

13 – Restorasi Citra (Bagian 1)

IF4073 Pemrosesan Citra Digital

Oleh: Dr. Ir. Rinaldi, M.T



Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2025

Tujuan Restorasi Citra

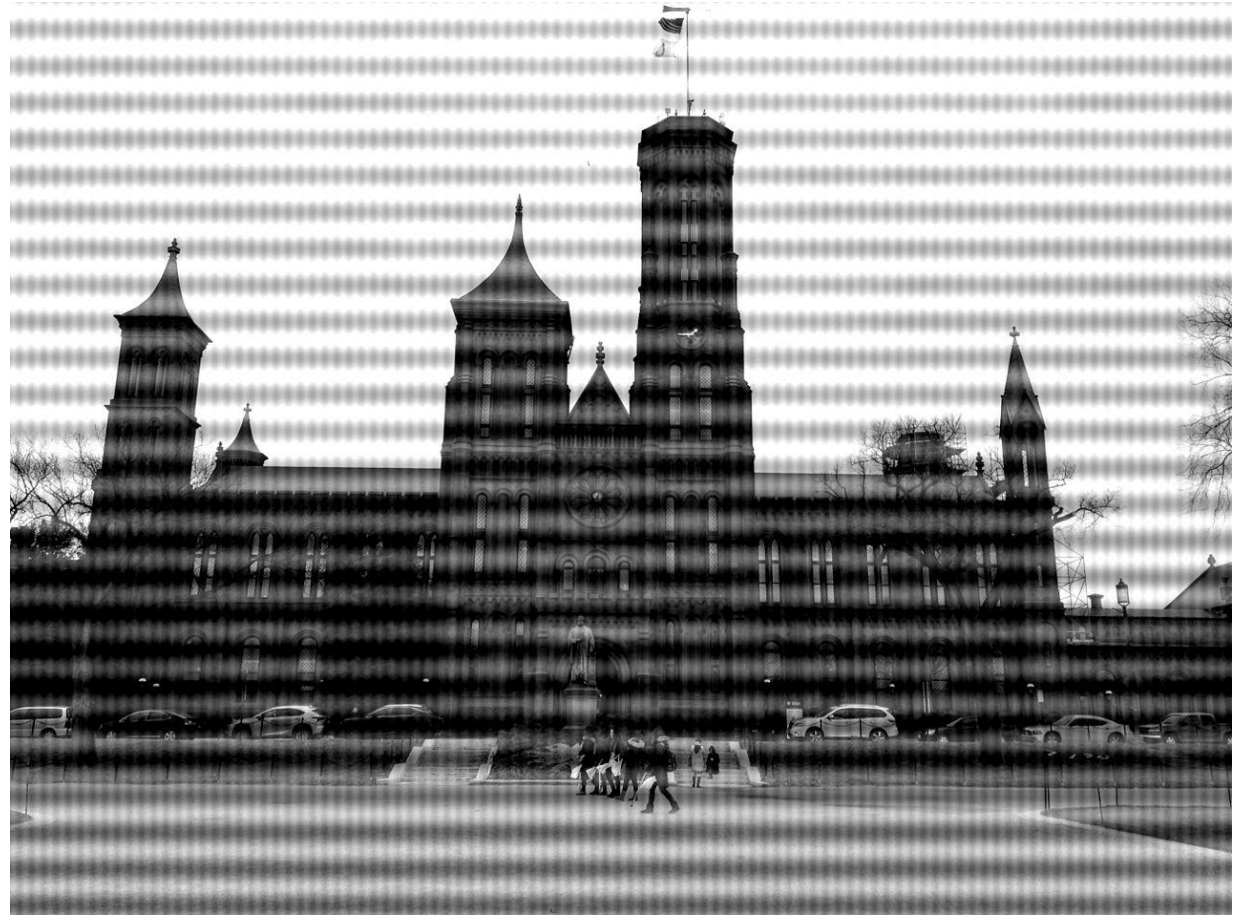
- Restorasi citra bertujuan untuk **merekonstruksi** kembali citra yang mengalami distorsi (*distorted image*) atau degradasi (*degraded image*) menjadi **bentuk citra semula** (*original image*) berdasarkan **model yang ideal**.
- Model diperoleh berdasarkan pengetahuan tentang penyebab citra mengalami distorsi.
- Modelkan degradasi (*original image* $\rightarrow$ *distorted image*), kemudian lakukan proses berkebalikan (*inverse*) untuk merekonstruksi citra semula (*distorted image* $\rightarrow$ *original image*)
- Citra hasil rekonstruksi sedapat mungkin menghasilkan citra yang mendekati citra semula. Jadi, citra hasil rekonstruksi merupakan **estimasi** citra semula.

Penyebab Distorsi

- Distorsi pada citra timbul selama proses akuisisi citra dan/atau selama transmisi citra.
- Faktor-faktor penyebab distorsi pada citra:
 - Degradasi citra akibat kondisi lingkungan selama proses transmisi citra
Contoh: petir, turbulensi atmosfer
 - Derau pada citra akibat kualitas penginderaan pada peralatan sensor (**sensor noise**)
Contoh: level cahaya dan temperatur sensor pada kamera CCD
 - *Blurring* pada citra diakibatkan oleh sensor
Contoh: kamera bergerak atau out-of-focus
 - Distorsi geometrik
Contoh: foto bumi yang diambil oleh kamera dari satelit



blurred image



Citra yang mengalami derau periodik



Citra dengan derau periodik lainnya

noisy image (salt and peppers)



Noisy image (salt and pepper noise)

Citra yang mengalami turbulensi
atmosfir





Geometric distortion

Image Restoration vs Image Enhancement

- *Image enhancement* memiliki area yang beririsan dengan *image restoration*.
- Beberapa teknik di dalam *image enhancement* juga digunakan di dalam *image restoration*, misalnya *median filtering*, *mean filtering*.

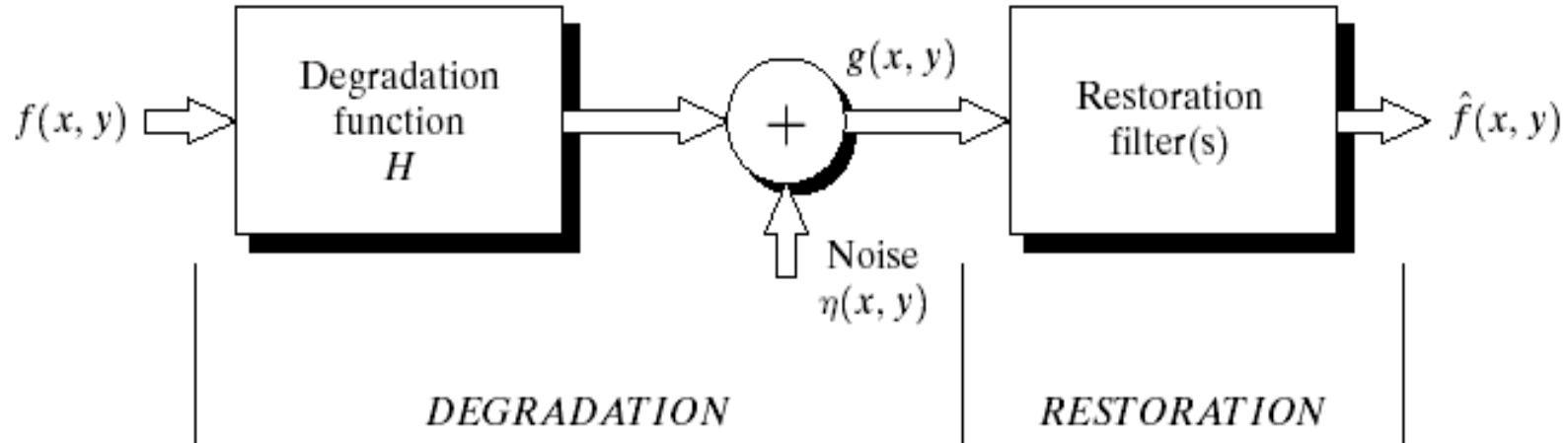
• Enhancement

- Berkaitan dengan ekstraksi fitur citra (kontras, kecerahan, histogram, dll)
- Sulit mengkuantisasi kinerja metodenya
- Lebih subyektif; membuat citra “tampil lebih baik (*look better*)”

• Restoration

- Berkaitan dengan pemulihan citra yang mengalami degradasi
- Kinerja metode dapat dikuantisasi
- Lebih obyektif; merekonstruksi menjadi citra semula

Model Degradasi Citra dan Proses Restorasi



$f(x, y)$: citra input

$g(x, y)$: citra degradasi

H : fungsi degradasi

$\eta(x, y)$: derau aditif

$\hat{f}(x, y)$: estimasi citra semula

Dalam ranah spasial: $g(x, y) = f(x, y) * h(x, y) + \eta(x, y)$

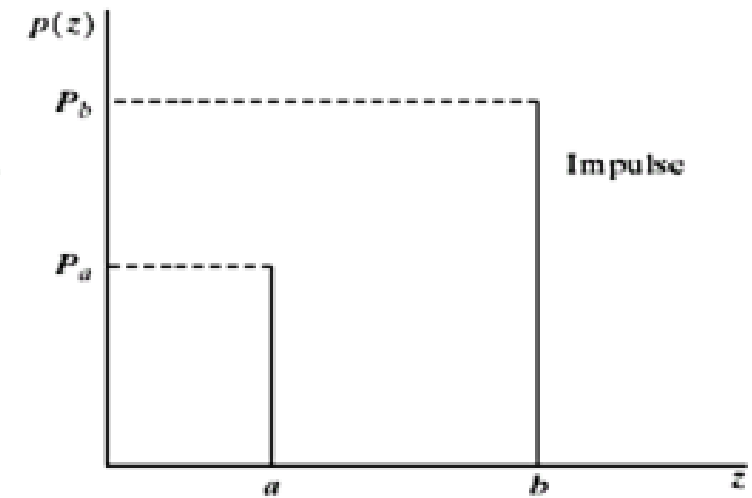
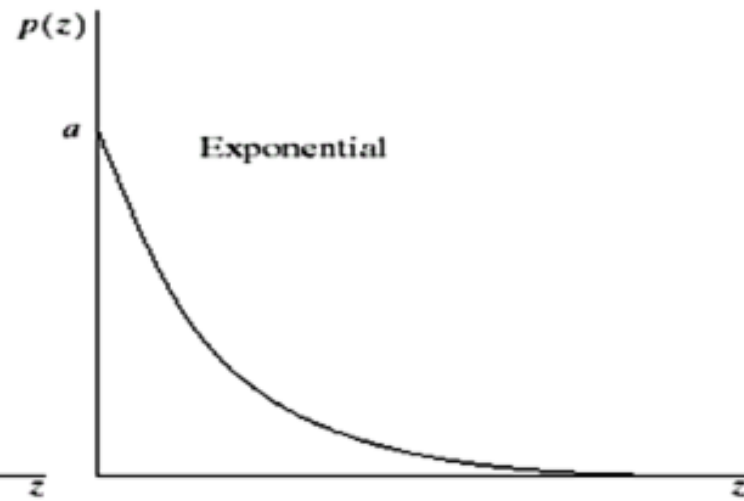
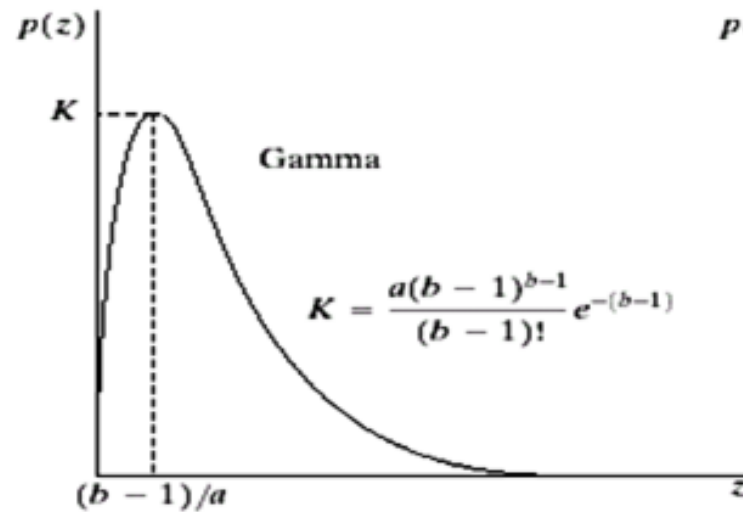
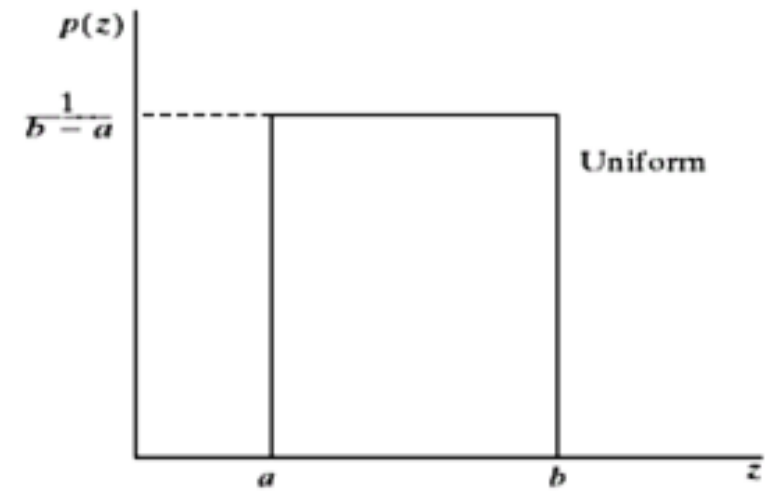
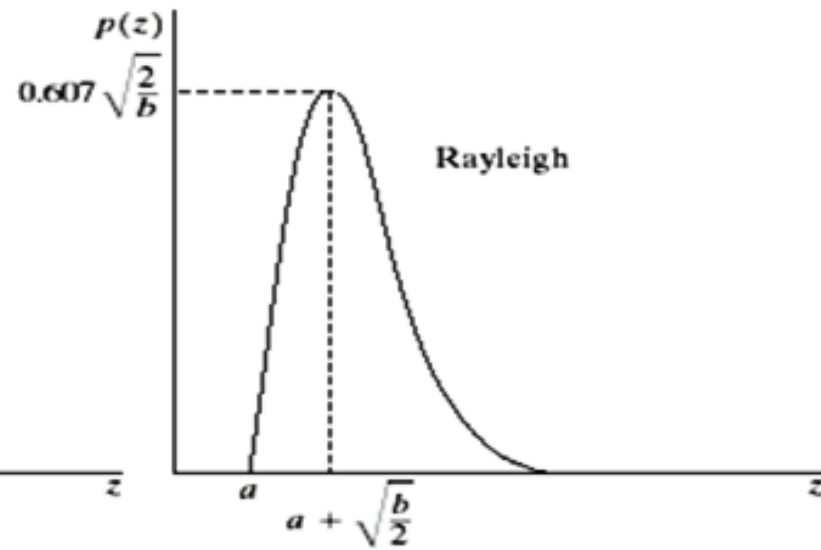
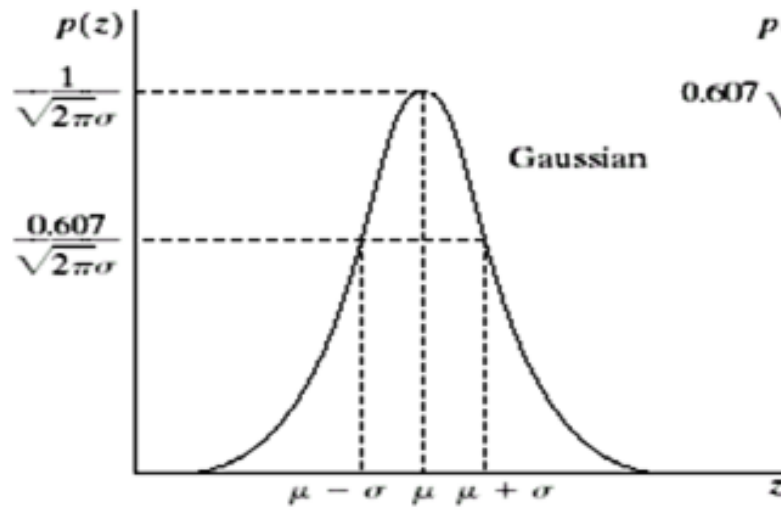
Dalam ranah frekuensi: $G(u, v) = F(u, v)H(u, v) + N(u, v)$

