COL783 Assignment 2 Report

Denoising

Gaussian Noise

I am working on this image:

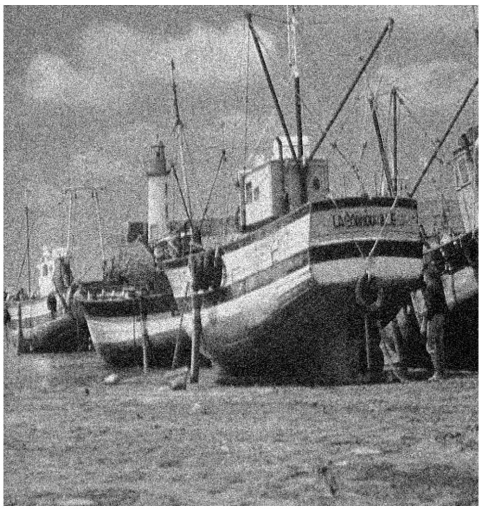
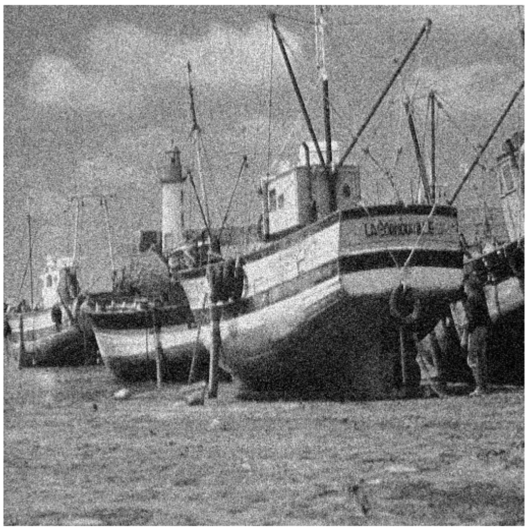


Fig: Original Image Fig: Noisy Image (Gaussian)

Now in this Image I have added a Gaussian noise of 0.01 variance and 0 mean

Median Filtering:



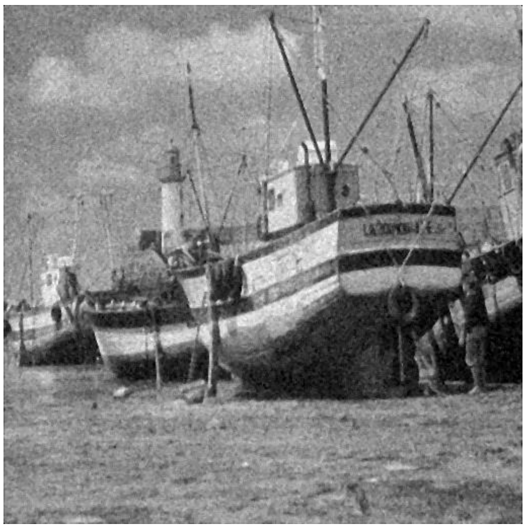


Fig: Noisy Image(Gaussian) PSNR = 20.12 Fig: Median filtered image(3x3) PSNR=25.67

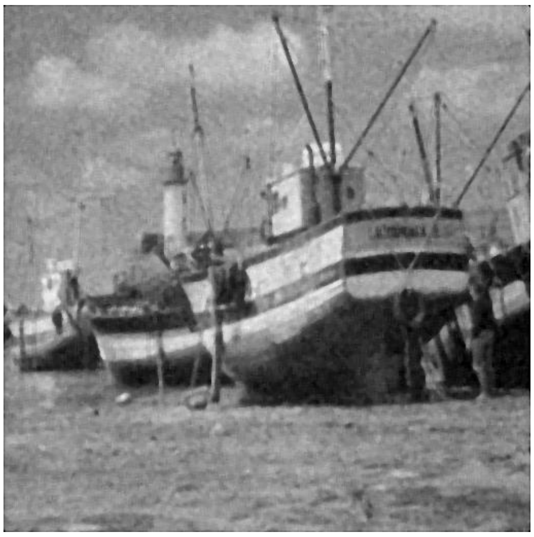


Fig: Median Filter (5x5) PSNR = 25.48

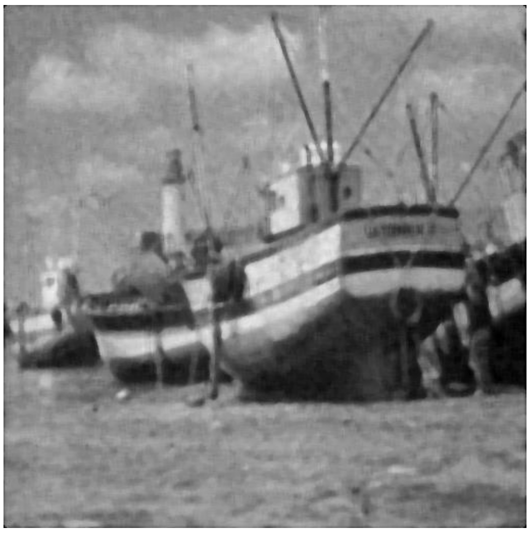


Fig: Median Filter (7x7) PSNR = 24.37

Mean Filtering:

