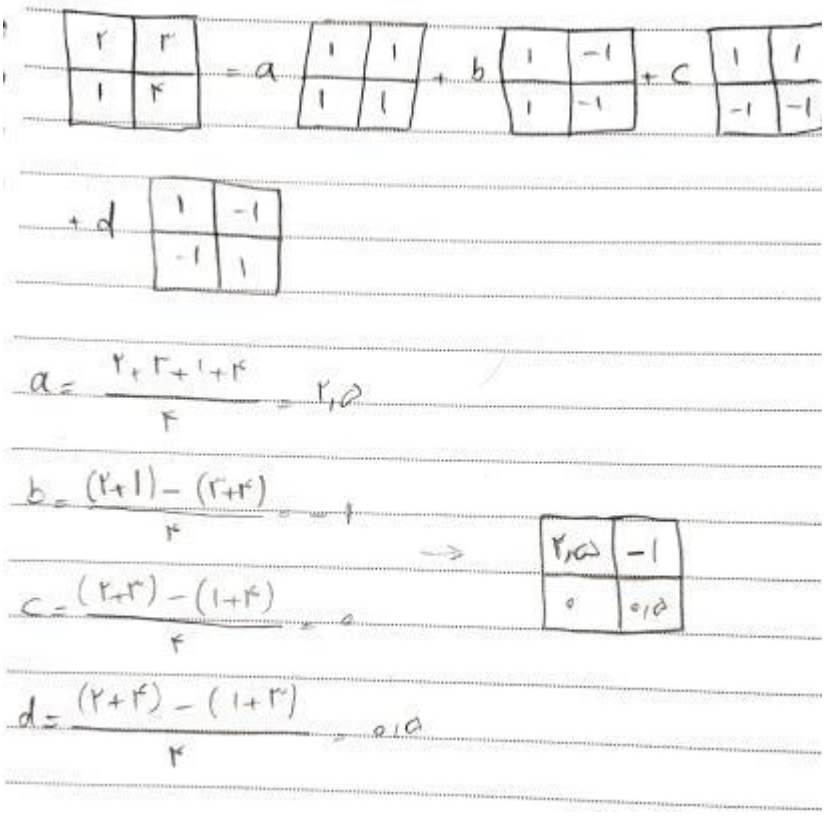
تمرین 4 بینایی کامپیوتر

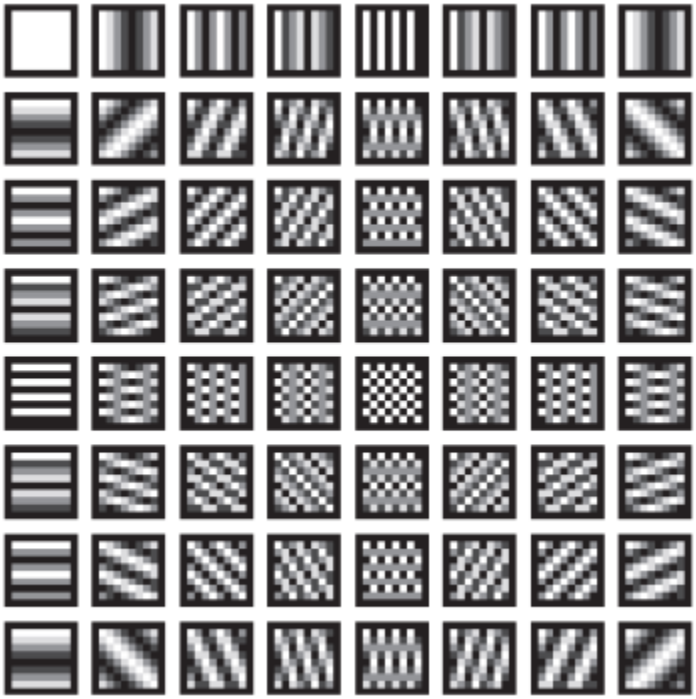
سینا اسکندری 97521054

.1



.2

الف)



طبق این تصویر فقط سطر بالا و چپ قابل مقدار دهی هستند و مقادیر وسط حاصل ضرب مقادیر بالا و چپ هستند پس تعداد مولفه آزاد 2n – 1 است.

