CV HW2

group 18

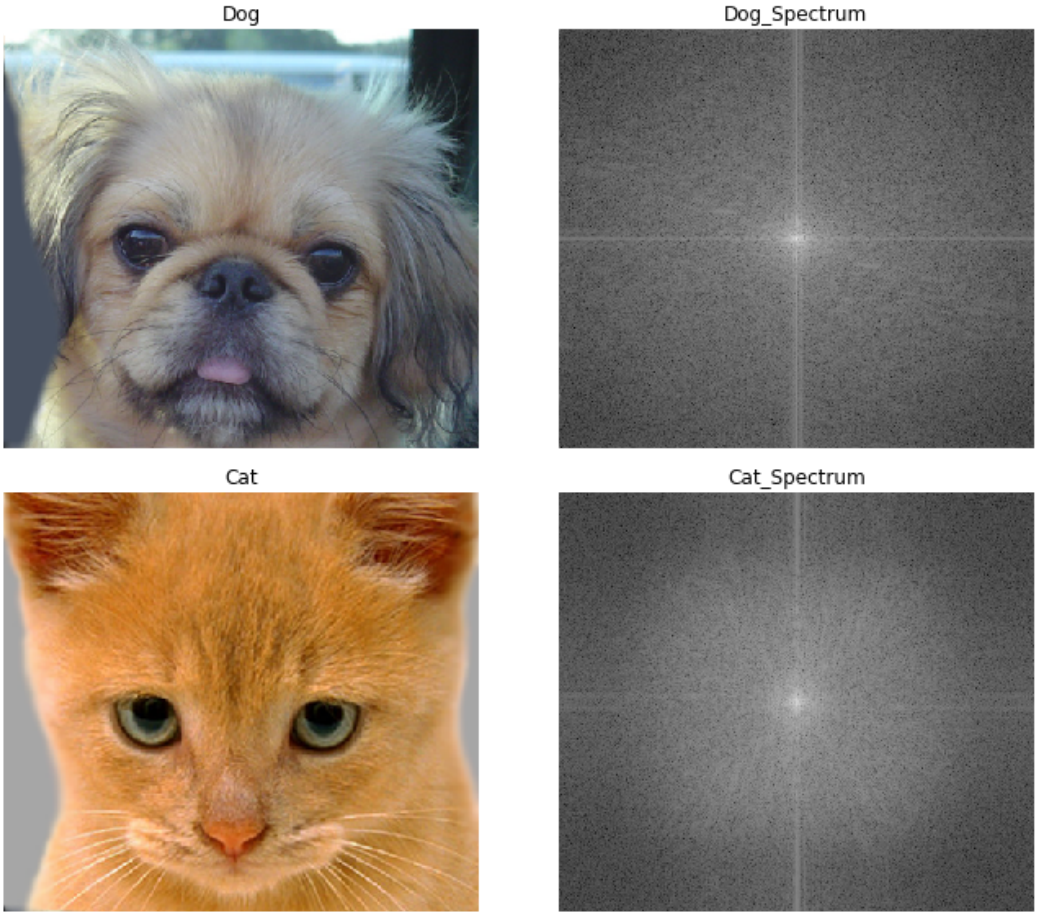
Task 1 (Hybrid Image)

Introduction

The “hybrid images” technique[1] is to produce static images with two interpretations, which change as a function of viewing distance. Hybrid images are based on the multiscale processing of images by the human visual system and are motivated by masking studies in visual perception. These images can be used to create compelling displays in which the image appears to change as the viewing distance changes.

Implementation Procedure

1. **Image spectrum visualization**
   Firstly, using numpy 2-dimensional discrete Fourier Transform(*fft2*) on the input image. Secondly, shifting the zero-frequency component to the center of the spectrum(*fftshift*). Last but not least, taking the absolute value and log to make pixels value between 0 to 255.

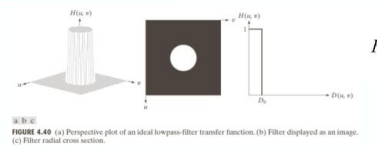


2. Implement Ideal low&high pass filter

The concept of cut off frequency operation is like erasing the circle. As result, we need to find the shortest image boundary. And, divided the boundary length to get the radius of the circle.
Next, we need to transform the pixel position into center of circle's offset.
For example [0,1,2,3] -> [-1.5, -0.5, 0.5, 1.5]

Afterward, follow the formula of ideal low pass filter, if the distance between current pixel and center of the circle is less than or equal to the cutofff frequency, then the filter value should be 1(True);otherwise, the filter value should be 0(False).
Additionally, the high pass filter just the opposite of low pass filter.
Finally, following the spectrum creation above with the ideal pass filter, and do the inverse discrete Fourier Transform(*ifft2*) to get the RGB image.
Note: the value would be overflow(uint8) when doing the operation

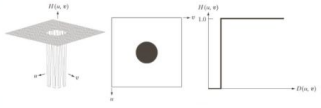
Ideal low pass filter



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

D_0: cutoff frequency
$D(u,v) = (u^2 + v^2)^{1/2}$

Ideal high pass filter

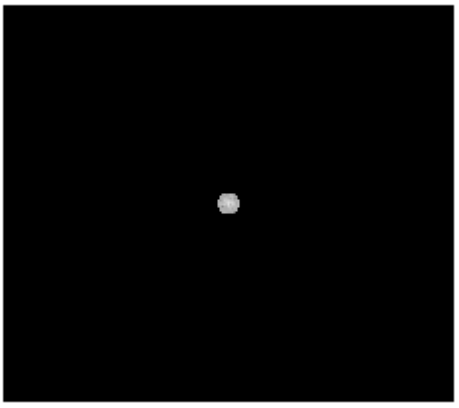


$$H(u,v)=\begin{cases}0 & if\ D(u,v) \le D_0\\ 1 & if\ D(u,v) > D_0\end{cases}$$

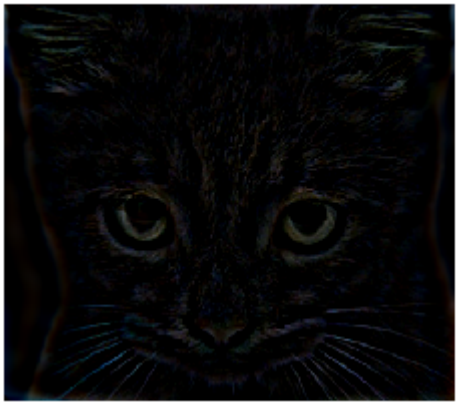
D_0: cutoff frequency
$D(u,v) = (u^2 + v^2)^{1/2}$

