

تمرين ۴ سوال ۱-۱

چکیده

در این تمرین ابتدا Magnitude Response انواعی از فیلتر ها را در فضای فوریه مشاهده کرده و با اعمال آن ها بر تصویر، متوجه کاربرد آنها می شویم. همچنین Separability را در فیلتر های داده شده بررسی کرده و در صورت جدایی پذیری، فیلتر های جدا شده را به صورت جداگانه بر تصویر اعمال می کنیم و نتایج را بررسی می کنیم.

مقدمه

○ بخش اعمال فیلتر

در تمارین بخش ۴، اعمال فیلتر را در فضای فوریه خواهیم داشت که برای اعمال هر فیلتر در این فضا، مراحل زیر را طی می کنیم:

۱. تصویر جدیدی با سایز ۲ برابر طول و عرض تصویر قبل می سازیم.
۲. به جای پیکسل های باقی مانده، مقدار صفر را قرار می دهیم. (عملیات Zero Padding)
۳. DFT را بر تصویر اعمال می کنیم.
۴. تصویر را شیفت داده تا مبدا به مرکز تصویر منتقل شود. (پس از DFT، مجموع تمام پیکسل های تصویر در مبدا قرار دارد که با توجه به خاصیت فیلتر ها، باید آن نقطه را به مرکز تصویر منتقل کنیم.)
۵. ضرب آرایه ای فیلتر (که خود در فضای فوریه بوده و Shift نیز خورده است).
۶. بار دیگر تصویر را شیفت معکوس می دهیم تا مرکز تصویر به مبدا منتقل شود. (این کار را برای اجرای صحیح Inverse DFT انجام می دهیم).

۷. IDFT را بر تصویر اعمال می کنیم تا تصویر به حوزه مکان بازگردد.
۸. قسمت حقیقی مقادیر را انتخاب می کنیم. (به علت محاسبات پیچیده ممکن است بخش های موهومی نیز به تصویر اضافه شده باشد).

۹. بخش بالا سمت چپ تصویر Zero Padded را به عنوان نتیجه برداشته و نمایش می دهیم.
بنابراین این مراحل را در کد پیاده سازی می کنیم و در تمارین بعدی این بخش نیز از آن استفاده می کنیم. توجه شود که فیلتر مورد استفاده در این الگوریتم باید در حوزه فرکانس و شیفت خورده شده باشد.

○ بخش بدست آوردن Magnitude

در این بخش که باز هم در تمارین آینده مورد کاربرد قرار می گیرد، به نمایش Magnitude فیلتر ها (به طور کلی و نه صرفا برای فیلتر ها، توضیح داده می شود) می پردازیم. برای این کار کافیست تا DFT را بر فیلتر یا تصویر اعمال کنیم و پس از شیفت دادن و انتقال دادن مجموع تمام پیکسل ها به مرکز تصویر، از آن لگاریتم گرفته و این مقدار را نمایش دهیم.

علت اعمال تابع لگاریتم در این بخش این است که مقادیر موجود در تصویر پس از انتقال به حوزه فوریه می‌توانند مقادیر بسیار بزرگی داشته باشند و برای قابل فهم شدن این مقادیر، باید از آنها لگاریتم بگیریم. با نمایش Magnitude می‌توانیم به اطلاعات زیادی درباره لبه‌های سیگنال پی برد که این کار را در بخش شرح نتایج انجام می‌دهیم.

○ بخش Separability

در بخش دیگری از تمرین از ما خواسته شده است تا Separability فیلترهای داده شده را بررسی کرده و در صورت جدا شدن، هر فیلتر را به صورت جداگانه بر تصویر اعمال کنیم. مفهوم کلی Separability به این برمیگردد که آیا می‌توانیم فیلتر را از ضرب دو فیلتر متفاوت دیگر بدست آوریم یا نه. با توجه به فیلترهای داده شده، فیلترهای b و c این خاصیت را ندارند و در فیلتر a، این خاصیت به صورت زیر دیده می‌شود:

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

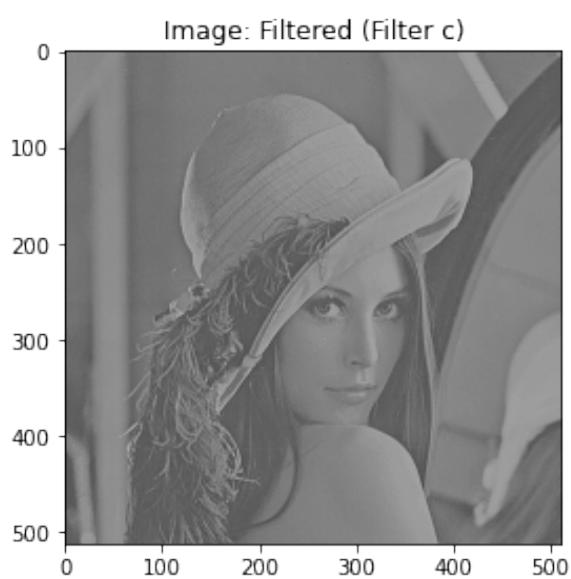
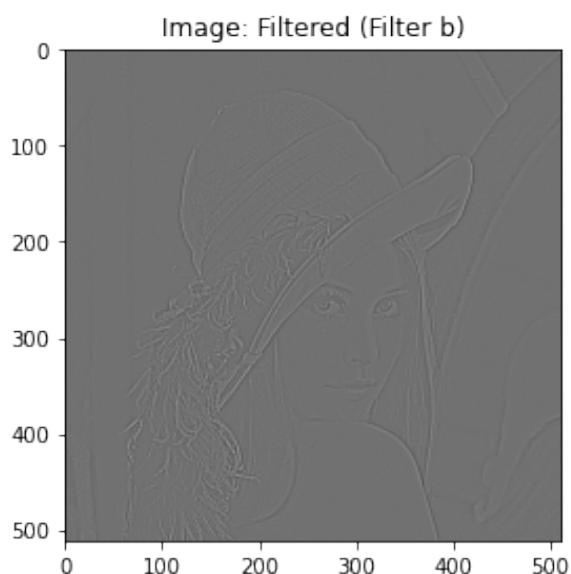
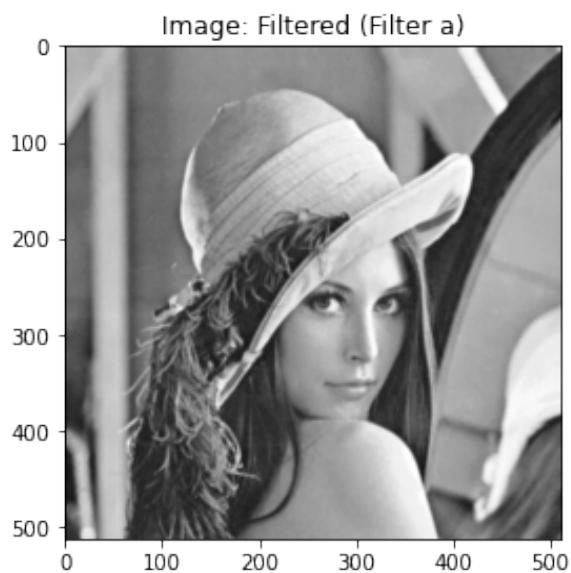
بنابراین هر کدام از دو فیلتر بدست آمده را به صورت جداگانه بر تصویر اعمال می‌کنیم. (برای اعمال مشخصاً لازم است تا به حوزه فرکانس منتقل شده و شیفت داده شوند).

