Assignment NO.3 Solutions

Digital Image Processing | Fall 1400 | Dr.Mohammadi

Teacher Assistant : Fatemeh Anvari

Student name : **Amin Fathi**

Student id : **400722102**

Problem 1.a

این مقاله قصد دارد با استفاده از median filter و wavelet ها نویز تصویر را کاهش داده و همزمان بیشترین مقدار دیتای ورودی حفظ شود ، بنابر نتایج به دست آمده از این مقاله استفاده از هر دو روش های گفته شده تاثیر به مراتب بهتری در حذف نویز نسبت به استفاده تکی از هر کدام از روش ها دارد. نگارنده در ادامه به تعریف انواع نویز پرداخته و سپس در یک دسته بندی کلی انواع نویز را در دو حالت Additive noise و Multiplicative nosie دسته بندی کرده است

Additive noise : $w(x, y) = s(x, y) + n(x, y)$
Multiplicative noise : $w(x, y) = s(x, y) \times n(x, y)$

نویز به کار رفته در این مقاله Additive Gaussian Noise AGN می باشد ،

نگارنده در ادامه ۴ روش را برای کاهش نویز به کار گرفته و نتایج حاصل از آن ها را با هم مقایسه می کند .

در روش اول از Discrete Wavelet Transform (DWT) استفاده کرده است که در آن برای تعیین مقدار نویز و threshold از فرمول زیر استفاده شده است :

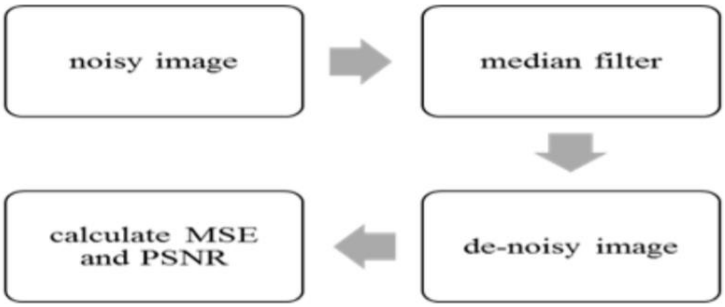
$$Th = \frac{\sigma^2}{\sigma_x}$$

$$\sigma^2 = [median(|x(i,j)|)/0.6745]^2$$

$$\sigma = \sqrt{\frac{\sum_{i=0}^{n-1}(xi-m)^2}{n}}$$

در این روش ابتدا با اعمال DWT روی تصاویر نویزی آن را به چهار باند فرعی A , H , V , D تقسیم کرده و روی هر کدام از باند ها مقدار Threshold را به دست می آوریم و چنانچه مقدار پیکسل در هر یکی از باند های فرعی از مقدار Threshold کمتر باشد ، مقدار آن پیکسل را برابر صفر قرار می دهیم و این کار را برای تمامی پیکسل ها انجام داده و سپس DWT معکوس گرفته تا تصویر بدون نویز به دست بیایید

نگارنده در روش دوم فقط از median استفاده کرده است که نحوه عملکرد این روش را در شکل زیر مشاهده می کنید .



در روش سوم هم از DWT و هم از median filter استفاده شده است و تنها تفاوت آن با روش چهارم در این است که در این روش ابتدا median filter اعمال می شود و سپس مقدار threshold محاسبه می شود اما در روش چهارم ابتدا threshold محاسبه می شود و سپس median filter اعمال می شود .

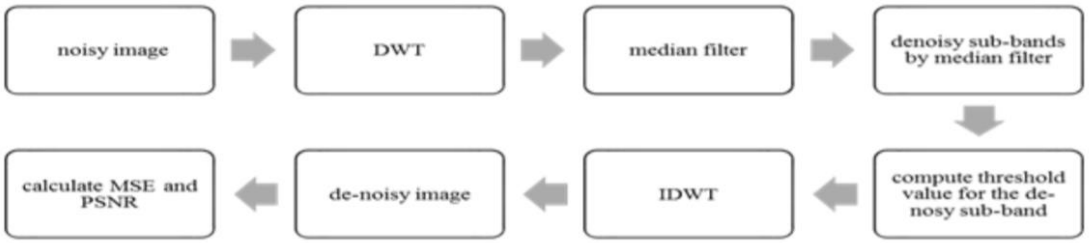


Figure 4: The block diagram for the third case, median filter before threshold.