Ideal low pass filter (0.05)

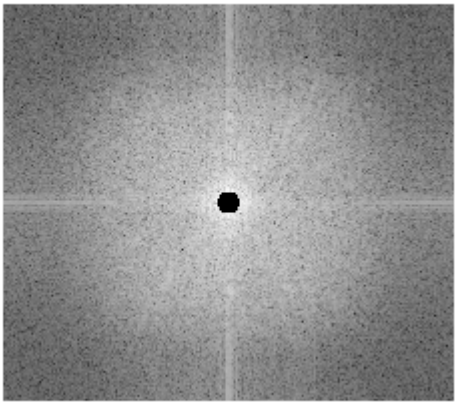
Ideal low pass filter Spectrum (0.05)



Ideal high pass filter (0.05)



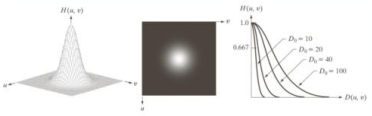
Ideal high pass filter Spectrum (0.05)



3. Implement Gaussian low&high pass filter

Similar to the Ideal pass filter. The only difference between Gaussian and Ideal pass filter is their formula. Gaussian filter as its name using Gaussian distribution to deal with the pixel value distribution. Same as Ideal pass filter, we can easily get high pass filter by using one minus low filter.

Gaussian low pass filter

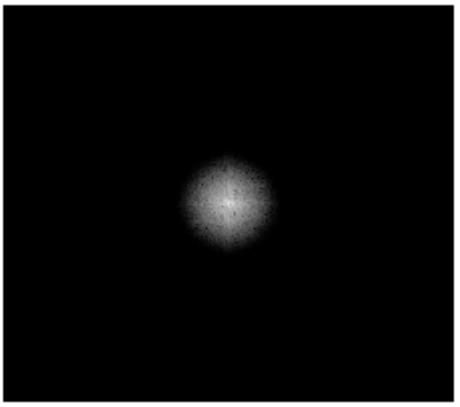


$$H(u,v) = e^{-D^2(u,v)/2D_0^2}$$

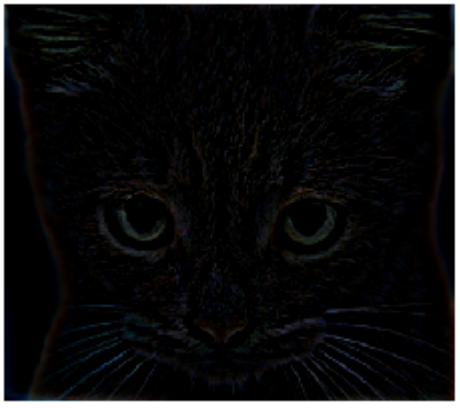
a b c
FIGURE 4.47 (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0.

Gaussian low pass filter (0.05)

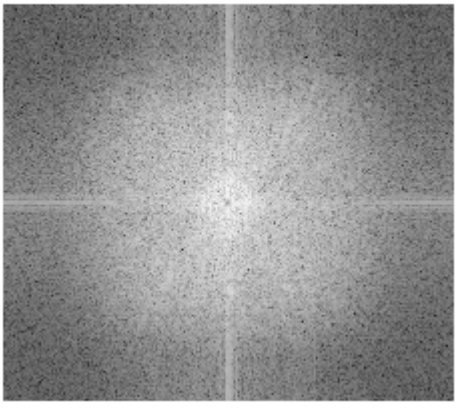
Gaussian low pass filter Spectrum (0.05)



Gaussian high pass filter (0.05)

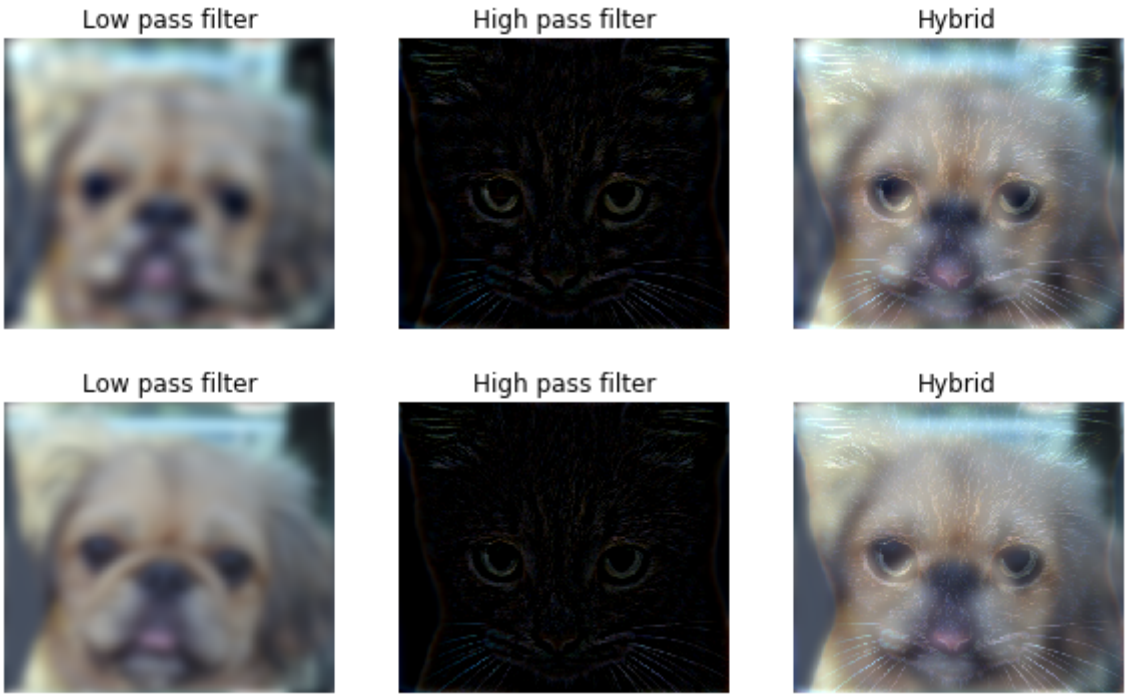


Gaussian high pass filter Spectrum (0.05)

