پاسخ سوال اول

الف)

تصویر داده شده نشان می دهد که تبدیل فوریه یک تصویر جزئیاتی از ساختار هندسی آن را ارائه می دهد. نوارهای افقی متوالی در تصویر اصلی به یک خط عمودی در تصویرنتیجه، تبدیل شده اند.همچنین چون تناوب ایجاد شده در طول تصویر قابل مشاهده است،این نقاط متوالیا در این راستا ظاهر شده اند.باید توجه داشت که این خط در یک طول قرار گرفته است(منظور از خط ،3 نقطه ی متوالی است) و این واقعیت را نشان می دهد که تمام خطوط عمودی در تصویر اصلی کاملاً یکنواخت از لحاظ اندازه هستند. در تصویر یک نقطه در مرکز وجود دارد که نشان دهنده مقدار متوسط تصویر است.

ب)در این حالت فاصله ی نقاط نسبت به حالت قبل نصف میشود.

ج)در این حالت فاصله ی نقاط نسبت به وقتی که اندازه خطوط 2 پیکسل بود،2برابر میشود.

منابع:

لینک اول

لینک دوم

لینک سوم

لینک چهارم

پاسخ سوال دوم

الف)

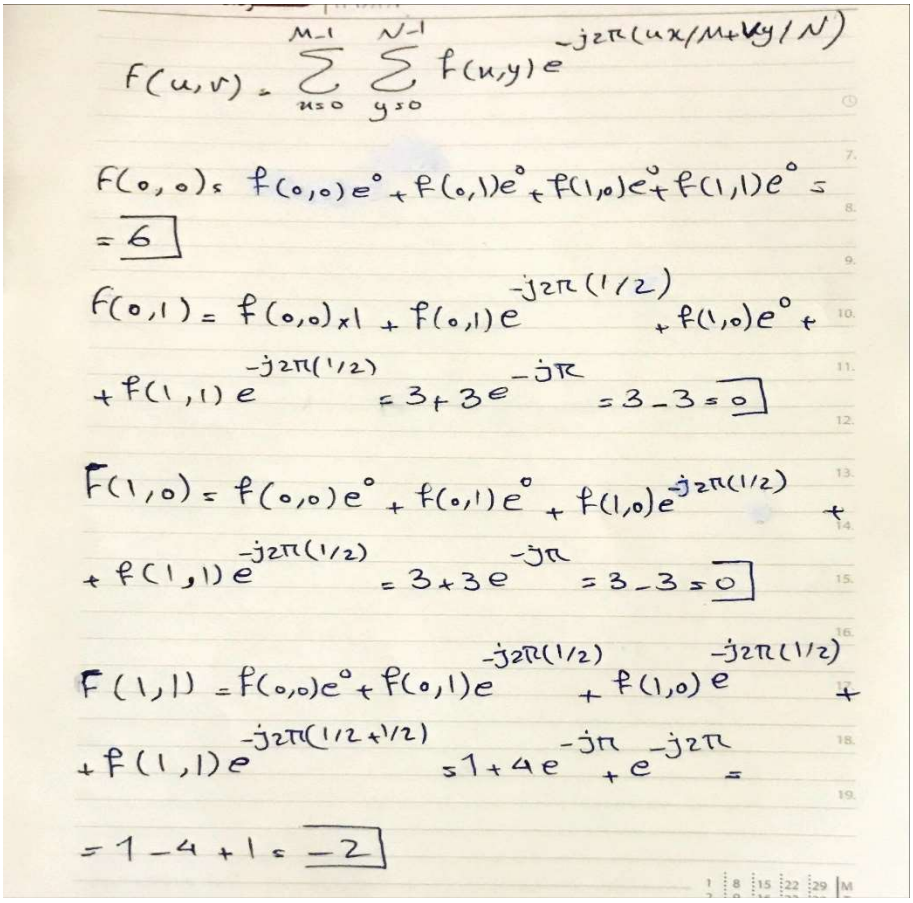
با استفاده از رابطه ی:

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

پیش می رویم اگر u و v را برابر با صفر بگیریم و میدانیم که M*N=64 و میانگین نیز برابر 20 است پس:

F(0,0)=20*64=1280

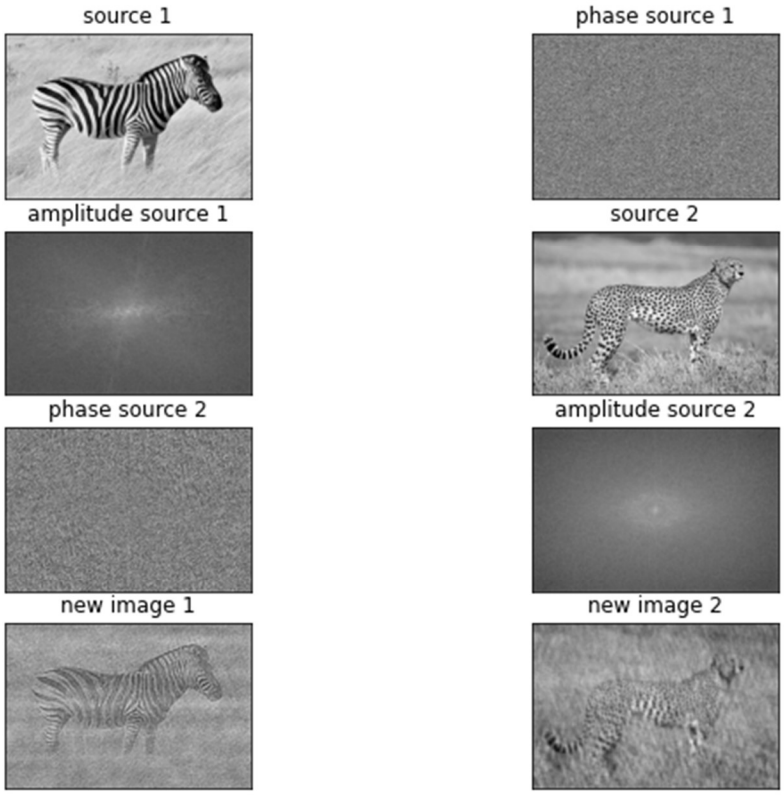
ب)



گزارش کد های پیاده سازی شده

پاسخ سوال سوم

ب)نتیجه های این سوال به صورت زیر است:



فاز تصویر دارای اطلاعات اصلی تصویر است و حتی فاز تصویر برای دوباره ساختن تصویر کافی است.ممکن دو تصویر که محتوای متفاوتی دارند دارای دامنه ی یکسانی باشند چیزی که این تفاوت محتوا رو مشخص میکند،فاز تصویر است.

منابع:

لینک اول

لینک دوم

لینک سوم

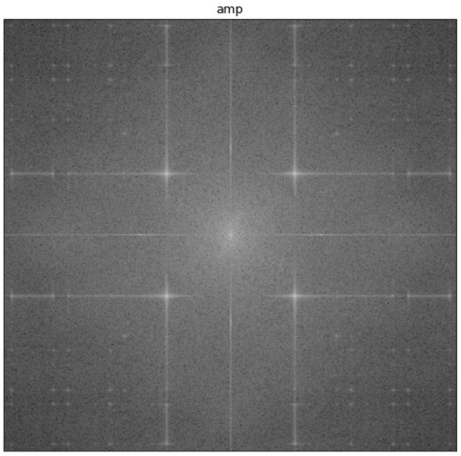
```python
def draw_phase_amplitude(image):
    '''
    Returns the phase image and the amplitude image from the input image.

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

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

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

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

    return phase, amp

def change_phase_domain(image1, image2):
    '''
    Substitutes the phase of image1 by the phase of image2 and returns two new im

    Parameters:
        image1 (numpy.ndarray): The input image1.
        image2 (numpy.ndarray): The input image2.

    Returns:
        tuple of numpy.ndarray: The tuple of result images.
    '''

    img1 = image1.copy()
    img2 = image2.copy()

    #Write your code here
    p1,a1=phase_amplitude(img1)
    p2,a2=phase_amplitude(img2)
    img1=np.real(np.fft.ifft2(np.multiply(a2,np.exp(1j*p1))))
    img2=np.real(np.fft.ifft2(np.multiply(a1,np.exp(1j*p2))))

    return img1, img2
```

برای پیاده سازی این سوال از توابع کتابخانه ی numpy استفاده کرده ام.برای محاسبه تبدیل فوریه از fft2 که برای دو بعد است استفاده کردم،سپس نتیجه را شیفت داده ام .در انتها با تابع angle فاز را بدست آوردم و با تابع abs و با کمک از log دامنه را محاسبه کرده ام.

