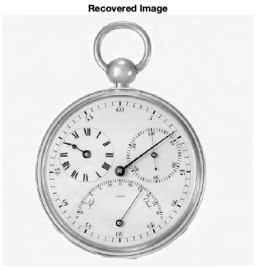
1. **Create degraded image**

a)
```
I = imread("~/Documents/HW04_input.png");
h = fspecial ('average', 5) ;
D = im2double(rgb2gray(I));
% Degrade Image
c = imfilter(D,h,'conv','circular');
% Recover the image
hf = fft2(h,size(c,1),size(c,2));
Fa = real(ifft2(fft2(c)./hf));
```

Original Image

Degraded Image

Here the original image is convoluted using an average filter i.e. a low pass filter which means it allows only low frequency to pass and blocks high frequency values. In this case frequency means the change in adjacent pixel values. So, around the edges the pixel frequency is very high and it is reduced which results in a blurred image. For convolution **'circular'** padding is used because the kernel falls outside for boundary pixels and they are assumed as Zero.



It is possible to recover the original image because the degraded image doesn't have any noise. So, when we divide the DFT of the blurred image with the DFT of the filter the pixels where the pixel frequency is reduced are amplified and the Image is restored. If an image has noise then the noise is also amplified and the image is further degraded.

b)

```
I = imread("~/Documents/HW04_input.png");
h = fspecial ('average', 5) ;
D = im2double(rgb2gray(I));
c = imfilter(D,h,'conv', 'circular');
y = sum(abs(c(:)).^2)/numel(c);
SNR = 30;
sigma = y / (10^(SNR/10));
N = imnoise(c, 'gaussian', 0, sigma);
psnr_N_D = psnr(N, D);
psnr_N_c = psnr(N, c);
```

PSNR Noisy(N) to Original (D) = 22.5097
PSNR Noisy(N) to Blurred (c) = 31.3151

PSNR = 10log(R2 / MSE). PSNR is inversely proportional to mean square error. If MSE value is less, better image and more PSNR value.

psnr(N,D) calculates peak signal to noise ratio(psnr) of image N with respect to image D. Higher the value the better the image quality. It is defined by what extinct image N is closer to image D.

Gaussian white noise is the only difference between N and c. So, PSNR between N and c will be second highest. It is much closer to SNR of the noise added.

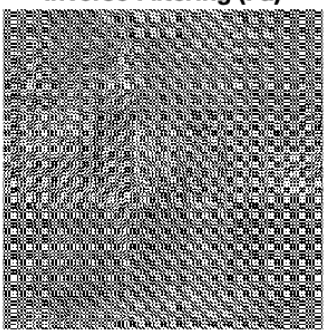
2. Inverse filter on blurred + Noisy Image.

```
I = imread("~/Documents/HW04_input.png");
h = fspecial ('average', 5) ;
D = im2double(rgb2gray(I));
c = imfilter(D,h,'conv', 'circular');
y = sum(abs(c(:)).^2)/numel(c);
SNR = 30;
sigma = y / (10^(SNR/10));
N = imnoise(c, 'gaussian', 0, sigma);
hf = fft2(h,size(N,1),size(N,2));
F = real(ifft2(fft2(N)./hf));
F1 = real(ifft2((abs(hf) > 0.001).*fft2(N)./hf));
```

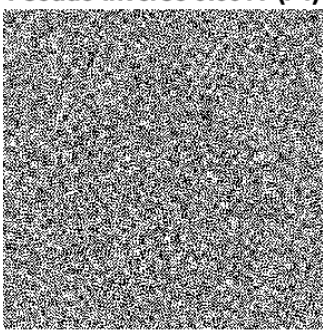
Original Image (D)

Blurred + Noise (c)

Inverse Filtering (Fa)



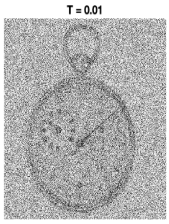
Pseudo Inverse 0.001T (F1)



Original Image is completely lost when we directly apply inverse filtering on Blurred + Noise image because when noise is divided by very small numbers leads to amplification of noise. So, to restore the original image we need to eliminate smaller noise values in the DFT of the filter. It is observed in the last picture that the noise is less amplified when we set the threshold as 0.01. Keep increasing the threshold value until PSNR value saturates.

```
F2 = real(ifft2((abs(hf) > 0.01).*fft2(N)./hf));
F3 = real(ifft2((abs(hf) > 0.1).*fft2(N)./hf));
F4 = real(ifft2((abs(hf) > 0.2).*fft2(N)./hf));
```

```
F5 = real(ifft2((abs(hf) > 0.25).*fft2(N)./hf));
F6 = real(ifft2((abs(hf) > 0.3).*fft2(N)./hf));
F7 = real(ifft2((abs(hf) > 0.35).*fft2(N)./hf));
```



The Quality of the image increases till threshold value reaches **0.25** then decreases again.

3. Apply the CLS filter in the DFT domain to restore the original image.

```
I = imread("~/Documents/HW04_input.png");
h = fspecial ('average', 5);
D = im2double(rgb2gray(I));
c = imfilter(D,h,'conv');
y = sum(abs(c(:)).^2)/numel(c);
SNR = 30;
sigma = y / (10^(SNR/10));
N = imnoise(c, 'gaussian', 0, sigma);
H = fft2(h,size(N,1),size(N,2));
Hc = conj(H);
Num = Hc.*fft2(N);

la = fspecial('laplacian');
lb = fspecial('laplacian', 0.5);
lc = fspecial('laplacian', 0.7);
```

```
La = fft2(la,size(N,1),size(N,2));
Lb = fft2(lb,size(N,1),size(N,2));
Lc = fft2(lc,size(N,1),size(N,2));
H2 = abs(H).^2;
La2 = abs(La).^2;
Lb2 = abs(Lb).^2;
Lc2 = abs(Lc).^2;
alpha1 = 0.01;
alpha2 = 0.05;
alpha3 = 0.07;
Denom1 = H2 + alpha1*La2;
Denom2 = H2 + alpha1*Lb2;
Denom3 = H2 + alpha1*Lc2;
Denom4 = H2 + alpha2*La2;
Denom5 = H2 + alpha2*Lb2;
Denom6 = H2 + alpha2*Lc2;
Denom7 = H2 + alpha3*La2;
Denom8 = H2 + alpha3*Lb2;
Denom9 = H2 + alpha3*Lc2;

S1 = real(ifftn(Num./Denom1));
S2 = real(ifftn(Num./Denom2));
S3 = real(ifftn(Num./Denom3));
S4 = real(ifftn(Num./Denom4));
S5 = real(ifftn(Num./Denom5));
S6 = real(ifftn(Num./Denom6));
S7 = real(ifftn(Num./Denom7));
S8 = real(ifftn(Num./Denom8));
S9 = real(ifftn(Num./Denom9));
```

a = 0.01 lap a = 0.2

a = 0.01 lap a = 0.5

a = 0.01 lap a = 0.7

a = 0.05 lap a = 0.2

a = 0.05 lap a = 0.5

a = 0.05 lap a = 0.7

a = 0.07 lap a = 0.2

a = 0.07 lap a = 0.5

a = 0.07 lap a = 0.7

Image Borders handling?
A Laplacian filter is also an edge detector, as it computes the second derivative measuring the rate at which the first derivative changes. That is why with the increase in alpha value of laplacian filter the image sharpness increases.

It is observed that alpha acts as the base line for minimum value in the DFT of Laplacian. If alpha is less noise is divided by small values and it amplifies. So, with increase in alpha value noise reduces and if we keep on increasing the image blurs again.