Restorasi dalam Ranah Spasial dan Frekuensi

- Beberapa teknik restorasi lebih tepat dilakukan dalam ranah spasial, sedangkan beberapa teknik restorasi lainnya hanya dapat dilakukan dalam ranah frekuensi.
- Degradasi yang disebabkan oleh derau aditif lebih tepat dilakukan dalam ranah spasial. Misalnya *salt & pepper noise*.
- Degradasi seperti citra kabur (*image blur*) dan citra yang mengalami derau periodik sulit dilakukan dalam ranah spasial. Degradasi seperti itu hanya dapat dilakukan dalam ranah frekuensi.

Model Derau (Noise)

- Asumsikan derau:
 - independen dari koordinat spasial
 - tidak berkorelasi dengan konten citra
- Derau dianggap sebagai variabel acak. Variabel acak tersebut dicirikan oleh fungsi kepadatan peluang (*probability density function*) atau PDF.
- Beberapa PDF yang umum ditemukan di dalam aplikasi pengolahan citra adalah: *Gaussian noise*, *Rayleigh noise*, *Erlang (gamma) noise*, *exponential noise*, *uniform noise*, dan *impulse (salt & pepper) noise*.



- **Gaussian noise**

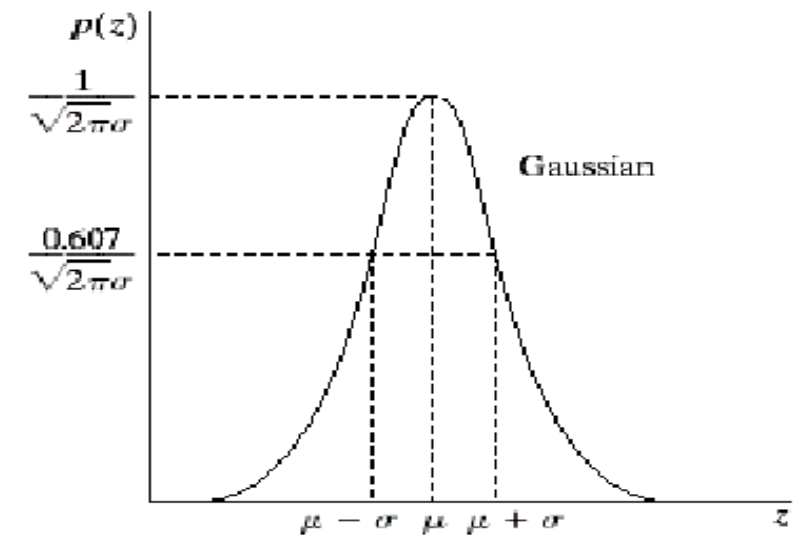
- Probability density function (PDF)

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z-\mu)^2 / 2\sigma^2}$$

- z : gray level (Gaussian random variable)
- μ : mean of average value of z
- σ : standard deviation of z
- σ^2 : variance of z

- **PDF of Gaussian noise**

- 70% of z in $[\mu - \sigma, \mu + \sigma]$
- 90% of z in $[\mu - 2\sigma, \mu + 2\sigma]$



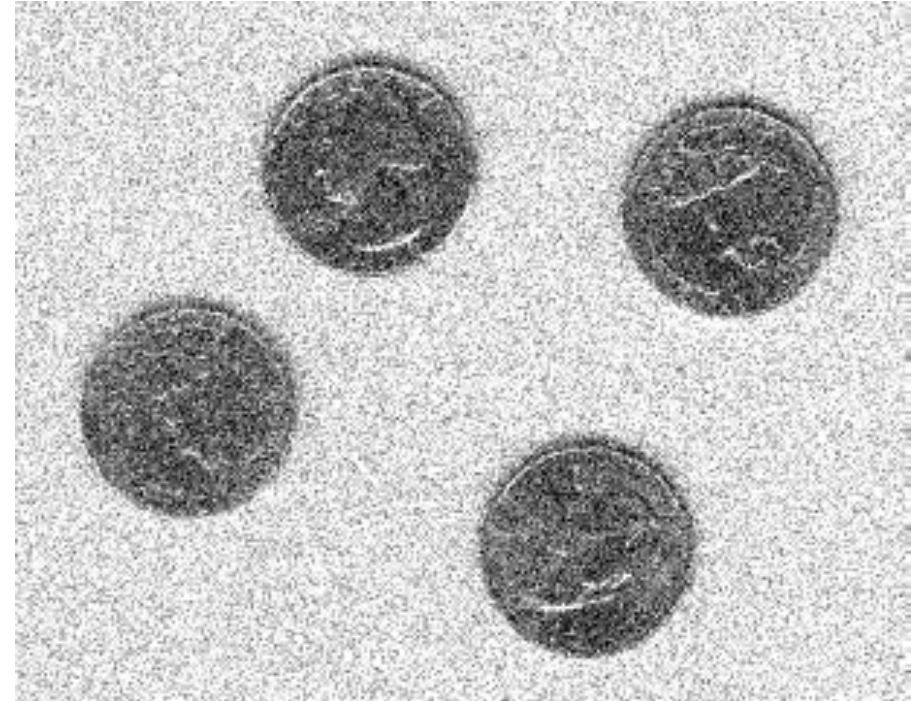
- Membuat citra yang mengandung derau *gaussian* dengan Matlab

```
A = imread('eight.png');  
B = imnoise(A, 'gaussian', 0, 0.02); % mean = 0.0, variansi = 0.02  
imshow(A), title('original image');  
figure, imshow(B), title('noisy image (gaussian)');
```

original image



noisy image (gaussian)

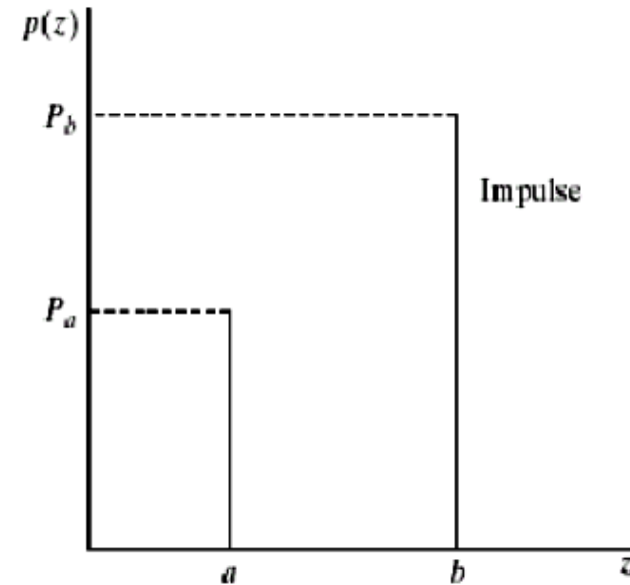


```
I = imread('house.jpg');  
I_noise = imnoise(I, 'gaussian'); % default mean = 0.0, variansi = 0.01  
imshow(I); figure, imshow(I_noise)
```



- **Impulse (salt-and-pepper) noise**

$$p(z) = \begin{cases} P_a & \text{for } z = a \\ P_b & \text{for } z = b \\ 0 & \text{otherwise} \end{cases}$$



- bipolar if $P_a \neq 0, P_b \neq 0$
 - unipolar if one of P_a and P_b is 0
 - noise looks like salt-and-pepper granules if $P_a \approx P_b$
 - negative or positive; scaling is often necessary to form digital images
 - extreme values occur (e.g. $a = 0, b = 255$)
- Dalam praktek, **impuls** biasanya lebih kuat dari nilai keabuan di dalam citra. Misalkan $a = 0$ (hitam) dan $b = 255$ (putih) untuk citra 8-bit.

- Membuat citra yang mengandung derau *salt & pepper* dengan Matlab

```
A = imread('eight.png');  
B = imnoise(A, 'salt & pepper', 0.02); % density = 0.02  
imshow(A), title('original image');  
figure, imshow(B), title('noisy image (salt and peppers)');
```

original image



noisy image (salt and peppers)



```
I = imread('house.jpg');  
I_noise = imnoise(I, 'salt & pepper'); % default d = 0.05  
imshow(I); figure, imshow(I_noise)
```



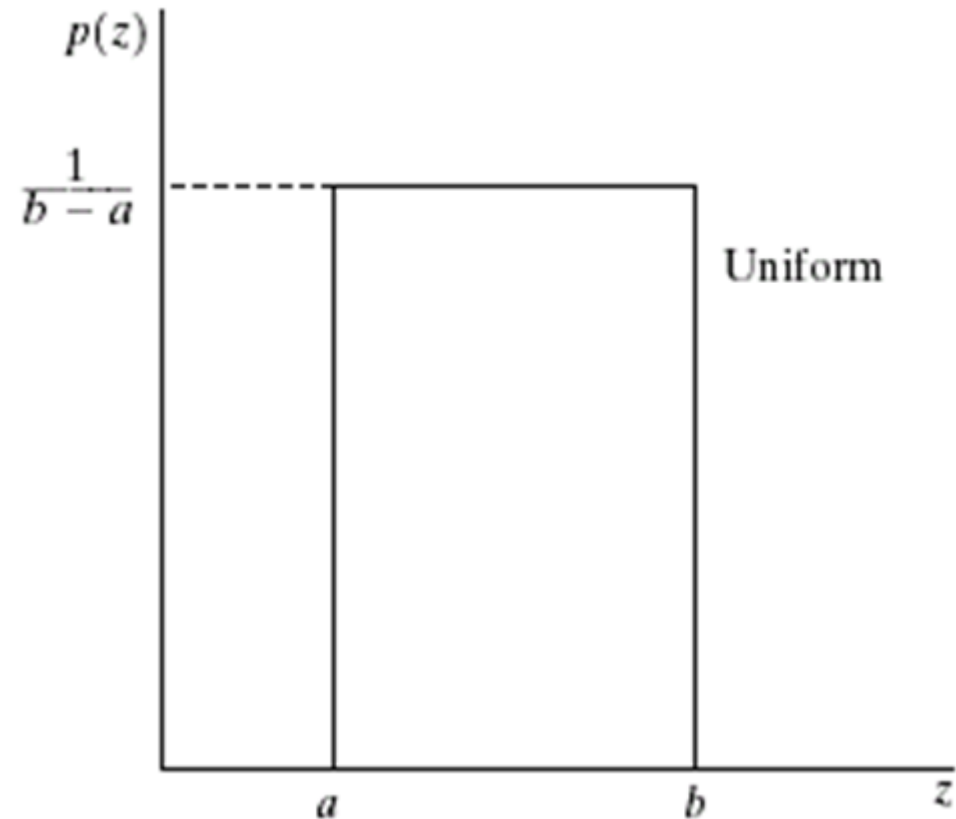
- **Uniform noise**

- kurang praktis, biasanya digunakan untuk pembangkit bilangan acak

$$p(z) = \begin{cases} \frac{1}{b-a} & \text{if } a \leq z \leq b \\ 0 & \text{otherwise} \end{cases}$$

