

14 – Restorasi Citra (Bagian 2)

IF4073 Pemrosesan Citra Digital

Oleh: Dr. Ir. Rinaldi, M.T



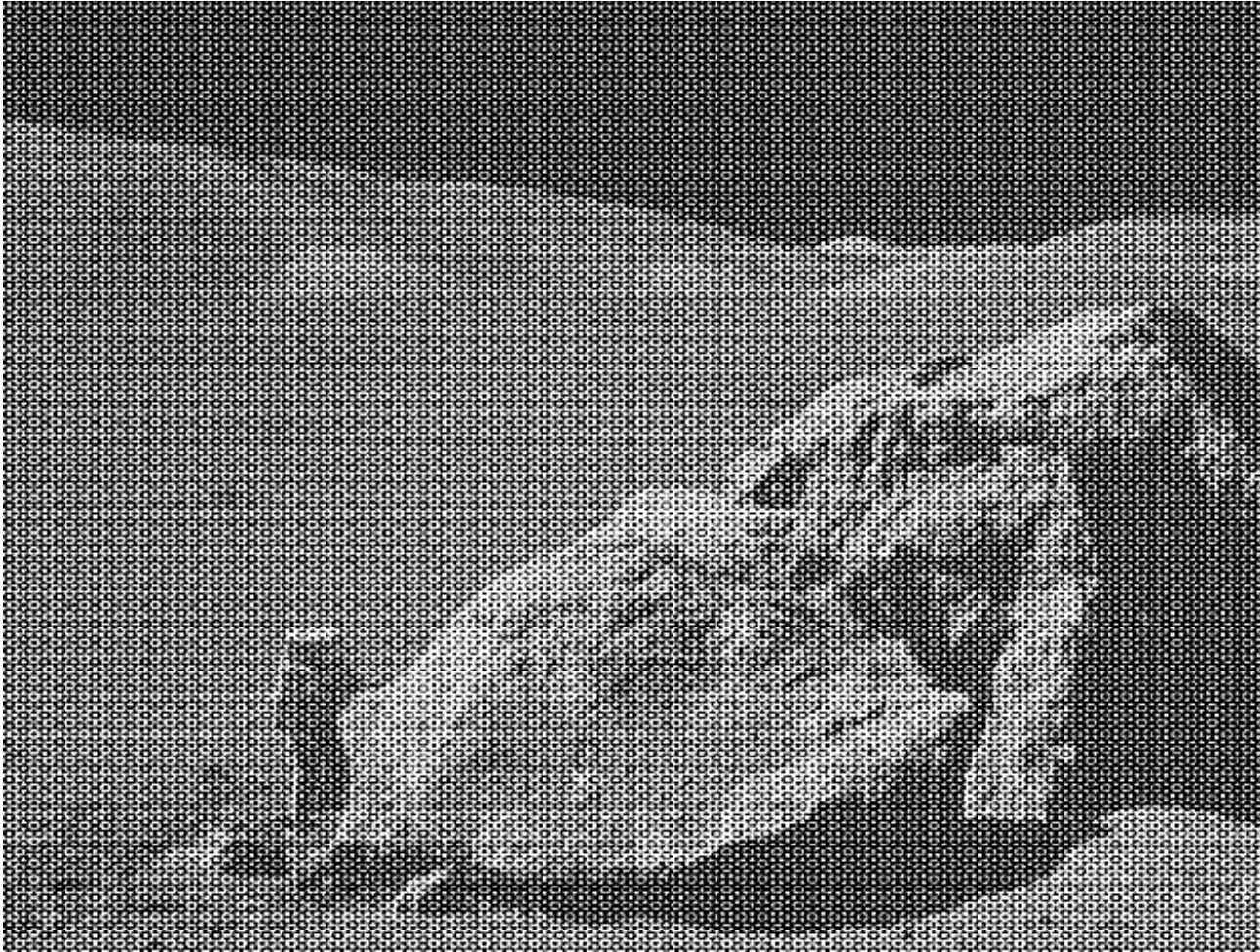
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Update 2025

Restorasi citra dalam ranah frekuensi

- Restorasi citra dalam ranah frekuensi digunakan untuk mengestimasi citra yang terdistorsi akibat:
 - derau periodik
 - motion blur

Derau Periodik





Citra dengan derau periodik lainnya



blurred image

Derau Periodik

- Derau periodik dihasilkan dari interferensi elektrik atau elektomekanik selama proses akuisisi citra
- Derau periodik akan tampak seperti pola yang berulang di dalam citra.
- Derau periodik dapat dihampiri dengan gelombang sinusoida berbentuk:

$$\left\{ \begin{array}{l} f(x, y) = A \sin(u_0 x + v_0 y) \\ F(u, v) = -j \frac{A}{2} \left[\delta\left(u - \frac{u_0}{2\pi}, v - \frac{v_0}{2\pi}\right) - \delta\left(u + \frac{u_0}{2\pi}, v + \frac{v_0}{2\pi}\right) \right] \end{array} \right.$$

- Program Matlab untuk menambahkan derau periodik ke dalam citra

```
f = imread('cameraman.bmp');  
s=size(f);  
[x,y] = meshgrid(1:s(1),1:s(2));  
p = 1+sin(x+y/1.5);  
f_pn = (im2double(f)+p/2)/2;  
imshow(f_pn), title ('Noisy image');  
imwrite(f_pn, 'cameraman_noise2.bmp');
```

Noisy image



Reduksi Derau Periodik

Menggunakan beberapa macam penapis:

- *Bandreject filters*
- *Bandpass filters*
- *Notch filters*

Bandreject Filters

- *Bandreject filter* berguna bila lokasi umum derau dalam ranah frekuensi diketahui (hasil estimasi).
- *Bandraject filter* akan memblokir frekuensi dalam rentang yang dipilih dan membiarkan frekuensi di luar rentang yang melewatinya.