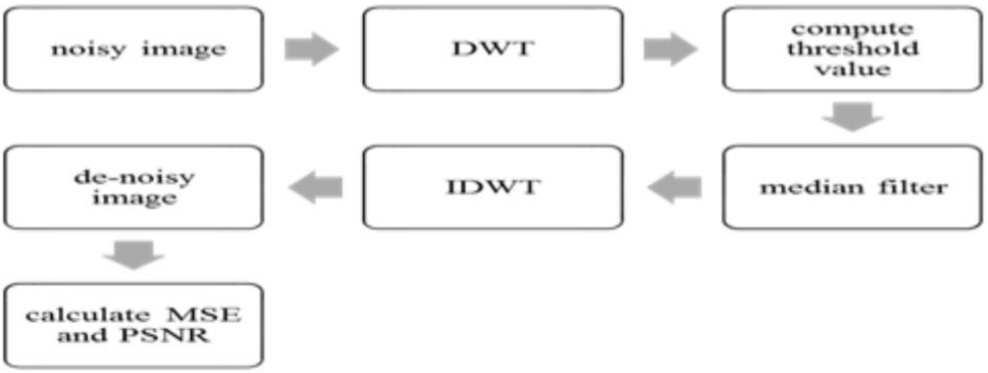


Figure 4: The block diagram for the fourth case, median filter after threshold.

با مقایسه شاخص های PSNR و MSE در باره چهار روش اعمال شده بر روی ۵ عکسی که در مقاله ضمیمه شده اند ، نتایج تست PSNR در جدول زیر آورده شده است .

Images	noise ratio	MF	DWT	MF befor thr.	MF after thr.
Lena	15	**26.5469**	25.1504	26.4146	26.2893
	20	**25.5292**	23.5935	25.1486	25.2302
	25	**24.6117**	22.3148	24.0477	24.1414
Barbara	15	22.3048	23.5818	**24.0892**	23.2405
	20	21.9372	22.3197	**23.1468**	22.6969
	25	21.5613	21.2449	**22.2978**	22.0692
Camera	15	24.6562	24.5340	**25.1232**	24.4323
	20	24.0361	22.9896	**24.2206**	23.8242
	25	**23.3887**	21.6859	23.2202	23.0355
Fruit	15	25.7715	24.7774	**25.9091**	25.3040
	20	**24.9298**	23.2299	24.8211	24.4766
	25	**24.1101**	22.1286	23.7675	23.6186
Buterfly	15	23.4540	24.0486	**24.7068**	23.7919
	20	22.8565	22.5758	**23.7164**	23.2238
	25	22.3289	21.3814	**22.8253**	22.5696

بنابه نتایج حاصله می توان گفت موفقیت عمل نویززدایی به ماهیت تصاویر بستگی دارد ، به طور مثال median filter بر روی تصاویر Camera , Lnea , Fruit تاثیر خوبی دارد ، ولی در کل استفاده از روش های سوم و چهارم غالبا نتیاج بهتری نسبت به دو روش اول به همراه دارد و در همه حالات هم استفاده از روش سوم نتایج بهتری نسبت به روش چهارم ثبت کرده است .

Problem 1.b

از کاربرد های wavelet میتوان به حذف نویز ، تشخیص چهره و فشرده سازی تصاویر اشاره کرد ، از الگوریتم های مورد بحث در این کاربرد میتوانEZW, SPITH,WDR , ASWDR را نام برد.

منبع :

http://leap.ee.iisc.ac.in/sriram/teaching/MLSP_16/refs/W24-Wavelets.pdf

Problem 2.a

$$\alpha_i = \frac{\langle w_i, z \rangle}{\langle w_i, w_i \rangle}$$

$$\alpha_1 = w = \frac{(6\times1)+(-2\times1)+(-3\times1)+(8\times1)}{4\times(1\times1)}$$

$$= \frac{9}{4}$$

$$\alpha_2 = x = \frac{(6\times1)+(-2\times-1)+(-3\times1)+(8\times-1)}{2\times(1\times1)+2\times(-1\times-1)}$$

$$= -\frac{3}{4}$$

$$\alpha_3 = y = \frac{(6\times1)+(-2\times1)+(-3\times-1)+(8\times-1)}{2\times(1\times1)+2\times(-1\times-1)}$$