Mean: $\mu = \frac{a+b}{2}$

Variansi: $\sigma^2 = \frac{(b-a)^2}{12}$



- Rayleigh Noise

$$p(z) = \frac{2}{b}(z-a)e^{-(z-a)^2/b} \quad \text{for } z \geq a$$

$$= 0 \quad \text{for } z < a$$

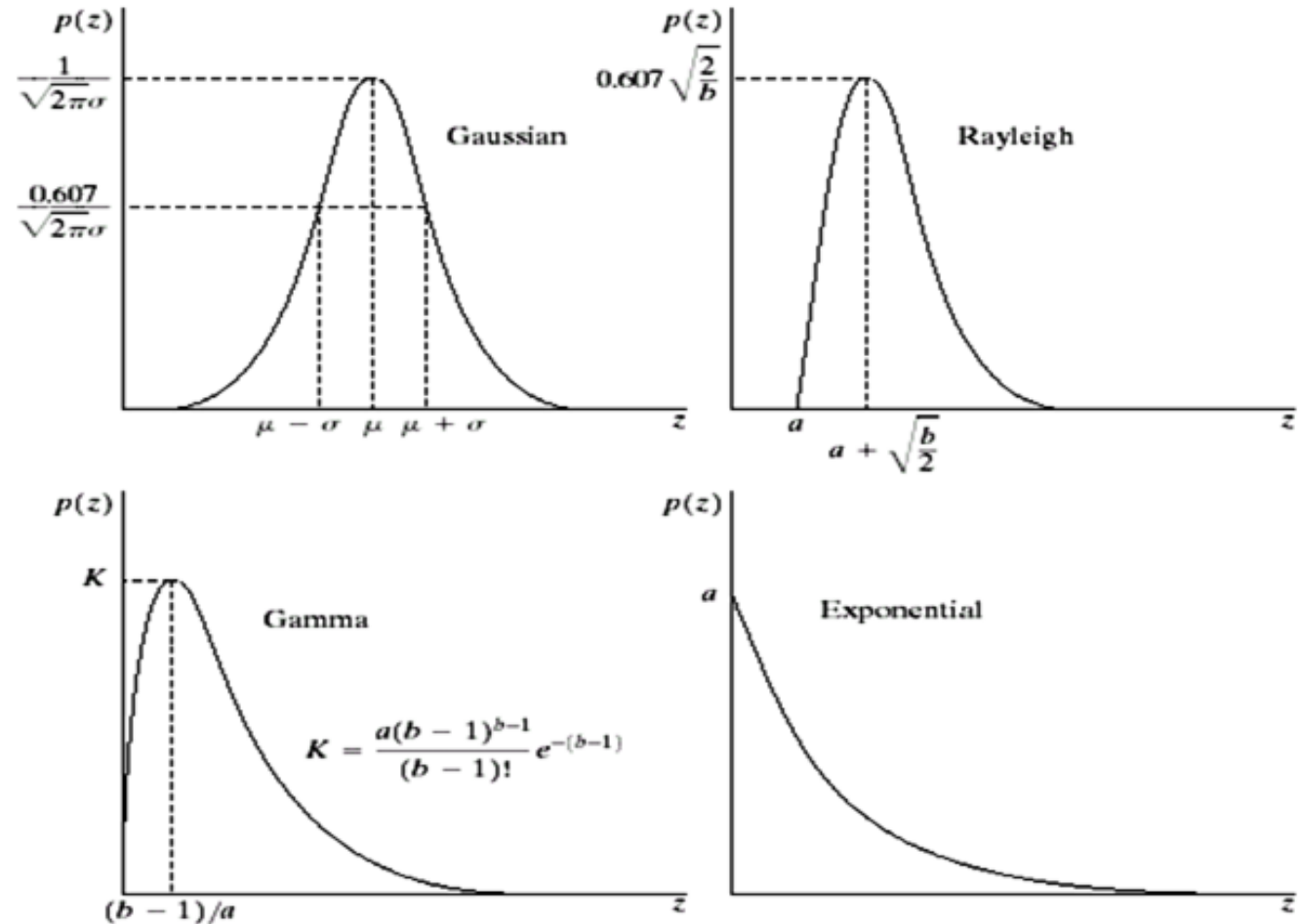
- Gamma(Erlang) Noise

$$p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az} & \text{for } z \geq 0 \\ 0 & \text{for } z < 0 \end{cases}$$

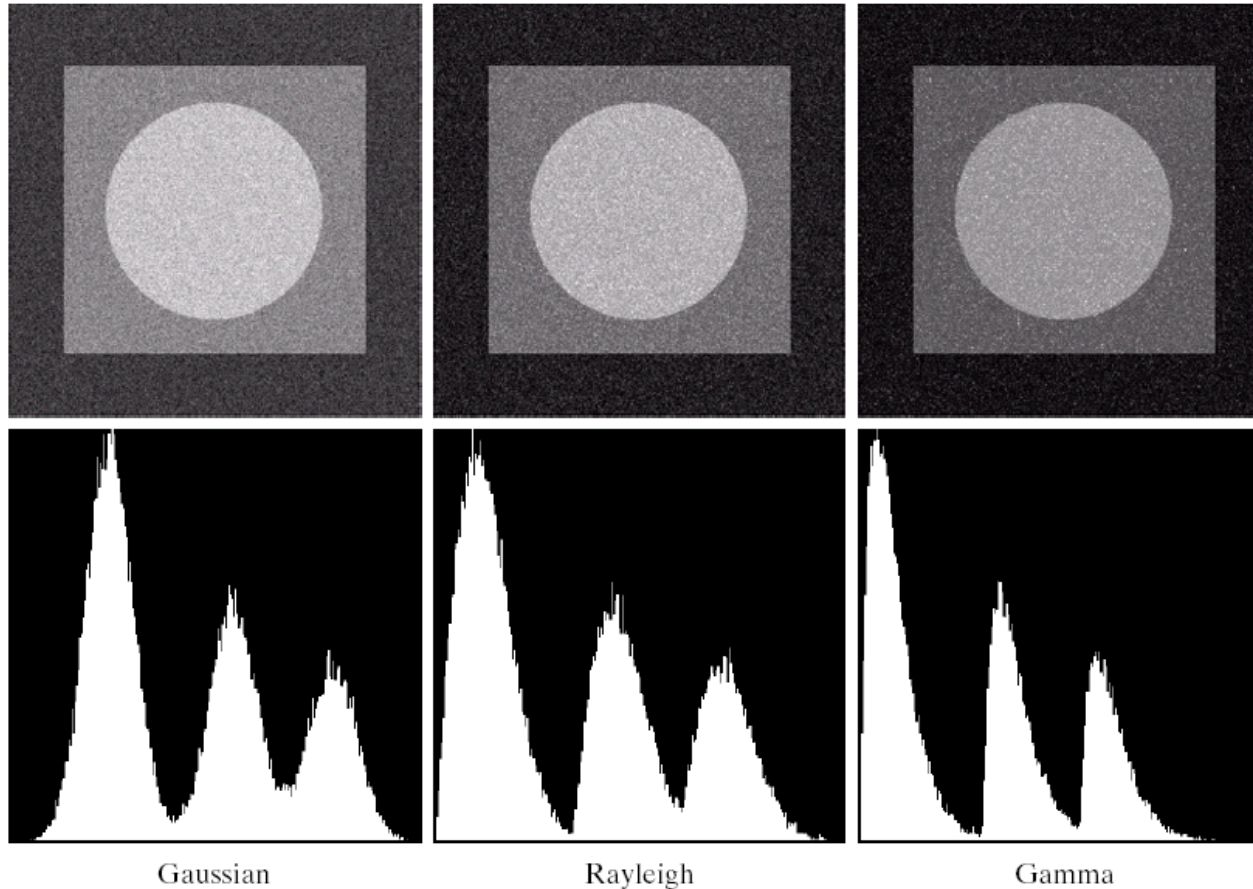
- Exponential Noise

$$p(z) = ae^{-az} \quad \text{for } z \geq 0$$

$$= 0 \quad \text{for } z < 0$$



Derau Aditif (Additive Noise)



Histograms

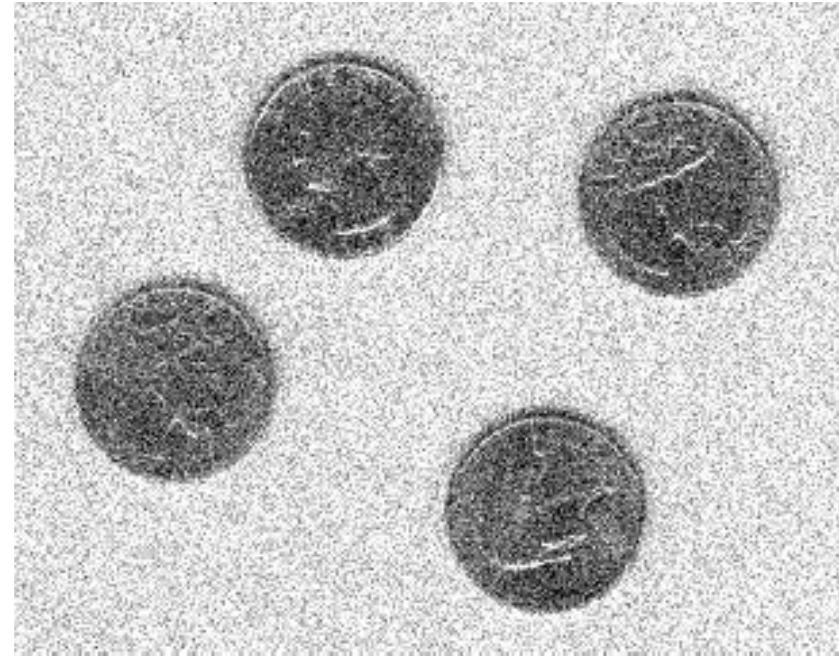
a	b	c
d	e	f

FIGURE 5.4 Images and histograms resulting from adding Gaussian, Rayleigh, and gamma noise to the image in Fig. 5.3.

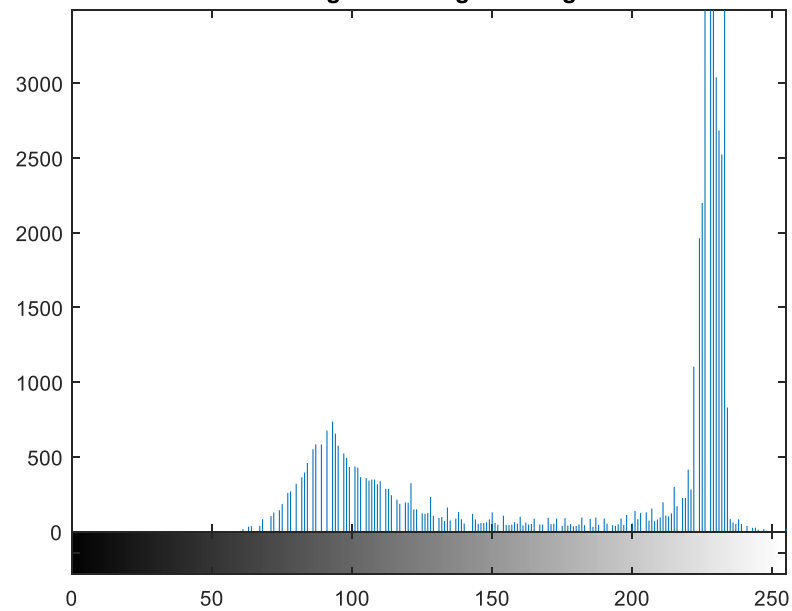
original image



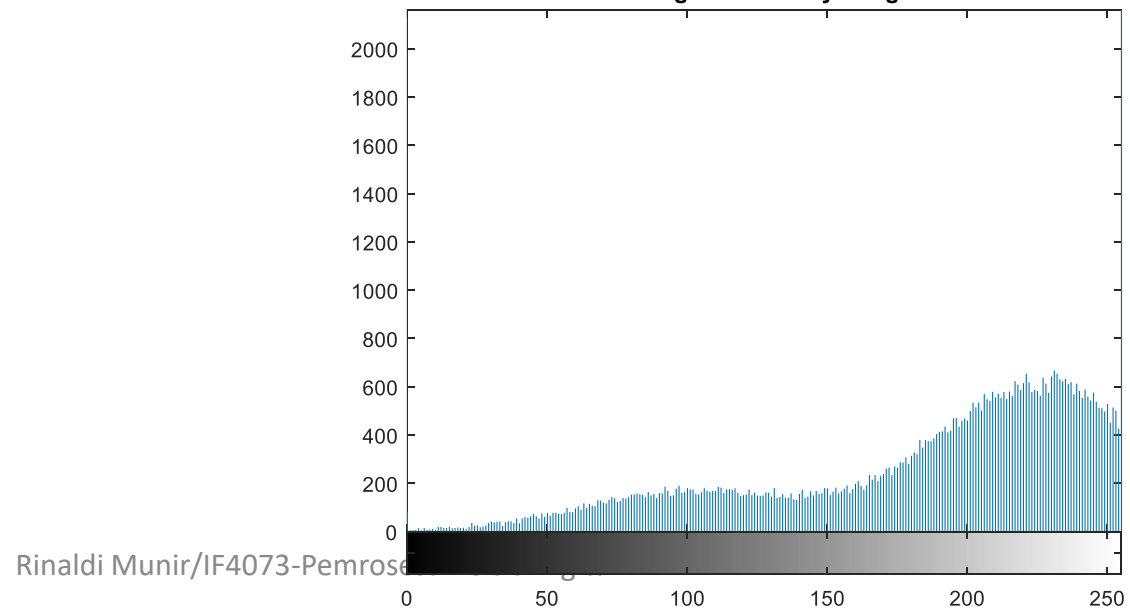
noisy image (gaussian)



