به نام خدا

تکلیف سری سوم

مرضیه امیری

دانشجو ارشد هوش مصنوعی

درس تصویر پردازی رقمی

بسمه تعالی

تمرین سری ۳

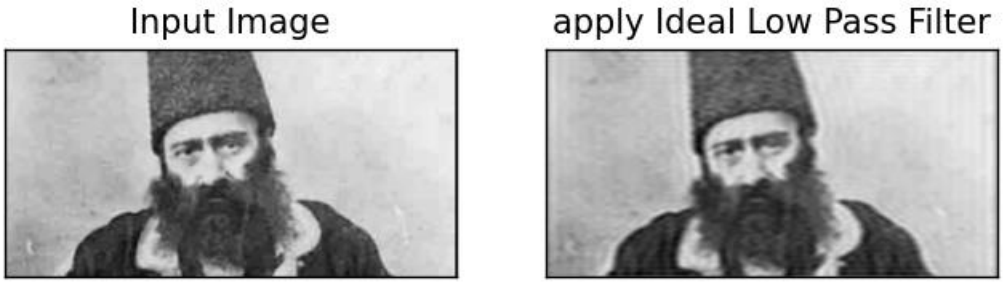
۱) یک تابع در پایتون بنویسید که فیلتر ایده آل پایین گذر را بر روی تصویر ورودی اعمال کند. این تابع ۲ پارامتر به عنوان ورودی دریافت کند. پارامتر اول، im (یک تصویر با فرمت uint8) و پارامتر دوم D0 که فرکانس cutoff می باشد.

الف) این تابع را برای یک تصویر اجرا کنید و نتیجه را چاپ نمایید. تابع را با فرکانسهای cutoff مختلف اجرا کنید.

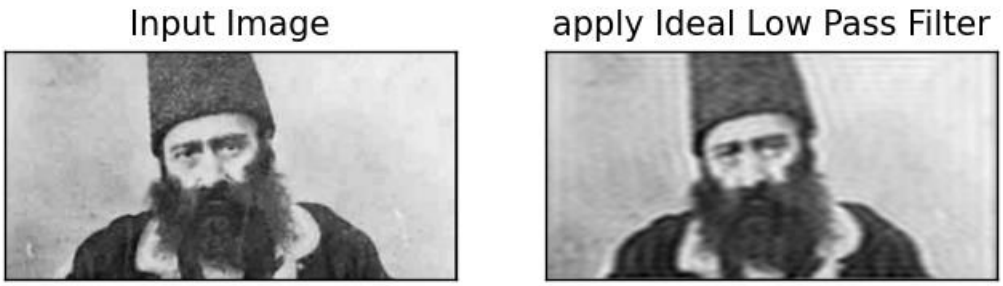
```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt


def Ideal_Lowpass_Filter(img,d):
    dft = cv.dft(np.float32(img),flags = cv.DFT_COMPLEX_OUTPUT)
    dft_shift = np.fft.fftshift(dft)
    rows, cols = img.shape
    crow,ccol = rows//2 , cols//2
    # create a mask first, center square is 1, remaining all zeros
    mask = np.zeros((rows,cols,2),np.uint8)
    mask[crow-d:crow+d, ccol-d:ccol+d] = 1
    # apply mask and inverse DFT
    fshift = dft_shift*mask
    f_ishift = np.fft.ifftshift(fshift)
    img_back = cv.idft(f_ishift)
    img_back = cv.magnitude(img_back[:,:,0],img_back[:,:,1])
    plt.subplot(121),plt.imshow(img, cmap = 'gray')
    plt.title('Input Image'), plt.xticks([]), plt.yticks([])
    plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
    plt.title('apply Ideal Low Pass Filter'), plt.xticks([]), plt.yticks([])
    plt.show()

img = cv.imread('3.jpg', 0)
assert img is not None, "file could not be read, check with os.path.exists()"
d = 40
Ideal_Lowpass_Filter(img,d)
d = 30
Ideal_Lowpass_Filter(img,d)
d = 20
Ideal_Lowpass_Filter(img,d)

d = 10
Ideal_Lowpass_Filter(img,d)
```

خروجی :

