به نام خدا

گزارش آزمایش چهارم

آزمایشگاه پردازش سیگنال

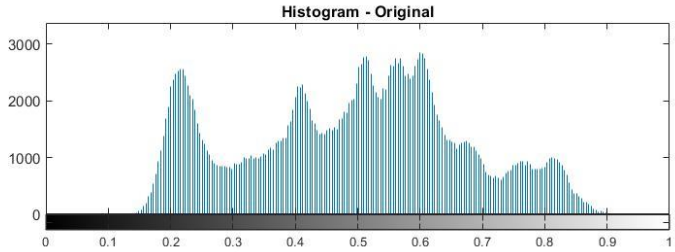
آیدین روزبه - ۹۹۲۳۰۳۷

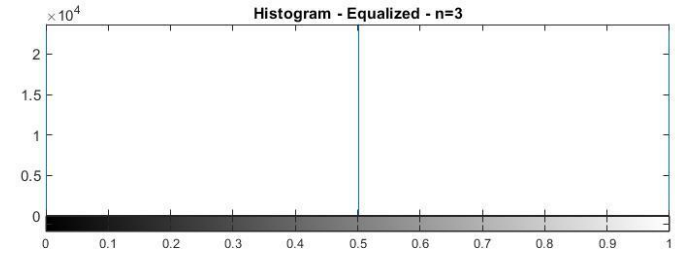
پاییز ۱۴۰۲

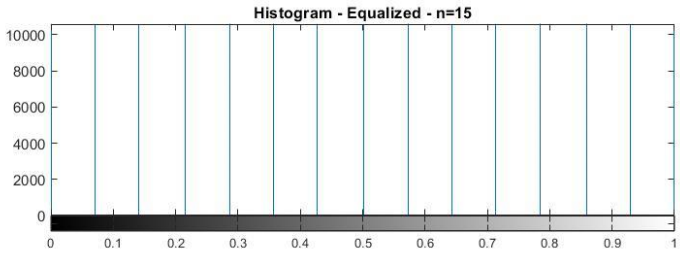
بخش ۴-۱)

```
% 4-1-a)
im1 = imread("lena.bmp","bmp");

% 4-1-b)
im1db = im2double(im1);

% 4-1-c , 4-1-d , 4-1-e)
im1db_eq3 = histeq(im1db,3);
im1db_eq15= histeq(im1db,15);

figure(1)
subplot(3,2,1);
imshow(im1db);
title("Image - Original");

subplot(3,2,2);
imhist(im1db);
title("Histogram - Original");

subplot(3,2,3);
imshow(im1db_eq3);
title("Image - Equalized - n=3");

subplot(3,2,4);
imhist(im1db_eq3);
title("Histogram - Equalized - n=3");

subplot(3,2,5);
imshow(im1db_eq15);
title("Image - Equalized - n=15");

subplot(3,2,6);
imhist(im1db_eq15);
title("Histogram - Equalized - n=15");
```

دستور histeq شدت نور تصویر را به گونه تغییر می‌دهد که تا هیستوگرام شدت نور، فقط به ازای n مقدار مختلف در بازه ۰ تا ۱ (با فاصله های برابر) وجود داشته باشد و تا حد امکان flat باشد. نتیجه شبیه سازی برای دو مقدار n=15 و n=3 در تصویر زیر قابل مشاهده است.







دقت شود که هر دو نمودار برای n=3 و n=15، مقدار 0 و 1 در نمودار وجود دارند.

بخش۴-۲-الف تا ۴-۲-د)

```
%% ---------- Part 2 -----------
% Operation are done on the same 'lena' picture
% 4-2-a to 4-2-d)
im_n = imnoise(im1db, 'gaussian', 0 , 0.04);

mat3=(1/9)*ones(3,3);
mat5=(1/25)*ones(5,5);
im_dn3 = conv2(im_n , mat3);
im_dn5 = conv2(im_n , mat5);

figure(2);
subplot(1,3,1);
imshow(im_n);
title("Noisy Image, Var=0.04");
```

```
subplot(1,3,2);
imshow(im_dn3);
title("Filtered Image, n=3");

subplot(1,3,3);
imshow(im_dn5);
title("Filtered Image, n=5");
```

مشاهده می‌شود که با افزایش سایز فیلتر از ۳ به ۵، با این که میزان نویز در تصویر کمتر شد، حاشیه ها و جزئیات در تصویر اصلی نیز کمرنگ تر و ناواضح تر شده اند. در این بخش نیز از تصویر lena استفاده شده است.