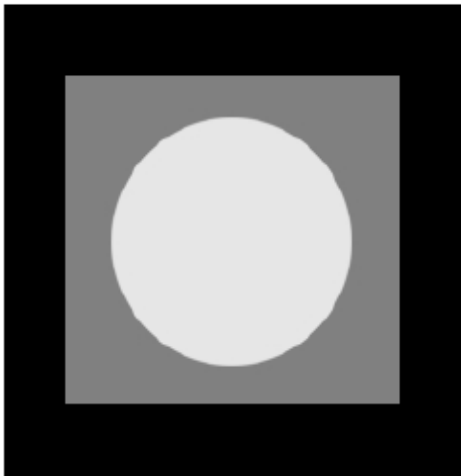
histogram of original image



histogram of noisy image

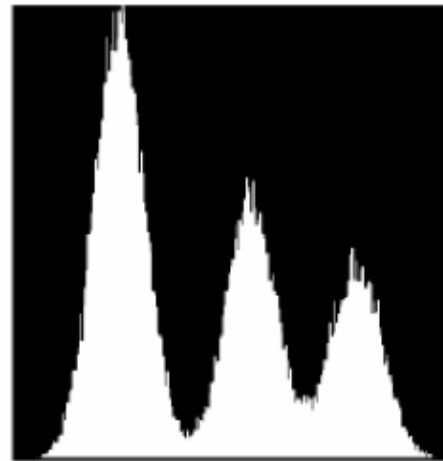
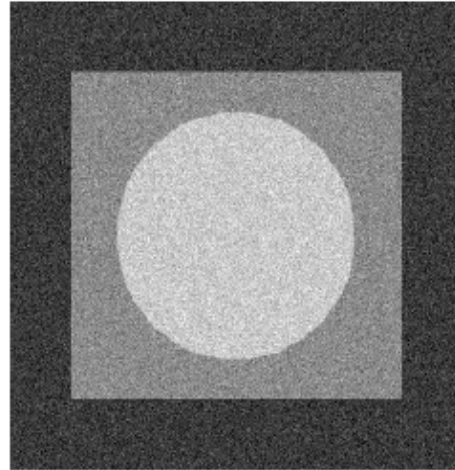


- **An example**

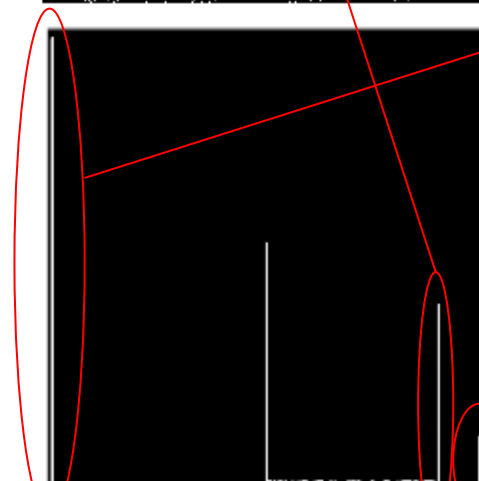
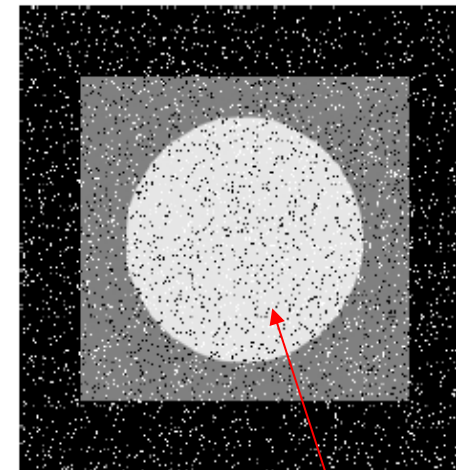


Test pattern

-3-levels
-simple constant areas
(spans from black to white)



Histogram
Gaussian noise added



pepper

salt

Histogram
Impulse noise added

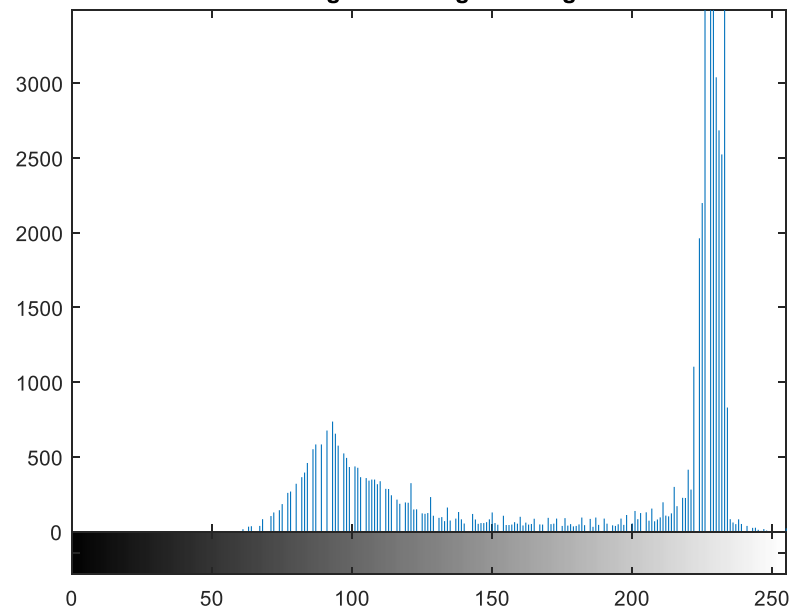
original image



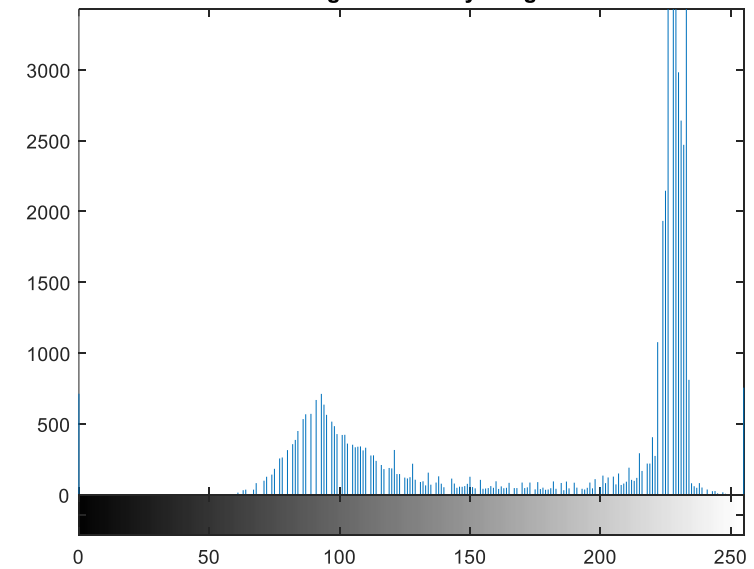
noisy image (salt and peppers)



histogram of original image



histogram of noisy image

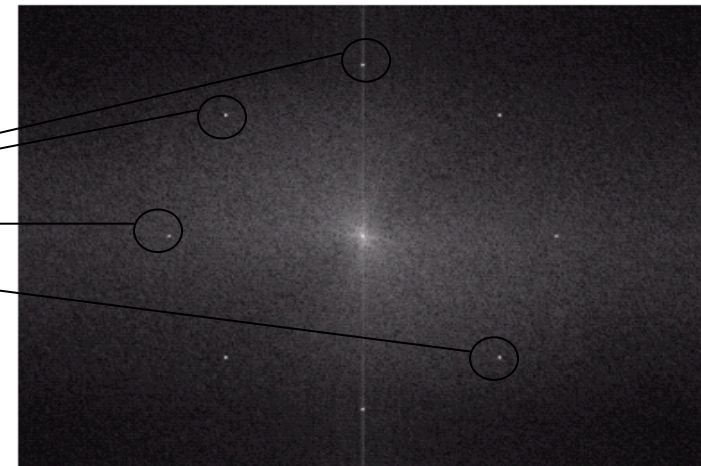
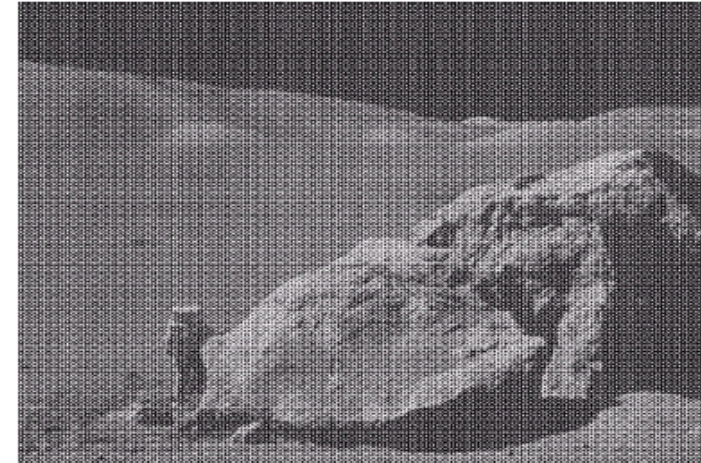


Derau Periodik (*Periodic Noise*)

a
b

FIGURE 5.5

(a) Image corrupted by sinusoidal noise.
(b) Spectrum (each pair of conjugate impulses corresponds to one sine wave). (Original image courtesy of NASA.)



Komponen derau
(sinusoidal noise)

Derau periodik dapat dikurangi dengan operasi dalam ranah frekuensi

Dihasilkan karena gangguan listrik atau elektromekanis selama akuisisi gambar

- Program Matlab untuk menambahkan derau periodik ke dalam citra

```
I = imread('lena.bmp');  
J = rgb2gray(I);  
s = size(J);  
[x,y] = meshgrid(1:s(1), 1:s(2));  
p = sin(x/3+y/5)+1;  
J_noise = (im2double(J)+ p'/2)/2;  
imshow(J);  
figure, imshow(J_noise);
```

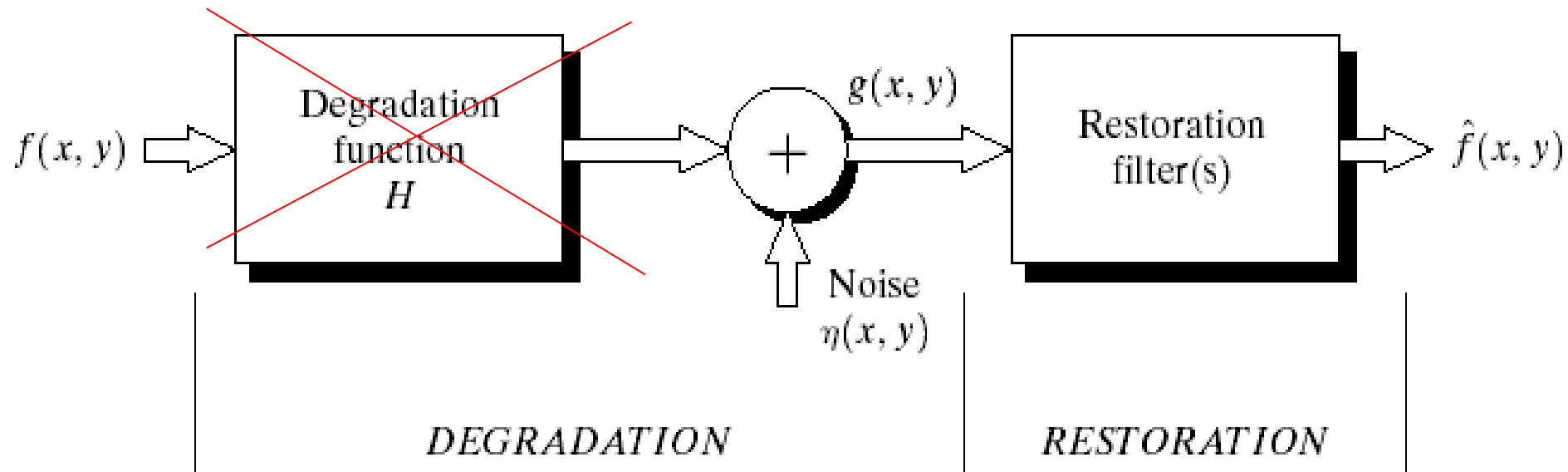


Restorasi dengan Penapisan Spasial

- Asumsikan degradasi di dalam citra hanya karena derau saja

$$g(x,y) = f(x,y) + \eta(x,y)$$

$$G(u,v) = F(u,v) + N(u,v)$$



- Penapisan spasial (*spatial filtering*):
 - merupakan metode restorasi jika hanya terdapat derau aditif di dalam citra
 - sama seperti *image enhancement* dalam ranah spasial
- Sebuah “jendela” (*window*) S berukuran $m \times n$ memuat sejumlah *pixel* digeser titik demi titik pada seluruh daerah citra. Nilai *pixel* yang di tengah jendela diganti dengan sebuah nilai hasil perhitungan
- Penapis yang digunakan:
 - Penapis rerata (*mean filters*)
 - Penapis orde-statistic (*order-statistics filters*)
 - Penapis adaptif (*adaptive filters*)



