2019 MMILAB.DIP Seminar Week2(1/6~1/15)

20151415 김성수

Contents

- Spatial domain filtering
- Frequency domain filtering
- Restoration

Spatial Domain Filtering

1. Convolution
2. Low pass filtering (Average filter, Gaussian LPF)
3. Laplacian filter
4. Unsharp masking(high boost filtering
5. Sobel operator

Convolution

* Convolution

$$f * h = \sum_k \sum_l f(k,l)\, h(m-k, n-l) \quad (f = \text{image}, h = \text{filter})$$

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

→ Y-flip →

h_7	h_8	h_9
h_4	h_5	h_6
h_1	h_2	h_3

→ X-flip →

h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

$\otimes$ →

$$f*h = f_1h_9 + f_2h_8 + f_3h_7 + f_4h_6 + f_5h_5 + f_6h_4 + f_7h_3 + f_8h_2 + f_9h_1$$

Filter response

Image

I(x-1,y-1)	I(x,y-1)	I(x+1,y-1)
I(x-1,y)	I(x,y)	I(x+1,y)
I(x-1,y+1)	I(x,y+1)	I(x+1,y+1)

Filter

0	-1	0
-1	5	-1
0	-1	0

Response : $J(x,y) = 5*I(x,y) - I(x+1,y) - I(x-1,y) - I(x,y+1) - I(x,y-1)$

Image Zero padding

Image

0	0	0
0	I(x,y)	I(x+1,y)
0	I(x,y+1)	I(x+1,y+1)

Filter

0	-1	0
-1	5	-1
0	-1	0

Edge of image : ignore or hallucinate 0

Correlation

* Correlation

$$f * h = \sum_k \sum_l f(k,l)\, h(m+k, n+l) \quad (f = \text{image},\ h = \text{filter})$$

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$\otimes$ f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

$f * h$

$= f_1 h_1 + f_2 h_2 + f_3 h_3$
$+ f_4 h_4 + f_5 h_5 + f_6 h_6$
$+ f_7 h_7 + f_8 h_8 + f_9 h_9$

If filter is symmetry ($h_1 = h_9, h_2 = h_8, h_3 = h_7, h_4 = h_6$)

$\longrightarrow$ Convolution $f * h$ = correlation $f * h$

Low pass filter

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Smoothing filter(a.k.a LPF)

Idea : Replace each pixel by a average of its neighbors

Pro : Removes/ Reduces noise

Con : Blurs the image, removes detail

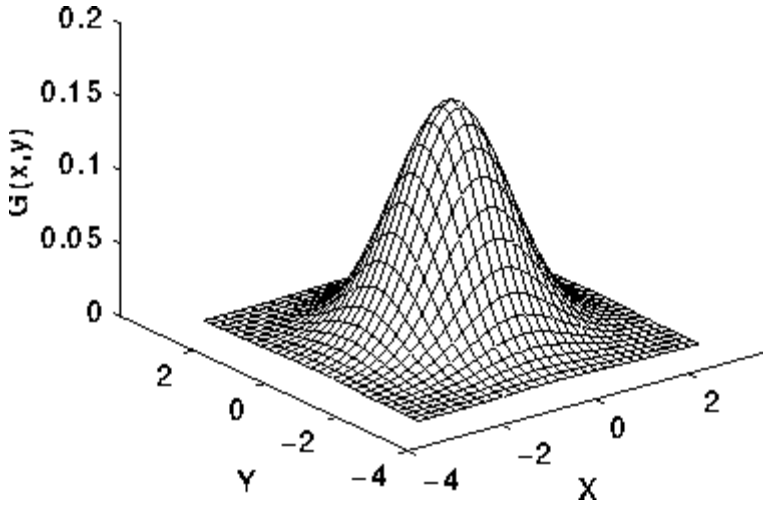
Application : Thresholding after Low pass filtering -> Removes small details while maintaining big detail

Low pass filter

Keep middle pixel weight

$\frac{1}{16}$	$\frac{2}{16}$	$\frac{1}{16}$
$\frac{2}{16}$	$\frac{4}{16}$	$\frac{2}{16}$
$\frac{1}{16}$	$\frac{2}{16}$	$\frac{1}{16}$

Gaussian smoothing filter

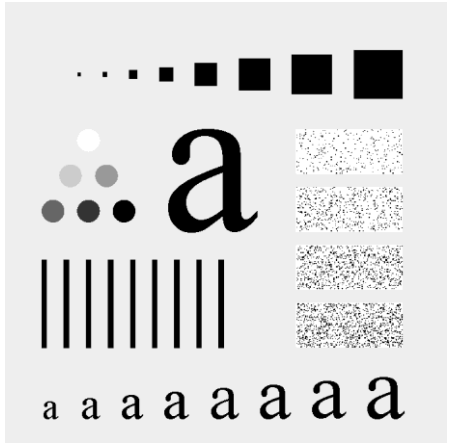


$$\frac{1}{273}$$

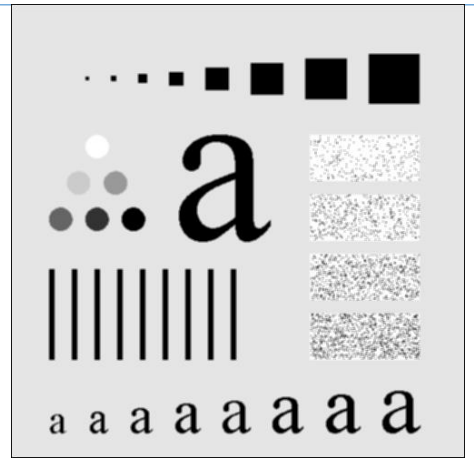
1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

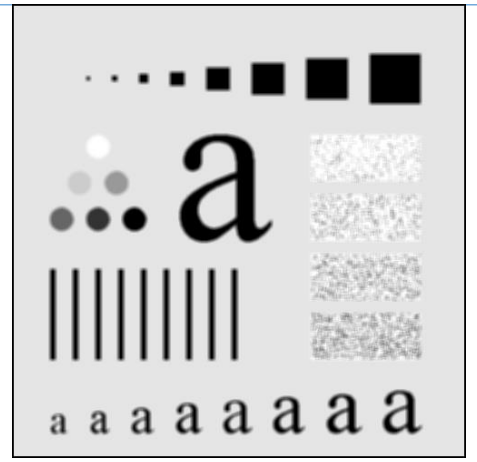
Low pass filter (Average filter)



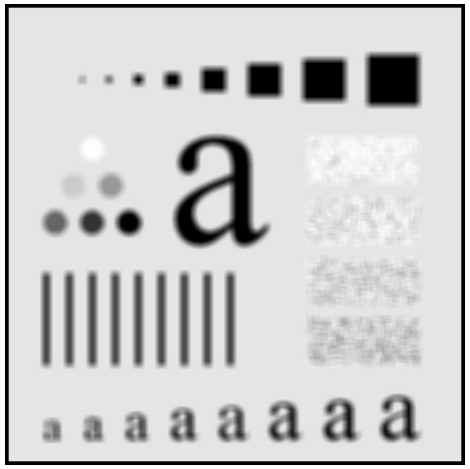
Original



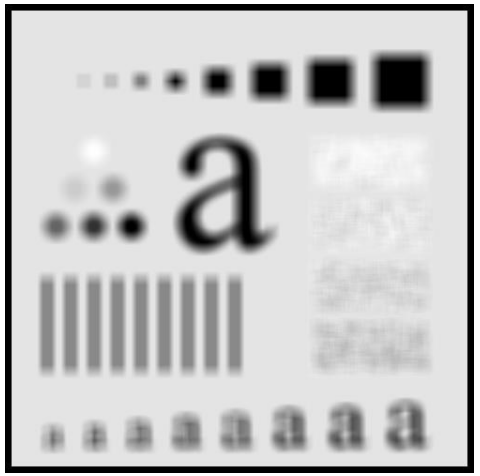
3 X 3



5 X 5



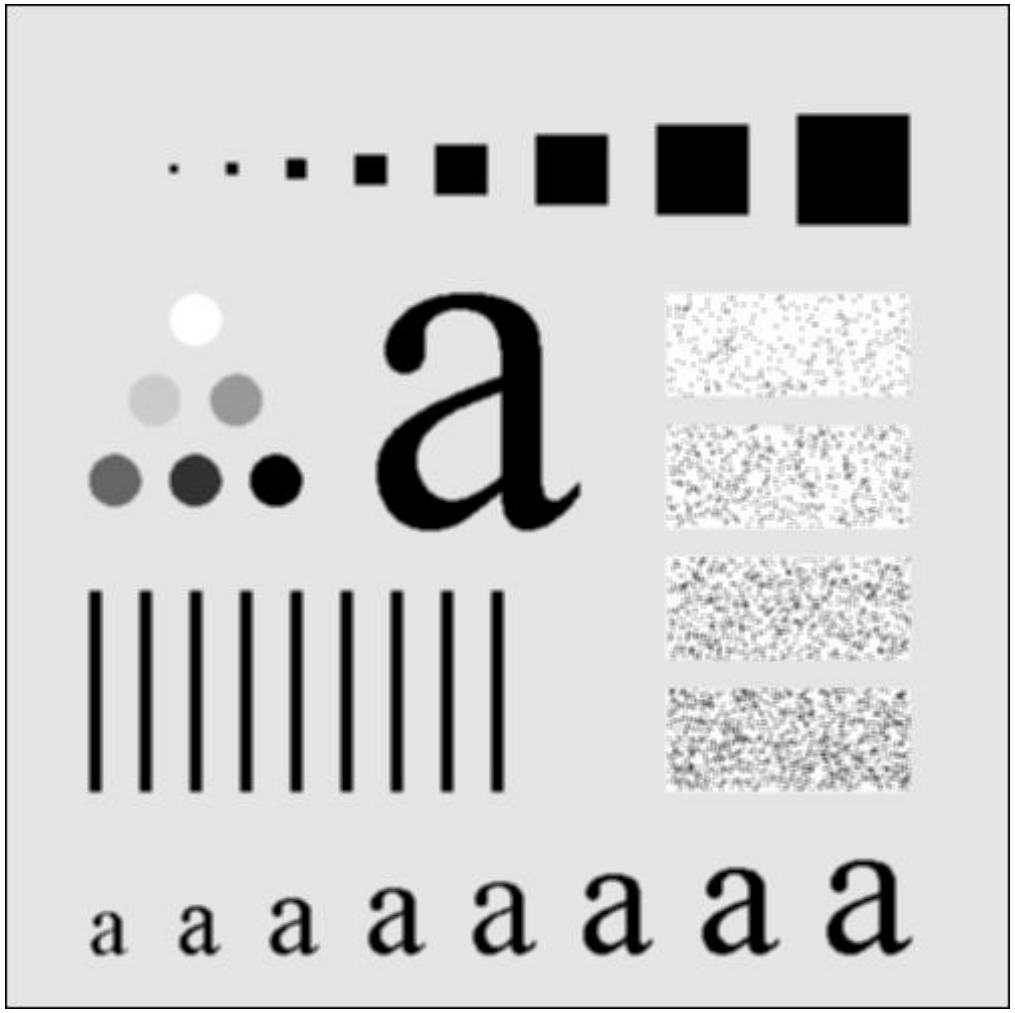
9 X 9



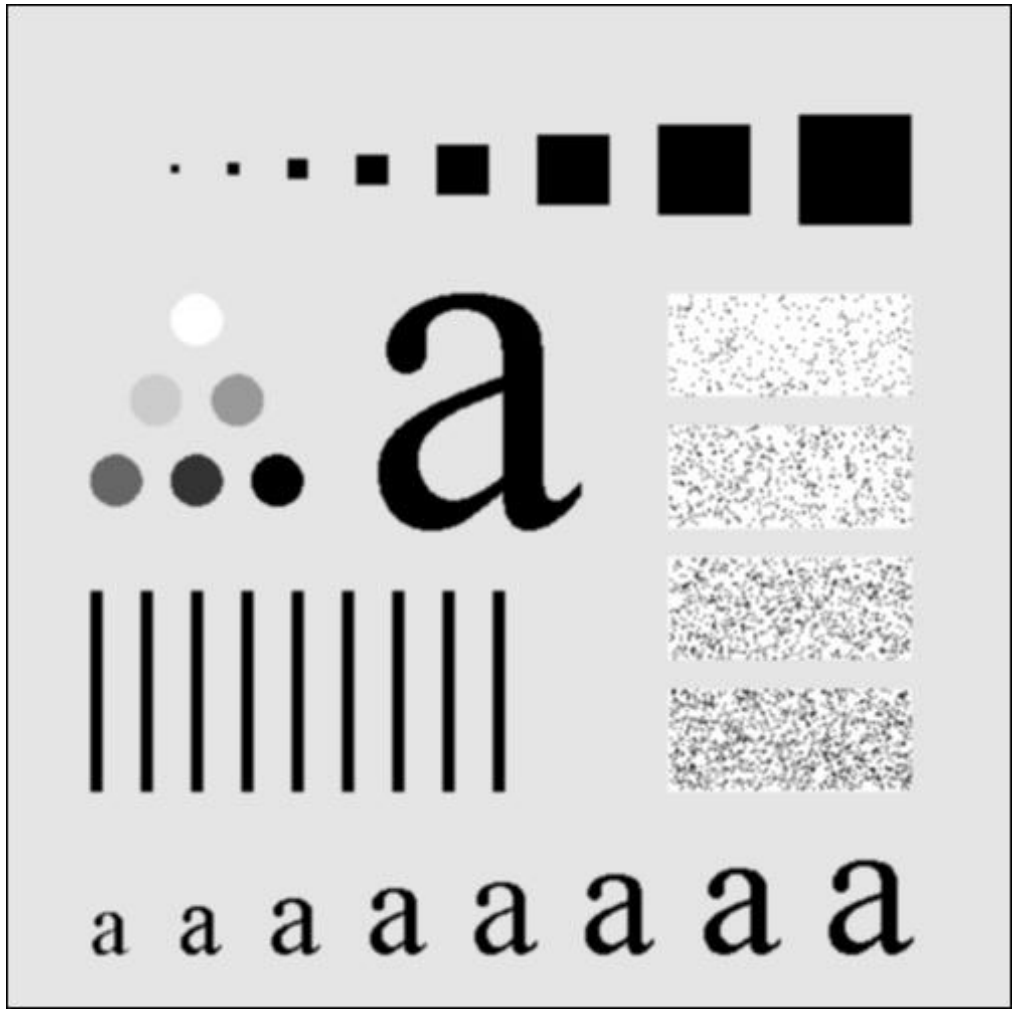
15 X 15

35 X 35

Low pass filter (Middle pixel weighted)



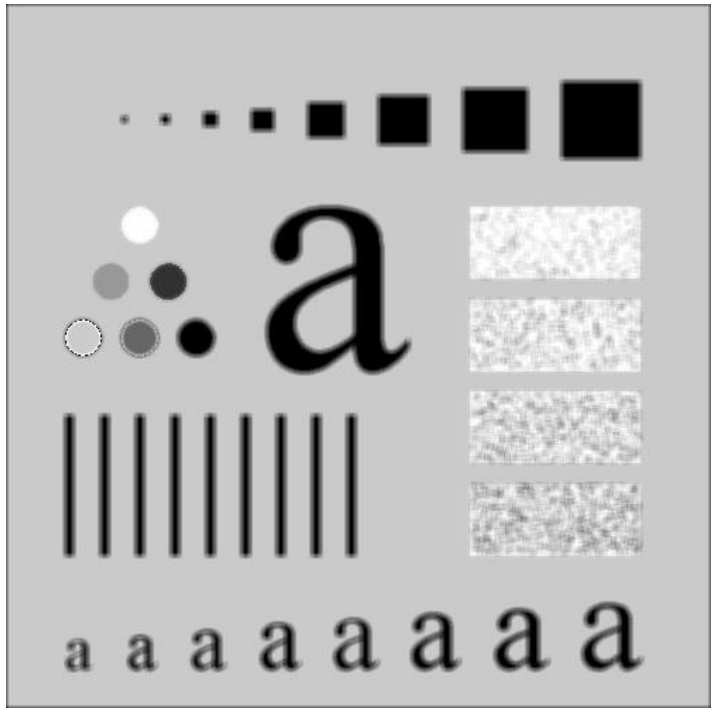
3x3 Average



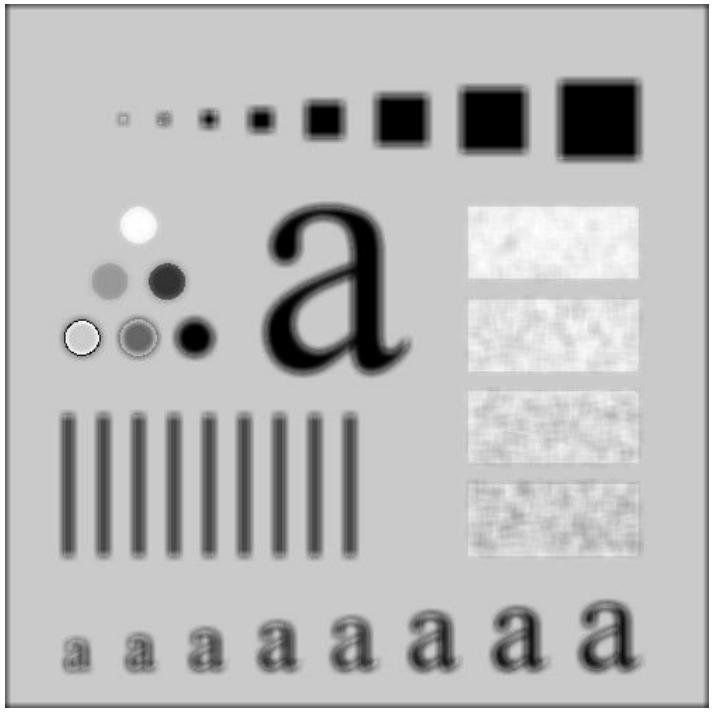
3x3 Middle pixel weighted

Low pass filter (Gaussian)

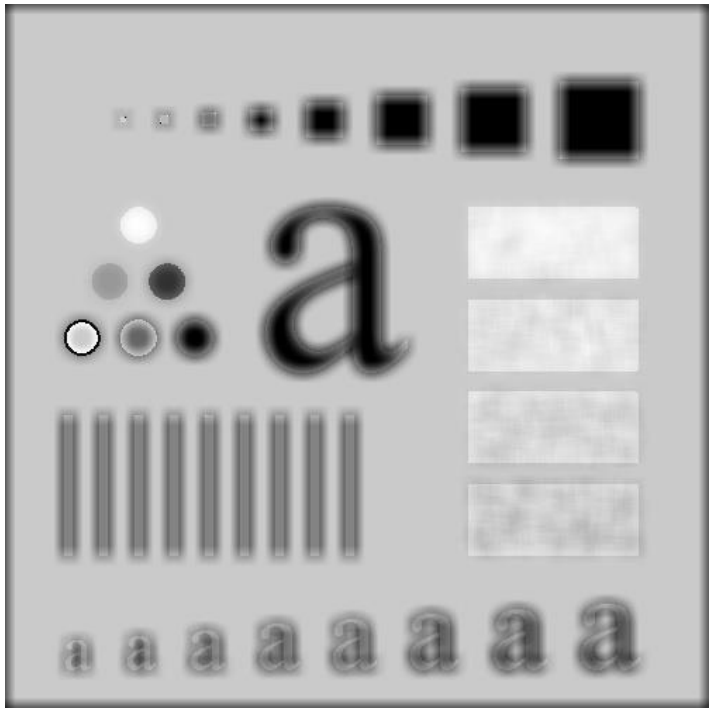
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



$\sigma = 10$ 5 X 5



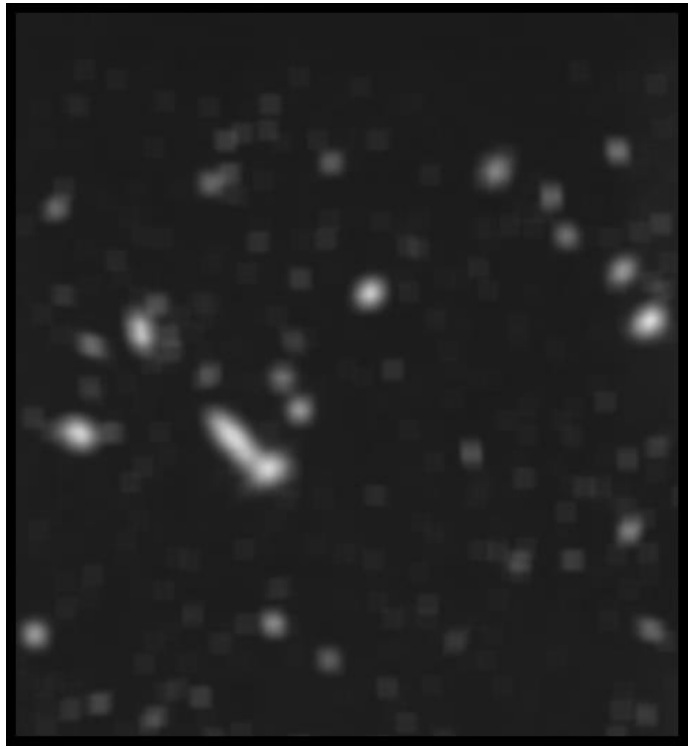
$\sigma = 10$ 9 X 9



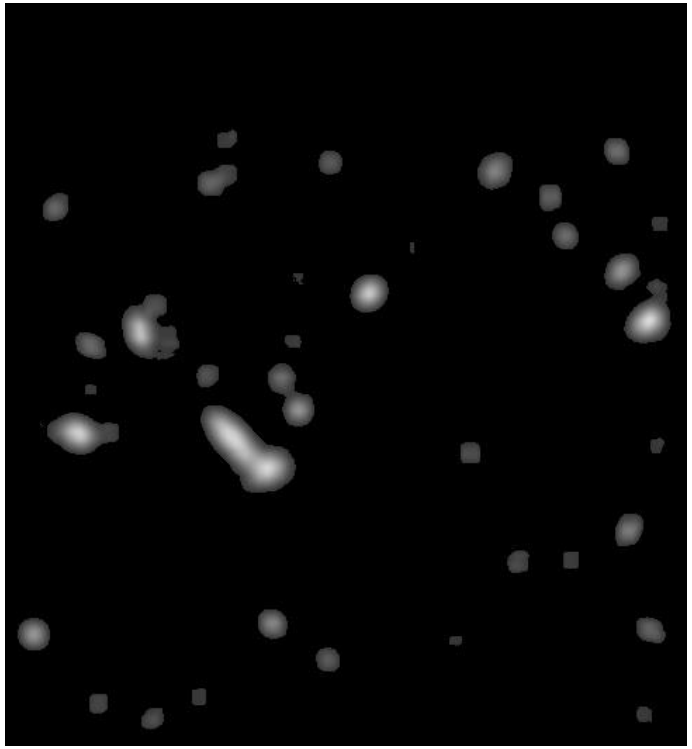
$\sigma = 10$ 15 X 15

Low pass filter application

Original



15 X 15 Average filter

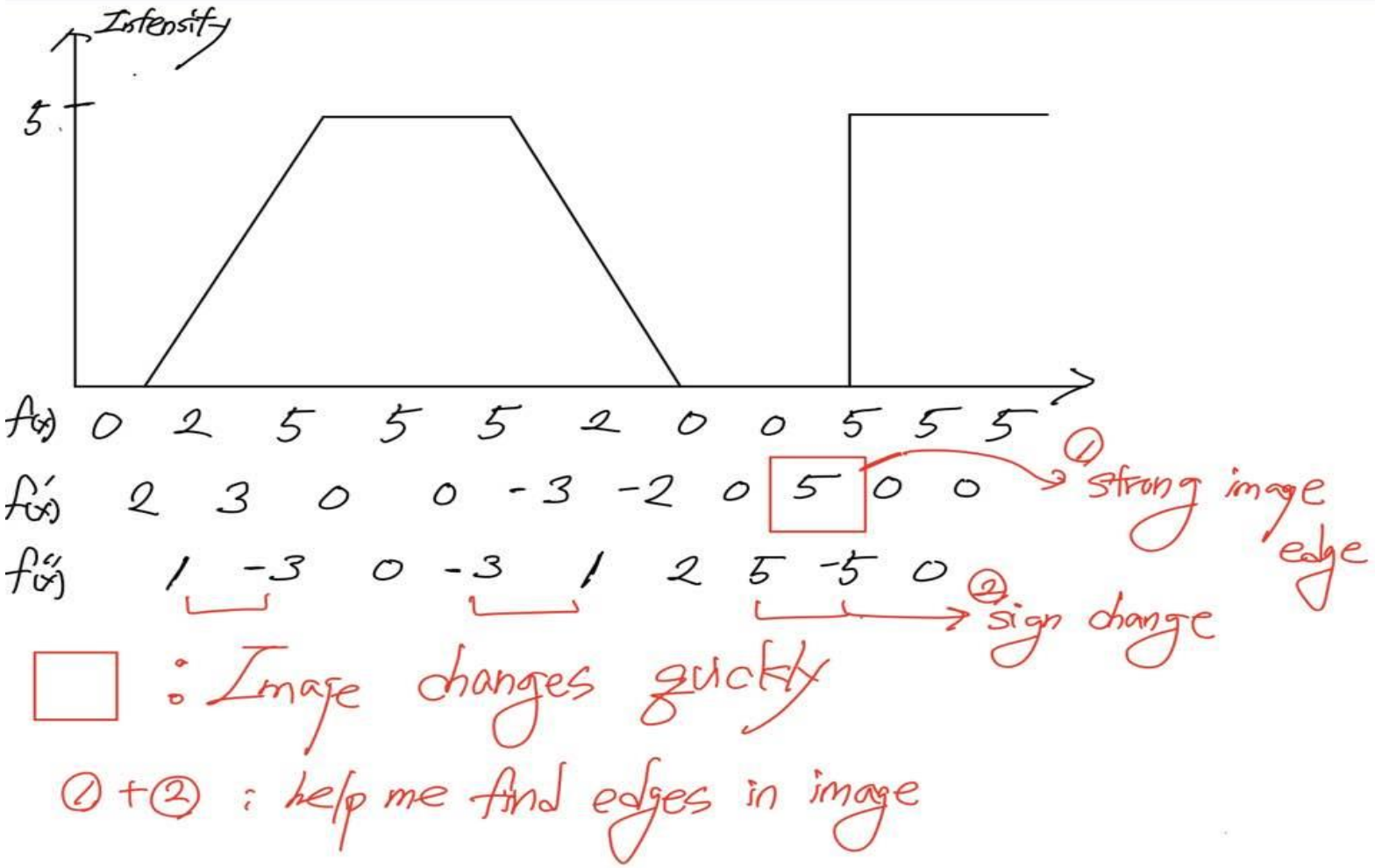


Thresholding

Sharpening Filter

Low pass filter -> Kind of Integration

Image Differenciation -> Result in Sharpening



Sharpening Filter (Laplacian)

First step in sharpening : Find a filter that reacts strongly to edges

In one dimension : [-1 1]
x x+1

G(x) = f(x+1) – f(x) = $\frac{df}{dx}$ ($\frac{df}{dx}$ is large when there is an edge)

[1 -2 1]
x-1 x x+1

G(x) = {f(x+1) – f(x)} – {f(x) – f(x-1)} = f(x+1) -2f(x) +f(x-1)= $\frac{d^2f}{dx^2}$ ($\frac{d^2f}{dx^2}$ show sign changes near edge)

In two dimension : Find edges in both x and y directions, we use an approximation of the Laplacian

Sharpening Filter (Laplacian)

In two dimension : Find edges in both x and y directions, we use an approximation of the Laplacian

Laplacian : $\nabla^2 F = \frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2}$

$F(x+1,y) + F(x-1,y) - 2F(x,y) + F(x,y+1) + F(x,y-1) - 2F(x,y)$

$= F(x+1,y) + F(x-1,y) + F(x,y+1) + F(x,y-1) - 4F(x,y)$

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

Constant : Zero response

Edge : Strongly react at middle point

How to enhance / Sharpen Images?

