Computer Vision

Assignment 4

Alireza Moradi

October 18, 2020

1

1.1

Because if we traverse a vertical line in the picture, The color will change in a periodic manner. But if we traverse a horizontal line the color won't change across the path, It's either black or white.

1.2

The colors will change in $\frac{1}{2}\times$ of the original frequency so the white dots will be farther apart.

1.3

The colors will change in $2\times$ of the original frequency so the white dots will be closer.

2

2.1

In (0,0), $e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})} = 1 \Rightarrow F(0,0) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y) = 64 \times 20 = 1280$

2.2

Fourier transform of an image can be computed using the below formula:

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})}$$

$M, N = 2$ so the above formula will change to:

$$F(u,v) = \sum_{x=0}^{1}\sum_{y=0}^{1} f(x,y)e^{-j2\pi(\frac{ux}{2}+\frac{vy}{2})}$$

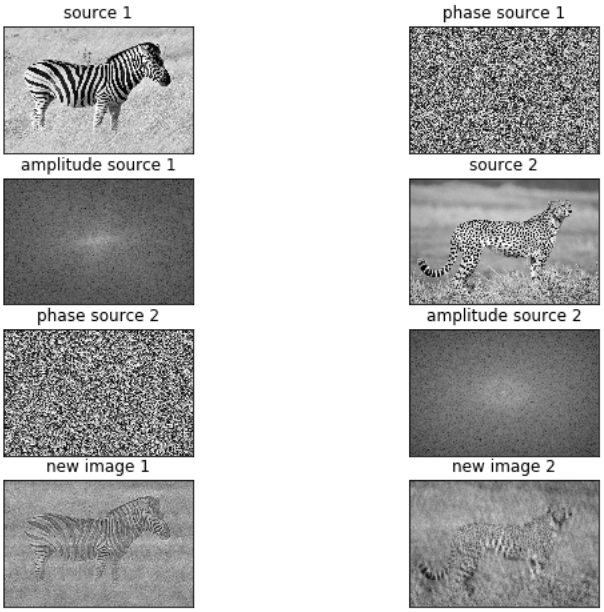
1	2
2	1

$$\Rightarrow \quad F(u,v) = \begin{cases} F(0,0) = 6 \\ F(0,1) = 1 + 2 + 3e^{-j\pi} = 0 \\ F(1,0) = 1 + 2 + 3e^{-j\pi} = 0 \\ F(1,1) = 1 + 4e^{-j\pi} + e^{-j2\pi} = -2 \end{cases}$$

3 Implementations

3.1 Phase and Domain

I used numpy's fft2 to compute fourier transform o image (As said in the class) and fftshift to bring the corners to the center (As said in the class). I computed *log* of amplitude for visualization (As said in the class). Source
For the second part I used element-wise matrix exponentiation and element-wise multiplication $|f1| * e^{\angle f2}$ and vice versa. Source
The conclusion is that most of the info is in the phase.