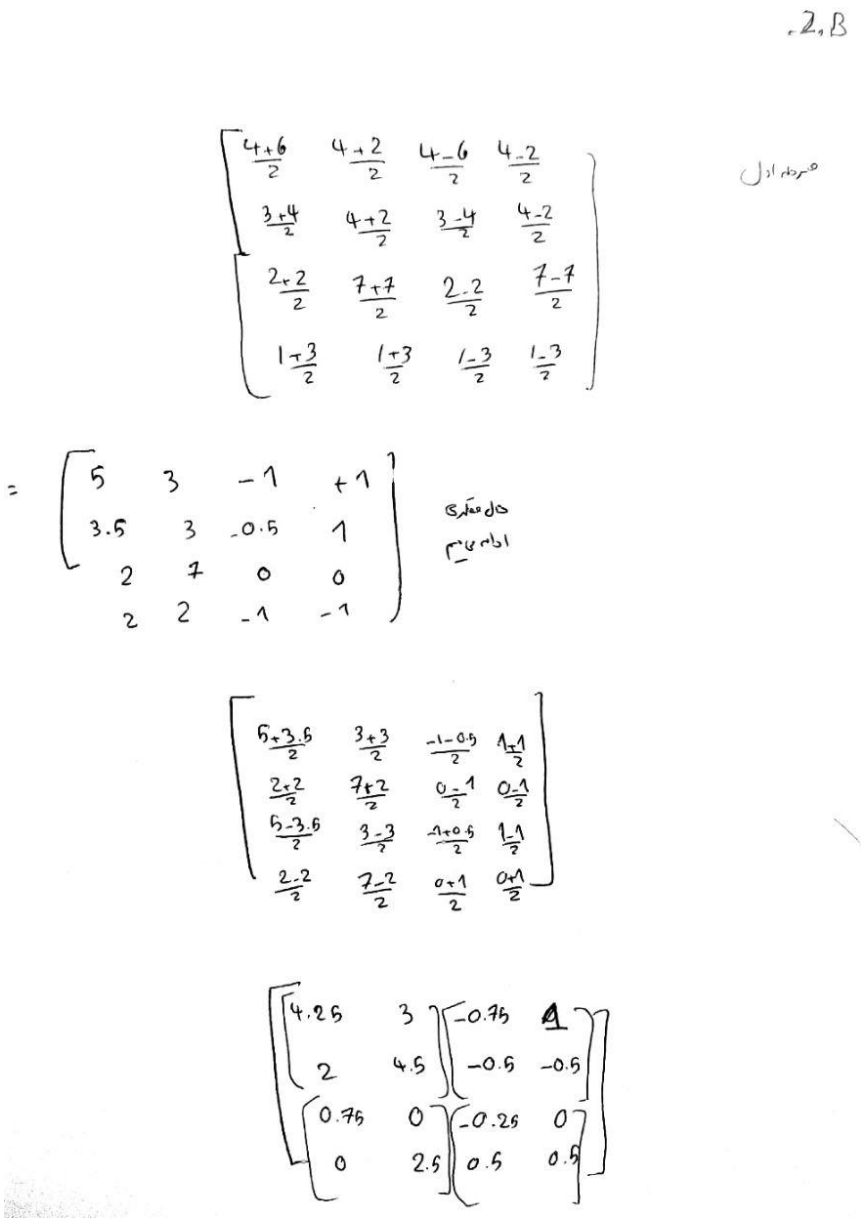
$$= -\frac{1}{4}$$

$$\alpha_4 = z = \frac{(6\times1)+(-2\times-1)+(-3\times-1)+(8\times1)}{2\times(1\times1)+2\times(-1\times-1)}$$

$$= \frac{19}{4}$$

ضریب a1 نشان دهنده میانگین تصاویر است و سایر ضرایب هم به ترتیب نشان دهنده وجود و بزرگی لبه های به ترتیب به صورت افقی ، عمودی و مورب است .

Problem 2.b



Problem 2.c

ابتدا ضرایب را به دست می آوریم

$$c_0(0) = \int_0^1 f(x)\,\varphi_{0,0}(x)\,dx$$

$$= \int_0^{0.5} 5\,dx + \int_{0.5}^1 (x-1)\,dx = 2.5 + \left(-\frac{1}{8}\right) = \frac{19}{8} = 2.375$$

$$d_0(0) = \int_0^1 f(x)\,\psi_{0,0}(x)\,dx$$

$$= \int_0^{0.5} f(x)\,dx - \int_{0.5}^1 f(x)\,dx$$

$$= \int_0^{0.5} 5\,dx - \int_{0.5}^1 (x-1)\,dx$$

$$= 2.5 + \frac{1}{8} = \frac{21}{8} = 2.625$$

$$d_1(0) = \int_0^1 f(x)\,\psi_{1,0}(x)\,dx = \int_0^{0.25} 5\sqrt{2}\,dx - \int_{0.25}^{0.5} 5\sqrt{2}\,dx$$

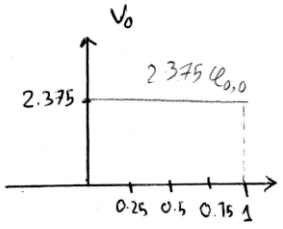
$$= 5\sqrt{2}\left[\left[\frac{1}{4} - 0\right] - \left[\frac{1}{2} - \frac{1}{4}\right]\right] = 0$$

$$d_1(1) = \int_0^1 f(x)\,\psi_{1,1}(x)\,dx = \int_{0.5}^{0.75} (x-1)\sqrt{2}\,dx - \int_{0.75}^1 (x-1)\sqrt{2}\,dx$$

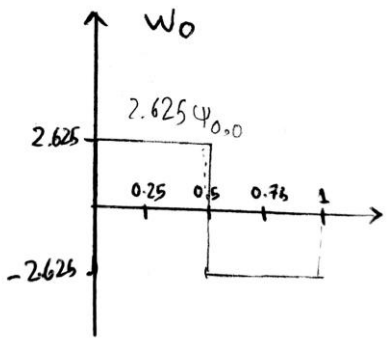
$$= \sqrt{2}\left[\left[\frac{9}{32} - \frac{3}{4} + \frac{1}{2} - \frac{1}{8}\right] - \left[\frac{1}{2} - 1 + \frac{3}{4} - \frac{9}{32}\right]\right]$$