بخش ۴-۲-هـ، ۴-۲-و)

```
% 4-2-e , 4-2-f) Salt and Pepper noise
im_sp = imnoise(im1db , "salt & pepper" , 0.1);
im_sp3 = conv2(im_sp , mat3);
im_sp5 = conv2(im_sp , mat5);

figure(3);
subplot(1,3,1);
imshow(im_sp);
title("Noisy Image - Salt & Pepper");
```

```
subplot(1,3,2);
imshow(im_sp3);
title("Filtered, n=3");

subplot(1,3,3);
imshow(im_sp5);
title("Filtered, n=5");
```

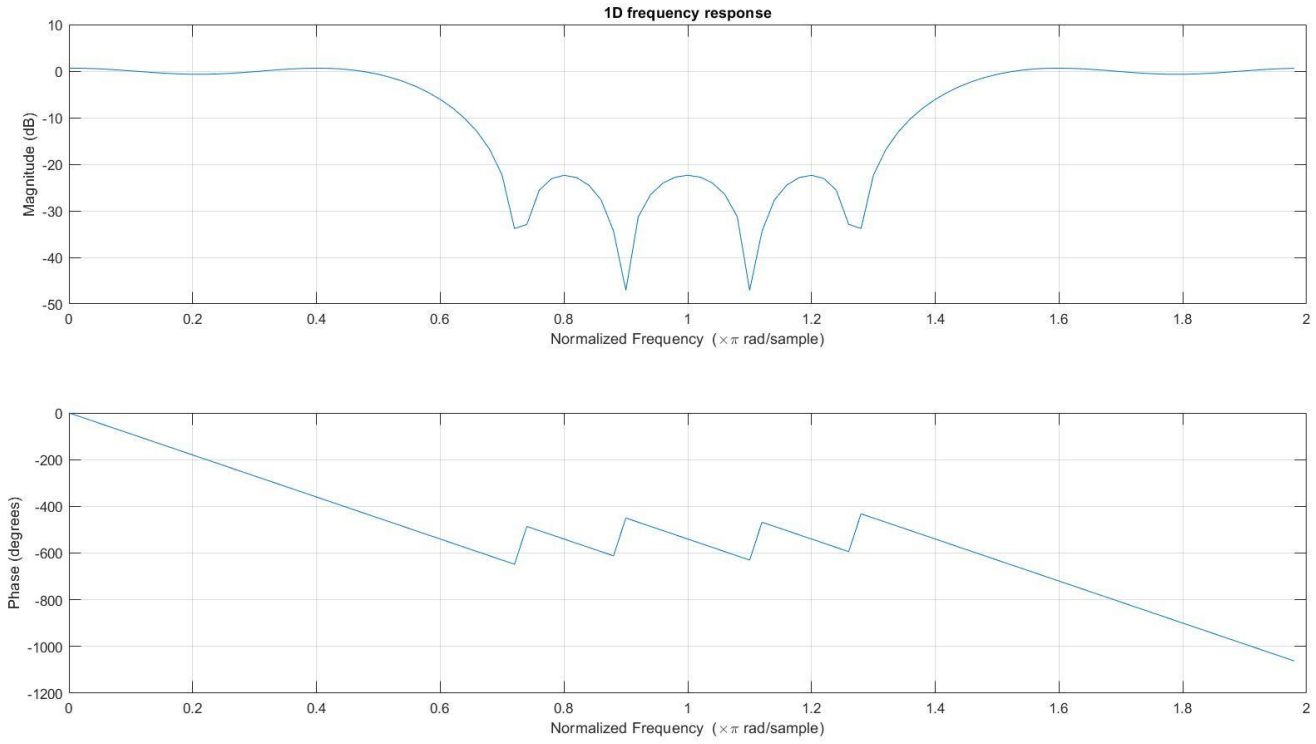
عکس با نویز نمک و فلفل در متغیر های im_sp3 و im_sp5 ذخیره شده اند. . نتیجه شبیه سازی به صورت زیر است:

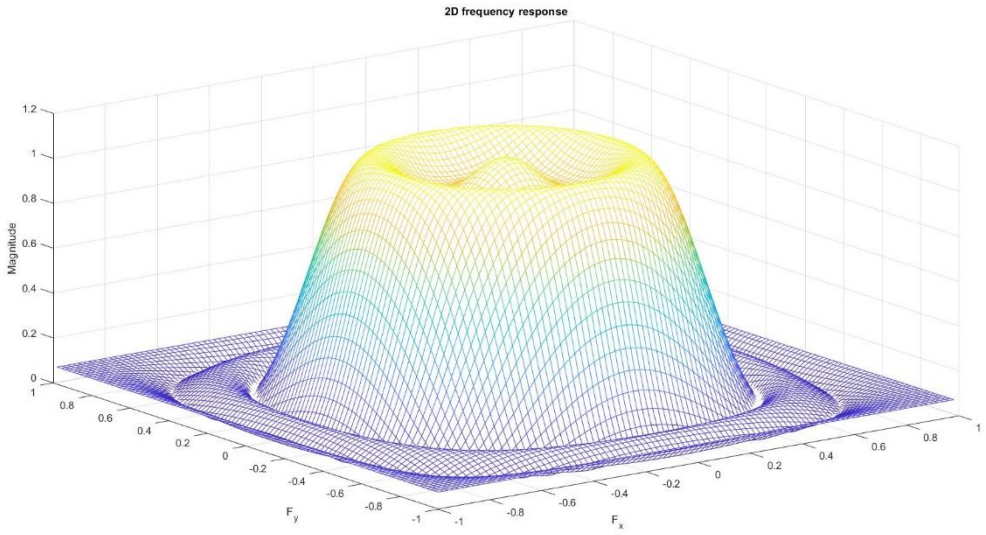
مشاهده می‌شود که فیلتر میانگین گیر با n=3 و n=5 در حذف نویز نمک و فلفل، آنچنان موفق عمل نکرده است. بدین ترتیب لازم است تا فیلتر دیگری برای حذف این نویز از تصویر طراحی بشود.

