Homework 1

Wen-Jie Tseng (0556146)
Li-Yang Wang (0656619)
Yi-Chen Lee (0656611)
Computer Vision - 2018 Spring Semester

1 Hybrid Image

1.1 Introduction

A hybrid image is an image that can be perceived in different ways depending on the viewing distance. Take 1 as an example. The image seems to be a photo of Albert Einstein in close range; however, it changes into a photo of Marilyn Monroe while perceived from a longer distance. A common way to create a hybrid image is combining low-frequency part of one image with high-frequency part of another.

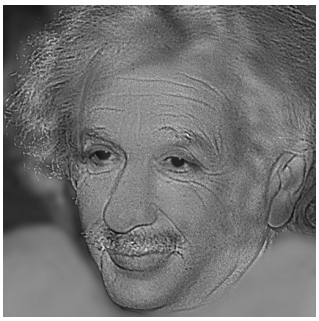

Figure 1: Photo of Marilyn Monroe

1.2 Implementation Details

Two images are given in this task. To create a hybrid image from the given data, we first create a Gaussian filtering mask in order to achieve the task of filtering out the low-frequency/ high-frequency parts. We then transform the images into frequency domain by using Fourier Transform. Finally, by multiplying the transformed images with the Gaussian Filter and apply inverse Fourier Transform, we obtain a hybrid image that is composed of given images.

1.2.1 Obtaining Gaussian Filter

To create a hybrid image, low-pass Gaussian filter and high-pass Gaussian filter are necessary. Their definition are as following equations:

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

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

Equation (1) refers to low-pass Gaussian filter, and equation (2) refers to high-pass Gaussian filter. According to these equations, we discovered a crucial component

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

Where $D^2(u,v)$ refers to the distance between point (u,v) and the origin, ${D_0}^2$ refers to a predefined value. To obtain the value of $D^2(u,v)$ for each (u,v), we first calculate the center of the given image, then derive the value of $H(u,v)$ at the point by using equation (1) or equation (2). By collecting all the values, we obtain the required low-pass and high-pass Gaussian filtering masks.

1.2.2 Fast Fourier Transform

To transform the given images into their frequency domain, we simply utilize the fast Fourier transform function built in the numpy library. Notice that after applying Fourier transform, a shifting process is needed to move all the DC terms(0-frequency terms) to the center for the following operations.

1.2.3 Image Generation

After the Fourier transformation, we multiply the transformed images with low-pass/ high-pass masks. Next, we apply inverse shifting and inverse Fourier transform to obtain two filtered image. Finally, we sum the images to get the final result.

Figure 2: Photo of Marilyn Monroe

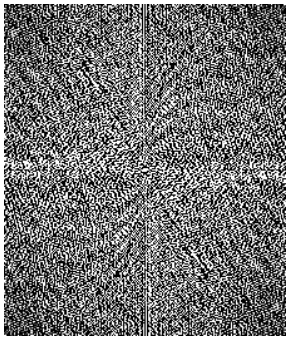

Figure 3: Fourier transformed result

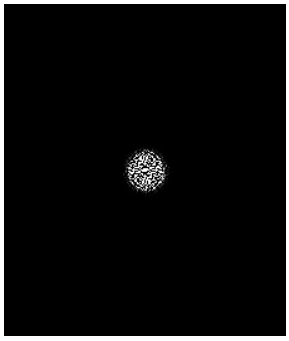

Figure 4: Filtered result

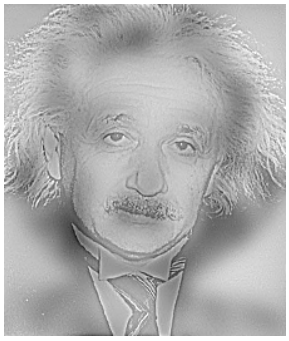

Figure 5: Hybrid image

1.3 Results

Figure 3 is the spectrum of figure 2, and figure 4 is the filtered result. Finally, we display the final hybrid image as figure 5.

2 Image Pyramid

2.1 Introduction

Pyramid, or pyramid representation, is a type of multi-scale signal representation developed by the computer vision, image processing and signal processing communities, in which a signal or an image is subject to repeated smoothing and subsampling.

2.1.1 Gaussian Pyramid

In a Gaussian pyramid, subsequent images are weighted down using a Gaussian average (Gaussian blur) and scaled down. Each pixel containing a local average that corresponds to a pixel neighborhood on a lower level of the pyramid.

2.1.2 Fast Fourier Transform (FFT)

Fourier Transform is used to analyze the frequency characteristics of various filters. For images, 2D Discrete Fourier Transform (DFT) is used to find the frequency domain. A fast algorithm called Fast Fourier Transform (FFT) is used for calculation of DFT.

2.2 Implementation Details

2.2.1 Gaussian Pyramid

Use the OpenCV functions pyrUp and pyrDown to downsample or upsample a given image.