$$= \sqrt{2}\left[-\frac{1}{16}\right] = -0.0884$$

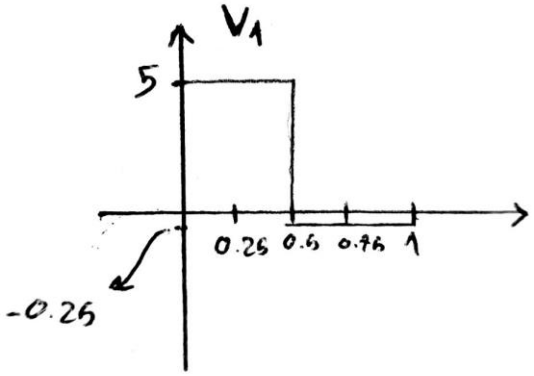
سپس V0 را ترسیم می کنیم (به کمک ضریب c0)



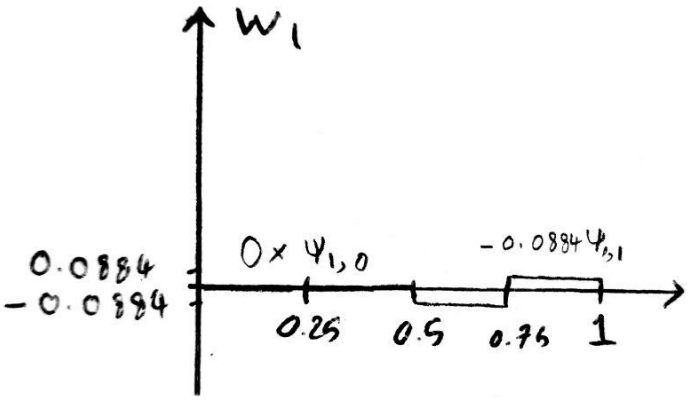
حال W0 را به کمک ضریب d0 ترسیم می کنیم .



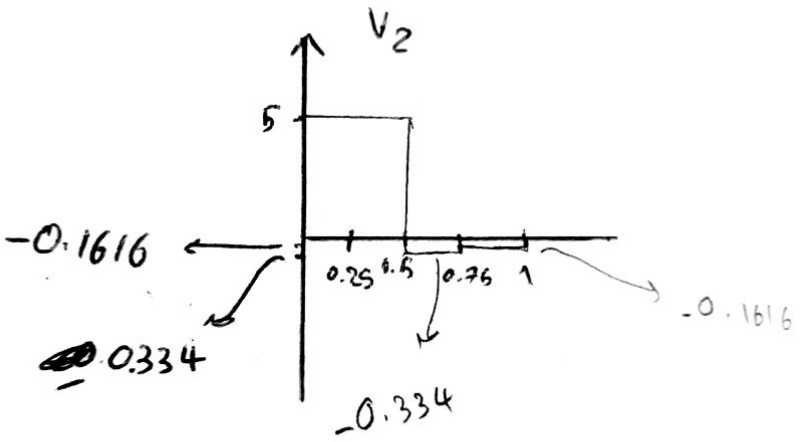
با جمع دو تابع W0 و V0 ، مقدار V1 را به دست می آوریم :



حال مقدار W1 را با استفاده از مقادیر ضریب d1 به دست می آوریم .



حال با جمع کردن دو تابع W1 و V1 ، مقدار V2 یا همان تقریب تابع به دست می آید .



Problem 3

DCT

ابتدا تصاویر را آپلود می کنیم .

```
from google.colab import files
uploaded = files.upload()
uploaded = files.upload()
```

سپس کتابخانه ها را ایمپورت می کنیم .

```
import numpy as np
import  cv2
from matplotlib import pyplot as plt
from mpl_toolkits.axes_grid1 import ImageGrid
from google.colab import files
from google.colab.patches import cv2_imshow
from scipy.fftpack import dct, idct
import matplotlib.pylab as plt
from math import log10, sqrt
import cv2
import numpy as np
```

بعد از خوانش عکس ، با توجه به اینکه کتابخانه مورد استفاده برای DCT تصویر با ابعاد زوج را نمیپذیرفت ، ابتدا تصویر را با اضافه کردن سه پیکسل به تصویری با ابعاد زوج تبدیل می کینم . لازم به ذکر است مقادیر سه پیکسل تازع اضافه شده را برابر با مقدار آخرین پیکسل قرار می دهیم . تصویر حاصل vis0 نام دارد .

```
#Read image
img = cv2.imread('3.jpg',cv2.IMREAD_GRAYSCALE)

rows, cols = img.shape
crow,ccol = rows//2 , cols//2
img2 = np.float32(img)
h = img.shape[0] +1
w = img.shape[1] +1
vis0 = np.zeros((h,w), np.float32)
vis0[:h-1, :w-1] = img
vis0[885 , 885] = vis0[884 , 884]
vis0[884 , 885] = vis0[884 , 884]
vis0[885 , 884] = vis0[884 , 884]
```