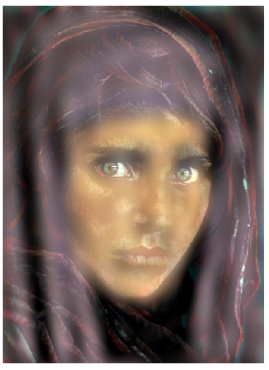
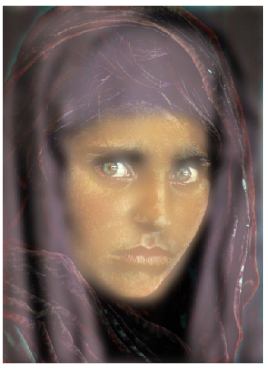
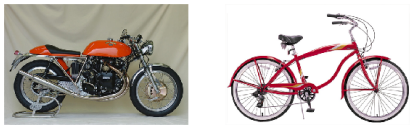


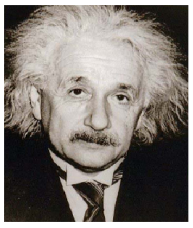
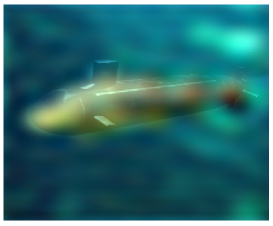
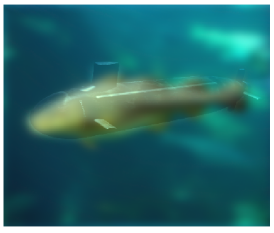
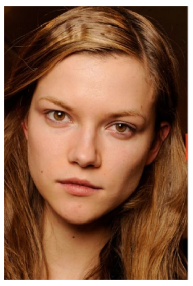
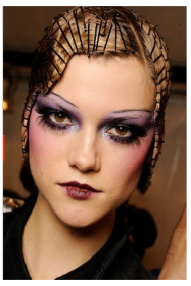
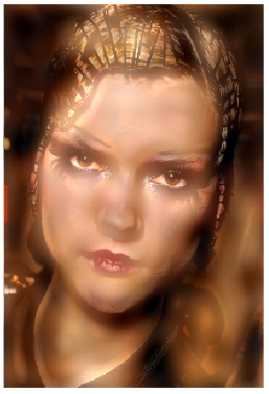
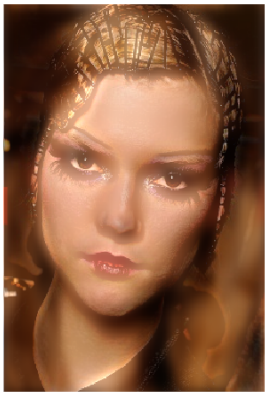
4. Combine low and high passed images

After filtering the images, we get low frequency image and high frequency image individually. The next step is to add them together to get hybrid image.
Note: the value would be overflow(uint8) when doing the operation.



Experimental Result

Original	Ideal	Gaussian	Cutoff Ratio (D0)
			0.05 (10)
			0.1 (15)

				0.05 (8.275)
				0.05 (5.625)
				0.05 (7.675)
				0.07 (8.715)

Note: Cutoff frequency D0 equal to the shortest boundary of image divided by 2 then multiply cutoff ratio.

(D0 = min(img.shape) / 2 * cutoff ratio). The ratio is like the radius length ratio to keep.

Discussion

1. Different image size may result in some problem such as alignment or compute error

A. We can use padding technique or only process to the minimal rows and cols, which keep images same size.

2. Cutoff ratio choice

A. It's hard to define the best cutoff ratio, due to the individual human visual system. In our experiment, most of the cases are between 0.1 to 0.05.

3. Numpy 2-dimensional discrete Fourier Transform(*fft2*) only accept 1 channel(gray) images

A. Do the transform channel by channel (R, G, B)

4. After doing the computation on the inverse Fourier Transform, there are some pixels may be overflow(over 255 or below 0). This problem occurs when we initialize the numpy images container with 'uint8' type, which only can store values between 0 to 255. Overflow may cause some of the pixels present sky blue or pink, which are quite conspicuous.

A. Initailize the numpy container with 'float' type, and clip the value beyond 255 or below 0. And convert the container back to 'uint8' type when print out the image.

Conclusion

Firstly, we start from Fourier Transform to get the image spectrum. Secondly, we implement two filter (Ideal and Gaussian) according to the formula and found the relation between low pass and high pass filters. Finally, we combine two output images and got the hybrid image. After this homework, we actually learn how to convert image to spectrum by Fourier Transform, and how cutoff operation works.

Additional references

[1] Aude Oliva, Antonio Torralba, and Philippe G. Schyns. "Hybrid Images". ACM Trans. Graph. 25, 3 (July 2006), 527–532.
[2] How to combine the phase of one image and magnitude of different image into 1 image by using python
[3] Dr. Chuan-Yu Chang, Chapter 4 Image Enhancement in the Frequency Domain

Task 2 (Image Pyramid):

Introduction

In task 2, we implement the gaussian image pyramid and laplacian image pyramid. Addition to get the image pyramid, we also get the magnitude spectrum for each image size. Gaussian image pyramid can be got by repeated gaussian smoothing and subsampling. After we have gaussian pyramid, we use level n image to minus the upsampled next level image so that we can get the level n laplacian image.

Implementation Procedure

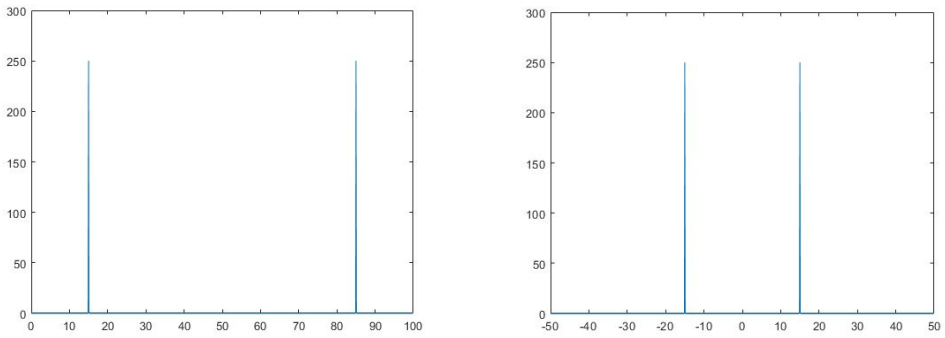
1. **Gaussian Blur**
   To reduce the time cost, we build the 5x5 gaussian filter to approximate the real gaussian.