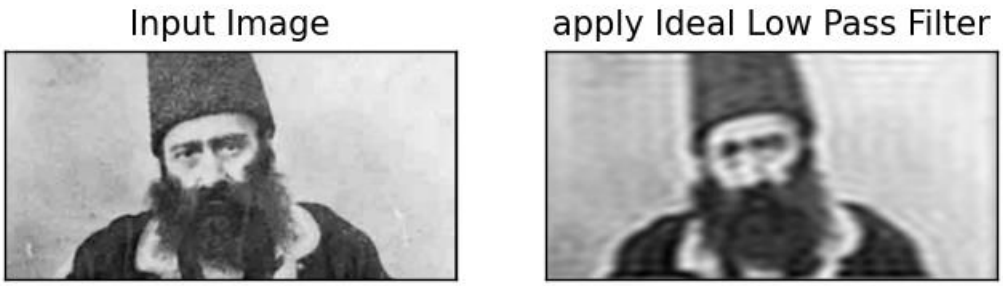
Cutoff = 40



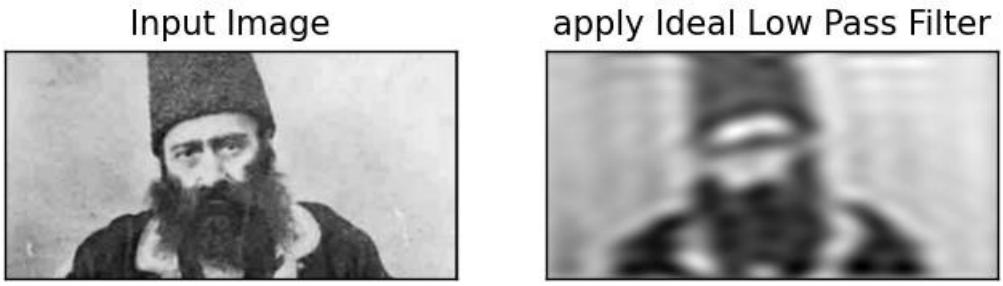
Cutoff = 30



Cutoff = 20



Cutoff = 10



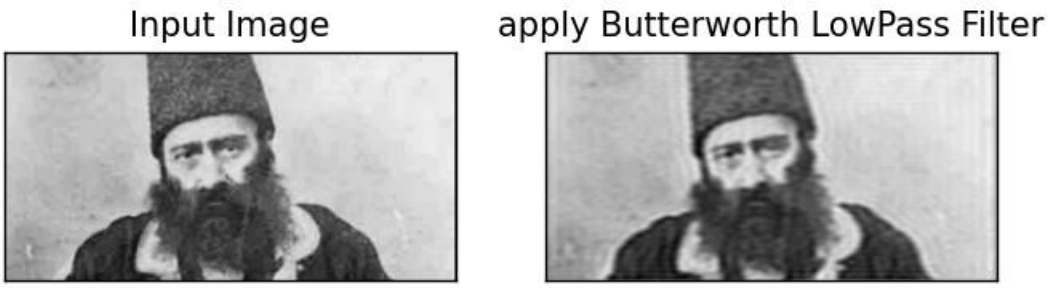
ب) یک تابع دیگر نیز برای یک فیلتر پایین گذر باتروورث با مقدار n=1 بنویسید و تابع را بر روی همان تصویر با فرکانسهای cutoff مختلف تست کنید.

```python
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt


def Butterworth_lowpass_filter(img,d):
    dft = cv.dft(np.float32(img),flags = cv.DFT_COMPLEX_OUTPUT)
    dft_shift = np.fft.fftshift(dft)
    rows, cols = img.shape
    n = 1
    # create a mask
    mask = np.zeros((rows,cols,2),np.uint8)
    for i in range(rows):
        for j in range(cols):
            distance = np.sqrt((i - rows//2) ** 2 + (j - cols//2) ** 2)
            mask[i, j] = 1 / (1 + (distance//d ) ** (2 * 1))
    # apply mask and inverse DFT
    fshift = dft_shift*mask
    f_ishift = np.fft.ifftshift(fshift)
    img_back = cv.idft(f_ishift)
    img_back = cv.magnitude(img_back[:,:,0],img_back[:,:,1])
    plt.subplot(121),plt.imshow(img, cmap = 'gray')
    plt.title('Input Image'), plt.xticks([]), plt.yticks([])
    plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
    plt.title('apply Butterworth LowPass Filter'), plt.xticks([]), plt.yticks([])
    plt.show()
```

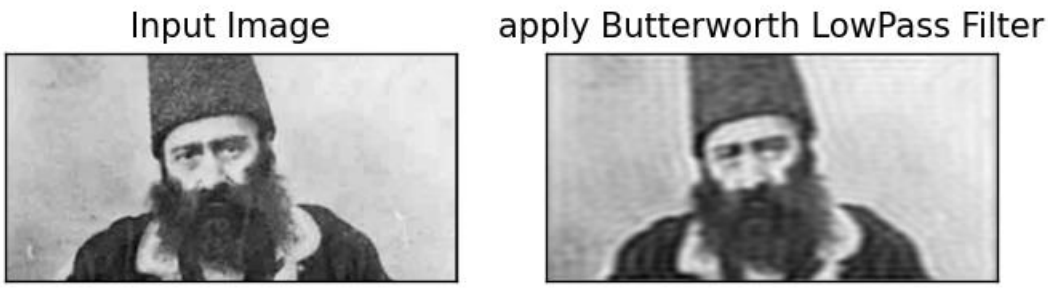
```
img = cv.imread('3.jpg', 0)

assert img is not None, "file could not be read, check with os.path.exists()"
d = 40
Butterworth_lowpass_filter(img,d)

d = 30
Butterworth_lowpass_filter(img,d)

d = 20
Butterworth_lowpass_filter(img,d)

d = 10
Butterworth_lowpass_filter(img,d)
```

خروجی :