شرح نتایج

○ بخش اعمال فیلتر

در این بخش ابتدا تصویر اصلی را مشاهده کرده و پس از آن فیلترهای a و b و c را به تصویر اعمال می‌کنیم.





همانطور که از نتایج بر می آید، فیلتر های اعمال شده هر کدام کاربرد های زیر را دارند. (این کاربرد ها را با استفاده از مقادیر فیلتر ها نیز بررسی می کنیم).

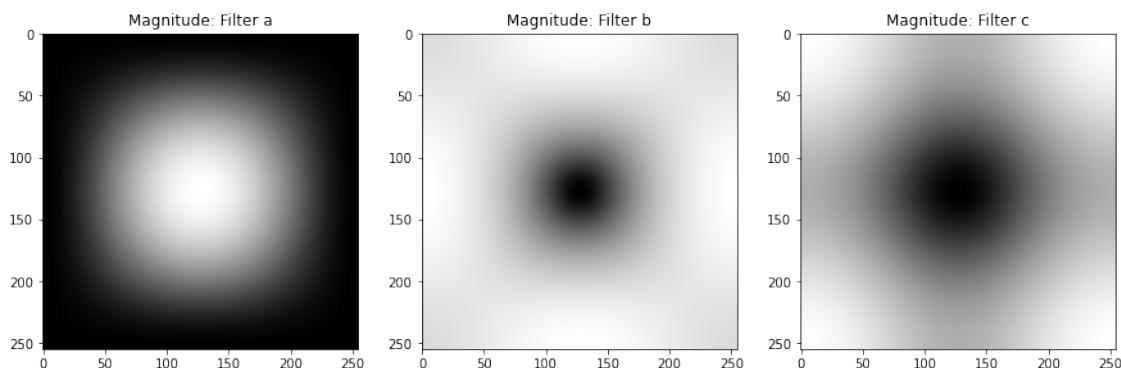
a. این فیلتر در نتیجه تصویر آن را Blur کرده است که با توجه به coefficients های فیلتر وجود وزن در آن، تصویر با فیلتر Gaussian تار شده است. (در این فیلتر مرکز دارای وزن بیشتری است).

b. با توجه به مقادیر موجود در فیلتر، کاربرد آن در بدست آوردن تمام لبه ها در جهات Vertical, Horizontal و Diagonal است که در تصویر نتیجه نیز می توان تمام لبه های تصویر در تمام جهات را مشاهده کرد.

c. این فیلتر با توجه به وجود ۵ در مرکز خود، گواهی از فیلتر های High boost می دهد. در این فیلتر ها ابتدا لبه ها را در جهات مشخص (در اینجا در جهات Vertical و Horizontal) تشخیص داده و با جمع کردن آنها با تصویر اصلی، لبه ها را در تصویر نتیجه برجسته تر می کنیم. این ویژگی نیز در تصویر مربوطه قابل مشاهده است.

○ بخش بدست آوردن Magnitude

در این بخش نتایج Magnitude هر فیلتر را مشاهده می کنیم:



این فیلتر ها پس از انتقال به حوزه فرکانس، شیفت خورده اند که به صورت واضح نیز دیده می شود که مجموع پیکسل ها در مرکز Magnitude قرار دارد. همچنین این نتایج پس از لگاریتم گرفتن نمایش داده می شوند.

○ بخش Separability

با جدا کردن فیلتر a و اعمال هر کدام به صورت جداگانه بر تصویر، به نتیجه زیر می رسیم.