Penapis Rerata (*Mean Filters*)

- *Arithmetic mean*

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

Window centered at (x,y)

- *Geometric mean*

$$\hat{f}(x, y) = \left[\prod_{(s,t) \in S_{xy}} g(s, t) \right]^{1/mn}$$

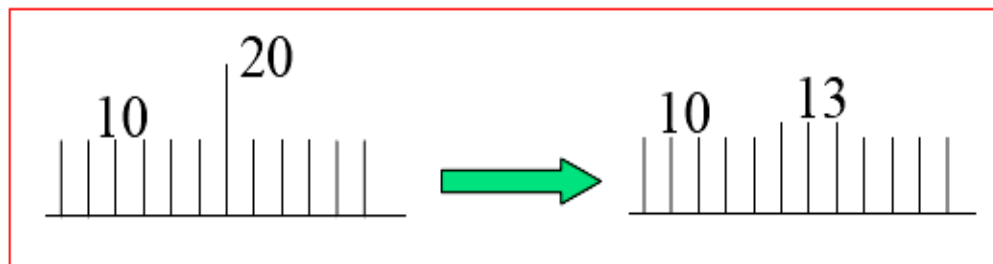
S_{xy} is the set of coordinates in a rectangular subimage window of size $m \times n$ centered at point (x, y)

- **Mean filters (noise reduced by blurring)**

- **Arithmetic mean filter**

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

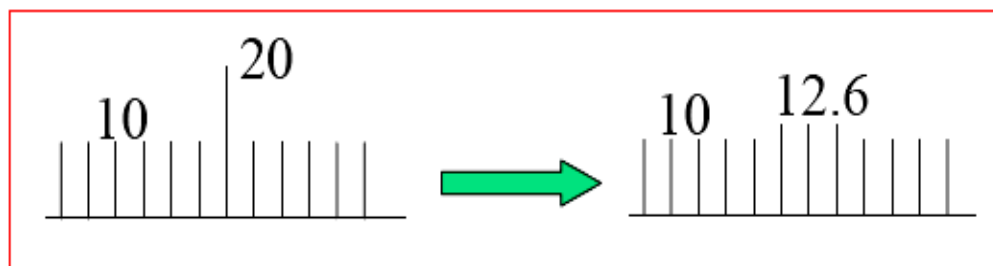
1 x 3 mask



- **Geometric mean filter**

$$\hat{f}(x, y) = \left[\prod_{(s,t) \in S_{xy}} g(s, t) \right]^{\frac{1}{mn}}$$

1D illustration



- ◆ Smoothing comparable to arithmetic mean filter
 - ◆ Losing less image details

Program Matlab untuk Arithmetic mean filtering

```
I = imread('zelda.bmp');  
Inoise = imnoise(I, 'gaussian'); %default mean = 0 dan variansi = 0.01  
figure, imshow(I), title ('Original image');  
figure, imshow(Inoise), title ('Noisy image');  
mean = fspecial('average', [5 5]); %default 3 x 3 filter  
Ifiltered = uint8(convn(double(Inoise), double(mean)));  
figure, imshow(Ifiltered), title ('Filtered image');
```

Original image



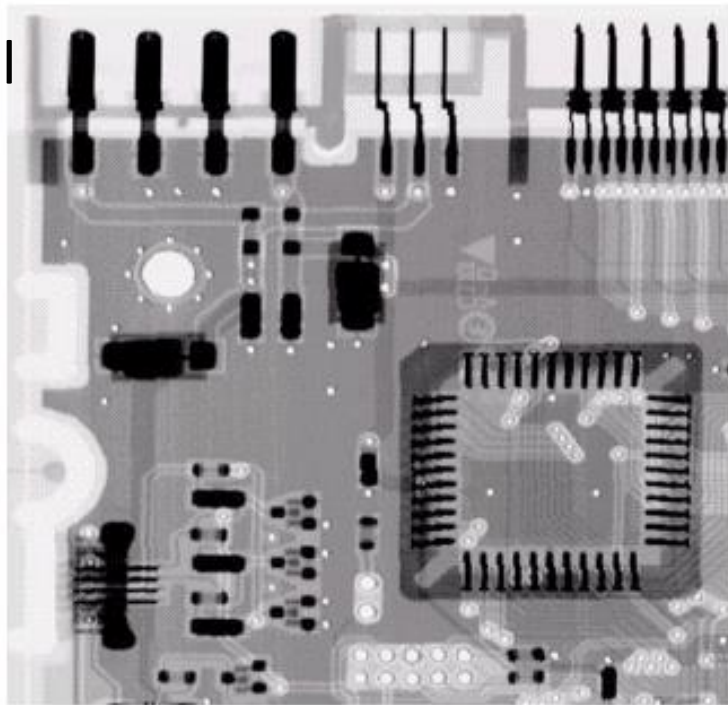
Noisy image



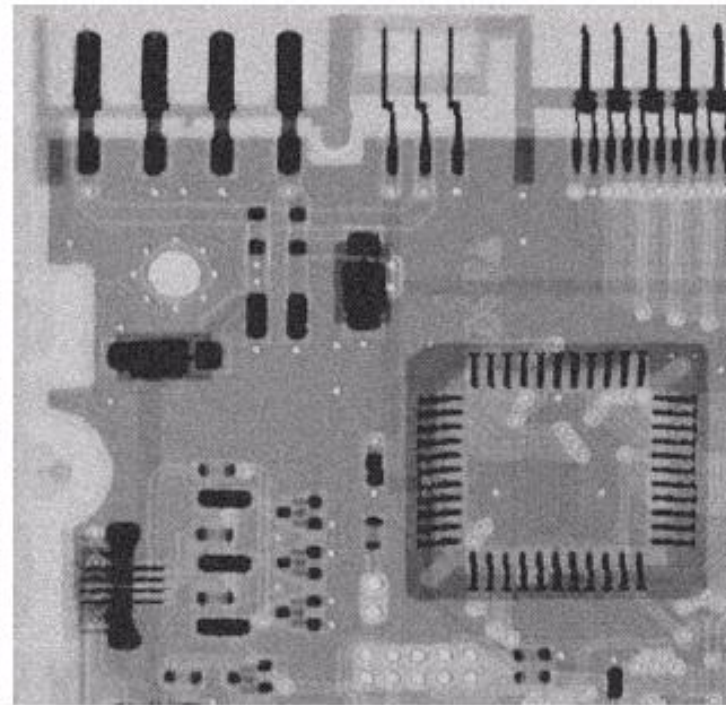
Filtered image



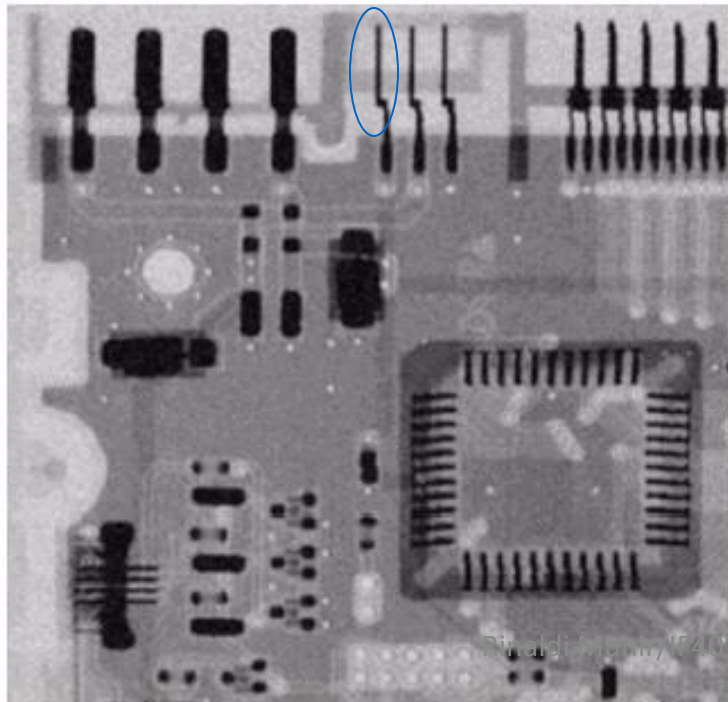
original



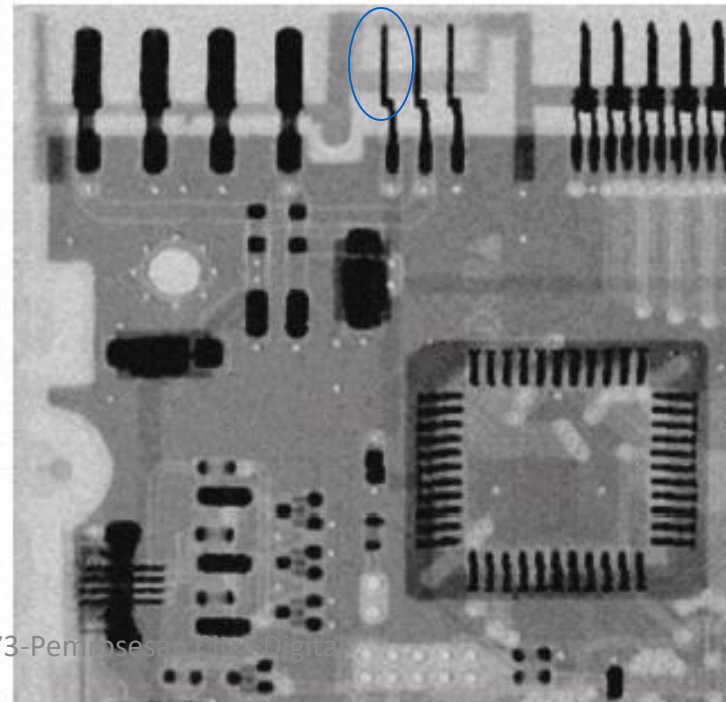
Noisy
Gaussian



Arith.
mean



Geometric
mean



- *Harmonic mean filter*

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

- *Contra-harmonic mean filter*

$$\hat{f}(x, y) = \frac{\sum_{(s,t) \in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t) \in S_{xy}} g(s,t)^Q}$$

Q=-1, harmonic

Q=0, arith. mean

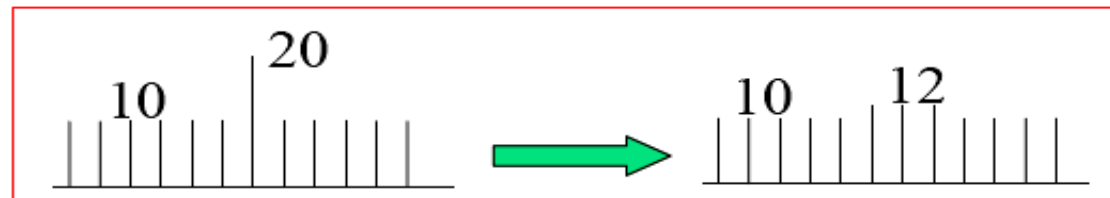
Q=+, ?