سپس مقدار DCT تصویر حاصل را حساب می کنیم که حاصل vis1 نام دارد

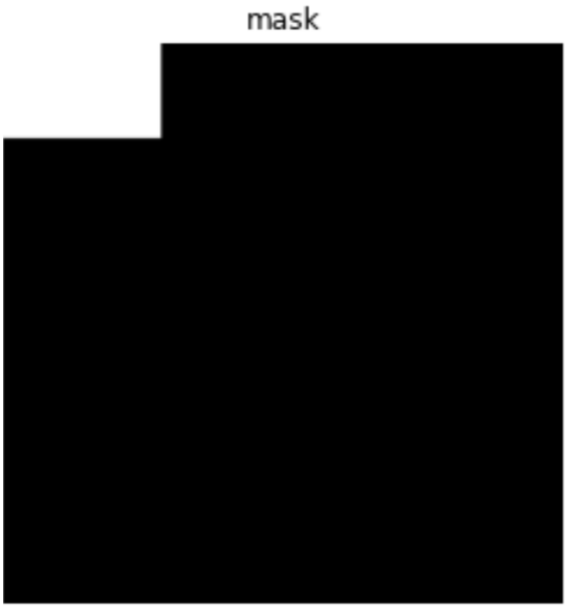
```
vis1 = cv2.dct(vis0)
```

که برابر است با شکل زیر

سپس ماسکی را ایجاد میکنیم تا مستطیلی در گوشه بالای چپ مقدار DCT را حفظ کرده و سایر مقادیر DCT را به جهت حذف نویز حذف کند .

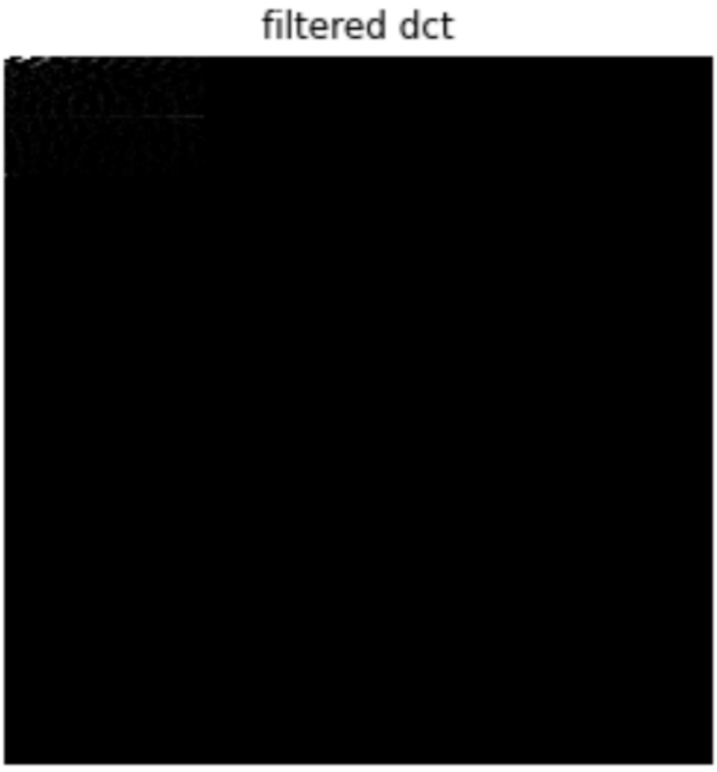
```
#mask

mask = np.zeros((rows+1,cols+1),np.uint8)
mask[0 :  150   , 0: 250 ] = 1
```

لازم به ذکر است مقدار ماسک به صورت تجربی و با مشاهده نتایج اعمال بر روی DCT به دست آمده است .