Idea : Strengthen edges of original image by adding a multiple of the edge map to it.
Sharpen image = Original image + Laplacian filter image

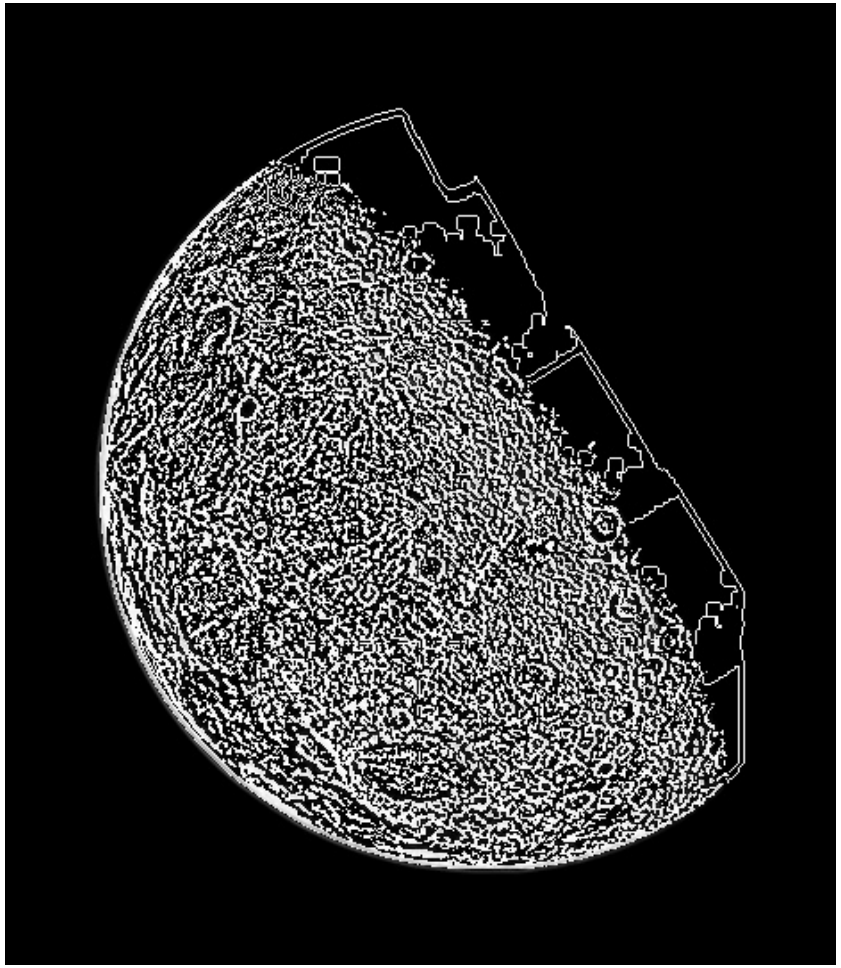
Result -> pros: Image sharpened up

cons: Also enhance the noise , Saturation by bumping up middle pixel

->Intensity normalization(scaling) requires.

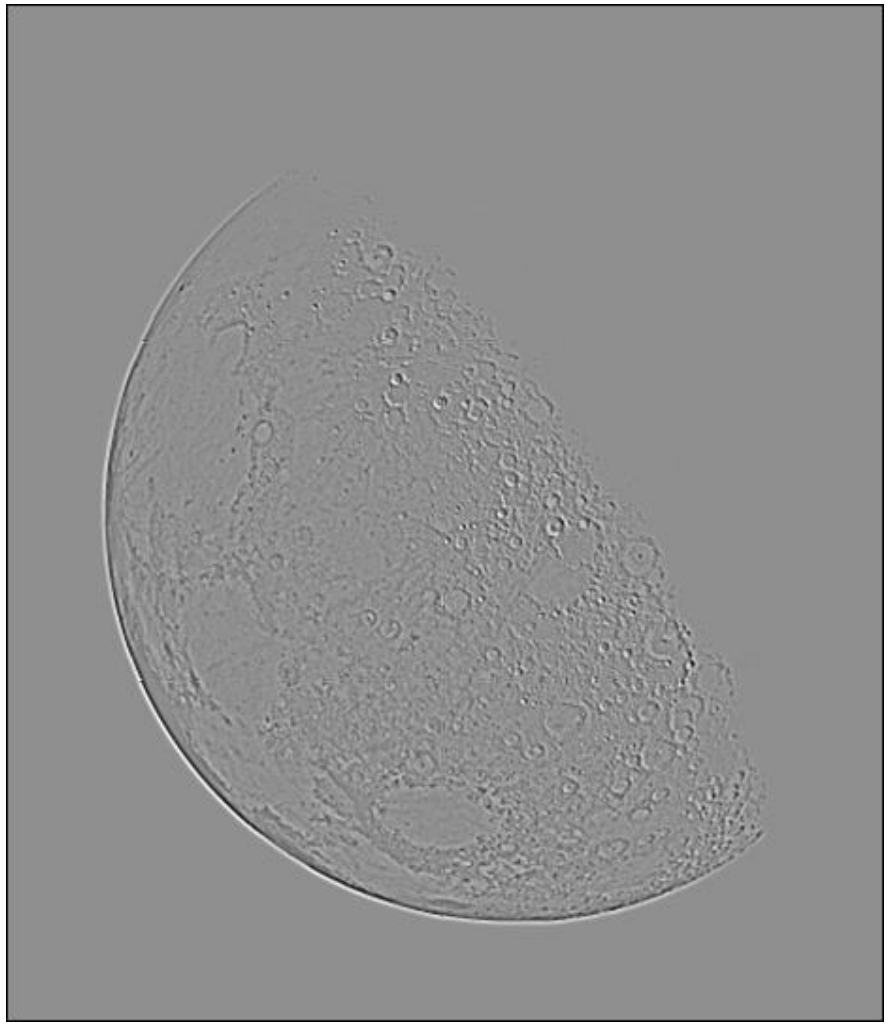
Laplacian Filter

Original



Laplacian filter result (No scaling)

Laplacian Filter



Laplacian filter(Min scaling)

Laplacian filter(Zero scaling)

Laplacian Filter

Original

Laplacian filter

Laplacian Filter

Original

Diagonal Laplacian filter

Laplacian Filter

Original

Laplacian filter

Diagonal Laplacian filter

Unsharp masking

Noise is also sharpened (Inevitable)
->We can just add a Franction of the edges back in , for a more subtle effect

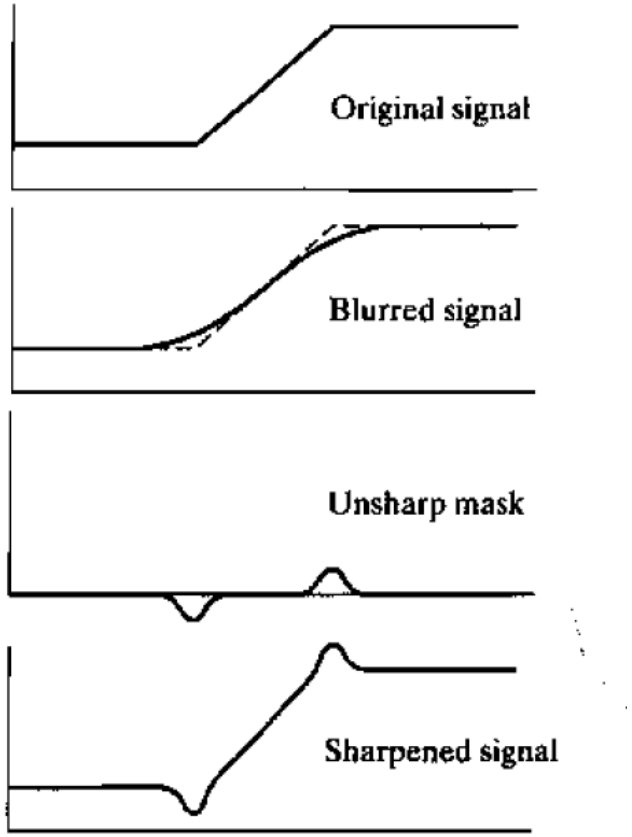
Original : F

Low-pass images: F′

Edge image : F- F′ = F_mask

Output = F + k * F_mask
(k = tunable parameter)