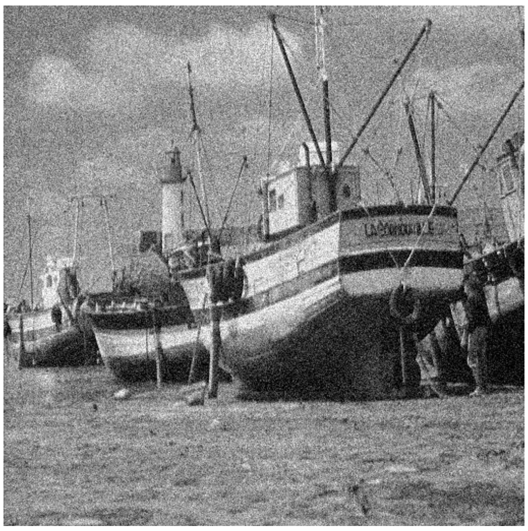


Fig: Noisy Image (Gaussian) PSNR = 20.12

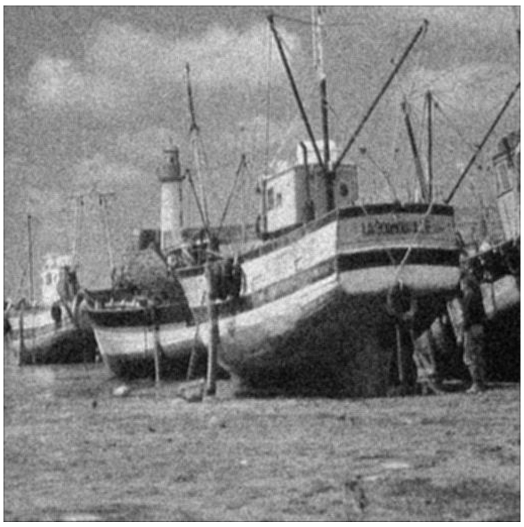


Fig: Mean Filter (3x3) PSNR = 26.2

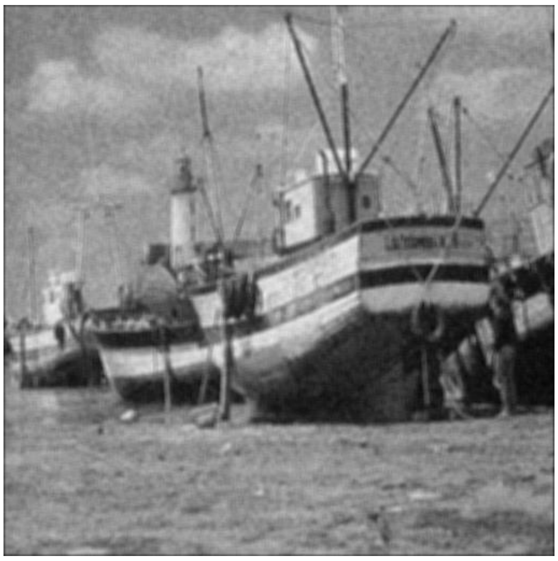


Fig: Mean Filter (5x5) PSNR = 25.01

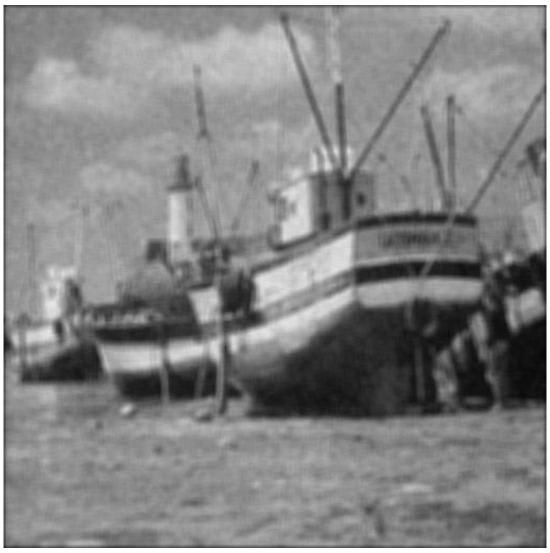


Fig: Mean Filter (7x7) PSNR = 23.6

Salt and Pepper Noise

Fig: Original Image

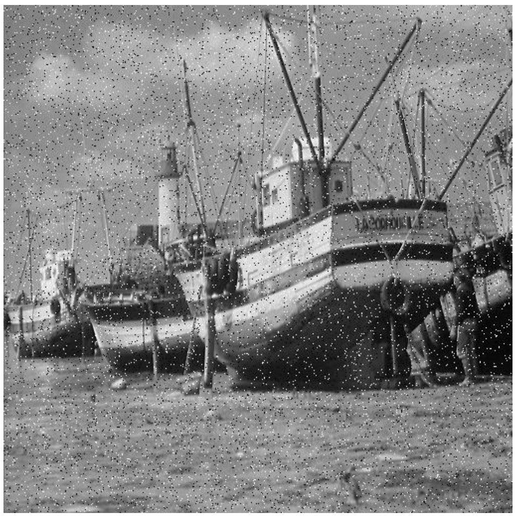


Fig: Noisy Image (Salt and Pepper)

Now in this Image I have added a Salt and Pepper noise of 0.05 noise distribution