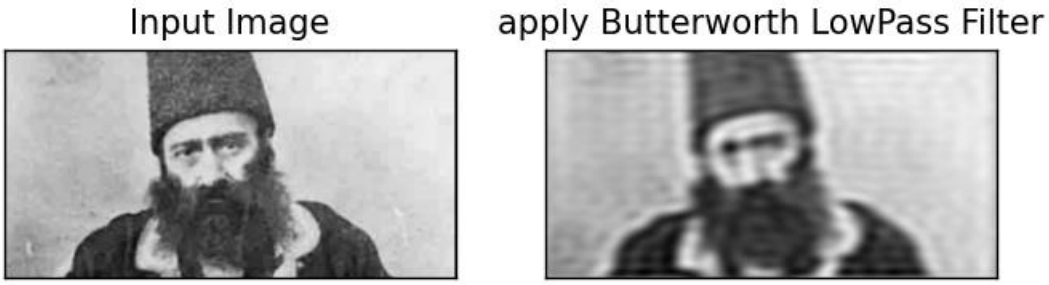
Cutoff = 40



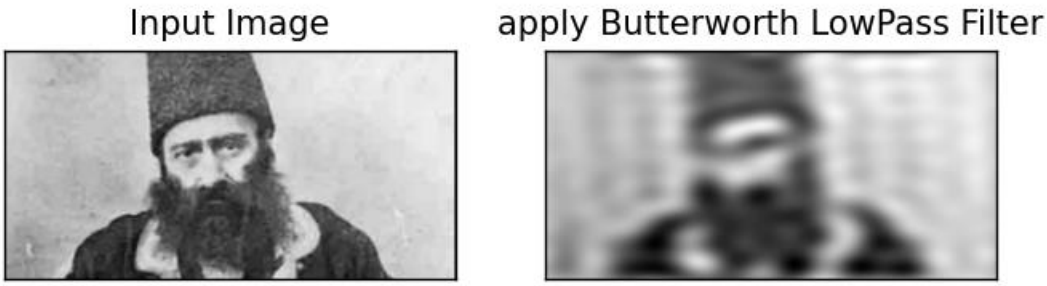
Cutoff = 30



Cutoff = 20



Cutoff = 10



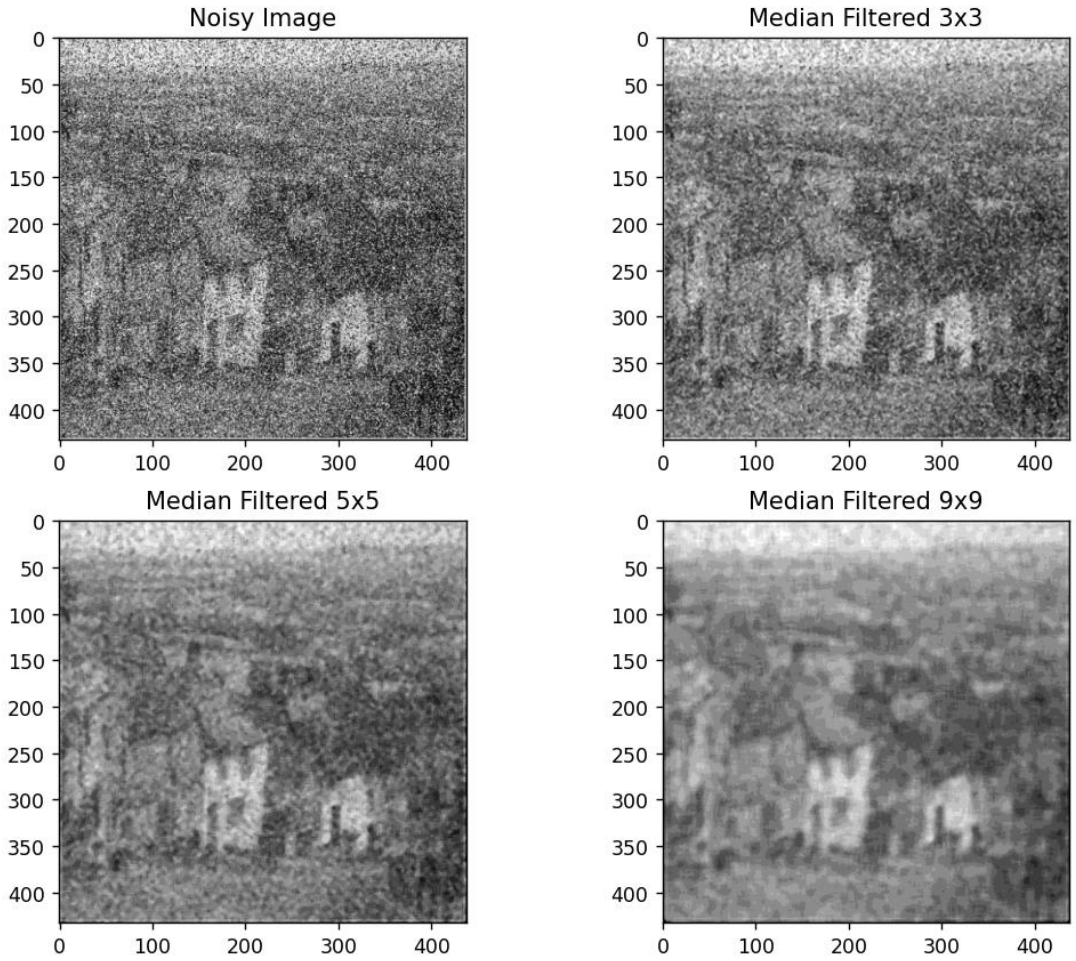
۲) سه تصویر زیر را که با نویز فلفل نمکی تخریب شده است را درنظر بگیرید:

الف) با فیلتر میانه با ماسک ۳*۳ نویز را حذف کنید. یکبار هم با ماسکهای ۵*۵ و ۹*۹ این عمل را انجام داده و نتایج را با هم مقایسه کنید. (کدامیک بهتر است) (از تابع imfilt2 نبایستی استفاده کنید و خودتان کد را بنویسید)

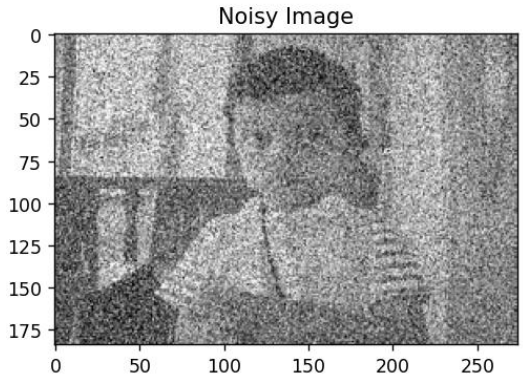
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