بخش ۲-۴-ز)

```
% 4-2-g)
f1 = load("fil1.mat").coeff;
f2 = ftrans2(f1);

figure(4);
freqz(f1,1,'whole',100);
title("1D frequency response");

figure(5);
freqz2(f2,[100 100]);
title("2D frequency response");
```

فیلتر یک بعدی متناظر با این سوال، با استفاده از ابزار fdatool طراحی شده و فایل sessio مرتبط با این فیلتر در fil1.fda و ضرایب محاسبه شده در فایل fil1.mat ذخیره شده اند. نمودار های زیر به ترتیب مربوط به حالت یک بعدی و دو بعدی این فیلتر هستند.

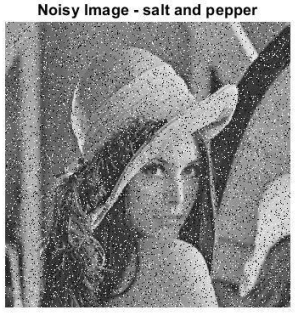


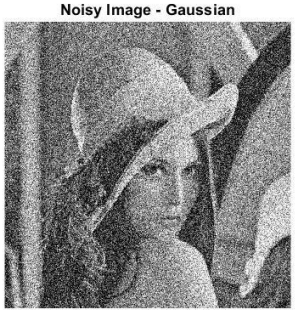


بخش ۴-۲-ح)

```
im_sp_fil = imfilter(im_sp , f2);
im_n_fil = imfilter(im_n , f2);

figure(6);
subplot(2,2,1);
imshow(im_sp);
title("Noisy Image - salt and pepper");

subplot(2,2,2);
imshow(im_sp_fil);
title("Filtered image using fdatool");

subplot(2,2,3);
imshow(im_n);
title("Noisy Image - Gaussian");

subplot(2,2,4);
imshow(im_n_fil);

title("Filtered image using fdatool");
```

فیلتر طراحی شده به هر دو تصویر نویزی (گاوسی و نمک‌وفلفل) اعمال شده است.





بخش ۴-۲-ط)

```
function img_out = median_filter(img_in , w)

if(~mod(w,2))
    error("WINDOWS SIZE IS NOT AN ODD NUMBER");
end


s=size(img_in);
img_out = zeros(s(1)-1 , s(2)-1);

for i=2:1:s(1)-1
    for j=2:1:s(2)-1
        img_out(i,j) = median( img_in(i-(w-1)/2:i+(w-1)/2 , j-(w-1)/2:j+(w+1)/2) ,
'all');
    end
end
```

هر پیکسل در شکل خروجی، میانه‌ی مجموعه خود آن پیکسل و ۸ همسایه آن است (برای حالت w=3)

بخش ۴-۲-ک و ۴-۲-ل)

```
im_med3 = median_filter(im_sp , 3);
im_med5 = median_filter(im_sp , 5);

figure(7)
subplot(1,3,1);
imshow(im_sp);
title("Image + Salt and Pepper noise");

subplot(1,3,2);
imshow(im_med3);
title("Median filtered , N=3");

subplot(1,3,3);
imshow(im_med5);
title("Median filtered , N=5");
```

مشاهده می شود که این فیلتر در حذف نویز نمک و فلفل بسیار موفق عمل کرده است. همچنین به ازای N=5 نسبت به N=3، اگر به جزئیات عکس دقت کنیم، نویز بیشتری از تصویر گرفته شده است. اما در شکل دوم مشاهده می شود با افزایش N، با این که نویز نمک و فلفل بهتر حذف می شود، اما لبه ها و حاشیه های تصویر نیز کیفیت خود را از دست داده اند. شکل دوم، همان تصویر اول از یک نمای نزدیک تر است.

Image + Salt and Pepper noise

Median filtered , N=3

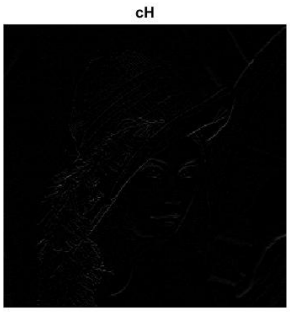
Median filtered , N=5

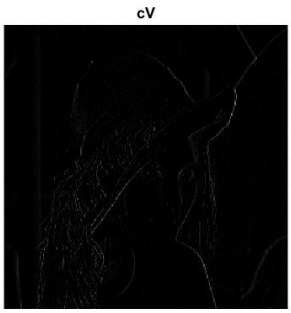
Median filtered ,

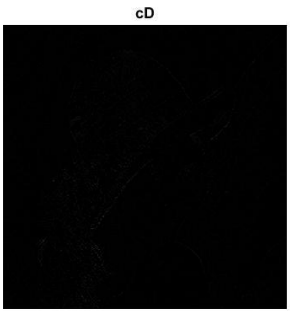
Median filtered , N=5

بخش ۴-۳-الف)

```
% 4-3-a)
[cA,cH,cV,cD] = dwt2(im1db , 'db1');

figure(8)
subplot(2,2,1);
imshow(cA);

subplot(2,2,2);
imshow(cH);

subplot(2,2,3);
imshow(cV);

subplot(2,2,4);
imshow(cD);
```





