Project 4

Image Denoising and Wiener Filtering

Md Zobaer Islam

PhD student, Oklahoma State University

CWID: A20270547

Explanation of submission files (only .m files in alphabetical order)

File name	What it is---
Part1_ALND.m	Code for arithmetic mean filtering, Gaussian smoothing filtering, and adaptive local noise denoising, including quantitative analysis using PSNR and SSIM
part2_adap_med_filt.m	Code for performing adaptive median filtering with optimum max size of the window and presenting the results.
part2_adapMedFilt_quant.m	Quantitative analysis for adaptive median filter with different maximum window sizes.
part2_fixed_med_filt.m	Code for performing adaptive median filtering with and presenting the results with quantitative analysis.
part3_inverse.m	Code for inverse filtering
part3_wiener_blur1.m	Code for wiener filtering of the image blur1.bmp
part3_wiener_blur2.m	Code for wiener filtering of the image blur2.bmp
part3_wiener_blur3.m	Code for wiener filtering of the image blur3.bmp
part3_wiener_quant_blur1.m	Code for quantitative analysis of the image blur1.bmp with wiener filtering
part3_wiener_quant_blur2.m	Code for quantitative analysis of the image blur2.bmp with wiener filtering
part3_wiener_quant_blur3.m	Code for quantitative analysis of the image blur3.bmp with wiener filtering

Part 1: Adaptive Local Noise Reduction

MATLAB Code - ALND

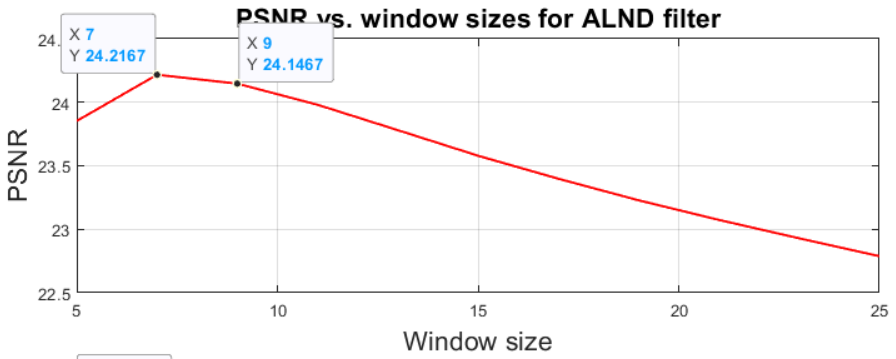
```
clc; clear; close all;f=20;  %Fontsize
OI = imread("circuit.bmp"); OI = im2double(OI);
NI = imread('circuit_gauss001.bmp');NI = im2double(NI);
ws_alnd = 7; % The size of the local window for ALND
ws_am = 5; % The size of the local window for AM
sigma = 1.3; %SD of Gaussian smoothing filter
noise_var = 0.01; %the noise variance
[r, c] = size(NI);
for i = 1:r
    for j = 1:c
        rmin=max(i-floor(ws_alnd/2),1);rmax=min(i+floor(ws_alnd/2),r);
        cmin=max(j-floor(ws_alnd/2),1);cmax=min(j+floor(ws_alnd/2),c);
        curr_window=NI(rmin:rmax, cmin:cmax);
        loc_mean=mean(curr_window(:)); loc_var = var(curr_window(:));
        ALNDI(i,j)=NI(i,j)-(noise_var/loc_var).*(NI(i,j)-loc_mean);
    end
end
AMI = imboxfilt(NI,ws_am); %arithmetic mean filter
GSI = imgaussfilt(NI,sigma); %Gaussian smoothing filter

figure(1);imshow(OI);title('Original image','FontSize',f);
figure(2);imshow(NI);title('Noisy image','FontSize',f);
figure(3);imshow(ALNDI);title('Denoised image by ALND','FontSize',f);
figure(4);imshow(AMI);title('Denoised image by arithmetic mean filter','FontSize',f);
figure(5);imshow(GSI);title('Denoised image by Gaussian smoothing filter','FontSize',f);
```

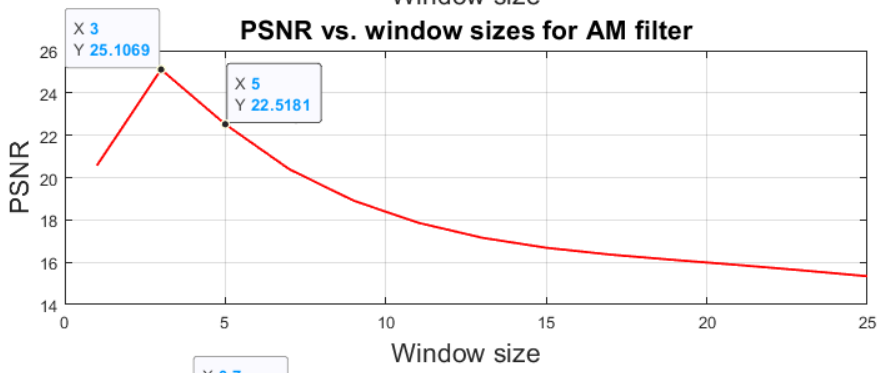
MATLAB Code (continued)

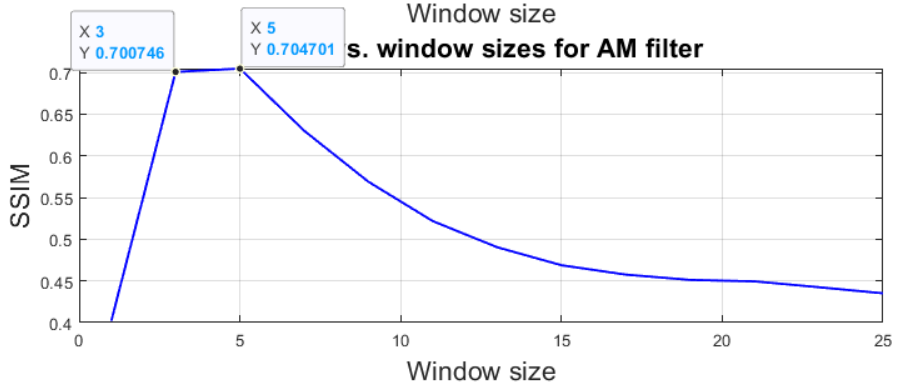
```
%quantitative evaluation
wsq = 1:2:25; sigmaq = linspace(.1,2.5,length(wsq));
for k=1:length(wsq)
    for i=1:r
        for j=1:c
            rmin=max(i-floor(wsq(k)/2), 1);rmax=min(i+floor(wsq(k)/2),r);
            cmin=max(j-floor(wsq(k)/2),1);cmax=min(j+floor(wsq(k)/2),c);
            curr_window=NI(rmin:rmax,cmin:cmax); % Get the current window
            loc_mean=mean(curr_window(:));loc_var = var(curr_window(:));
            ALNDIq(i,j) = NI(i,j) - (noise_var/loc_var).*(NI(i,j) - loc_mean);
        end
    end
    AMIq = imboxfilt(NI,wsq(k));
    GSIq = imgaussfilt(NI,sigmaq(k));
    PSNR_ALND(k) = psnr(OI,ALNDIq);SSIM_ALND(k) = ssim(OI,ALNDIq);
    PSNR_AM(k) = psnr(OI,AMIq);SSIM_AM(k) = ssim(OI,AMIq);
    PSNR_GS(k) = psnr(OI,GSIq);SSIM_GS(k) = ssim(OI,GSIq);
end
figure(6);subplot(321);plot(wsq,PSNR_ALND,'r',LineWidth=1.4);xlabel("Window size",FontSize=f-4);grid on;
ylabel('PSNR',FontSize=f-4); title("PSNR vs. window sizes for ALND filter",FontSize=f-4);
subplot(322);plot(wsq,SSIM_ALND,'b',LineWidth=1.4);xlabel("Window size",FontSize=f-4);grid on;
ylabel('SSIM',FontSize=f-4); title("SSIM vs. window sizes for ALND filter",FontSize=f-4);
subplot(323);plot(wsq,PSNR_AM,'r',LineWidth=1.4);xlabel("Window size",FontSize=f-4);grid on;
ylabel('PSNR',FontSize=f-4); title("PSNR vs. window sizes for AM filter",FontSize=f-4);
subplot(324);plot(wsq,SSIM_AM,'b',LineWidth=1.4);xlabel("Window size",FontSize=f-4);grid on;
ylabel('SSIM',FontSize=f-4); title("SSIM vs. window sizes for AM filter",FontSize=f-4);
subplot(325);plot(sigmaq,PSNR_GS,color=[0 0.5 0],LineWidth=1.4);xlabel("\sigma",FontSize=f-4);grid on;
ylabel('PSNR',FontSize=f-4); title("PSNR vs. \sigma for Gaussian smoothing filter",FontSize=f-4);
subplot(326);plot(sigmaq,SSIM_GS,'m',LineWidth=1.4);xlabel("\sigma",FontSize=f-4);grid on;
ylabel('SSIM',FontSize=f-4); title("SSIM vs. \sigma for Gaussian smoothing filter",FontSize=f-4);
psnr_noisy_im = psnr(OI,NI)
ssim_noisy_im = ssim(OI,NI)
```

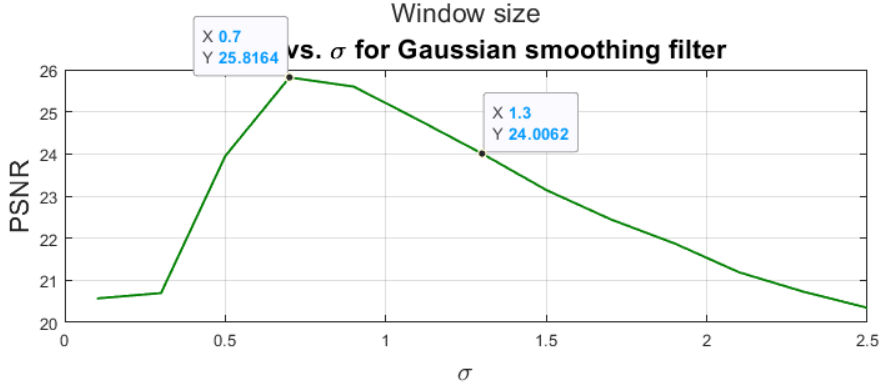
Quantitative Analysis (Using PSNR and SSIM)

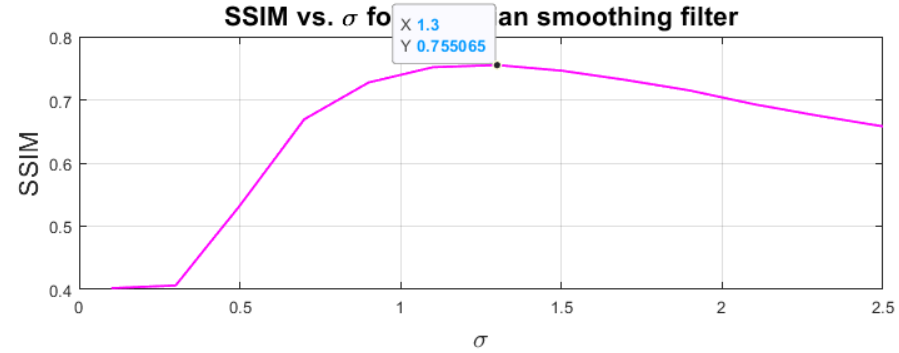
PSNR and SSIM have been plotted against window sizes (for AM and ALND) or standard deviation (for Gaussian filter), in order to choose the best possible parameters.





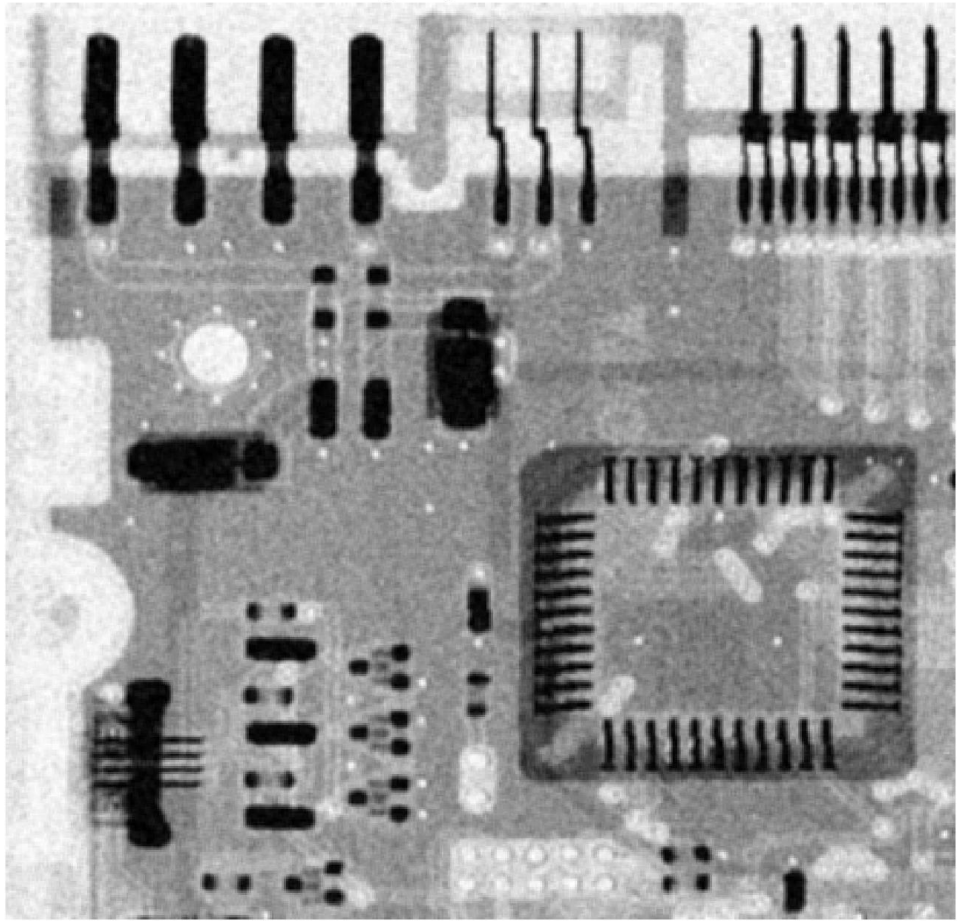






Arithmetic Mean Filtering Results

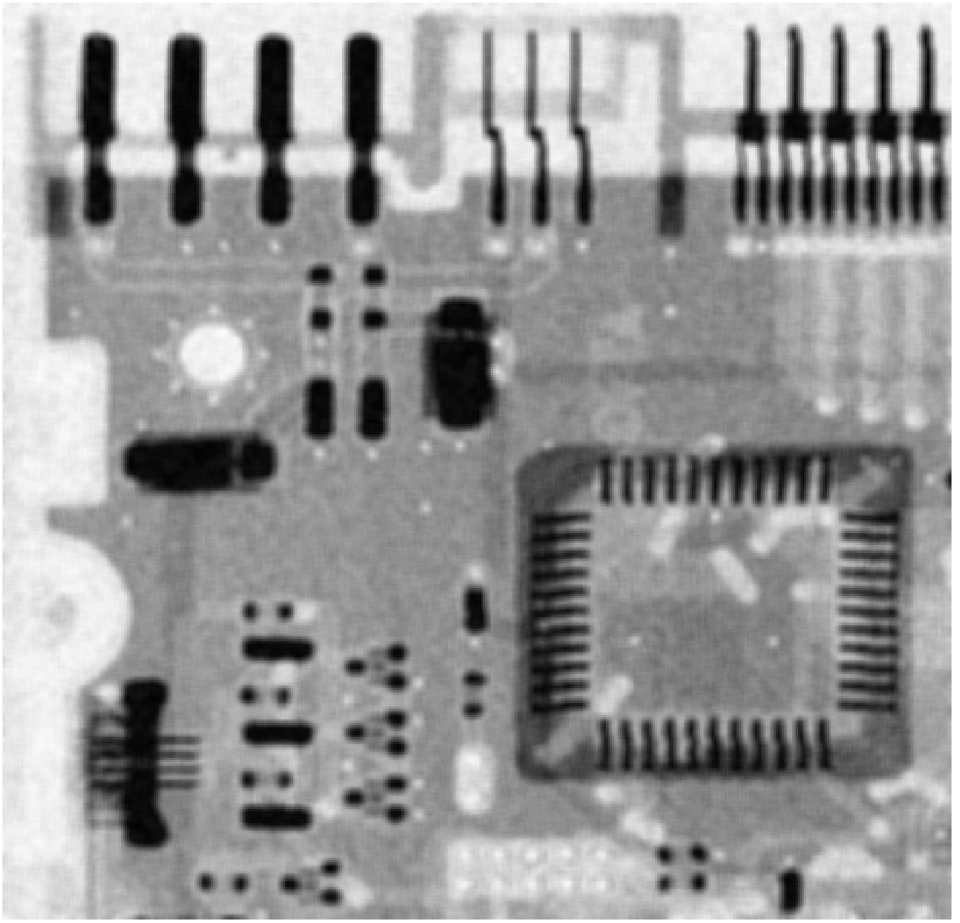
Denoised image by arithmetic mean filter