#Read images
img1 = cv2.imread("4.jpg",0)
img2 = cv2.imread("6.jpg",0)
img3 = cv2.imread("7.png",0)

def median_filter(image, ksize):
    #Create a new image of the same size as the input image
    filtered_img = np.zeros_like(image)
    #Apply Zero Padding
```

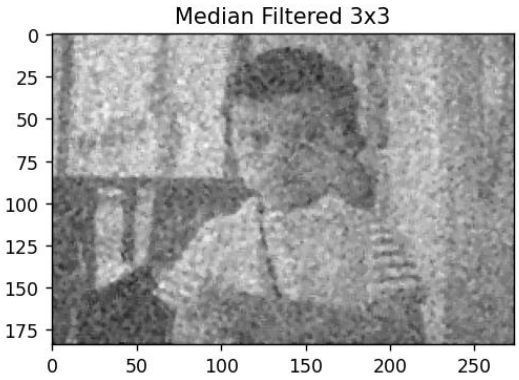
```
    padded_img = np.pad(image, ksize//2, mode='constant')
    #Loop over the image pixels
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
          #Get the neighboring pixels
            neighbors = padded_img[i:i+ksize, j:j+ksize]
          #Find the median value
            median = np.median(neighbors)
          #Assign the median value to the filtered image
            filtered_img[i,j] = median
    return filtered_img

#Apply median filter to images
median1_3x3 = median_filter(img1, 3)
median1_5x5 = median_filter(img1, 5)
median1_9x9 = median_filter(img1, 9)

median2_3x3 = median_filter(img2, 3)
median2_5x5 = median_filter(img2, 5)
median2_9x9 = median_filter(img2, 9)

median3_3x3 = median_filter(img3, 3)
median3_5x5 = median_filter(img3, 5)
median3_9x9 = median_filter(img3, 9)

#Show images
f, subplt1 = plt.subplots(2,2,figsize=(10, 20))
subplt1[0,0].imshow(img1,cmap='gray')
subplt1[0,0].set_title("Noisy Image")
subplt1[0,1].imshow(median1_3x3,cmap='gray')
subplt1[0,1].set_title("Median Filtered 3x3")
subplt1[1,0].imshow(median1_5x5,cmap='gray')
subplt1[1,0].set_title("Median Filtered 5x5")
subplt1[1,1].imshow(median1_9x9,cmap='gray')
subplt1[1,1].set_title("Median Filtered 9x9")

f, subplt2 = plt.subplots(2,2,figsize=(10, 20))
subplt2[0,0].imshow(img2,cmap='gray')
subplt2[0,0].set_title("Noisy Image")
subplt2[0,1].imshow(median2_3x3,cmap='gray')
subplt2[0,1].set_title("Median Filtered 3x3")
subplt2[1,0].imshow(median2_5x5,cmap='gray')
subplt2[1,0].set_title("Median Filtered 5x5")
subplt2[1,1].imshow(median2_9x9,cmap='gray')
subplt2[1,1].set_title("Median Filtered 9x9")
```

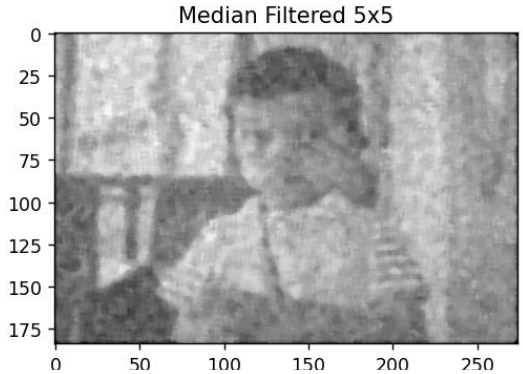
```
f, subplt3 = plt.subplots(2,2,figsize=(10, 20))
subplt3[0,0].imshow(img3,cmap='gray')
subplt3[0,0].set_title("Noisy Image")
subplt3[0,1].imshow(median3_3x3,cmap='gray')
subplt3[0,1].set_title("Median Filtered 3x3")
subplt3[1,0].imshow(median3_5x5,cmap='gray')
subplt3[1,0].set_title("Median Filtered 5x5")
subplt3[1,1].imshow(median3_9x9,cmap='gray')
subplt3[1,1].set_title("Median Filtered 9x9")