تفاوتش که این فیلتر ها با یکدیگر دارند در این است که در فیلتر $\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$ میانگین گیری وزن دار بین پیکسل های بالا و پایین انجام شده و در فیلتر $\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$ در راست و چپ آن اعمال می شود.

```
In [ ]: import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from google.colab import drive
```

```
In [ ]: drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
In [ ]: img_gray = cv2.imread("/content/drive/MyDrive/Images/4/Lena.bmp", 0)
```

```
In [ ]: def prepare_filter(R, C, filter):
    R_filter, C_filter = filter.shape

    filter_padded = np.zeros((R, C))
    filter_padded[0:R_filter, 0:C_filter] = filter

    filter_DFT = np.fft.fft2(filter_padded)

    filter_shift = np.fft.fftshift(filter_DFT)

    return filter_shift
```

```
In [ ]: def get_filter_magnitude(filter):
    filter_shift = prepare_filter(255, 255, filter)

    magnitude = np.log(np.abs(filter_shift)) + 1

    return magnitude
```

```
In [ ]: def apply_filter(image, filter):
    R, C = image.shape
    R_padded = 2 * R
    C_padded = 2 * C

    image_padded = np.zeros((R_padded, C_padded))
    image_padded[0:R, 0:C] = image

    image_DFT = np.fft.fft2(image_padded)

    image_shift = np.fft.fftshift(image_DFT)
    filter_shift = prepare_filter(R_padded, C_padded, filter)

    image_filtered = image_shift * filter_shift

    image_filtered_inverse_shift = np.fft.ifftshift(image_filtered)

    image_filtered_IDFT = np.fft.ifft2(image_filtered_inverse_shift)

    image_filtered_real = image_filtered_IDFT.real

    image_filtered_result = image_filtered_real[0:R, 0:C]

    return image_filtered_result
```

```
In [ ]: filter_a = np.array([[1/16, 2/16, 1/16], [2/16, 4/16, 2/16], [1/16, 2/16, 1/16]])
filter_b = np.array([[-1, -1, -1], [-1, 8, -1], [-1, -1, -1]])
filter_c = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])
```

```
In [ ]: # Filters Magnitudes
```

```
filter_a_magnitude = get_filter_magnitude(filter_a)
filter_b_magnitude = get_filter_magnitude(filter_b)
filter_c_magnitude = get_filter_magnitude(filter_c)

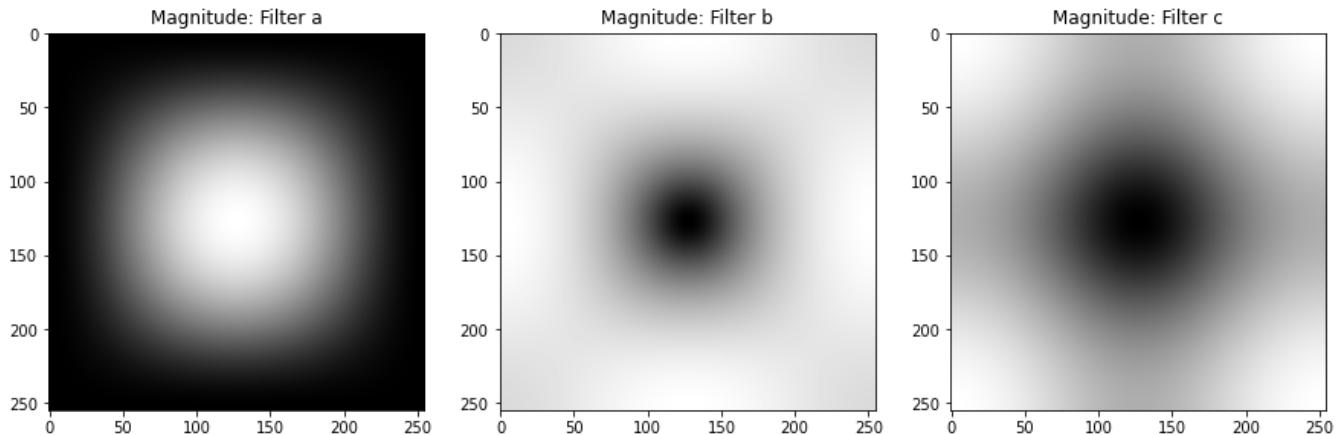
fig, plot = plt.subplots(1, 3, figsize = (15, 5))

plot[0].imshow(filter_a_magnitude, cmap='gray')
plot[0].set_title("Magnitude: Filter a")

plot[1].imshow(filter_b_magnitude, cmap='gray')
plot[1].set_title("Magnitude: Filter b")

plot[2].imshow(filter_c_magnitude, cmap='gray')
plot[2].set_title("Magnitude: Filter c")
```

```
Out[ ]: Text(0.5, 1.0, 'Magnitude: Filter c')
```



```
In [ ]: # Applying Filters on Image
```

```
filtered_image_filter_a = apply_filter(img_gray, filter_a)
filtered_image_filter_b = apply_filter(img_gray, filter_b)
filtered_image_filter_c = apply_filter(img_gray, filter_c)

fig, plot = plt.subplots(4, 1, figsize = (5, 20))

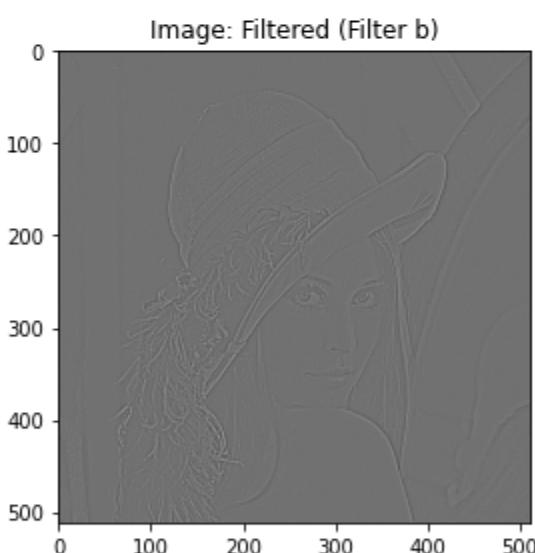
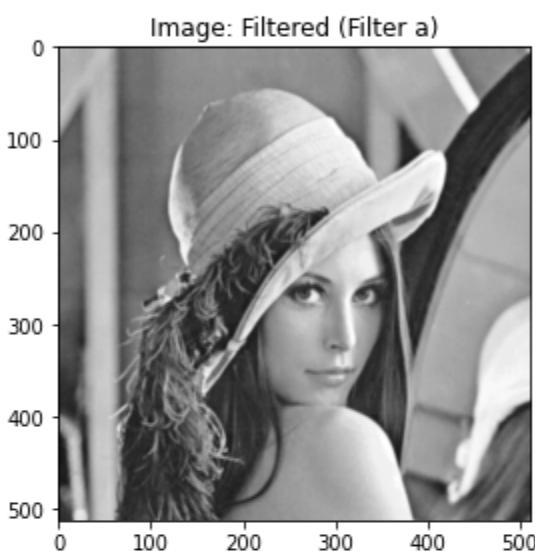
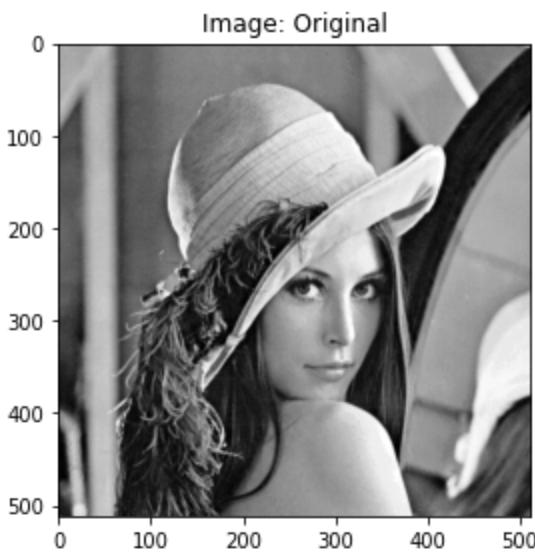
plot[0].imshow(img_gray, cmap='gray')
plot[0].set_title("Image: Original")

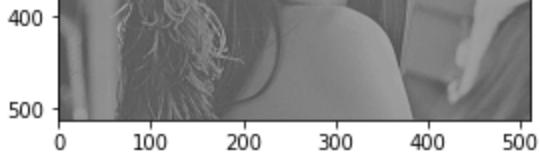
plot[1].imshow(filtered_image_filter_a, cmap='gray')
plot[1].set_title("Image: Filtered (Filter a)")

plot[2].imshow(filtered_image_filter_b, cmap='gray')
plot[2].set_title("Image: Filtered (Filter b)")

plot[3].imshow(filtered_image_filter_c, cmap='gray')
plot[3].set_title("Image: Filtered (Filter c)")
```

Out[]: Text(0.5, 1.0, 'Image: Filtered (Filter c)')





In []: # Separability Test on Filter a

```
sperated_filter_a_1 = np.array([[1/4], [1/2], [1/4]])
sperated_filter_a_2 = np.array([[1/4, 1/2, 1/4]])

filtered_image_filter_a = apply_filter(img_gray, filter_a)
filtered_image_filter_a_separated_1 = apply_filter(img_gray, sperated_filter_a_
1)
filtered_image_filter_a_separated_2 = apply_filter(img_gray, sperated_filter_a_
2)

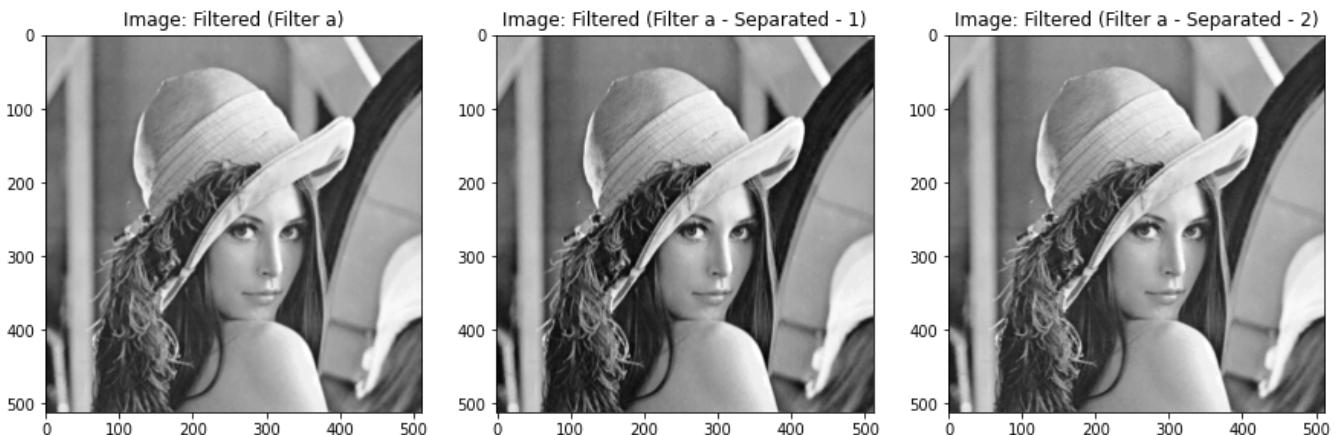
fig, plot = plt.subplots(1, 3, figsize = (15, 5))

plot[0].imshow(filtered_image_filter_a, cmap='gray')
plot[0].set_title("Image: Filtered (Filter a)")

plot[1].imshow(filtered_image_filter_a_separated_1, cmap='gray')
plot[1].set_title("Image: Filtered (Filter a - Separated - 1)")

plot[2].imshow(filtered_image_filter_a_separated_2, cmap='gray')
plot[2].set_title("Image: Filtered (Filter a - Separated - 2)")
```