Window size = 3 (Maximum PSNR occurs here)

PSNR = 25.1069 dB, SSIM = 0.7

Denoised image by arithmetic mean filter



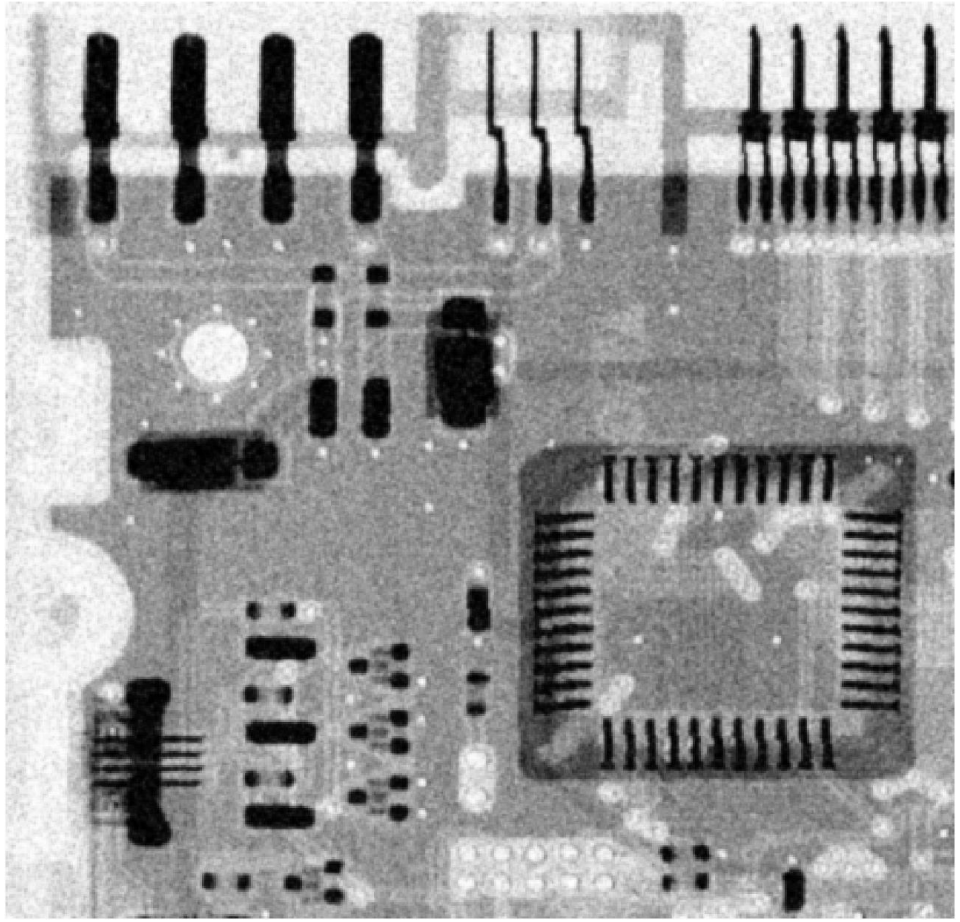
Window size = 5 (Maximum SSIM occurs here)

PSNR = 22.5181 dB, SSIM = 0.705

Although window size 3 is giving the best PSNR, it is not the best one visually, because of noisy background. Window size = 5 gives the highest SSIM, with a bit lower PSNR, but visually looks better. So, window size 5 has been chosen as the best option.

Gaussian Smoothing Filtering Results

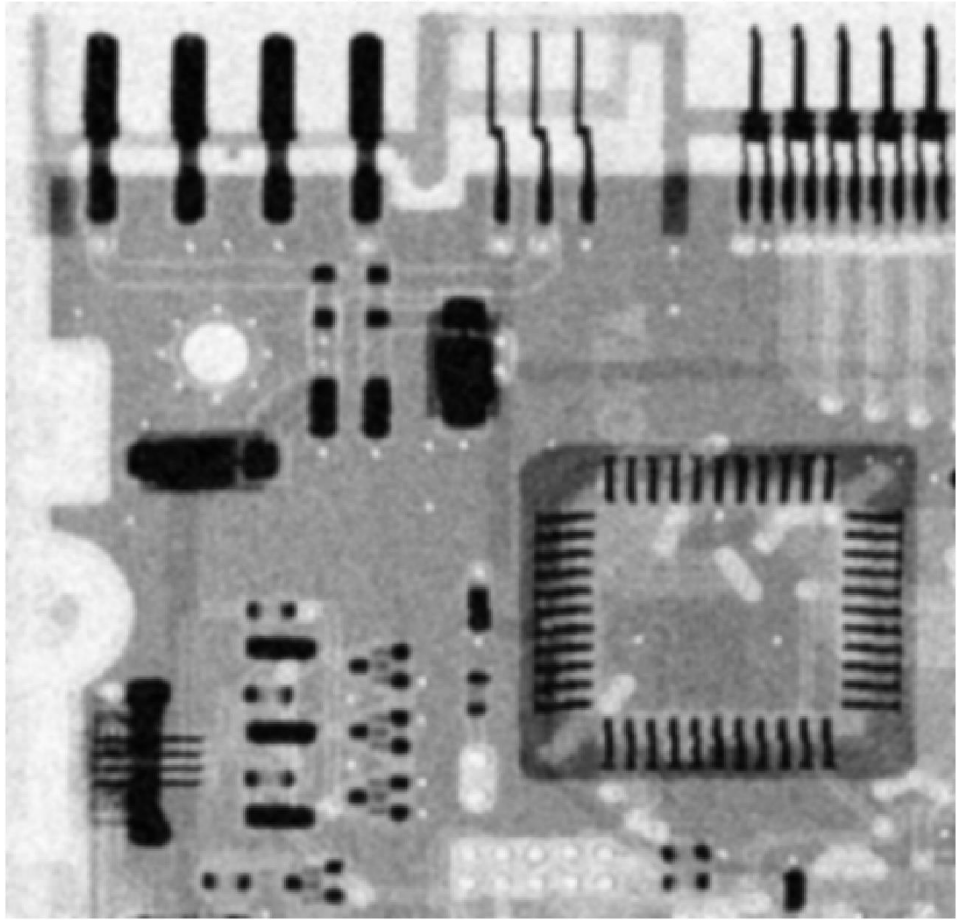
Denoised image by Gaussian smoothing filter



Sigma = 0.7 (Maximum PSNR occurs here)

PSNR = 25.8164 dB, SSIM = 0.669

Denoised image by Gaussian smoothing filter

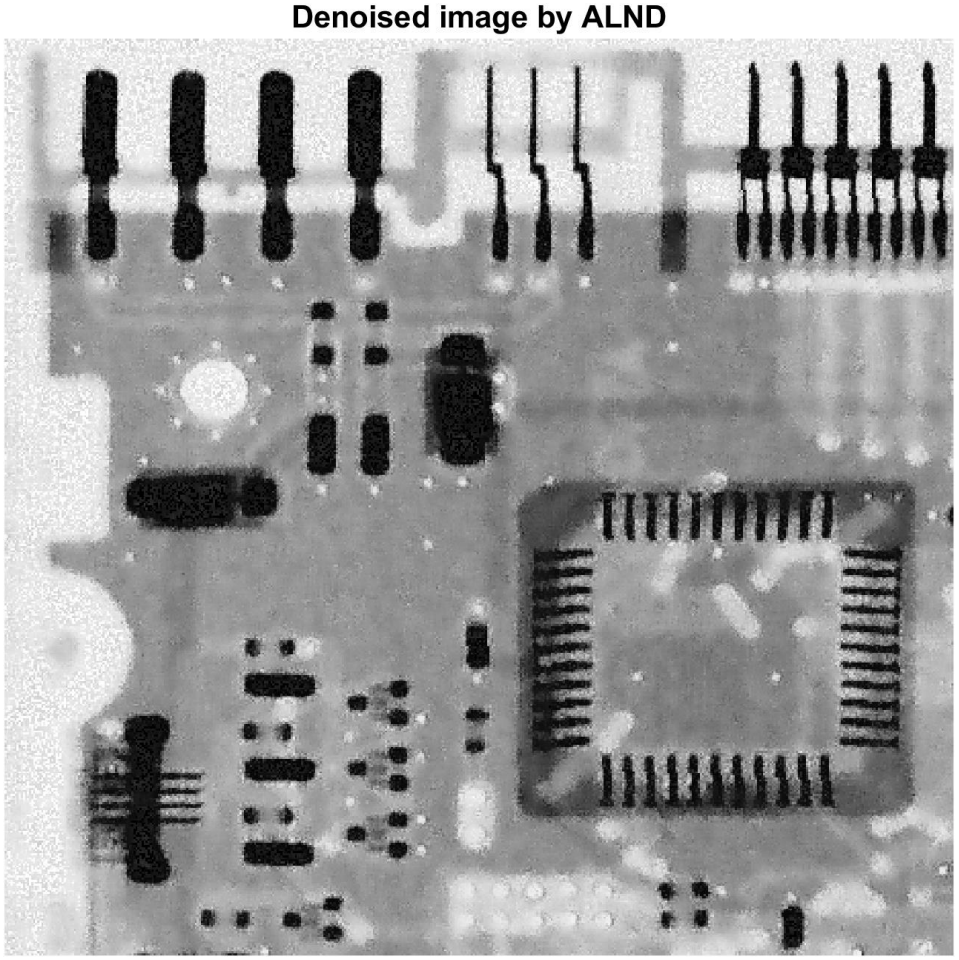


Sigma = 1.3 (Maximum SSIM occurs here)

PSNR = 24.0062 dB, SSIM = 0.755

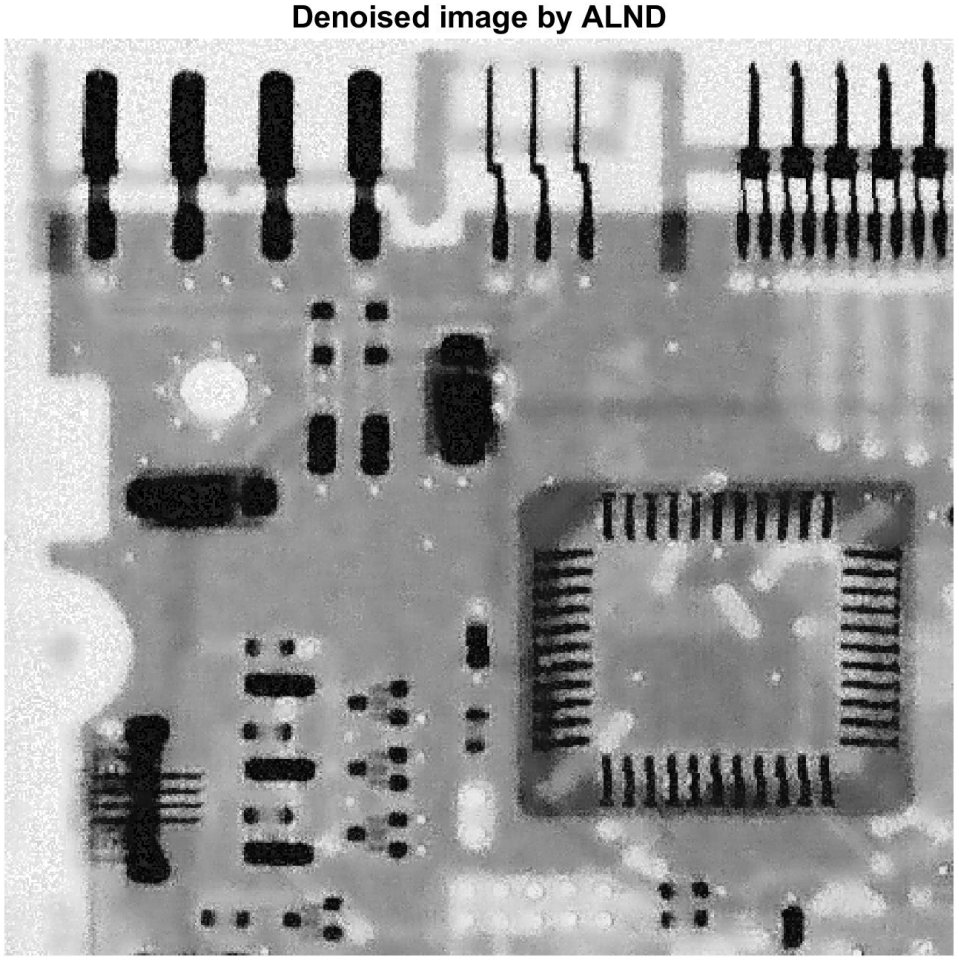
Sigma = 3 is producing the best PSNR but SSIM is not as high. Background noise is clearly visible with Sigma = 3. Sigma = 1.3 is producing the smoother image with the highest SSIM, hence this one is chosen as the best option.

Adaptive Local Noise Denoising Results



ALND Local window size = 7 (Maximum PSNR occurs here)

PSNR = 24.2167 dB, SSIM = 0.6174



ALND Local window size = 9 (Maximum SSIM occurs here)

PSNR = 24.1467 dB, SSIM = 0.6285

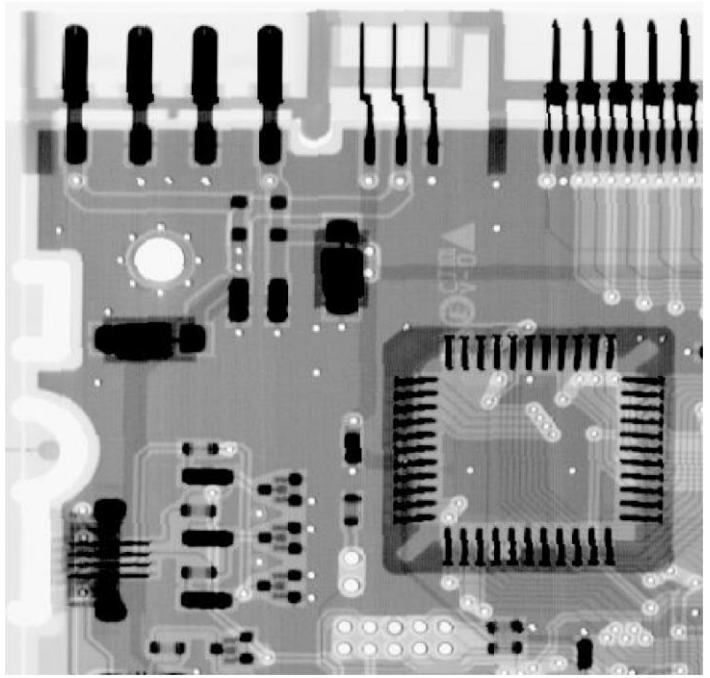
They are almost the same. Both have almost same PSNR. The one with higher SSIM (window size = 9) gives slightly better visual perception, as expected. So, window size 9 has been chosen as the best option.