plt.show()
```

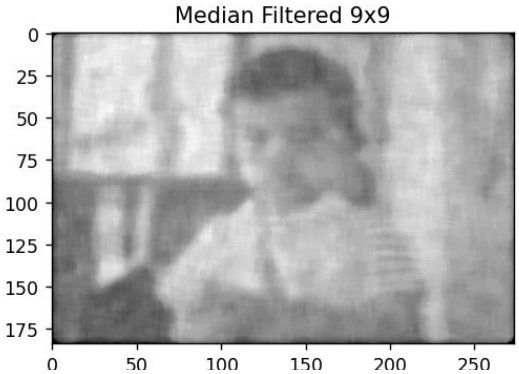


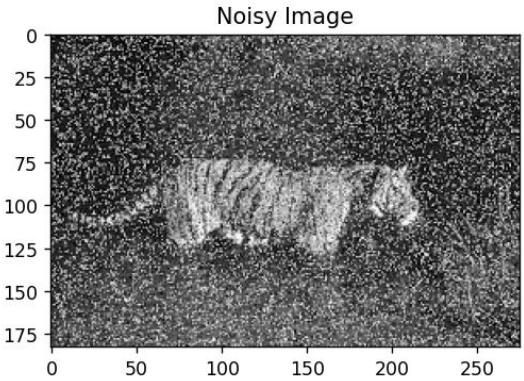
فیلتر ۵*۵ به نسبت دو اندازه دیگر نتیجه بهتری دارد.

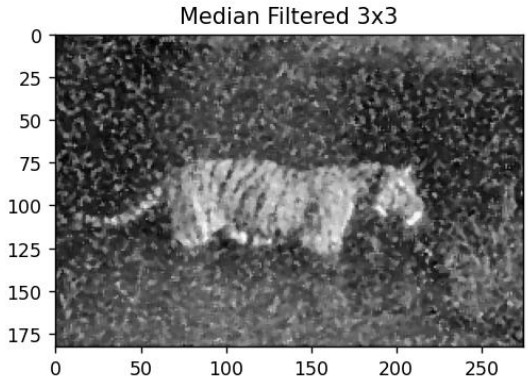

Noisy Image

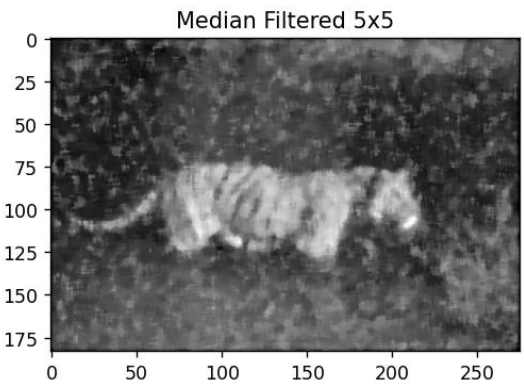

Median Filtered 3x3

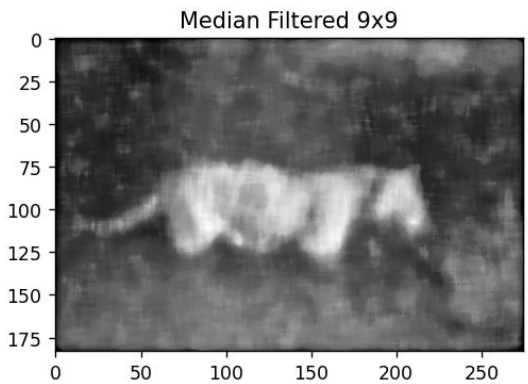

Median Filtered 5x5


Median Filtered 9x9


Noisy Image


Median Filtered 3x3


Median Filtered 5x5


Median Filtered 9x9

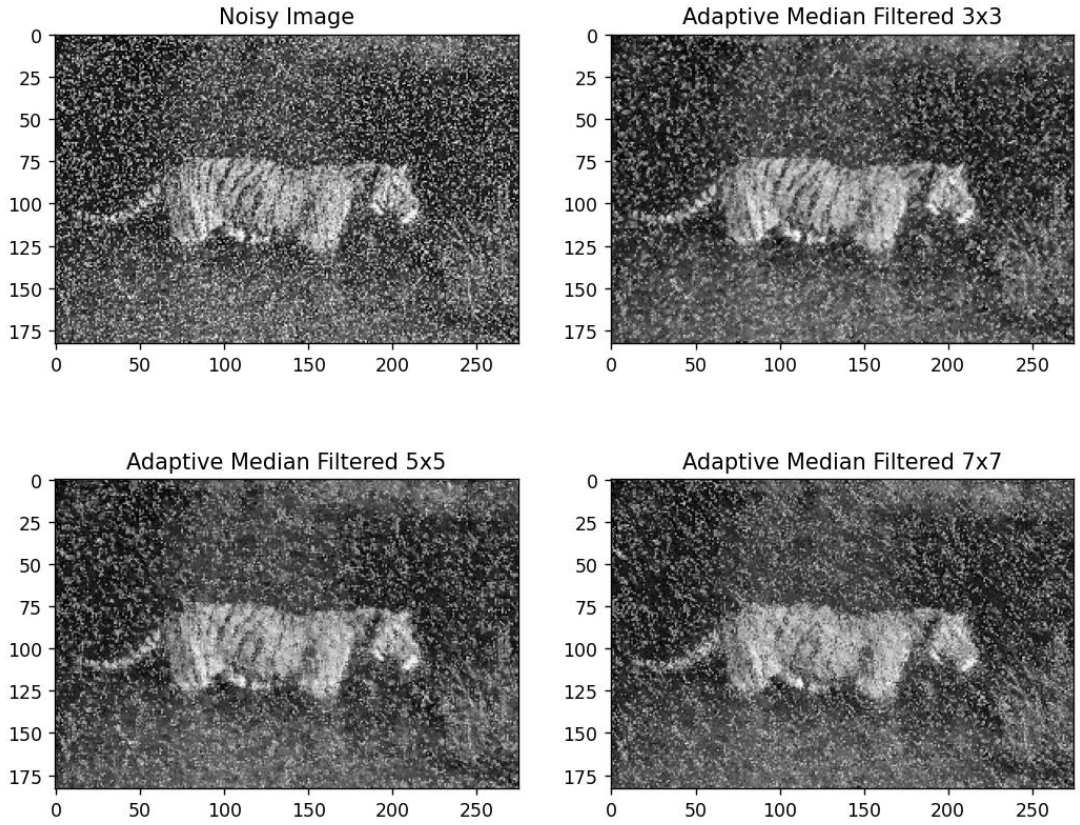
ب) الگوریتم میانه تطبیقی adaptive median را پیاده سازی کرده و نتایج را نمایش دهید. (پارامتر پنجره ماکزیمم را چه مقداری در نظر بگیریم مناسب است؟)

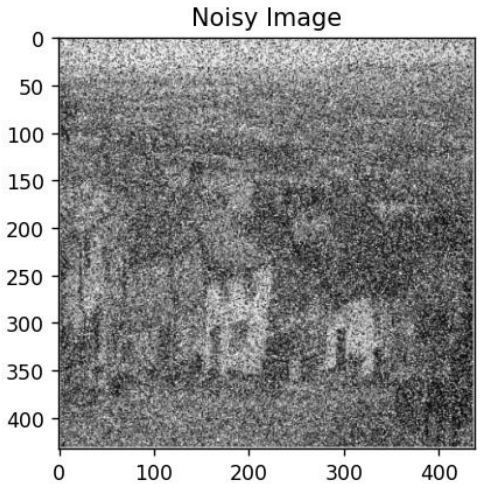
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

#Read images
img1 = cv2.imread("4.jpg",0)
img2 = cv2.imread("6.jpg",0)
img3 = cv2.imread("7.png",0)

def adaptive_median_filter(image, S_max):

    #Create a new image of the same size as the input image
    filtered_img = np.zeros_like(image)
    #Apply Zero Padding
    padded_img = np.pad(image, S_max//2, mode='constant')

    #Loop over the image pixels
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):

            #Initialize the window size
            size = 3
            #Initialize a flag for noise detection
            flag = False

            #Loop until noise is detected or maximum size is reached
            while not flag and size <= S_max:

                #Get the neighboring pixels
                neighbors = padded_img[i:i+size, j:j+size]
                #sort the neighbors
                neighbors = np.sort(neighbors.ravel())
                Z_min = neighbors[0] #min
                Z_max = neighbors[-1] #max
                Z_median = neighbors[len(neighbors)//2] #median

                #condition A
                if Z_min < Z_median < Z_max:

                    #condition B
                    if Z_min < image[i,j] < Z_max:
                        #if the current pixel is not a noise return it
```