If k >1 -> high boosting



Unsharp masking

Original

5 x 5 Gaussian LPF blurred image

Unsharp mask

Unsharp masking and high boost filtering

Original

Unsharp masking k=1

Unsharp masking k=3

Unsharp masking k=4.5

Sobel Operator

Filter needs not be symmetric

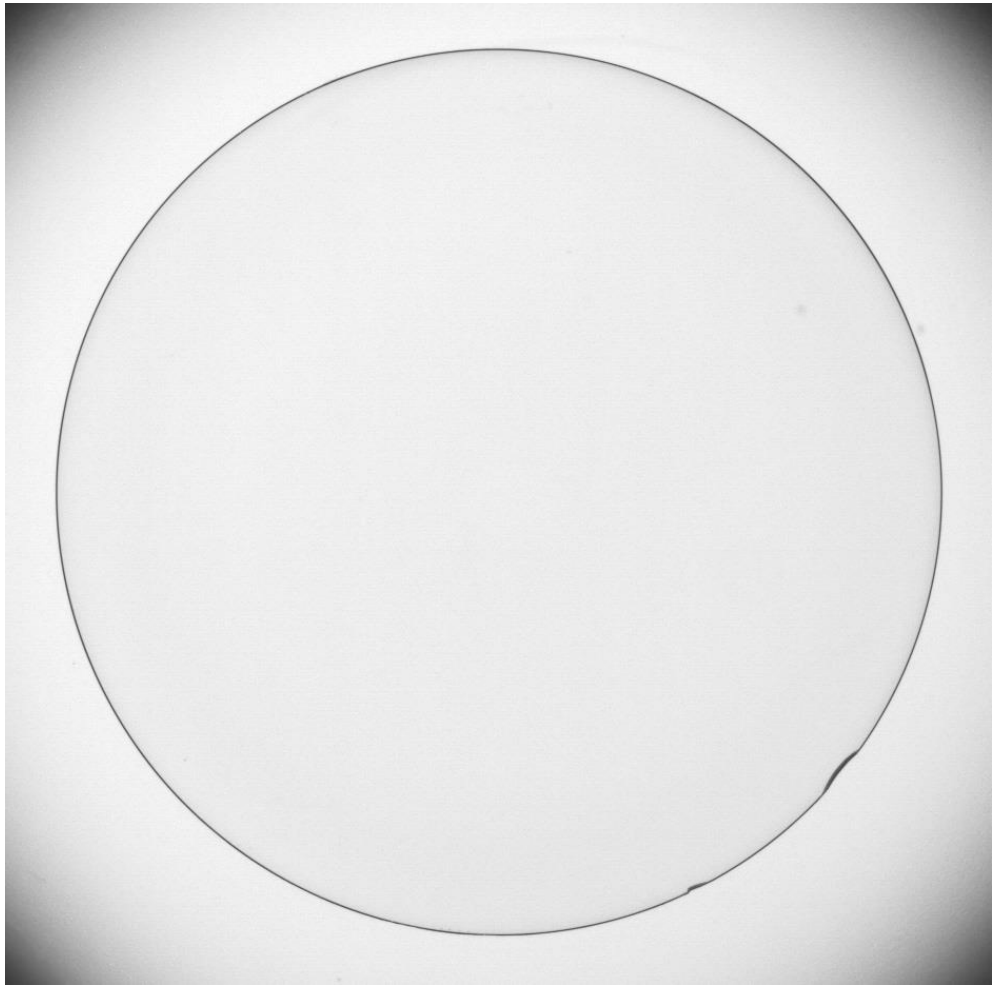
-1	-2	-1
0	0	0
1	2	1

Sobel horizontal edge detector

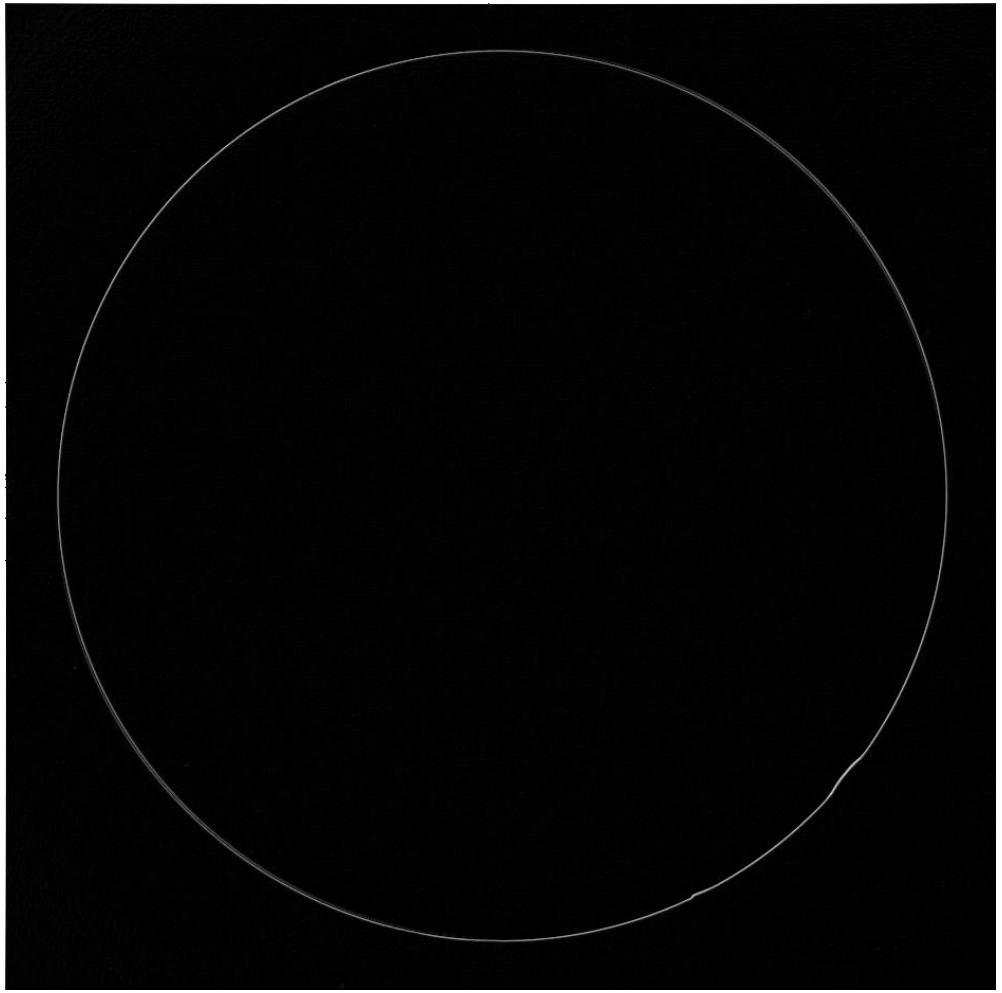
-1	0	1
-2	0	2
-1	0	1

Sobel vertical edge detector

Sobel Operator

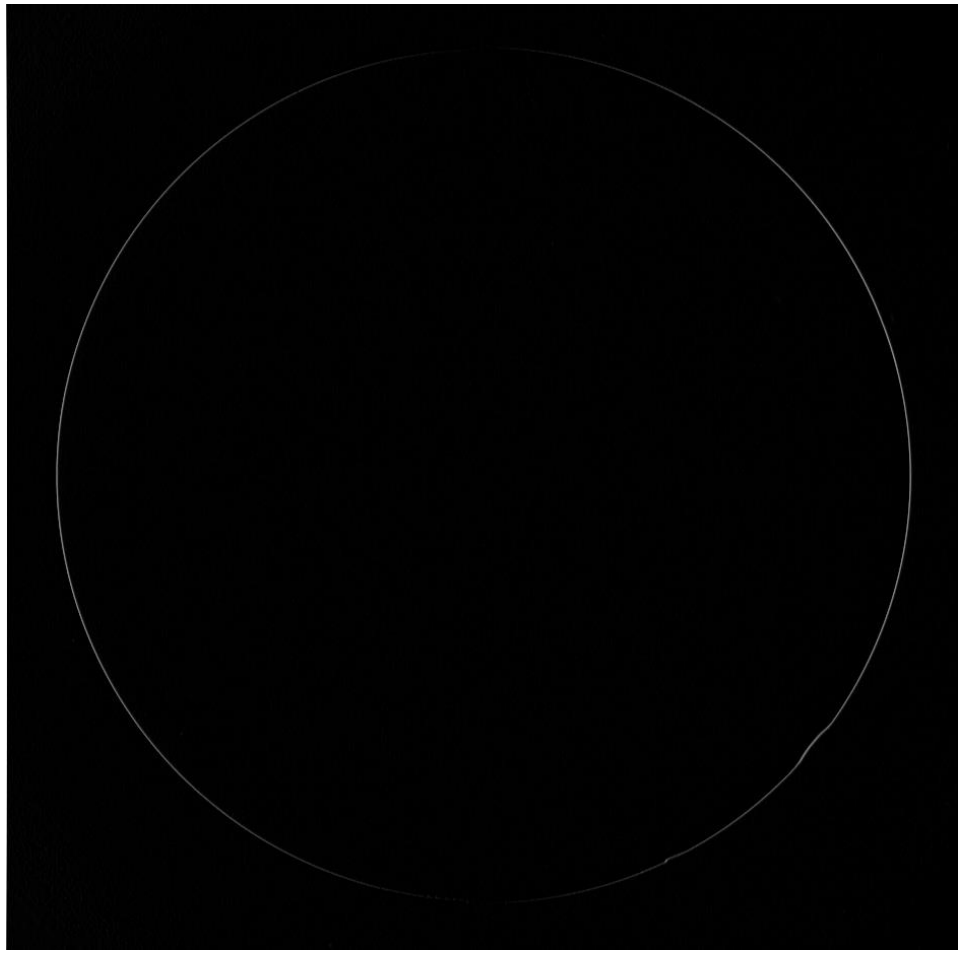


Original

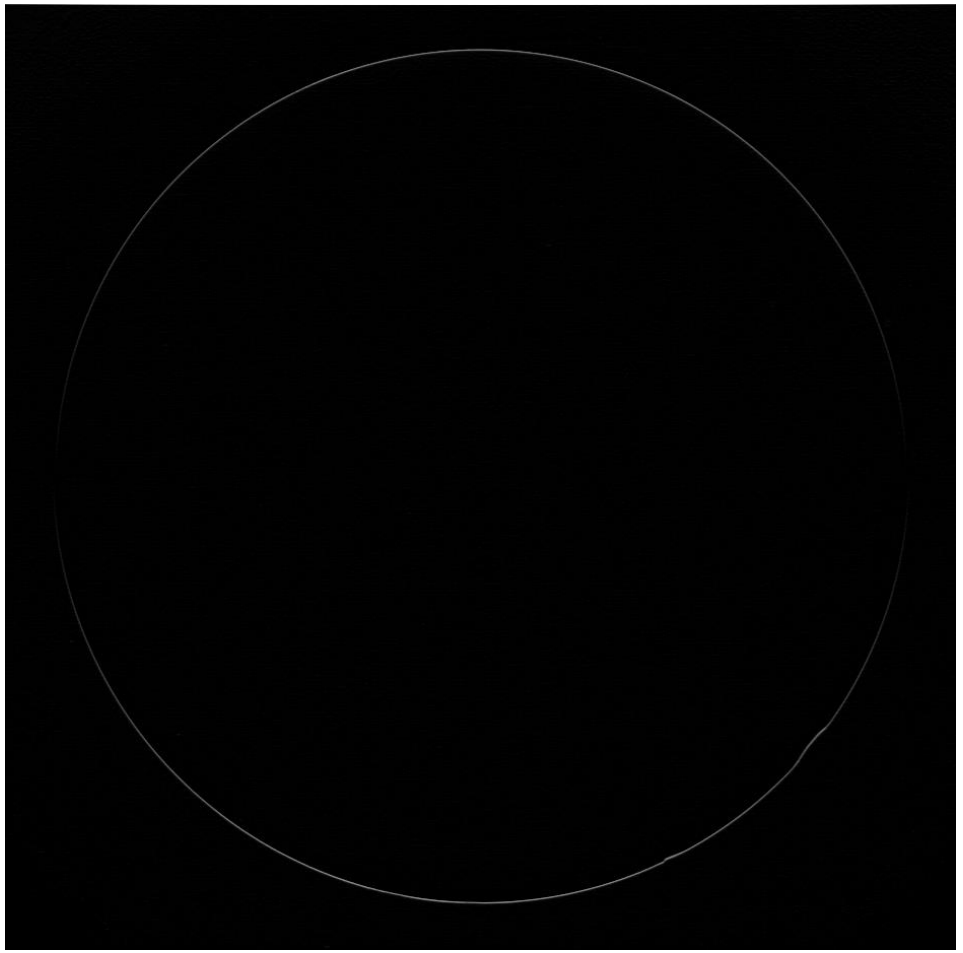


Sobel operator

Sobel Vertical, Horizontal

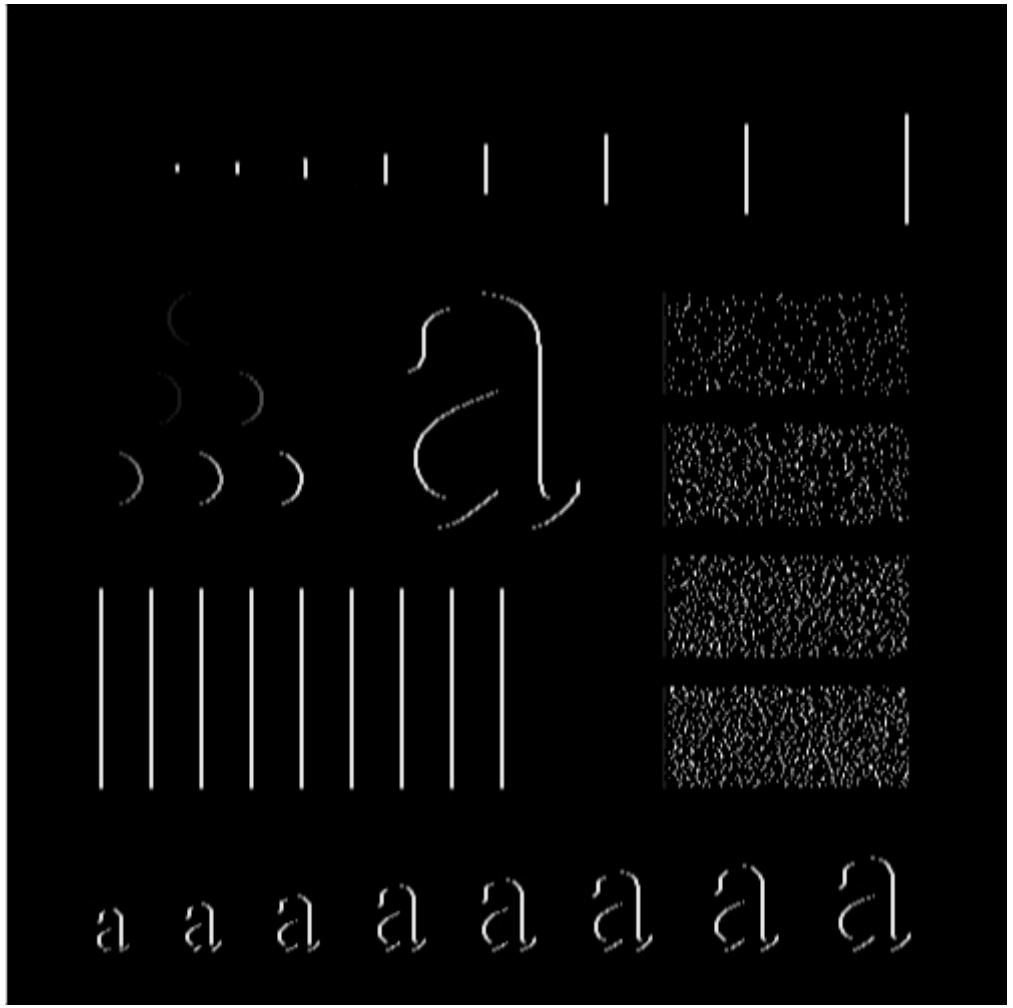


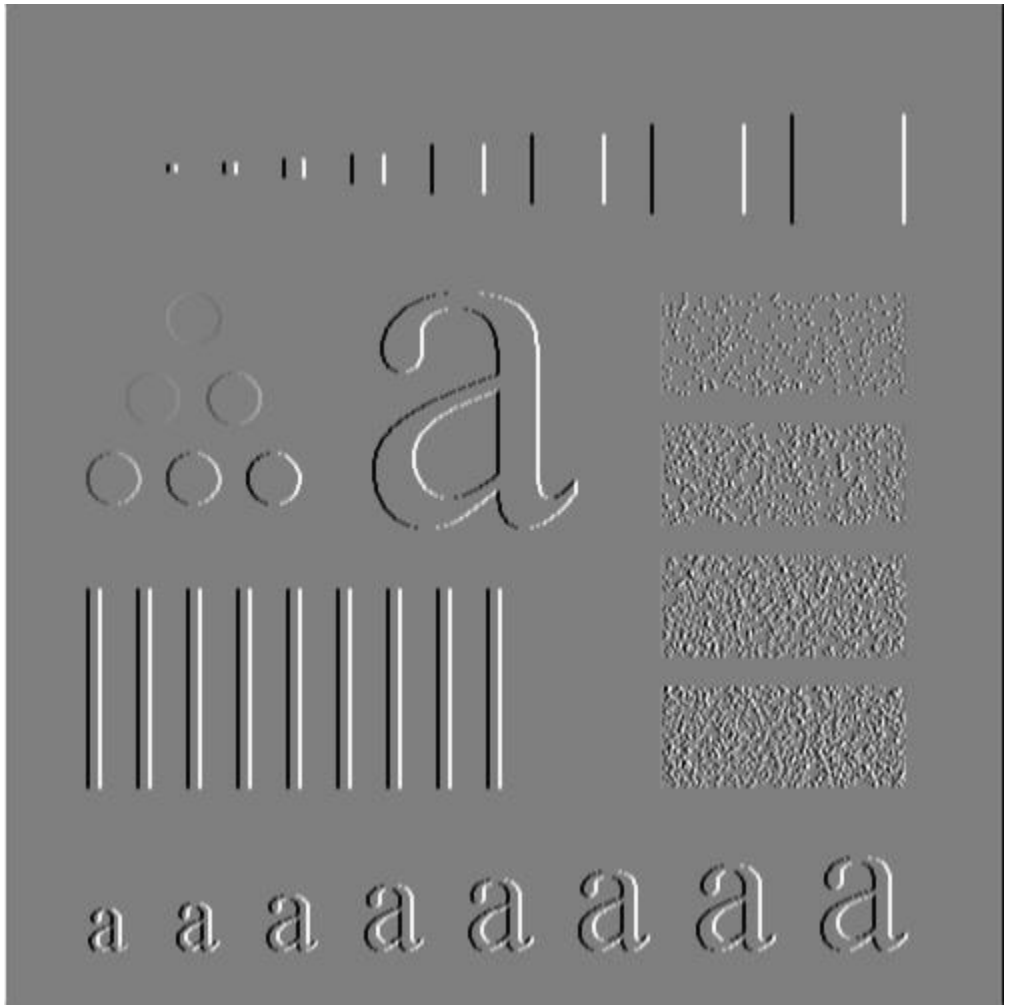
Sobel vertical



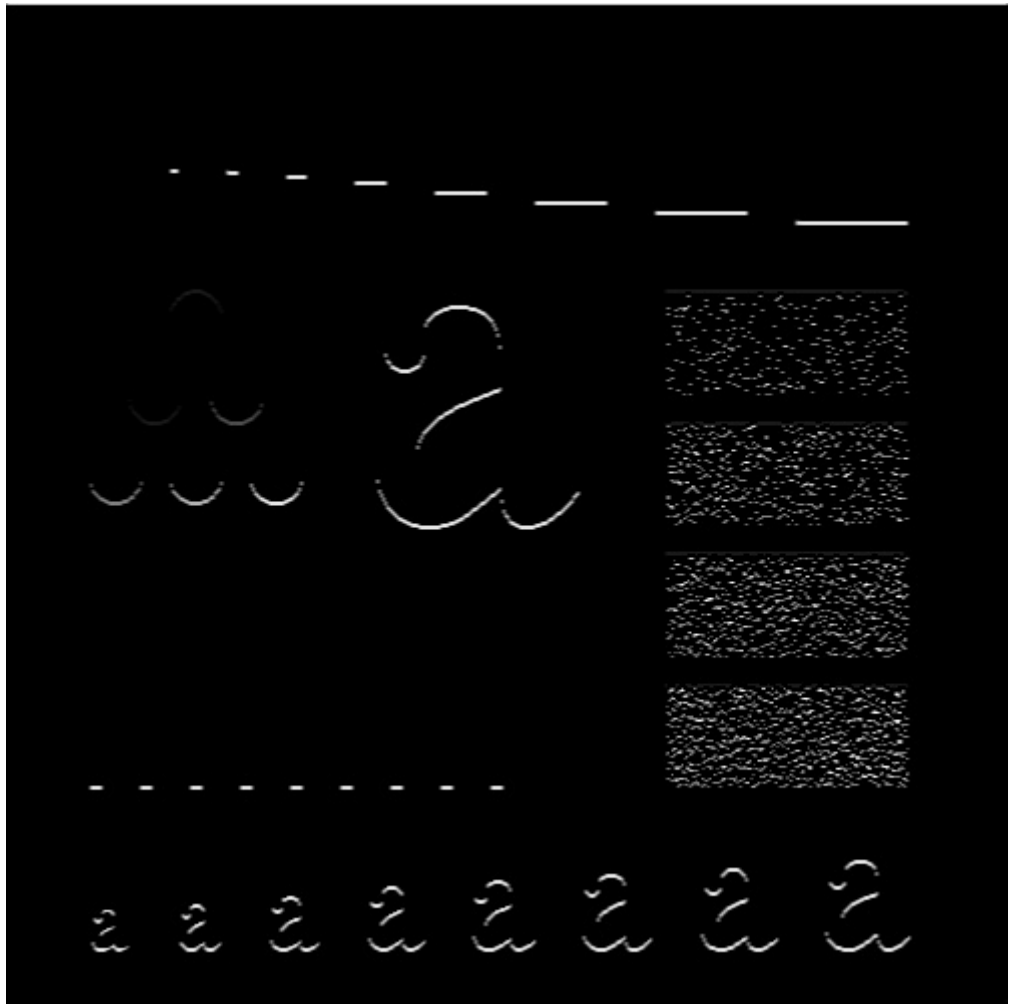
Sobel horizontal

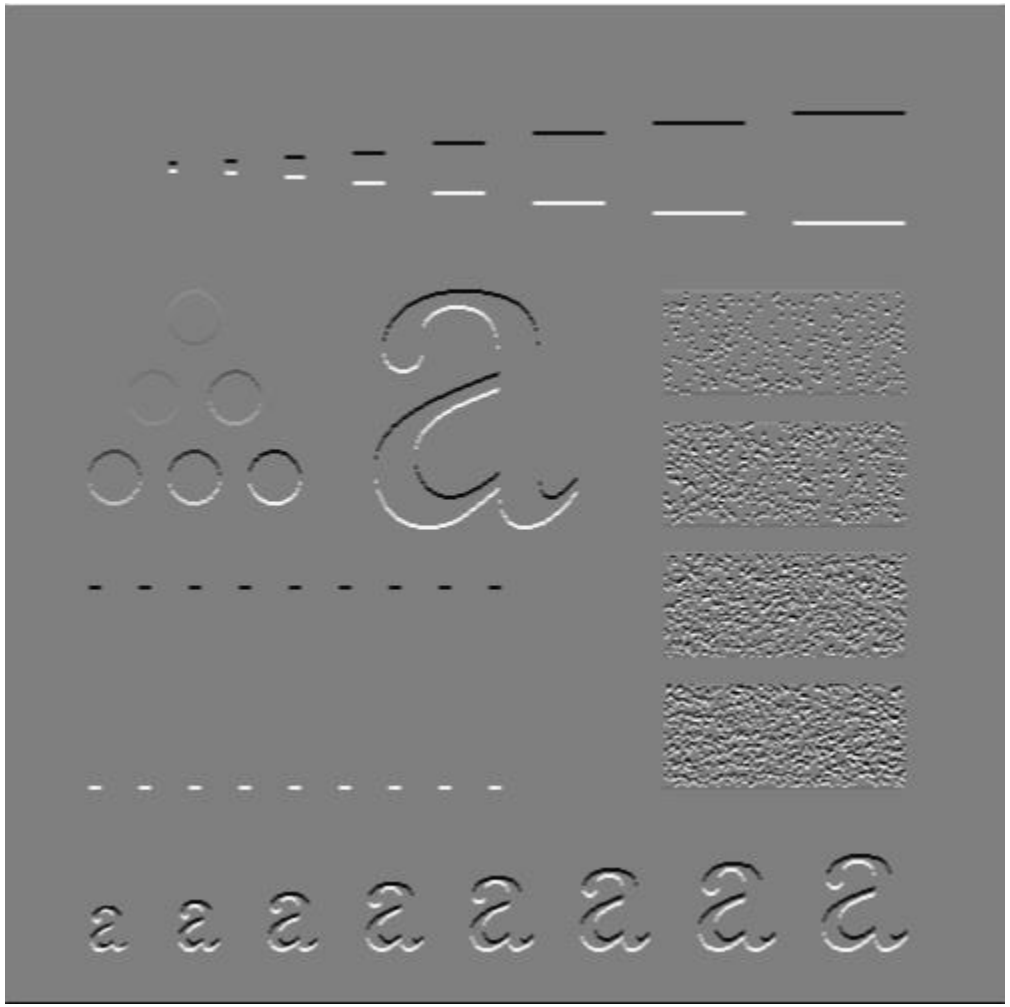
Sobel Vertical





Sobel Horizontal



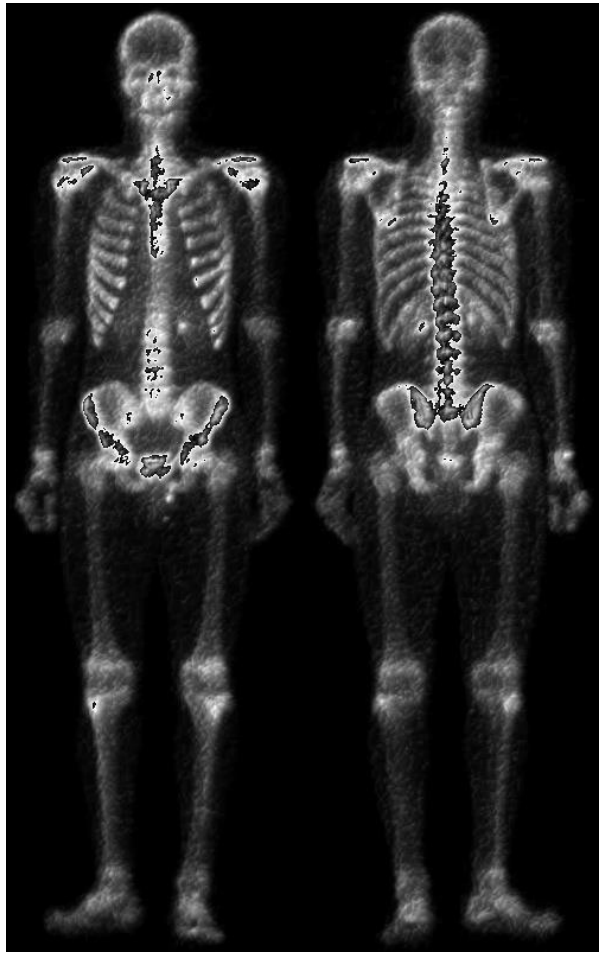


Unsharp masking & Sobel operator

Original

Unsharp masking k=1

Sobel + Unsharp masking

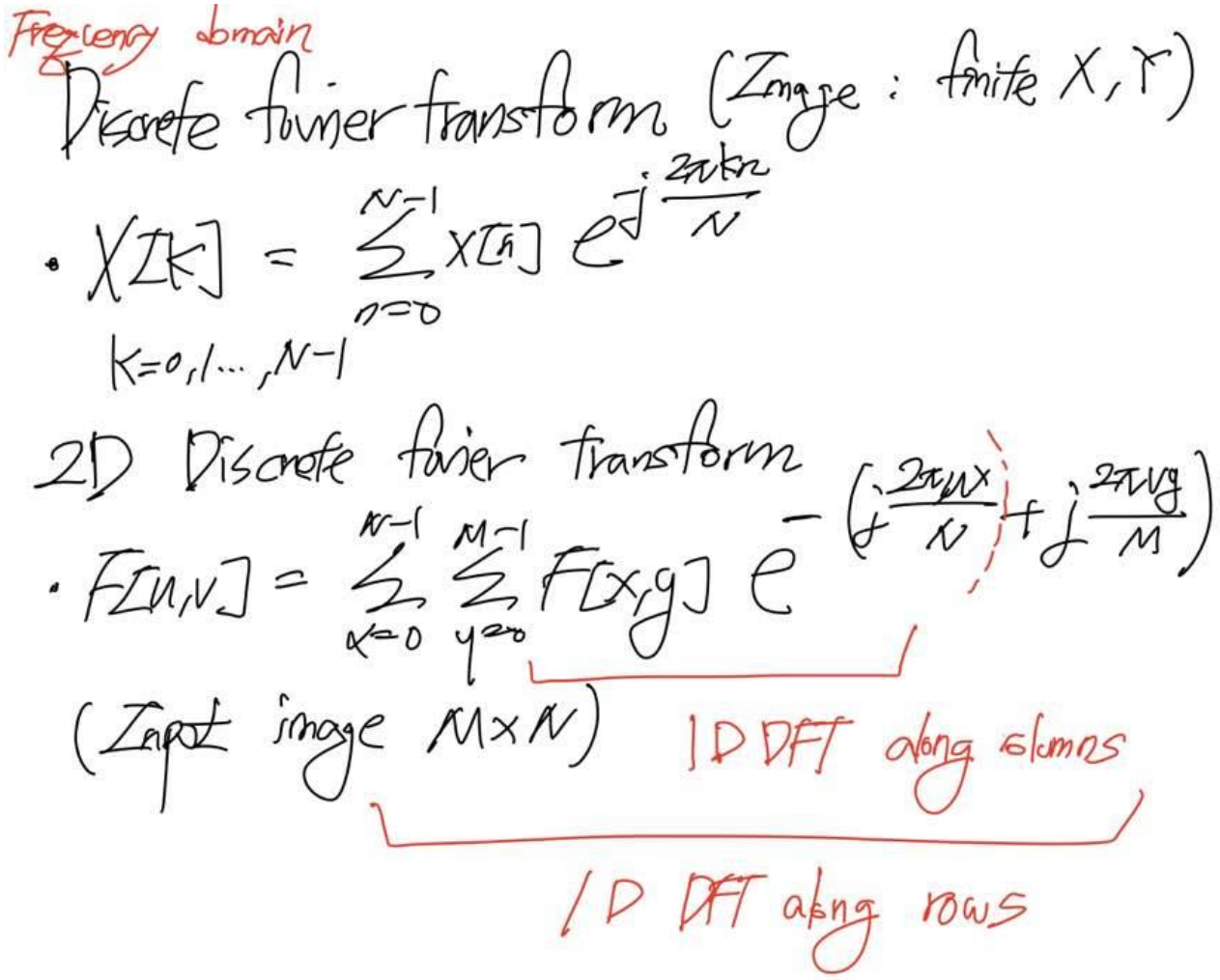


5X5 smoothing image

Frequency domain filtering

1. 2D DFT
2. Filtering
3. LPF,HPF
4. Notch Filter

Frequency domain 2D DFT



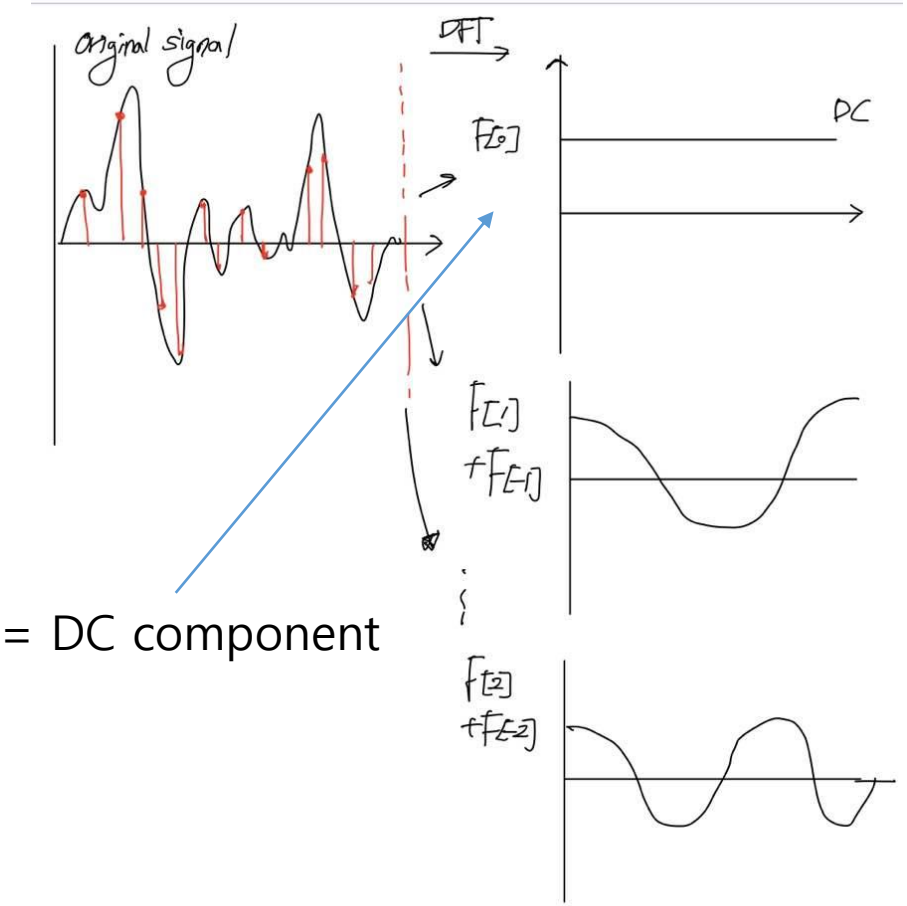
Convolution in spatial domain

= Multiplication of two DFT in frequency domain

Frequency domain 2D DFT

2D DFT is like a decomposition of image into complex exponentials(sin and cos)

$$e^{-j\frac{2\pi ux}{N}} = \cos\frac{2\pi ux}{N} - j\sin\frac{2\pi ux}{N}$$



Original signal = A different set of basis functions

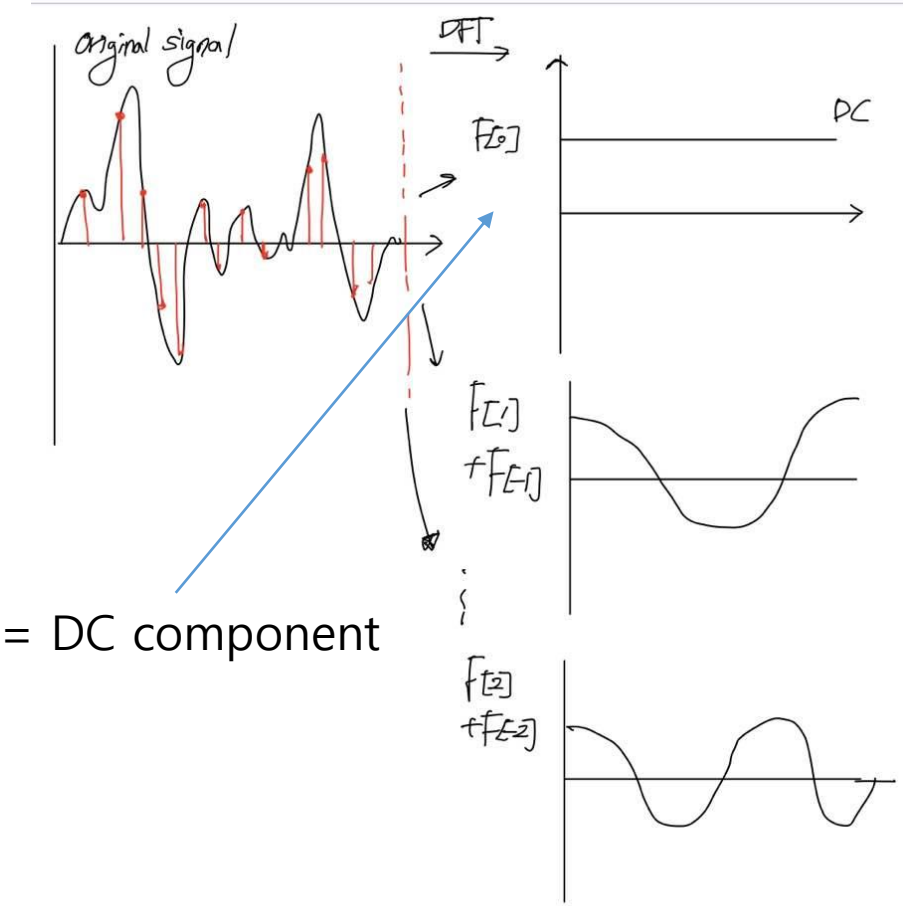
Transforming image into a certain better basis
will help us throw away perceptually
insignificant parts of the image

Basis function F[0,0] = DC component
->Flat image

Frequency domain 2D DFT

2D DFT is like a decomposition of image into complex exponentials(sin and cos)

$$e^{\frac{-j2\pi ux}{N}} = \cos\frac{2\pi ux}{N} - j\sin\frac{2\pi ux}{N}$$

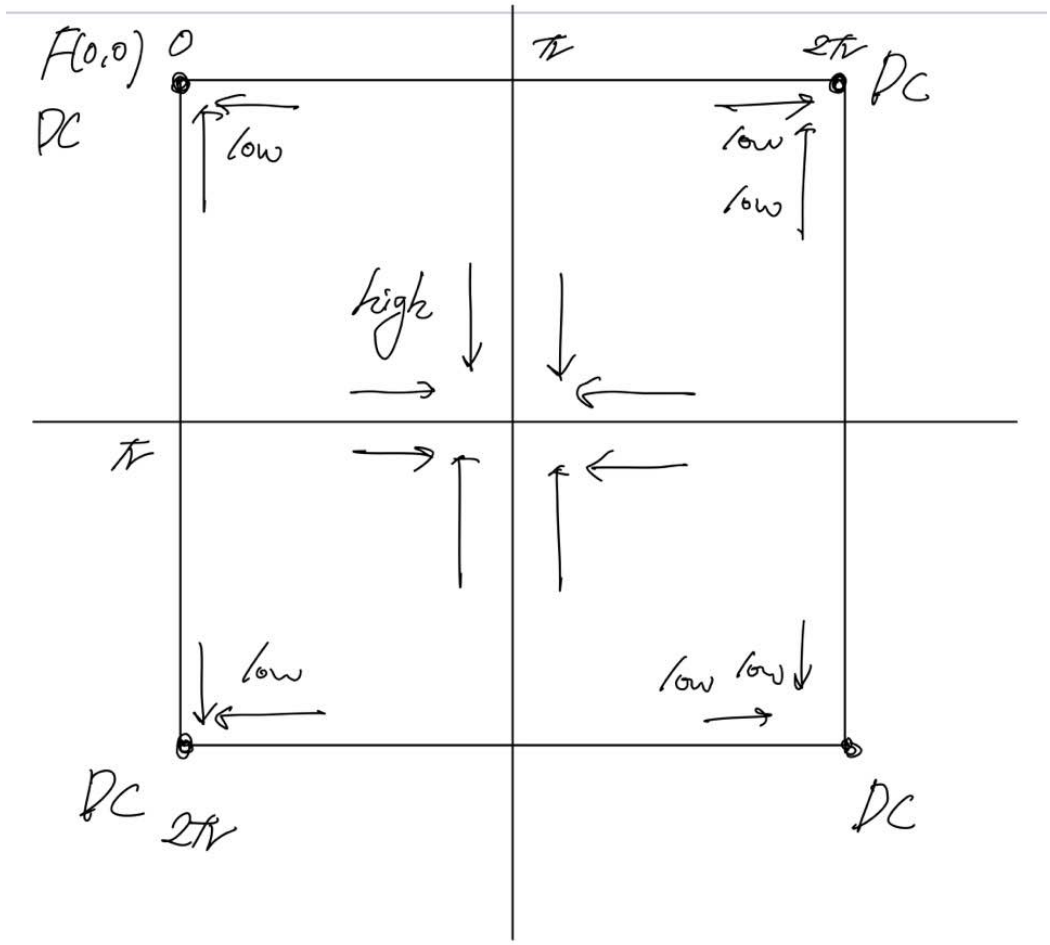


Original signal = A different set of basis functions

Transforming image into a certain better basis
will help us throw away perceptually
insignificant parts of the image

Basis function F[0,0] = DC component
->Flat image

Image region in frequency



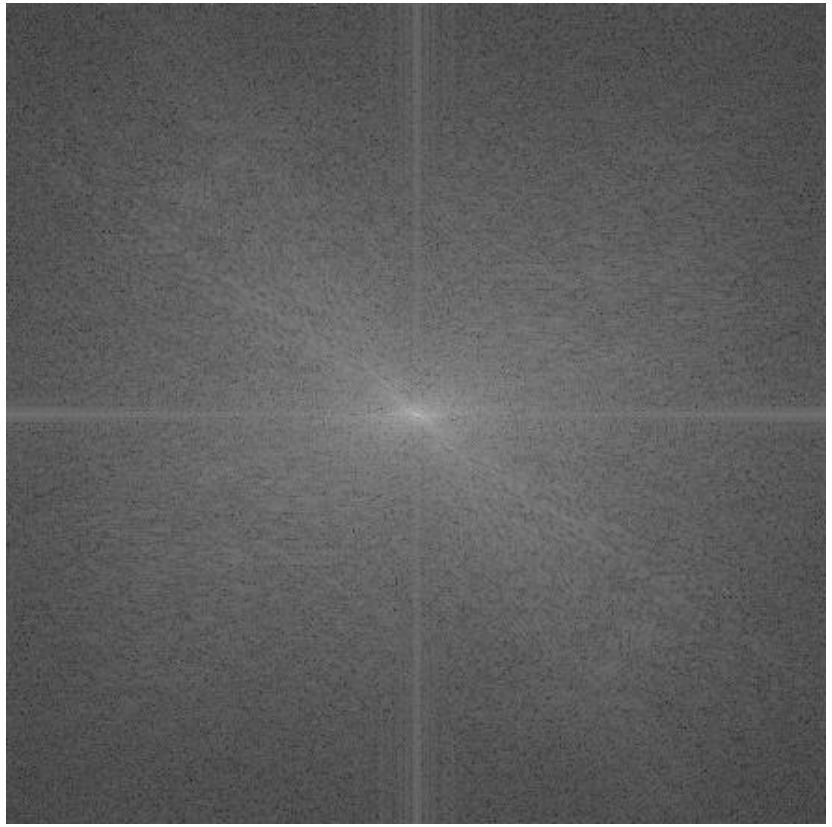
$$e^{\frac{-j2\pi ux}{N}} = \cos\frac{2\pi ux}{N} - j\sin\frac{2\pi ux}{N}$$

Low frequency
: flat, slowly varying region in spatial domain

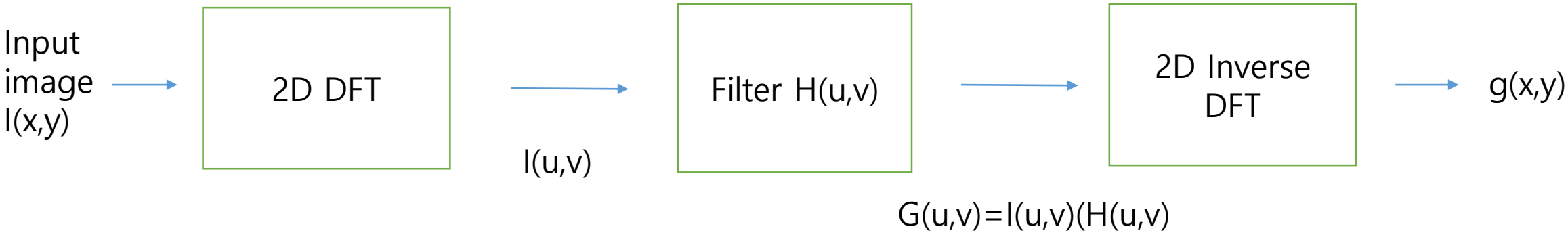
High frequency
:edge, noisy region in spatial domain

Using fftshift
-> Shift DC component into cental.