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

2. **Subsampling**
   To do the subsampling, we use the idea:

$$p_k = \sum_{i \in win(k)} I_i / s^2$$

   In task 2, s = 2, because the size of images in each level is defined as 2^n x 2^n, we divide the image height and width by 2 in each recursion.
3. **Magnitude spectrum**
   We simply use the function of numpy - np.fft.fft2 to computes the *n*-dimensional discrete Fourier Transform by Fast Fourier Transform.
   Then we use np.fft.fftshift to translate the DC part to the center.

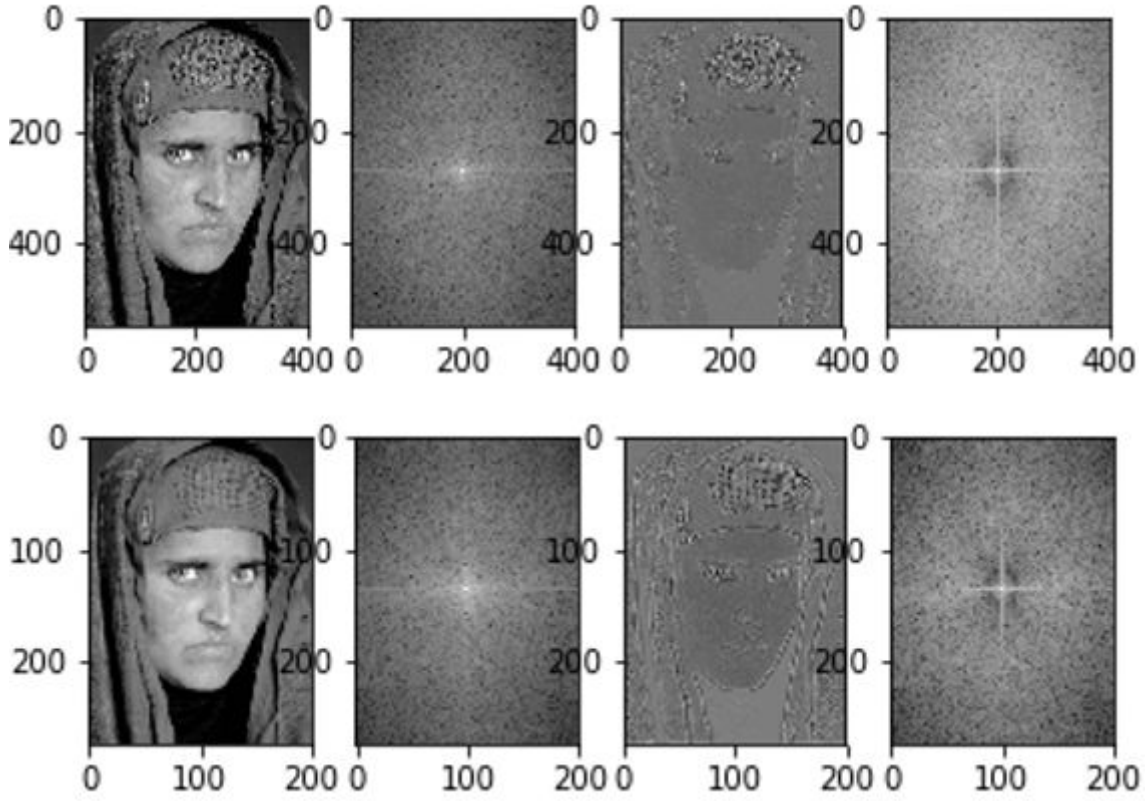


At last, 20*np.log(np.abs(fshift)) is used to reduce the scale of the data.

Experimental Result

Gaussian Pyramid	Gaussian Magnitude Spectrum	Laplacian Pyramid	Laplacian Magnitude Spectrum