Median Filtering

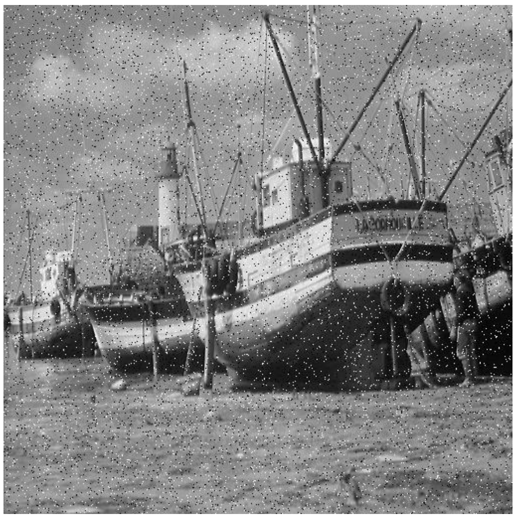


Fig: Noisy Image (Salt and Pepper) PSNR = 18.45

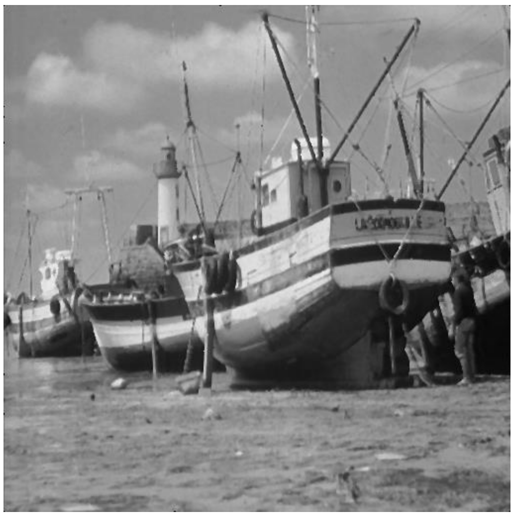


Fig: Median Filter (3x3) PSNR = 30.31

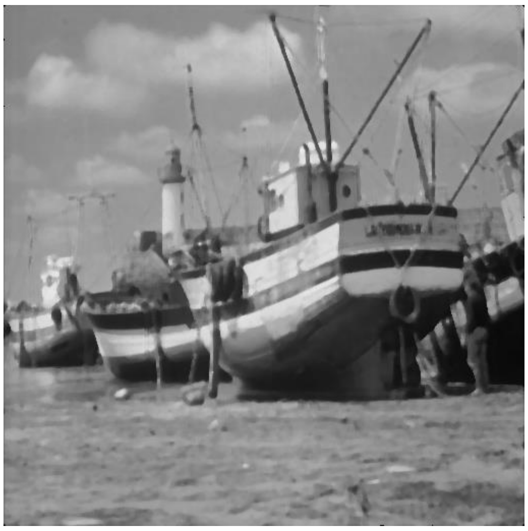


Fig: Median Filter (5x5) PSNR = 27.05

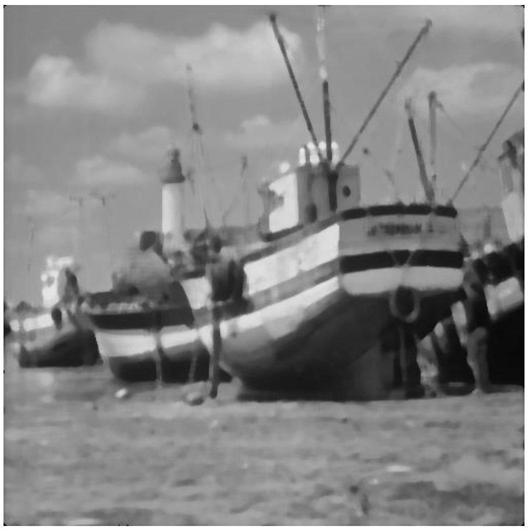


Fig: Median Filter (7x7) PSNR = 25.10

Mean Filtering

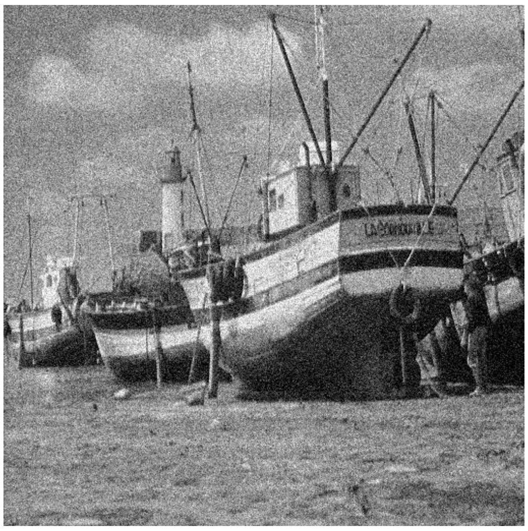


Fig: Noisy Image (Salt and Pepper) PSNR = 18.45

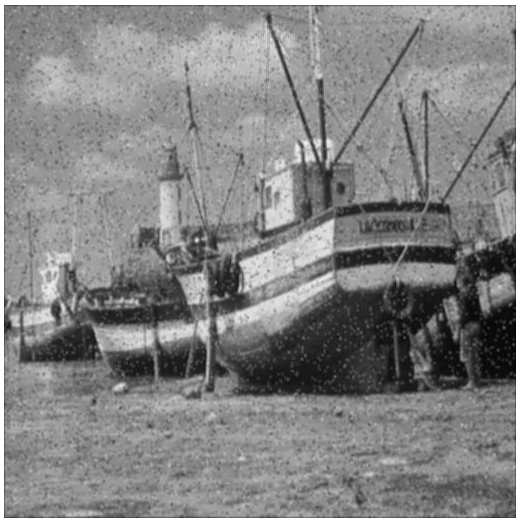


Fig: Mean Filter (3x3) PSNR = 25.18

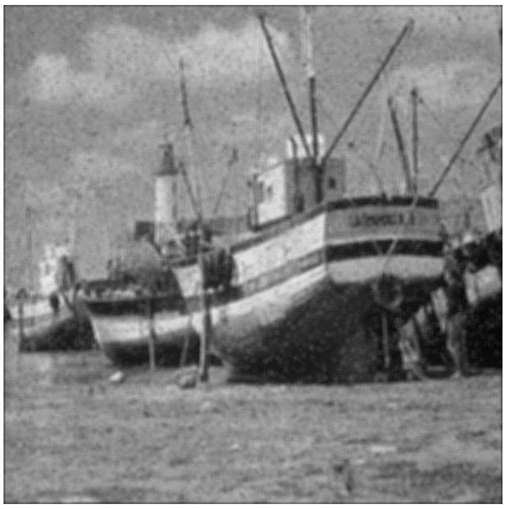


Fig: Mean Filter (5x5) PSNR = 24.54

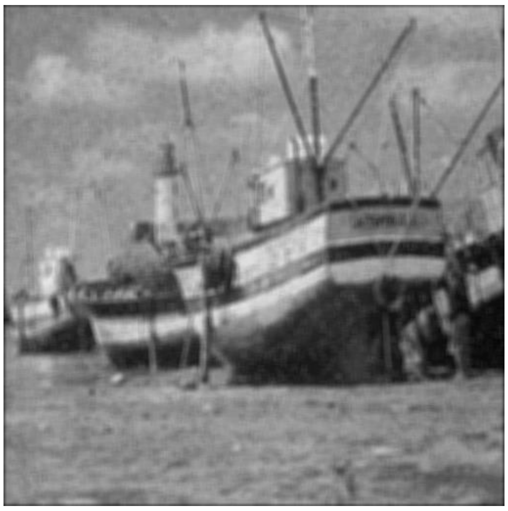