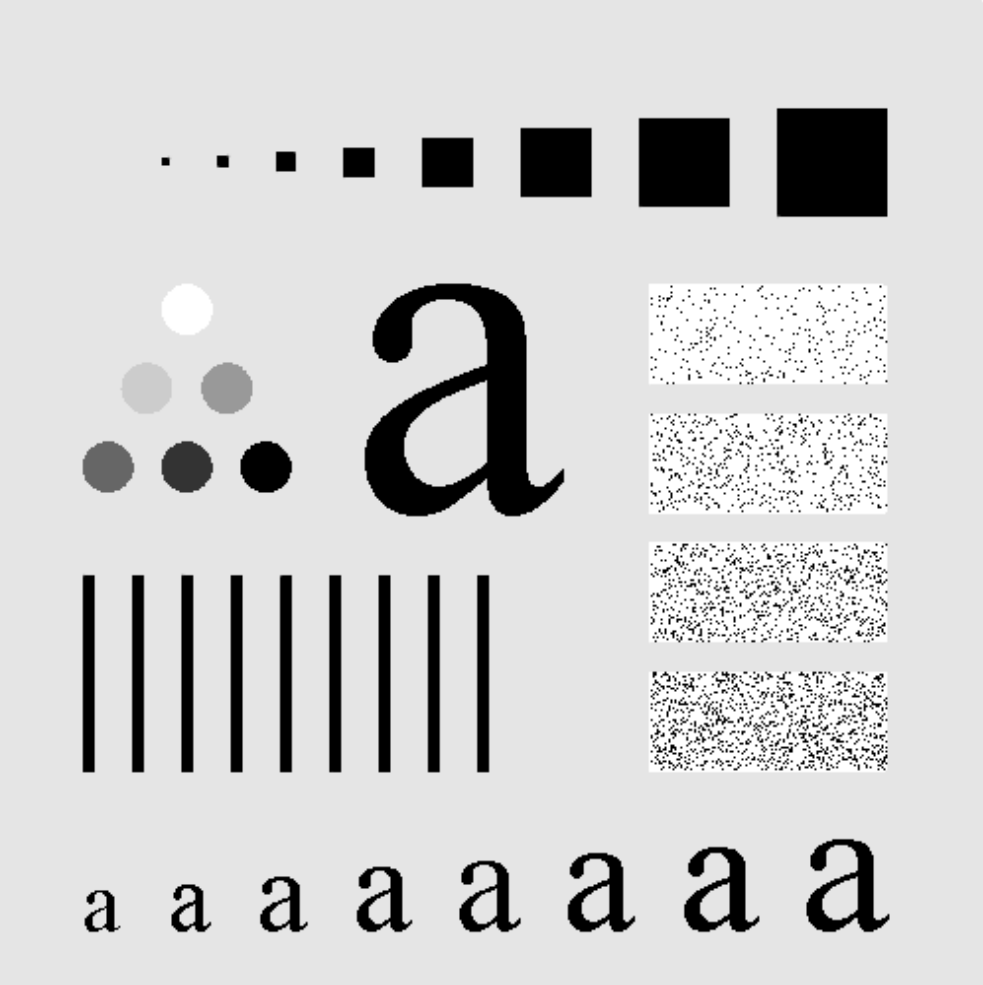
Image region in frequency

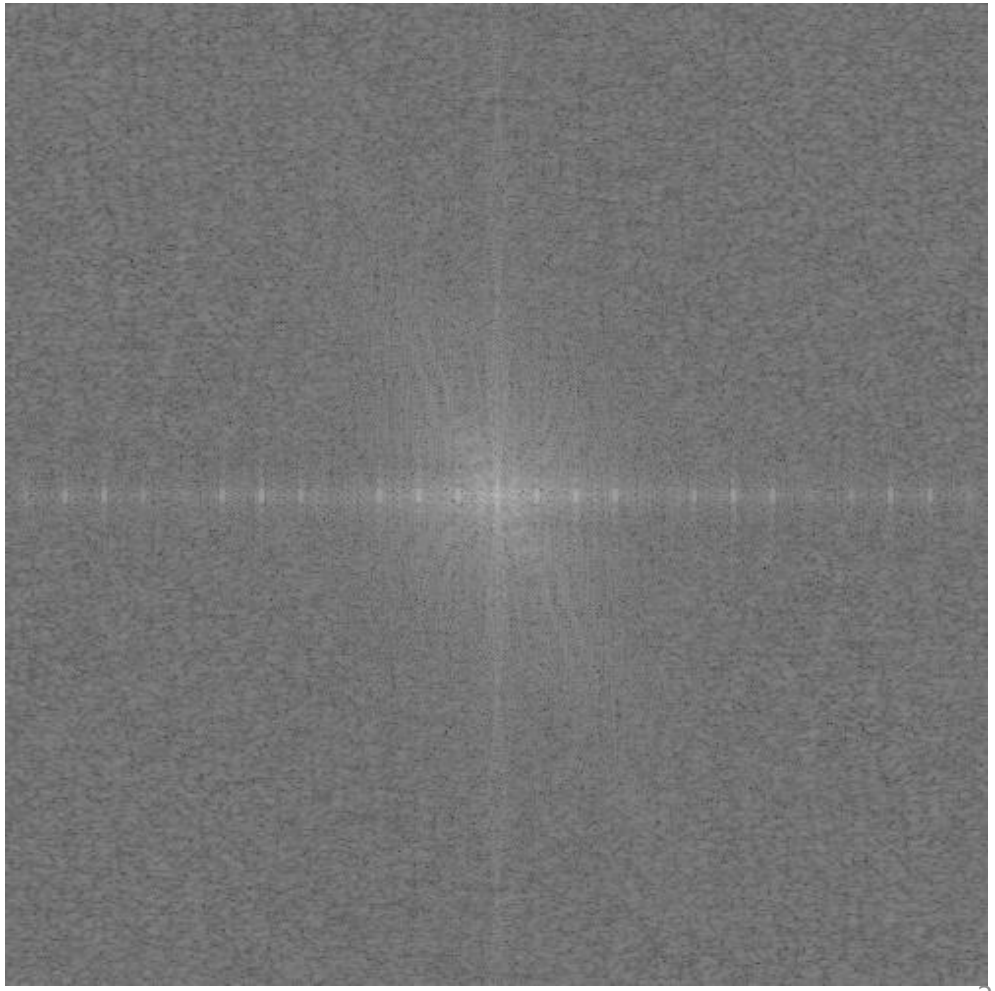


Filtering process

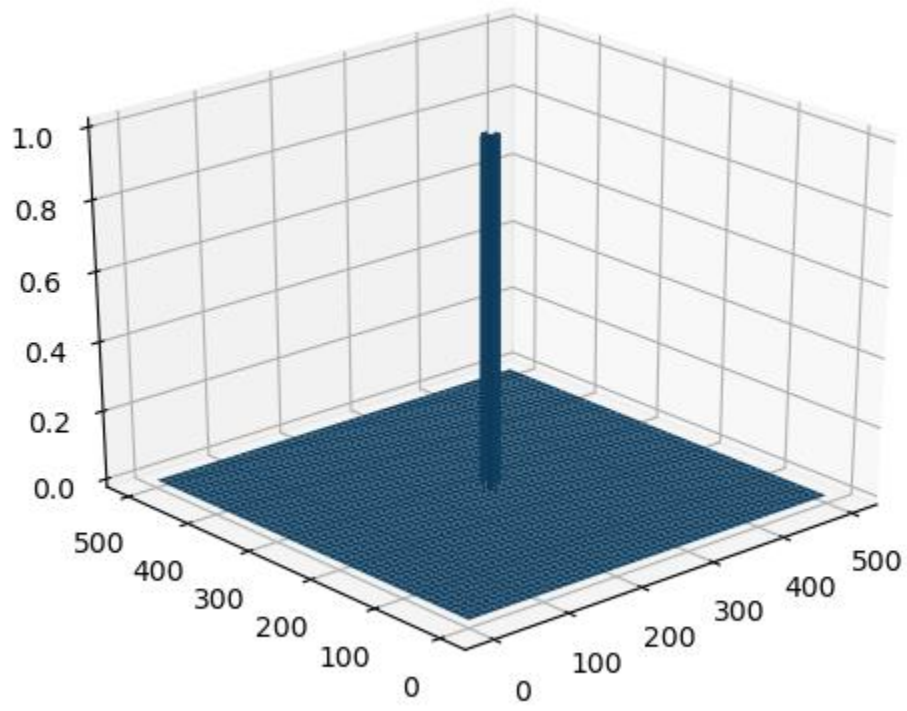


Filtering process

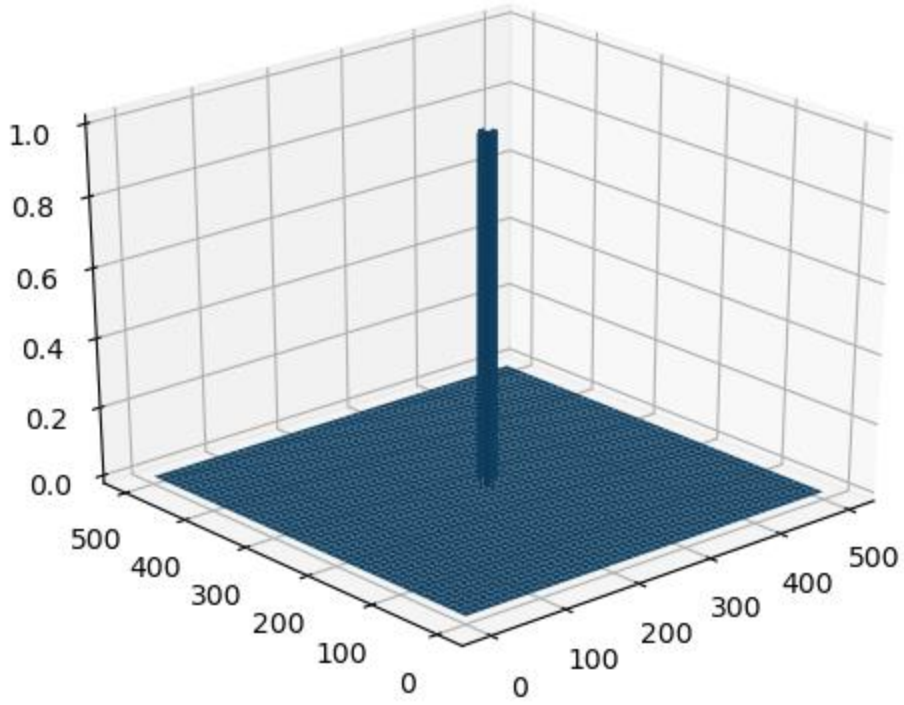




Ideal Low pass filter

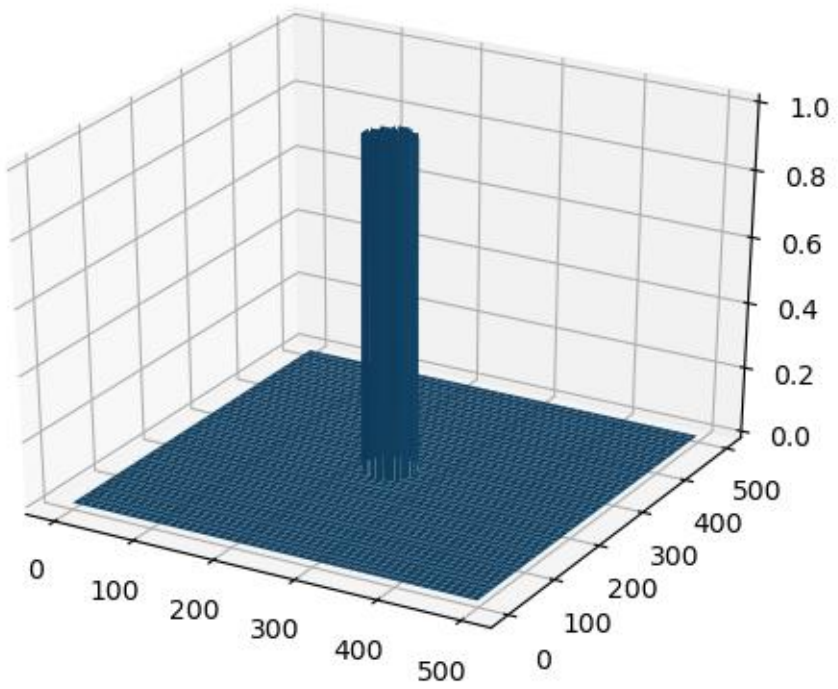


Ideal Low pass filter



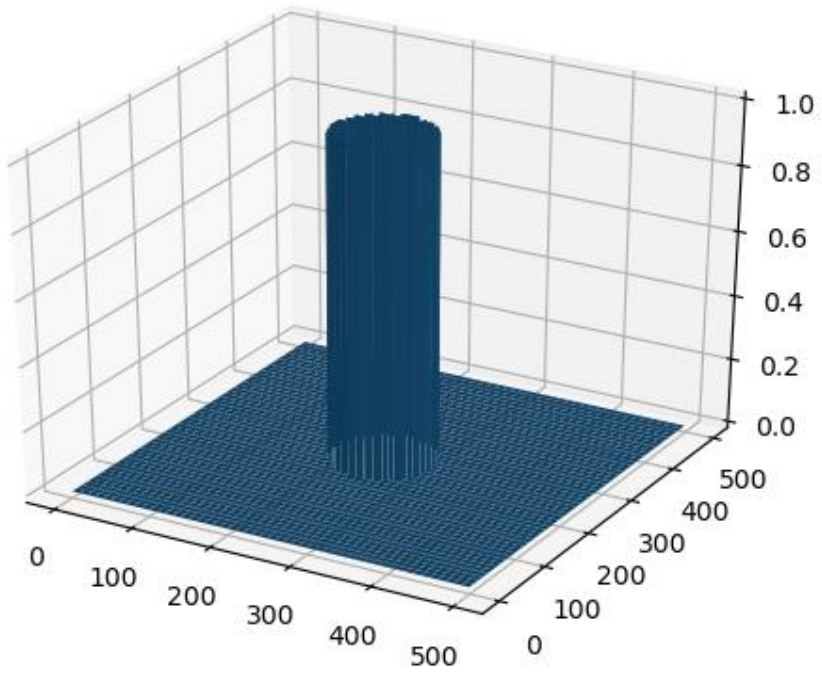
R =10

Ideal Low pass filter



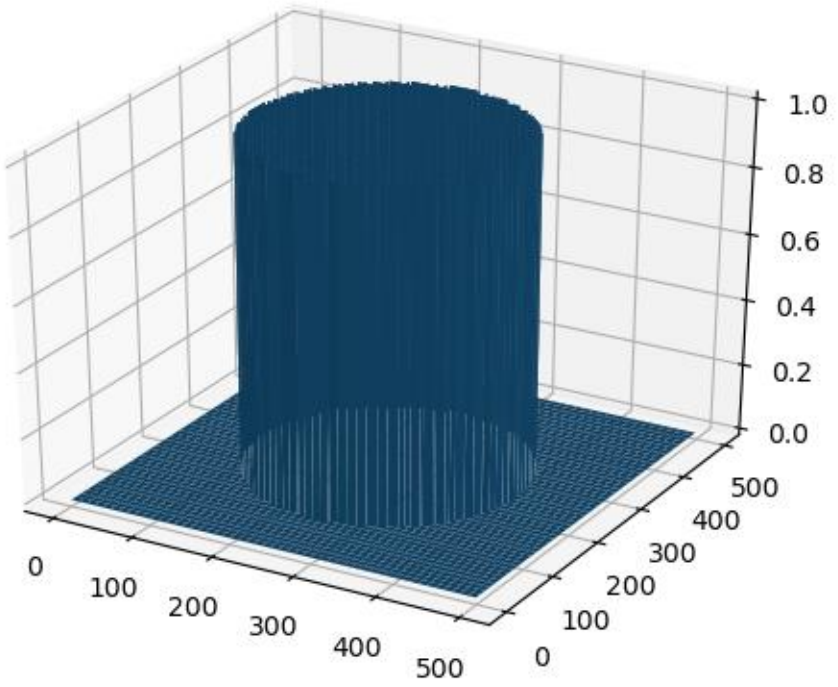
R= 30

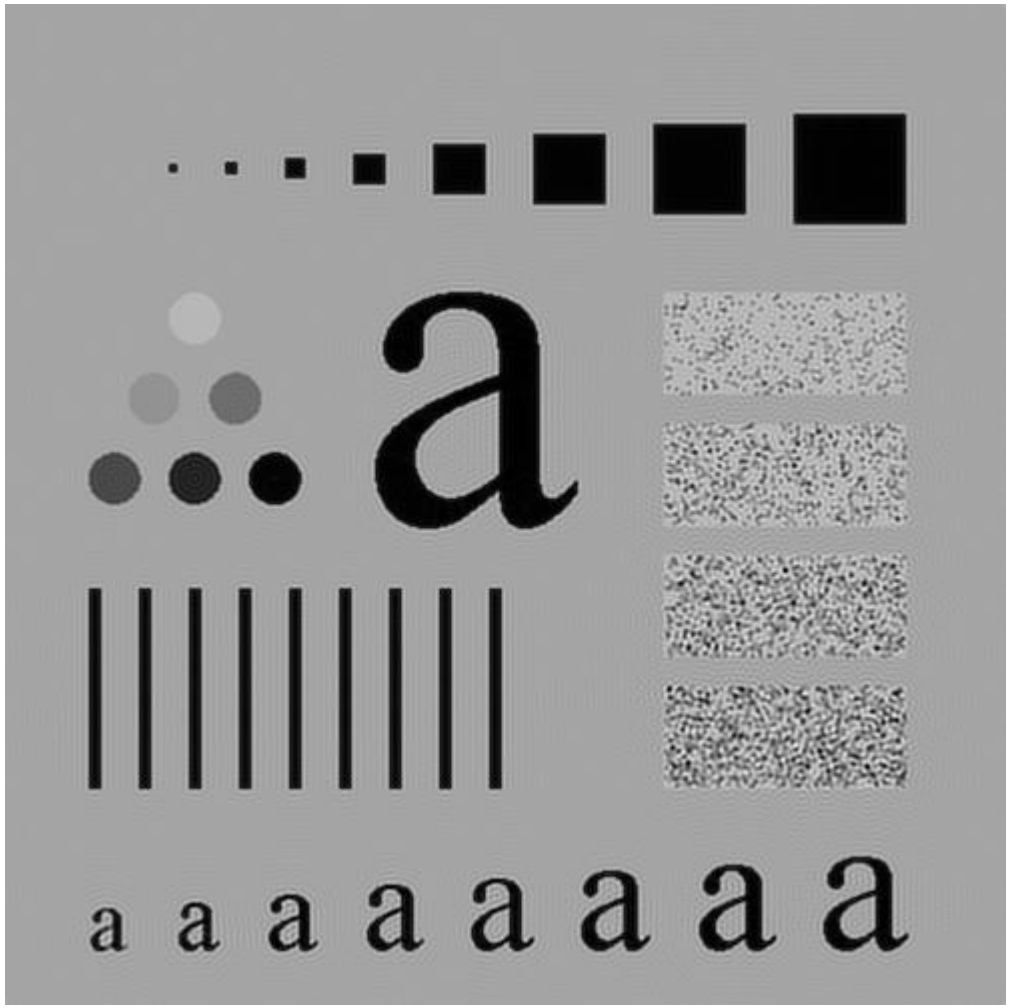
Ideal Low pass filter



R=60

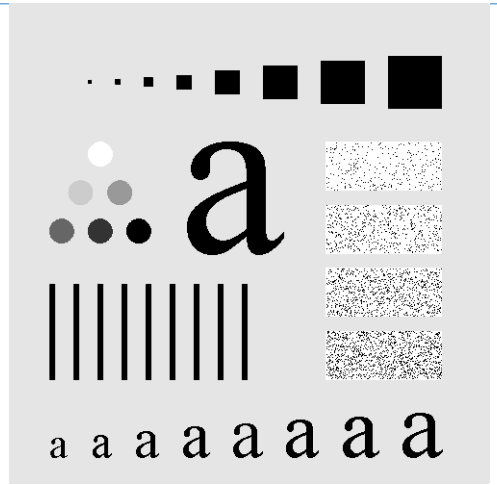
Ideal Low pass filter

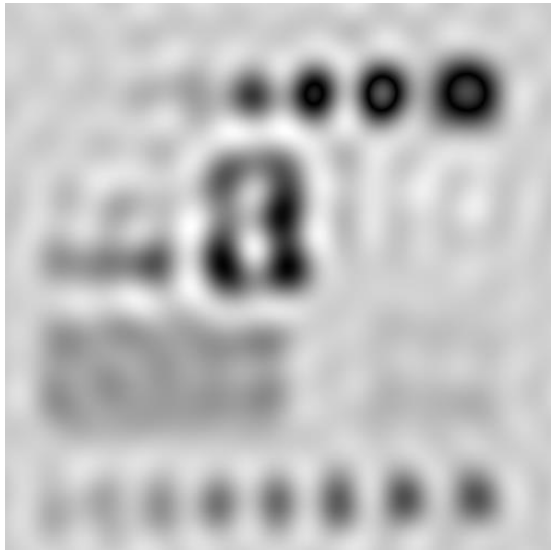


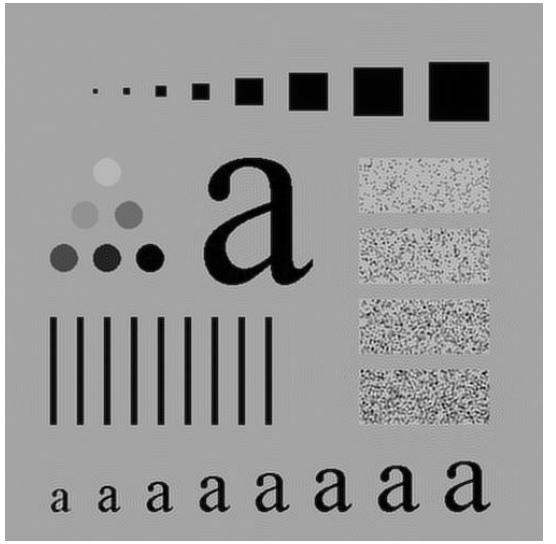


R=160

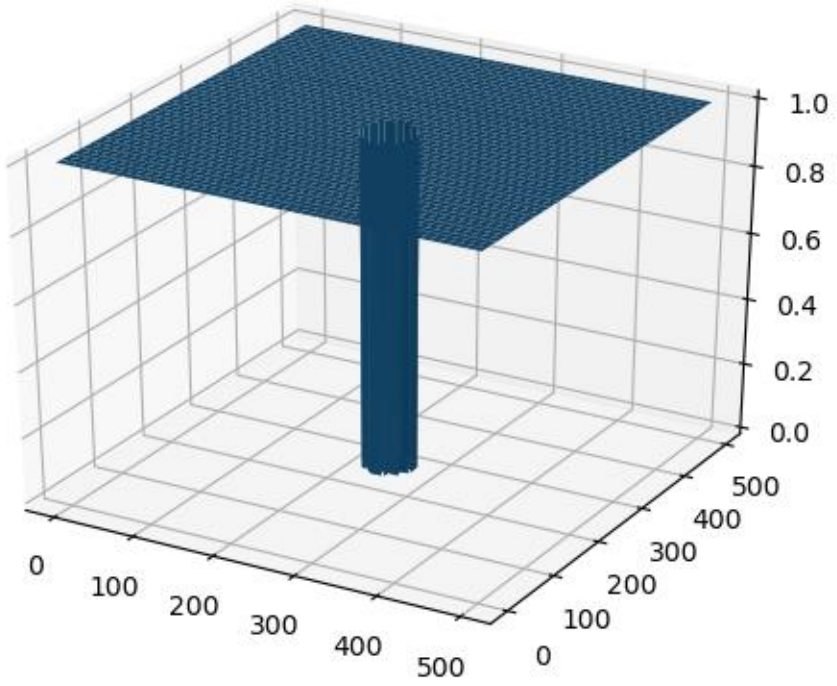
Ideal Low pass filter

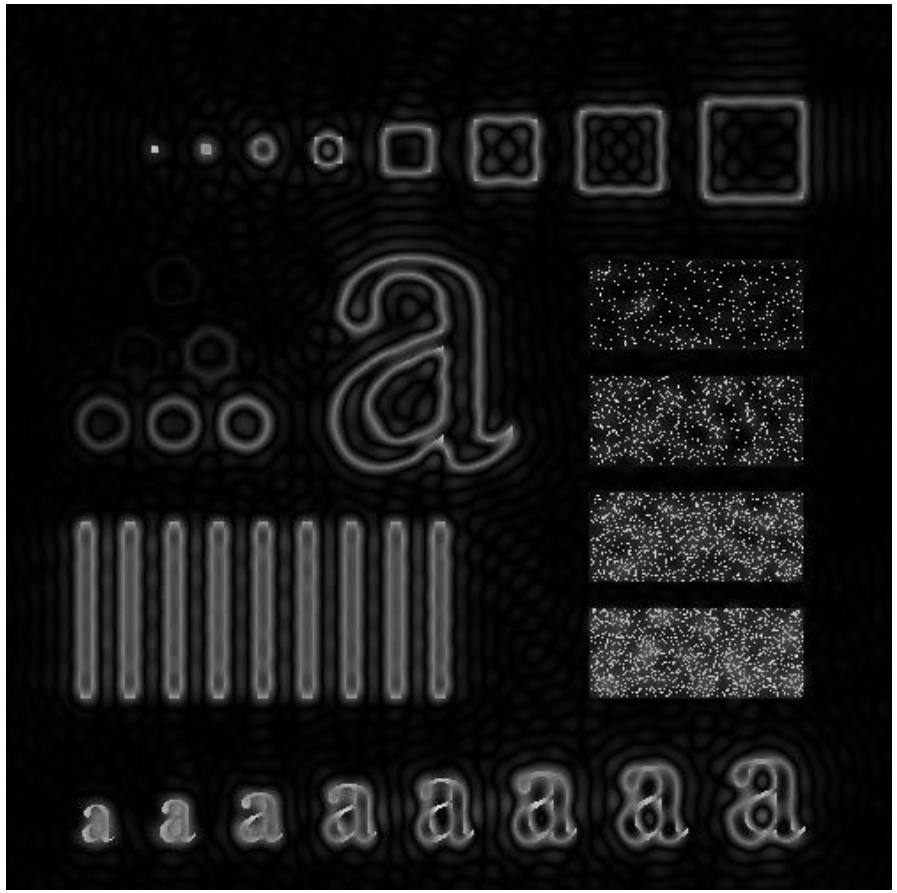






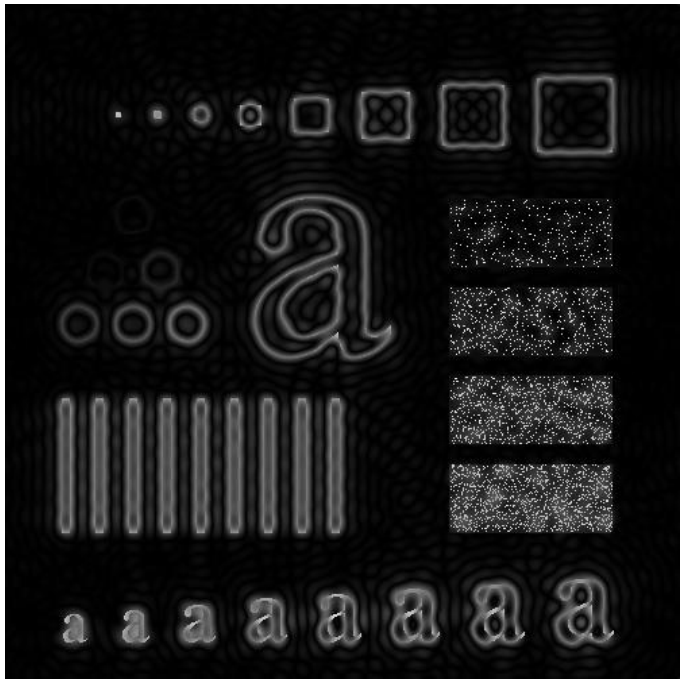
Ideal High pass filter



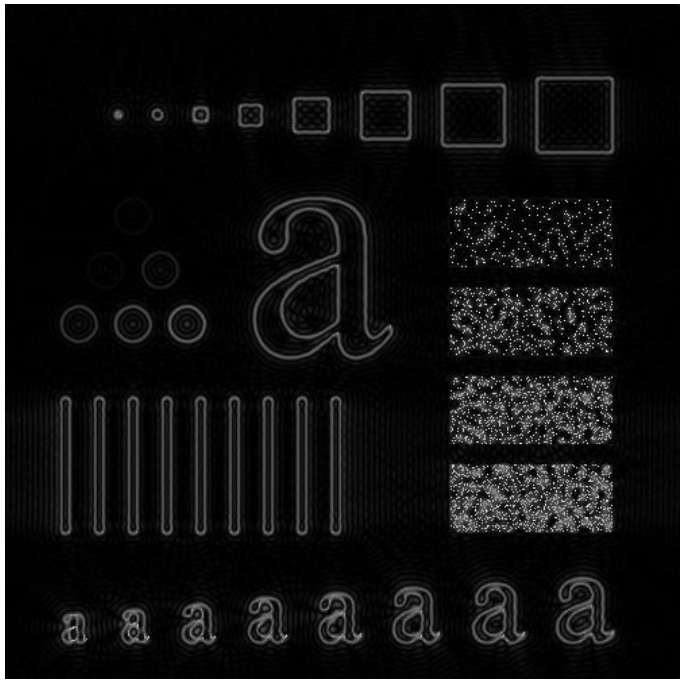


R=30

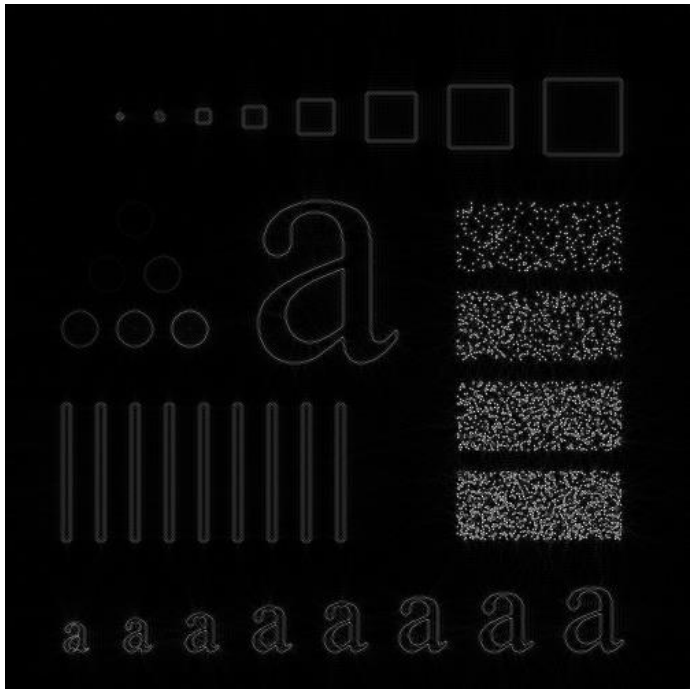
Ideal High pass filter



R=30

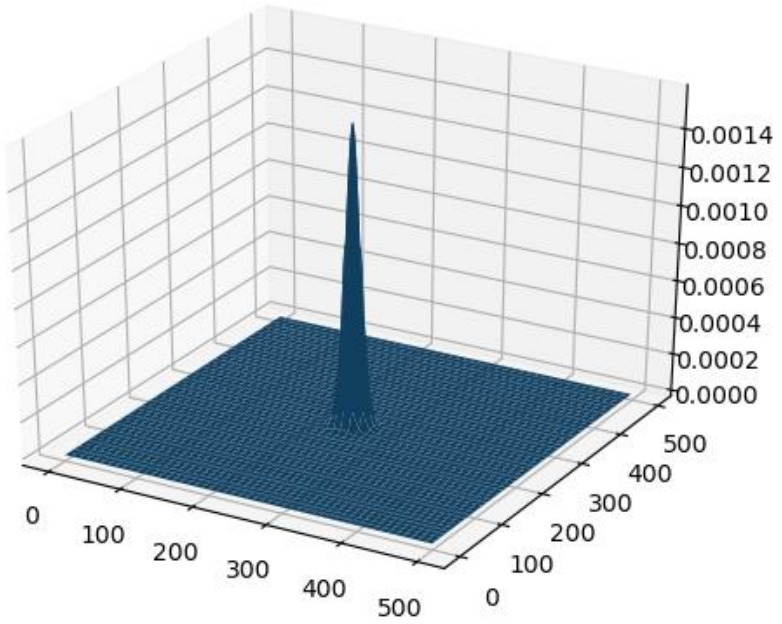


R=60



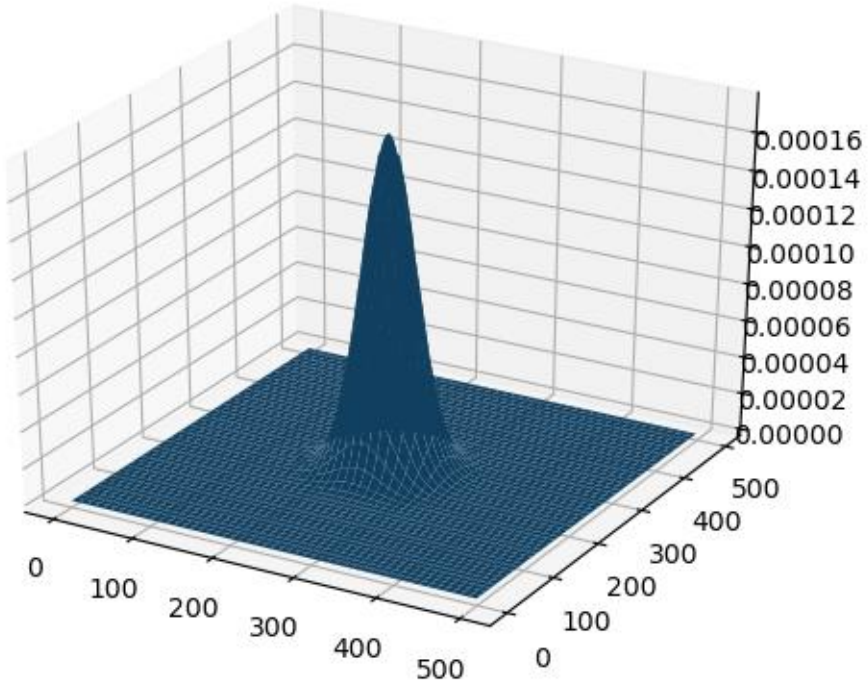
R=160

Gaussian Low pass filter