- **Mean filters (noise reduced by blurring)**

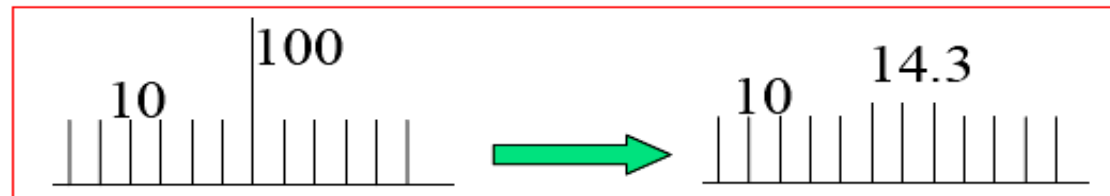
- **Harmonic mean filter**

- ◆ good for Gaussian noise
 - ◆ good for salt noise
 - ◆ bad for pepper noise

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

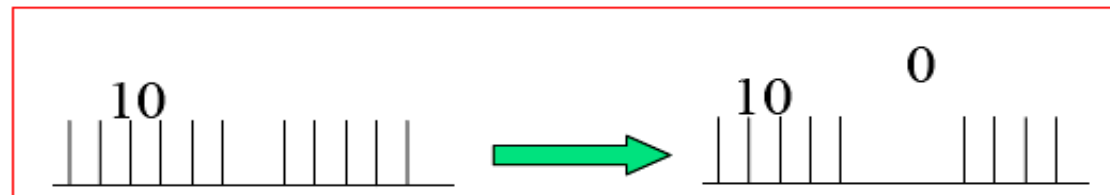


good



good

1 x 3 mask



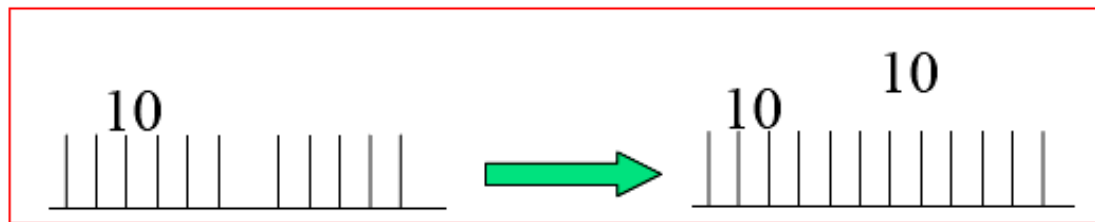
failed

- **Mean filters (noise reduced by blurring)**

- **Contraharmonic mean filter**

- ◆ $Q > 0$: eliminating pepper noise
- ◆ $Q < 0$: eliminating salt noise
- ◆ **cannot do both simultaneously**
- ◆ $Q = 0$: arithmetic mean filter
- ◆ $Q = -1$: harmonic mean filter

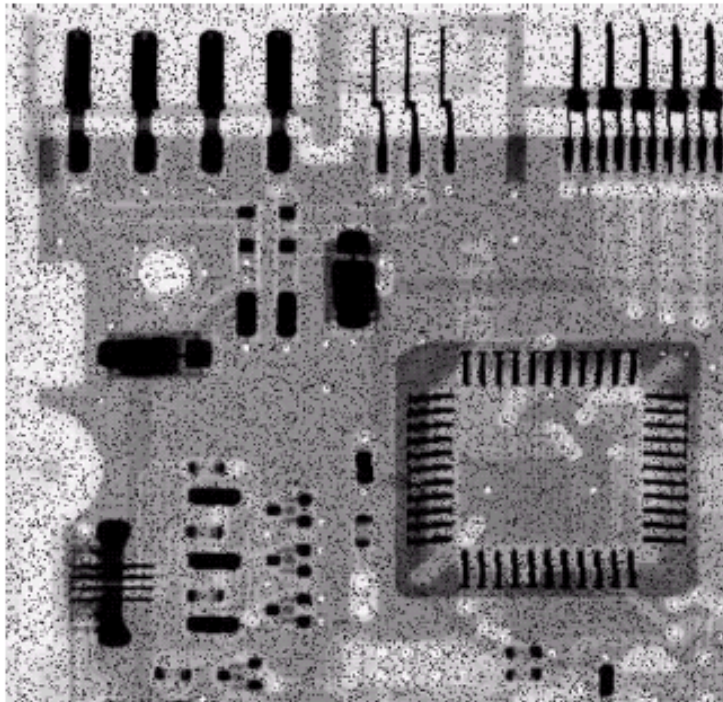
$$\hat{f}(x, y) = \frac{\sum_{(s,t) \in S_{xy}} g(s, t)^{Q+1}}{\sum_{(s,t) \in S_{xy}} g(s, t)^Q}$$



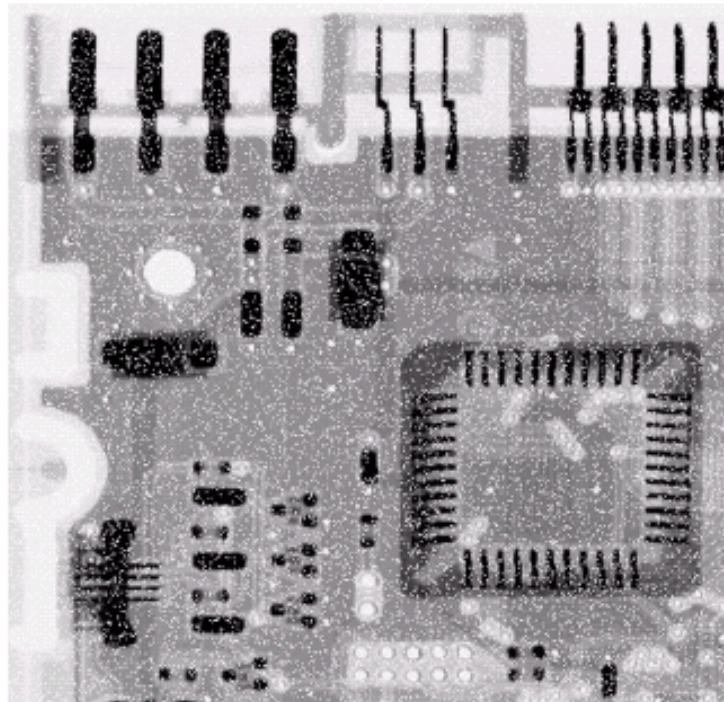
$Q = 1$

1 x 3 mask

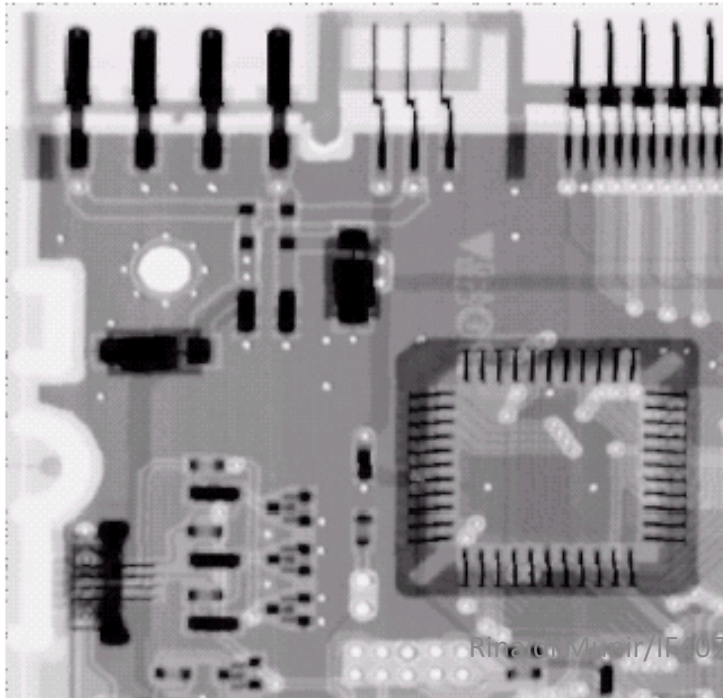
Pepper
Noise



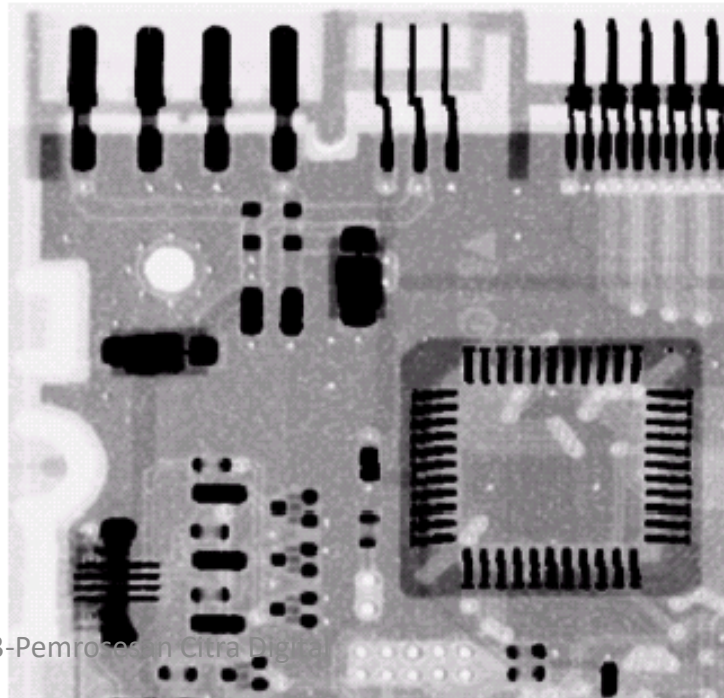
Salt
Noise



Contra-
harmonic
 $Q=1.5$

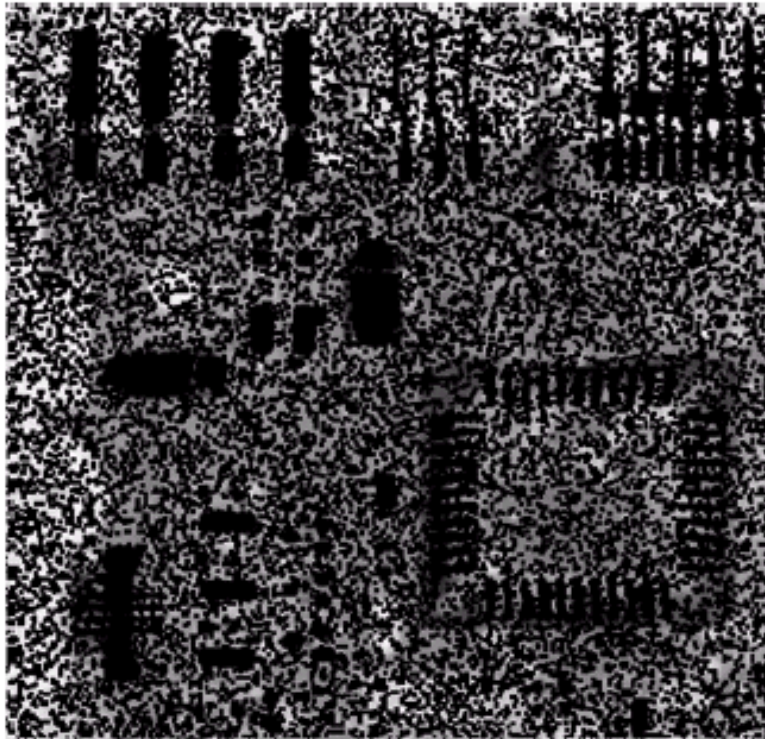


Contra-
harmonic
 $Q=-1.5$

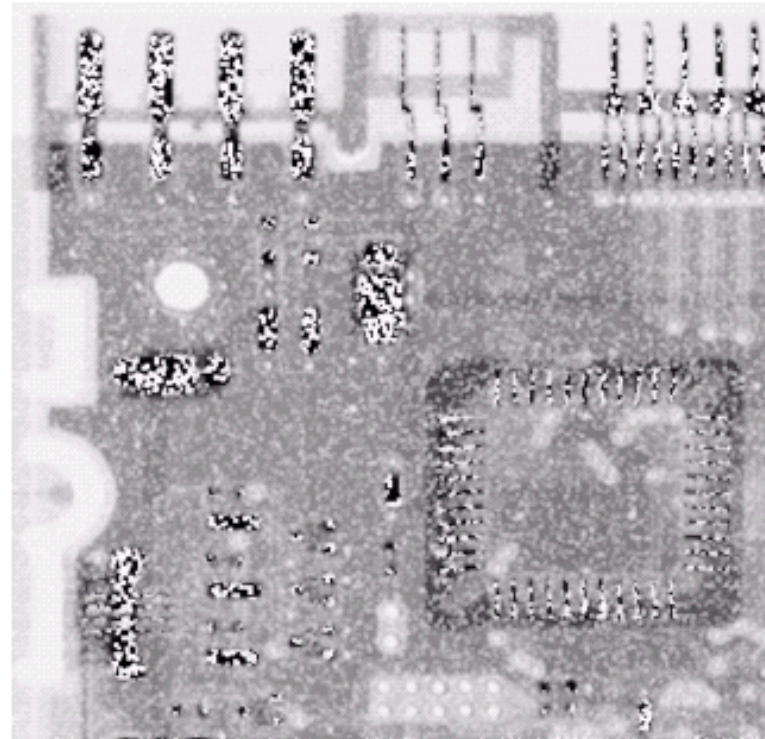


Wrong sign in contra-harmonic filtering

- **Mean filters (noise reduced by blurring)**
 - Arithmetic mean filter and geometric mean filter are well suited for random noise such as Gaussian noise
 - Contraharmonic mean filter is well suited for impulse noise
 - Disadvantage: must know pepper noise or salt noise in advance



$Q=-1.5$



$Q=1.5$

Order-statistics filters

- Didasarkan pada pengurutan (pe-ranking-an) pixel-pixel
 - Cocok untuk derau unipolar atau derau bipolar (salt and pepper noise)
- *Median filter*: mengganti nilai *pixel* di tengah *window* dengan nilai median di dalam *window*
- *Max/min filter*: mengganti nilai *pixel* di tengah *window* dengan nilai maksimum/minimum di dalam *window*.
- *Midpoint filter*: mengganti nilai *pixel* di tengah *window* dengan $(\text{nilai maksimum} + \text{nilai minimum})/2$ di dalam *window*
- *Alpha-trimmed mean filters*: membuang beberapa pixel lalu merata-ratakan sisanya.