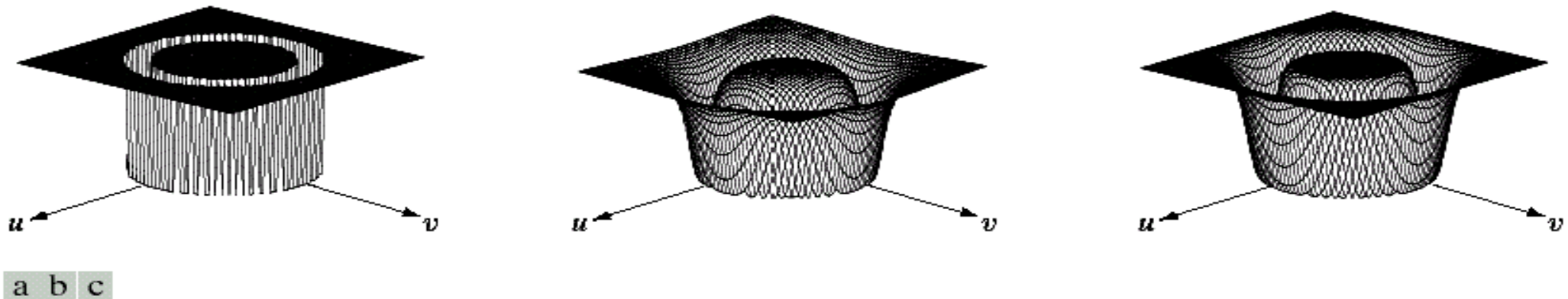


FIGURE 5.15 From left to right, perspective plots of ideal, Butterworth (of order 1), and Gaussian bandreject filters.

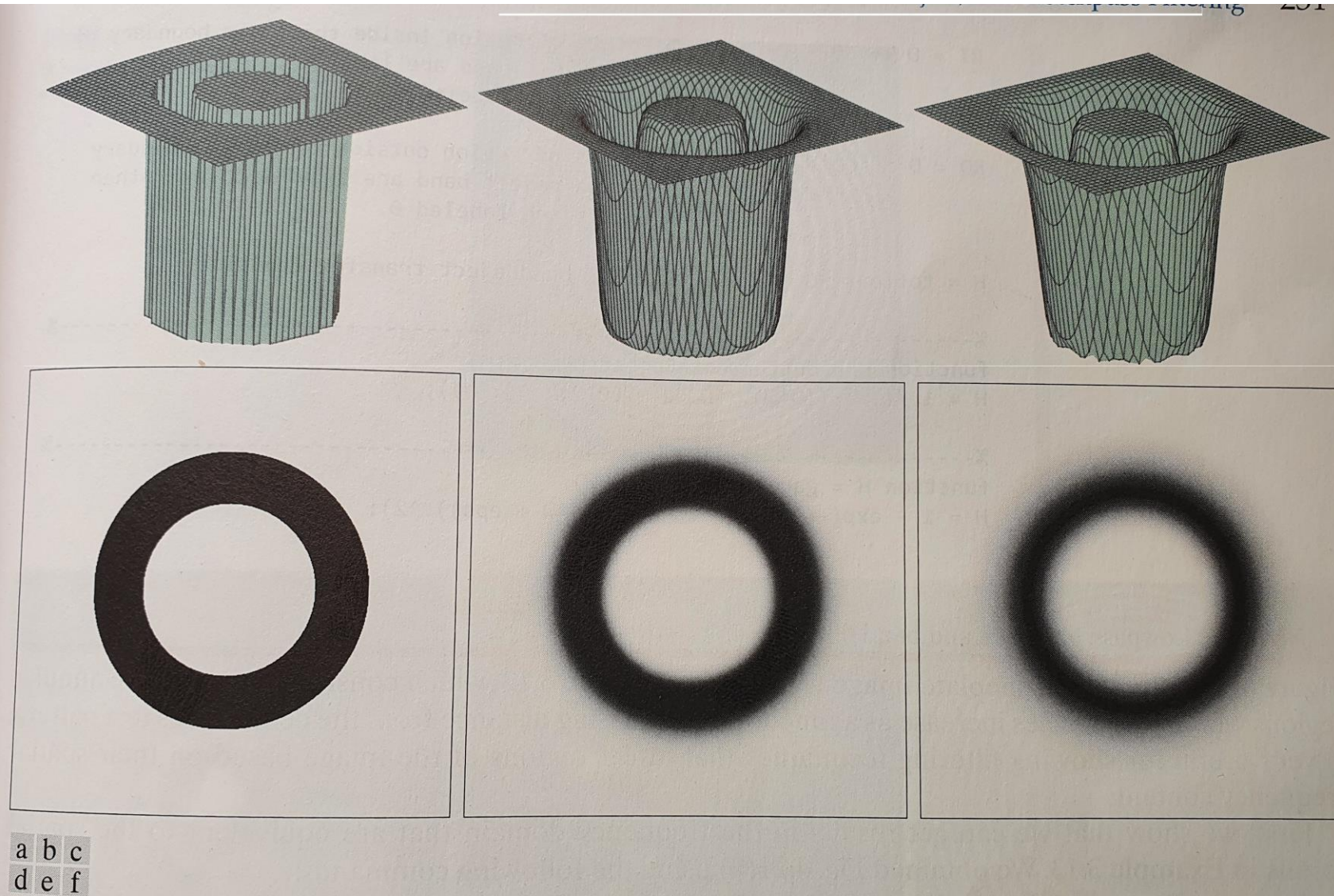
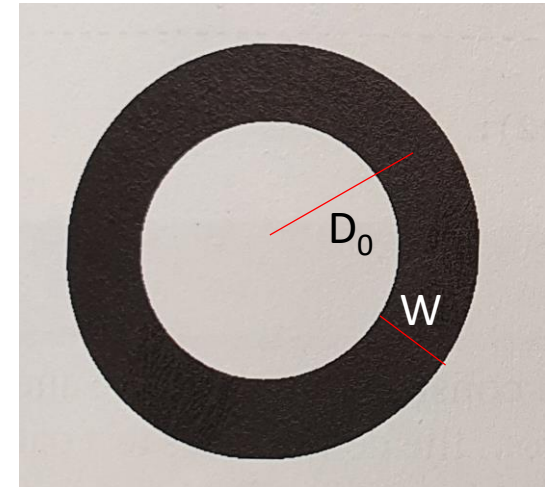