Out[]: Text(0.5, 1.0, 'Image: Filtered (Filter a - Separated - 2)')



تمرين ۴ سوال ۱-۲

چکیده

در اين تمرين به کاربردهای Shift و Log در Magnitude اشاره کرده و با اعمال حالات مختلف آنها بر تصاویر، کاربرد عملی هر کدام را مشاهده می کنيم. همچنان به علت وجود تصاویر مختلف می توانيم لبه های تصاویر را در Magnitude شناسايي کرده و به لبه های تصویر و جهات آنها پي ببريم.

مقدمه

نحوه بدست آوردن آوردن Magnitude و علت استفاده از Shift و Log را در تمرین قبل بررسی کرده و اینجا نتایج مربوطه را می بینیم.

علاوه بر اين کاربردها، به بررسی و شناسایی لبه های تصویر در Magnitude هر کدام می پردازیم. به صورت کلی در حوزه فوریه، برای شناسایی لبه های پر تکرار، به Magnitude دقیق می کنیم که در آن خطوطی عمود بر لبه های پر تکرار را می توانیم ببینیم. بنابراین جهات لبه های اصلی، عمود بر جهات خطوط مشخص شده در Magnitude است.

شرح نتایج

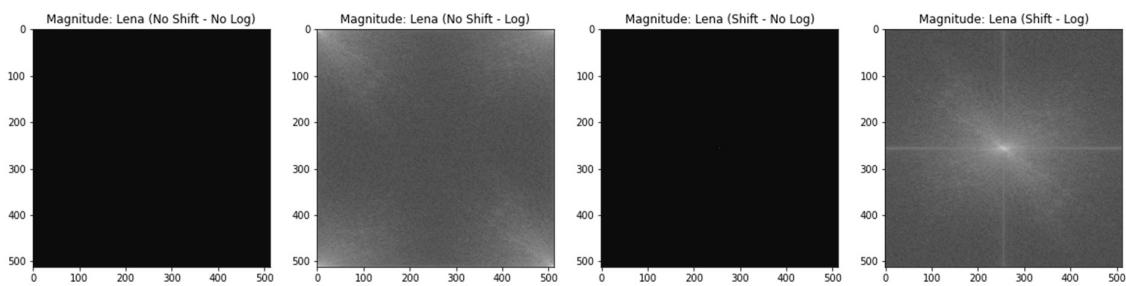
در هر مرحله و برای هر تصویر اصلی را نمایش داده و پس از آن به نمایش Magnitude ها می پردازیم. در همه Magnitude های مشاهده شده، در حالتی که از log استفاده نکنیم، چیزی بجز صفحه سیاه در نتیجه دیده نمی شود و علت آن بزرگ بودن و غیر قابل درک بودن در محیط مورد استفاده است. همچنان نتیجه شیفت دادن در هر کدام به صورت واضح قابل مشاهده است.

:Lena

تصویر اصلی



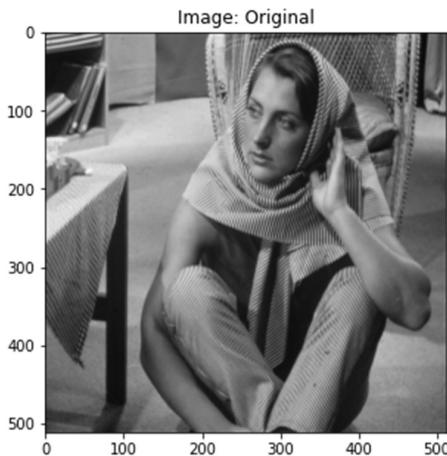
ها Magnitude



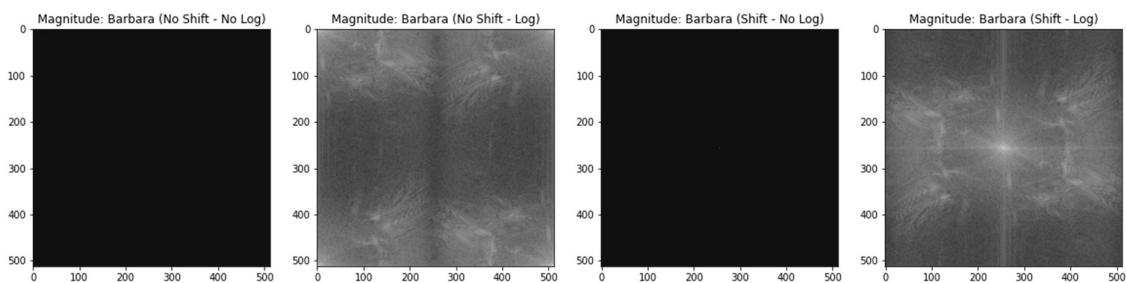
همانطور که دیده می شود، اکثر لبه های تصویر در جهات افقی و عمودی قرار دارد که در Magnitude بدست آمده نیز خط هایی در جهات عمود بر این دو خط دیده می شود. علاوه بر آن، در جهات دیگر هم خطوط کمرنگ تری دیده می شود چراکه تصویر در جهات دیگر نیز دارای لبه است.

:Barbara

تصویر اصلی



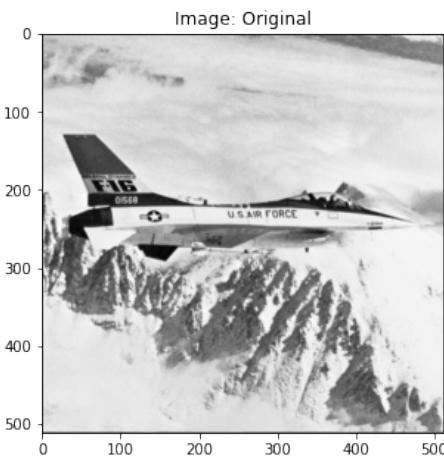
ها Magnitude



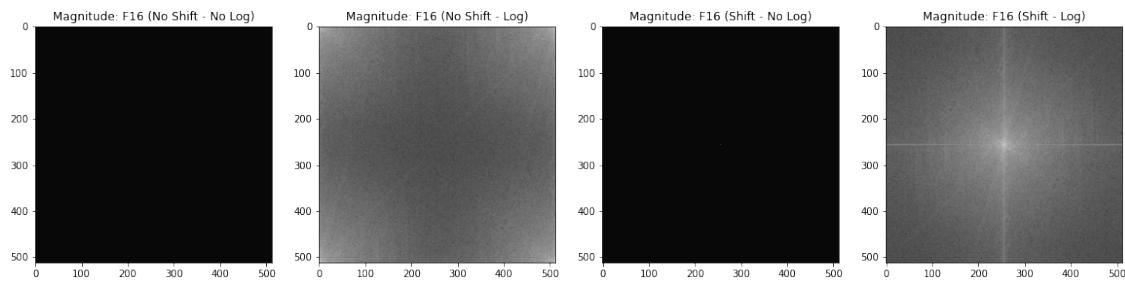
این تصویر علاوه بر لبه در جهات افقی و عمودی، دارای Texture در روسری و شلوار و موکت اتاق و ... است که باعث بوجود آمدن برخی لبه ها می شود که در تصویر به عنوان خطوط محو اطراف آن دیده می شود.