Order-statistics filters

- Median filter

$$\hat{f}(x, y) = \underset{(s,t) \in S_{xy}}{\text{median}} \{g(s, t)\}$$

- Max/min filters

$$\hat{f}(x, y) = \underset{(s,t) \in S_{xy}}{\max} \{g(s, t)\}$$

$$\hat{f}(x, y) = \underset{(s,t) \in S_{xy}}{\min} \{g(s, t)\}$$

- **Order-statistics filters**

- **Median filter**

- ◆ handling both bipolar or unipolar impulse noise

$$\hat{f}(x, y) = \underset{(s, t) \in S_{xy}}{\text{median}} \{g(s, t)\}$$

- **Max filter**

- ◆ finding the brightest points in an image
 - ◆ reducing pepper noise

$$\hat{f}(x, y) = \underset{(s, t) \in S_{xy}}{\max} \{g(s, t)\}$$

- **Min filter**

- ◆ finding the darkest points in an image
 - ◆ reducing salt noise

$$\hat{f}(x, y) = \underset{(s, t) \in S_{xy}}{\min} \{g(s, t)\}$$

Contoh penapis median (*median filter*):

13	10	15	14	18
12	10	10	10	15
11	11	35	10	10
13	9	12	10	12
13	12	9	8	10

(a) *Pixel* bernilai 35 terkena derau

13	10	15	14	18
12	10	10	10	15
11	11	10	10	10
13	9	12	10	12
13	12	9	8	10

(b) 35 diganti dengan median dari kelompok 3×3 *pixel*

Misalkan *pixel* di tengah, 35, akan diproses.

Urutkan *pixel-pixel* tersebut:

9 10 10 10 **10** 10 11 12 35

Median dari kelompok tersebut adalah 10 (dicetak tebal, warna biru).

Titik tengah dari jendela (35) sekarang diganti dengan nilai median (10).

```
I = imread('zelda.bmp');  
Inoise = imnoise(I, 'salt & pepper', 0.1);  
figure; imshow(I);  
figure; imshow(Inoise);  
Ifiltered = medfilt2(Inoise, [3 3]);  
figure; imshow(Ifiltered)
```

Penapis Median



Original image



Noisy image

Rinaldi Munir/IF4073-Pemrosesan Citra Digital



Filtered image

```
A = imread('eight.png');  
B = imnoise(A, 'salt & pepper', 0.02); % density = 0.02  
imshow(A), title('original image');  
figure, imshow(B), title('noisy image (salt and peppers)');  
Ifiltered = medfilt2(B, [3 3]);  
figure; imshow(Ifiltered), title('Median filtering');
```

original image



noisy image (salt and peppers)

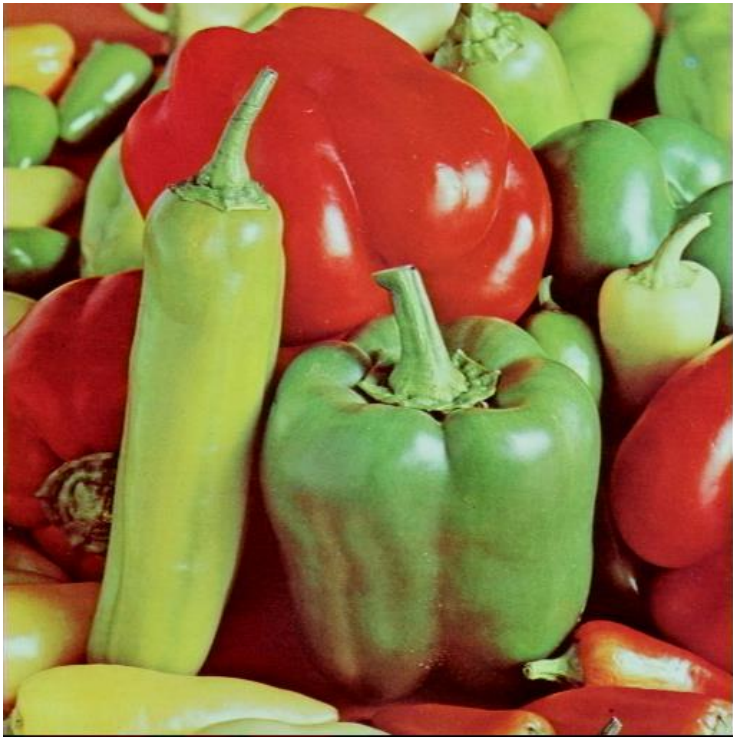


Median filtering



```
I = imread('peppers512.bmp');  
Inoise = imnoise(I, 'salt & pepper', 0.2);  
figure; imshow(I); figure; imshow(Inoise);  
r = Inoise(:, :, 1); g = Inoise(:, :, 2); b = Inoise(:, :, 3);  
Ifiltered_r = medfilt2(r, [3 3]);  
Ifiltered_g = medfilt2(g, [3 3]);  
Ifiltered_b = medfilt2(b, [3 3]);  
Ifiltered = cat(3, Ifiltered_r, Ifiltered_g, Ifiltered_b);  
figure; imshow(Ifiltered)
```