FIGURE 4.19 Bandreject transfer functions of (a) ideal, (b) Butterworth, and (c) Gaussian frequency domain filters. Bottom row: Transfer functions displayed as images. The parameters used to generate the transfer functions were $M = N = 512$, $C_0 = 128$, and $W = 60$. It required $n = 3$ for the Butterworth function to have a response in between the ideal and Gaussian transfer functions.

- **Ideal Bandreject Filter**

Frekuensi di luar rentang yang diberikan dilewatkan tanpa redaman dan frekuensi di dalam rentang yang diberikan diblokir. Perilaku ini membuat *ideal bandreject filter* menjadi sangat tajam.

$$\begin{aligned} H(u,v) &= 1 && \text{if } D(u,v) < D_0 - \frac{W}{2} \\ &= 0 && \text{if } D_0 - \frac{W}{2} \leq D(u,v) \leq D_0 + \frac{W}{2} \\ &= 1 && \text{if } D(u,v) > D_0 + \frac{W}{2} \end{aligned}$$



$D(u,v)$ = jarak dari titik pusat persegi panjang frekuensi ($M/2, N/2$)
 $= [(u - M/2)^2 + (v - N/2)^2]^{1/2}$

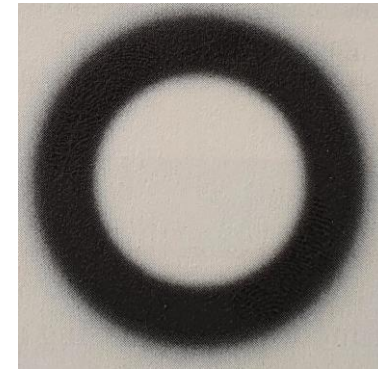
W = lebar pita

D_0 = jari-jari pita (di dalam buku Gonzalez ditulis C_0)

- **Butterworth Bandreject Filter orde n**

Dengan *Butterworth bandreject filter*, frekuensi di tengah pita sepenuhnya diblokir dan frekuensi di tepi pita dilemahkan. Filter *Butterworth* tidak memiliki diskontinuitas yang tajam antara frekuensi yang dilewatkan dan frekuensi yang difilter.

$$H(u, v) = \frac{1}{1 + \left[\frac{D(u, v)W}{D^2(u, v) - D_0^2} \right]^{2n}}$$

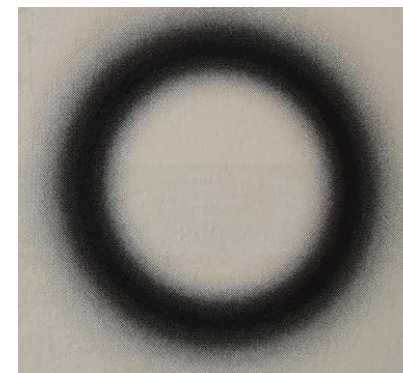


Catatan: Butterworth orde n rendah mendekati Gaussian, dan Butterworth orde tinggi mendekati Ideal.