که حاصل ضرب ماسک حاصل در DCT برابر است با :

```
dct_filter = vis1 * mask
```



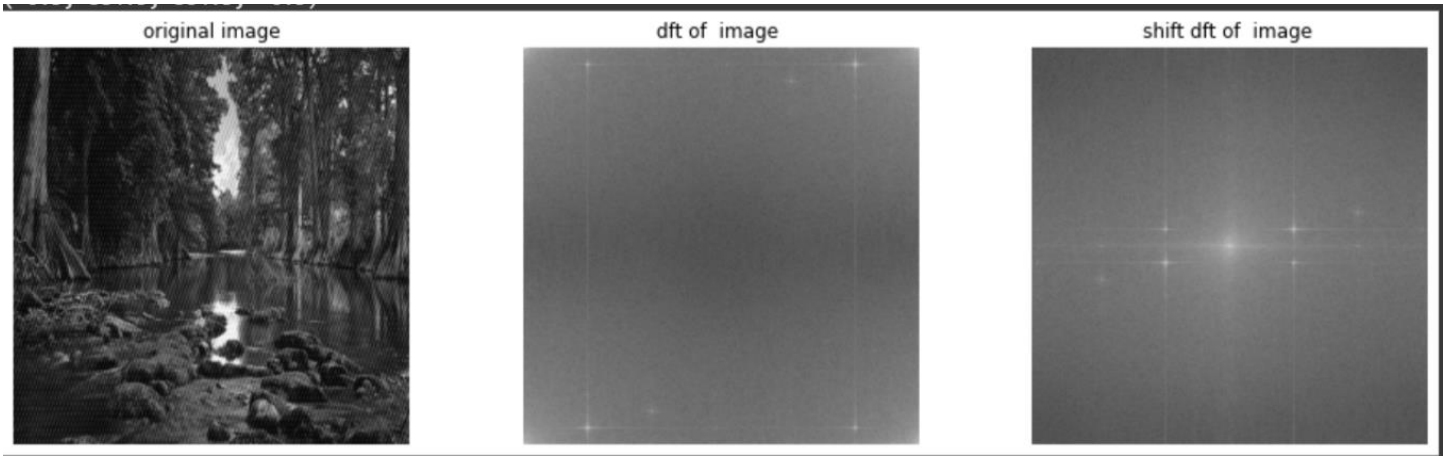
حال مقدار dct فیلتر شده در مرحله قبل را معکوس کرده تا عکس بدون نویز حاصل شود ، همچنین با توجه به لزوم مقایسه بین تصاویر در قسمت ب سوال ، ابعاد تصویر حاصله را به حالت قبل باز می گردانیم و در vis3 ذخیره میکنیم تا در آینده در مقایسه ها به کار بگیریم.

```
reconstructed = cv2.idct(dct_filter)
'''
برای مقایسه در قسمت ب سوال ، لازم است تا مجددا عکس نهایی پس از اعمال فیلتر را به اندازه اولیه آن نگاشت کنیم
'''
vis3 = np.zeros((h-1,w-1), np.float32)
vis3[:h-1, :w-1] = reconstructed[:h-1 , :w-1]
```

DFT

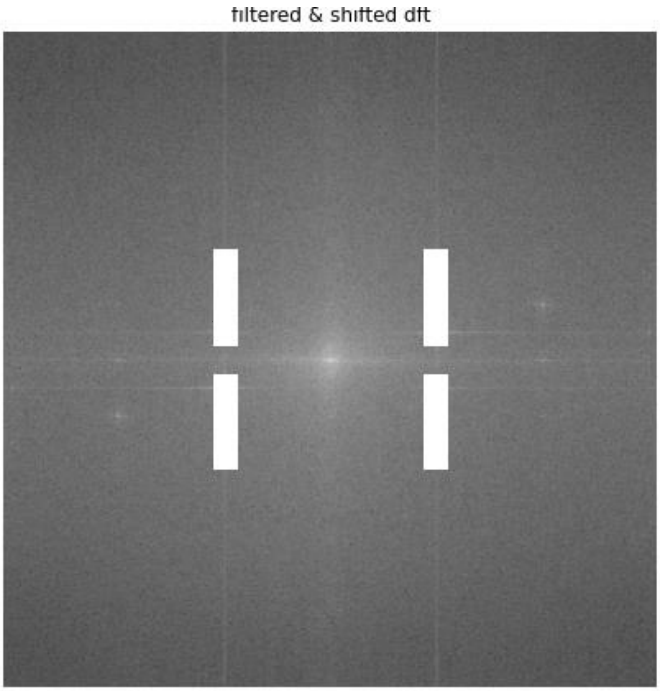
ابتدا مقدار DFT و DFT شیفت یافته تصویر را محاسبه کرده و نمایش می دهیم .

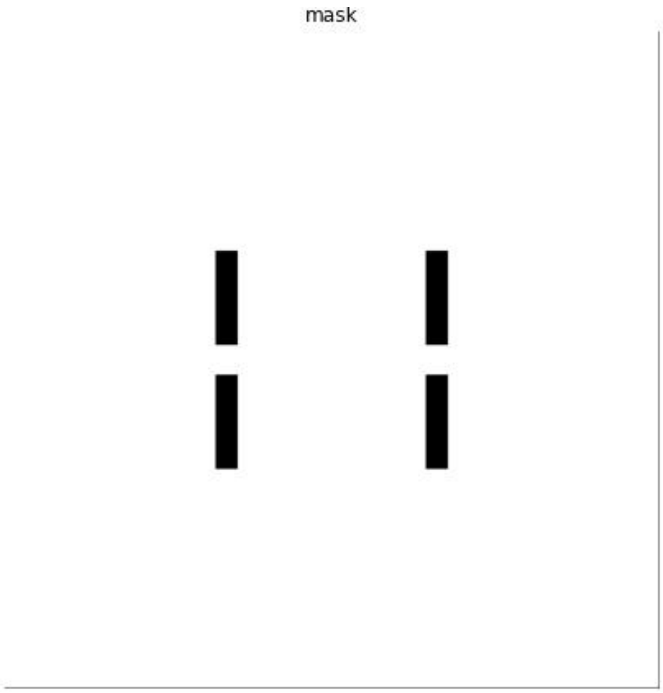
```
dft = cv2.dft(np.float32(img),flags = cv2.DFT_COMPLEX_OUTPUT)

dft_shift = np.fft.fftshift(dft)

magnitude_spectrum = np.log(cv2.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))
magnitude_spectrum2 = np.log(cv2.magnitude(dft[:,:,0],dft[:,:,1]))
fig = plt.figure(figsize=(16,16))
grid = ImageGrid(fig, 111,
                 nrows_ncols=(1, 3),
                 axes_pad=1,
                 )
grid[0].imshow(img,cmap='gray',vmin=0,vmax=255)
grid[0].set_title('original image')
grid[0].axis('off')
grid[1].imshow(magnitude_spectrum2, cmap='gray')
grid[1].set_title('dft of  image ')
grid[1].axis('off')
grid[2].imshow(magnitude_spectrum, cmap='gray')
grid[2].set_title('shift dft of  image ')
grid[2].axis('off')
```