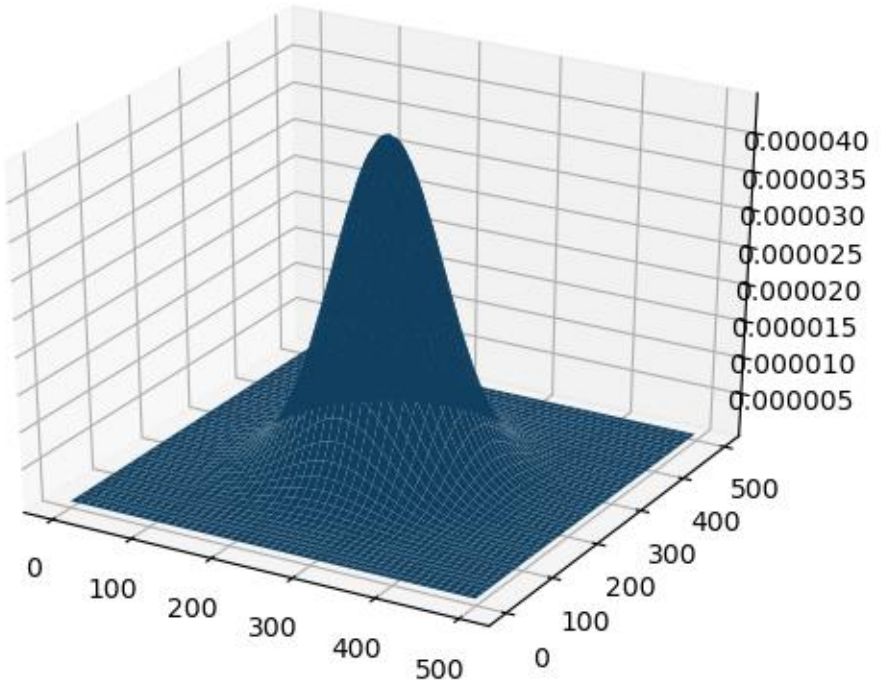
σ=10

Gaussian Low pass filter



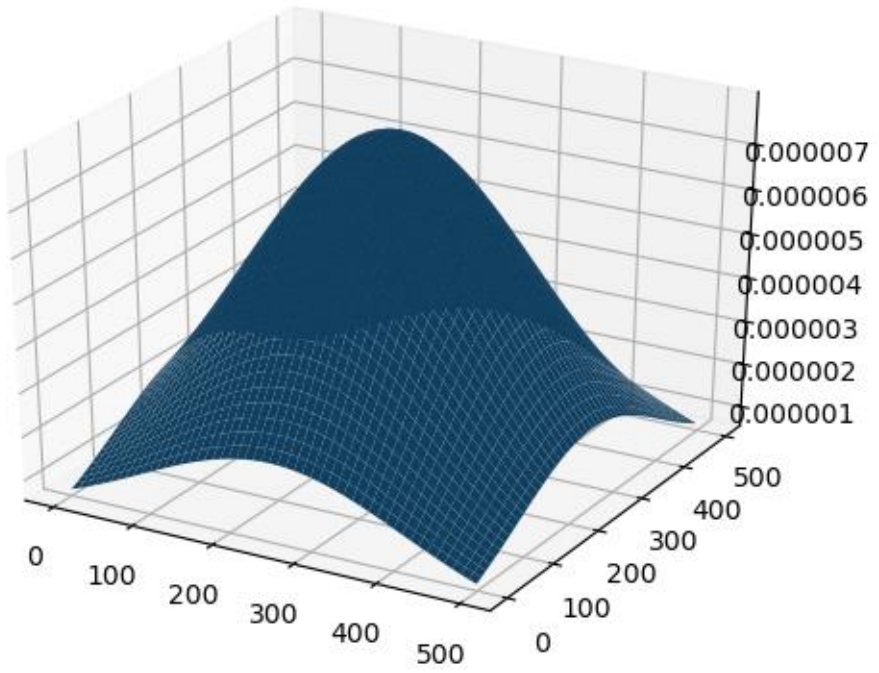
$\sigma=30$

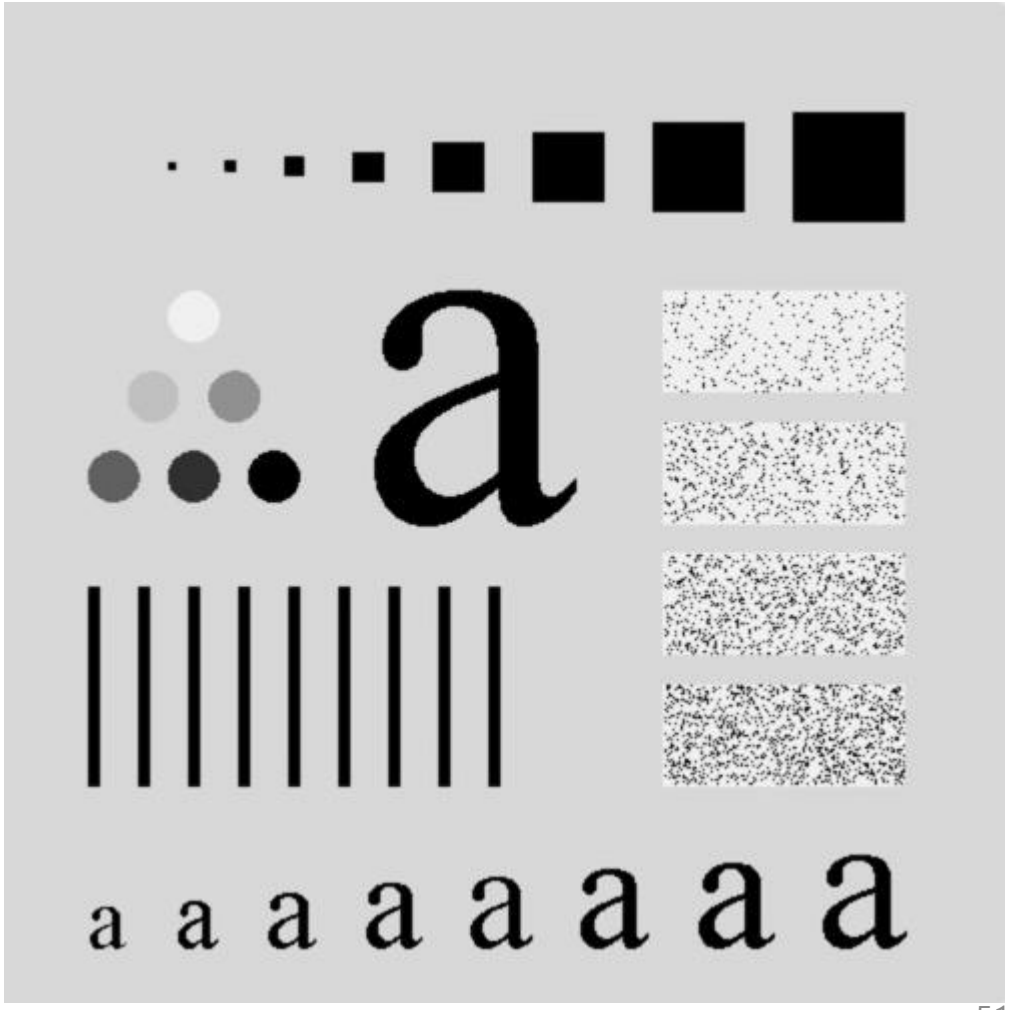
Gaussian Low pass filter



σ=60

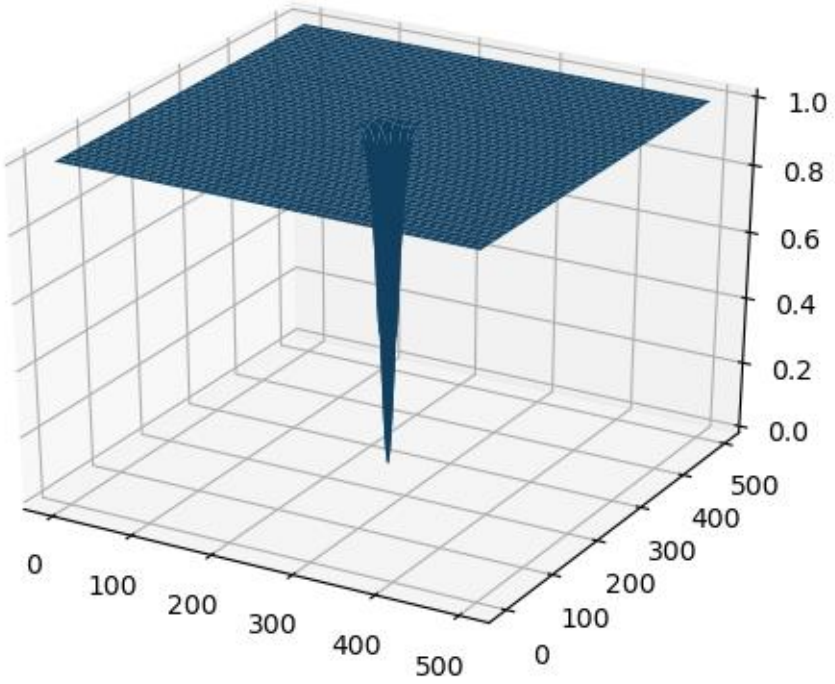
Gaussian Low pass filter

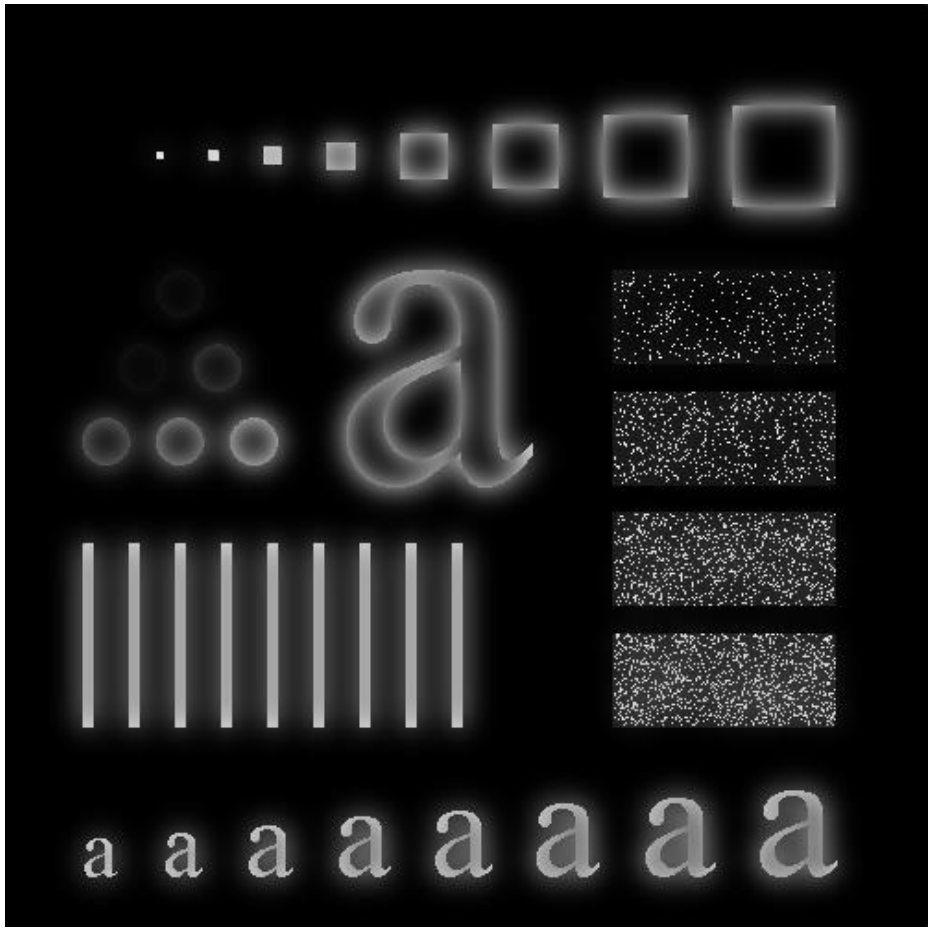




σ=160

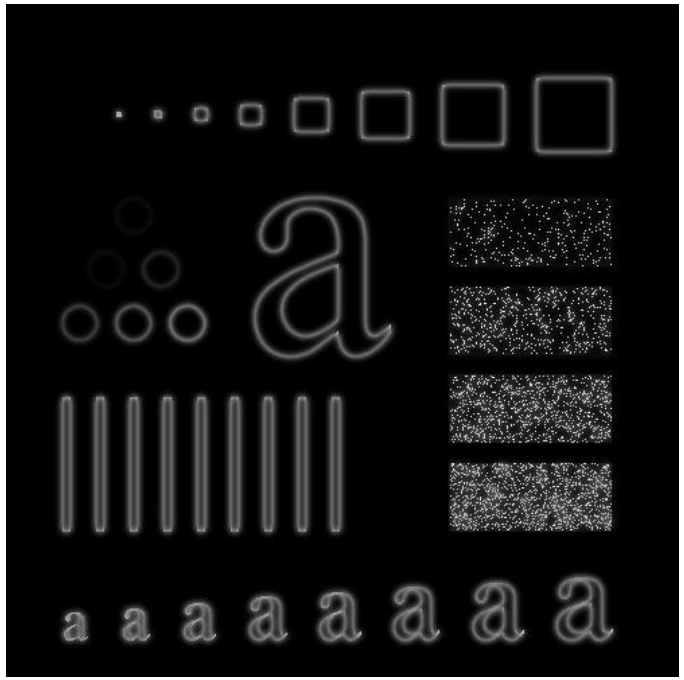
Gaussian High pass filter



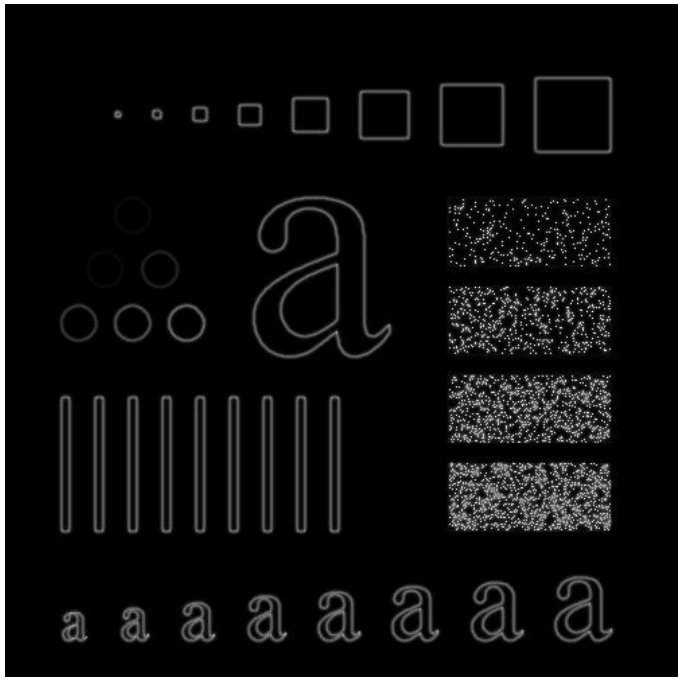


σ=10

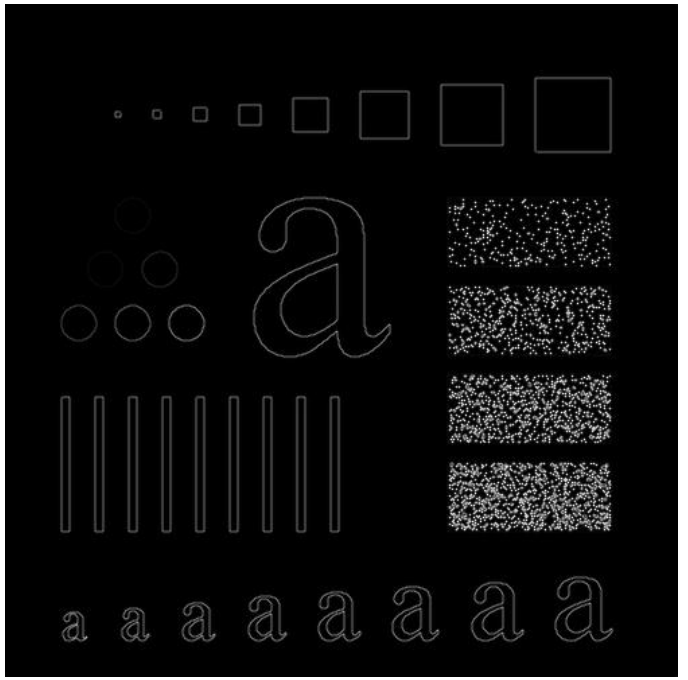
Gaussian High pass filter



σ=30



σ=60

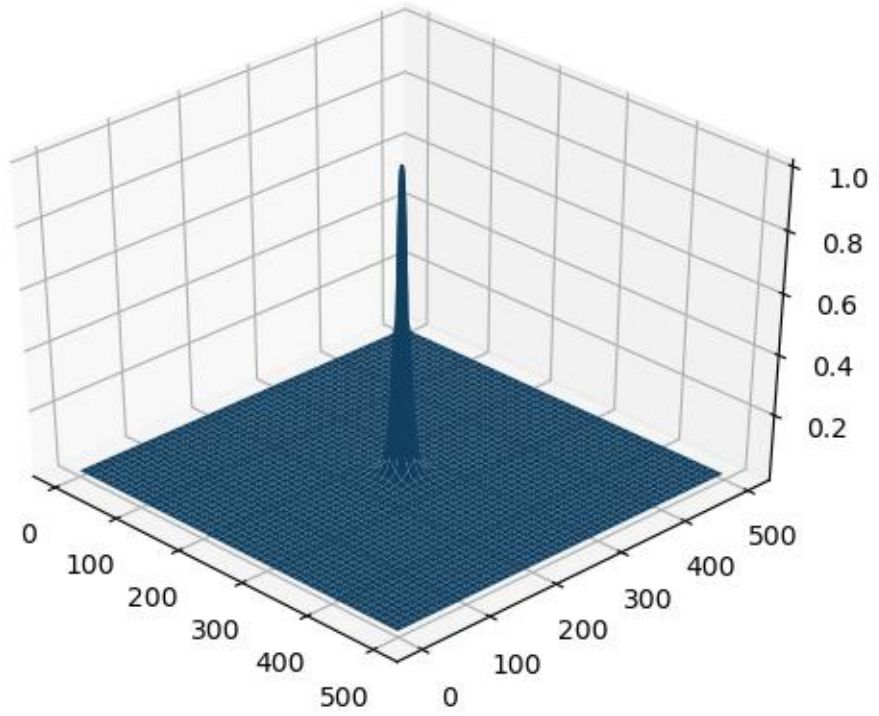


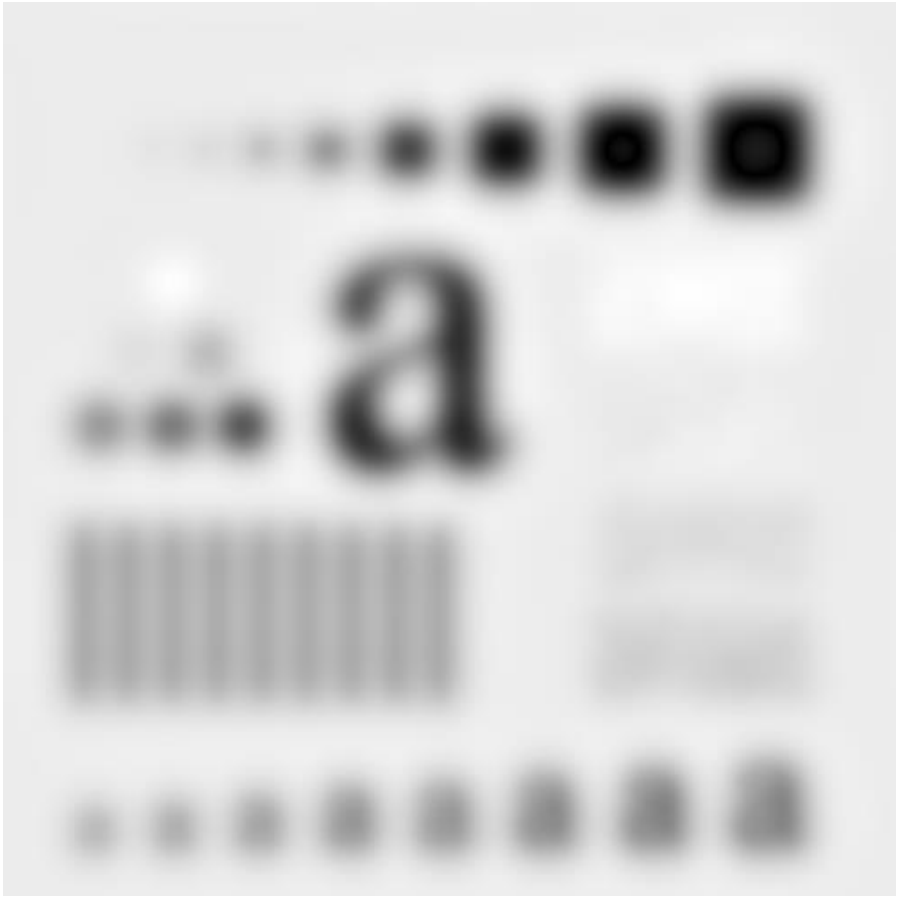
σ=160

Butterworth Lowpass filter

$$H(u,v) = \frac{1}{1+[\frac{R}{D(u,v)}]^{2n}}$$

R = radious
N= order
D(u,v)= coordinate distance



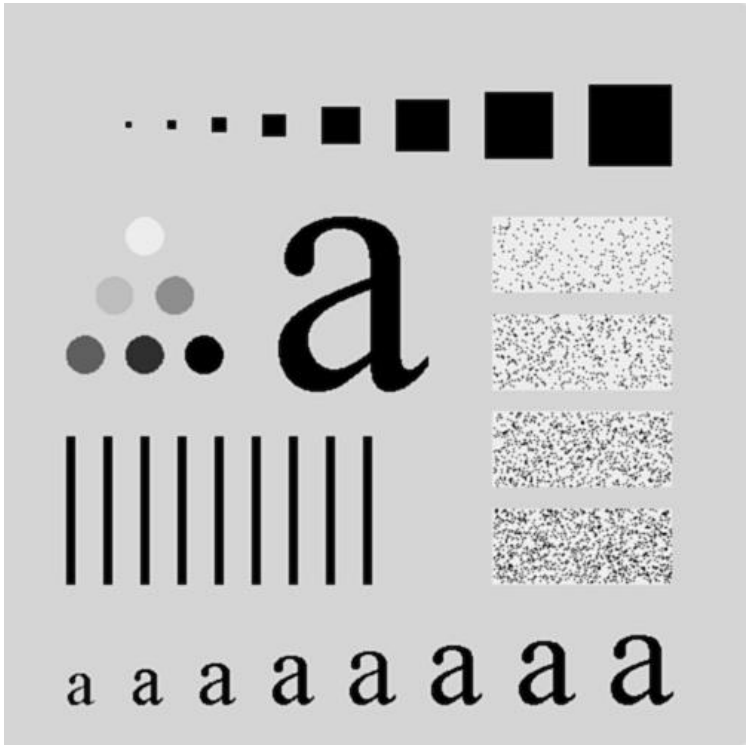


R=10

Butterworth Lowpass filter

R=30

R=60

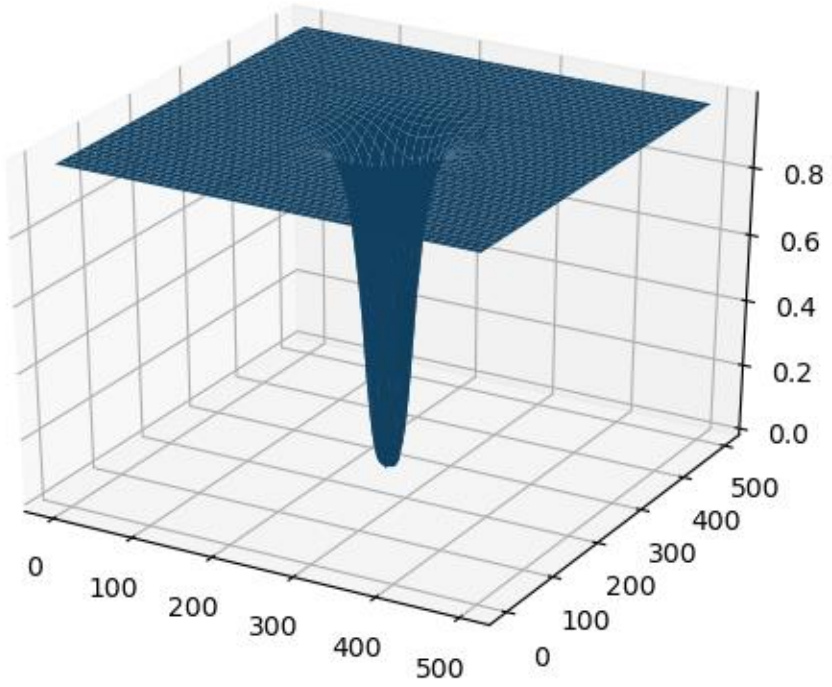


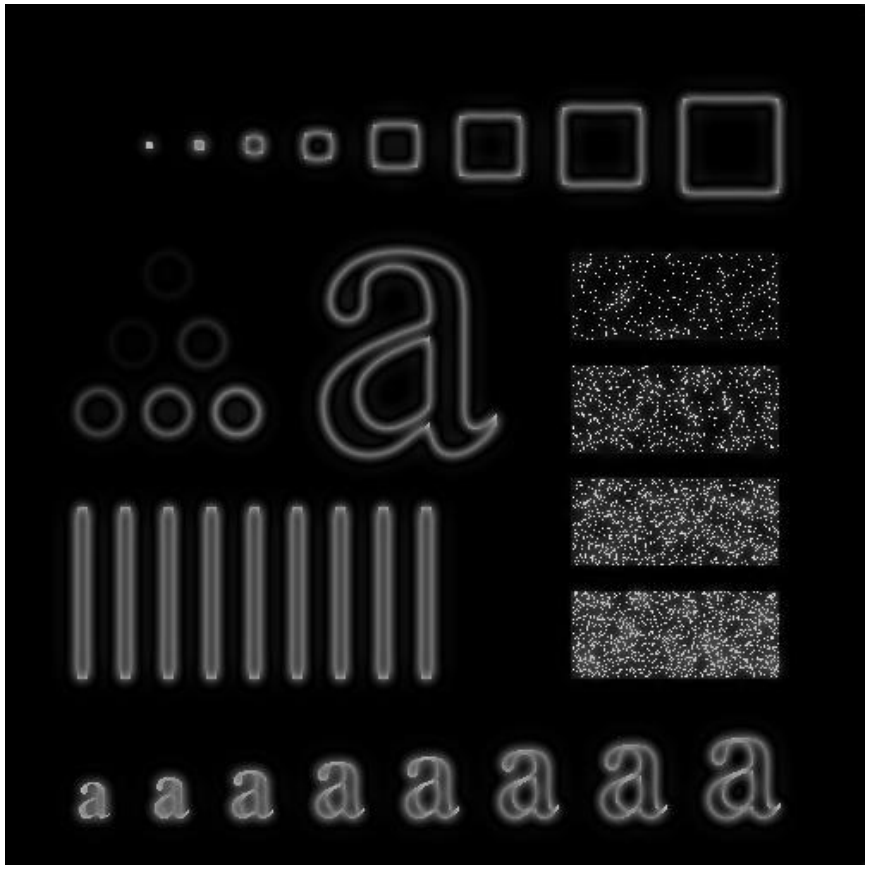
R=160

Butterworth Highpass filter

$$H(u,v) = \frac{1}{1+[\frac{D(u,v)}{R}]^{2n}}$$

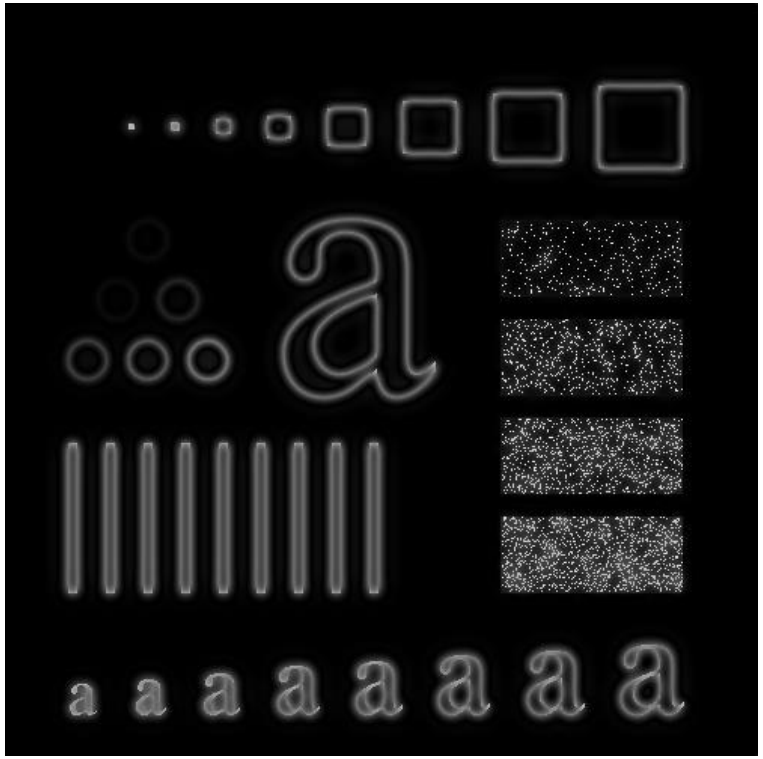
R = radious
N= order
D(u,v)= coordinate distance



