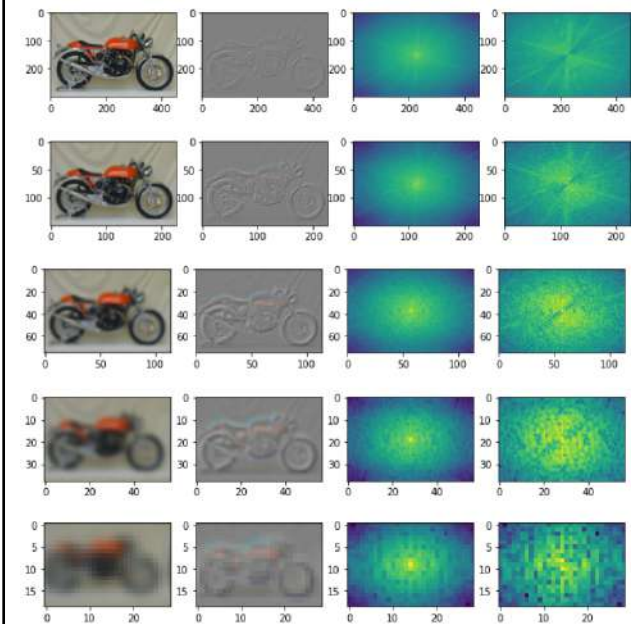
0\_Afghan\_girl\_before



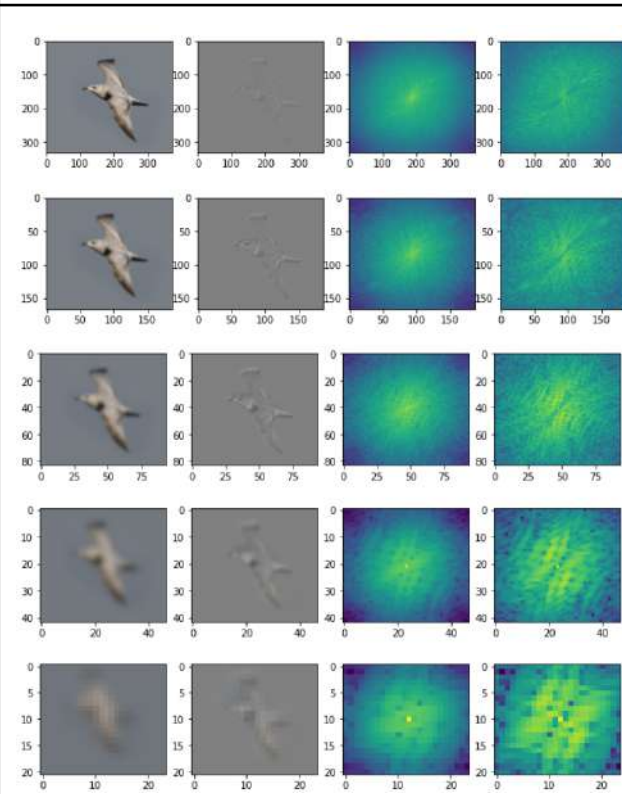
1\_bicycle



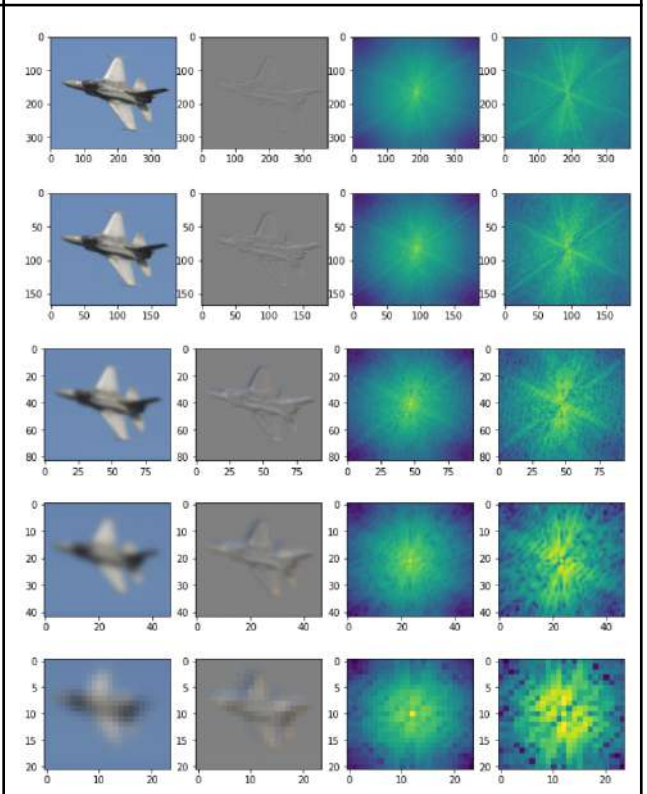
1\_motorcycle



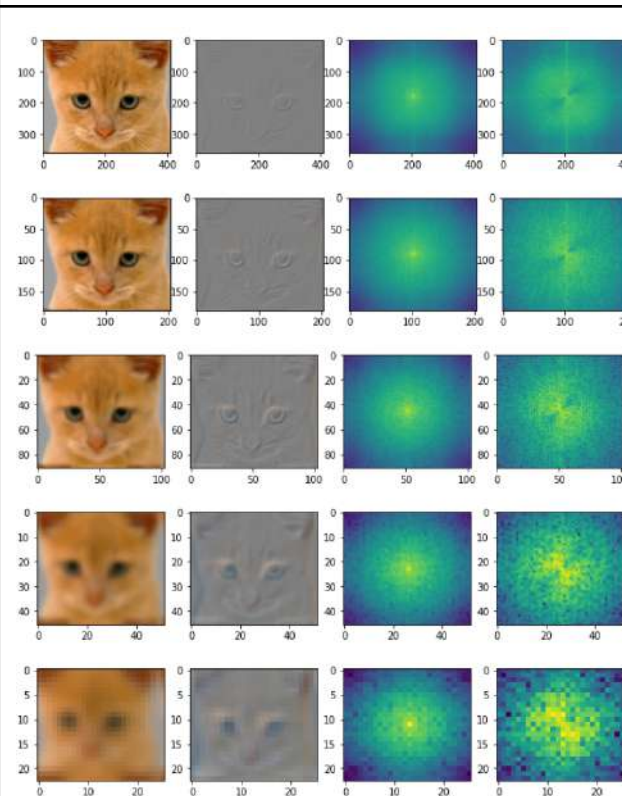
2\_bird



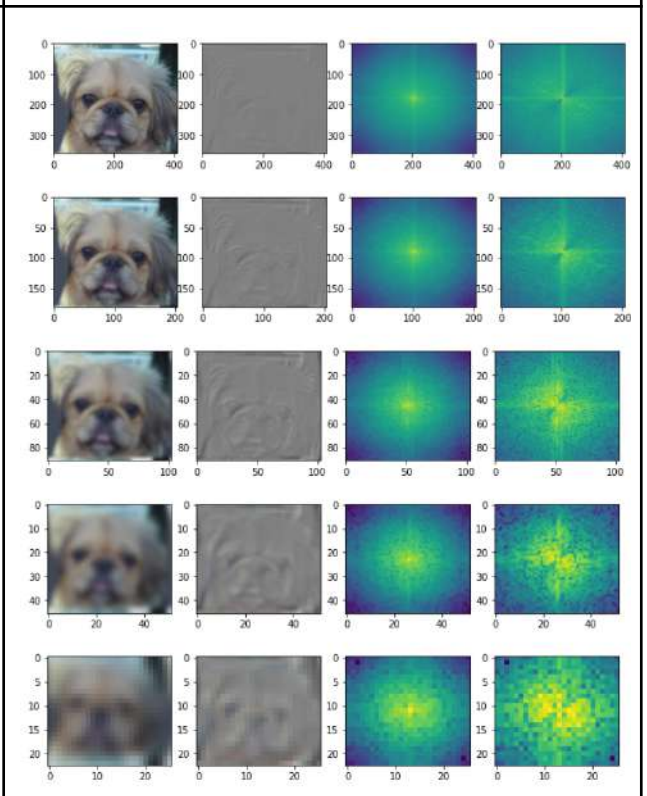
2\_plane



3\_cat

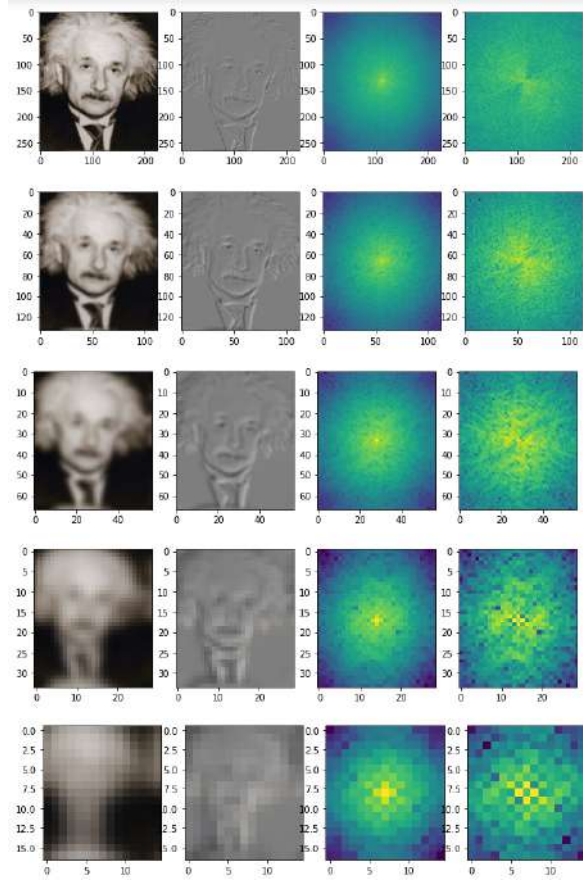


3\_dog

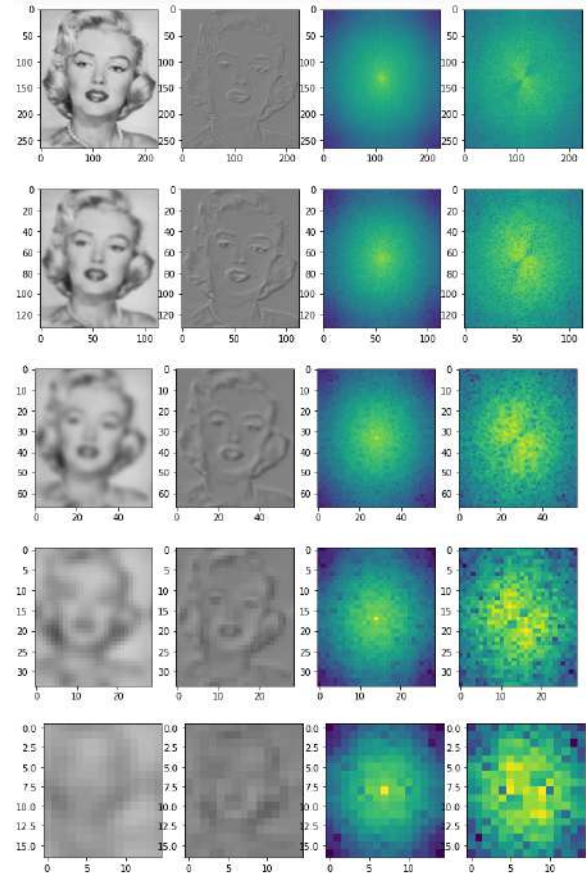




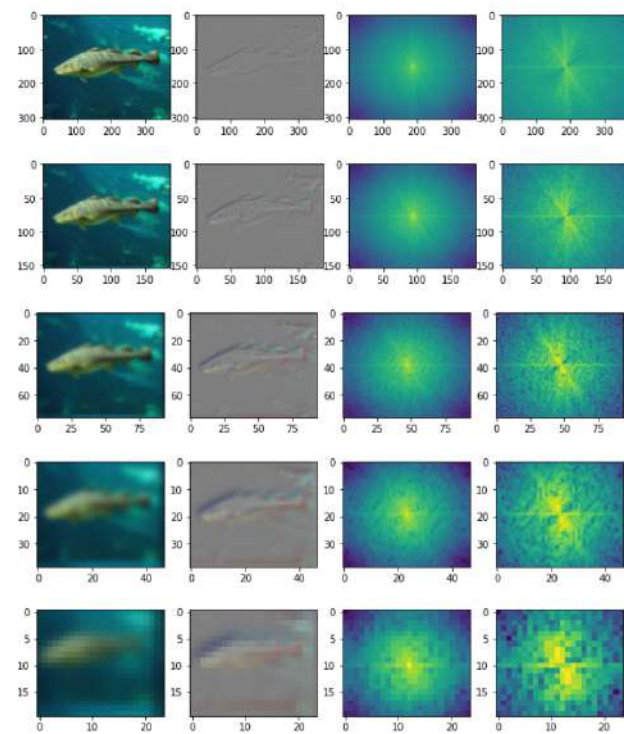
4\_einstein



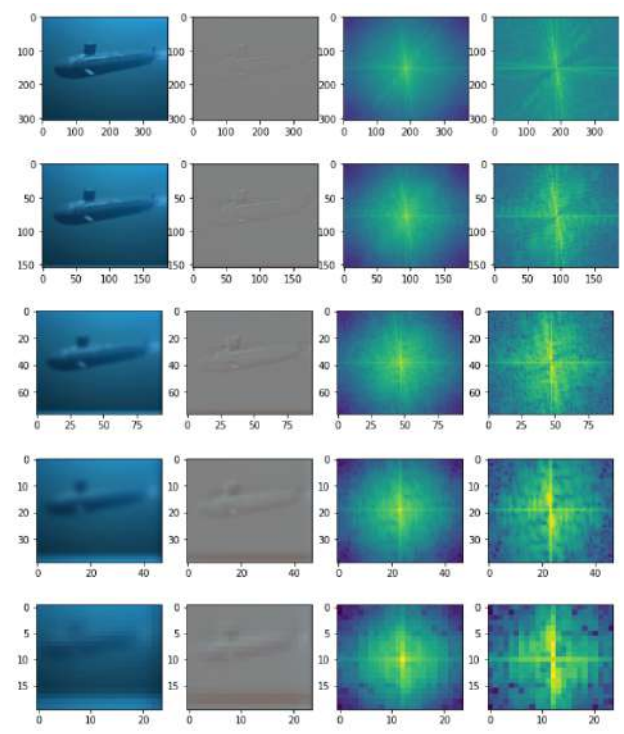
4\_marilyn



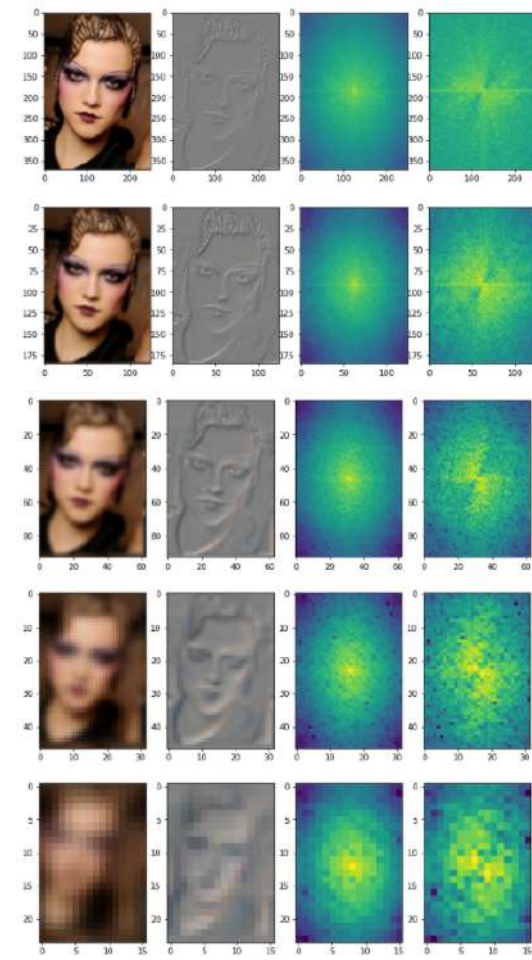