Fig: 0_Afghan_girl_after



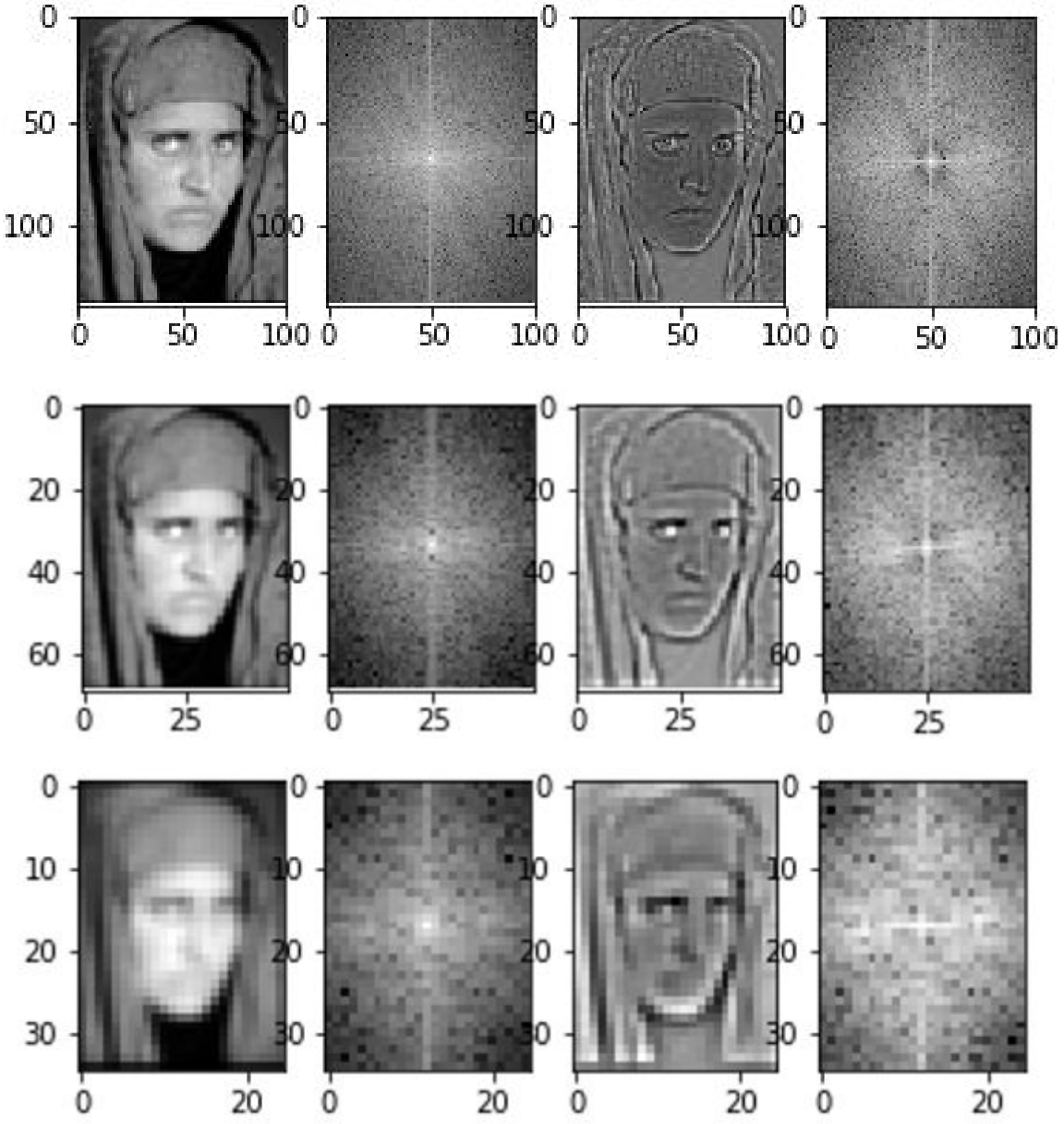
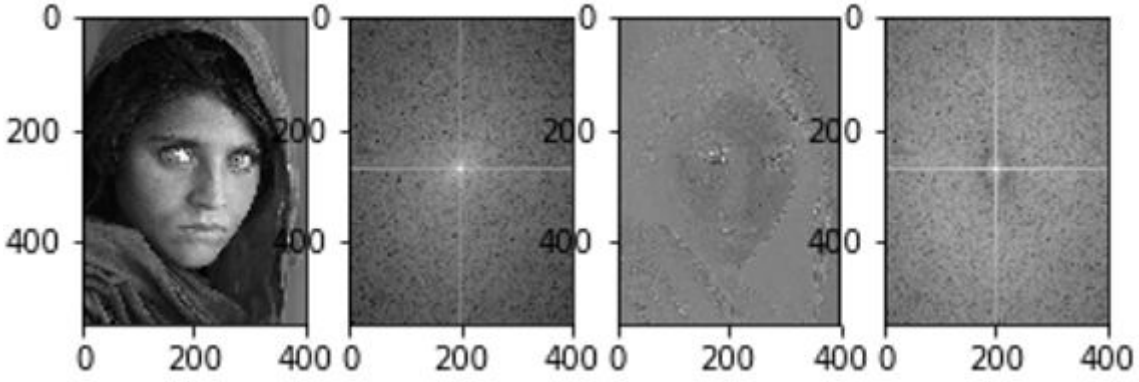