حال ابتدا ماسکی طراحی میکنیم که بتواند ۴ نقطه نویز دار به شکل به علاوه را در DFT شیفت یافته فیلتر کند ، که مختصات و شکل ماسک حاصل به شکل زیر است .

```
rows, cols = img.shape
crow,ccol = rows//2 , cols//2
a = rows//2
b = cols //6 - 10
c = rows // 6
d = cols // 2
# create a mask first, center square is 1, remaining all zeros
mask2 = np.zeros((rows,cols,2),np.uint8)
mask2[crow-a :crow+a, ccol- b+10:ccol +b - 10] = 1
mask2[crow + c : crow +a , ccol -d  : ccol + d ] =1
mask2[crow - a : crow -c   , ccol -d  : ccol + d ] =1
mask2[crow - 20 : crow +20   , ccol -d  : ccol + d ] =1
mask2[crow - a : crow + a , ccol -d  : ccol - b - 20] =1
mask2[crow - a : crow + a , ccol +b + 20  : ccol + d] =1
```

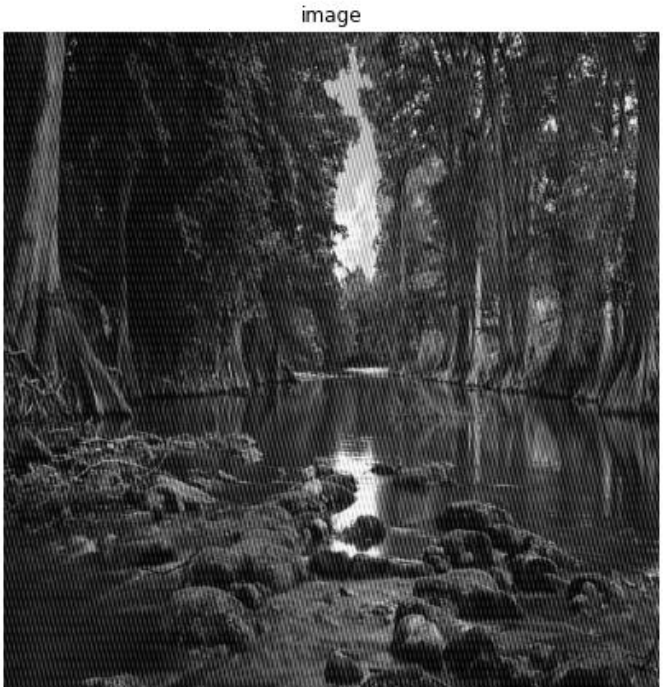


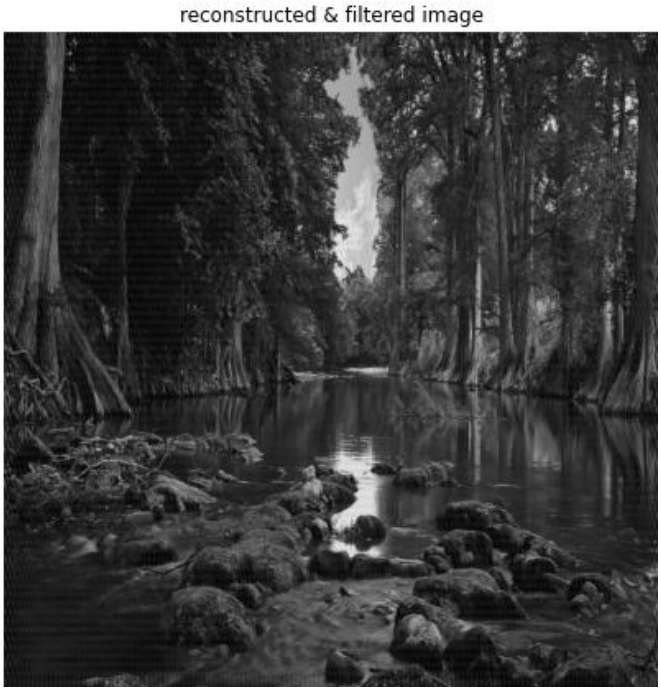


سپس ، با اعمال معکوس شیفت و معکوس DFT تصویر نهایی را به دست می آوریم

```
# apply mask
fshift = dft_shift*mask2
'''
حال معکوس شیفت یافته تصویر حاصل را به دست می آوریم
'''
f_ishift = np.fft.ifftshift(fshift)
'''
حال معکوس دی اف تی  تصویر حاصل را به دست می آوریم
'''
img_back = cv2.idft(f_ishift)

f_ishift2= np.log(cv2.magnitude(fshift[:,:,0],fshift[:,:,1]))

img_back = cv2.magnitude(img_back[:,:,0],img_back[:,:,1])
```