پاسخ سوال چهارم

در این سوال ابتدا تصویرتبدیل را نمایش دادم محل هایی که نویز بودند را حدودی مشخص کردم کجا هستند و آنها را حذف کردم.

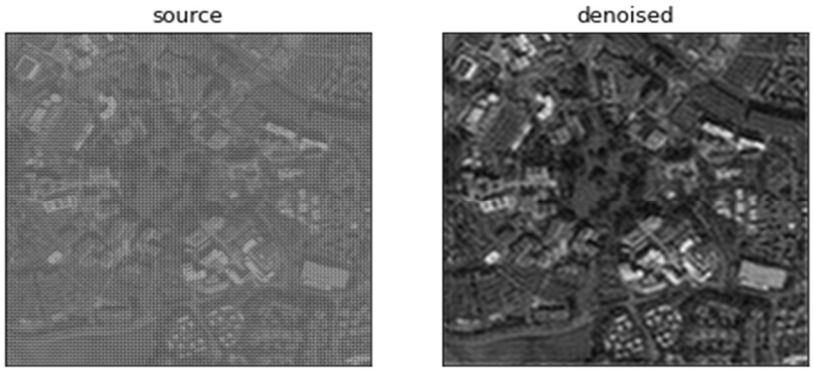
همانطور که مشخص است حدود 206 واحد از طرفین عکس نویز دارند که حذف کردم.



کد نوشته برای این سوال به شکل زیر است که مانند سوال قبل از توابع fft کتابخانه numpyاستفاده شده است:

```
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(image)
    fshift = np.fft.fftshift(f)
    amp=np.log(np.abs(fshift))
    m,n=image.shape
    #print(m,n)
    for i in range(0,206):
        for j in range(0,512):
            fshift[i,j]=0
            fshift[j,i]=0
    for i in range(306,512):
        for j in range(0,512):
            fshift[i,j]=0
            fshift[j,i]=0


    denoised=np.real(np.fft.ifft2(np.fft.ifftshift(fshift)))

    return denoised
```

نتیجه به شکل زیر بود که بسیار رضایت بخش است:



منبع:برای حل این سوال از خانم مریم سادات هاشمی راهنمایی گرفتم.

پاسخ سوال پنجم

در این سوال با استفاده از یک فیلتر بالا گذر ،باید تصویر برا بهبود میدادیم.من فیلتر butterworth را در نظر گرفتم.همانطور که میدانیم برای محاسبه ی فیلتر های بالا گذر باید چنین اقدام کرد:

$$H_{\mathrm{HP}}(u,v) = 1 - H_{\mathrm{LP}}(u,v)$$

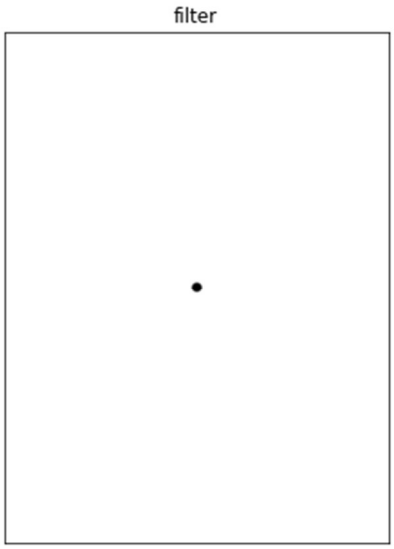
یعنی اختلاف عدد یک و فیلتر پایین گذر را حساب نمود.رابطه ی مورد نظر برای فیلتر پایین گذر butterworth به صورت زیر است:

$$H(u,v) = \frac{1}{1 + \left[D(u,v)/D_0\right]^{2n}}$$

در اینجا D0 در واقع شعاع فیلتر است و n مرتبه ی این فرمول است.با ایجاد تغییر در این دو مقدار میتوان شارپ بودن تصویر را کنترل کرد.

تصویر فیلتری که من استفاده کردم به صورت زیر است:

من D0=20 و n=10 گرفتم.