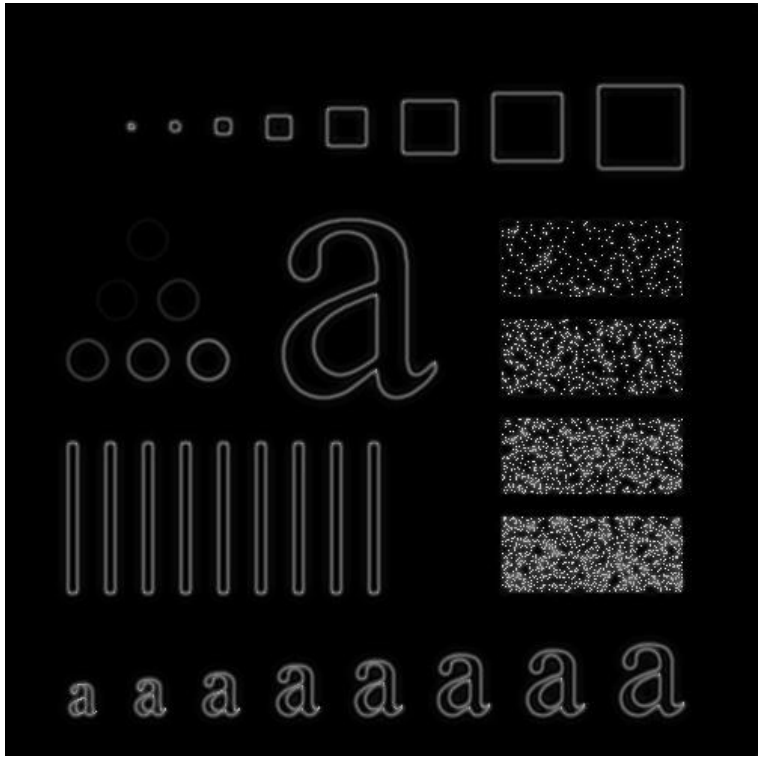


$R=30$

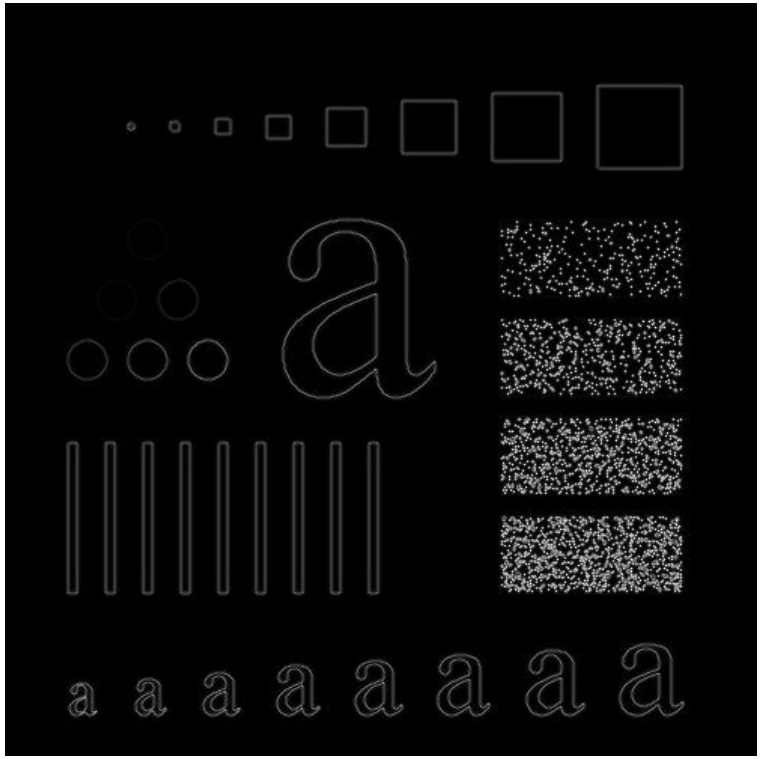
Butterworth Highpass filter



$R=30$



$R=60$

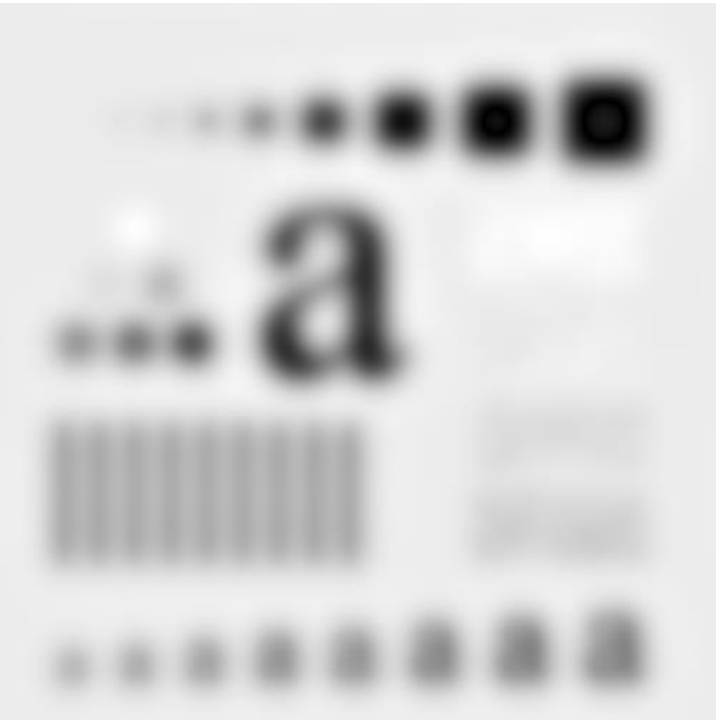


$R=160$

Comparison (Low pass filter)

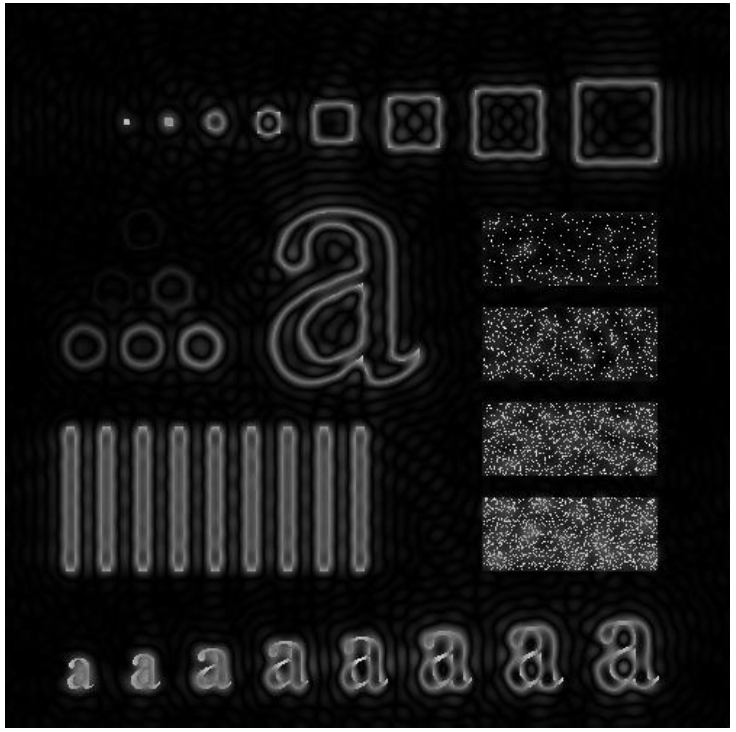
$ILPF\ R$=10

$GLPF\ \sigma$=10

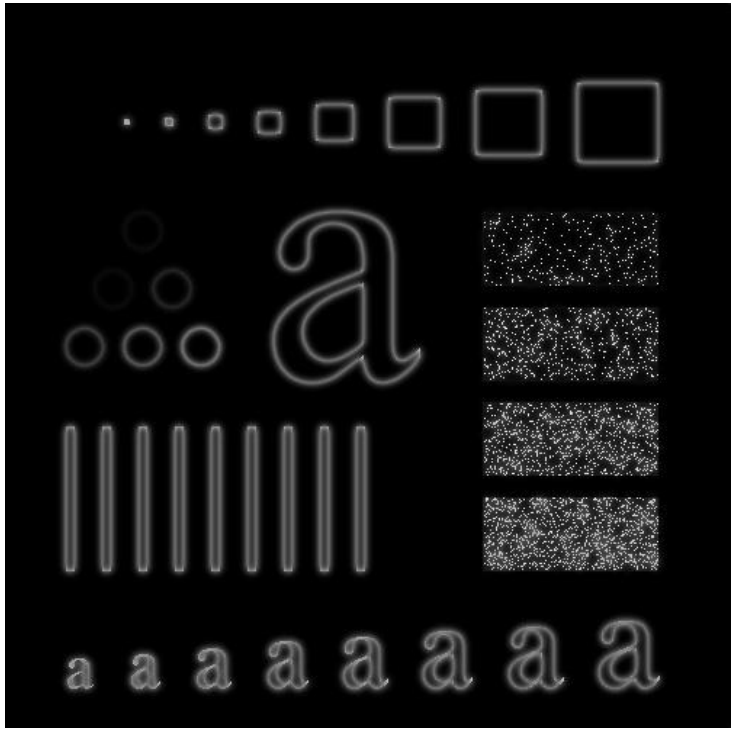


$BLPF\ R$=10

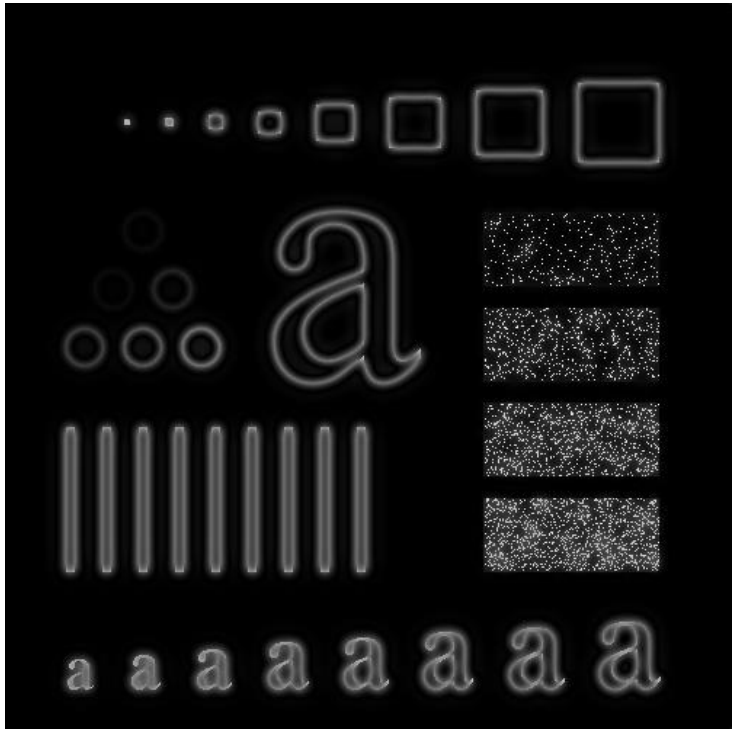
Comparison (High pass filter)



$IHPF$ R=30



GHPF σ=30



BHPF R=30

Notch filter

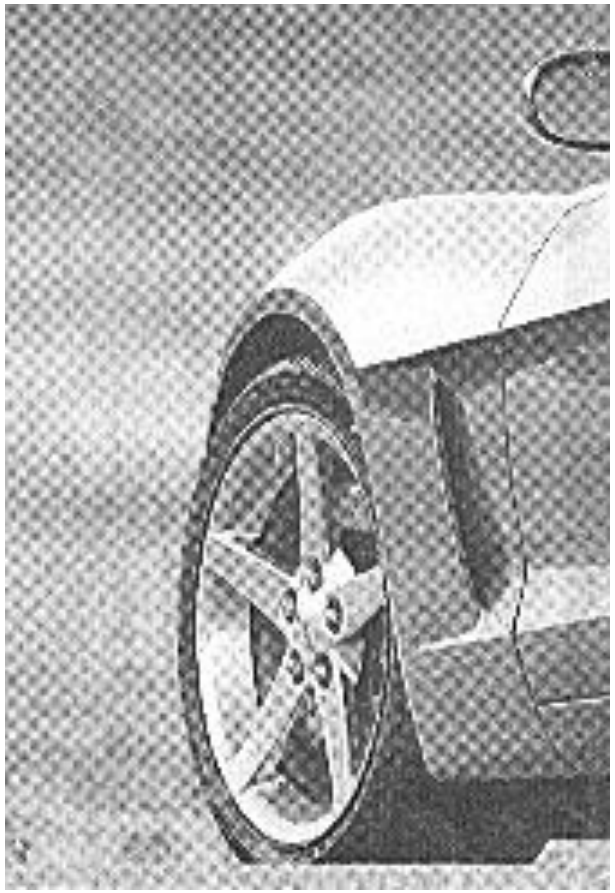
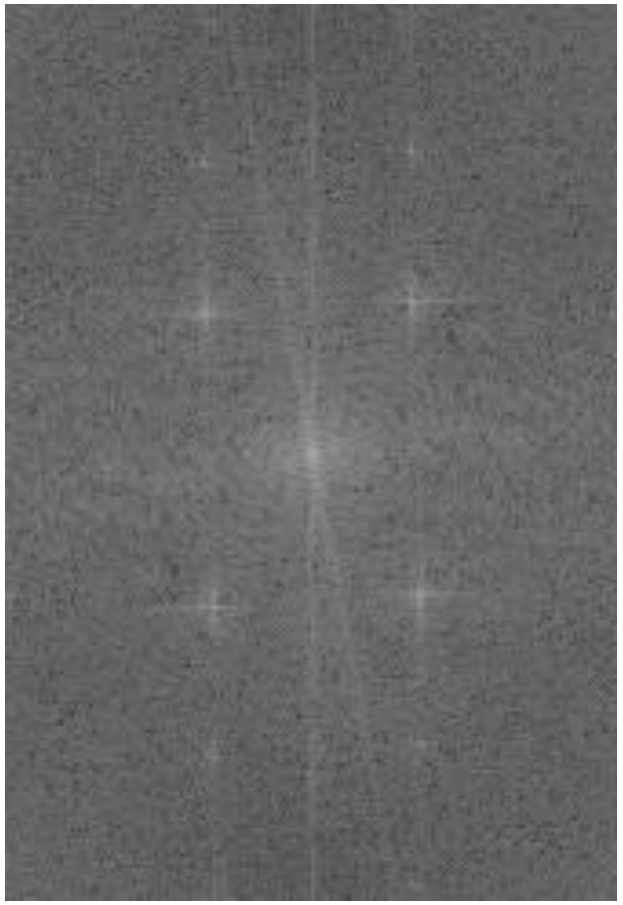
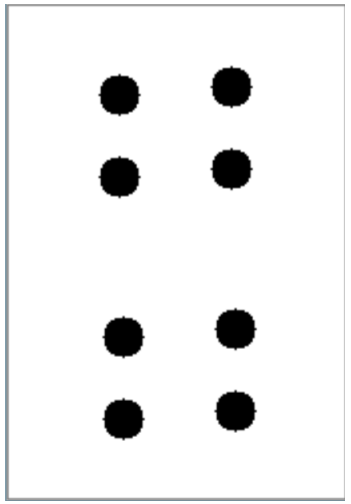


Image with noise

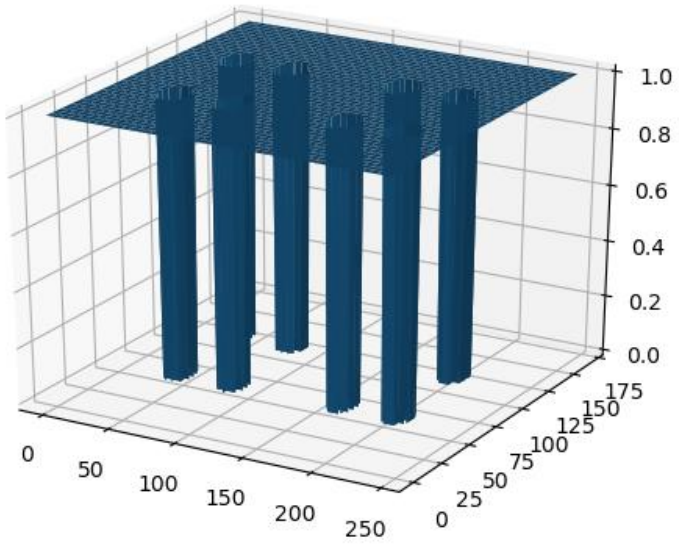


DFT



Notch Filter
(R=8)

Notch filter



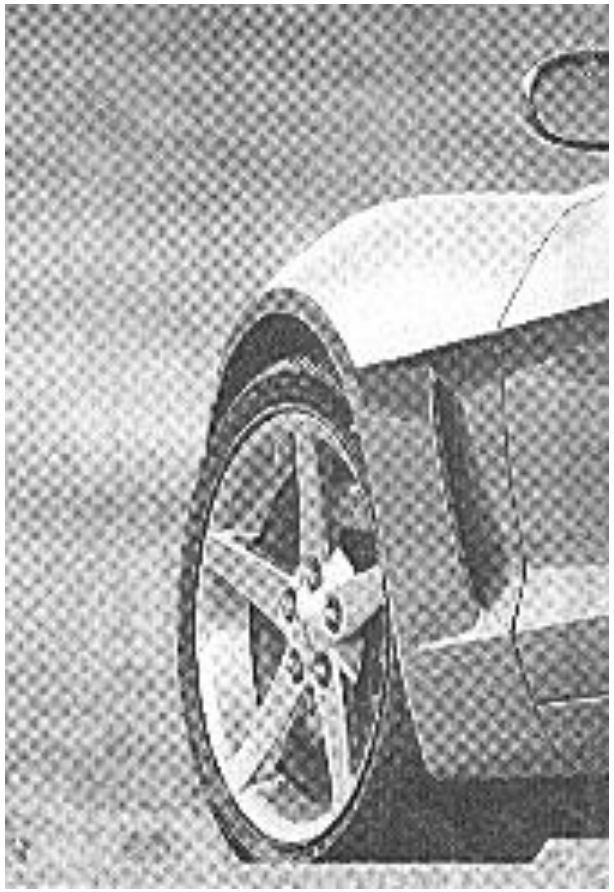
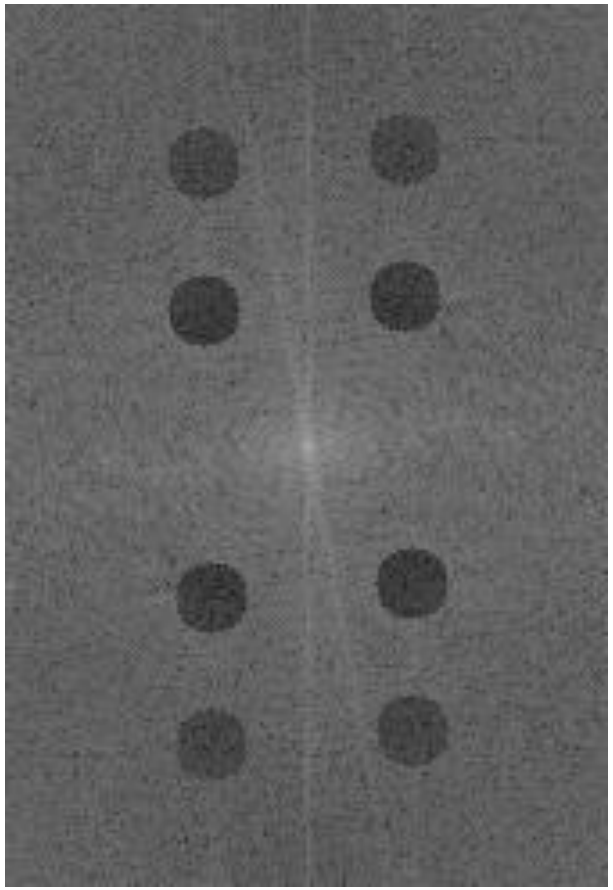


Image with noise

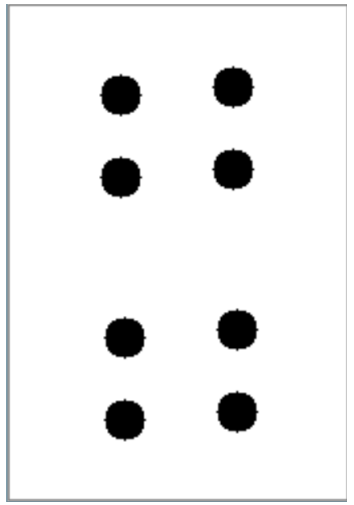
After Notch filtering

Notch filter

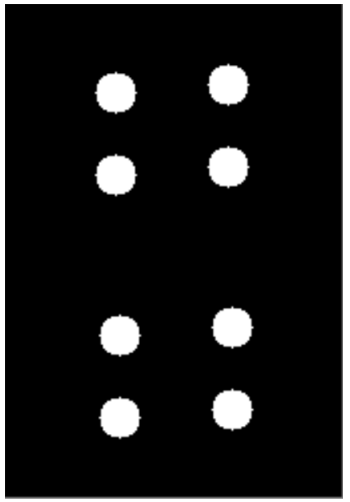
After notch filtering



DFT image



Notch filter



Reverse filter

Notch filter

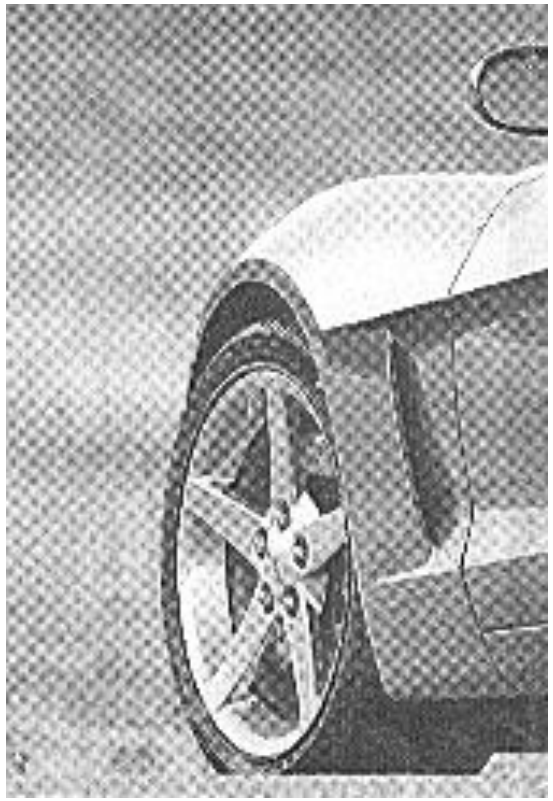
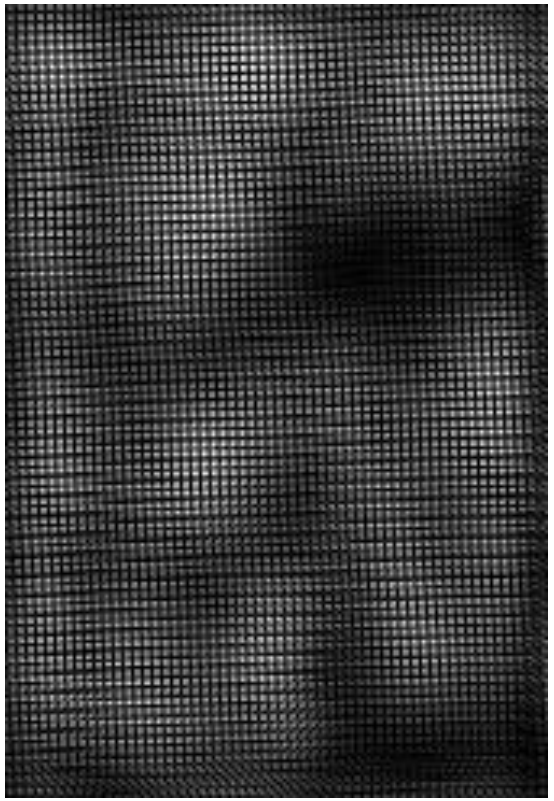


Image with noise



Noise component
(R=8)

Noise component
(R=15)

Restoration

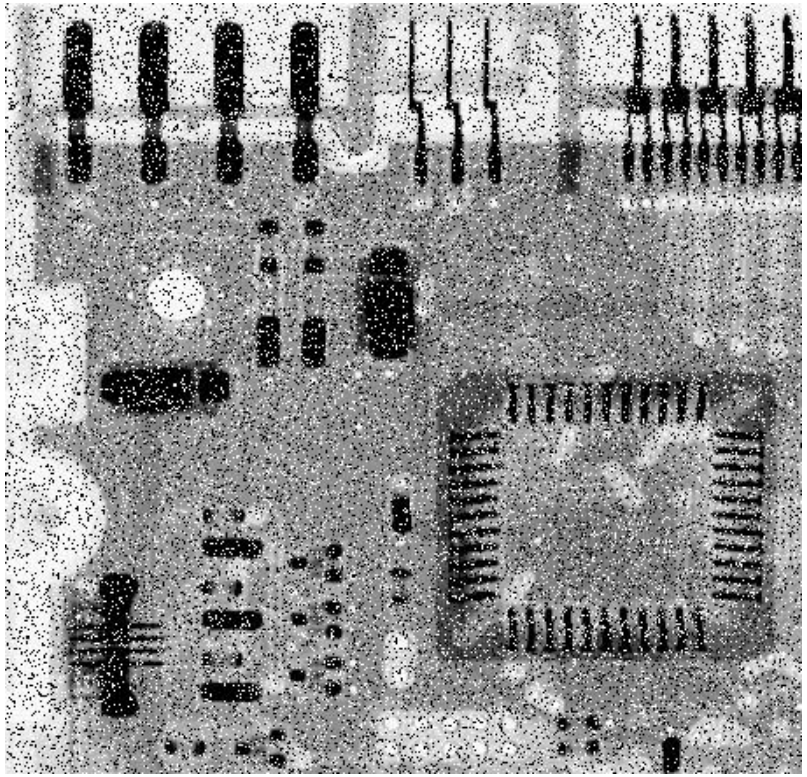
1.Spatial Domain Restoration

- Median filter
- Min – max filter
- Midpoint filter
- Alpha – Trimmed filter
- Adaptive median filter