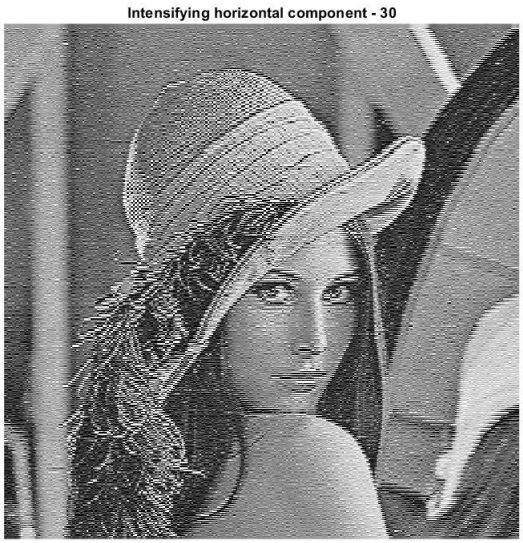


ورودی این تابع، تصویر `lena` و خروجی آن ضرایب `wavelet` این تصویر می باشد.

بخش ۴-۳-ب)

برای هایلایت کردن مولفه های افقی تصویر، لازم است پس از تجزیه (analysis)، اندازه ضرایب مرتبط با مولفه های مد نظر (cV) را تقویت کرده و سپس تصویر جدید را از روی ضرایب تغییر یافته بسازیم (synthesis).

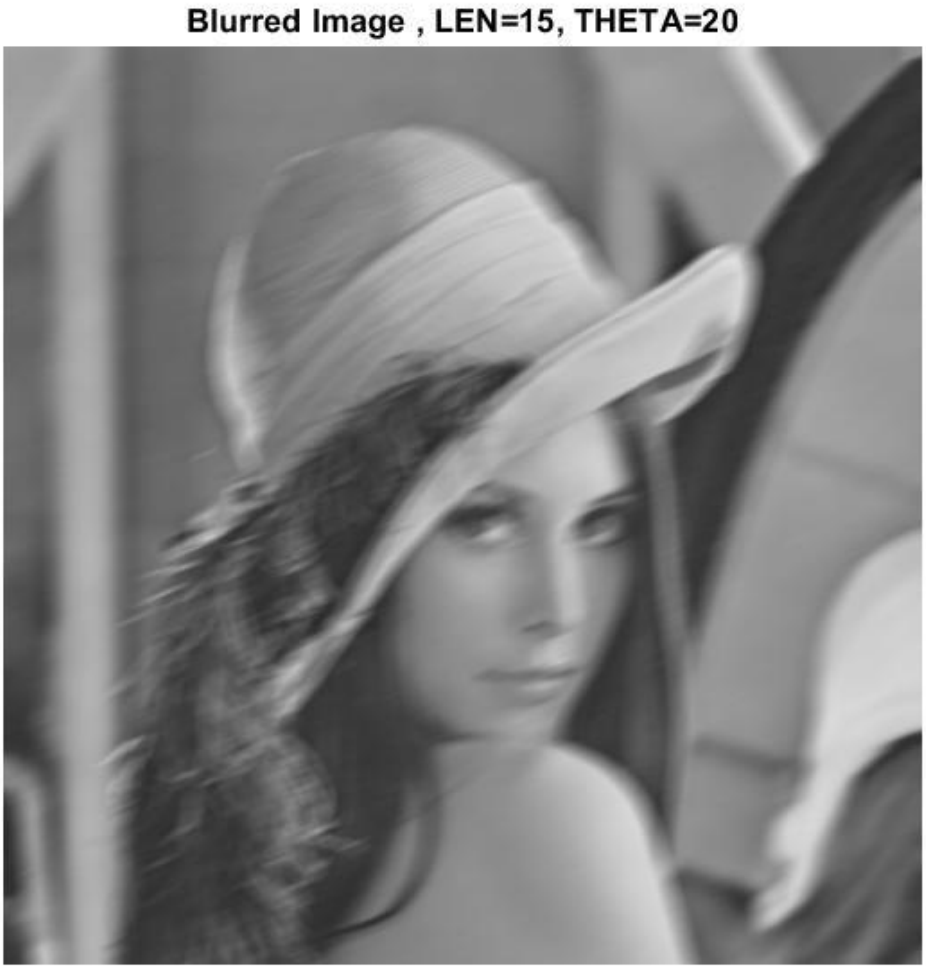
```
% 4-3-b)
% image reconstruction
figure(9);

subplot(1,2,1);
imshow(idwt2(cA,cH,30*cV,cD,'db1'));
title("Intensifying vertical component - 30");

subplot(1,2,2);
imshow(idwt2(cA,30*cH,cV,cD,'db1'));
title("Intensifying horizontal component - 30");
```



در تصویر راست و چپ به ترتیب مولفه های افقی و عمودی تقویت شده اند (اندازه آن ها ۳۰ برابر شده است)