کد من به صورت مقابل است:

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

def butterworthHP(D0,imgShape,n):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            base[y,x] = 1-1/(1+(distance((y,x),center)/D0)**(2*n))
    return base
```

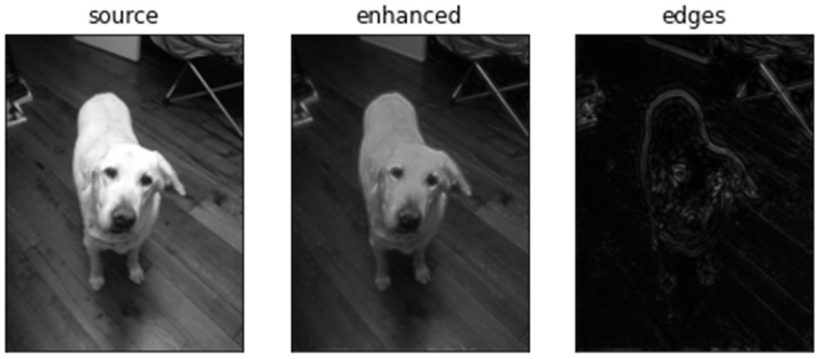
Implement this function for enhancing input image and return result image.

```
def enhance_image(image):

    enhanced = image.copy()

    #Write your code here
    r,c = image.shape
    #Write your code her
    original = np.fft.fft2(image)
    center = np.fft.fftshift(original)
    HighPassCenter = center * butterworthHP(20,image.shape,10)
    HighPass = np.fft.ifftshift(HighPassCenter)
    inverse_HighPass = np.fft.ifft2(HighPass)
    enhanced = np.array(enhanced,np.float64) + np.abs(inverse_HighPass)
    return enhanced,np.abs(inverse_HighPass),butterworthHP(20,image.shape,10)
```

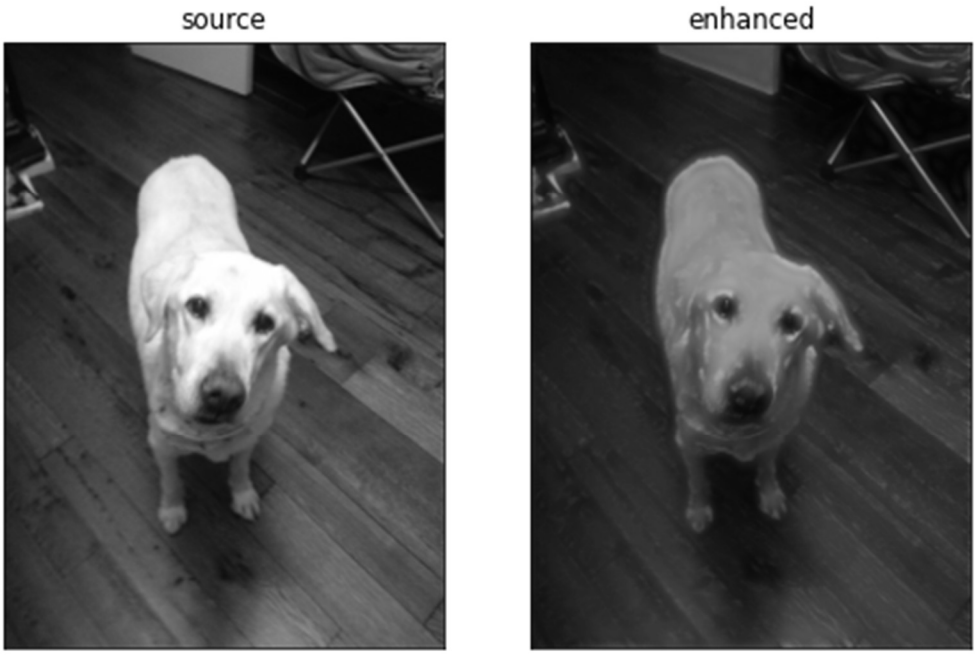
که در آن ابتدا لبه های تصویر را بدست آوردم و با خود تصویر جمع کردم تا تصویری شارپ تر حاصل شود.برای بدست آوردن لبه از فیلتر های حوزه فرکانس بالا گذر استفاده کردم.

ابتدا از تصویر fft گرفتم و فیلتر را روی آن اعمال کردم سپس از این مقدار ifft گرفتم و با تصویر اصلی نیز جمع کردم نتیجه ی زیر حاصل شد:

Enhancedتصویر بهبود یافته است و edges لبه هایی که با استفاده از فیلتر بالاگذرbutterworth بدست آوردیم.



خروجی سوال به صورت دقیق تر:



منبع:

لینک اول

لینک دوم

لینک سوم

همچنین از خانم مریم سادات هاشمی کمک گرفتم.