Comparison among best results

$PSNR_{ALND} > PSNR_{Gauss} > PSNR_{AM}$
$SSIM_{Gauss} > SSIM_{AM} > SSIM_{ALND}$

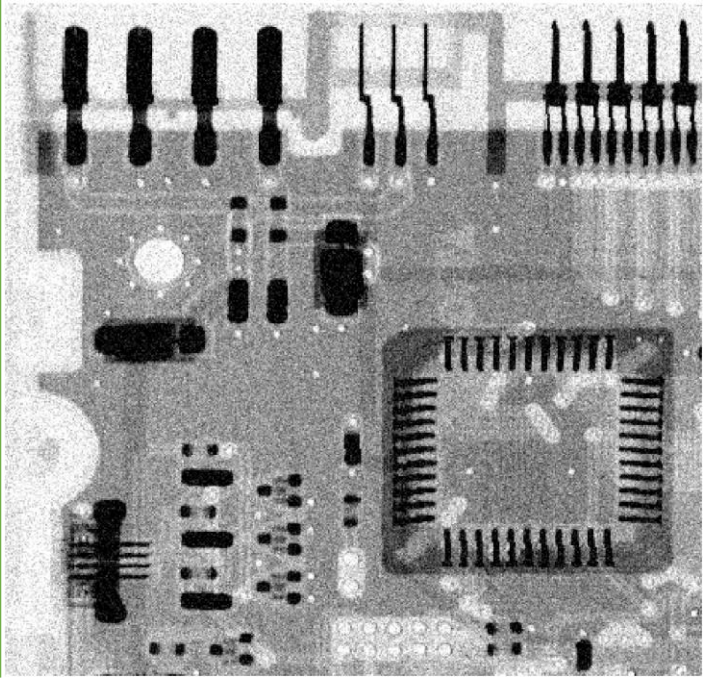
ws_alnd = 9; % The size of the local window for ALND
ws_am = 5; % The size of the local window for AM
sigma = 1.3; %SD of Gaussian smoothing filter

Original image



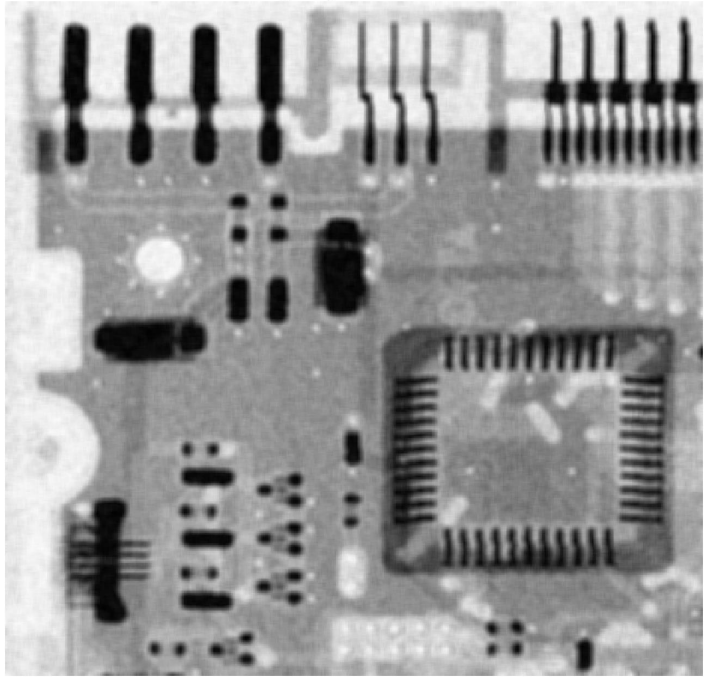
Noisy image

PSNR = 20.5691 dB, SSIM = 0.402



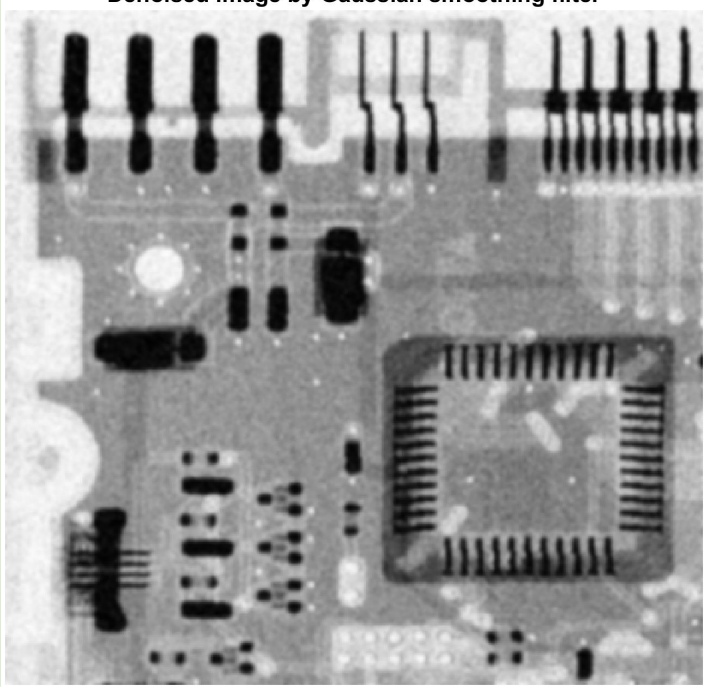
Denoised image by arithmetic mean filter

PSNR = 22.5181 dB, SSIM = 0.705



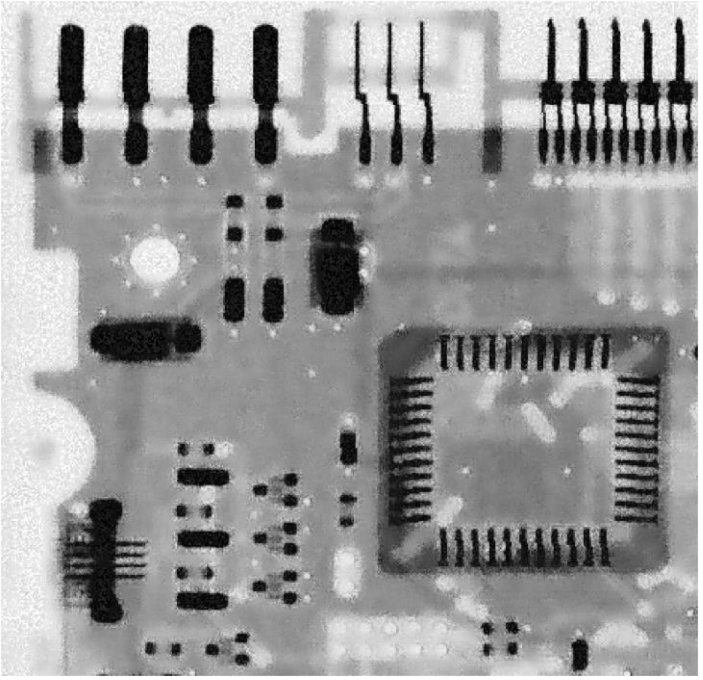
Denoised image by Gaussian smoothing filter

PSNR = 24.0062 dB, SSIM = 0.755



Denoised image by ALND

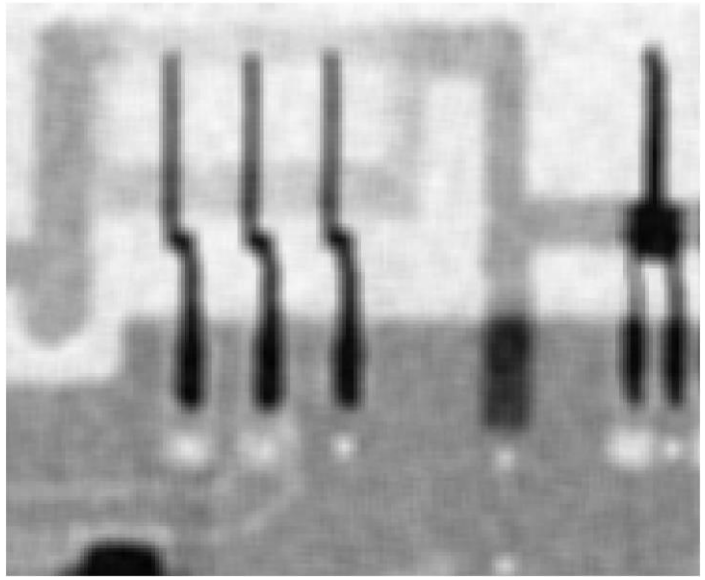
PSNR = 24.1467 dB, SSIM = 0.6285



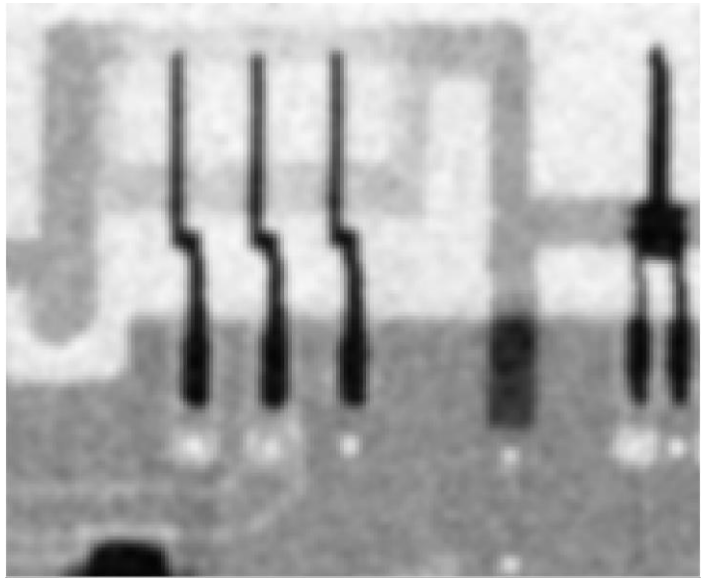
Discussion

$PSNR_{ALND} > PSNR_{Gauss} > PSNR_{AM}$
$SSIM_{Gauss} > SSIM_{AM} > SSIM_{ALND}$

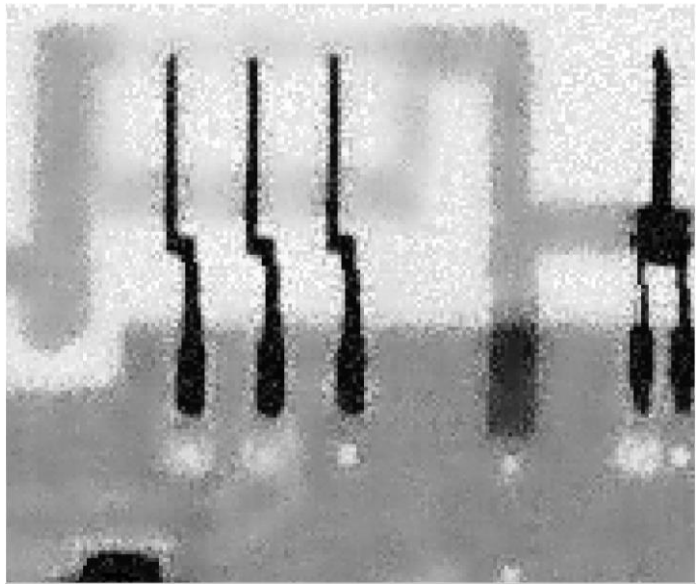
1. ALND is giving the best result in terms of PSNR. The SSIM values are a bit counter-intuitive, since peak SSIM for ALND is less than both of Gaussian and AM. But still visually ALND is producing the most appealing result with sharpness not only at the edges, but also everywhere in the image.
2. AM and Gaussian filter is producing blurry results, although Gaussian one is better than the AM one, since Gaussian filter puts more weight on the pixels closer to the current pixel, while doing averaging.
3. The lower SSIM with ALND might be attributed to the artifacts in the background and protrusions from the structural edges (visible when zoomed in). AM and Gaussian filter are producing blurry and hence less visually appealing images, but they don't have that kind of artifacts or protrusions, thus producing higher structural similarity with the original image.



AM filter



Gaussian filter



ALND filter

Part 2: Adaptive Median Filter

MATLAB Code – Median Filter with Fixed Window Size

```
clc; clear; close all;
f=16;  %Fontsize
orig_image = imread("circuit.bmp"); orig_image = im2double(orig_image);
noisy_image = imread('circuit_sp05.bmp');
noisy_image = im2double(noisy_image);
wsq = 1:2:25;
for i=1:length(wsq)
    denoised_medfilt2 = medfilt2(noisy_image, [wsq(i) wsq(i)],'symmetric');
    PSNR_medfilt2(i) = psnr(orig_image,denoised_medfilt2);
    SSIM_medfilt2(i) = ssim(orig_image,denoised_medfilt2);
end

figure(1);subplot(121);plot(wsq,PSNR_medfilt2,'r',LineWidth=1.4);xlabel("Window size",FontSize=f);grid on;
ylabel('PSNR',FontSize=f-4); title("PSNR vs. window sizes for fixed-size median filter",FontSize=f);
subplot(122);plot(wsq,SSIM_medfilt2,'b',LineWidth=1.4);xlabel("Window size",FontSize=f);grid on;
ylabel('SSIM',FontSize=f-4); title("SSIM vs. window sizes for fixed-size median filter",FontSize=f);

psnr_noisy_im = psnr(orig_image,noisy_image)
ssim_noisy_im = ssim(orig_image,noisy_image)

ws = 7;
denoised_medfilt2 = medfilt2(noisy_image, [ws ws],'symmetric');
figure(2);imshow(noisy_image);title('Noisy Image',FontSize=f);
figure(3);imshow(denoised_medfilt2);title('Denoised Image using medfilt2',FontSize=f);
```

Window size is varied and PSNR and SSIM were calculated for each case

Symmetric padding was used to remove the artifacts from the border caused by zero-padding

Results are being displayed for window size = 7

MATLAB Code : Adaptive Median Filter – Quantitative Analysis

Maximum window size Smax has been varied and PSNR and SSIM were calculated and stored for each cases.

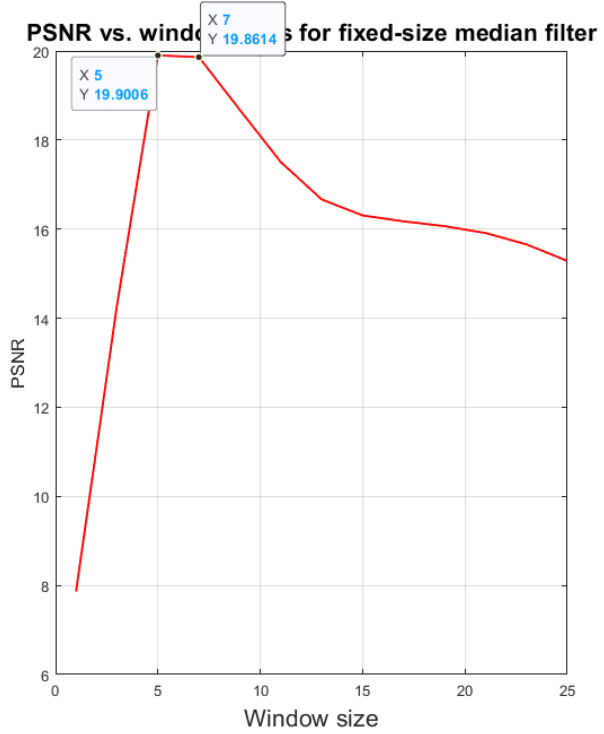
```
clc; clear; close all; f=16; OI=imread("circuit.bmp"); OI=im2double(OI);
NI=imread('circuit_sp05.bmp'); NI=im2double(NI); Smax=3:2:61; [r,c]=size(NI);
for k=1:length(Smax)
    FI=zeros(r,c);
    for i=1:r
        for j=1:c
            S=3;
            while (S<=Smax(k))
                rmin=max(i-floor(S/2),1); rmax=min(i+floor(S/2),r);
                cmin=max(j-floor(S/2),1); cmax=min(j+floor(S/2),c);
                curr_window=NI(rmin:rmax,cmin:cmax);
                Zmin=min(curr_window(:)); Zmax=max(curr_window(:)); Zmed=median(curr_window(:));
                A1=Zmed-Zmin; A2=Zmed-Zmax;
                if (A1>0&&A2<0)
                    B1=NI(i,j)-Zmin; B2=NI(i,j)-Zmax;
                    if(B1>0&&B2<0)
                        FI(i,j)=NI(i,j);
                    else
                        FI(i,j)=Zmed;
                    end
                    break; % Exit the inner loop (while loop)
                end
                S=S+2;
                if (S>Smax(k))
                    FI(i,j)=NI(i,j);
                end
            end
        end
    end
    psnr_FI(k)=psnr(OI,FI); ssim_FI(k)=ssim(OI,FI);
end
figure(1);subplot(121);plot(Smax,psnr_FI,'r',LineWidth=1.4);xlabel("Max window size",FontSize=f);grid on;
ylabel('PSNR',FontSize=f-4); title("PSNR vs. max window sizes for adaptive median filter",FontSize=f);
subplot(122);plot(Smax,ssim_FI,'b',LineWidth=1.4);xlabel("Max window size",FontSize=f);grid on;
ylabel('SSIM',FontSize=f-4); title("SSIM vs. max window sizes for adaptive median filter",FontSize=f);
```

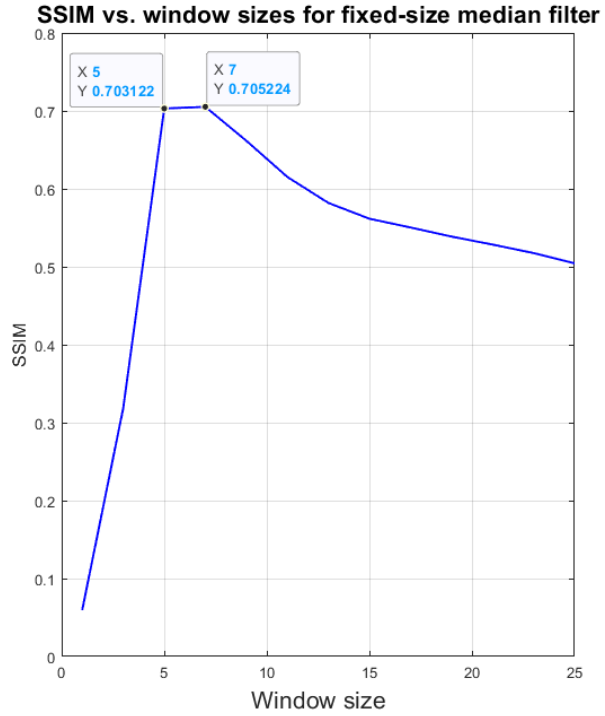
MATLAB Code : Adaptive Median Filter – window size (max) = 43