Fig: Mean Filter (7x7) PSNR = 23.37

CODE (in MATLAB) FOR MEAN and MEDIAN FILTERS:

```
img = imread('boat.png');
sigma=0.01;
noise=uint8(floor(randn(512,512)*sigma));
im = img+noise;
peaksnr = psnr(im,img)
img_med = medfilt2(im,[7,7]);
peaksnrmed = psnr(img_med,img)
imwrite(img_med,'med_img_7.png')
filter = ones(7,7)/49;
img_mean = uint8(conv2(im, filter, 'same'));
peaksnrmed = psnr(img_mean,img)
imwrite(img_mean,'mean_img_7.png')
```

Conclusion: We notice that for gaussian with zero mean noise the averaging filter performs better than the median filter as expected and as we increase filter size the PSNR decreases. The median filter works much better for salt and pepper noise as median filter more or less gets rid of the salt and the pepper noises but averaging filter averages them and as a result it contributes to less PSNR. Overall we see a change in sharp features. Averaging filter blurs the image and as a result we lose sharp edges.

NON LOCAL MEANS

Here I try to apply edge preserving filter named NON LOCAL MEANS

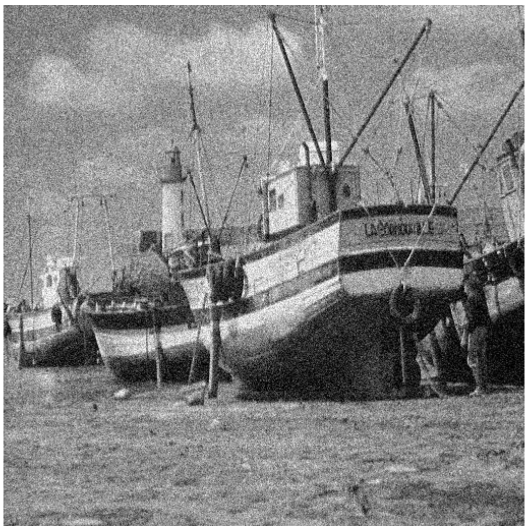


Fig: Noisy Image(Gaussian) PSNR = 20.12

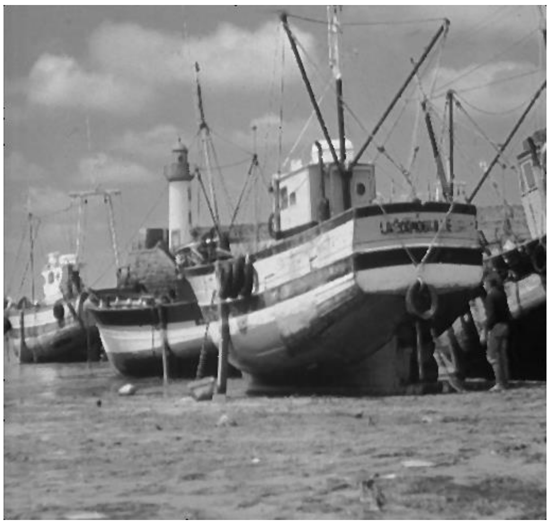


Fig: Filtered Image(NLM) PSNR = 31.23

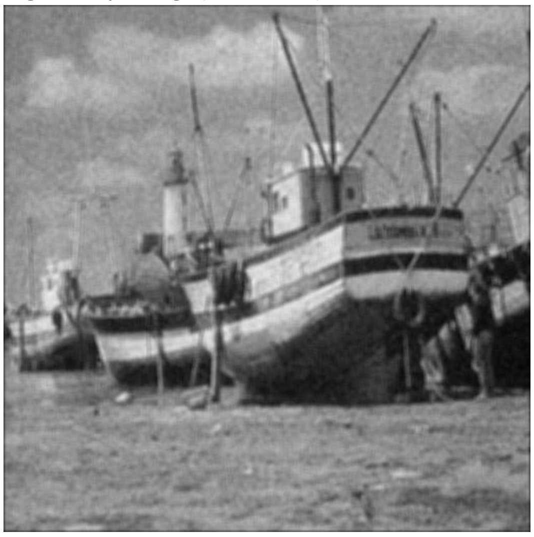


Fig: Filtered Image(Mean filter)