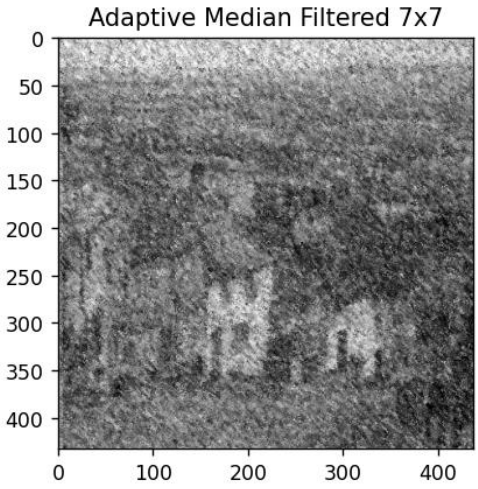
```
                        filtered_img[i,j] = image[i,j]
                    else:
                        #if the current pixel is a noise, take the median
                        filtered_img[i,j] = Z_median

                    #noise flag is true
                    flag = True

                else:#if the median is a noise increase window size
                    size += 2
            #if maximum size is reached and no noise is detected
            if not flag:
               #output the original pixel value
                filtered_img[i,j] = image[i,j]

    return filtered_img

#Apply adaptive median filter to images
adaptive_median1_3x3 = adaptive_median_filter(img1, 3)
adaptive_median1_5x5 = adaptive_median_filter(img1, 5)
adaptive_median1_7x7 = adaptive_median_filter(img1, 7)

adaptive_median2_5x5 = adaptive_median_filter(img2, 5)
adaptive_median2_7x7 = adaptive_median_filter(img2, 7)
adaptive_median2_9x9 = adaptive_median_filter(img2, 9)

adaptive_median3_7x7 = adaptive_median_filter(img3, 7)
adaptive_median3_9x9 = adaptive_median_filter(img3, 9)
adaptive_median3_11x11 = adaptive_median_filter(img3, 11)

#Show images
f, subplt1 = plt.subplots(2,2,figsize=(10,20))
subplt1[0,0].imshow(img1,cmap='gray')
subplt1[0,0].set_title("Noisy Image")
subplt1[0,1].imshow(adaptive_median1_3x3,cmap='gray')
subplt1[0,1].set_title("Adaptive Median Filtered 3x3")
subplt1[1,0].imshow(adaptive_median1_5x5,cmap='gray')
subplt1[1,0].set_title("Adaptive Median Filtered 5x5")
subplt1[1,1].imshow(adaptive_median1_7x7,cmap='gray')
subplt1[1,1].set_title("Adaptive Median Filtered 7x7")

f, subplt2 = plt.subplots(2,2,figsize=(10,20))
subplt2[0,0].imshow(img2,cmap='gray')
subplt2[0,0].set_title("Noisy Image")
subplt2[0,1].imshow(adaptive_median2_5x5,cmap='gray')
```

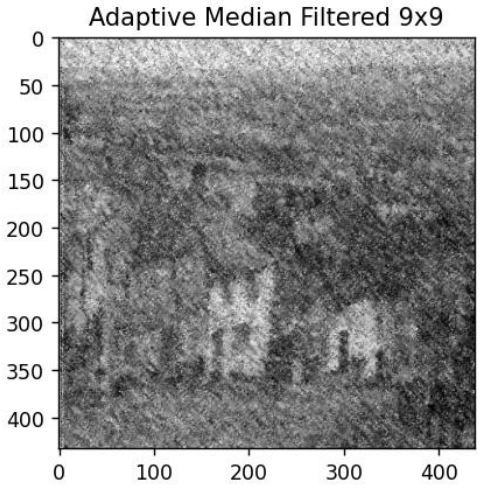
```
subplt2[0,1].set_title("Adaptive Median Filtered 5x5")
subplt2[1,0].imshow(adaptive_median2_7x7,cmap='gray')
subplt2[1,0].set_title("Adaptive Median Filtered 7x7")
subplt2[1,1].imshow(adaptive_median2_9x9,cmap='gray')
subplt2[1,1].set_title("Adaptive Median Filtered 9x9")

f, subplt3 = plt.subplots(2,2,figsize=(10,20))
subplt3[0,0].imshow(img3,cmap='gray')
subplt3[0,0].set_title("Noisy Image")
subplt3[0,1].imshow(adaptive_median3_7x7,cmap='gray')
subplt3[0,1].set_title("Adaptive Median Filtered 7x7")
subplt3[1,0].imshow(adaptive_median3_9x9,cmap='gray')
subplt3[1,0].set_title("Adaptive Median Filtered 9x9")
subplt3[1,1].imshow(adaptive_median3_11x11,cmap='gray')
subplt3[1,1].set_title("Adaptive Median Filtered 11x11")

plt.show()
```

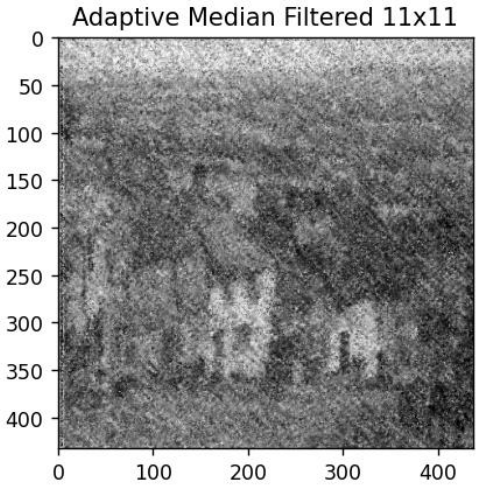
بهترین اندازه فیلتر بستگی به میزان نویز دارد ولی به طور متوسط اندازه ۷ * ۷ خوب عمل میکند.