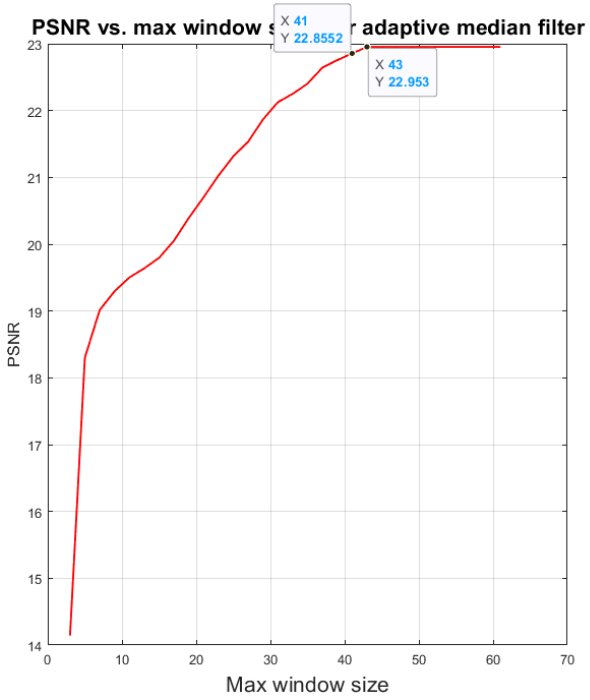
```
clc; clear; close all; f=16;
OI=imread("circuit.bmp"); OI=im2double(OI);
NI=imread('circuit_sp05.bmp'); NI=im2double(NI);
Smax=43; [r,c]=size(NI); FI=zeros(r,c);
for i=1:r
    for j=1:c
        S=3;
        while (S<=Smax)
            rmin=max(i-floor(S/2),1); rmax=min(i+floor(S/2),r);
            cmin=max(j-floor(S/2),1); cmax=min(j+floor(S/2),c);
            curr_window=NI(rmin:rmax,cmin:cmax);
            Zmin=min(curr_window(:)); Zmax=max(curr_window(:));
            Zmed=median(curr_window(:));
            A1=Zmed-Zmin; A2=Zmed-Zmax;
            if (A1>0 && A2<0)
                B1=NI(i,j)-Zmin; B2=NI(i,j)-Zmax;
                if (B1>0 && B2<0)
                    FI(i,j)=NI(i,j);
                else
                    FI(i,j)=Zmed;
                end
                break;
            end
            S=S+2;
            if (S>Smax)
                FI(i,j)=NI(i,j);
            end
        end
    end
end
figure(1); imshow(NI); title('Noisy Image',FontSize=f);
figure(2); imshow(FI);
title('Denoised Image using adaptive median filter',FontSize=f);
psnr_FI=psnr(OI,FI)
ssim_FI=ssim(OI,FI)
```

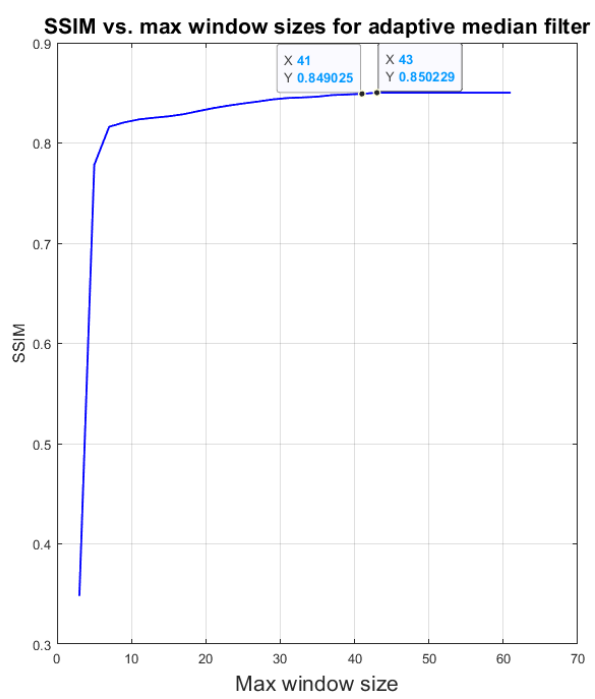
Quantitative Analysis
(Using PSNR and SSIM)

PSNR and SSIM have been plotted against window sizes (for fixed median filter) or maximum window sizes (for adaptive median filter), in order to choose the best possible parameters.

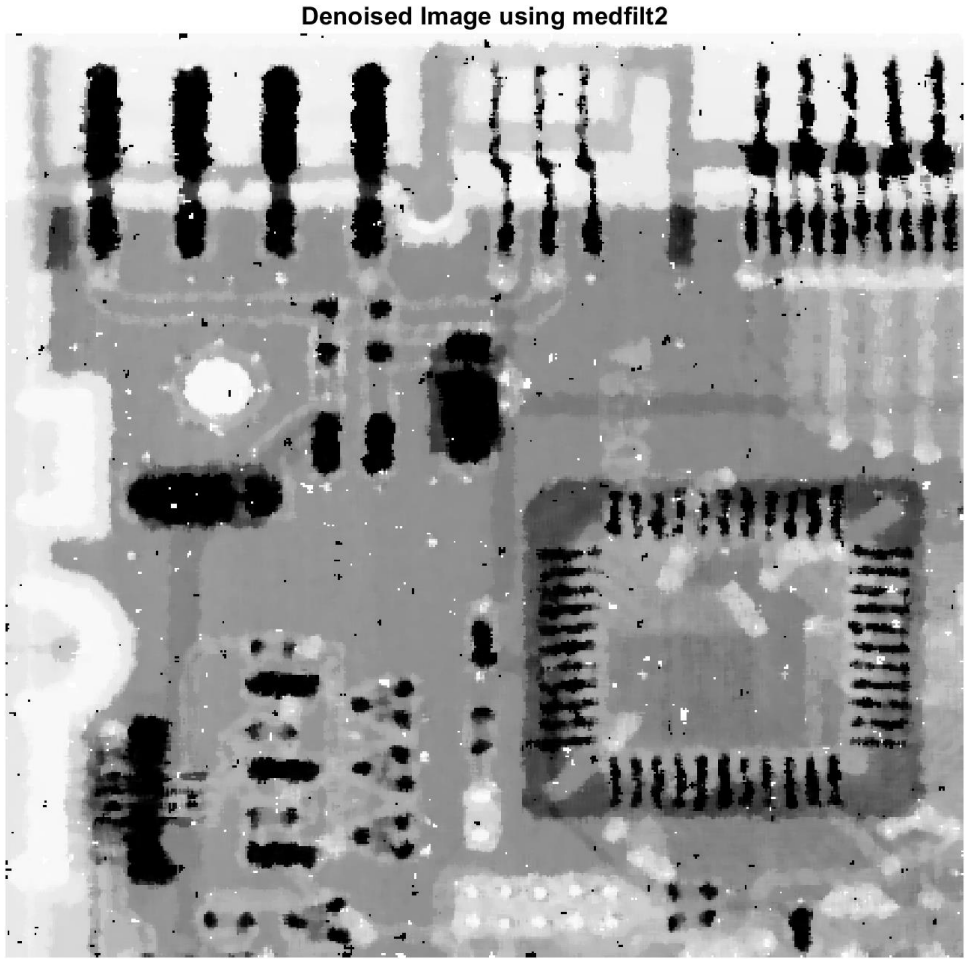






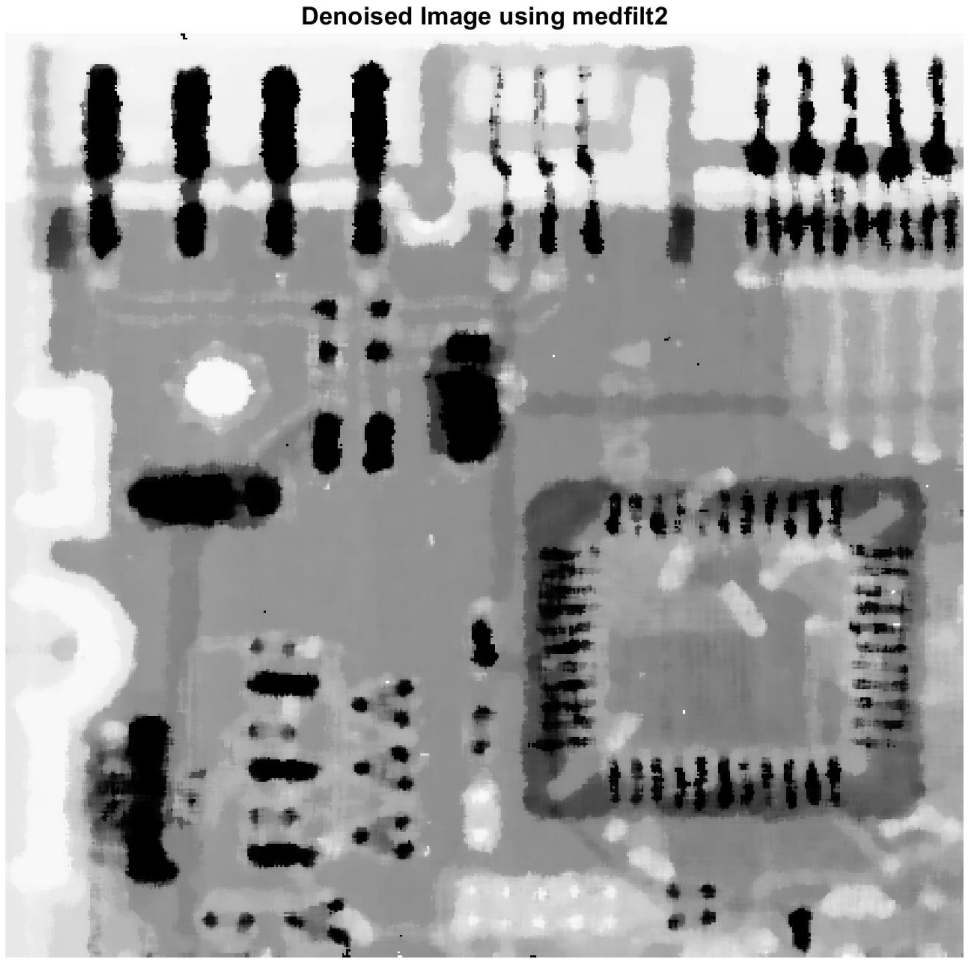


Fixed Median Filter Results



Window size = 5 (Maximum PSNR occurs here)

PSNR = 19.9006 dB, SSIM = 0.7031

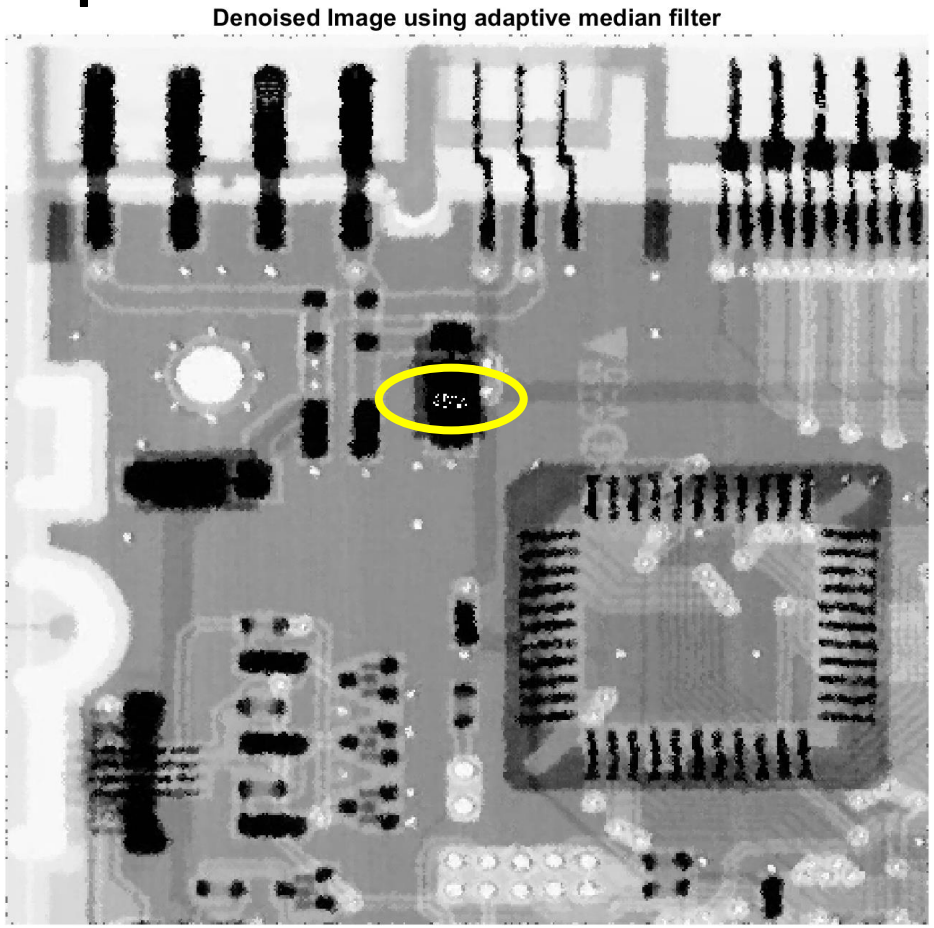


Window size = 7 (Maximum SSIM occurs here)

PSNR = 19.8614 dB, SSIM = 0.7052

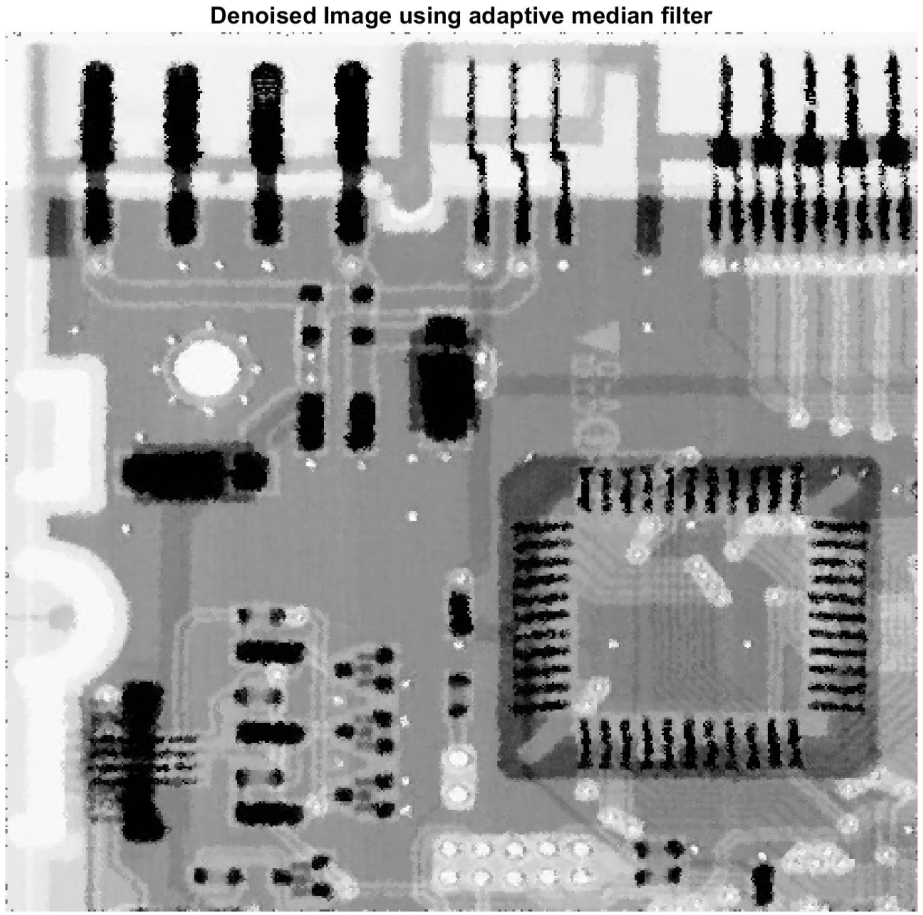
Window size 5, where PSNR reaches its maximum, retains some noise components (black and white dots or spots) in the image. With window size 7, the structural integrity deteriorates slightly; but the noise is almost completely suppressed, producing a higher SSIM, hence we will take that.

Adaptive Median Filter Results



Max window size = 41

PSNR = 22.8552 dB, SSIM = 0.849



Max window size = 43

PSNR = 22.953 dB, SSIM = 0.85

The peak PSNR and SSIM occurs at maximum window size 43. When windows size is less than that, then some white artifacts are noticed on the black islands of the image (shown in the left image inside a yellow contour). If window size is higher than 43, then no noticeable change occurs because of the adaptive nature of the filter which might cause window sizes higher than certain value (like 43) completely remain unused.