ب)

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

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

با در نظر گرفتن این 2 فرمول و قرار دادن 0,0 ، قسمت سینوسی مقدار 1 می گیرد و حالت تناوبی ندارد و می توان نتیجه گرفت نقطه مبدا در تبدیل فوریه تصویر، میانگین روشنایی تصویر را نشان می دهد.

3.

الف)

```
zero_padding_image = np.pad(image, 1, constant_values=0) # add zero padding to image
rotated_kernel = np.rot90(kernel, k=2) # rotate kernel 180°
x, y = rotated_kernel.shape
for i in range(zero_padding_image.shape[0] - rotated_kernel.shape[0] + 1):
    for j in range(zero_padding_image.shape[1] - rotated_kernel.shape[1] + 1):
        result[i, j] = np.sum(np.multiply(zero_padding_image[i:i + x, j:j + y], rotated_kernel))
return result
```

با استفاده از تابع pad ابتدا به تصویر خود padding اضافه می کنیم. برای پیاده سازی کانولوشن کرنل را 180 درجه می چرخانیم و با استفاده از 2 حلقه روی تمام پیکسل ها و همسایه ها به اندازه سایز کرنل عملگر correlate که همان جمع عناصر ضرب 2 ماتریس است، اعمال می کنیم.

ب)

برای پیاده سازی کرنل میانگین گیر ماتریس واحد را تقسیم بر سایز آن می کنیم.

```
result = np.ones((size, size))
#########################
# Your code goes here. #
#########################
result /= (size*size)
return result
```

تصویر smooth شده:

پ)

برای پیاده سازی فیلتر میانه گیر مانند قسمت الف عمل می کنیم و تنها تفاوت این است که به جای کانولوشن از میانه ماتریس با سایز ورودی استفاده می کنیم.

```
result = np.zeros(image.shape)
########################
# Your code goes here. #
########################
zero_padding_image = np.pad(image, 1, constant_values=0) # add zero padding to image
for i in range(zero_padding_image.shape[0] - size + 1):
    for j in range(zero_padding_image.shape[1] - size + 1):
        result[i, j] = np.median(zero_padding_image[i:i+size, j:j+size])
return result
```

خروجی به دست آمده:

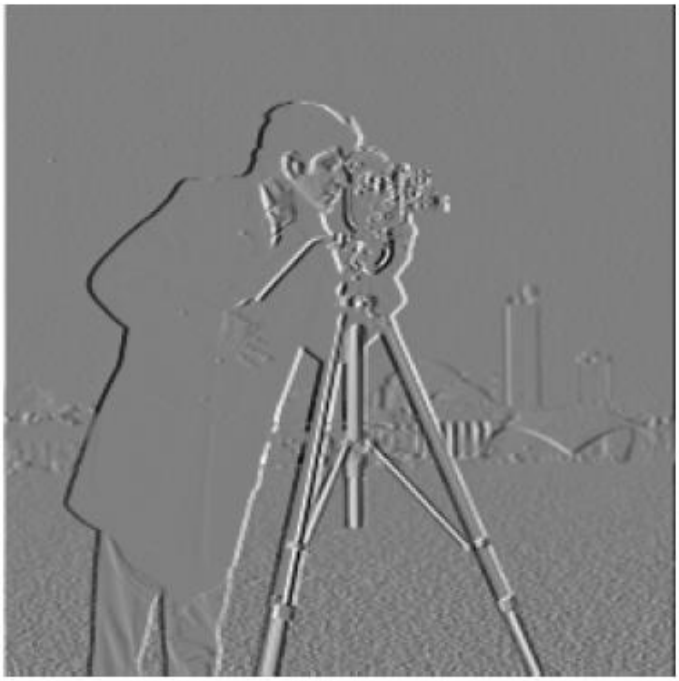
ت)

$$\frac{\partial f(x)}{\partial x} \approx \frac{f(x+1) - f(x-1)}{2}$$

با استفاده از این فرمول می توان به این نتیجه رسید که کرنل مشتق در راستای افقی به این صورت می باشد.

```
derivative_kernel = np.array([
    [0, 0, 0],
    [0.5, 0, -0.5],
    [0, 0, 0],
])
```

عکس مشتق گرفته شده در راستای افقی:



4.