:F16

تصویر اصلی



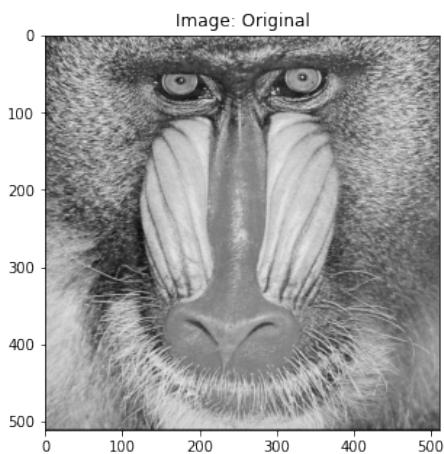
ها Magnitude



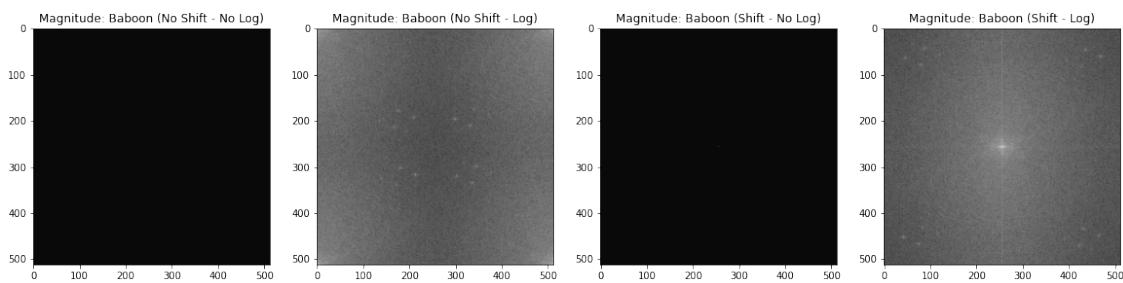
در این تصویر نیز مانند تصویر Lena خطوط در جهات افقی و عمودی قوی تر بوده که نشان دهنده وجود لبه ها در جهات عمود بر آنهاست.

:Baboon

تصویر اصلی



ها Magnitude



در این تصویر لبه های در جهت خاص کمتر بوده ولی به جای آن Texture مو ها را داریم که به صورت نقطه نقطه در چهار گوشه Magnitude دیده می شوند.

```
In [ ]: import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from google.colab import drive
```

```
In [ ]: drive.mount('/content/drive')
```

Mounted at /content/drive

```
In [ ]: img_lena = cv2.imread("/content/drive/MyDrive/Images/4/Lena.bmp", 0)
img_barbara = cv2.imread("/content/drive/MyDrive/Images/4/Barbara.bmp", 0)
img_f16 = cv2.imread("/content/drive/MyDrive/Images/4/F16.bmp", 0)
img_baboon = cv2.imread("/content/drive/MyDrive/Images/4/Baboon.bmp", 0)
```

```
In [ ]: def get_image_magnitude(image, shift_flag, log_flag):
    image_DFT = np.fft.fft2(image)

    if shift_flag:
        image_shift = np.fft.fftshift(image_DFT)
    else:
        image_shift = image_DFT.copy()

    if log_flag:
        magnitude = np.log(np.abs(image_shift) + 1)
    else:
        magnitude = image_shift.copy()

    return magnitude.real
```

In []: # Lena

```
img = img_lena.copy()
name = "Lena"

magnitude_no_shift_no_log = get_image_magnitude(img, False, False)
magnitude_no_shift_yes_log = get_image_magnitude(img, False, True)
magnitude_yes_shift_no_log = get_image_magnitude(img, True, False)
magnitude_yes_shift_yes_log = get_image_magnitude(img, True, True)

fig, plot = plt.subplots(1, 1, figsize = (5, 5))

plot.imshow(img, cmap='gray')
plot.set_title("Image: Original")

fig, plot = plt.subplots(1, 4, figsize = (20, 5))

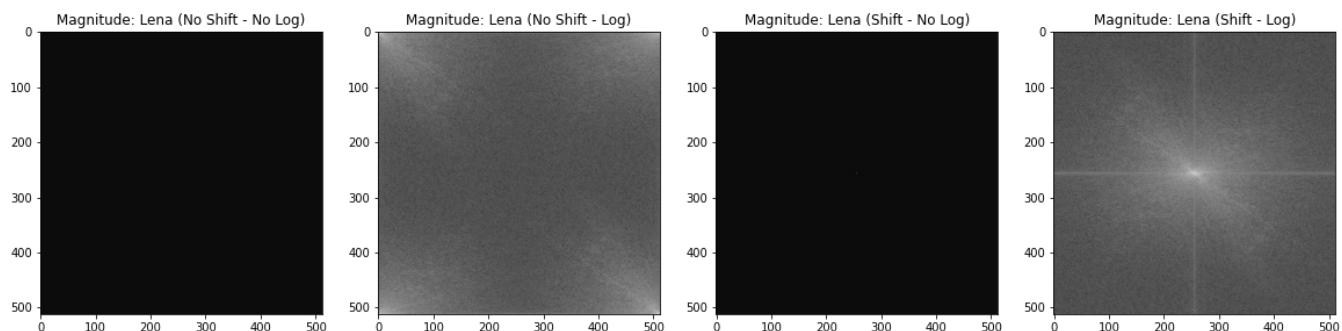
plot[0].imshow(magnitude_no_shift_no_log, cmap='gray')
plot[0].set_title("Magnitude: " + name + " (No Shift - No Log)")

plot[1].imshow(magnitude_no_shift_yes_log, cmap='gray')
plot[1].set_title("Magnitude: " + name + " (No Shift - Log)")

plot[2].imshow(magnitude_yes_shift_no_log, cmap='gray')
plot[2].set_title("Magnitude: " + name + " (Shift - No Log)")

plot[3].imshow(magnitude_yes_shift_yes_log, cmap='gray')
plot[3].set_title("Magnitude: " + name + " (Shift - Log)")
```

Out[]: Text(0.5, 1.0, 'Magnitude: Lena (Shift - Log)')