Best Results & Discussion

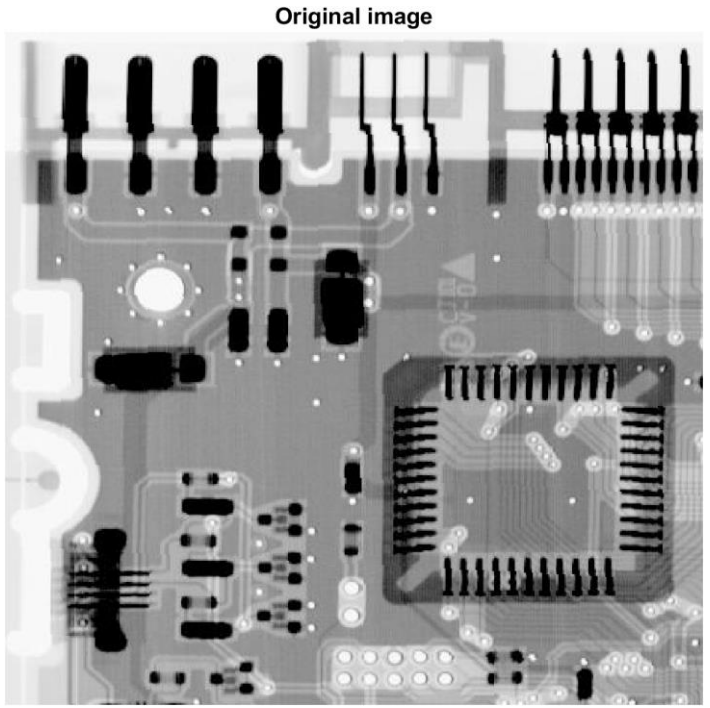
$PSNR_{adaptive} > PSNR_{fixed}$
$SSIM_{adaptive} > SSIM_{fixed}$

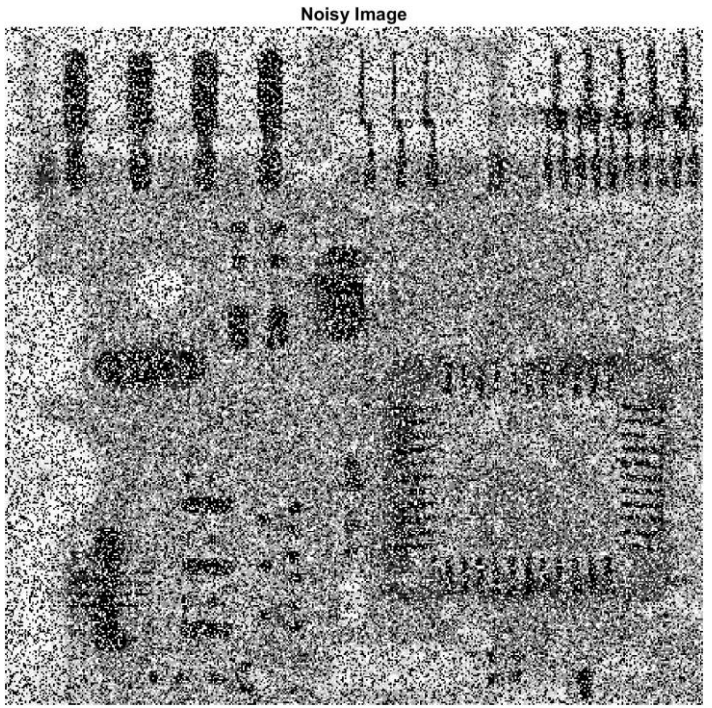
Window size for fixed median filter = 7

Max window size for adaptive median filter = 43

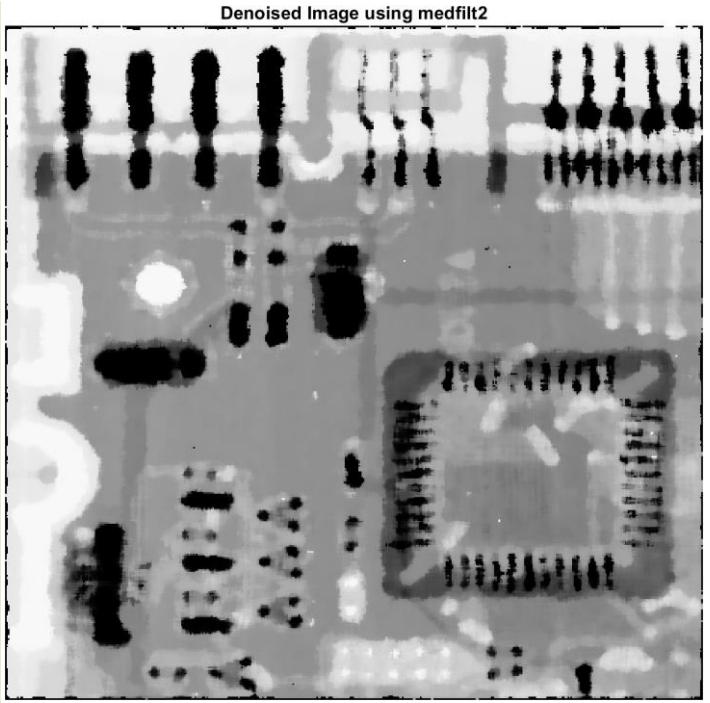
1. With fixed window-based median filter, many structures in the image became faint and disconnected.

2. But, after using adaptive window size in median filtering, the image quality has been significantly improved, which is evident both visually and through improved PSNR and, more importantly, SSIM values.

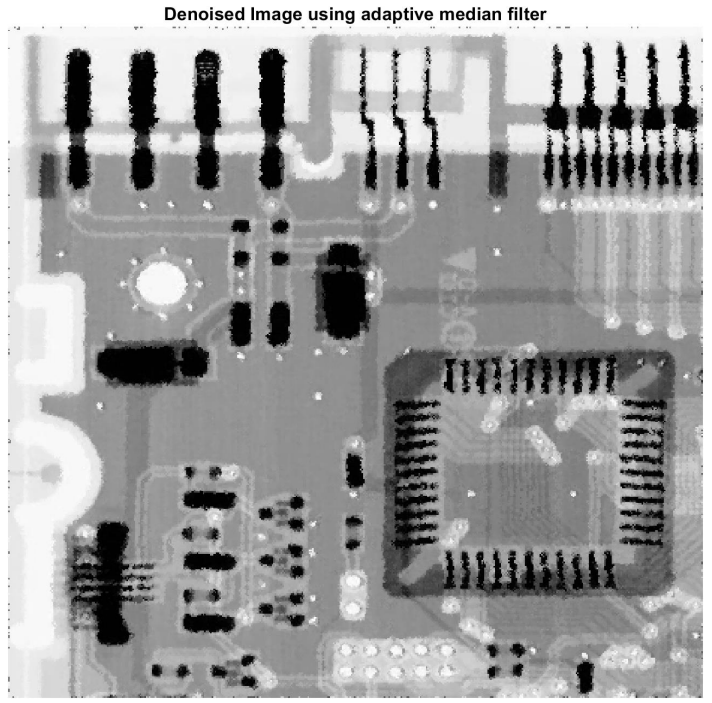




PSNR = 7.8655 dB, SSIM = 0.0594



PSNR = 19.8614 dB, SSIM = 0.7052



PSNR = 22.953 dB, SSIM = 0.8502

Part 3: Wiener Filtering

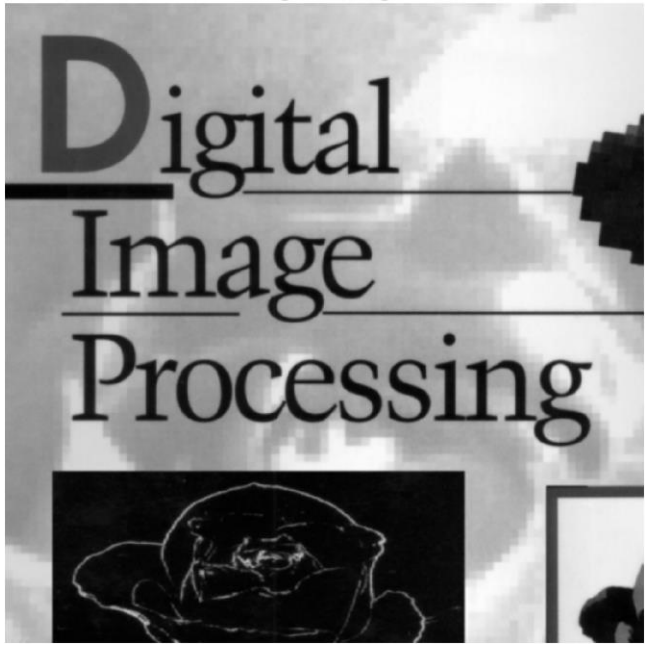
MATLAB code - Inverse filtering

```
clc; clear; close all; f = 20;
OI = im2double(imread('original.bmp'));BI = im2double(imread('blur1.bmp'));
psnr_BI = psnr(OI,BI); ssim_BI = ssim(OI,BI);
figure(1), imshow(OI), title('Original Image');
figure(2), imshow(BI),
title(sprintf('Blurred Image (PSNR=%.2fdB, SSIM=%.2f)',psnr_BI,ssim_BI),FontSize=f);
[M, N] = size(BI);
BIF=fftshift(fft2(BI));
a = 0.1; b = 0.1; T = 1; IF=ones(M,N);
for u=1:M
     for v=1:N
         uu=u-M/2-1;vv=v-N/2-1;
         t = uu*a+vv*b;
         H(u,v) = T*sinc(t)*exp(-1j*pi*t);
         if abs(H(u,v))>.2
            IF(u,v)=1/abs(H(u,v));
         end
     end
end
RDF=BIF.*IF; RDI=abs(ifft2(ifftshift(RDF)));RDI=RDI/max(max(RDI));
psnr_IF=psnr(RDI,OI); ssim_IF=ssim(RDI,OI);
figure(3); imshow(abs(H),[]),title('Degradation function');
figure(4); imshow(IF,[]),title('Inverse degradation function');
figure(5), imshow(RDI),
title(sprintf('After Inverse FIltering (PSNR=%.2fdB, SSIM=%.2f)',psnr_IF,ssim_IF),FontSize=f);
```

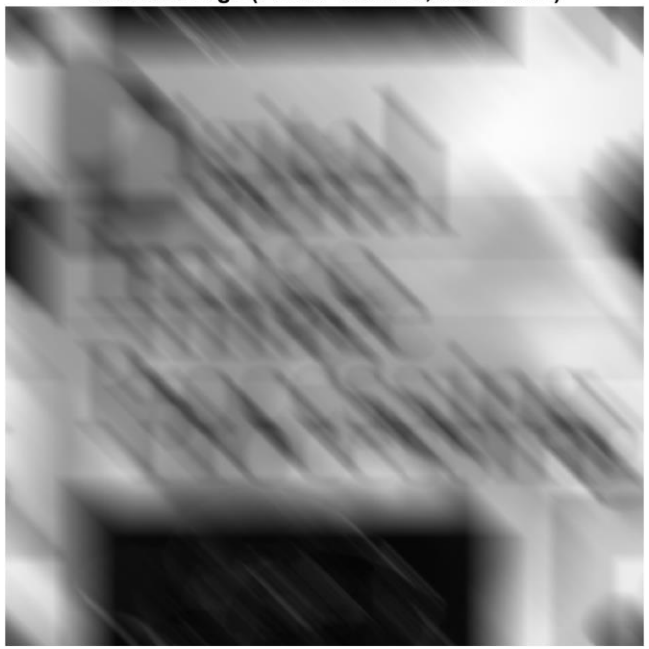
This is for blur1.bmp. Image name can be changed to do the same for other images.

Inverse Filtering Results with blur1.bmp

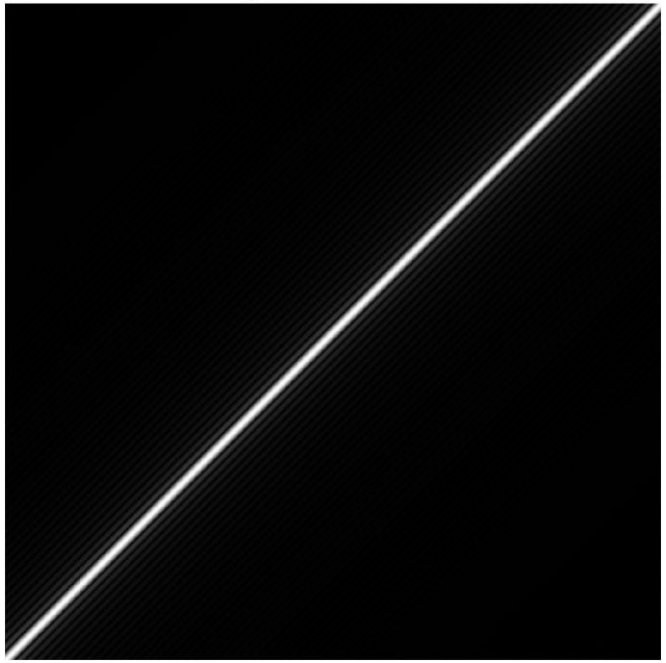
Original Image



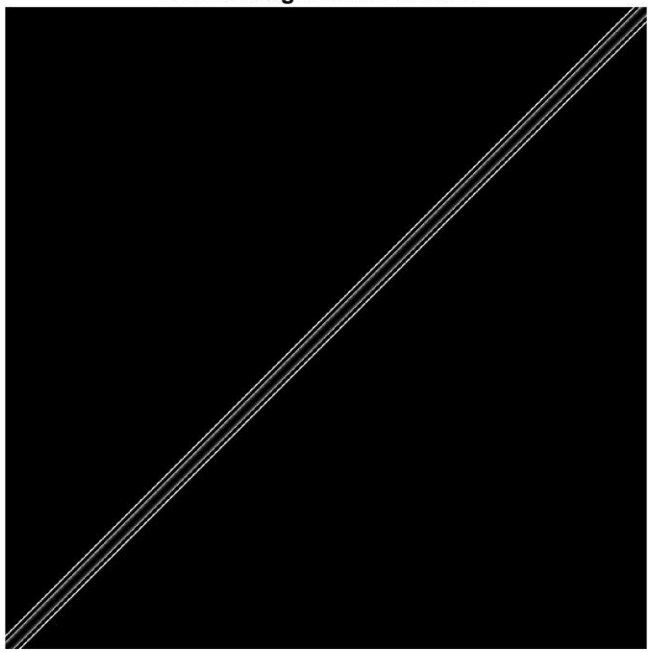
Blurred Image (PSNR=11.81dB, SSIM=0.54)



Degradation function



Inverse degradation function



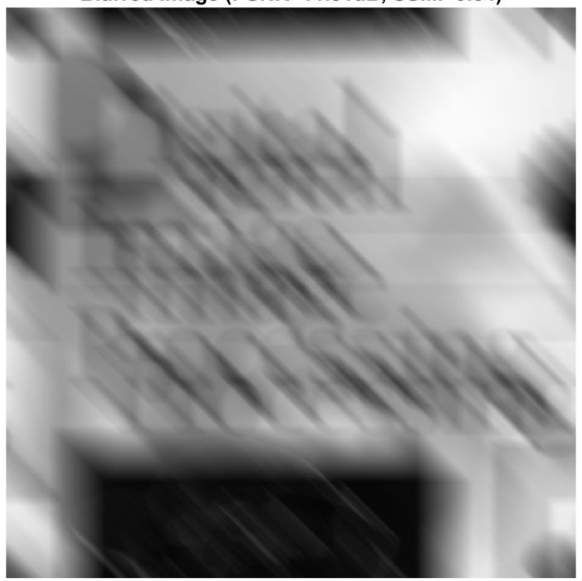
After Inverse FIltering (PSNR=10.94dB, SSIM=0.48)

Threshold (explained later) = 0.2 for inverse filtering

Inverse Filtering Results with blur 1, blur2 and blur3.bmp

For Blur1.bmp

Blurred Image (PSNR=11.81dB, SSIM=0.54)

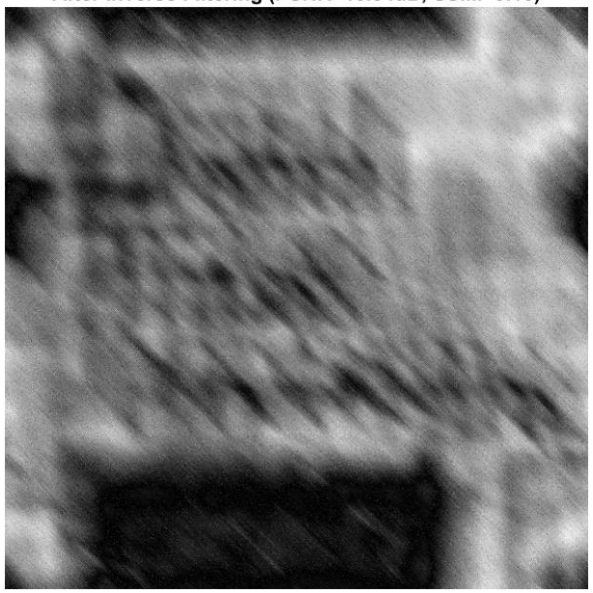


After Inverse FIltering (PSNR=10.94dB, SSIM=0.48)