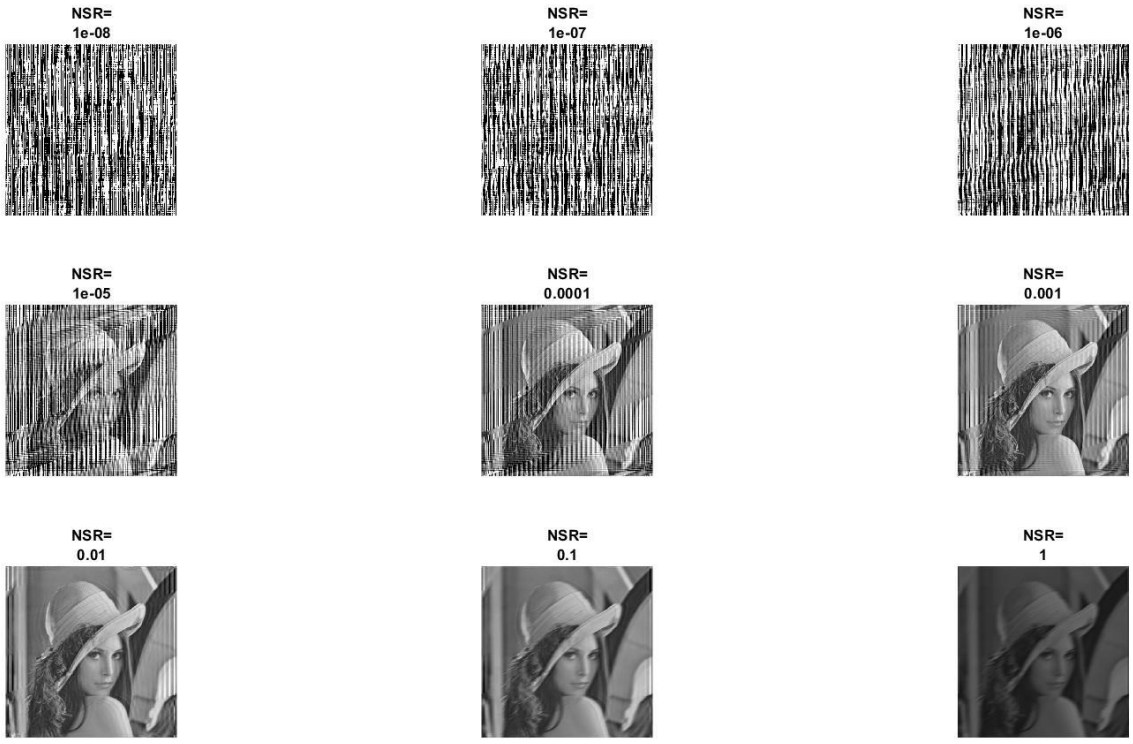
بخش ۴-۴-الف)

```
% Motion Bluring
LEN = 15;
THETA = 20;

% 4-4-a)
mot_fil = fspecial('motion',LEN,THETA);
im_mot = imfilter(im1db,mot_fil,'replicate');

figure(10);
imshow(im_mot);
title("Blurred Image , LEN=15, THETA=20");
```



بخش ۴-۴-ب)

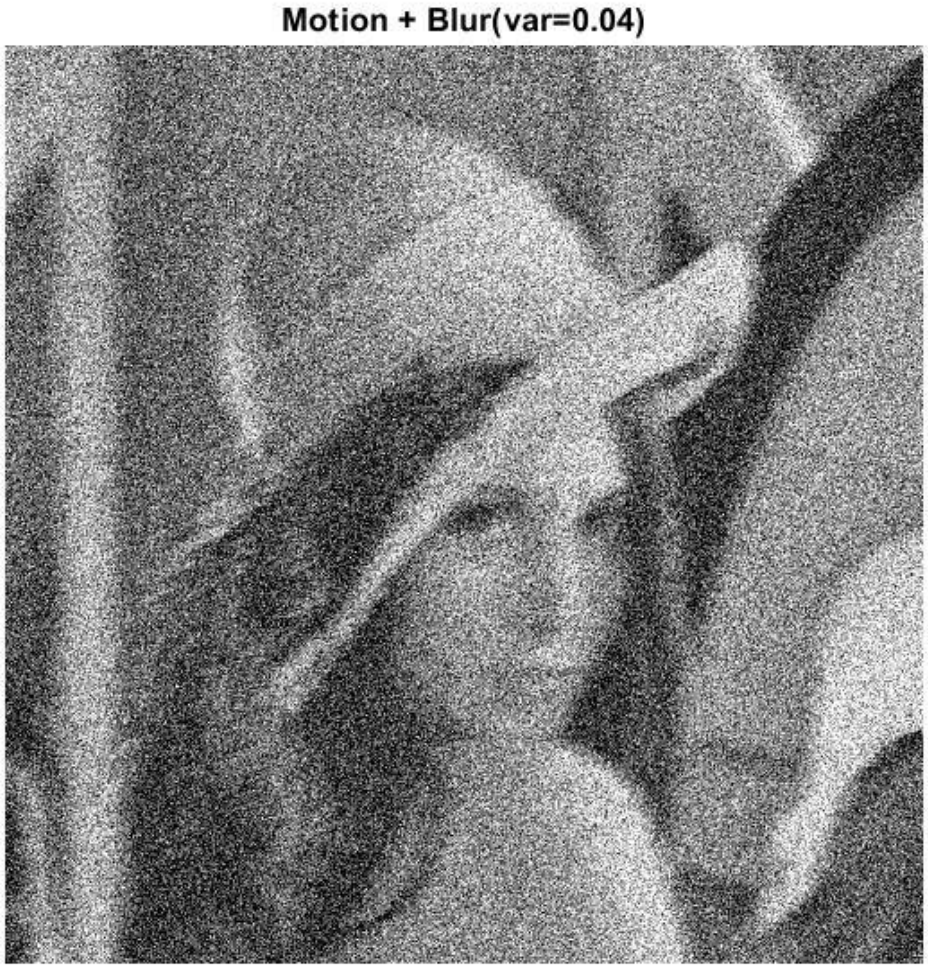
```
% 4-4-b)
figure(11);
for i=0:1:8
    subplot(3,3,i+1);
    imshow(deconvwnr(im_mot, mot_fil, 10^(-8+i)));
end
```