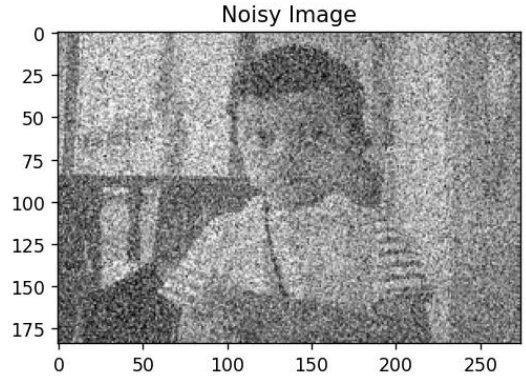


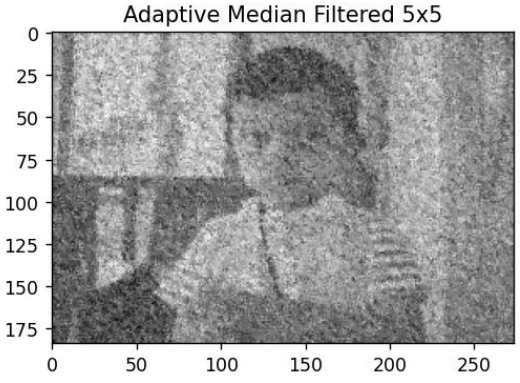

Noisy Image

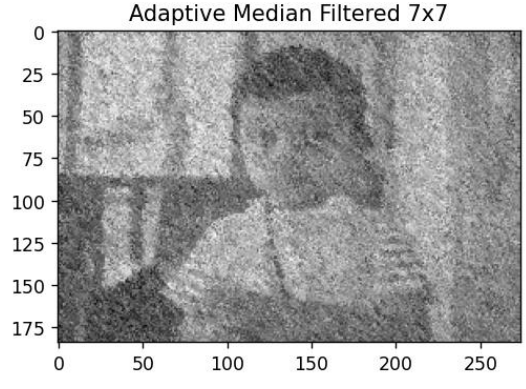

Adaptive Median Filtered 7x7

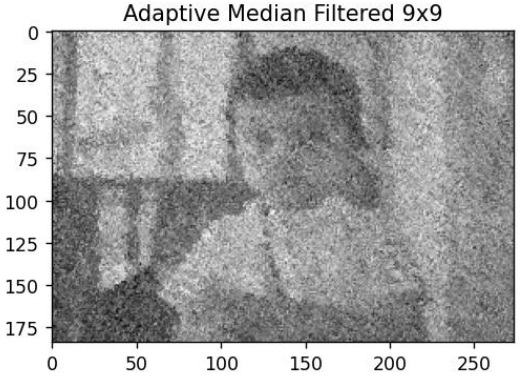

Adaptive Median Filtered 9x9


Adaptive Median Filtered 11x11


Noisy Image


Adaptive Median Filtered 5x5

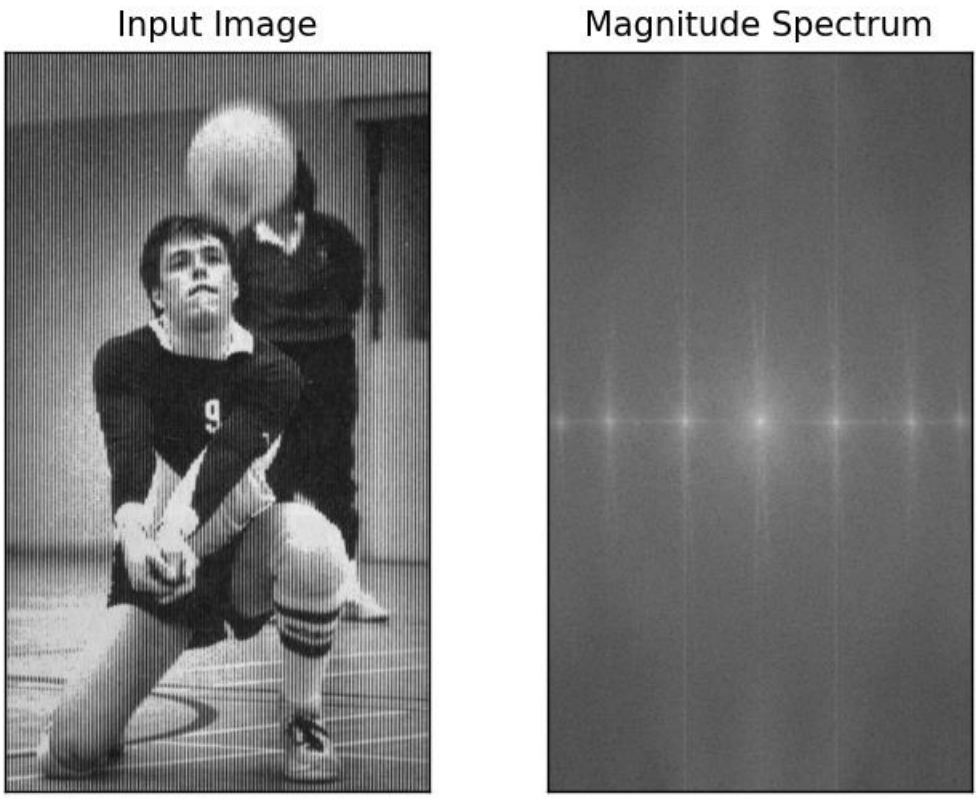

Adaptive Median Filtered 7x7


Adaptive Median Filtered 9x9

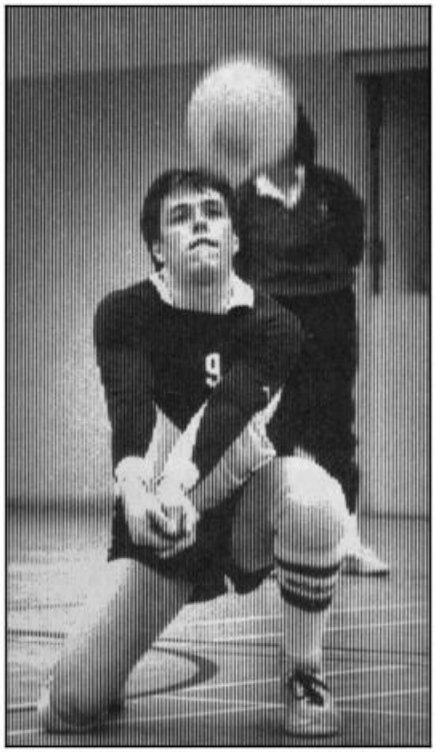
۳) تصویر زیر را که دارای نویز می باشد را بخوانید و برنامه ای نوشته و نویز آنرا حذف نمائید.
راهنمایی: بایستی از تصویر را به حوزه فرکانس ببرید
برای اینکار میتوانید فیلتر را خودتان بصورت دستی در حوزه فرکانس و در نرم افزار paint اعمال کنید و یا با جستجو در طیف فوریه نویز را تشخیص دهید.

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt

img = cv.imread('1.jpg', cv.IMREAD_GRAYSCALE)
assert img is not None, "file could not be read, check with os.path.exists()"
dft = cv.dft(np.float32(img),flags = cv.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)
magnitude_spectrum = 20*np.log(cv.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))
plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()



#First soloution
rows, cols = img.shape