In []: # Barbara

```
img = img_barbara.copy()
name = "Barbara"

magnitude_no_shift_no_log = get_image_magnitude(img, False, False)
magnitude_no_shift_yes_log = get_image_magnitude(img, False, True)
magnitude_yes_shift_no_log = get_image_magnitude(img, True, False)
magnitude_yes_shift_yes_log = get_image_magnitude(img, True, True)

fig, plot = plt.subplots(1, 1, figsize = (5, 5))

plot.imshow(img, cmap='gray')
plot.set_title("Image: Original")

fig, plot = plt.subplots(1, 4, figsize = (20, 5))

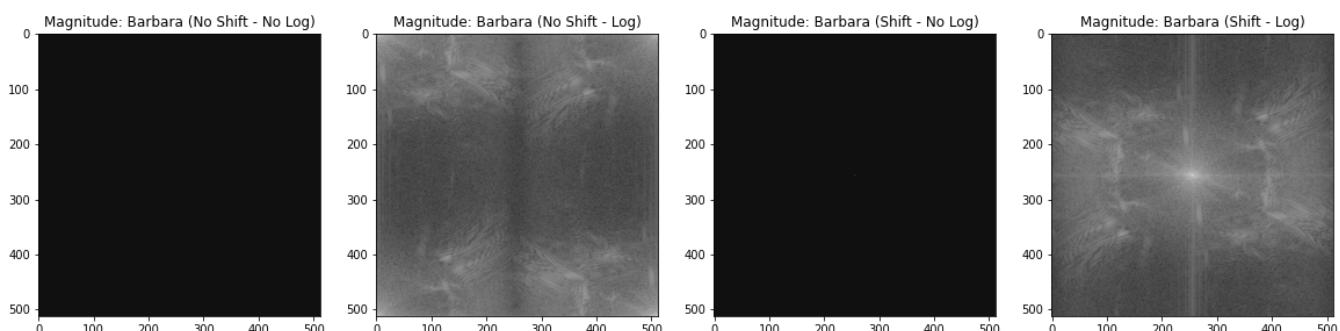
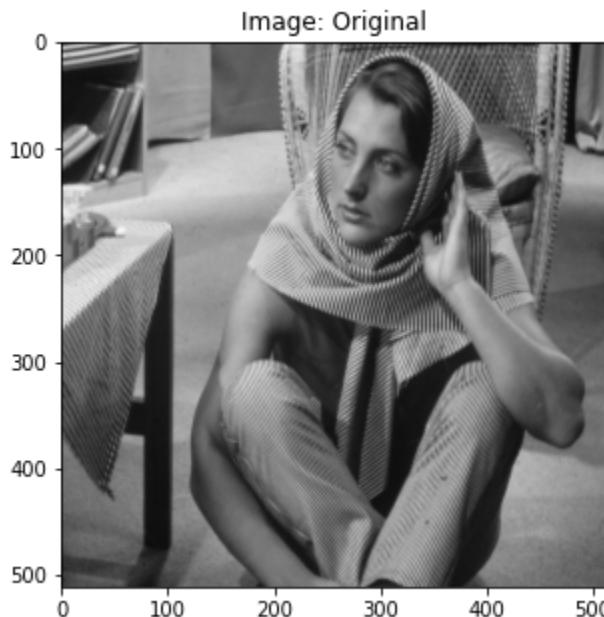
plot[0].imshow(magnitude_no_shift_no_log, cmap='gray')
plot[0].set_title("Magnitude: " + name + " (No Shift - No Log)")

plot[1].imshow(magnitude_no_shift_yes_log, cmap='gray')
plot[1].set_title("Magnitude: " + name + " (No Shift - Log)")

plot[2].imshow(magnitude_yes_shift_no_log, cmap='gray')
plot[2].set_title("Magnitude: " + name + " (Shift - No Log)")

plot[3].imshow(magnitude_yes_shift_yes_log, cmap='gray')
plot[3].set_title("Magnitude: " + name + " (Shift - Log)")
```

Out[]: Text(0.5, 1.0, 'Magnitude: Barbara (Shift - Log)')



In []: # F16

```
img = img_f16.copy()
name = "F16"

magnitude_no_shift_no_log = get_image_magnitude(img, False, False)
magnitude_no_shift_yes_log = get_image_magnitude(img, False, True)
magnitude_yes_shift_no_log = get_image_magnitude(img, True, False)
magnitude_yes_shift_yes_log = get_image_magnitude(img, True, True)

fig, plot = plt.subplots(1, 1, figsize = (5, 5))

plot.imshow(img, cmap='gray')
plot.set_title("Image: Original")

fig, plot = plt.subplots(1, 4, figsize = (20, 5))

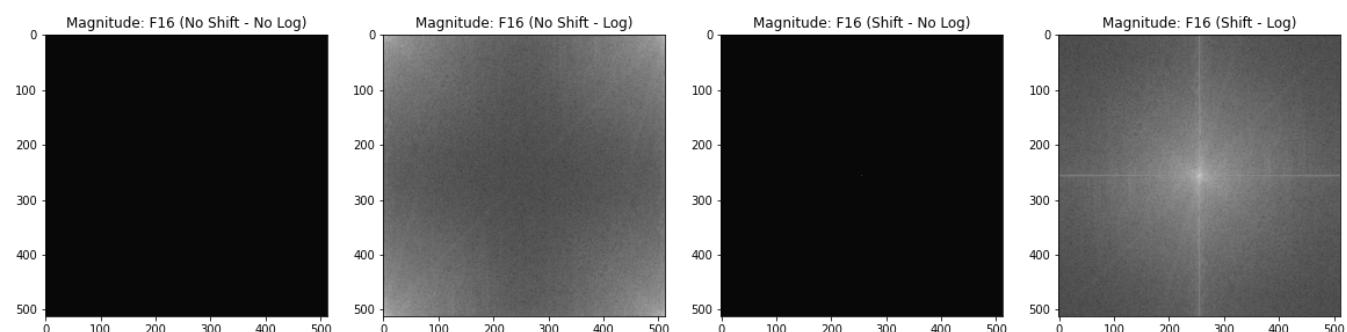
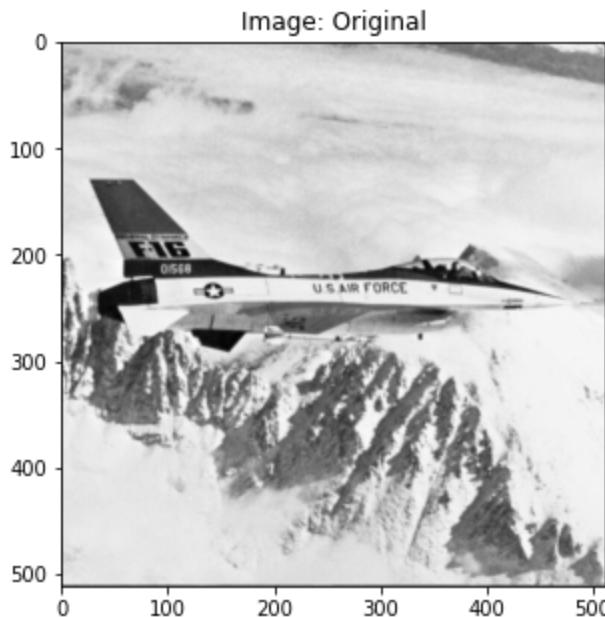
plot[0].imshow(magnitude_no_shift_no_log, cmap='gray')
plot[0].set_title("Magnitude: " + name + " (No Shift - No Log)")

plot[1].imshow(magnitude_no_shift_yes_log, cmap='gray')
plot[1].set_title("Magnitude: " + name + " (No Shift - Log)")

plot[2].imshow(magnitude_yes_shift_no_log, cmap='gray')
plot[2].set_title("Magnitude: " + name + " (Shift - No Log)")

plot[3].imshow(magnitude_yes_shift_yes_log, cmap='gray')
plot[3].set_title("Magnitude: " + name + " (Shift - Log)")
```