CODE (in MATLAB) FOR NON LOCAL MEANS

```
function [output]=NLmeans(img1,t,f,h)

[m n]=size(img1);


 % Memory for the output
 Output=zeros(m,n);
 % Replicate the boundaries of the input image
 img2 = padarray(img1,[f f],'symmetric');

 % Used kernel
 kernel = make_kernel(f);
 kernel = kernel / sum(sum(kernel));

 h=h*h;

 for i=1:m
 for j=1:n
         i
         i1 = i+ f;
         j1 = j+ f;

         W1= img2(i1-f:i1+f , j1-f:j1+f);

         wmax=0;
         average=0;
         sweight=0;

         rmin = max(i1-t,f+1);
         rmax = min(i1+t,m+f);
         smin = max(j1-t,f+1);
         smax = min(j1+t,n+f);

         for r=rmin:1:rmax
         for s=smin:1:smax

                if(r==i1 && s==j1) continue; end;

                W2= img2(r-f:r+f , s-f:s+f);

                d = sum(sum(kernel.*double(W1-W2).*double(W1-
W2)));

                w=exp(-d/h);
                if w>wmax
                    wmax=w;
                end

                sweight = sweight + w;
                average = average + w*img2(r,s);
            end
            end

            average = average + wmax*img2(i1,j1);
            sweight = sweight + wmax;

            if sweight > 0
                output(i,j) = average / sweight;
            else
                output(i,j) = img(i,j);
            end
 end
 end
end

function [kernel] = make_kernel(f)

kernel=zeros(2*f+1,2*f+1);
for d=1:f
  value= 1 / (2*d+1)^2 ;
  for i=-d:d
  for j=-d:d
    kernel(f+1-i,f+1-j)= kernel(f+1-i,f+1-j) + value ;
  end
  end
end
kernel = kernel ./ f;
 end
```

Conclusion : We can easily that NON LOCAL MEANS produces a result better than any of the averaging filter. It preserves the sharp edges as expected because it weights the pixels non locally thus reducing blurring.

DEBLURRING

I have used 21x21 disk shaped point spread function to blur the image

Fig: Original Image

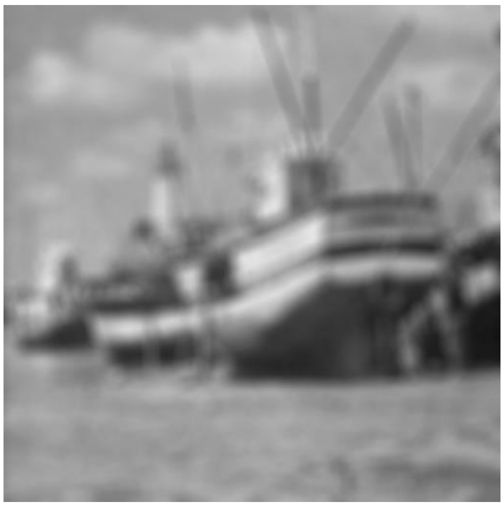


Fig: Blurred Img after adding noise

Weiner Filter

Fig: Blurred Noisy Image

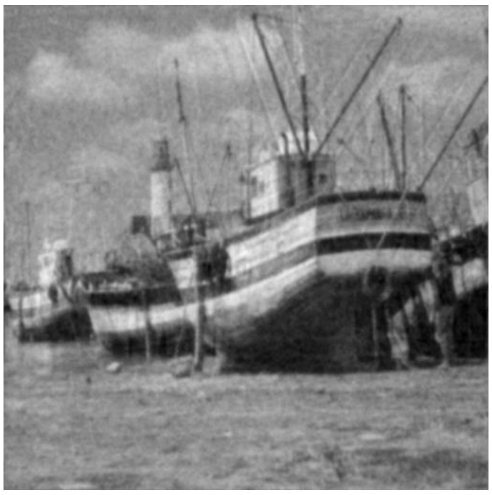


Fig: Deblurred using Weiner Filter

Original Image Spectrum

CODE FOR WEINER FILTER (MATLAB)

```
h = fspecial('disk',10);
blurred = imfilter(im,h,'conv','circular');
h = padarray(h,[491 491],0,'post');
H = fft(h);
h_norm = abs(H).^2;
F = norm(fft(img),'fro')^2;             % using original spectrum
G = fft(blurred)
F_hat = (conj(H).*G)./((h_norm) + (0.0001*512^4)/F);
f_hat = ifft(F_hat);
f_hat = abs(f_hat);
f_hat = uint8((255/max(max(f_hat)))*f_hat);
```