For Blur2.bmp

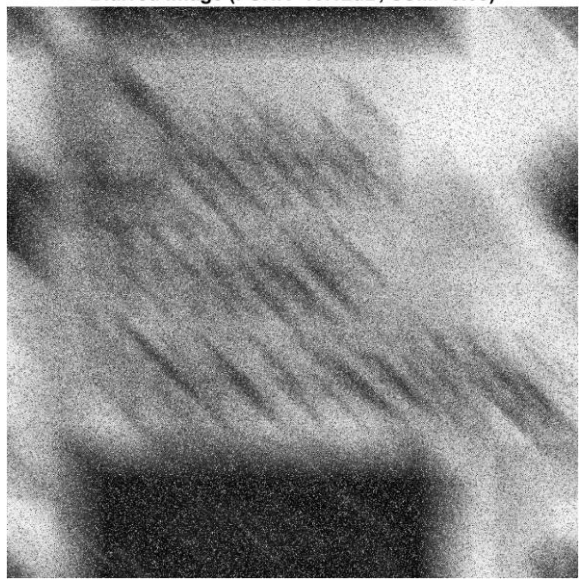
Blurred Image (PSNR=11.53dB, SSIM=0.13)

After Inverse FIltering (PSNR=10.54dB, SSIM=0.16)

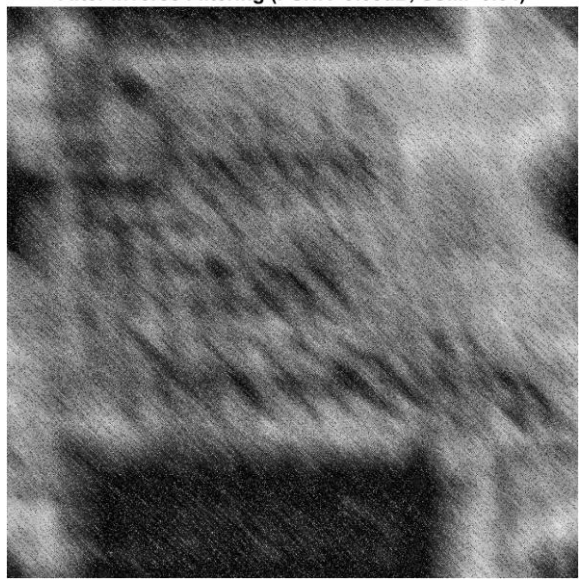


For Blur3.bmp

Blurred Image (PSNR=10.12dB, SSIM=0.03)



After Inverse FIltering (PSNR=9.63dB, SSIM=0.04)



Discussion on Inverse Filtering

- Even if we know the degradation model exactly, full inverse filtering will not generate good result because:
  - The degradation might be lossy
  - When the transfer function has very low value then it gives high value in the filtered image, resulting in magnification of noise
- However, inverse filtering results can be improved by doing it in a limited region in the frequency domain, for example: within a circle of fixed radius from the center.
- In the given code as help for Project 4, the degradation function (atmospheric turbulence) was seen to have high values within a circle around the center. Hence, inverse filtering within such a circle provided better result. However, the degradation function of motion blur has a diagonal effect, the values are high in a diagonal window, hence circular low pass filtering might not be the best in such case.
- In order to do the inverse filtering, a threshold is used (threshold = 0.2 in the code, found by trial and error). Inverse filtering is performed only when the magnitude of degradation function H(u,v) is greater than that threshold. Playing with the threshold value around 0.2 changed the results very slightly, hence we used 0.2 all the time. Increasing or decreasing the threshold a lot resulted in inverse filtering in very large area (including small values) or very small area (poor filtering), hence the results became worse.
- The result was not good, as expected and was even worse than the initial blurred and noisy images. Later, we showed that Wiener filtering with prefiltering does better restoration.
- Ideally, for motion blur, the inverse filtering should have been done in a diagonal rectangle or ellipse. But it needs rigorous math and the result should not be dramatically better with inverse filtering, no matter what. Hence, such windowing is not performed here. However, after the thresholding, it has been seen that the inverse degradation function has a diagonal shape as expected.

MATLAB Code : Wiener Filter– Quantitative Analysis – for blur2.bmp

```
clc; clear; close all; f = 16;OI = im2double(imread('original.bmp'));BI = im2double(imread('blur2.bmp'));
bwidth = 7; gsd = 0.8; mwidth= 9;BIPB = imboxfilt(BI,bwidth);BIPG = imgaussfilt(BI,gsd);BIPM = medfilt2(BI, [mwidth mwidth],'symmetric');
psnr_BI = psnr(OI,BI); ssim_BI = ssim(OI,BI);psnr_BIPB = psnr(OI,BIPB); ssim_BIPB = ssim(OI,BIPB);psnr_BIPG = psnr(OI,BIPG); ssim_BIPG = ssim(OI,BIPG);
psnr_BIPM = psnr(OI,BIPM); ssim_BIPM = ssim(OI,BIPM);
figure(1), subplot(231);imshow(OI), title('Original image',FontSize=f-2);
subplot(232); imshow(BI),title(sprintf('Blurred & noisy image \n (PSNR=%.3fdB, SSIM=%.3f)',psnr_BI,ssim_BI),FontSize=f-2);
subplot(233); imshow(BIPB),title(sprintf('Prefiltered - box filter (width = %d) \n (PSNR=%.3fdB, SSIM=%.3f)',bwidth, psnr_BIPB,ssim_BIPB),FontSize=f-2);
subplot(234); imshow(BIPG),title(sprintf('Prefiltered - Gaussian filter (SD = %0.1f) \n (PSNR=%.3fdB, SSIM=%.3f)',gsd, psnr_BIPG,ssim_BIPG),FontSize=f-2);
subplot(235); imshow(BIPM),title(sprintf('Prefiltered - median filter (width = %d) \n (PSNR=%.3fdB, SSIM=%.3f)',mwidth,psnr_BIPM,ssim_BIPM),FontSize=f-2);
[M, N] = size(BI);BIF=fftshift(fft2(BI));BIFPB=fftshift(fft2(BIPB));BIFPG=fftshift(fft2(BIPG));BIFPM=fftshift(fft2(BIPM));
a = 0.1; b = 0.1; T = 1; WF=ones(M,N); K = .02:.08:1.5;
for p=1:length(K)
    for u=1:M
        for v=1:N
            uu=u-M/2-1;vv=v-N/2-1;
            t = uu*a+vv*b;
            H(u,v) = T*sinc(t)*exp(-1j*pi*t);
            WF(u,v) = (1/H(u,v))*(abs(H(u,v))^2)/(abs(H(u,v))^2+K(p)^2);
        end
    end
    RDF=BIF.*WF; RDI=abs(ifft2(ifftshift(RDF)));RDI=RDI/max(max(RDI));RDFB=BIFPB.*WF; RDIB=abs(ifft2(ifftshift(RDFB)));RDIB=RDIB/max(max(RDIB));
    RDFG=BIFPG.*WF; RDIG=abs(ifft2(ifftshift(RDFG)));RDIG=RDIG/max(max(RDIG));
    RDFM=BIFPM.*WF; RDIM=abs(ifft2(ifftshift(RDFM)));RDIM=RDIM/max(max(RDIM));
    psnr_FI(p) = psnr(OI,RDI); ssim_FI(p) = ssim(OI,RDI);psnr_FIB(p) = psnr(OI,RDIB); ssim_FIB(p) = ssim(OI,RDIB);
    psnr_FIG(p) = psnr(OI,RDIG); ssim_FIG(p) = ssim(OI,RDIG);psnr_FIM(p) = psnr(OI,RDIM); ssim_FIM(p) = ssim(OI,RDIM);
end
figure(2);subplot(121);plot(K,psnr_FI,LineWidth=1.4); hold on; grid on;title("PSNR vs. K for Weiner filter",FontSize=f);
xlabel("K",FontSize=f);ylabel('PSNR (dB)',FontSize=f-4); plot(K,psnr_FIB,LineWidth=1.4);plot(K,psnr_FIG,LineWidth=1.4);plot(K,psnr_FIM,LineWidth=1.4);
legend("Without prefiltering","With box filtering","With Gauss filtering", "With median filtering",FontSize=f);
subplot(122);plot(K,ssim_FI,LineWidth=1.4); hold on;grid on;title("SSIM vs. K for Weiner filter",FontSize=f);
xlabel("K",FontSize=f);ylabel('SSIM',FontSize=f-4); plot(K,ssim_FIB,LineWidth=1.4);plot(K,ssim_FIG,LineWidth=1.4);plot(K,ssim_FIM,LineWidth=1.4);
legend("Without prefiltering","With box filtering","With Gauss filtering", "With median filtering",FontSize=f);
```

Similar codes were written for blur3 and blur1 images, and included in the submitted project folder.

MATLAB Code : Wiener Filter with prefiltering using best parameters– for blur2.bmp

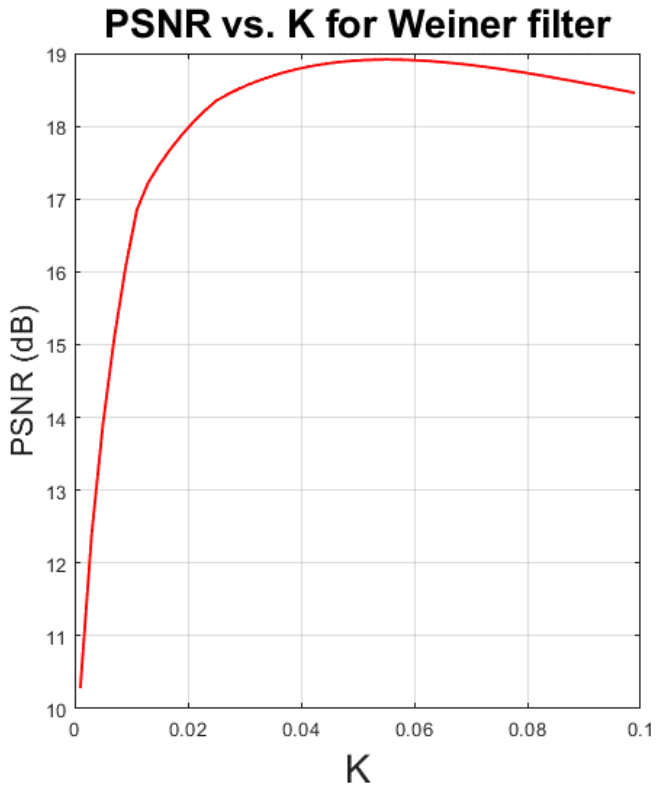
```
clc; clear; close all; f = 16;
OI = im2double(imread('original.bmp'));BI = im2double(imread('blur2.bmp'));
psnr_BI = psnr(OI,BI); ssim_BI = ssim(OI,BI);
figure(1); subplot(221);imshow(OI); title('Original Image',FontSize=f);
subplot(222);imshow(BI);
title(sprintf('Blurred & noisy Image \n (PSNR=%.2fdB, SSIM=%.2f)',psnr_BI,ssim_BI),FontSize=f);
mwidth = 9;
BIM = medfilt2(BI,[mwidth mwidth],'symmetric');
psnr_BIM = psnr(OI,BIM); ssim_BIM = ssim(OI,BIM);
%BI = imboxfilt(BI,5);
subplot(223); imshow(BIM);
title(sprintf('After median filtering (width = %d) \n(PSNR=%.2fdB, SSIM=%.2f)',mwidth,psnr_BIM,ssim_BIM),FontSize=f);
[M, N] = size(BIM);
BIF=fftshift(fft2(BIM));
a = 0.1; b = 0.1; T = 1; WF=ones(M,N);
K = .1; %for blur2
for u=1:M
     for v=1:N
         uu=u-M/2-1;vv=v-N/2-1;
         t = uu*a+vv*b;
         H(u,v) = T*sinc(t)*exp(-1j*pi*t);
         WF(u,v) = (1/H(u,v))*(abs(H(u,v))^2)/(abs(H(u,v))^2+K^2);
     end
end
RDF=BIF.*WF; RDI=abs(ifft2(ifftshift(RDF)));RDI=RDI/max(max(RDI));
psnr_WF=psnr(RDI,OI); ssim_WF=ssim(RDI,OI);
subplot(224); imshow(RDI);
title(sprintf('After Wiener Filtering, K = %.2f \n (PSNR=%.3fdB, SSIM=%.3f)',K, psnr_WF,ssim_WF),FontSize=f);
```

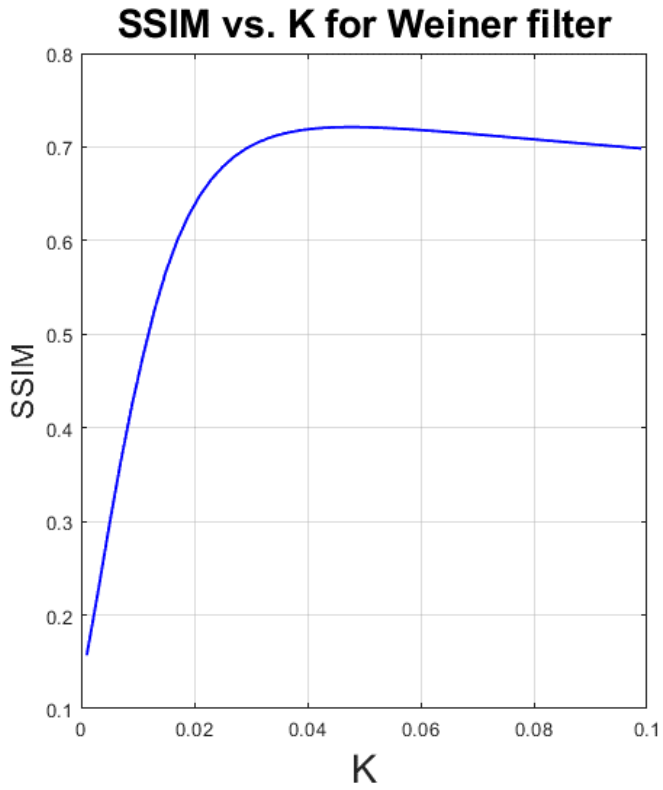
Similar codes were written for blur3 and blur1 images, and included in the submitted project folder.

Quantitative Analysis (Using PSNR and SSIM) for blur1.bmp

$$W(u,v) = \frac{1}{H(u,v)}\left(\frac{|H(u,v)|^2}{|H(u,v)|^2 + K}\right)$$

PSNR and SSIM have been plotted against K, in order to choose the best possible parameters.





No pre-filtering

Maximum PSNR = 18.9156 dB, for K = 0.055
Maximum SSIM = 0.7213 , for K = 0.047

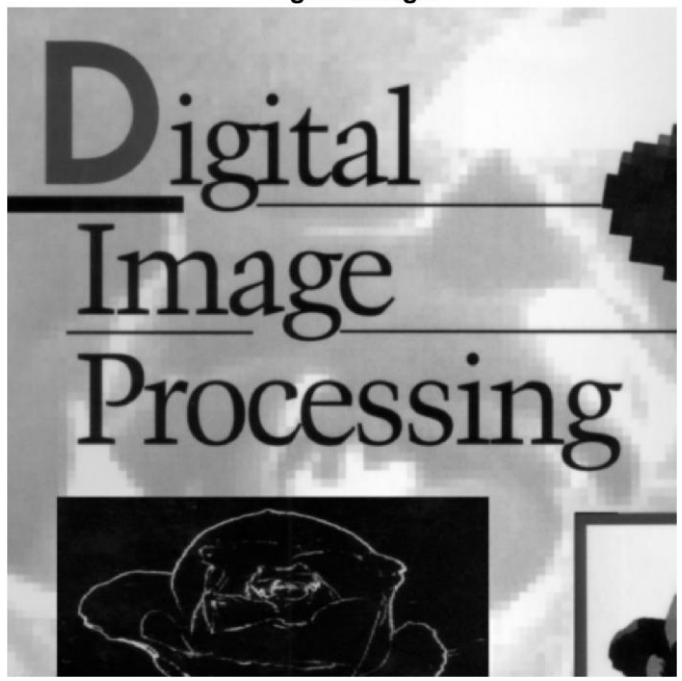
Wiener Filtering Results for blur1.bmp

Discussion

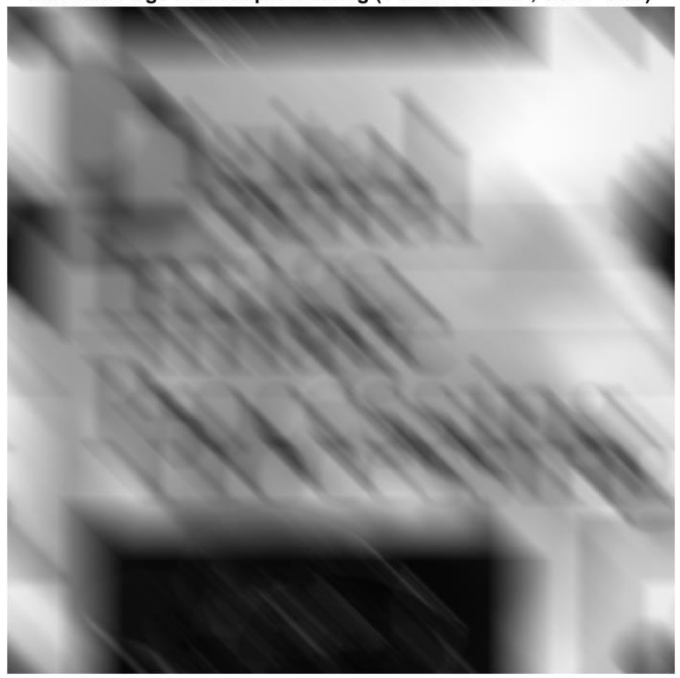
1. The one with the highest SSIM (K=0.047) looks visually better. The image with K=0.055 is more blurred than the one with K = 0.047. So, we will use K=0.055 for further comparison of different types of pre-filtering.