- **Gaussian Band-Reject Filter**

Dengan jenis filter ini, transisi antara frekuensi tidak terfilter dan terfilter sangat mulus.

$$H(u, v) = 1 - e^{-\frac{1}{2} \left[\frac{D^2(u, v) - D_0^2}{D(u, v)W} \right]^2}$$



- Citra hasil rekonstruksi, G , diperoleh dengan mengalikan citra yang mengalami derau (dalam ranah frekuensi), F , dengan H :

$$G(u,v) = F(u,v)H(u,v)$$

- Selanjutnya lakukan *inverse Fourier Transform* untuk memperoleh citra $g(x,y)$ dalam ranah spasial

Contoh:



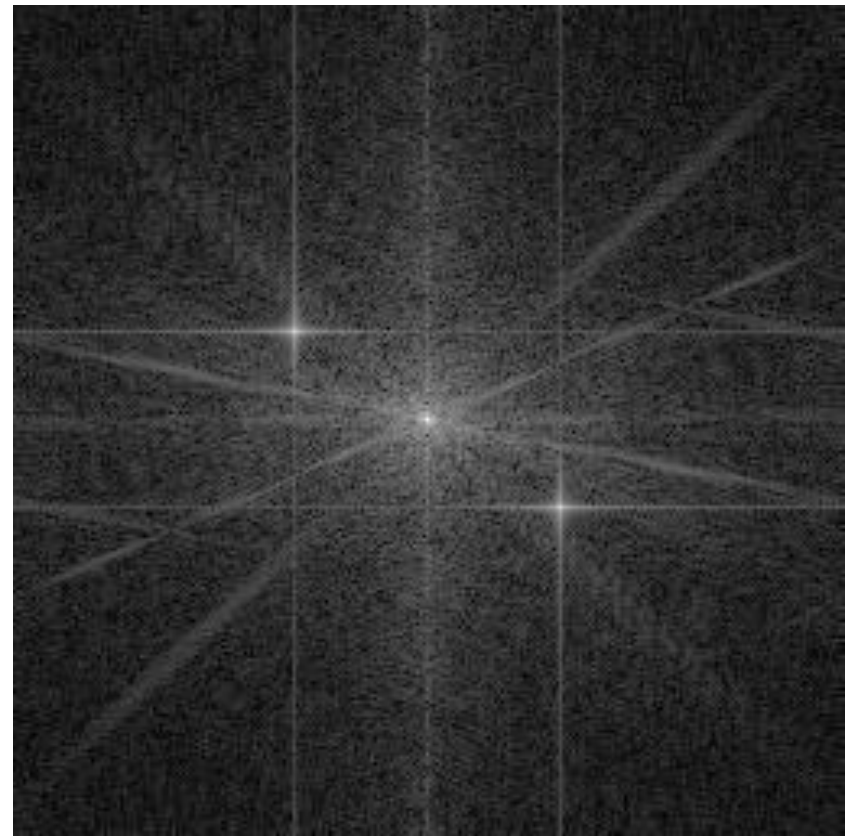
Original image

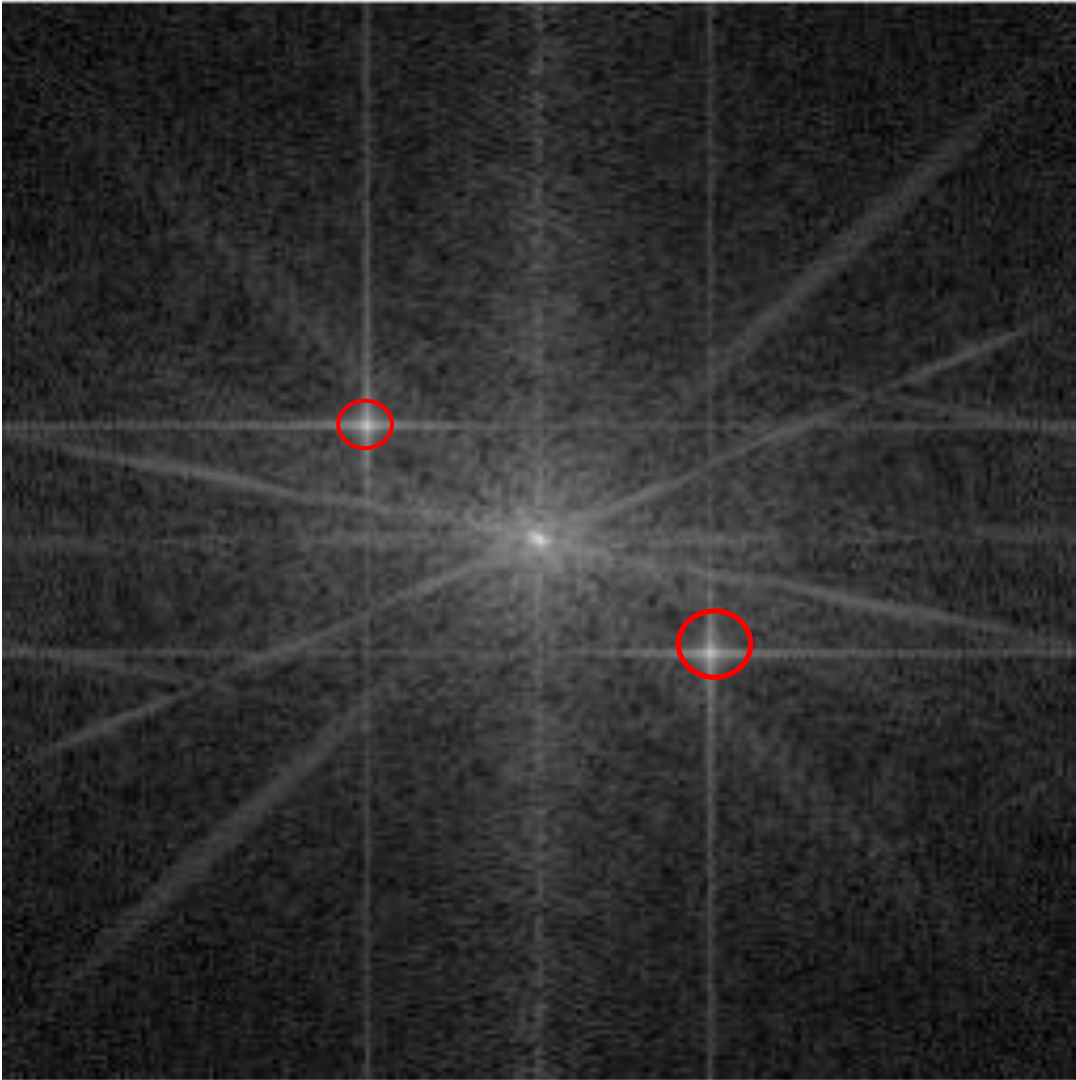


Citra cameraman dengan derau periodik

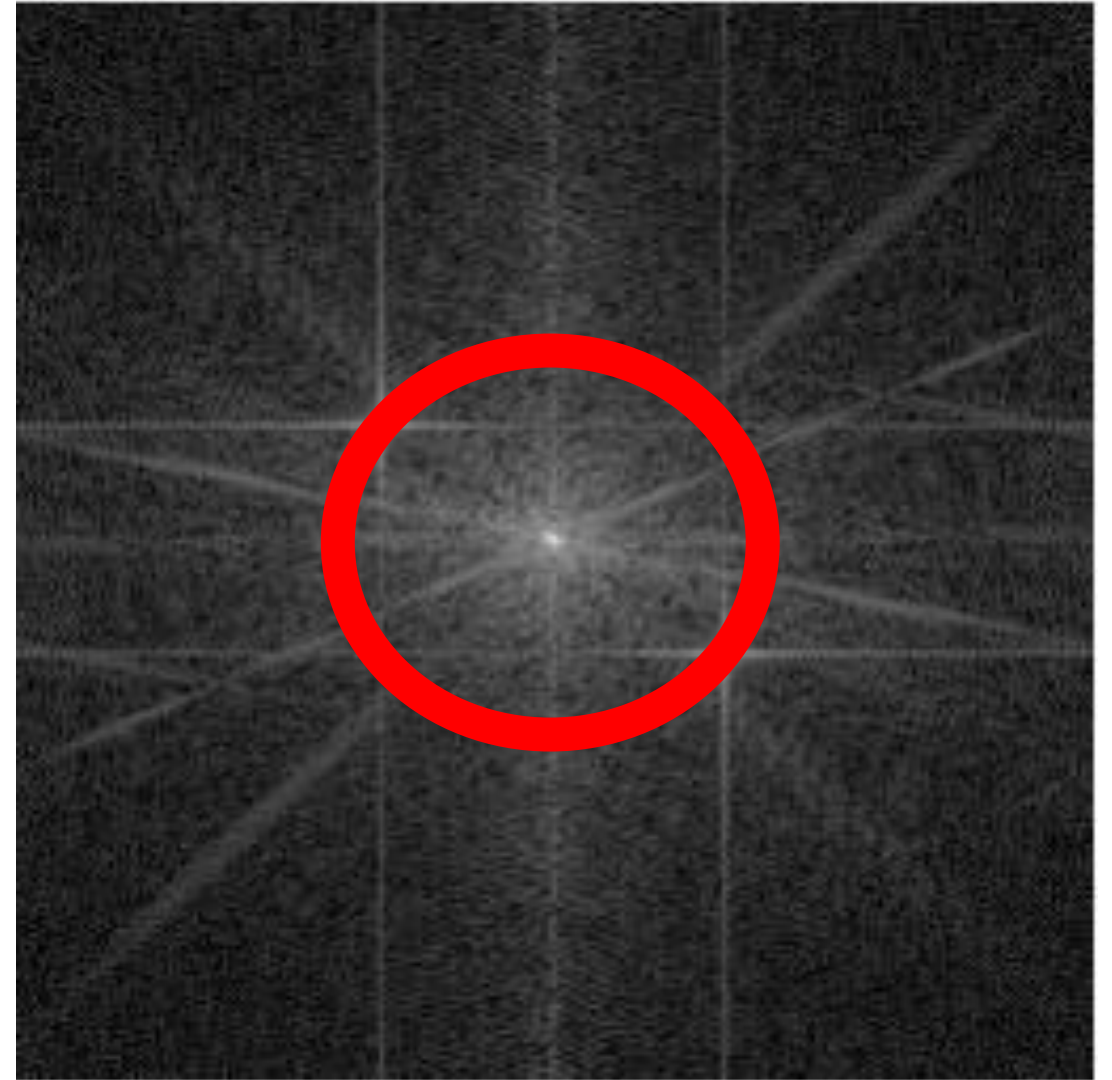


Fourier spectrum





Titik-titik *spikes* pada posisi
(156, 170) dan (102, 88),
jarak ke titik pusat = 49



Jari-jari lingkaran band filter = 49
($C0 = 49$, $W = 5$)