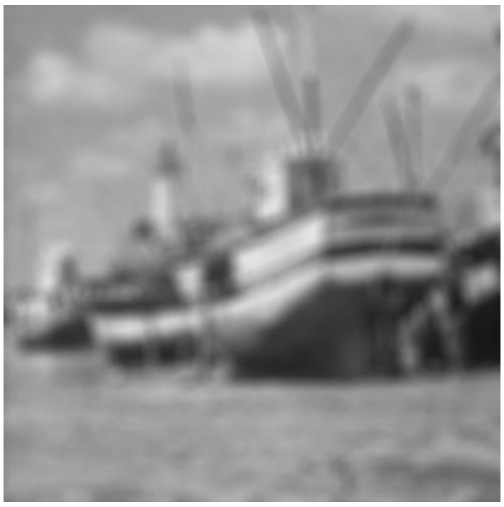


Fig: Blurred Noisy Image

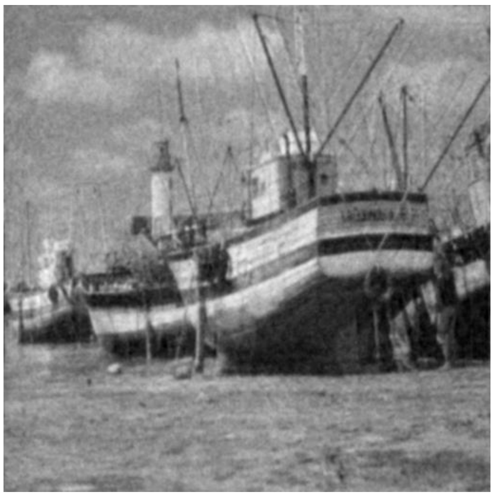


Fig: Deblurred using constant spectrum

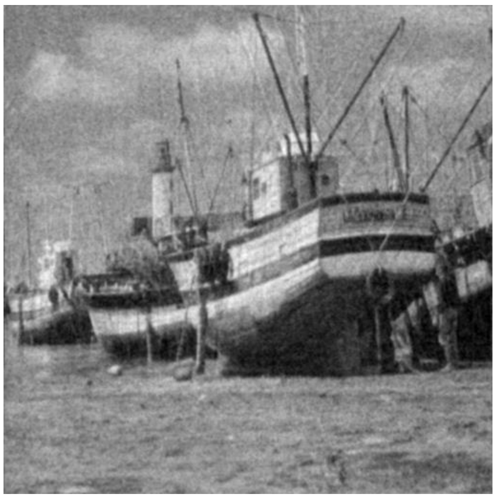


Fig: Deblurred Image

Third type of spectrum for alpha = 0.95

Here spectrum of noise was calculated using the parseval's theorem that spectrum in spatial domain is proportional to spectrum in frequency domain. In spatial we can create an estimator for variance as (1/MN)* $\sum|\eta(x,y)|^2$ and therefore spectrum in spatial domain can be written as MN*variance where (M,N) is the size of the image. And then the spectrum in Fourier domain is MN*(MN*variance).

Using this we note that there is considerable deblurring from the weiner filter. I get the best result when I use the ground truth spectrum with a PSNR of 25.14, then for the third type of spectrum with a PSNR of 24.06 and worst for the constant spectrum with a PSNR of 23.34. For the third type the optimal alpha for this image was found at 0.95.

Real World Image Deblurring

I have been working on this Image:

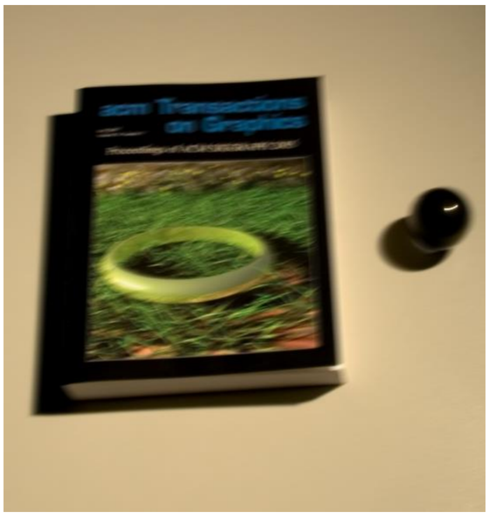


Fig: Blurred real world image

Now to estimate the noise I took a small 30 x 30 patch from the rightmost part of the image which has roughly constant intensity and visualized the histogram. The histogram was something like this :

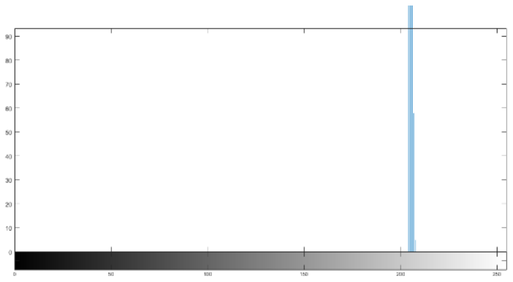


Fig: Histogram of the patch

Fig: The 30x30 patch

Looking at the histogram of this patch we can conclude that we can estimate the noise to be zero as the histogram has almost constant intensity.