5\_fish



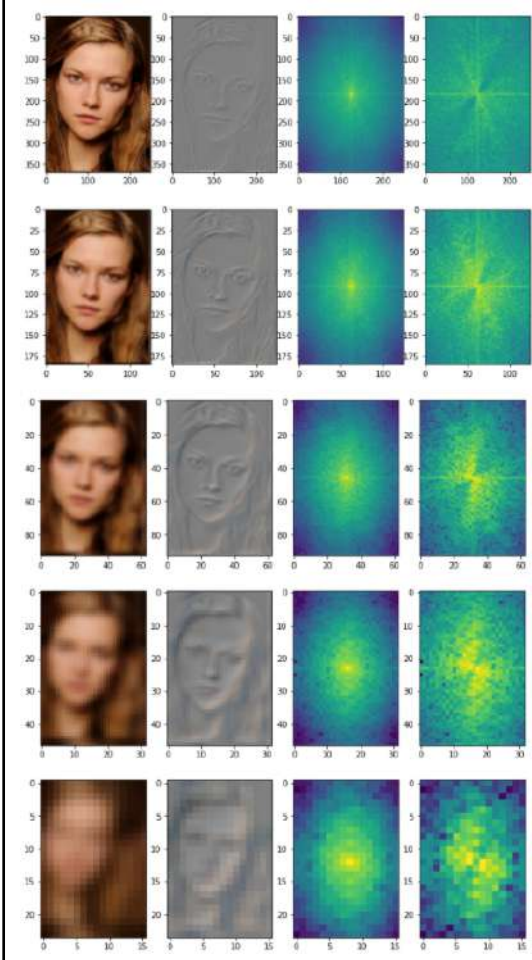
5\_submarine



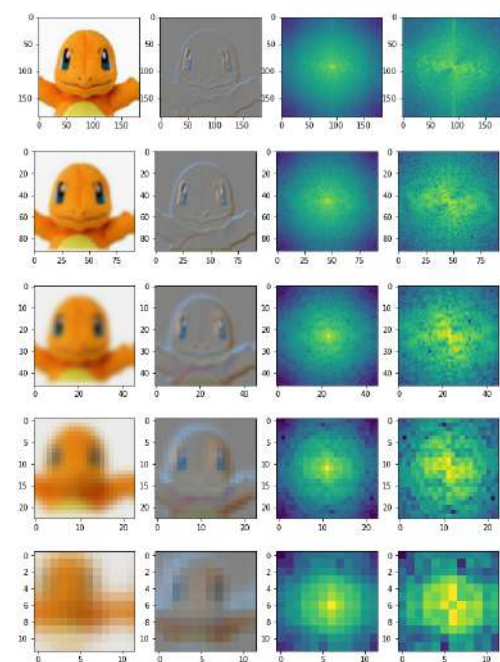
6\_makeup\_after



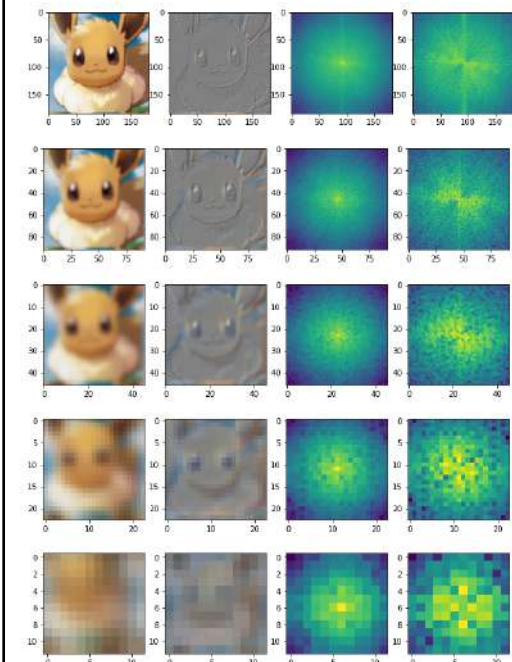
6\_makeup\_before



our image

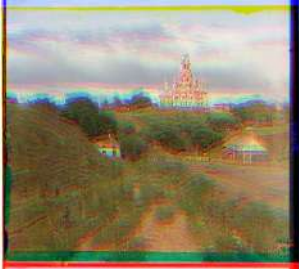










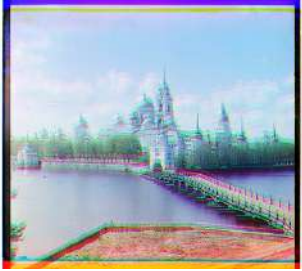
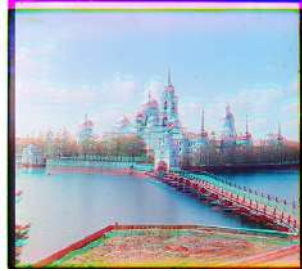
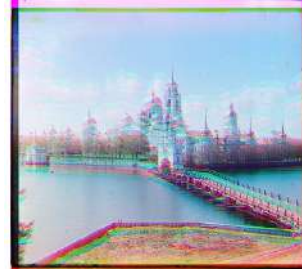






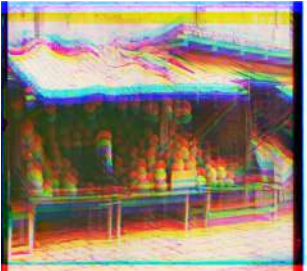


our image




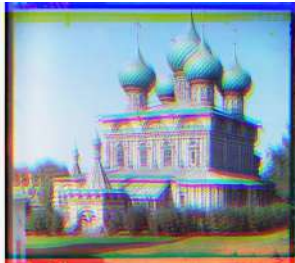
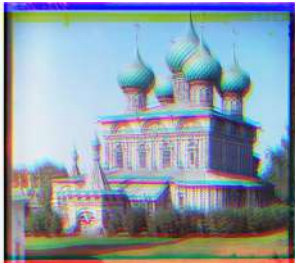