Program Matlab

```
% Bandreject filtering
f=imread('cameraman_noise2.bmp');
imshow(f);
[M,N] = size(f);
imshow(f);title('original image');

% Lakukan transformasi Fourier pada f(x, y)
% Display the Fourier Spectrum
f = im2double(f);
F = fft2(f);
F = fftshift(F); % move the origin of the transform to the center of the
frequency rectangle
S2 = log(1+abs(F)); % use abs to compute the magnitude (handling imaginary)
and use log to brighten display
figure, imshow(S2, []); title('Fourier spectrum');
```

```

%Bangkitkan fungsi penapis H berukuran M x N, misalkan penapis
%yang digunakan adalah Butterworth Bandreject Filter orde n = 1

% Set up range of variables.
u = 0:(M-1);
v = 0:(N-1);

% Compute the indices for use in meshgrid
idx = find(u > M/2);
u(idx) = u(idx) - M;
idy = find(v > N/2);
v(idy) = v(idy) - N;

% Compute the meshgrid arrays
[V, U] = meshgrid(v, u);
D = sqrt(U.^2 + V.^2);

D0 = 49;
W = 5;
n = 1; H = 1./(1 + ((D*W)./(D.^2 - D0^2)).^(2*n));
H = fftshift(H);
figure;imshow(H);title('Butterworth Bandreject Filter orde n = 1');

```

```
%Kalikan F dengan H
```

```
G = H.*F;
```

```
%Ambil bagian real dari inverse FFT of G:
```

```
G = real(ifft2(G)); % apply the inverse, discrete Fourier transform
```

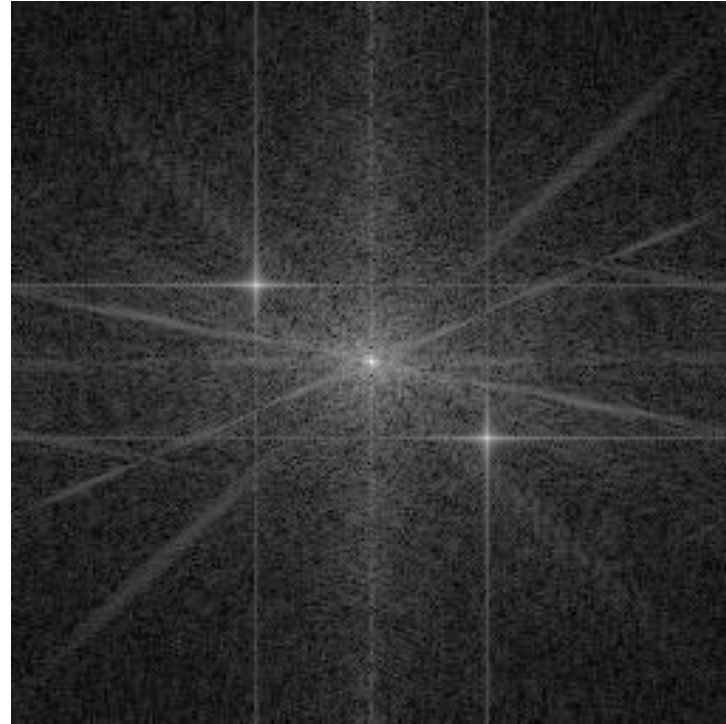
```
figure; imshow(G);title('Output image');
```