Fig: 0_Afghan_girl_before



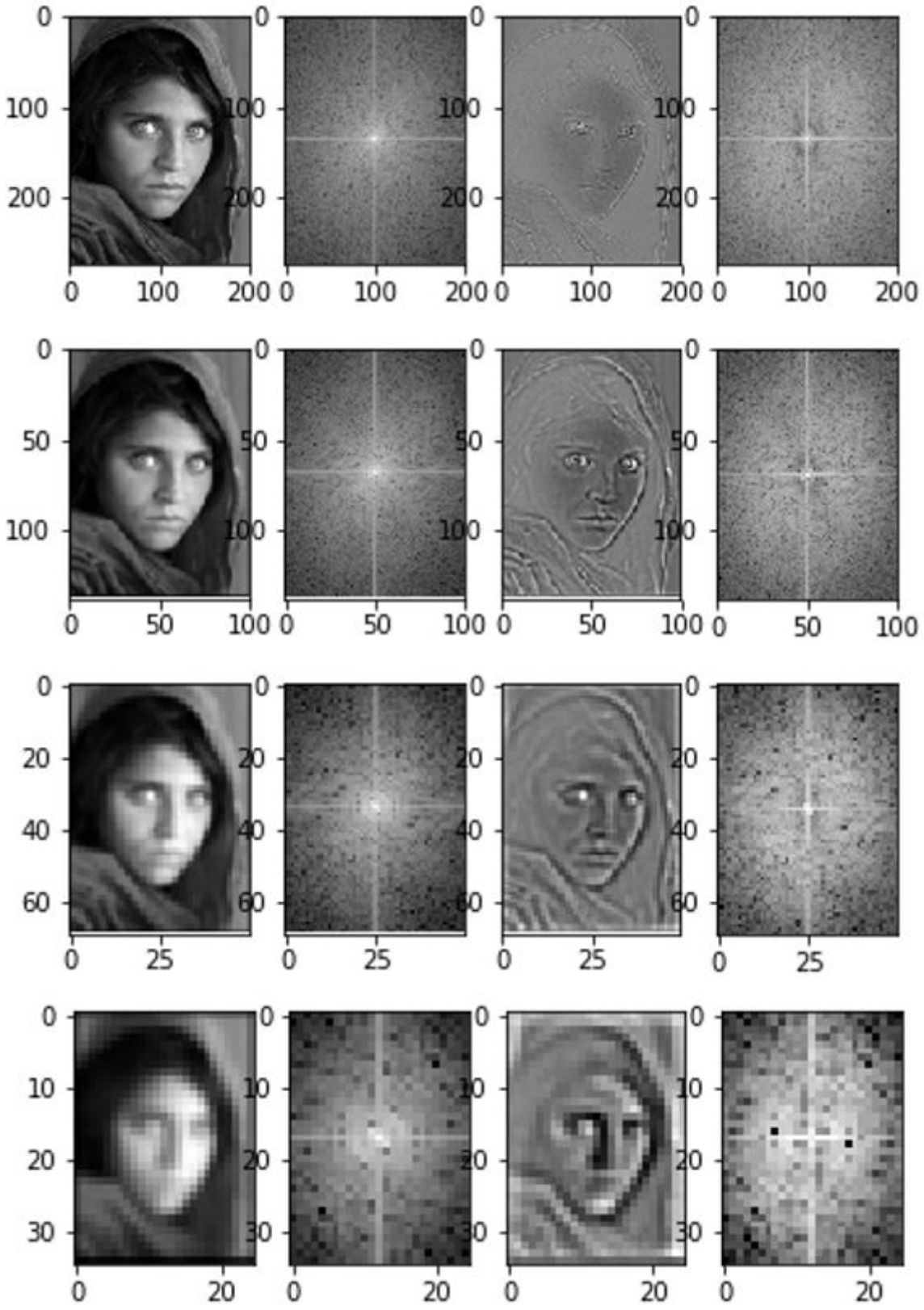


Fig: 4_einstein

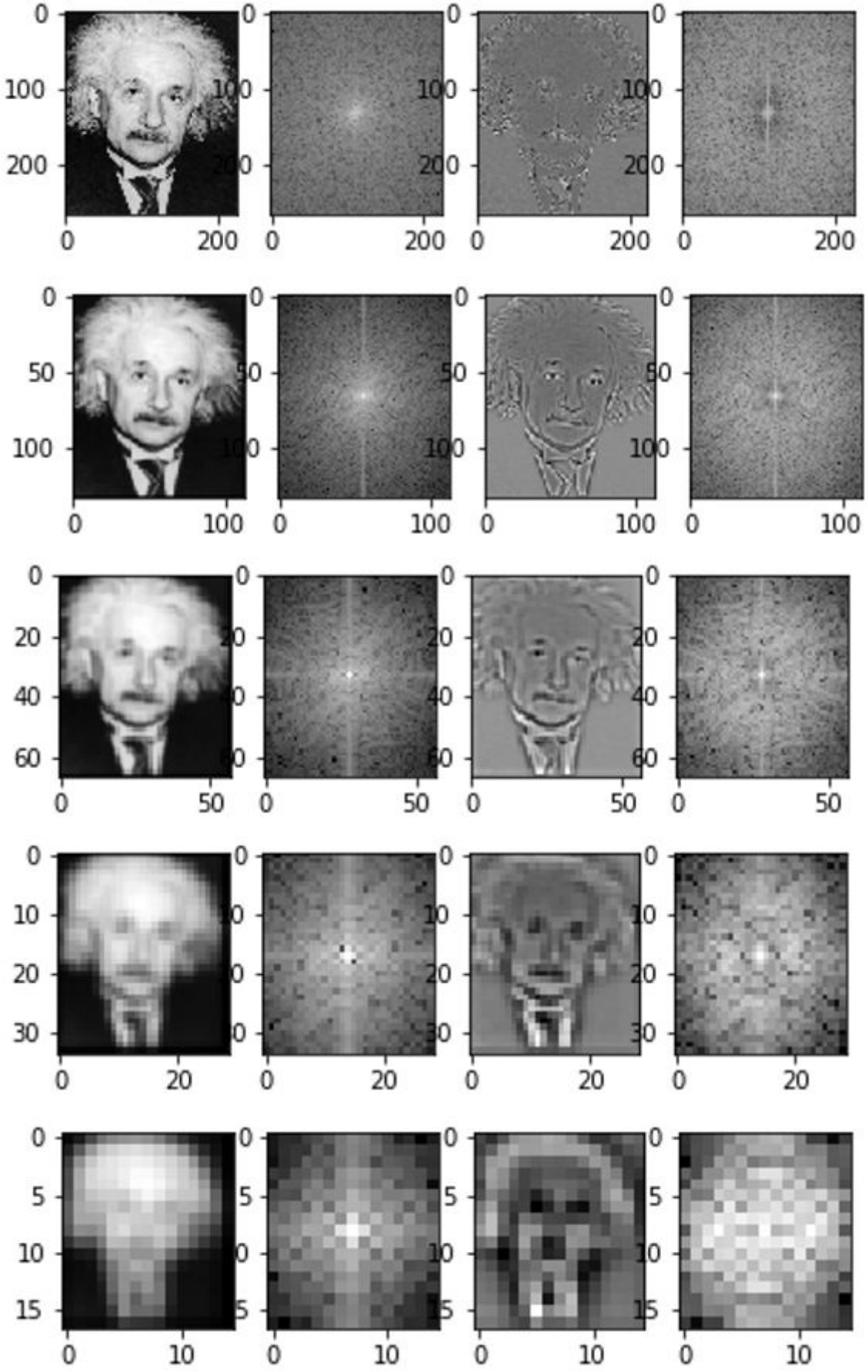


Fig: my own picture

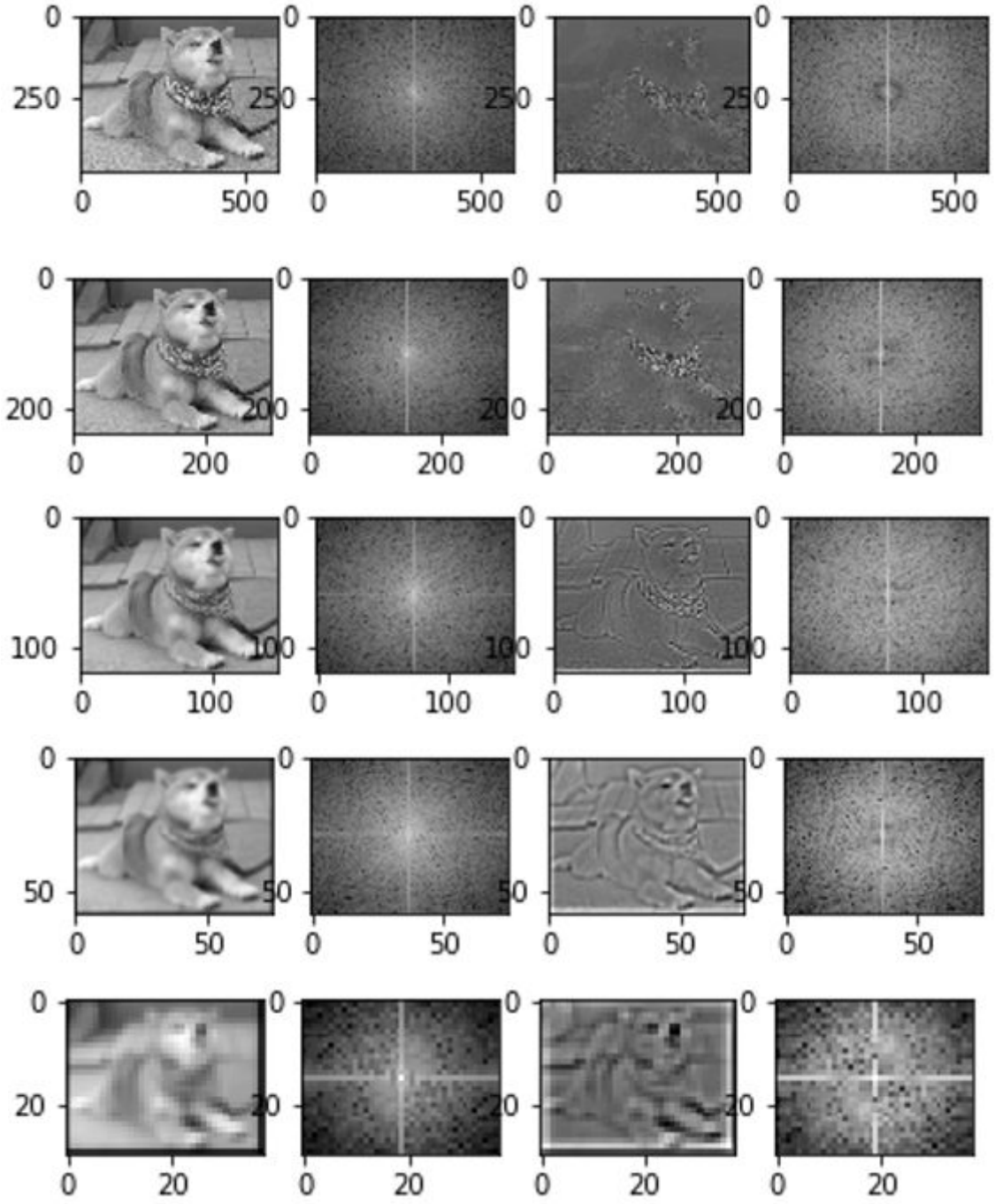
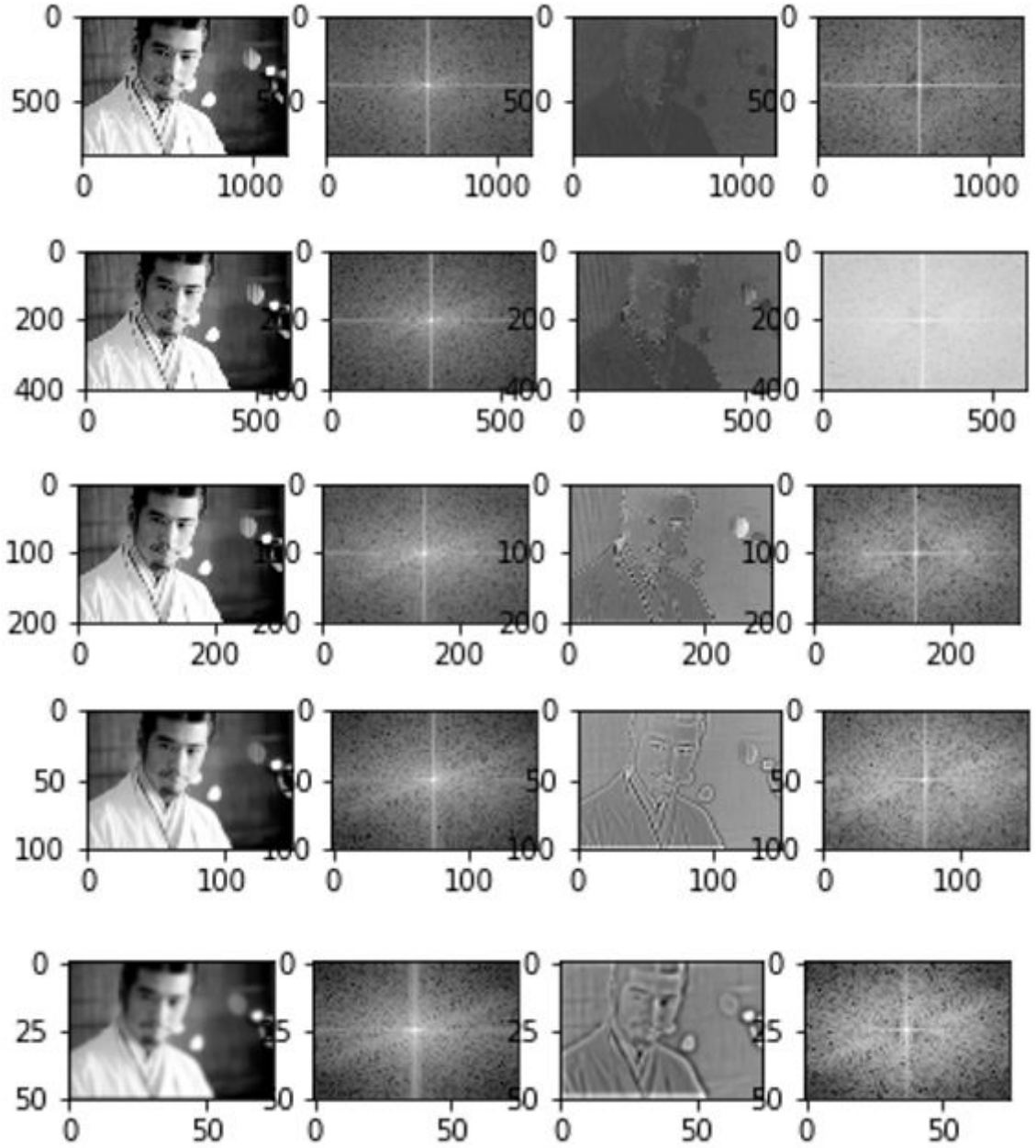


Fig: my own picture



Discussion

Gaussian filter is a low pass filter, so we can observe that in gaussian image magnitude spectrum, the high frequency part is removed when we do gaussian blur. Laplacian is high pass filter, so we can see the laplacian image magnitude spectrum, compare to the same level gaussian image, the high frequency part is retained.

We can apply image pyramid technical on feature detection and template matching because of its multi-scale property.

With different size, there seems to be different finest size for each image. If we choose the small size image like **4_einstein,** the best n=2, and if we choose larger size image like my own image, the best n=4.

Conclusion

We try to build the gaussian image pyramid by smoothing and subsampling, and we get the laplacian image by upsampling and substracting from the gaussian image.

Task 3 (Colorizing the Russian Empire):

Introduction

This task is to automatically produce a color image from the digitized Prokudin-Gorskii glass plate images.The glass plate images record three exposures of every scene onto a glass plate using a red, a green, and a blue filter.
So we need to divide the original image into three images with same size and auto align the G and R channels to the B channel with a simple x,y translation model. However, the full-size glass plate images are very large, the alignment procedure will need to be fast and efficient.