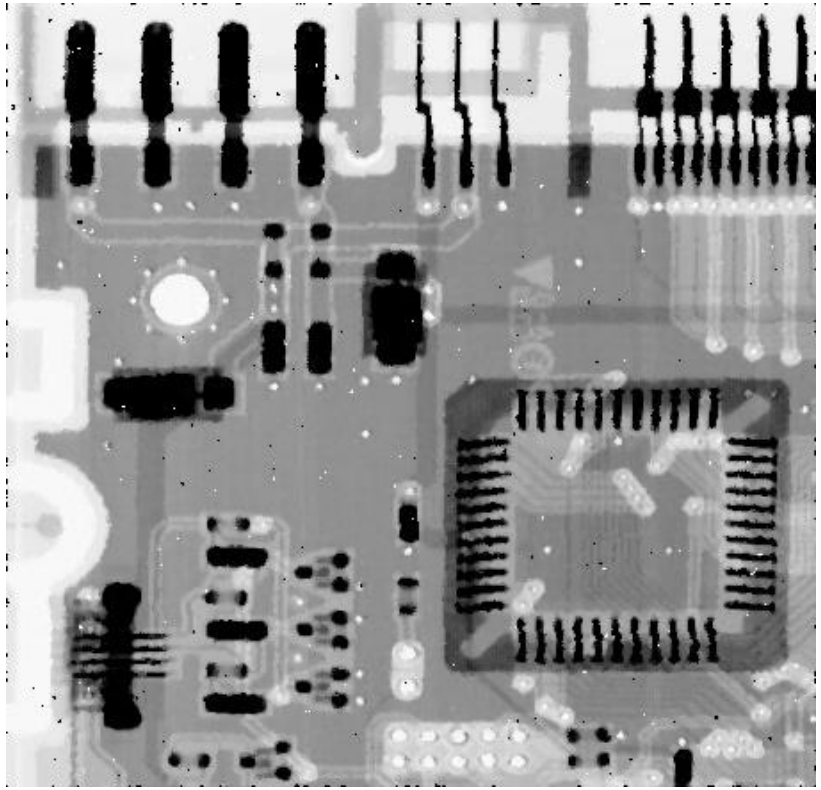
2.Frequency Domain Restoration

- Inverse Filtering
- Wiener Filtering

Spatial Domain – Median filter



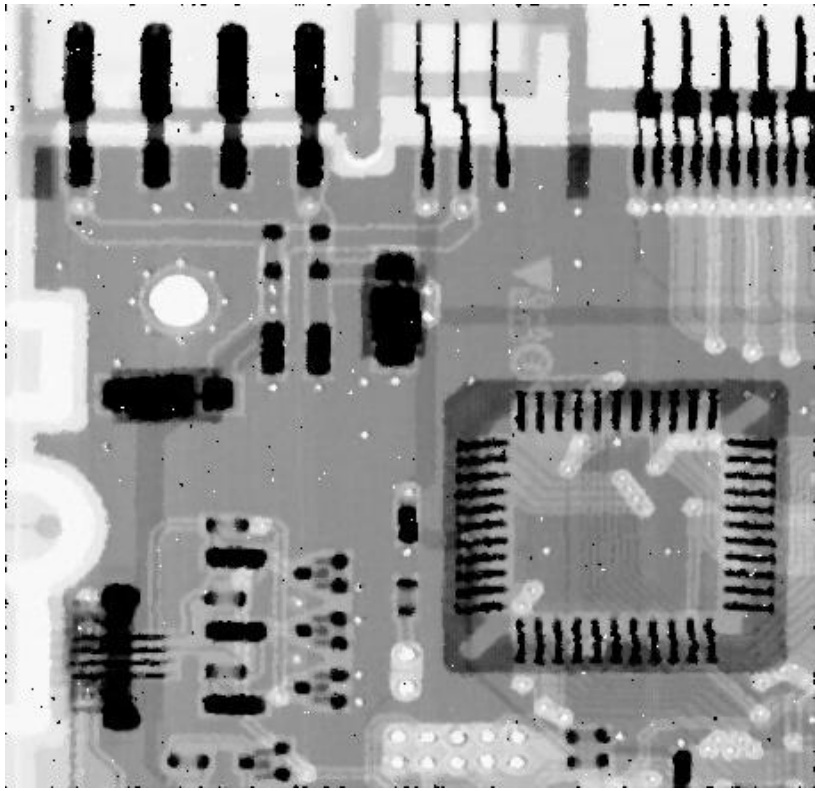
Salt and pepper noise



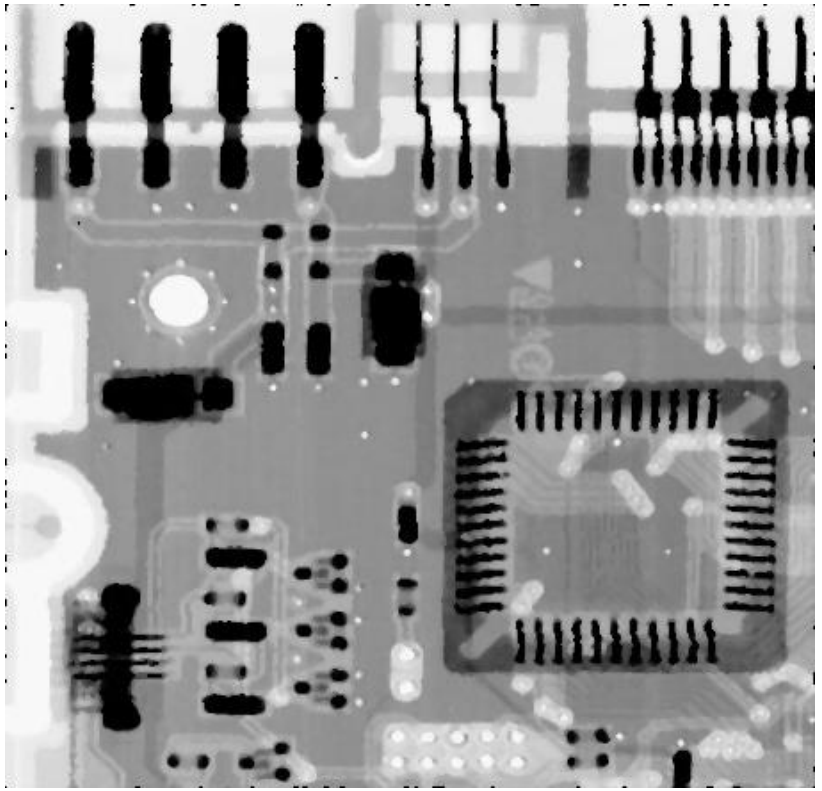
3 x 3 median filter

$$f(x,y) = median\{g(s,t)\}$$

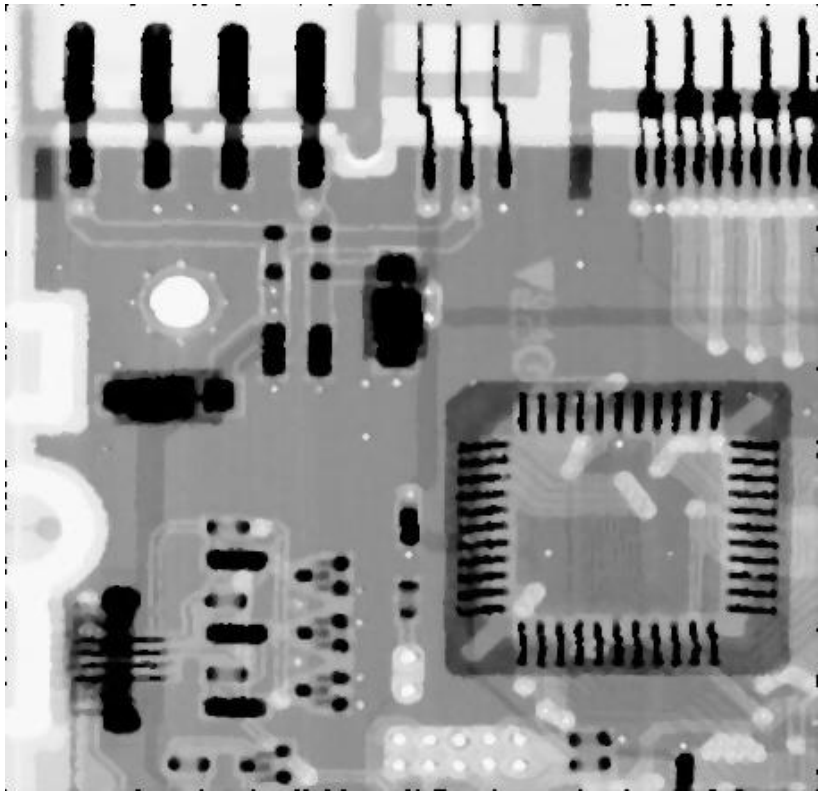
Spatial Domain – Median filter



3 x 3 median filter

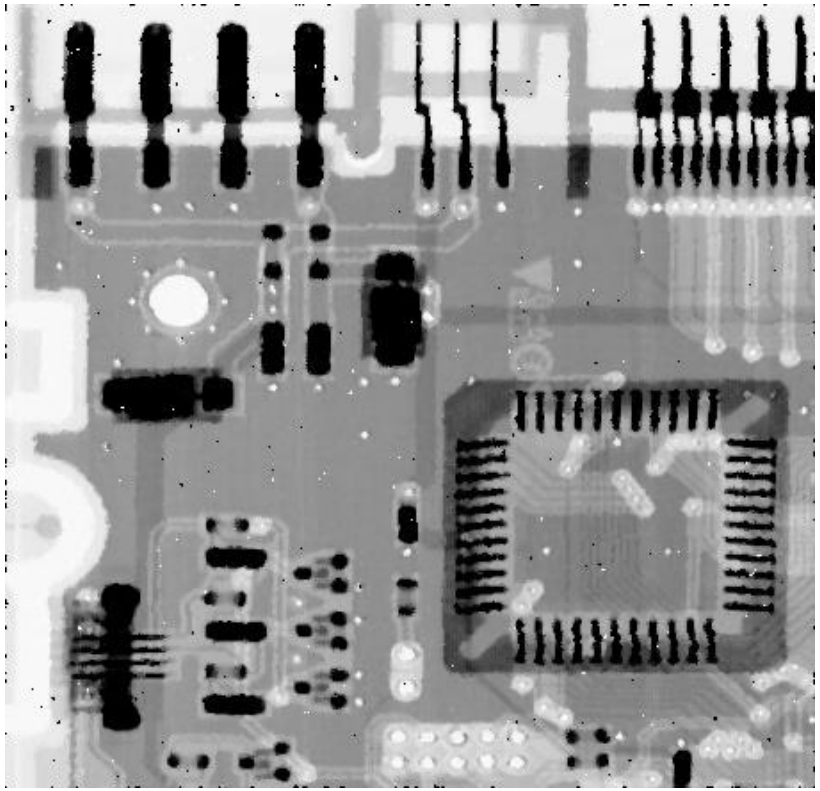


3 x 3 median filter X2

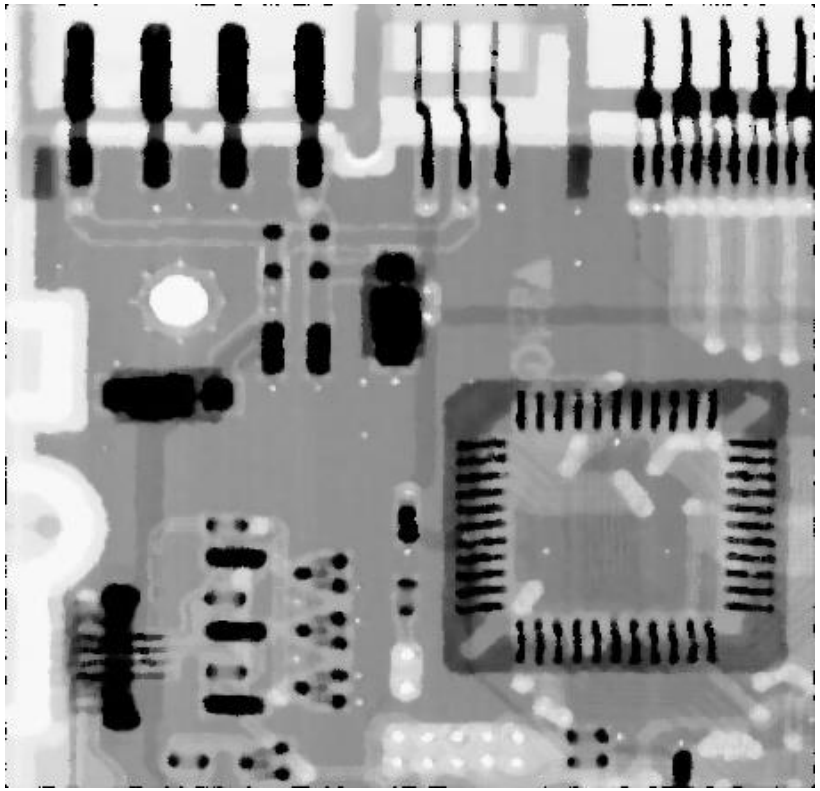


3 x 3 median filter X3

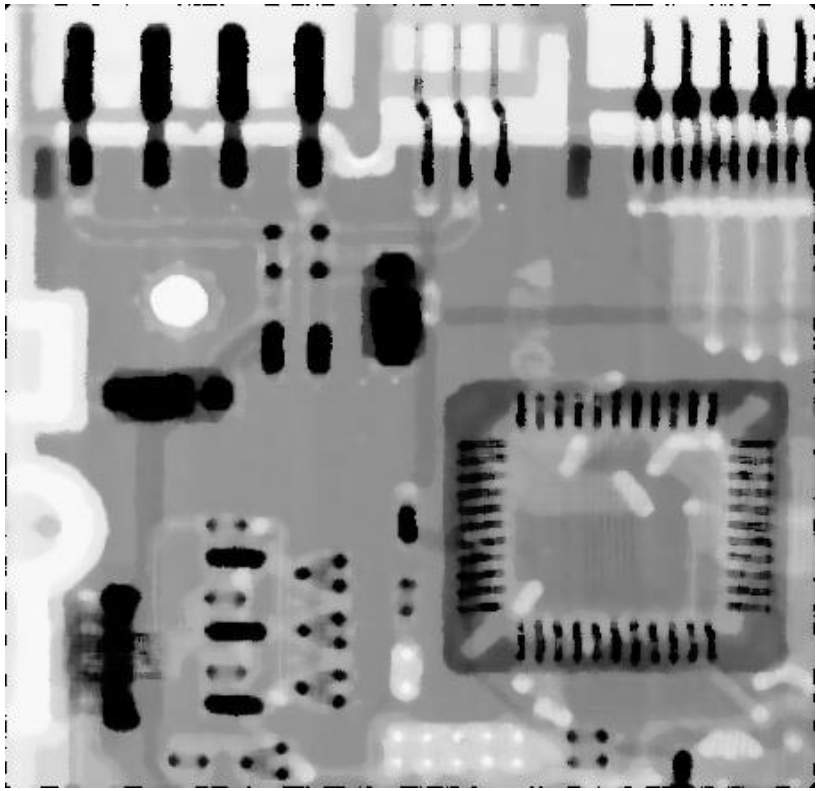
Spatial Domain – Median filter



3 x 3 median filter

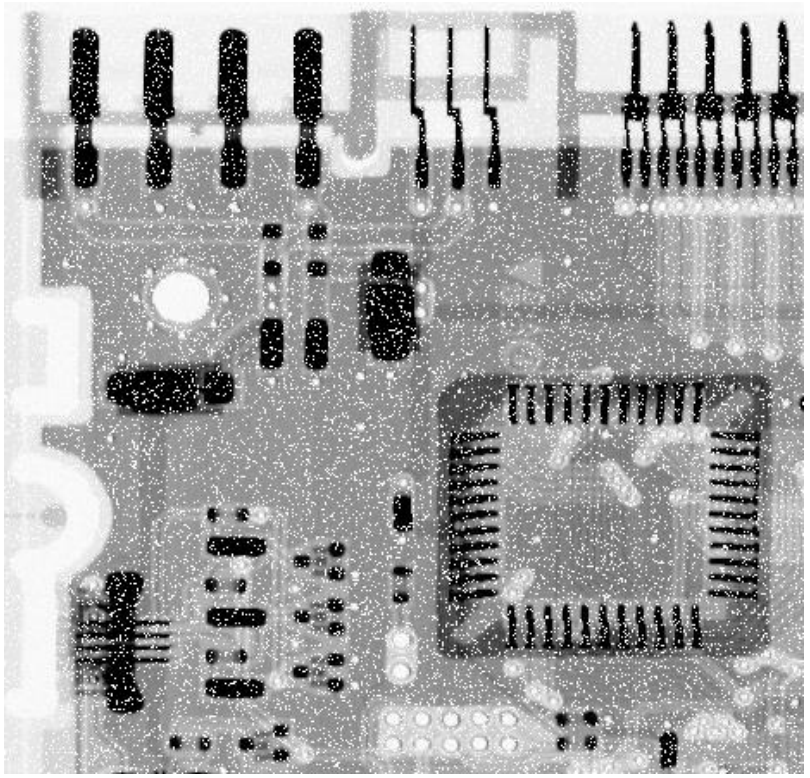


5 x 5 median filter

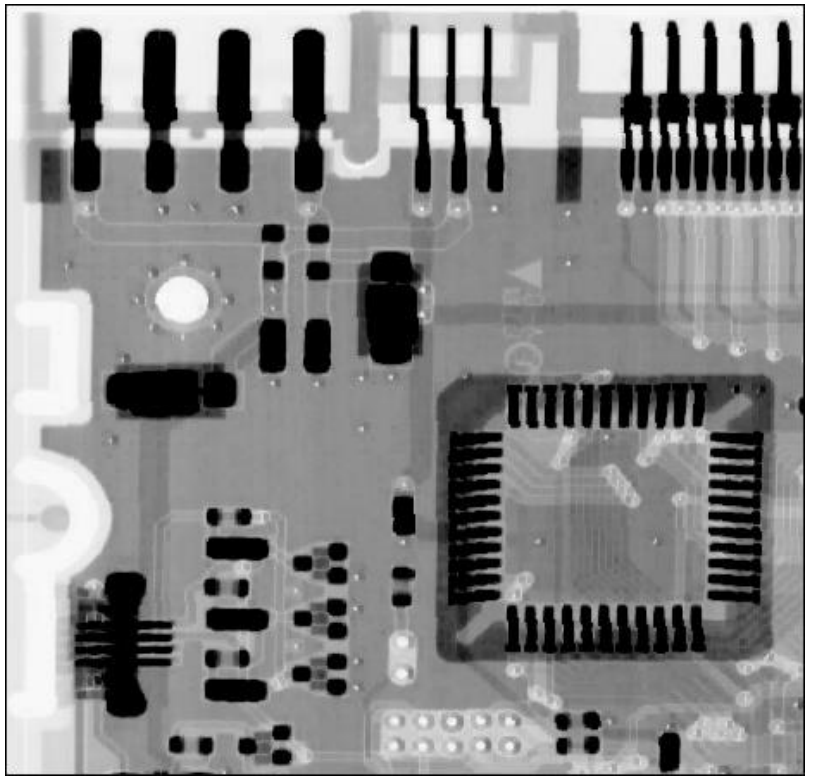


7 x 7 median filter

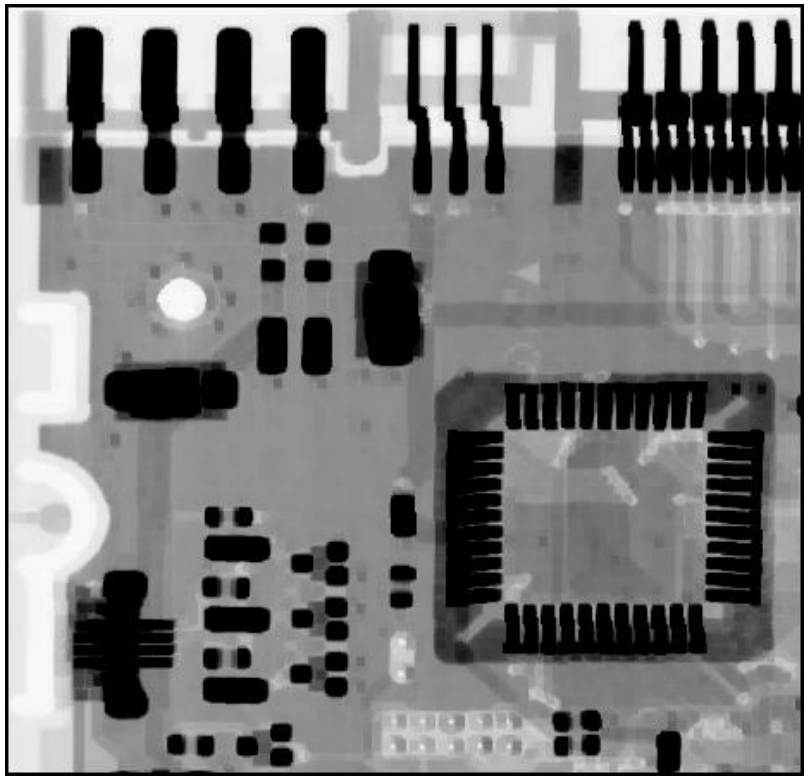
Spatial Domain – Min filter



Salt noise image



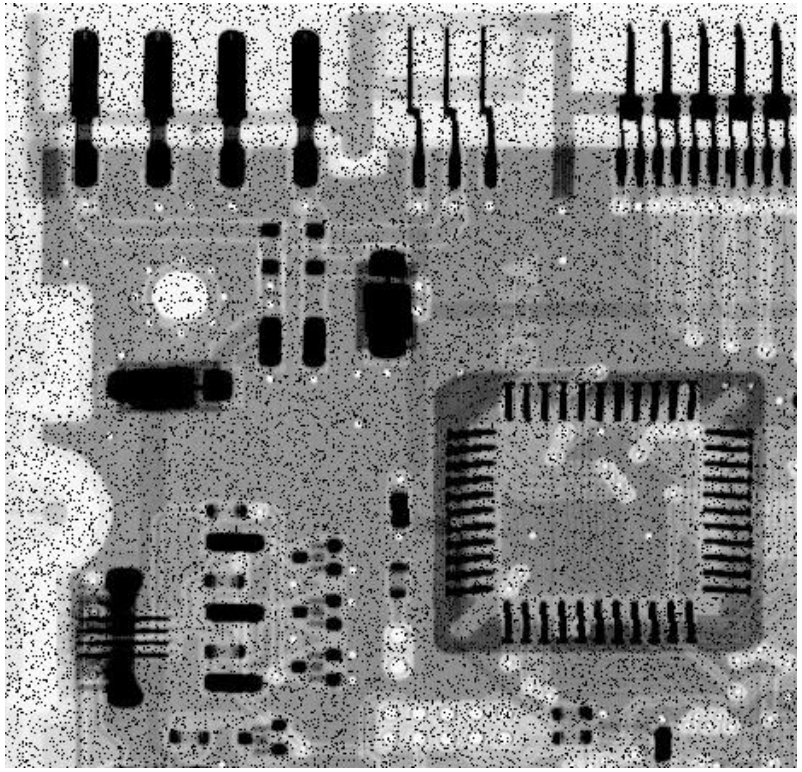
3 x 3 min filter



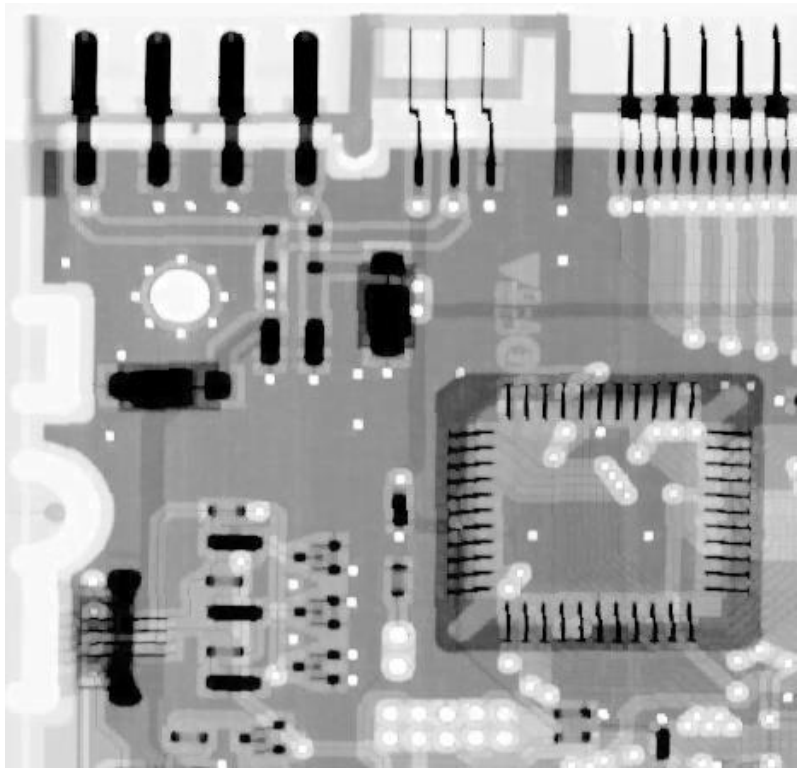
5 x 5 min filter

$$f(x,y) = min\{g(s,t)\}$$

Spatial Domain – Max filter

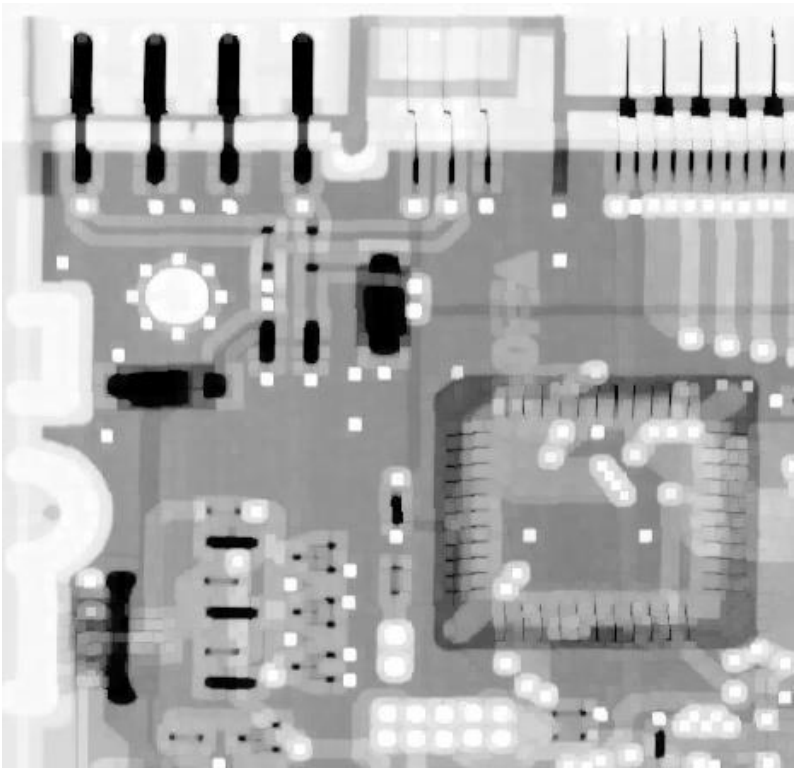


Pepper noise image



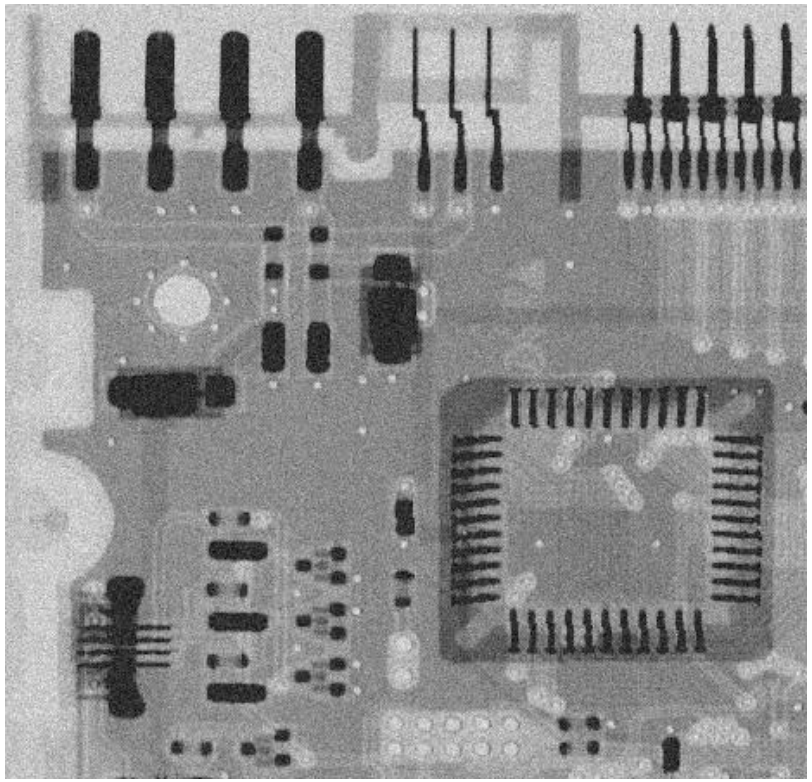
3 x 3 max filter

$$f(x,y) = max\{g(s,t)\}$$

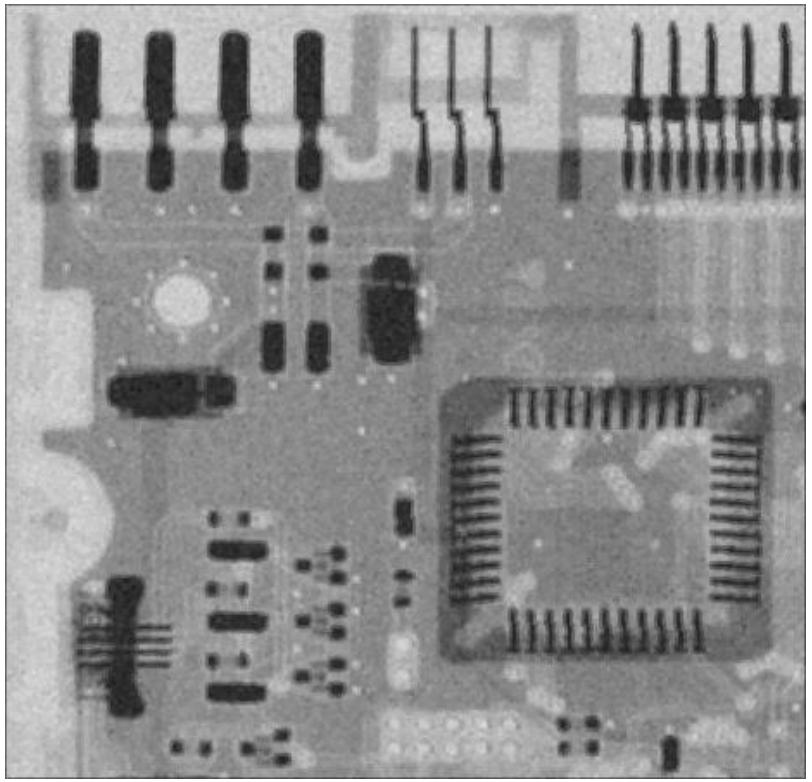


5 x 5 max filter

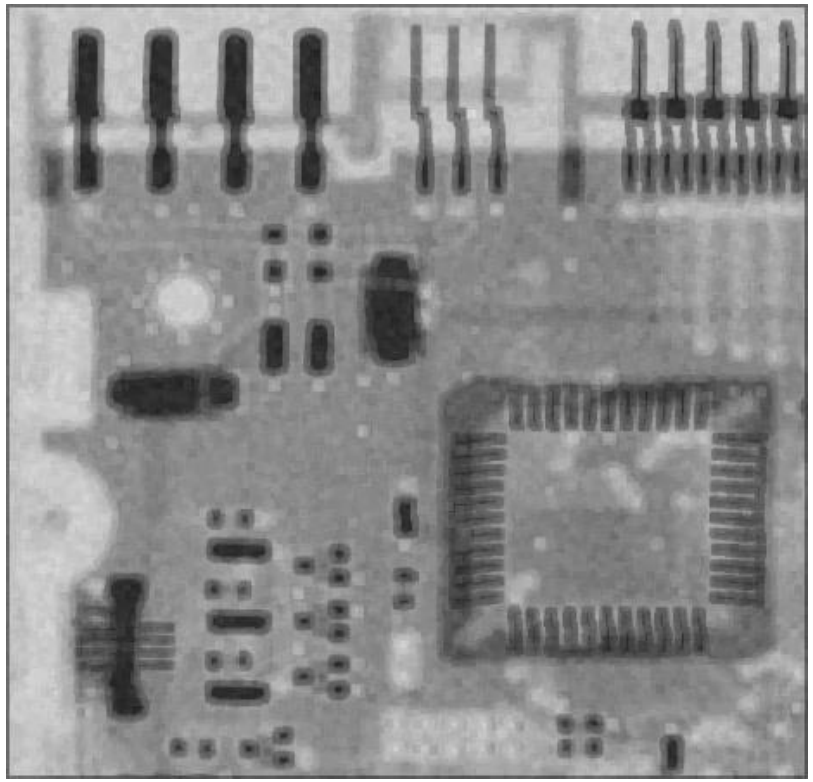
Spatial Domain – Mid point filter (Gaussian noise)