Out[]: Text(0.5, 1.0, 'Magnitude: F16 (Shift - Log)')



In []: # Baboon

```
img = img_baboon.copy()
name = "Baboon"

magnitude_no_shift_no_log = get_image_magnitude(img, False, False)
magnitude_no_shift_yes_log = get_image_magnitude(img, False, True)
magnitude_yes_shift_no_log = get_image_magnitude(img, True, False)
magnitude_yes_shift_yes_log = get_image_magnitude(img, True, True)

fig, plot = plt.subplots(1, 1, figsize = (5, 5))

plot.imshow(img, cmap='gray')
plot.set_title("Image: Original")

fig, plot = plt.subplots(1, 4, figsize = (20, 5))

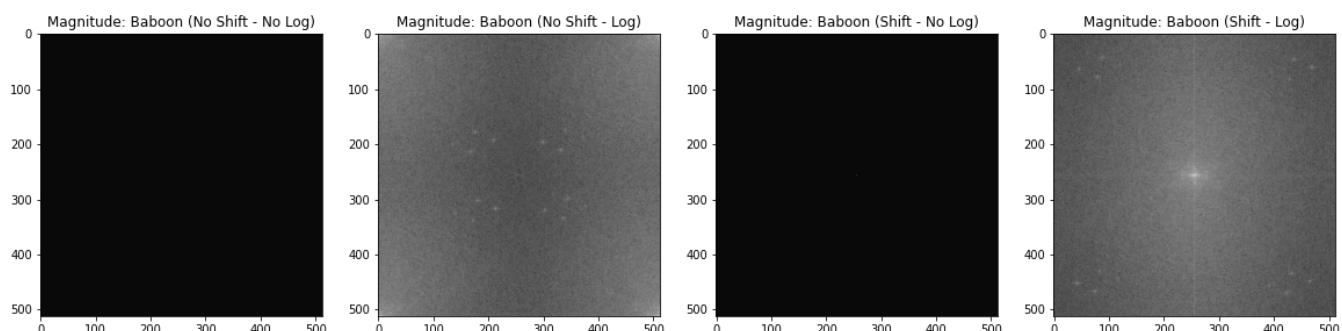
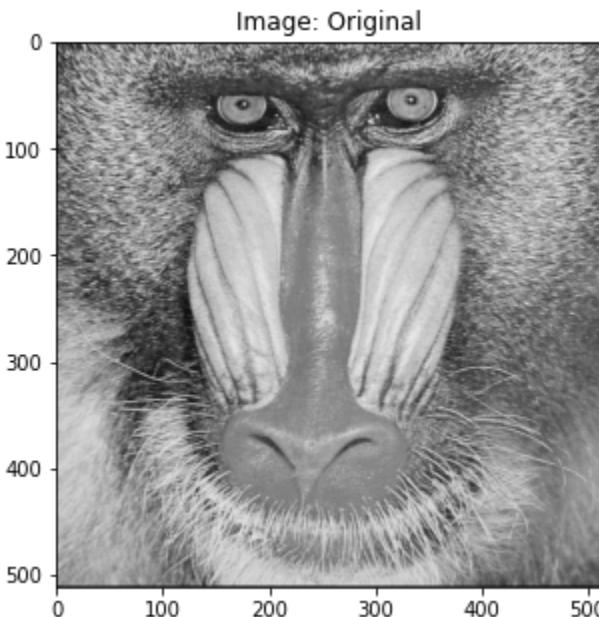
plot[0].imshow(magnitude_no_shift_no_log, cmap='gray')
plot[0].set_title("Magnitude: " + name + " (No Shift - No Log)")

plot[1].imshow(magnitude_no_shift_yes_log, cmap='gray')
plot[1].set_title("Magnitude: " + name + " (No Shift - Log)")

plot[2].imshow(magnitude_yes_shift_no_log, cmap='gray')
plot[2].set_title("Magnitude: " + name + " (Shift - No Log)")

plot[3].imshow(magnitude_yes_shift_yes_log, cmap='gray')
plot[3].set_title("Magnitude: " + name + " (Shift - Log)")
```

Out[]: Text(0.5, 1.0, 'Magnitude: Baboon (Shift - Log)')



تمرين ۴ سوال ۱-۲-۴

چکیده

در این سوال به بررسی سایز DFT با توجه به سایز های تصویر و فیلتر می پردازیم. همچنین در ادامه برابری نتایج کانولوشن و ضرب در محیط فوریه را بررسی می کنیم.

مقدمه

برای بدست آوردن سایز DFT باید به سایز تصویر و فیلتر توجه داشته و نتیجه Multiply آن ها را به عنوان سایز DFT در نظر بگیریم و با اعمال Inverse DFT به نتیجه مورد نیاز می رسیم. همچنین برای برابری نتایج دو بخش، می دانیم که کانولوشن در فضای Spatial با ضرب در فضای فوریه معادل هستند اما آیا این معادل بودن باعث برابری کامل نتیجه هر دو با یکدیگر می شود؟ برای پاسخ باید بگوییم که در صورتی که سایز فیلتر ها طوری تعریف شود که با یکدیگر هیچ Overlap نداشته باشند، می توان گفت که پاسخ ها برابر هستند.

شرح نتایج

برای پاسخ بخش اول می گوییم که با سایز تصویر 256 در 256 و فیلتر 11 در 11 باید سایز DFT به اندازه 266 در 266 تعریف شود.

در بخش دوم هم برای عدم Overlap و برابری نتایج می گوییم که تا وقتی که m و n کمتر از 11 باشد، نتایج برابر بوده و پاسخ ها پکسان هستند.

تمرین ۴ سوال ۲-۲-۴

چکیده

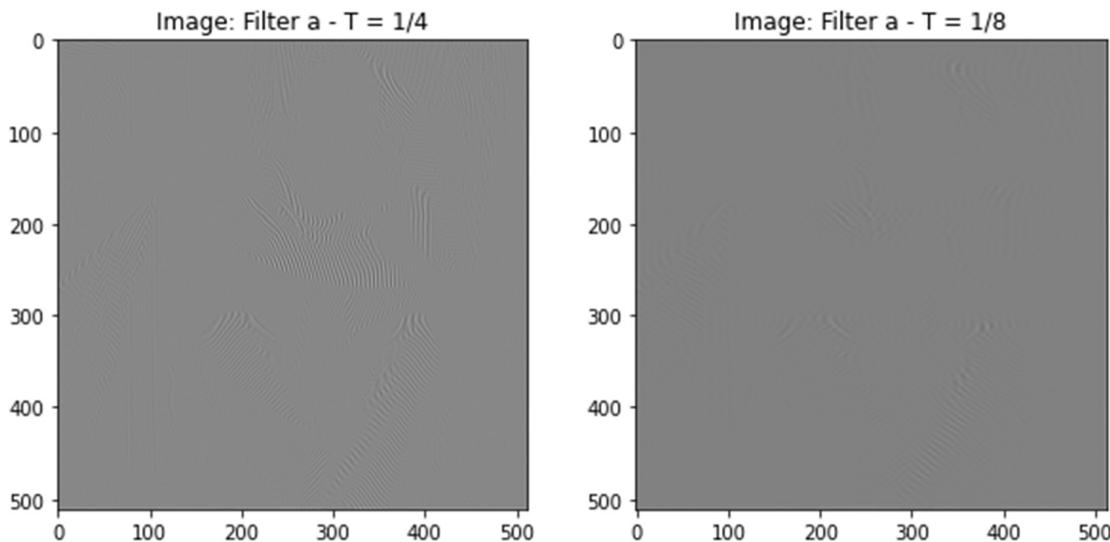
در این تمرین فیلتری را که با حذف برخی قسمت های تصویر در حوزه فوریه بدست آمده است را طراحی می کنیم.