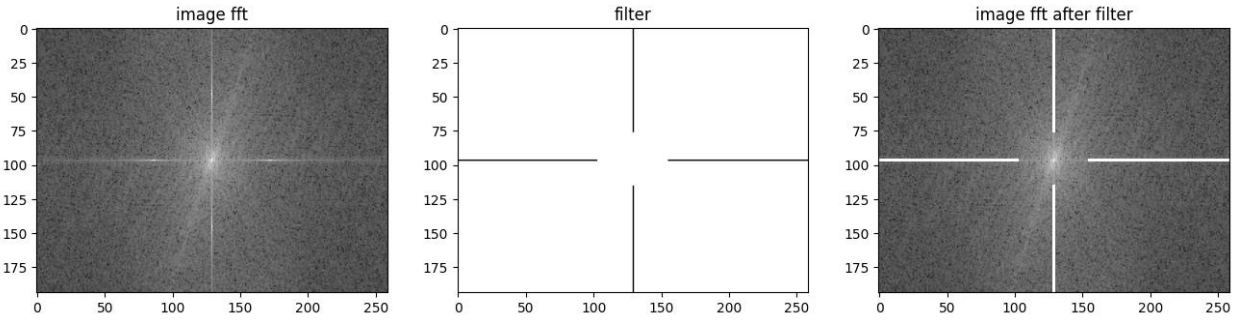
الف)

برای حذف نویز تصویر تابع denoise_image را به اینگونه کامل می شود.

```
########################
# Your code goes here. #
########################
fft = np.fft.fft2(image)
x, y = fft.shape
fft_shift = np.fft.fftshift(fft)
filter = np.ones(fft_shift.shape)
filter[int(0.5 * x)] = 0
filter[:,int(0.5 * y)] = 0
filter[int(0.4 * x):int(0.6 * x),int(0.4 * y):int(0.6 * y)] = 1
fft2_shift = fft_shift * filter
fft_2 = np.fft.ifftshift(fft2_shift)
denoised = np.fft.ifft2(fft_2).real
plt.figure(figsize=(16, 16))
plt.subplot(131), plt.imshow(20*np.log(np.abs(fft_shift)), cmap = 'gray'), plt.title('image fft')
plt.subplot(132), plt.imshow(filter, cmap='gray'), plt.title('filter')
plt.subplot(133), plt.imshow(20*np.log(np.abs(fft2_shift)), cmap='gray'), plt.title('image fft after filter')
return denoised
```

ابتدا با استفاده از دستور fft2 تبدیل فوریه تصویر را بدست می آوریم و با استفاده از fftshift آن را شیفت می دهیم و بعد از آن باید فیلتری برای رفع نویز طراحی کنیم(جزئیات فیلتر جلوتر توضیح داده می شود) و آن را در تبدیل فوریه اعمال کنیم و در آخر نیز روی تبدیل فوریه جدید، تبدیل فوریه معکوس اعمال می کنیم.

تبدیل فوریه اولیه، فیلتر و تبدیل فوریه ثانویه را رسم می کنیم.



چون نویز متناوب داریم از این فیلتر استفاده می کنیم که شبیه به اسلاید 12 جلسه می باشد ولی تفاوتی که دارد این است که اینجا در 2 راستا نویز داریم.

همانطور که قابل مشاهده است نویز تصویر به طرز قابل توجهی کاهش یافته است.

ب)

```
PSNR between noisy image and original image =  8.122255096865844
PSNR between denoised image and original image =  31.567562064707225
```

با بررسی این مقدار میتوان فهمید که مقدار سیگنال به نویز بسیار افزایش یافته است و با دیدن تصویر رفع نویز می توان صحت این را متوجه شد.

پ)

این نویز جمع شونده است و مقادیر نویز به مقادیر اصلی تصویر اضافه شده اند و برای حذف آن، در واقع با استفاده از فیلتر منها کردیم.

منابع:

https://docs.opencv.org/4.6.0/de/dbc/tutorial_py_fourier_transform.html

https://scipy-lectures.org/intro/scipy/auto_examples/solutions/plot_fft_image_denoise.html