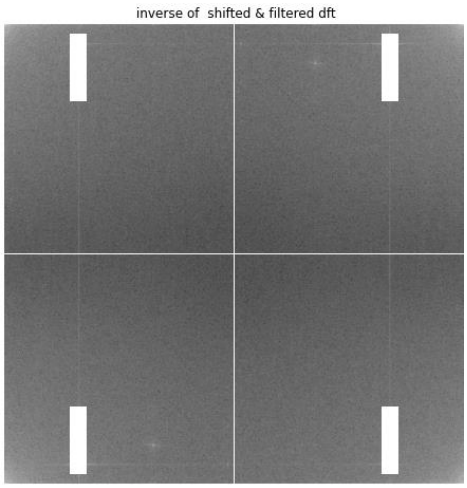
که نتیجه حاصل را در شکل زیر مشاهده می کنید .





در این مرحله معکوسِ DFT شیفت یافته و فیلتر شده را به دست می آوریم .

```
f_ishift5 = np.fft.ifftshift(f_ishift2)
fig = plt.figure(figsize=(8,8))
grid = ImageGrid(fig, 111,
                 nrows_ncols=(1, 1),
                 axes_pad=1,
                 )
grid[0].imshow(f_ishift5 , cmap = 'gray')
grid[0].set_title(' inverse of  shifted & filtered dft')
grid[0].axis('off')
```



Problem 3.b

مقدار شاخص PSNR را همانند تصویر زیر به دست می آوریم

```
def PSNR(original, compressed):
  mse = np.mean((original - compressed) ** 2)
  if(mse == 0): # MSE is zero means no noise is present in the signal .
        # Therefore PSNR have no importance.
    return 100
  max_pixel = 255.0
  psnr = 20 * log10(max_pixel / sqrt(mse))
  return psnr

print('DFT' , PSNR( imgreal,img_back ))
print('DCT' , PSNR(imgreal ,vis3 ))
```

```
DFT -106.45303427194285
DCT 23.43655820704397
```

لازم به ذکر است ، مقدار منفی در DFT نشان دهنده آن است که در تصویر جدید ، مقدار نویز از مقدار سیگنال بیشتر می باشد ، و این بحث به صورت شهودی و با مقایسه تصاویر به دست آمده از DCT و DFT قابل تطبیق است .

مقایسه شاخص MSE هم نشان می دهد تصویر به دست آمده از DCT عملکرد بهتری دارد در حذف نویز ها دارد اما تصویر تار تری ارایه داده است .(هر چه شاخص به ۰ نزدیک تر باشد بهتر است)

```
def MSE(orginal , compressed):
    mse = np.mean((orginal - compressed) ** 2)
    return mse

print('DFT', MSE(img,img_back))

print('DCT', MSE(img,vis3))
```

```
DFT 2873318600000000.0
DCT 1053.1588
```

همچنان با مقایسه مقدار شاخص SSIM هم میتوان مشاهده کرد که DCT عملکرد بهتری دارد (هر چه SSIM به عدد۱ نزدیک تر باشد عملکرد بهتری دارد و هر چه به صفر نزدیک تر باشد ، فیلترینگ ضعیفی اعمال شده است)

```
from skimage.metrics import structural_similarity as ssim
def compare(imageA, imageB):
   s = ssim(imageA, imageB)
   return s

print('DFT' ,compare(imgreal , img_back) )
print('DCT' ,compare(imgreal , vis3) )
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3:
  This is separate from the ipykernel package so we can avoid d
DFT 4.888342133939366e-12
DCT 0.5831845413357537
```

همچنین مشاهده شهودی نتایج هم مشخص میکند که هر چند تصویر حاصل از فیلتر DFT وضوح بالاتری دارد اما درای نویز بیشتری نسبت به تصویر DCT می باشد .

لازم به ذکر است ، با توجه به عدم مشاهده ارور مربوط به لزوم زوج بودن ابعاد ورودی برای تابع DCT ، ارور مشاهده شده در ادامه ضمیمه می شود.

```
grid[2].axis('off')
```

```
---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
<ipython-input-45-eae77e9475f8> in <module>()
     15 crow,ccol = rows//2 , cols//2
     16 img4 = np.float32(img2)
---> 17 c = cv2.dct(img4)
     18 # print(cv2.dct(img2))
     19 # c = c*255/c.max()  # scale between 0 and 255

error: OpenCV(4.1.2) /io/opencv/modules/core/src/dxt.cpp:4149: error: (-213:The function/feature is not implemented) Odd-size DCT's are not implemented in function 'apply'

SEARCH STACK OVERFLOW
```