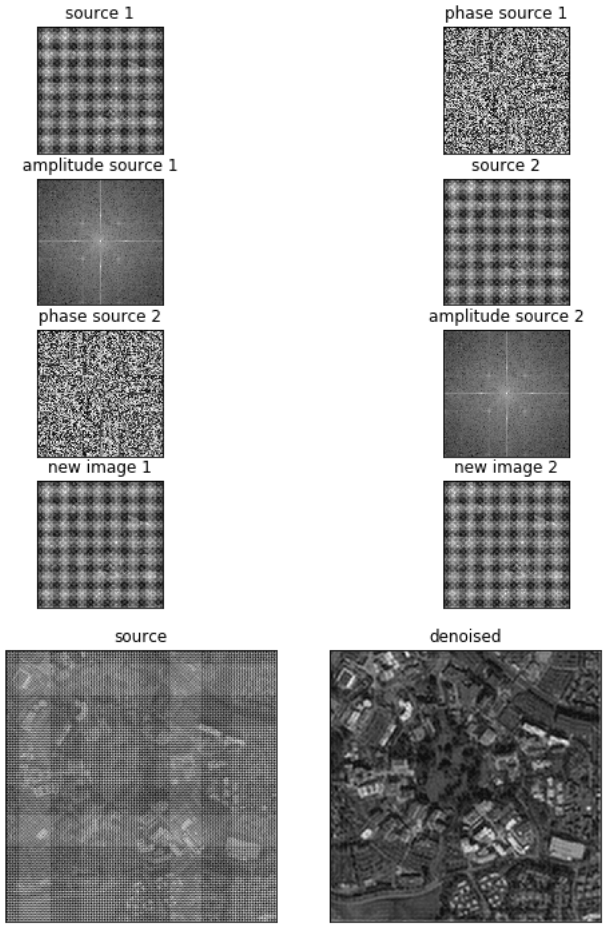



```python
def draw_phase_amplitude(image):
    '''
    Returns the phase image and the amplitude image from the input image.

    Parameters:
        image (numpy.ndarray): The input image.

    Returns:
        tuple of numpy.ndarray: The tuple of the phase image and the amplitude image.
    '''

    phase = image.copy()
    amp = image.copy()

    #Writer your code here
    f = np.fft.fft2(phase)
    fshift1 = np.fft.fftshift(f)
    phase = np.angle(fshift1)
    amp = 20*np.log(np.abs(fshift1))
```

```
20
21      return phase, amp
22
23  def change_phase_domain(image1, image2):
24      '''
25      Substitutes the phase of image1 by the phase of image2 and returns two new images.
26
27      Parameters:
28          image1 (numpy.ndarray): The input image1.
29          image2 (numpy.ndarray): The input image2.
30
31      Returns:
32          tuple of numpy.ndarray: The tuple of result images.
33      '''
34
35      img1 = image1.copy()
36      img2 = image2.copy()
37
38      #Write your code here
39      f1 = np.fft.fft2(img1)
40      f2 = np.fft.fft2(img2)
41
42      combined1 = np.multiply(np.abs(f2), np.exp(1j*np.angle(f1)))
43      combined2 = np.multiply(np.abs(f1), np.exp(1j*np.angle(f2)))
44      img1 = np.real(np.fft.ifft2(combined1))
45      img2 = np.real(np.fft.ifft2(combined2))
46
47      return img1, img2
```

3.2 Noise Removal

As we can see in the image, we have some vertical and horizontal noises in the image. Which can be seen in the amplitude as 4 dots and 2 vertical and horizontal lines. So if we remove these the noise will be gone.




```python
def denoise_image(image):
    '''
    Denoises the input image.

    Parameters:
        image (numpy.ndarray): The input image.

    Returns:
        numpy.ndarray: The result denoised image.
    '''

    denoised = image.copy()

    #Write your code here
    f = np.fft.fft2(denoised)
```

```
    fshift = np.fft.fftshift(f)

    for i in range(fshift.shape[0]):
        for j in range(fshift.shape[1]):
            if not (i >= 195 and i <= 317 and j >= 195 and j <= 317):
                fshift[i][j] = 0

    ifshift = np.fft.ifftshift(fshift)
    f = np.real(np.fft.ifft2(ifshift))
    denoised = f
    return denoised
```

3.3 Image Enhancement

I used `Gaussian High Pass` filter here. I used a threshold `D0` to specify the length of the filter (radius of the circle in the middle of the filter which determines what are the high frequencies that can pass the filter). Then I multiplied the filter with the fourier transform of the image and computed the invert fourier transform of the result. At last I added the filtered image to the original image. Source


```python
def distance(point1,point2):
    return np.sqrt((point1[0]-point2[0])**2 + (point1[1]-point2[1])**2)

def gaussianHP(D0,imgShape):
    base = np.zeros(imgShape[:2],dtype=np.float64)
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            base[y,x] = 1 - np.exp(((-distance((y,x),center)**2)/(2*(D0**2))))
    return base

def enhance_image(image):
    '''
    Enhances the input image by applying a filter in the frequency domain.

    Parameters:
        image (numpy.ndarray): The input image.

    Returns:
        numpy.ndarray: The result enhanced image.
    '''

    enhanced = image.copy()
    r,c = image.shape
    #Write your code here

    kernel = gaussianHP(50,image.shape)
```

```
29
30      f = np.fft.fft2(enhanced)
31      fshift = np.fft.fftshift(f)
32
33      fshift = fshift * kernel
34
35      ifshift = np.fft.ifftshift(fshift)
36      f = np.real(np.fft.ifft2(ifshift))
37
38      enhanced = np.array(enhanced,np.float64) + f
39
40      return enhanced
```