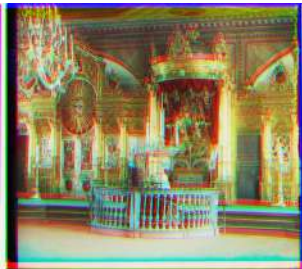














### 3. Colorizing the Russian Empire

	Original picture	Simple align	Pyramid align
cathedral.jpg		 G to B: [1, -1] R to B: [7, -1] time: 0.2813122272491455	 G to B: [1, -1] R to B: [7, -1] time: 0.4370107650756836
tobolsk.jpg		 G to B: [3, 2] R to B: [6, 3] time: 0.28705644607543945	 G to B: [3, 2] R to B: [6, 3] time: 0.42159104347229004
nativity.jpg		 G to B: [3, 1] R to B: [8, 0] time: 0.3023056983947754	 G to B: [3, 1] R to B: [7, 1] time: 0.41492319107055664

monastery.jp g		 G to B: [-3, 1] R to B: [11, 1] time: 0.27700304985046387	 G to B: [-6, 0] R to B: [9, 1] time: 0.4213285446166992
lady.tif		 G to B: [15, -9] R to B: [15, -8] time: 36.09106206893921	 G to B: [57, -6] R to B: [123, -17] time: 23.162781476974487
train.tif		 G to B: [7, -8] R to B: [15, -6] time: 36.607020139694214	 G to B: [111, -7] R to B: [107, 1] time: 21.874396800994873
melons.tif		 G to B: [14, -11]	 G to B: [83, 4]

		R to B: [15, -15] time: 36.922974586486816	R to B: [177, 8] time: 21.736844062805176
village.tif		 G to B: [15, -7] R to B: [15, 9] time: 37.5227587223053	 G to B: [144, -7] R to B: [118, -15] time: 23.453142404556274
onion_church.tif		 G to B: [14, -3] R to B: [15, -3] time: 36.926156997680664	 G to B: [52, 22] R to B: [108, 0] time: 24.91279101371765
icon.tif		 G to B: [-7, -5] R to B: [15, -6] time: 35.810056924819946	 G to B: [42, 16] R to B: [89, 22] time: 23.25602960586548



three_generations.tif			
		G to B: [15, -1] R to B: [15, 4] time: 35.16501069068909	G to B: [52, 5] R to B: [108, 7] time: 23.481301069259644
emir.tif			
		G to B: [15, 14] R to B: [15, 15] time: 34.999653816223145	G to B: [-3, 7] R to B: [107, 17] time: 23.901625871658325
workshop.tif			
		G to B: [15, 3] R to B: [15, -9] time: 35.4511399269104	G to B: [53, -5] R to B: [69, -16] time: 22.777587890625

## Discussion

### 1. Hybrid images

When trying to hybrid the given images, we have a problem about different size of a pair of images, so we use nearest-neighbor interpolation to keep them in the same size.