Gaussian noise image



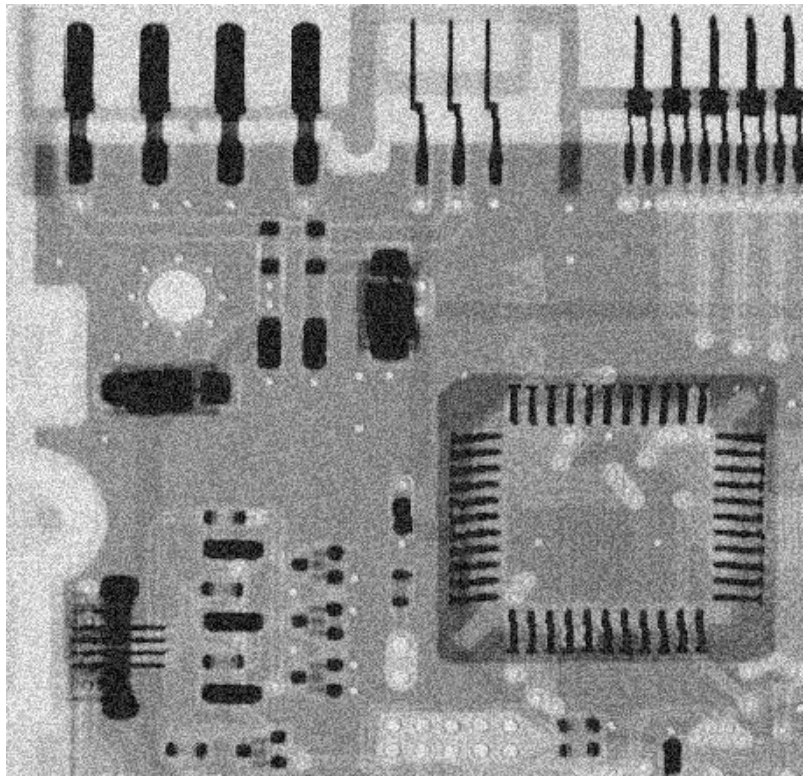
3 X 3 mid point filter



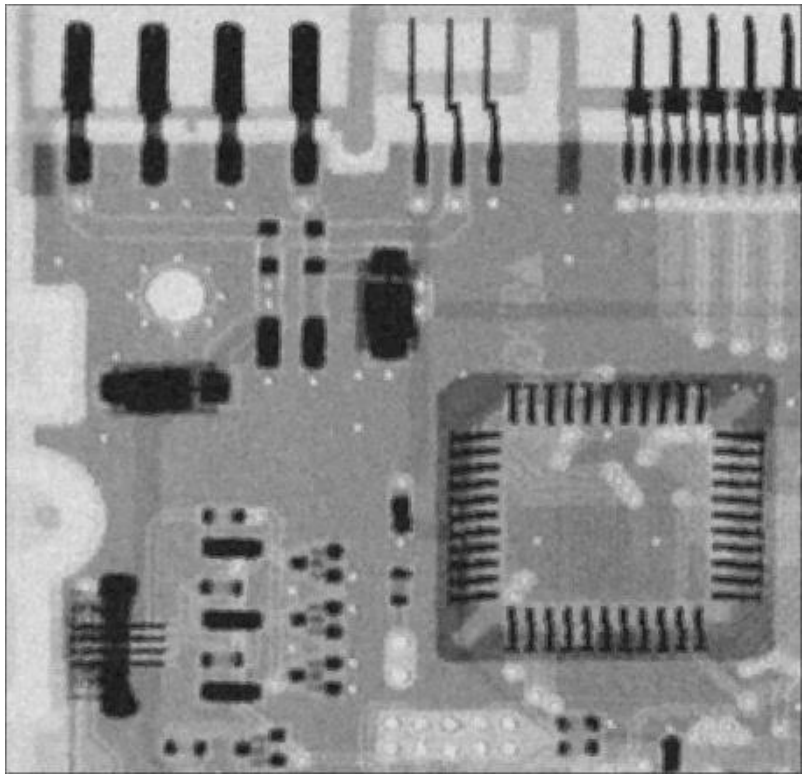
5 X 5 mid point filter

$$f(x,y) = \frac{1}{2} * [\min\{g(s,t)\} + max\{g(s,t)\}]$$

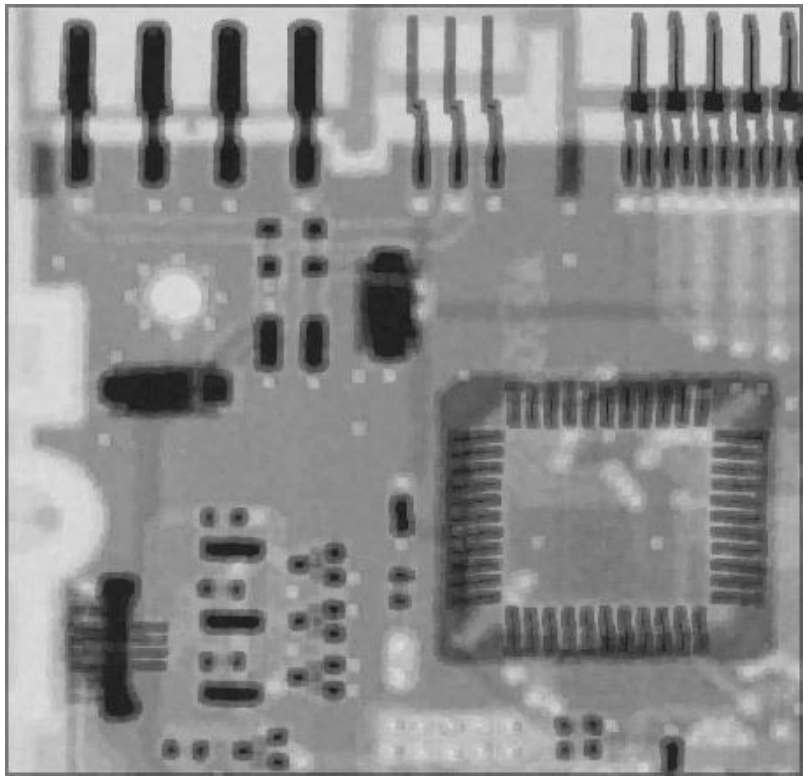
Spatial Domain – Mid point filter (Uniform noise)



Uniform noise image



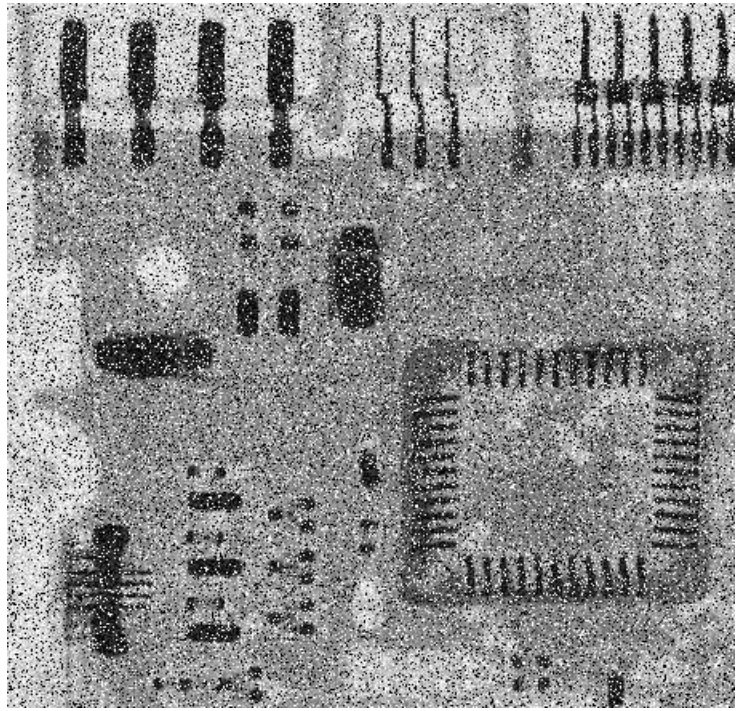
3 X 3 mid point filter



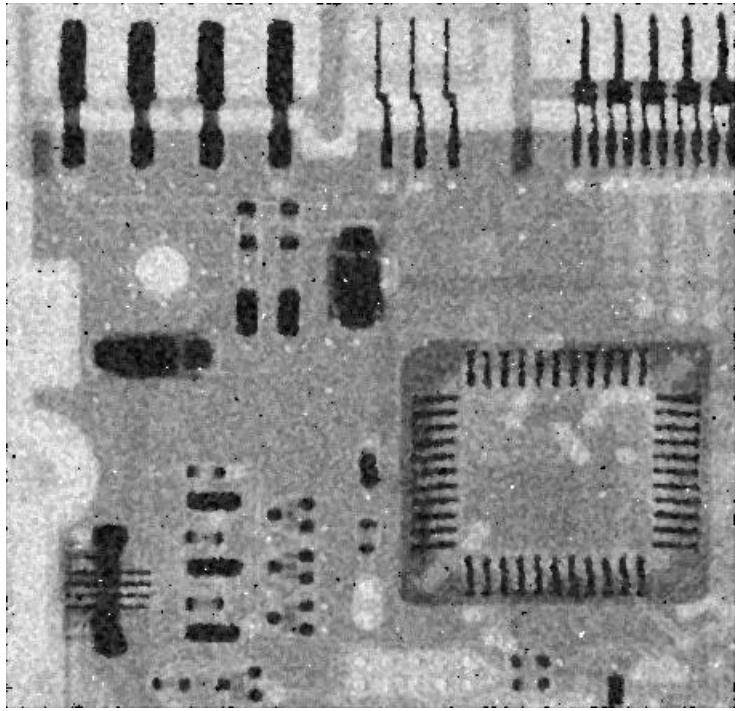
5 X 5 mid point filter

$$f(x,y) = \frac{1}{2} * [\min\{g(s,t)\} + max\{g(s,t)\}]$$

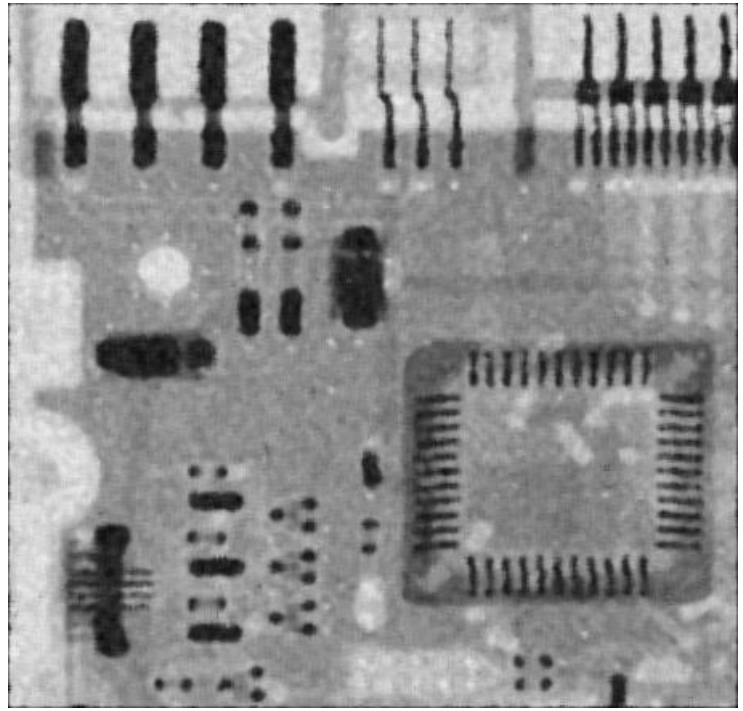
Spatial Domain – Alpha Trimmed mean filter



Salt and pepper
+uniform noise image



5 X 5 Median filter



5 X 5 Alpha trimmed filter

$$\hat{f}(x, y) = \frac{1}{mn - d} \sum_{(s,t) \in S_{xy}} g_r(s, t)$$

(d=5)

Spatial Domain – Adaptive median filter(Algorithm)

z_{min} = minimum intensity value in S_{xy}

z_{max} = maximum intensity value in S_{xy}

z_{med} = median of intensity values in S_{xy}

z_{xy} = intensity value at coordinates (x, y)

S_{max} = maximum allowed size of S_{xy}

Stage A:

$A1 = z_{med} - z_{min}$

$A2 = z_{med} - z_{max}$

If $A1 > 0$ AND $A2 < 0$, go to stage B

Else increase the window size

If window size $\leq S_{max}$ repeat stage A

Else output z_{med}

age B:

$B1 = z_{xy} - z_{min}$

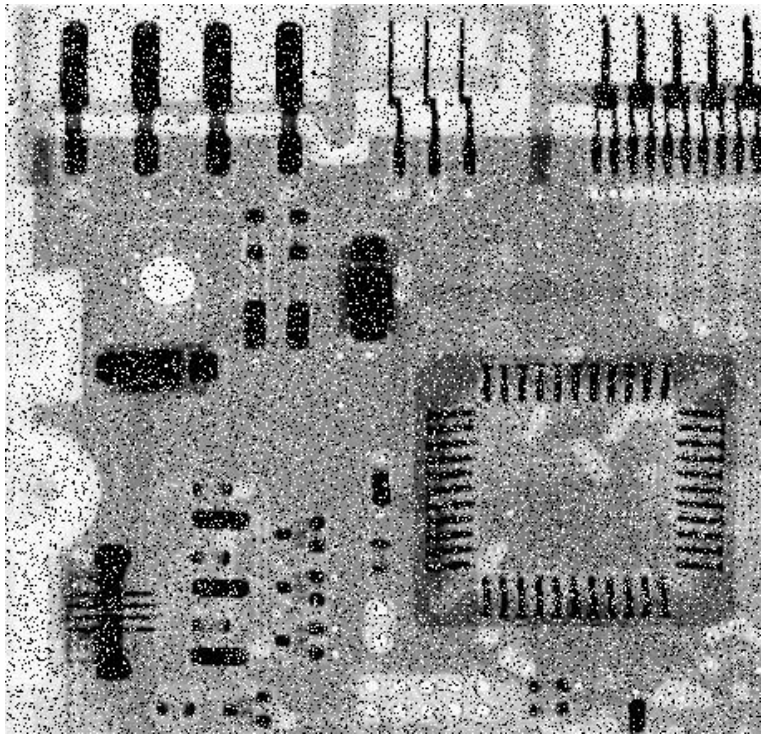
$B2 = z_{xy} - z_{max}$

If $B1 > 0$ AND $B2 < 0$, output z_{xy}

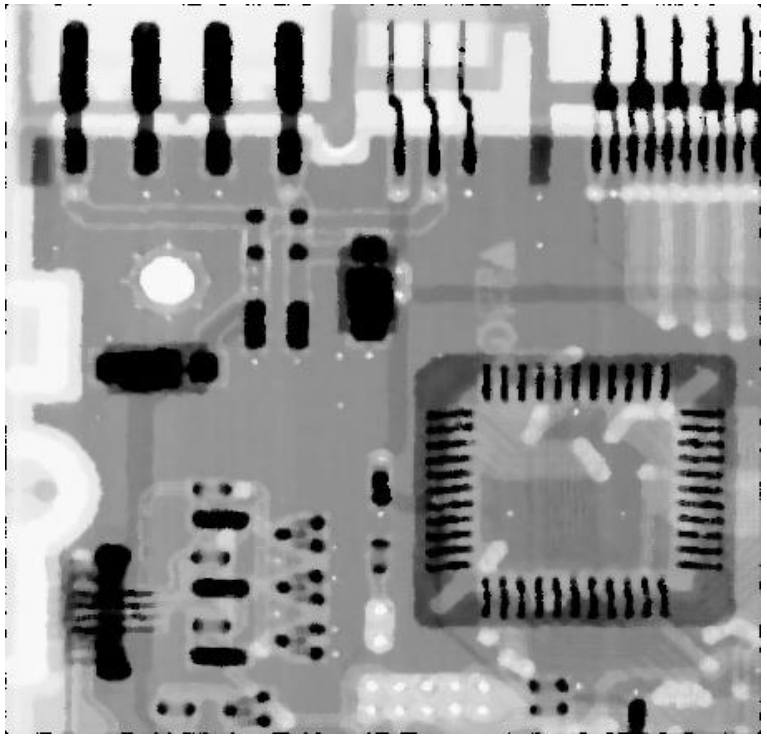
Else output z_{med}

Better at maintaining edges, details

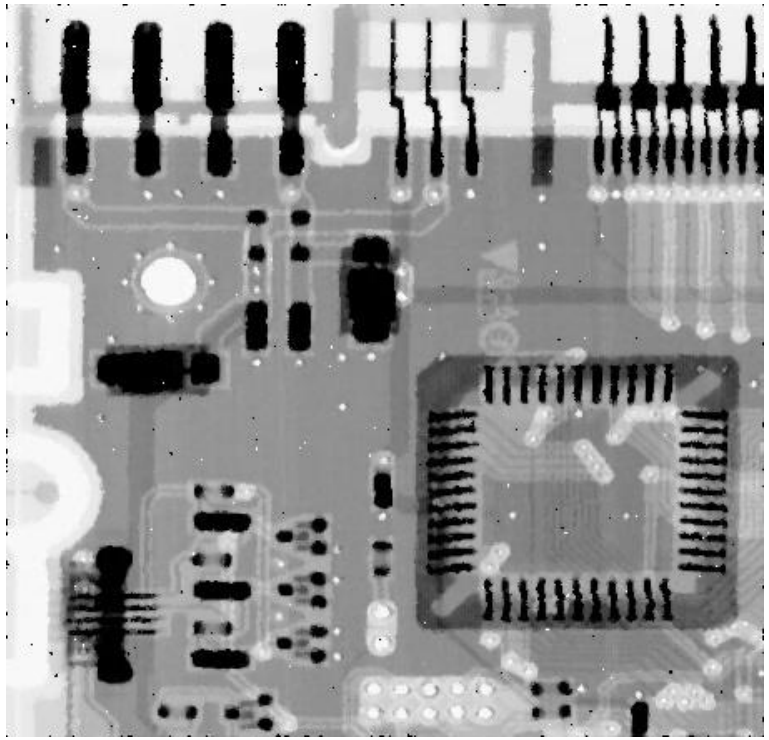
Spatial Domain – Adaptive median filter



Salt pepper noise image



7 X 7 Median filter



Smax=7 Adaptive median filter

Frequency domain – Inverse Filtering

$$\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)}$$

$$\hat{F}(u,v) = F(u,v) + \frac{N(u,v)}{H(u,v)}$$

Not work good at practice, usually bad. Why?

->If H(u,v) is very small for some (u,v), $\frac{N(u,v)}{H(u,v)}$ is very high => poor reconstuction

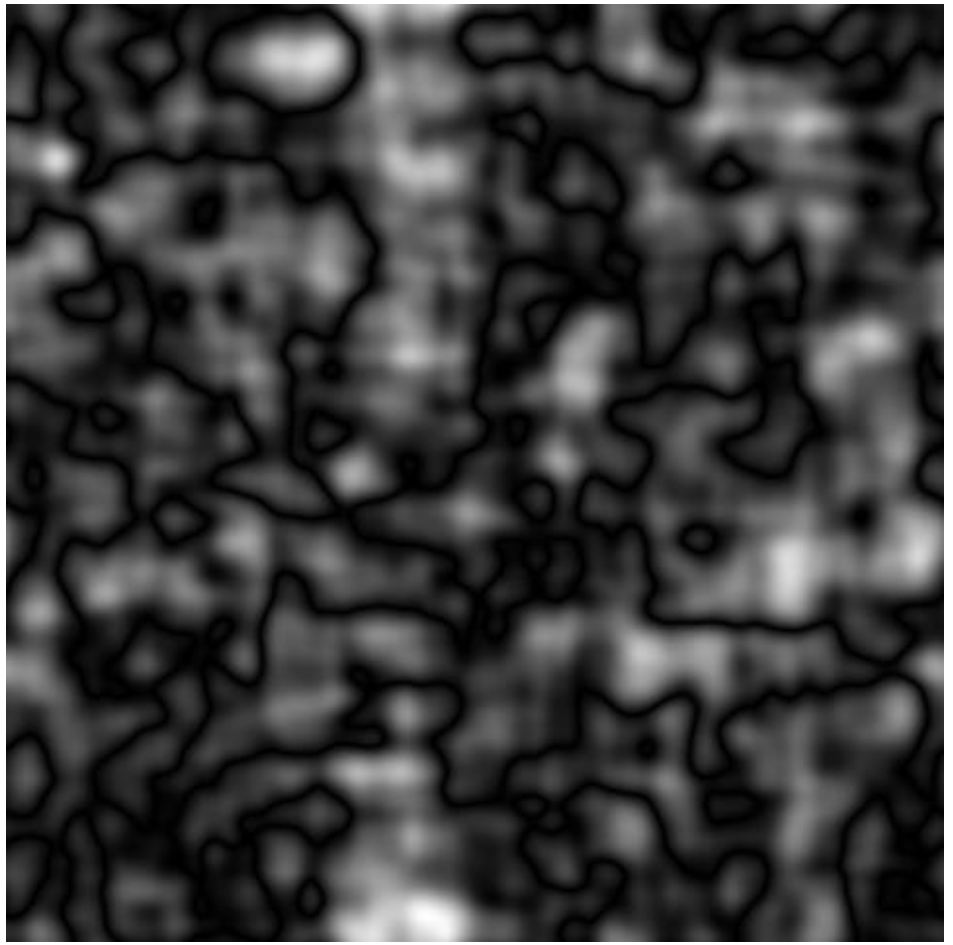
Solution?

->Only apply inverse filter at Low frequencies

Frequency domain – Inverse Filtering

Noise model: $H(u, v) = e^{-k[(u-M/2)^2+(v-N/2)^2]^{5/6}}$

Blurred image, k =0.0025



Inverse Filtered image, k =0.0025

Frequency domain – Inverse Filtering with Butterworth LPF

Blurred image, k =0.0025

LPF after Inverse filter, k=0.0025,R= 40

Frequency domain – Inverse Filtering with Butterworth LPF

BLPF after Inverse filter
k=0.0025, $R = 40$

BLPF after Inverse filter
k=0.0025, $R = 50$

BLPF after Inverse filter
k=0.0025, $R = 60$

BLPF after Inverse filter
k=0.0025, $R = 66$

Frequency domain – Inverse Filtering with Butterworth LPF

Original(No noise)

Blurred image

$$H(u, v) = e^{-k[(u-M/2)^2+(v-N/2)^2]^{5/6}}$$

BLPF after Inverse filter
k=0.0025, $R = 66$

Frequency domain – Wiener filter

$H(u, v)$ = degradation function

$H^*(u, v)$ = complex conjugate of $H(u, v)$

$|H(u, v)|^2 = H^*(u, v)H(u, v)$

$S_\eta(u, v) = |N(u, v)|^2$ = power spectrum of the noise [see Eq. (4.6–18)]†

$S_f(u, v) = |F(u, v)|^2$ = power spectrum of the undegraded image

$$\hat{F}(u, v) = \left[\frac{H^*(u, v)S_f(u, v)}{S_f(u, v)|H(u, v)|^2 + S_\eta(u, v)}\right]G(u, v)$$

$$= \left[\frac{H^*(u, v)}{|H(u, v)|^2 + S_\eta(u, v)/S_f(u, v)}\right]G(u, v)$$

$$= \left[\frac{1}{H(u, v)}\frac{|H(u, v)|^2}{|H(u, v)|^2 + S_\eta(u, v)/S_f(u, v)}\right]G(u, v)$$

$$\hat{F}(u, v) = \left[\frac{1}{H(u, v)}\frac{|H(u, v)|^2}{|H(u, v)|^2 + K}\right]G(u, v)$$

K = tunable parameter(SNR)
No noise => Same as Inverse filter
Noise => Doesn't have bad numerical properties
Inverse filter does

Frequency domain – Wiener filter

Wiener filter
k=0.0025, $K = 0.01$

Wiener filter
k=0.0025, $K = 0.001$

Wiener filter
k=0.0025, $K = 0.0001$

Frequency domain – Wiener filter

Original(No noise)

Blurred image

$$H(u,v) = e^{-k[(u-M/2)^2+(v-N/2)^2]^{5/6}}$$

Wiener filter
k=0.0025, $K = 0.0001$

Frequency domain – Comparsion

$$H(u, v) = e^{-k[(u-M/2)^2+(v-N/2)^2]^{5/6}}$$

Original(No noise)

BLPF after Inverse filter
k=0.0025, $\sigma = 66$

Wiener filter
k=0.0025, $K = 0.0001$

THANK YOU
for your attention