برای NSR های مختلف از 10^{-8} تا 10^{0}، فیلتر `wiener` به تصویر اعمال شده است. در حالت اول، NSR بسیار نزدیک به صفر است.

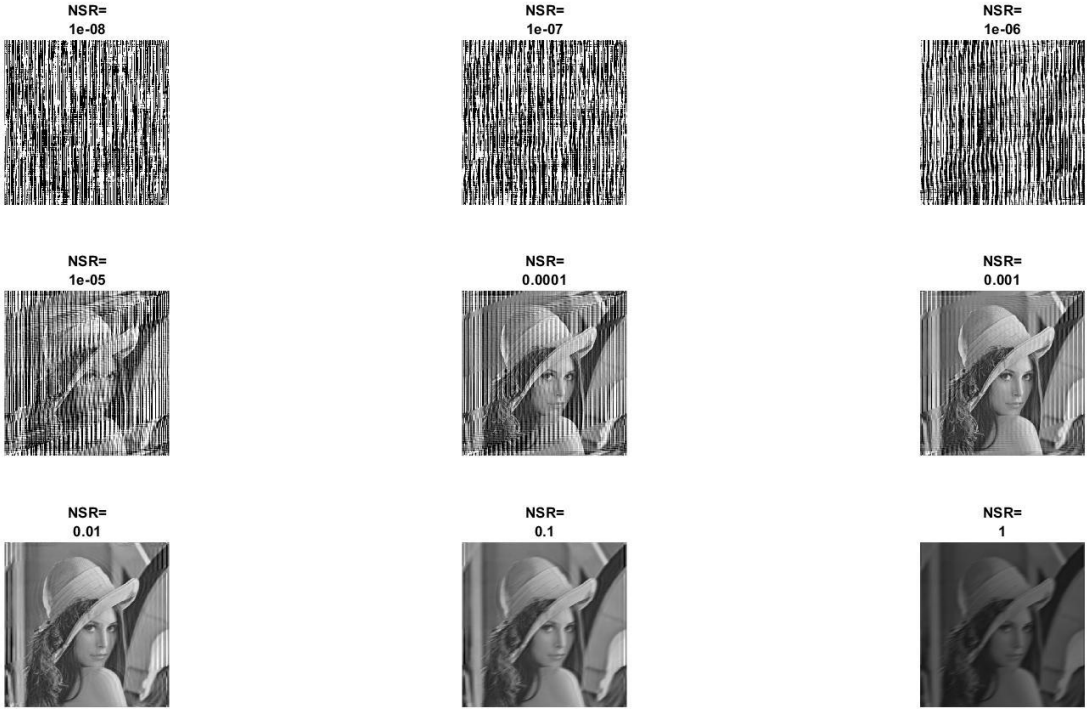
بخش ۴-۴-ج)

نتیجه چندین آزمایش مختلف نشان می‌دهد برخلاف خواسته سوال، در صورتی که نویز با واریانس ۱۰ به این تصویر اضافه شود، هرگز امکان بازیابی تصویر اصلی (به دلیل توان بسیار بالای نویز) وجود ندارد. به همین دلیل تصمیم بر آن شد تا از نویز گوسی با توان `0.04` استفاده شود. تصویر زیر، از افزودن نویز مذکور به تصویر `blur` شده حاصل شده است.



بخش ۴-۴-د)

```
% 4-4-d)
figure(13);
for i=0:1:8
    subplot(3,3,i+1);
    imshow(deconvwnr(im_mot_blur, mot_fil, 10^(-8+i)));
end
```