Now we need to estimate the point spread function. As sir discussed in class that all the images

I checked all the values for radius of point spread function between 2 and 20 and I got the best result for 5.

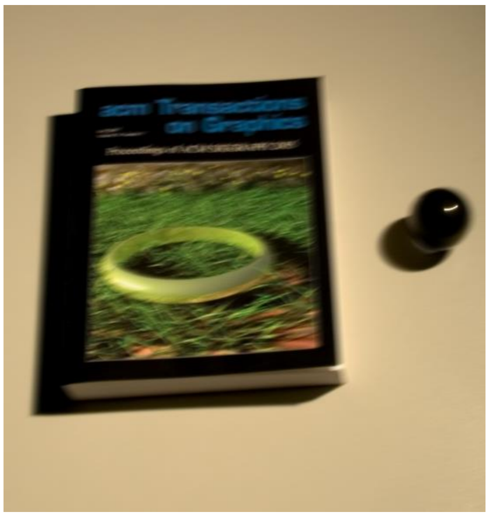


Fig: Blurred Image

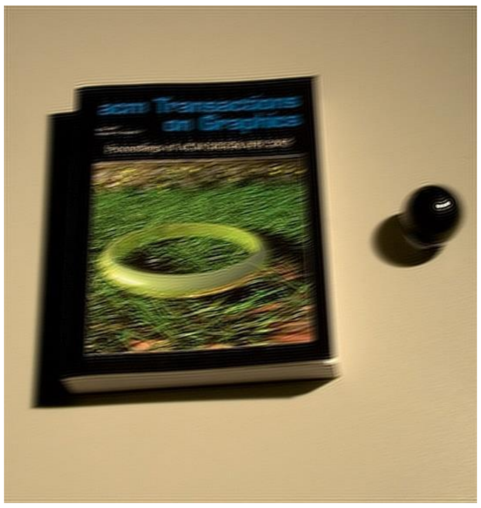


Fig: Deblurred Image

I experimented quite a while but could not get better results and as there are minors ahead I could not experiment more. But clearly there is a slight enhancement but there is ringing effect as the estimate for the point spread function may not be correct. I used the same weiner filter above but with slightly more time I could have implemented other deblurring algorithms.

I have also worked on this Image:

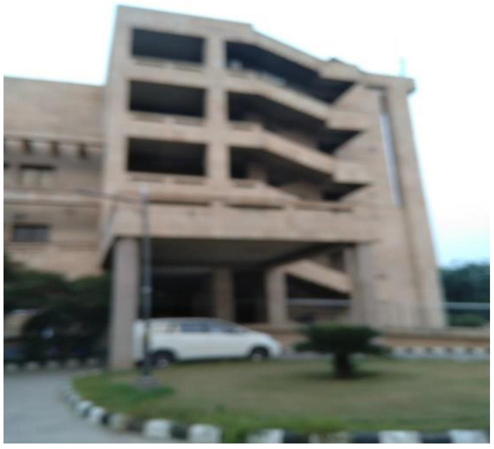
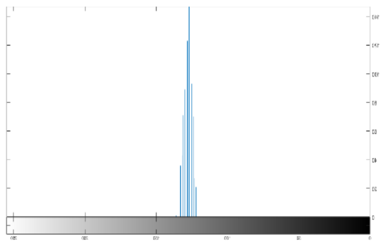
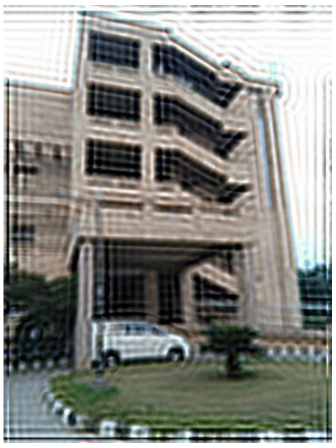


Fig: Real World Blurred Image

I took out the following 30x30 seemingly constant intensity patch:



This Noise can be estimated to be Gaussian with a stddev of around 0.01. The same weiner filter is used deblur this image using a disk shaped point spread function of radius 10. I got the following result.



I am able to deblur the image somewhat but there is excessive ringing in this restored image.

Fig: Deblurred Image