2. In the filtered image, there are some artifacts which look like some light shadows of the text “Digital Image Processing”. This is happening because of the lossy nature of the motion blur degradation.

Original Image



Blurred Image without pre-filtering (PSNR=11.81dB, SSIM=0.54)

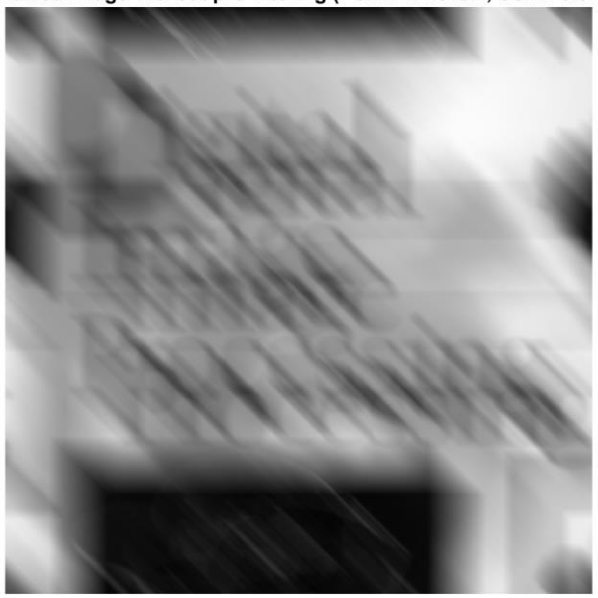


After Wiener Filtering, K = 0.055 (PSNR=18.916dB, SSIM=0.720)

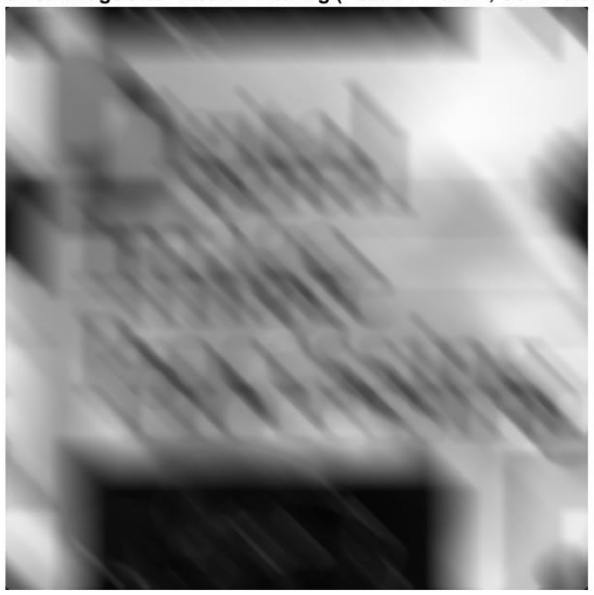
After Wiener Filtering, K = 0.047 (PSNR=18.884dB, SSIM=0.721)

Wiener Filtering Results with and without prefiltering for blur1.bmp

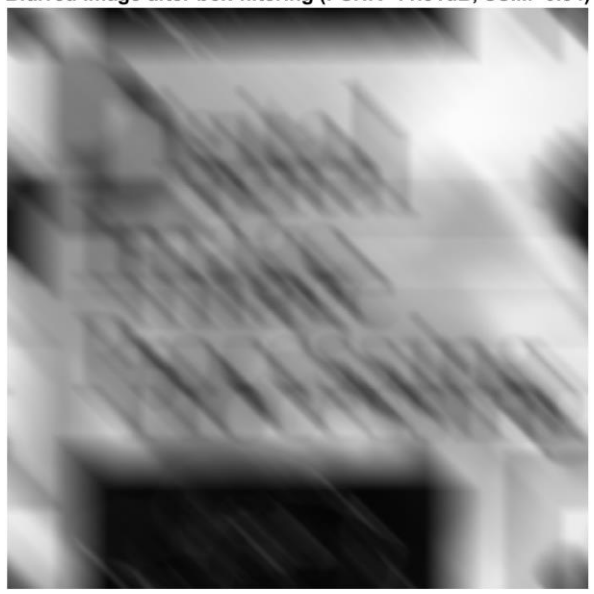
Blurred Image without pre-filtering (PSNR=11.81dB, SSIM=0.54)



Blurred Image after median-filtering (PSNR=11.82dB, SSIM=0.55)

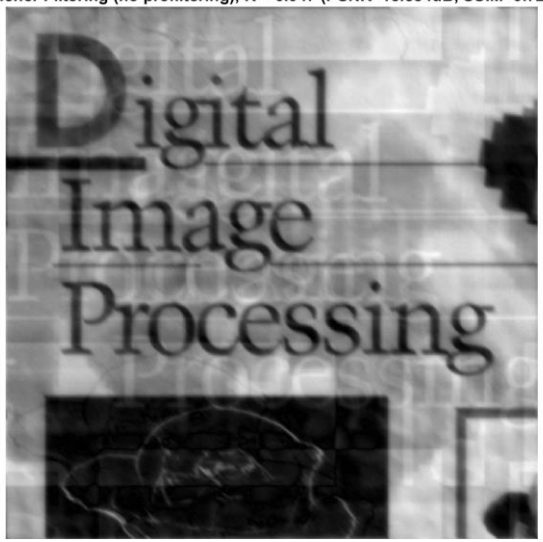


Blurred Image after box-filtering (PSNR=11.81dB, SSIM=0.54)



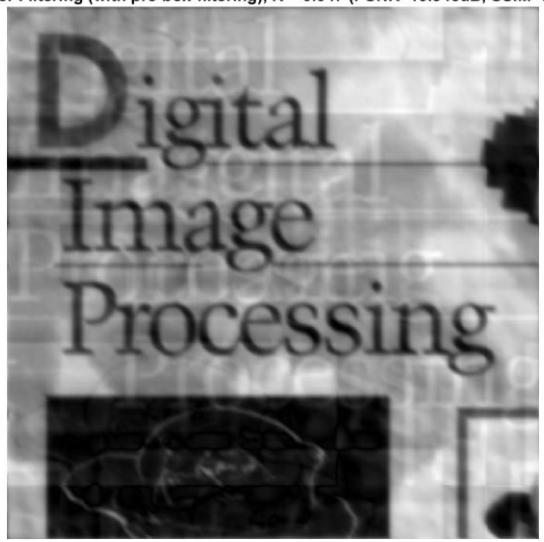
Blurred Image after Gaussian-filtering (PSNR=11.85dB, SSIM=0.56)

Wiener Filtering (no prefiltering), K = 0.047 (PSNR=18.884dB, SSIM=0.721)



Wiener Filtering (with pre-median-filtering), K = 0.047 (PSNR=7.695dB, SSIM=0.507)

Wiener Filtering (with pre-box-filtering), K = 0.047 (PSNR=19.343dB, SSIM=0.731)



Wiener Filtering (with pre-Gauss-filtering), K = 0.047 (PSNR=18.116dB, SSIM=0.678)

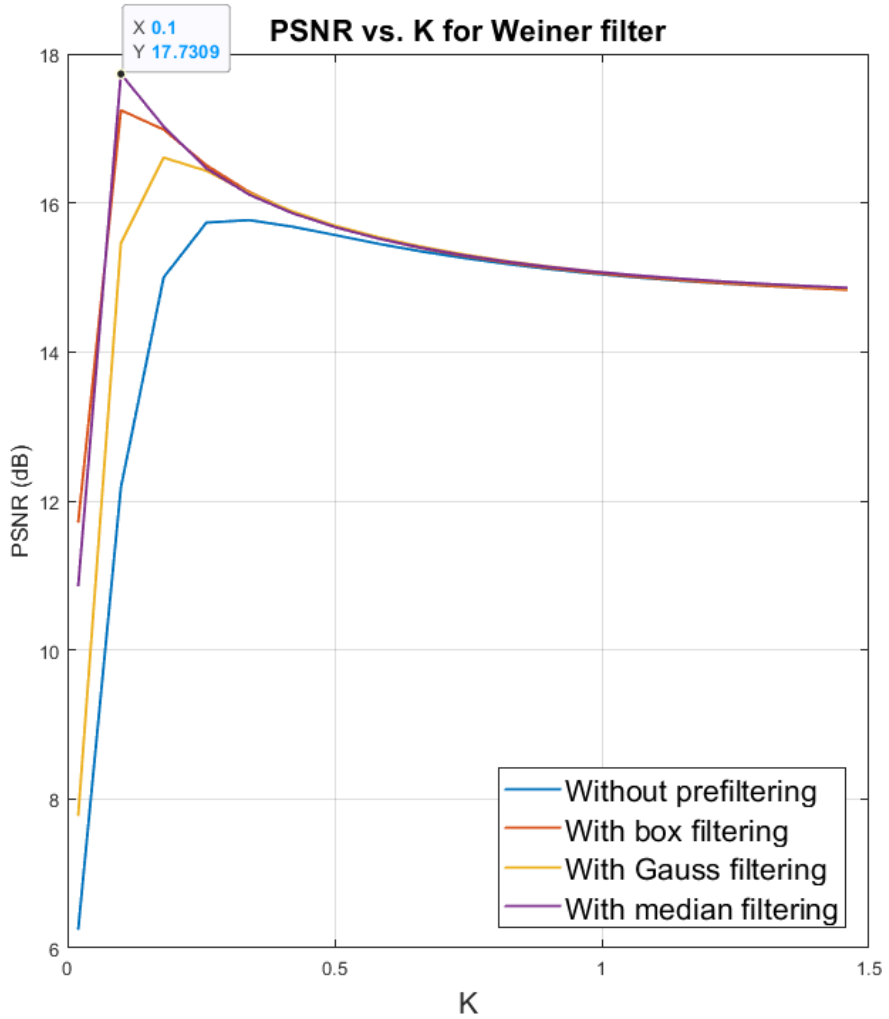
Discussion on Wiener Filtering Results - for blur1.bmp

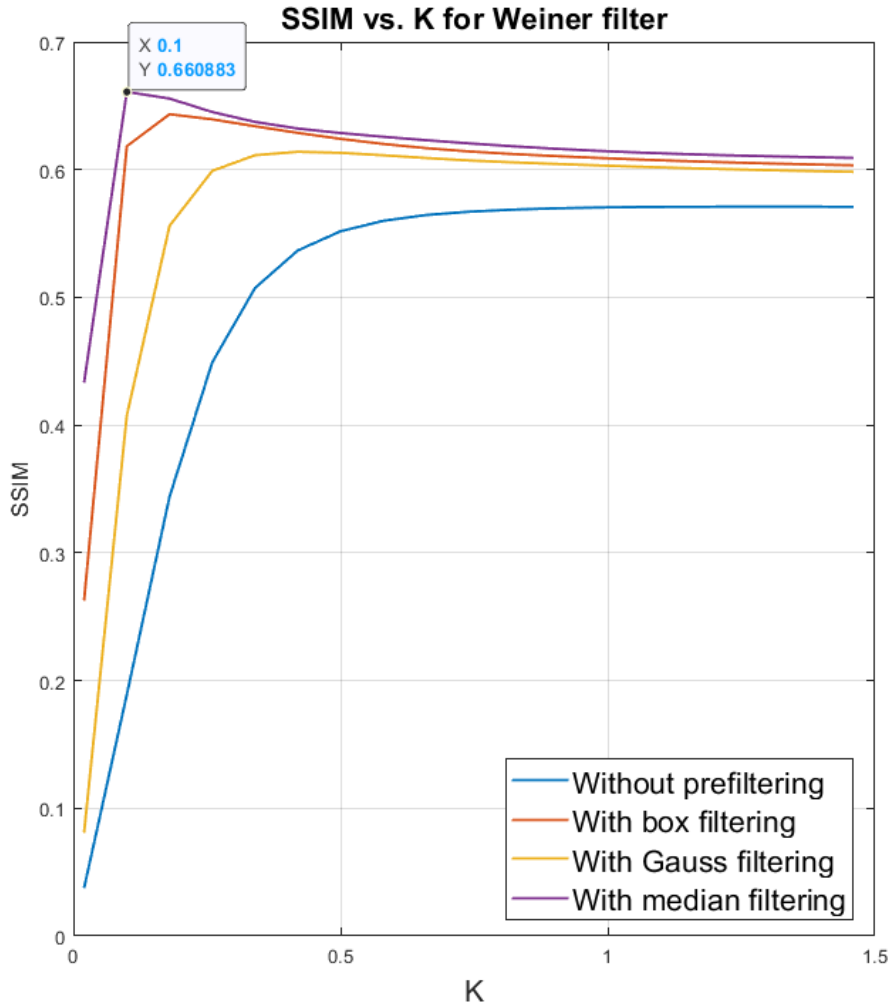
1. Since no noise was added to the original image, prefiltering is not super-necessary. But still it was found that, after wiener filtering, the final image is becoming slightly better with higher PSNR and SSIM, if some box filtering or Gaussian filtering was used. The reason can be the pre-filtering removes some high frequency components and makes the degradation less bad.
2. Median filtering didn't work well because of its non-linear nature, as seen from the wiener filter result after pre median-filtering that caused the image have PSNR = 7.695 and SSIM = 0.507 which is even worse than the initial blurred image. Since blur1.bmp didn't have additional noise, some simple linear smoothing (in our case, arithmetic mean filtering or box filtering) proved to prepare better images that can give better result after Wiener filtering.

Image	PSNR (dB)	SSIM	
Blurred image without pre-filtering	11.81	0.54	
Blurred image with median-filtering	11.82	0.55	
Blurred image with box-filtering	11.81	0.54	
Blurred image with Gaussian-filtering	11.85	0.56	
After Wiener filtering without pre-filtering	18.884	0.721	
After Wiener filtering with pre-median-filtering	7.695	0.507	← Pretty bad
After Wiener filtering with pre-box-filtering	**19.343**	**0.731**	← **Best result**
After Wiener filtering with pre-Gaussian-filtering	18.116	0.678	

Quantitative Analysis (Using PSNR and SSIM) for blur2.bmp

Since, blur2.bmp is corrupted by additional noise, prefiltering was seen to have an important role here.





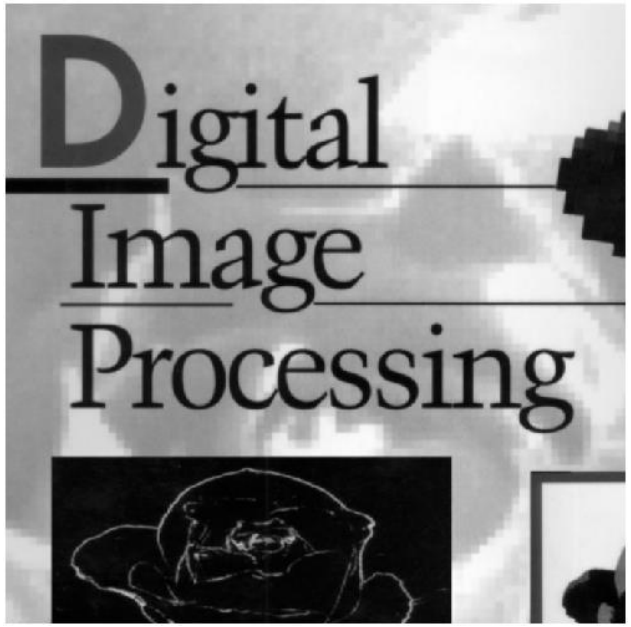
Width of box filter = 7
Width of median filter = 9
SD of Gaussian filter = 0.8

Median filter is giving the highest PSNR and SSIM for K =0.1, hence we will use median filtering as our pre-filtering method.