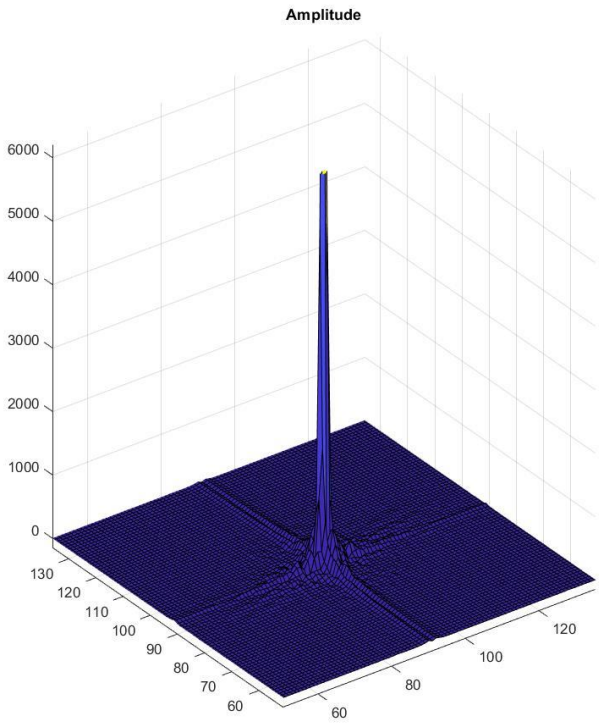


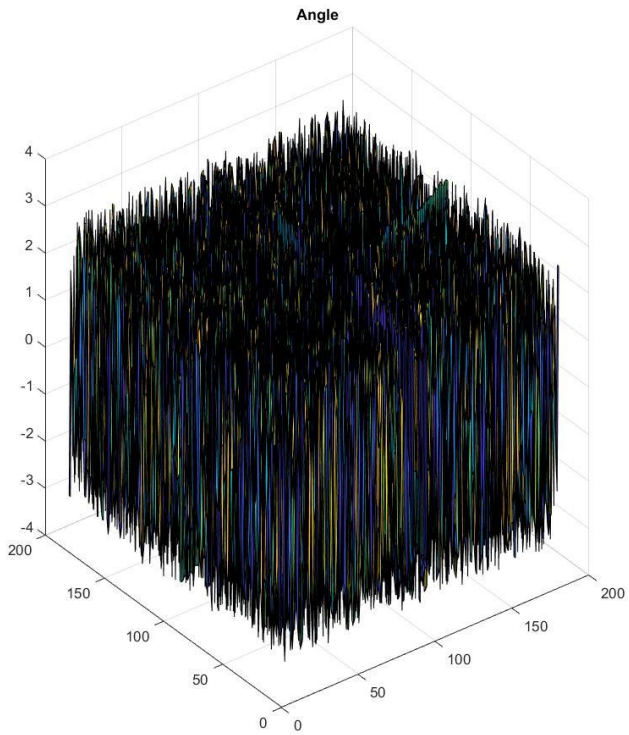
مشابه حالت قبلی با مقادیر مختلف NSR این فیلتر اعمال شده است.

بخش ۴-۵-الف و ۴-۵-ب)

```
% 4-5-a and 4-5-b)
im_temp = imread('glass.png'); % Loading the glass image

%cropping the image to make a square
im_temp_crop = im_temp(1:min(size(im_temp)) , 1:min(size(im_temp)));

im_glass = im2double(im_temp_crop);
im_fft = fft2(im_glass);
im_fft_amp = fftshift(abs(im_fft));
im_fft_ang = fftshift(angle(im_fft));

figure(14);

subplot(1,2,1);
surf(im_fft_amp);
title("Amplitude");

subplot(1,2,2);
surf(im_fft_ang);
title("Angle");
```

برای محاسبه سریع تر و دقت بالاتر فیلتر، ابتدا ابعاد تصویر به یک مربع کاهش داده شده است.



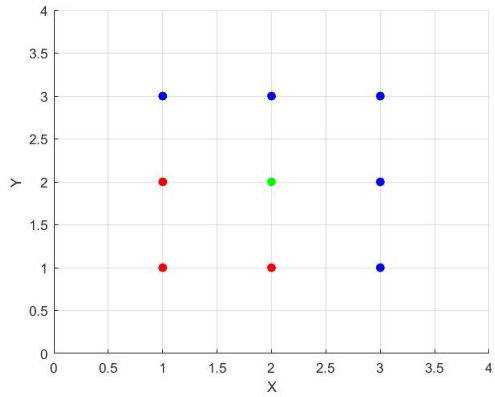


بخش ۴-۵-ج)

```
function im_out = fft_lp_2d(im_in , w)

if(w>pi || w<0)
    error("CUTOFF FREQUENCY SIZE IS NOT CORRECT");
end

temp = fftshift(fft2(im_in));
im = abs(temp);
ang = angle(temp);

s = size(im);
c= round(s/2);

r=w/pi;
R=round(r*s(1)/2);

x=c(1)+R;
y=c(2);
R2=R^2;
d=R2;

while(x>=c(1))
    if(d<R2)
        d=d+2*(y-c(2))+1;
        y=y+1;
    else
       im( x:s(1) , y:s(2))=0;
       im( x:s(1) , 1:s(2)-y)=0;
       im( 1:s(1)-x , y:s(2))=0;
       im( 1:s(1)-x , 1:s(2)-y)=0;

       d=d-2*(x-c(1))+1;
       x=x-1;
    end
end

im_out=ifft2(ifftshift(im.*exp(1i*ang)));

end
```