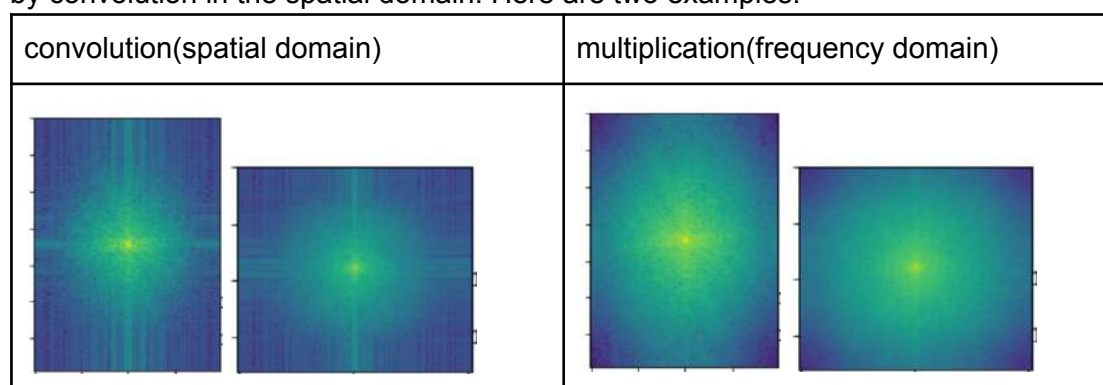
Comparing the results by using Ideal filter to Gaussian filter, we found that Ideal filter caused ringing effect, but Gaussian filter didn't. The ringing effect is particularly evident in the second image and the fifth image.

Regarding how we choose a cutoff frequency, we hope that as the distance increases, the hybrid image will be seen to change from right image to left image. Therefore, the cutoff frequency of every image would be different, and adjusted according to the effect we want.

## 2. Image pyramid

Comparing the spatial domain approach to frequency domain approach, the largest difference is the execution time. Doing convolution in the frequency domain is much faster than in the spatial domain.

Besides, we found that the magnitude spectrums of Gaussian images are different. The spectrums by multiplication in the frequency domain are more smooth than that by convolution in the spatial domain. Here are two examples:



There are many vertical frequencies and horizontal frequencies by convolution, which means there are more horizontal edges and vertical edges in the corresponding image by using convolution than that by using multiplication.

## 3. Colorizing the Russian Empire

The simple align method produces quite good results in most of the small images(.jpg images), but it can not align the large images(.tif images) well. We think the primary reason is that **our window size [-15, 15] is too small considering the large image size in the latter cases**. The .tif images usually have size larger than 3000x3000, so the search box only covers only a little fraction of the images' heights and widths. The displacements we found using pyramid align prove our claim. It takes around 100 pixels of displacement to achieve a good alignment. To solve the problem, we can expand our searching range. However, it will largely **increase the searching time**. By using pyramid alignment, we can both increase the searching range and reduce runtime. It takes less than 30 seconds to align an image with our method.

There are still some cases where the aligned images seem to be blurred. We tried to use another channel instead of the blue one as the base. Below are the results. On the left-hand side are the images aligned to the blue channel, and the right-hand side are the images aligned to the green channel. It is obvious that fixing different channels does influence the outcome a lot.



## Conclusion

The effects of hybrid image are satisfactory, we can see the result like different images at different distances. We also compared the Gaussian filter to Ideal filter, and we found that Ideal filter could cause ringing effect.

In the image pyramid task, we implemented convolution in both spatial domain and frequency domain, and experienced the differences between two approaches.

The results show that the pyramid alignment can align the image well without taking much time. The method works for both small and large images. We also found that choosing the right image channel to be the base and align the others may has a big influence on the final result.

## Work assignment plan

task1&2: 陳怡安

task3: 李思賢