مقدمه

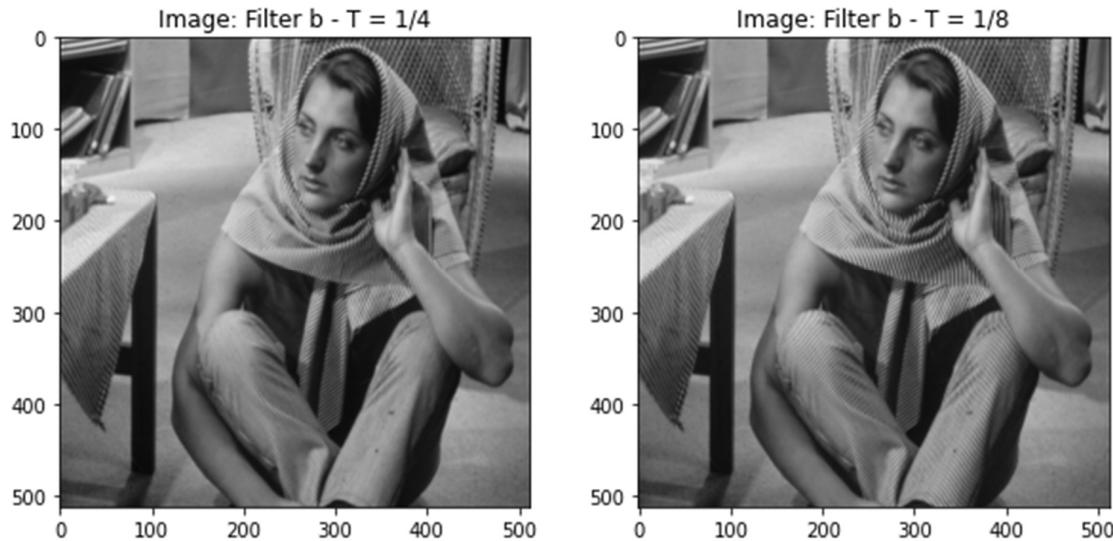
در این تمرین مانند تمرین اول این بخش، به اعمال فیلتر در حوزه فرکانس می پردازیم ولی در این مرحله این کار را با تغییر دستی مقادیر DFT اعمال می کنیم. به طور کلی فیلتر های مراحل قبل نیز با صفر کردن برخی نقاط در DFT آن ها را اعمال می کردند و در این تمرین این کار را بدون فیلتر آماده و به صورت دستی انجام می دهیم.
دو فیلتر a و b شرح داده شده اند که در هر دو باید بخشی از تصویر را در حوزه فرکانس سیاه کنیم.
در فیلتر a، باید یک مربع از وسط تصویر حوزه فوریه برداشته و بقیه قسمت ها را به فرم قبلی حفظ کنیم. در فیلتر b نیز باید تنها مربع وسط تصویر را نگه داشته و اطراف آن را حذف کنیم.
برای هر کدام از این فیلتر ها، پس از بردن تصویر به حوزه فوریه و شیفت دادن آن، فیلتر ها را با صفر کردن مقادیر اعمال کرده و باز هم مانند مراحل قبل، با اعمال IDFT به حوزه مکان باز می گردانیم. هر فیلتر ها را با $T = 1/4$ و $T = 1/8$ بررسی می کنیم.

شرح نتایج

در قسمت نتایج فیلتر های اعمال شده a و b را می بینیم:



با توجه به اینکه اطلاعات اصلی تصویر و جمع همه پیکسل های آن در مرکز DFT شیفت خورده قرار دارد، با حذف این قسمت و صفر کردن آن، اطلاعات تصویر را از دست می دهیم.



اما به صورت برعکس و با اعمال فیلتر بر همه نقاط بجز وسط تصویر، اطلاعات اصلی آن را حفظ کرده و فرکانس ها را محدود می کنیم که این کار باعث حذف بخشی از Texture در تصویر شده است.

```
In [ ]: import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from google.colab import drive
```

```
In [ ]: drive.mount('/content/drive')
Mounted at /content/drive
```

```
In [ ]: img_gray = cv2.imread("/content/drive/MyDrive/Images/4/Barbara.bmp", 0)
```

```
In [ ]: def apply_filter(image, filter_type, T):
    R, C = image.shape
    R_padded = 2 * R
    C_padded = 2 * C

    left_value_row = int(T * R_padded)
    right_value_row = int((1 - T) * R_padded)
    left_value_column = int(T * C_padded)
    right_value_column = int((1 - T) * C_padded)

    image_padded = np.zeros((R_padded, C_padded))
    image_padded[0:R, 0:C] = image

    image_DFT = np.fft.fft2(image_padded)
    image_shift = np.fft.fftshift(image_DFT)

    image_filtered = image_shift.copy()

    if filter_type == "a":
        image_filtered[left_value_row:right_value_row, left_value_column:right_value_column]=0
    else:
        image_filtered[0:left_value_row, :]=0
        image_filtered[:, right_value_column:C_padded]=0
        image_filtered[right_value_row:R_padded, :]=0
        image_filtered[:, 0:left_value_column]=0

    image_filtered_inverse_shift = np.fft.ifftshift(image_filtered)
    image_filtered_IDFT = np.fft.ifft2(image_filtered_inverse_shift)
    image_filtered_real = image_filtered_IDFT.real
    image_filtered_result = image_filtered_real[0:R, 0:C]

    return image_filtered_result
```

```
In [ ]: filter_a_1_4 = apply_filter(img_gray, "a", 1/4)
filter_a_1_8 = apply_filter(img_gray, "a", 1/8)
filter_b_1_4 = apply_filter(img_gray, "b", 1/4)
filter_b_1_8 = apply_filter(img_gray, "b", 1/8)

fig, plot = plt.subplots(2, 2, figsize = (10, 10))

plot[0][0].imshow(filter_a_1_4, cmap='gray')
plot[0][0].set_title("Image: Filter a - T = 1/4")

plot[0][1].imshow(filter_a_1_8, cmap='gray')
plot[0][1].set_title("Image: Filter a - T = 1/8")

plot[1][0].imshow(filter_b_1_4, cmap='gray')
plot[1][0].set_title("Image: Filter b - T = 1/4")

plot[1][1].imshow(filter_b_1_8, cmap='gray')
plot[1][1].set_title("Image: Filter b - T = 1/8")
```

Out[]: Text(0.5, 1.0, 'Image: Filter b - T = 1/8')