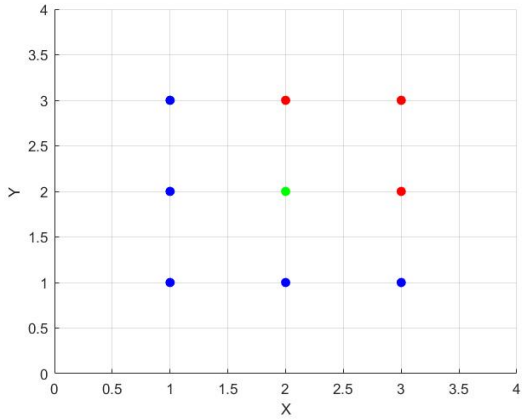
در این فیلتر فرض می‌شود ورودی، یک ماتریس مربعی با مقادیر مختلط است (مشابه تبدیل فوریه یک تصویر). بعد از تفکیک بخش حقیقی و موهومی، ضرایبی که فاصله آن ها از مرکز (فرکانس صفر) بیشتر از مقدار مشخص شده (**w** که به شعاع تعبیر می شود) باشد، برابر با صفر قرار خواهد گرفت. به اختصار درباره الگوریتم این تابع توضیح می‌دهیم:

برای پیدا کردن نقاطی که فاصله آن ها از مرکز بیشتر از حد مورد نظر (در اینجا شعاع با اندازه **r**) است، لازم به بررسی همه نقاط نیست. اگر یک نقطه (با مختصات **(x,y)**) در این شرط صدق کند، قطعا نقاطی که از آن نقطه نسبت به مبدا فاصله کمتری دارند (از بین همسایه هایش) نیز شرط فاصله را ارضا می‌کنند.



مثلا در تصویر بالا، اگر نقطه سبز فاصله مناسب داشته باشد، دیگر نیازی به بررسی نقاط قرمز نیست (فعلا درباره نقاط آبی با اطمینان صحبت نمی‌کنیم)

عکس حالت توضیح داده شده نیز، می‌تواند به ما کمک کند. یعنی اگر مطمئن باشیم یک نقطه خاص در شرط فاصله صدق نمی‌کند، دیگر نیاز به بررسی نقاط همسایه که فاصله بیشتر از مبدا دارند، نیست.



در شکل بالا، نقاط قرمز رنگ، در صورتی که نقطه سبز شرط فاصله را ارضا نکند، قطعا خارج از ناحیه عبور فیلتر هستند.

بدین ترتیب، می‌توان روشی سریع تر ($O(N)$) برای پیدا کردن نقاط مطلوب یافت. به ترتیب زیر عمل می‌کنیم:

۱- از نقطه $(x=R,y=0)$ حرکت خود را آغاز کن (متغیر فاصله در ابتدا برابر با R^2)

۲- شرط فاصله را بررسی کن – شرط توقف: رسیدن به نقطه $(x=0 \ , \ y=R)$

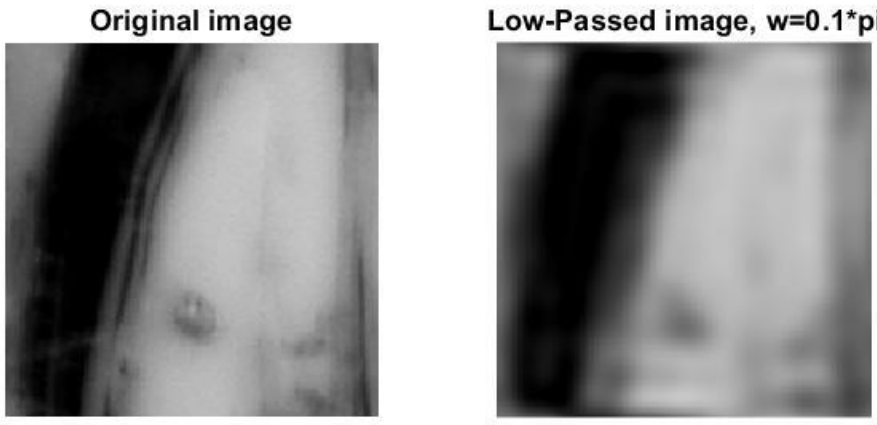
- اگر نقطه فعلی در شرط صدق می کرد>> فاصله را آپدیت کن، یک خانه به بالا حرکت کن >> به مرحله ۲ برو

  در غیر این صورت >> فاصله را آپدیت کن، یک خانه به چپ حرکت کن، ادامه سطر و ستون از سمت راست و بالا را برابر با صفر قرار بده >> به مرحله ۲ برو

در واقع الگوریتم بالا در تلاش است تا فقط نقاط نزدیک به مرز دایره با شعاع R را بررسی کند.

بخش ۴-۵-د)

```
% 4-5-d)

im_fft_fil= fft_lp_2d(im_glass , 0.1*pi);

figure(15);
subplot(1,2,1);
imshow(im_glass);
title("Original image");

subplot(1,2,2);
imshow(im_fft_fil);
title("Low-Passed image, w=0.1*pi");
```



بخش ۴-۵-هـ)

```
% 4-5-e)
im_dwn = downsample(downsample(im_glass,4)',4)';
im_dwn_fil = fft_lp_2d(im_dwn , 0.6*pi);

figure(16);
subplot(1,2,1);
imshow(im_dwn);
title("Downsampled glass image");

subplot(1,2,2);
imshow(im_dwn_fil);
title("Filtered, w=0.1*pi");
```