Prefiltered images with quantitative results

– for blur2.bmp

Original image



Blurred & noisy image
(PSNR=11.527dB, SSIM=0.134)

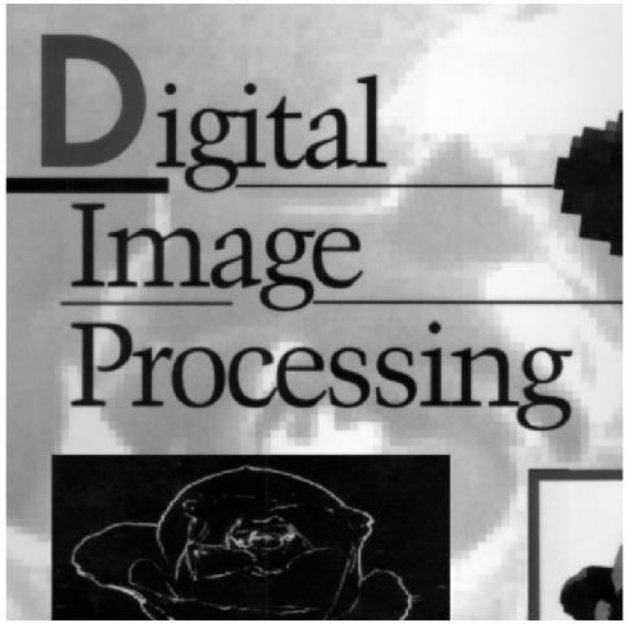
Prefiltered - box filter (width = 7)
(PSNR=11.828dB, SSIM=0.531)

Prefiltered - Gaussian filter (SD = 0.8)
(PSNR=11.796dB, SSIM=0.432)

Prefiltered - median filter (width = 9)
(PSNR=11.824dB, SSIM=0.545)

Wiener Filtering Results
for blur2.bmp

Original Image



Blurred & noisy Image
(PSNR=11.53dB, SSIM=0.13)

After Wiener Filtering, K = 0.10
(PSNR=17.731dB, SSIM=0.661)

After median filtering (width = 9)
(PSNR=11.82dB, SSIM=0.54)

Comparison – without prefiltering vs. with prefiltering – for blur2.bmp

After Wiener Filtering, K = 0.10
(PSNR=12.183dB, SSIM=0.189)

Without prefiltering

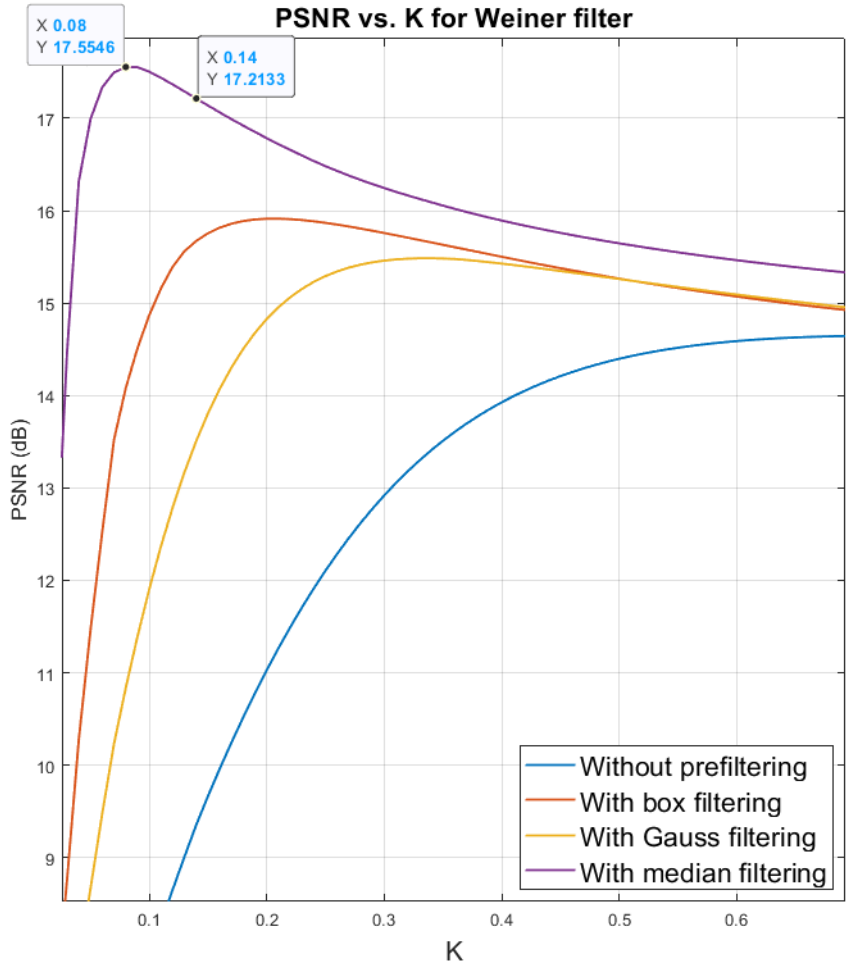
After Wiener Filtering, K = 0.10
(PSNR=17.731dB, SSIM=0.661)

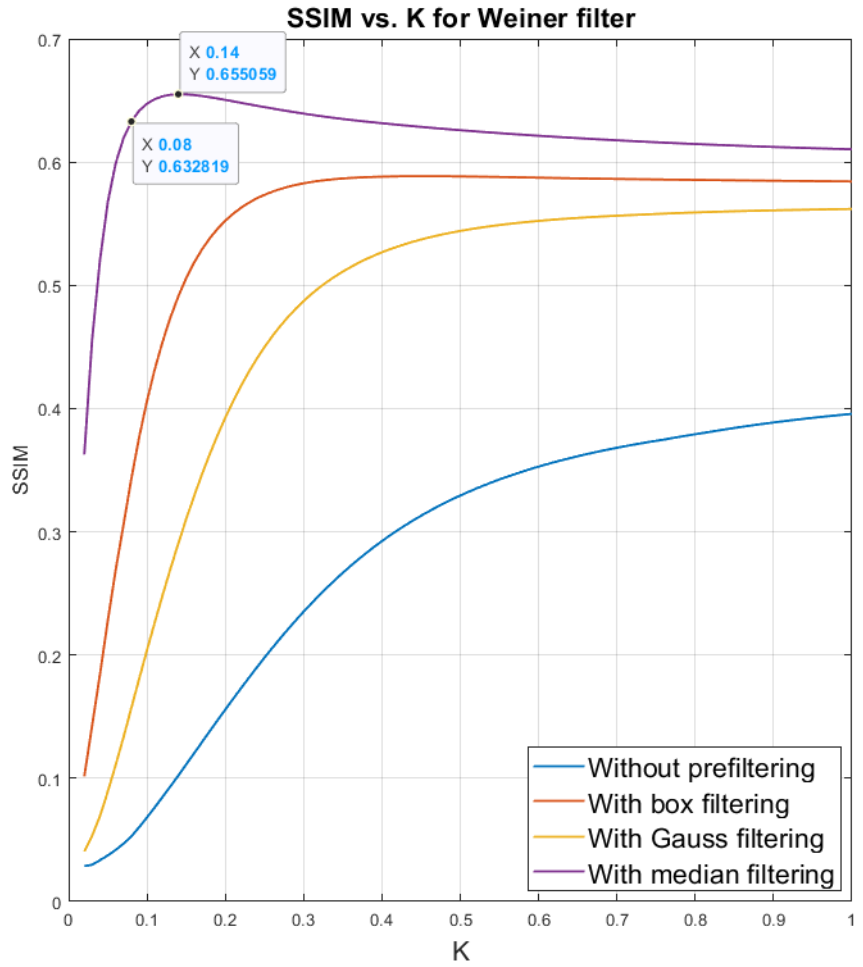
With prefiltering

Prefiltering wins clearly, both quantitively and qualitatively.

Quantitative Analysis (Using PSNR and SSIM) for blur3.bmp

Since, blur3.bmp is corrupted by more noise, prefiltering was seen to have an important role here too.





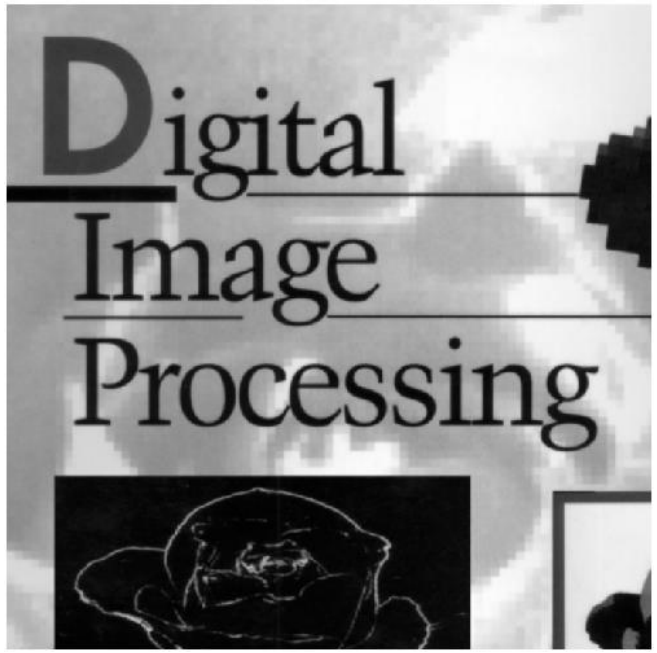
Width of box filter = 7
Width of median filter = 9
SD of Gaussian filter = 1

K=0.14 is giving the highest SSIM. Median filter is performing better than box and gaussian filters, hence we will use median filtering as our pre-filtering method.

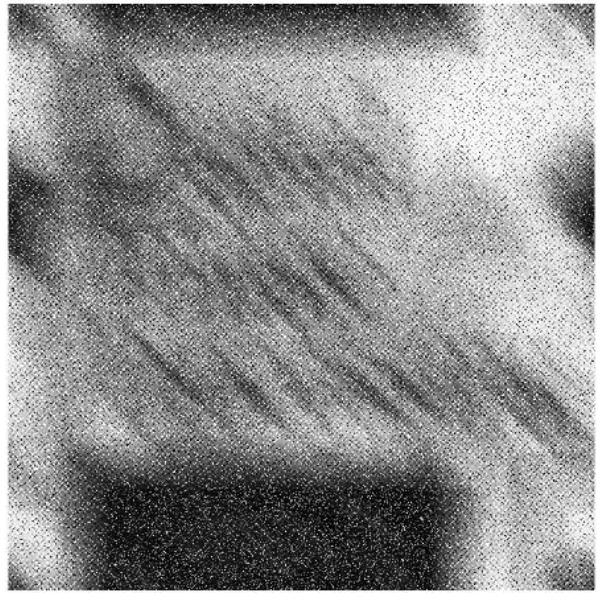
Prefiltered images with quantitative results

– for blur3.bmp

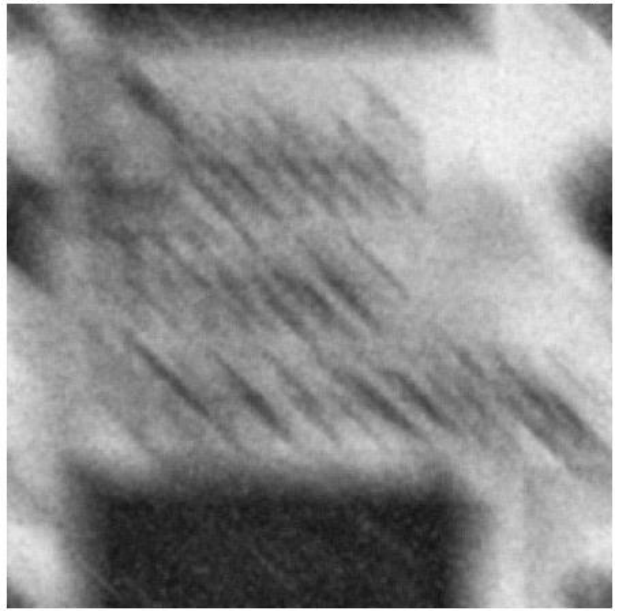
Original image



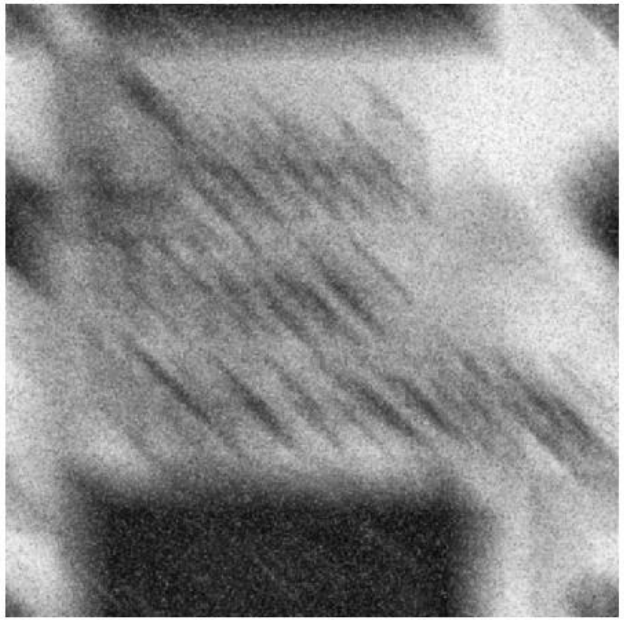
Blurred & noisy image
(PSNR=10.116dB, SSIM=0.026)



Prefiltered - box filter (width = 7)
(PSNR=11.893dB, SSIM=0.462)



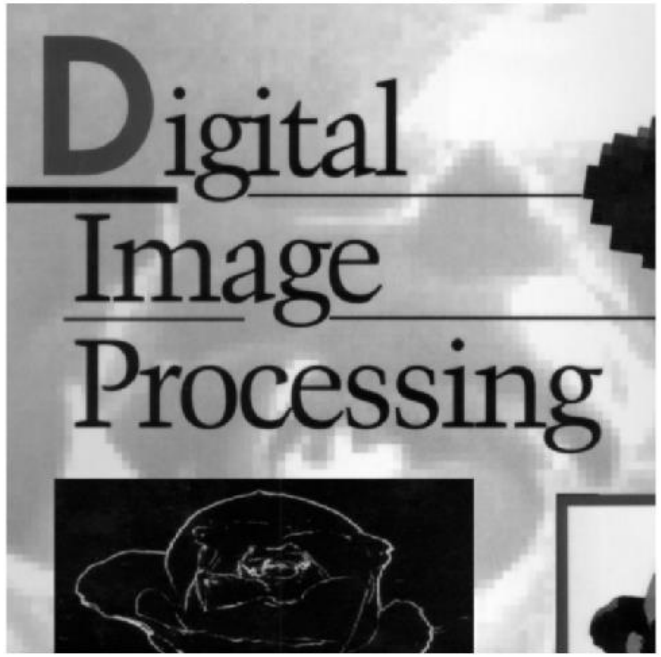
Prefiltered - Gaussian filter (SD = 1.0)
(PSNR=11.793dB, SSIM=0.273)



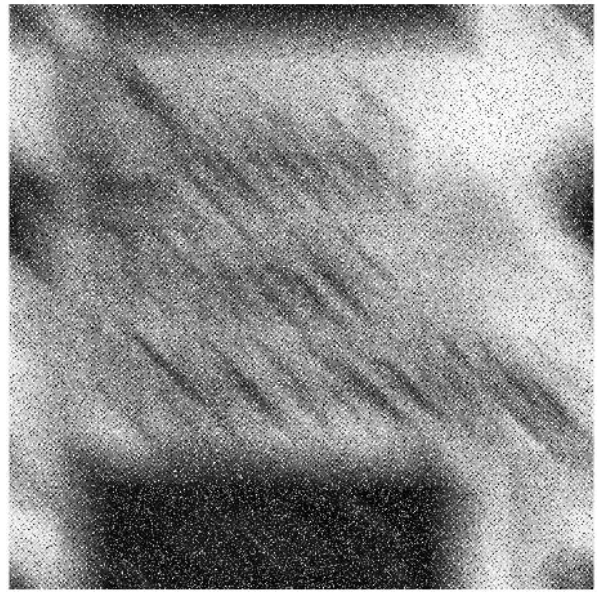
Prefiltered - median filter (width = 9)
(PSNR=11.844dB, SSIM=0.541)

Wiener Filtering Results for blur3.bmp

Original Image



Blurred & noisy Image
(PSNR=10.12dB, SSIM=0.03)



After Wiener Filtering, K = 0.14
(PSNR=17.213dB, SSIM=0.655)

After median filtering (width = 9)
(PSNR=11.84dB, SSIM=0.54)

Comparison – without prefiltering vs. with prefiltering – for blur3.bmp

After Wiener Filtering, K = 0.10
(PSNR=7.928dB, SSIM=0.068)

Without prefiltering

After Wiener Filtering, K = 0.14
(PSNR=17.213dB, SSIM=0.655)

With prefiltering

Prefiltering wins clearly, both quantitively and qualitatively.

Further comparison among the results with all three images

Wiener Filtering (with pre-box-filtering), K = 0.047 (PSNR=19.343dB, SSIM=0.731)

For **blur1.bmp** image
with pre-box-filtering

After Wiener Filtering, K = 0.10 (PSNR=17.731dB, SSIM=0.661)

For **blur2.bmp** image
with pre-median-filtering

After Wiener Filtering, K = 0.14 (PSNR=17.213dB, SSIM=0.655)

For **blur3.bmp** image
with pre-median-filtering

Discussion:

From the above quantitative results and visual perception, we can conclude that added noise level is limiting the restoration performance. Blur1.bmp was restored better because of no added noise. Blur3.bmp had the highest level of noise, hence we could reach lower performance with lower PSNR and SSIM.

Thanks