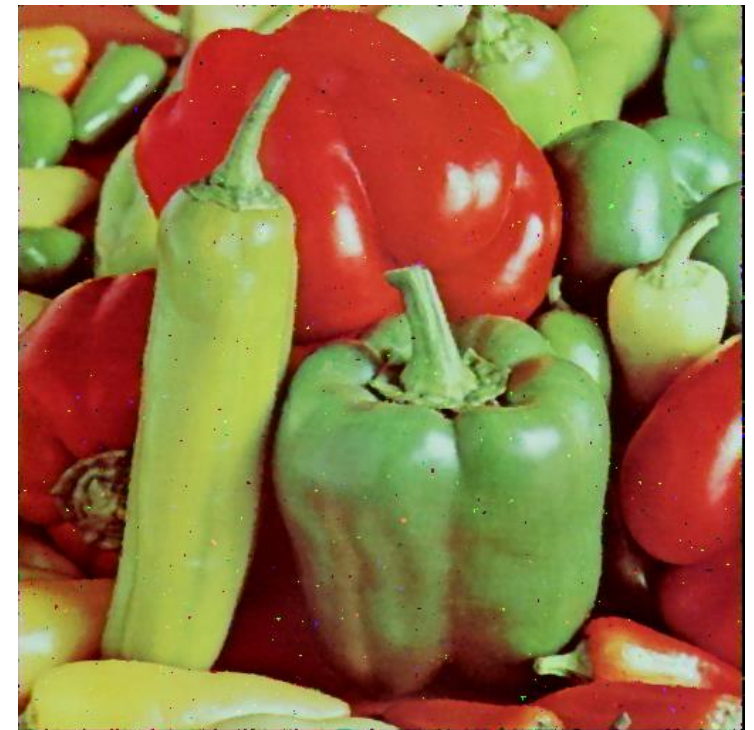
Penapis Median



Original image

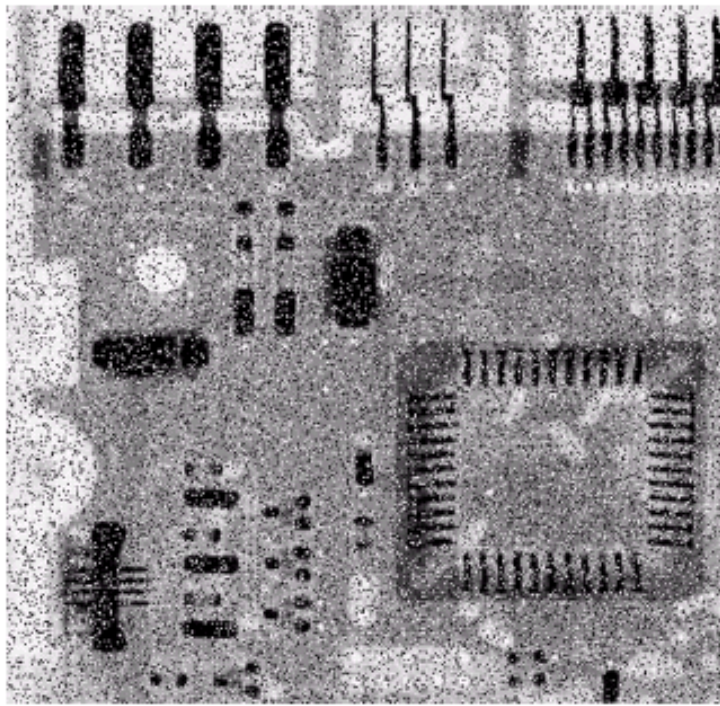


Noisy image

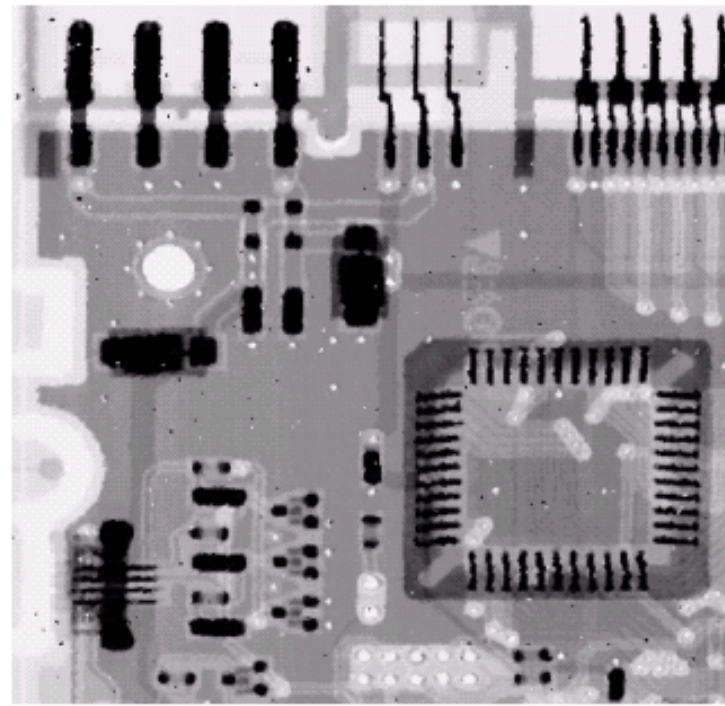


Filtered image

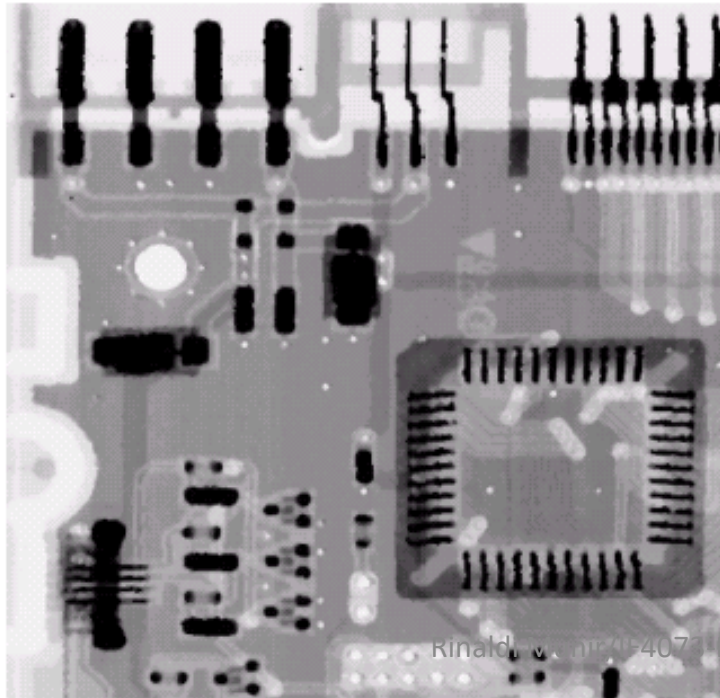
bipolar
Noise
 $P_a = 0.1$
 $P_b = 0.1$



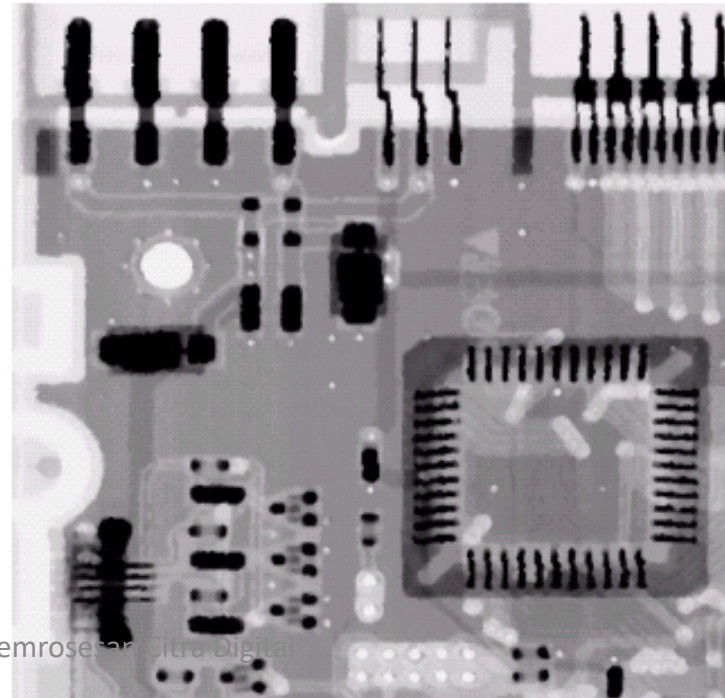
3x3
Median
Filter
Pass 1



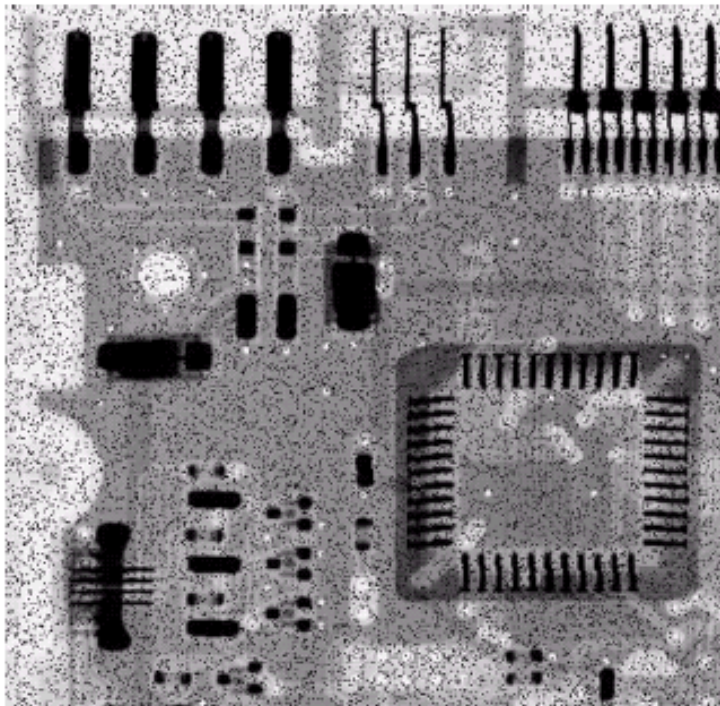
3x3
Median
Filter
Pass 2



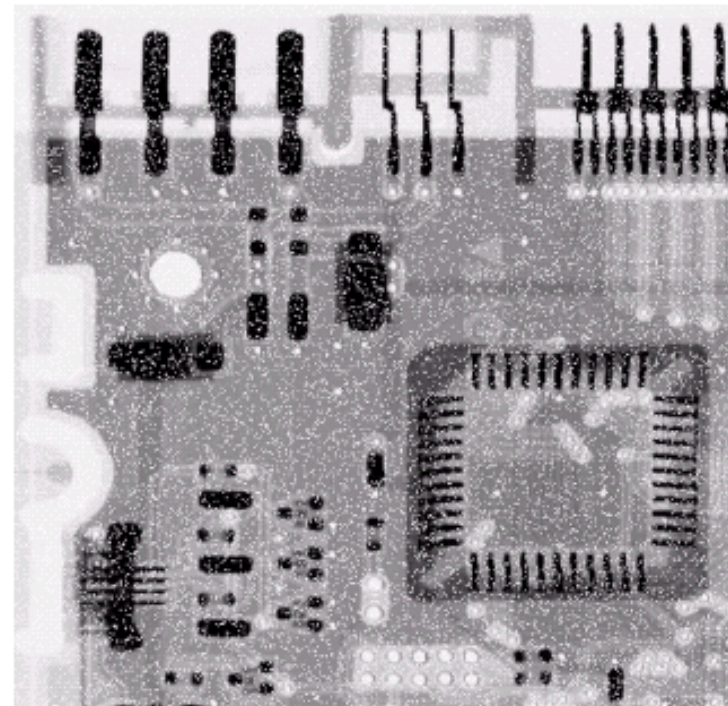
3x3
Median
Filter
Pass 3



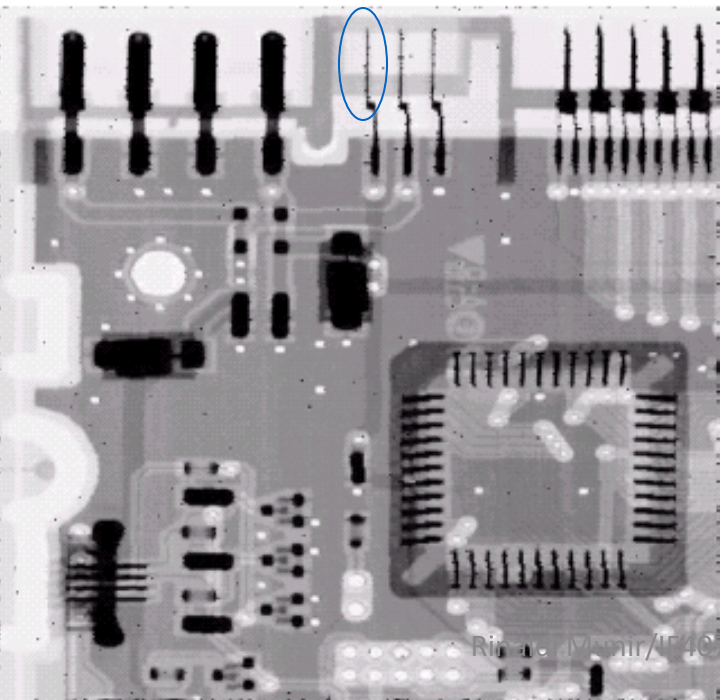
Pepper
noise



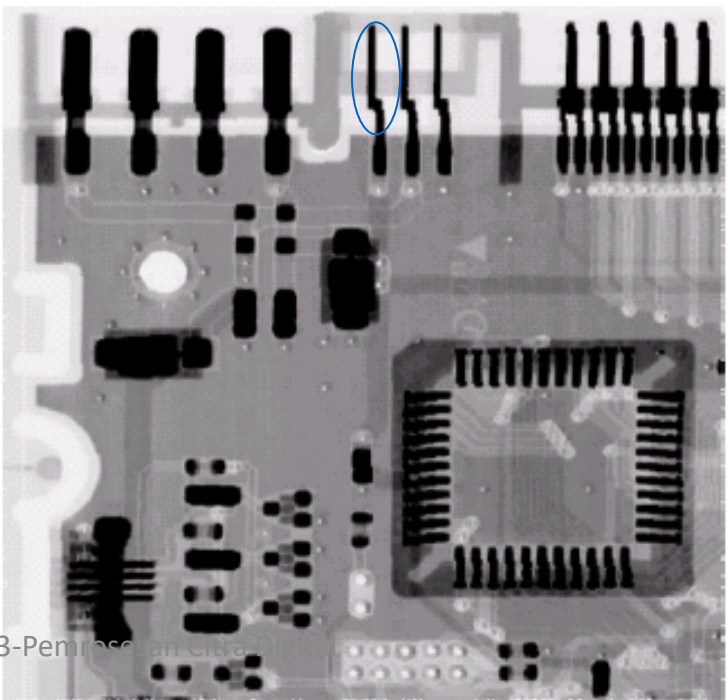
Salt
noise



Max
filter



Min
filter



Order-statistics filters (cont.)

- Midpoint filter

$$\hat{f}(x, y) = \frac{1}{2} \left[\max_{(s,t) \in S_{xy}} \{g(s, t)\} + \min_{(s,t) \in S_{xy}} \{g(s, t)\} \right]$$

- Alpha-trimmed mean filter

- Hapus $d/2$ pixel dengan nilai keabuan terendah dan hapus $d/2$ pixel dengan nilai keabuan tertinggi, lalu rata-ratakan sisanya

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

← Middle ($mn-d$) pixels

- **Order-statistics filters**

- **Midpoint filter**

- ◆ order statistics
 - ◆ averaging
 - ◆ work well for Gaussian noise

$$\hat{f}(x, y) = \frac{1}{2} \left[\max_{(s,t) \in S_{xy}} \{g(s,t)\} + \min_{(s,t) \in S_{xy}} \{g(s,t)\} \right]$$

- **Alpha-trimmed mean filter**

- ◆ $d = 0$: arithmetic mean filter
 - ◆ $d = mn - 1$: median filter
 - ◆ suitable for the situation involving multiple types of noise

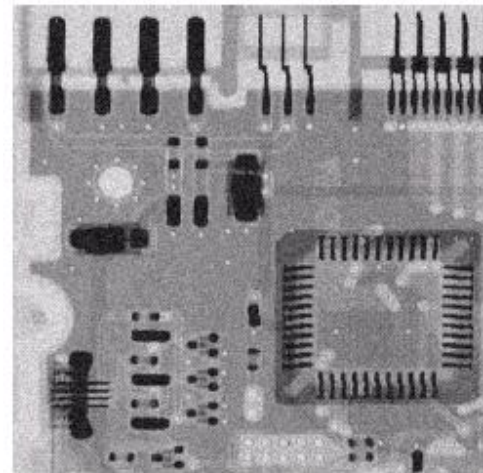
$$\hat{f}(x, y) = \frac{1}{mn - d} \sum_{(s,t) \in S_{xy}} g_r(s, t)$$

delete $d/2$ lowest and
 $d/2$ highest values first,
then average the remaining

Uniform noise

$$\mu=0$$

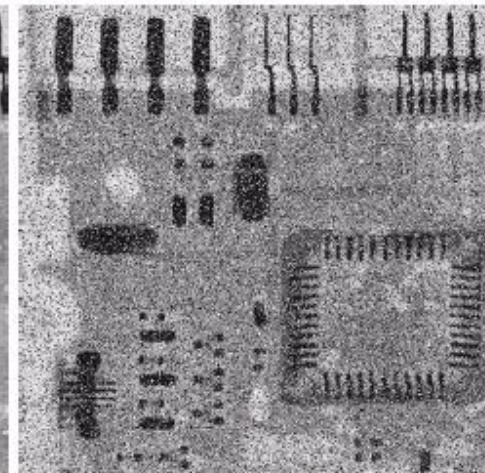
$$\sigma^2=800$$



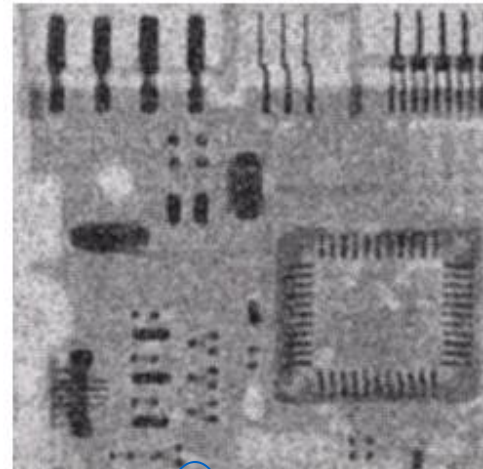
Left +
Bipolar Noise

$$P_a = 0.1$$

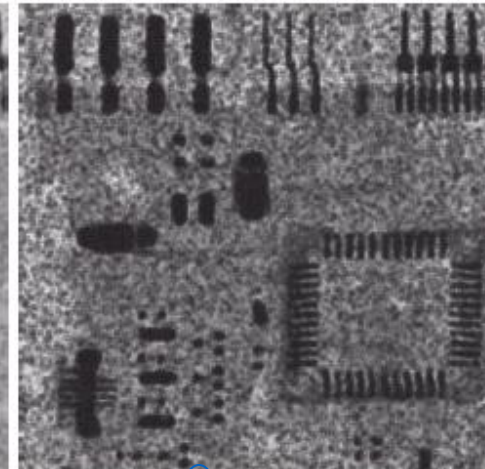
$$P_b = 0.1$$



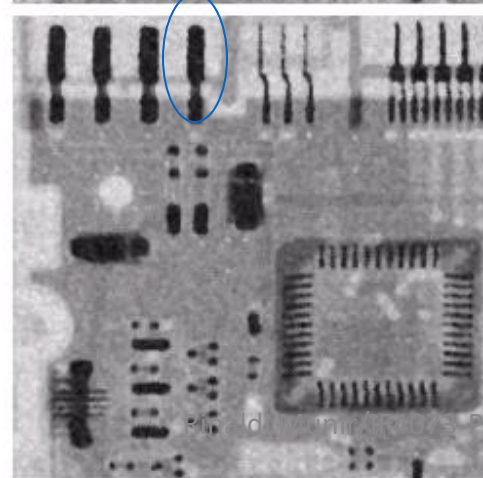
5x5
Arith. Mean
filter



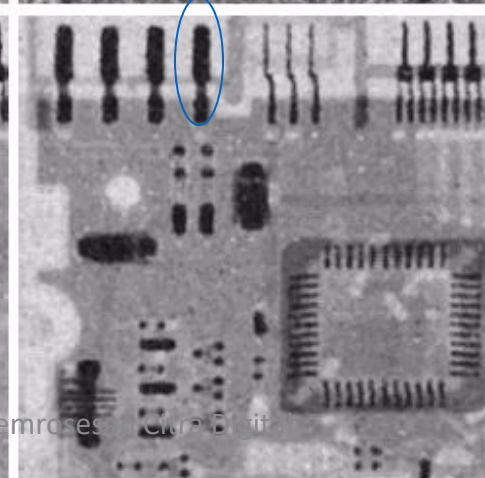
5x5
Geometric
mean



5x5
Median
filter



5x5
Alpha-trim.
Filter
 $d=5$



Bersambung ke Bagian 2