Figure 6: Gaussian pyramid

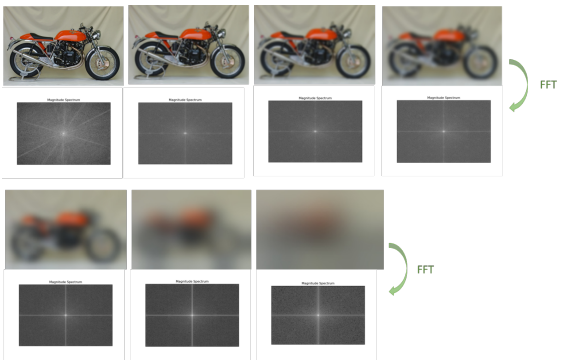

Figure 7: Fast Fourier Transform

2.2.2 Fast Fourier Transform (FFT)

First, find Fourier Transform using Numpy. Numpy has an FFT package to do this. np.fft.fft2() provides us the frequency transform which will be a complex array. Then, input the magnitude spectrum formula and plot.magnitude spectrum.

2.3 Results

The results were shown in Figure 6 and 7.

3 Colorizing the Russian Empire

3.1 Introduction

The goal of this assignment is using image processing techniques, automatically produce a color image from the digitized Prokudin-Gorskii glass plate images with as few visual artifacts as possible.

3.2 Implementation Details

Two formats of images were given in this task. In the images of jpg format, one can simply apply naive method to combine three channels (RGB) into one colorized image (as shown in Figure). However, in the images of tif format, naive method cannot handle such high resolution image. We applied Sobel operator and multiscale image pyramid to speedup.

3.2.1 Cropping

To remove the black frame of original image, we cropped 10% of the pixels from each side after splitting glass plate into three channels.

3.2.2 Alignment

We searched over a window of possible displacements (e.g. [-15, 15] pixels), score each one using the Sum of Squared Differences (SSD), find the offset of G and R channels with respect to B. The function, numpy.roll, provides an effortless method for offsetting images. As shown in Figure 8, our alignment improve the quality of figure.

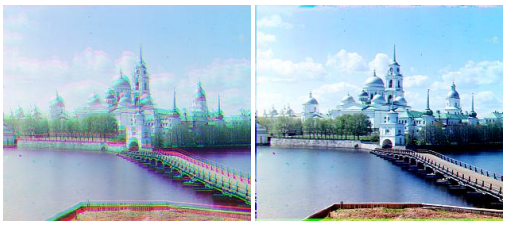

Figure 8: The glass plate monastery.jpg before and after alignment. G (-3, 2), R (3, 2) for the displacement to B channel.

3.2.3 Auto-Contrasting

A contrast factor 1.5 and compared to median of image was applied to all output images to enhance quality of image.

3.2.4 Coarse-to-Fine Pyramid Speedup

We applied Sobel operator to the image from x-axis and y-axis of each channel. The function convolve2d in scipy.signal helps a lot. After convolving through the Sobel operator, the edge information will remain. Thus we computed the displacement through SSD in each scale of pyramid image. At last, reconstruct the image with the displacement accumulated in the previous procedure.

3.3 Results

The jpg images take about 1 seconds to complete colorizing process. Without speedup, naive method takes about 3 minutes to colorizing one tif image but property was poor. With our sobel operator and pyramid speedup, the execution time takes about 20 seconds to complete (This value may varied depends on image resolution and cropping margin). The result of melon.tif was shown in Figure 9. More colorized images were in our zip file.

Figure 9: The colorizing result of melons.tif, using pyramid speedup. G (79, 9), R (175, 13) for the displacement to B channel.

4 Conclusion

We implemented three tasks

1) Compute hybrid image with Gaussian filter

2) Generate image pyramid

3) Colorize glass plate image

For the source code and execution command, see our GitHub repository: `https://github.com/wenjietseng/cv18-nctu` or Code Usage section.

5 Code Usage

Language: Python 3

Library: cv2, numpy, skimage

Hybrid Image: python hybrid-image.py /path-to-img1/img1 /path-to-img2/img2

Image Pyramid: python pyramid-FFT.py /path-to-img/img

Colorizing the Russian Empire: python colorizing-naive.py /path-to-img/jpg-img
python colorizing-pyramid.py /path-to-img/tif-img

6 Work Assignment Plan

Each person was in charge of the code implementation and the report of one task.

Task 1: Li-Yang Wang

Task 2: Yi-Chen Lee

Task 3: Wen-Jie Tseng

7 References

1. Hybrid Image Wikipedia
2. Hybrid Image Code Reference
3. Image Pyramid, Python OpenCV Tutorial
4. Fast Fourier Transform, Python OpenCV Tutorial
5. OpenCV and FFT, CSDN
6. CS194, Image Manipulation and Computational Photography
7. Colorizing Code Reference