Implementation Procedure

1. **Remove image border**
   Because the original input images have some black borders surrounding them, and we don't want the borders to affect our result images, so we need to remove those black borders before starting colorizing.
   The method is simply just crop the image by some constant scale (eg. remove 0.01height from top and 0.02height from bottom), the reason we choose the constant is actually by try and error.
   Although it is simple to do it, the scale we choose to remove actually correlates heavily to our result image quality.
   The reason that we crop the image border before splitting input image is because it will cause a better result than cropping after splitting.

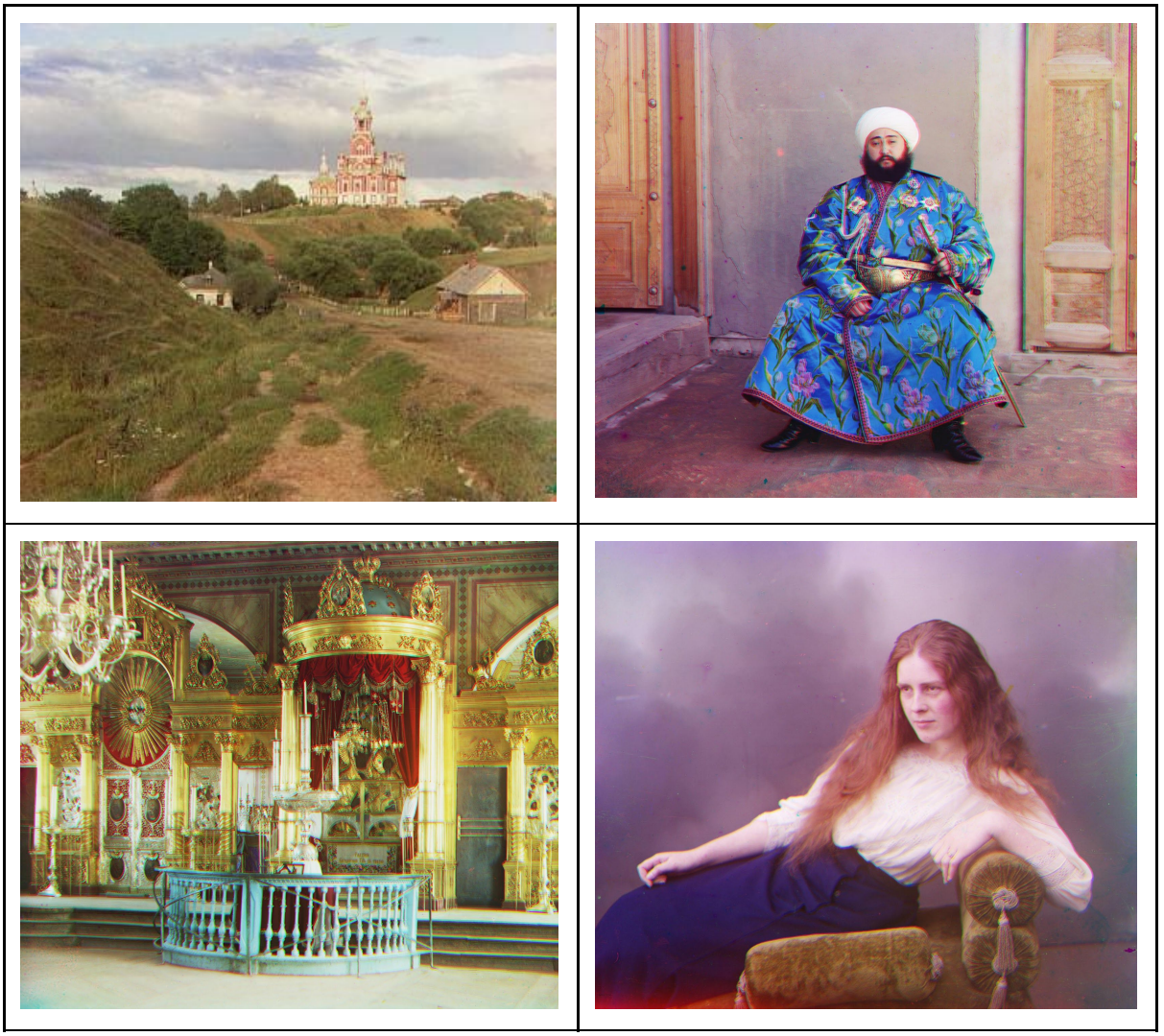
2. **Splitting input image into RGB**
   Equally split input images into three by height.

3. **Normalized cross-correlation(NCC)**
   The method I choose to solve the alignment problem of RGB image is Normalized cross-correlation. The step to calculate the NCC between matrix a and b are:
   1. extract both a by a's mean and b by b's mean.
   2. divide a and b by their L2 norm.
   3. NCC is calculated by a and b's dot product.
   We tried the possible shifting from 1 to 20 (in jpg images) and from 1 to 15 (in tif images) to find the best correlation between red, green, blue images.
   After finding the best shifting size, we shift the red and green image by their best shifting size to align the blue image.

4. **Remove image border**
   After we align them all, we already got some colorized images. But still we have some bad quality effects around our aligned images, so we have to crop the border again to get out result images.

Experimental Result

Execution time for an image: 10s(average)



Discussion

1. **Performance**
   In the tif input images, it took so long to calculate the ncc between RGB channels. For example, the emir.tif image took us over 10 minutes to calculate the ncc because of its 3702*9627 high resolution. So to deal with the problem, we search for some possible solutions on the net. We found that we can simply resize the original images to its 1/10, and then calculate their ncc, finally times the shifting size by 10 on the original RGB channel images. This method improves our performance by 100 times, and the quality of result image is still good enough that human eyes can't distinguish their difference.

2. **Quality**

The quality of our images are mostly quite well, but there are still some misalignment in our result images. For example:

We can see that if we amplify the image, in the first one, it has some blur effect, and there's even a misalignment of green channel in the second one.

Conclusion

We can do the image registration task simply by three steps: remove image border, splitting input image into RGB and do normalized cross-correlation(NCC).
Although it still has some potential improvement due to or work, like the alignment issue, it is still an interesting task to do with. Image registration still has many applications to do with, like medical imaging, astrophotography…etc.

Work assignment plan between team members

0856621 王彥儒	Task1 Hybrid Image
0856618 向修沅	Task2 Image Pyramid
0856630 朱恩逵	Task3 Colorizing