crow,ccol = rows//2 , cols//2
# create a special mask
mask3 = np.ones((rows,cols,2),np.uint8)
mask3[3:1089, 190:206] = 0
mask3[3:1089, 410:430] = 0
#mask3[543:552, 3:620] = 0
#mask3[339:785, 322:309] = 0
#mask3[353:719, 296:328] = 0
# apply mask and inverse DFT
fshift = dft_shift*mask3
f_ishift = np.fft.ifftshift(fshift)
img_back = cv.idft(f_ishift)
img_back = cv.magnitude(img_back[:,:,0],img_back[:,:,1])
plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
plt.title('"Noch Reject Filter"'), plt.xticks([]), plt.yticks([])
plt.show()
```

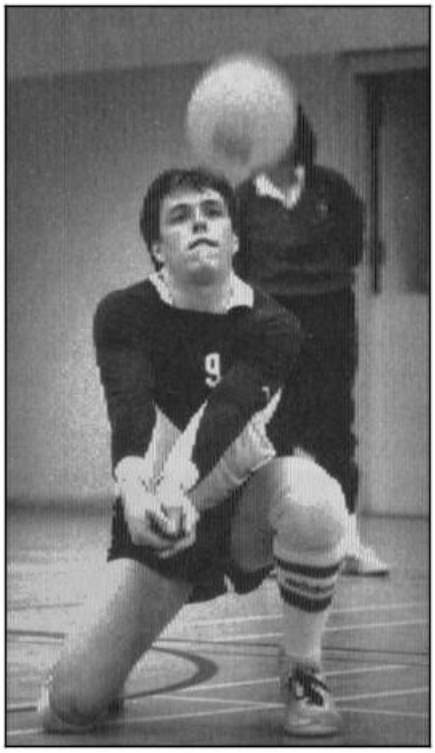
```
#Second soloution
rows, cols = img.shape
crow,ccol = rows//2 , cols//2
# create a mask first, center square is 1, remaining all zeros
mask = np.zeros((rows,cols,2),np.uint8)
mask[crow-70:crow+70, ccol-70:ccol+70] = 1
# apply mask and inverse DFT
fshift = dft_shift*mask
f_ishift = np.fft.ifftshift(fshift)
img_back = cv.idft(f_ishift)
img_back = cv.magnitude(img_back[:,:,0],img_back[:,:,1])
plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
plt.title('"Ideal LowPass Filter"'), plt.xticks([]), plt.yticks([])
plt.show()
```



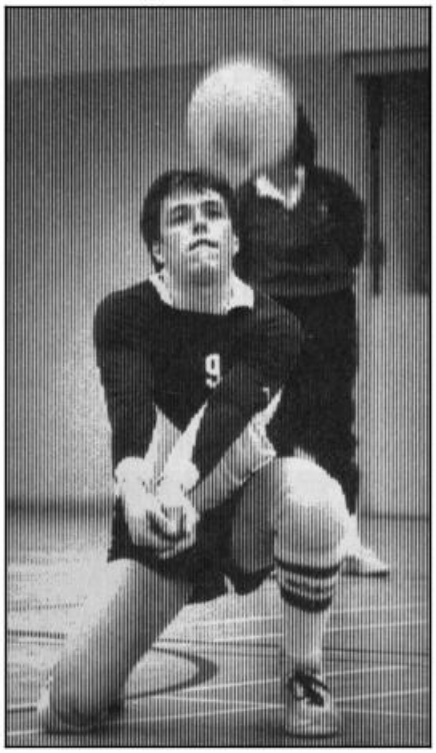
Input Image



"Noch Reject Filter"



Input Image



"Ideal LowPass Filter"