Hasil run program

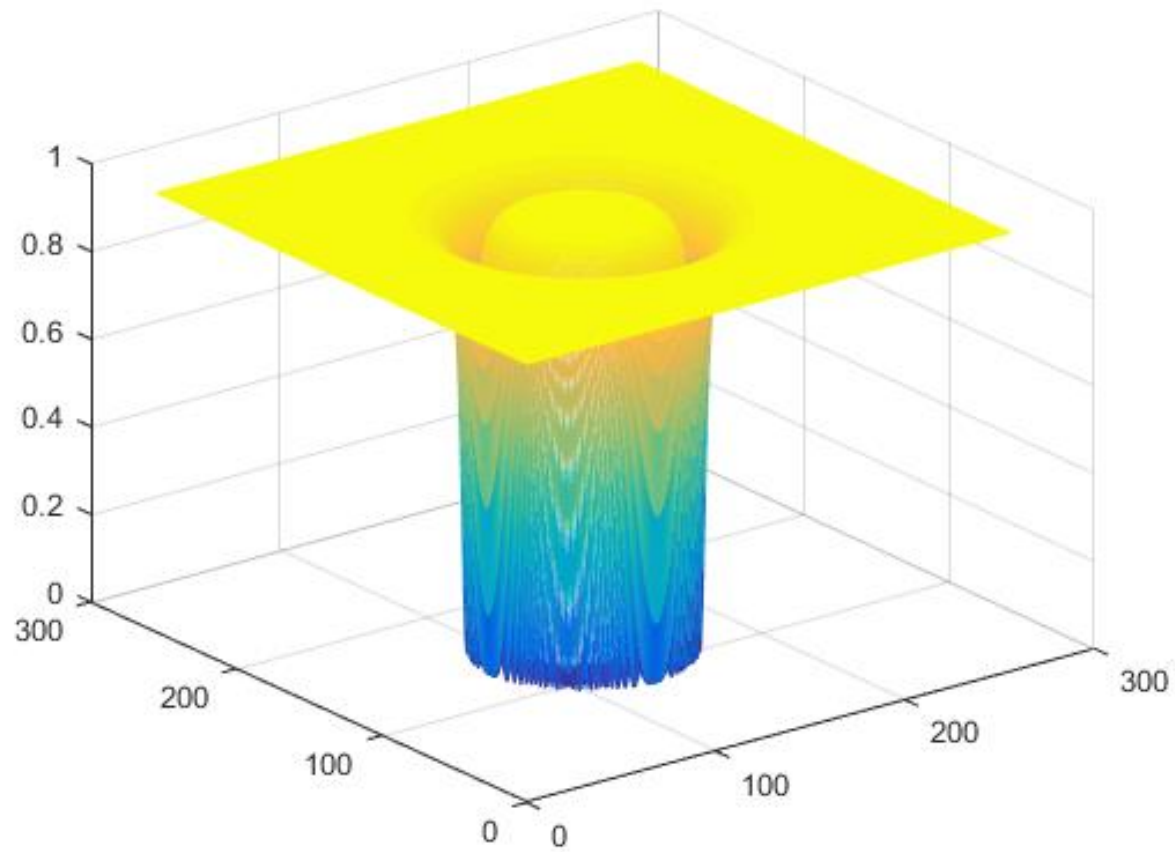
Noisy image



Fourier spectrum

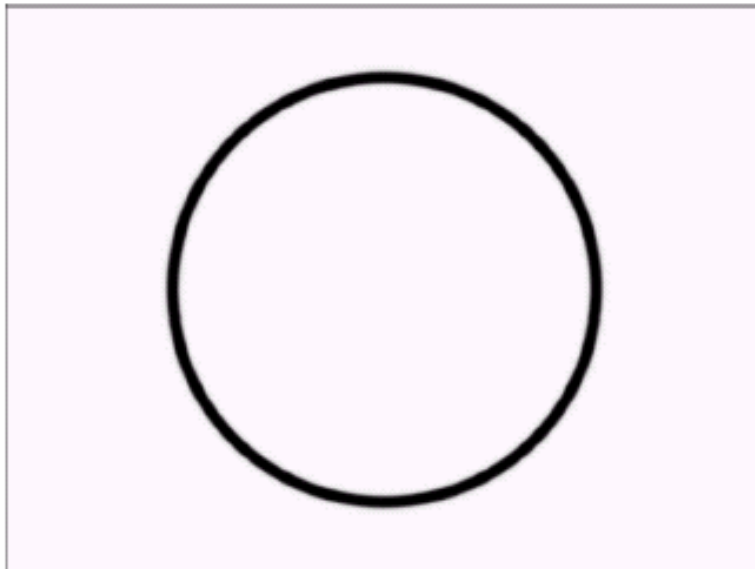
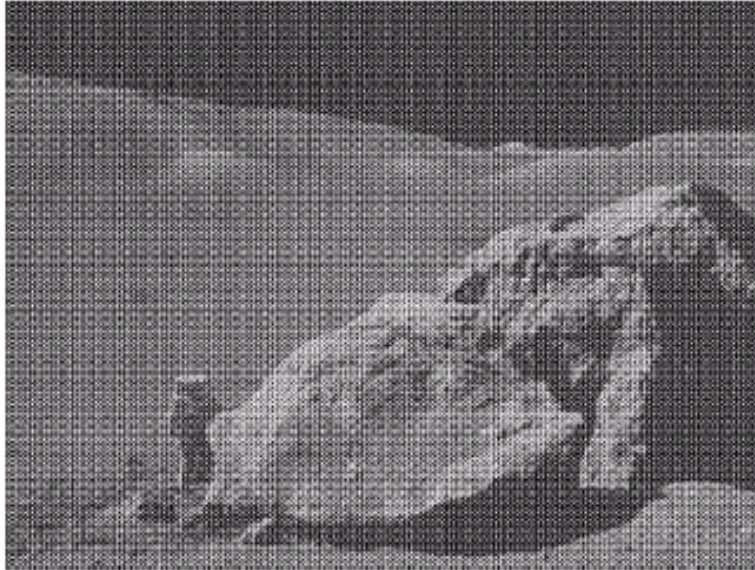


Butterworth Bandreject Filter orde $n = 1$



Output image





a	b
c	d

FIGURE 5.16

(a) Image corrupted by sinusoidal noise. (b) Spectrum of (a). (c) Butterworth bandreject filter (white represents 1). (d) Result of filtering. (Original image courtesy of NASA.)

Contoh lainnya:

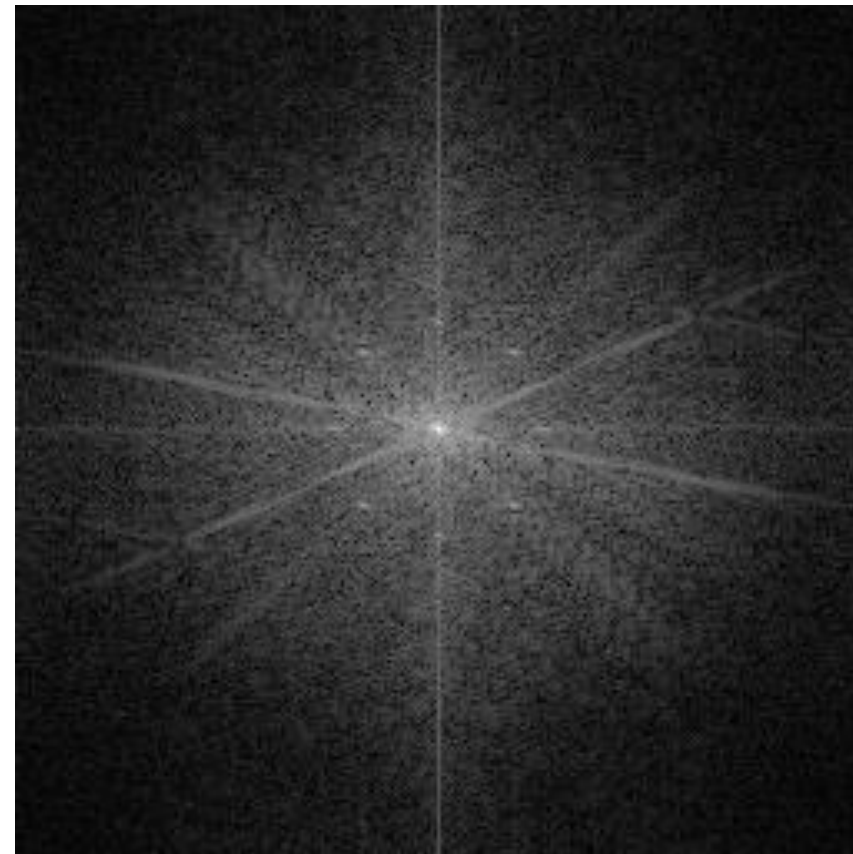


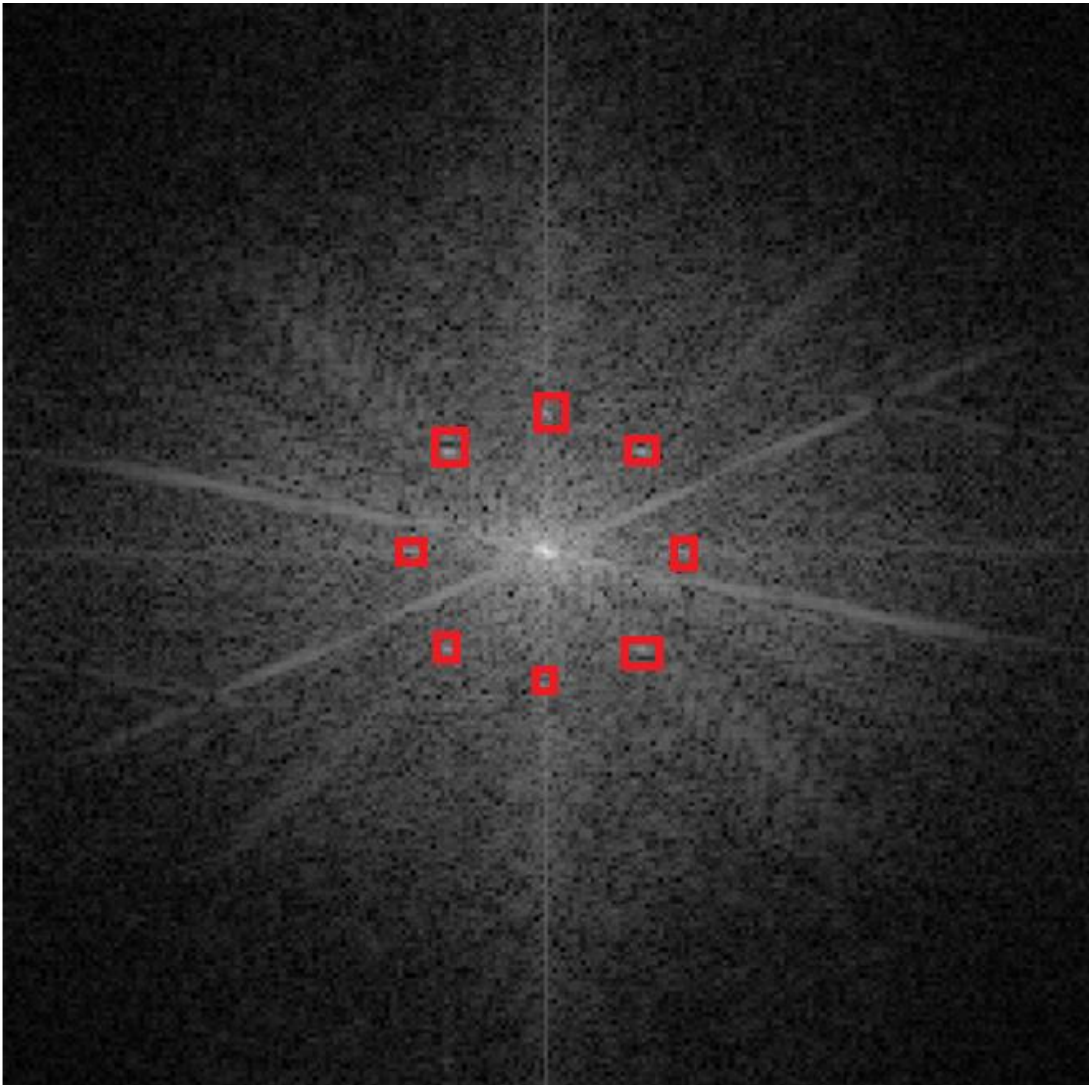
Original image



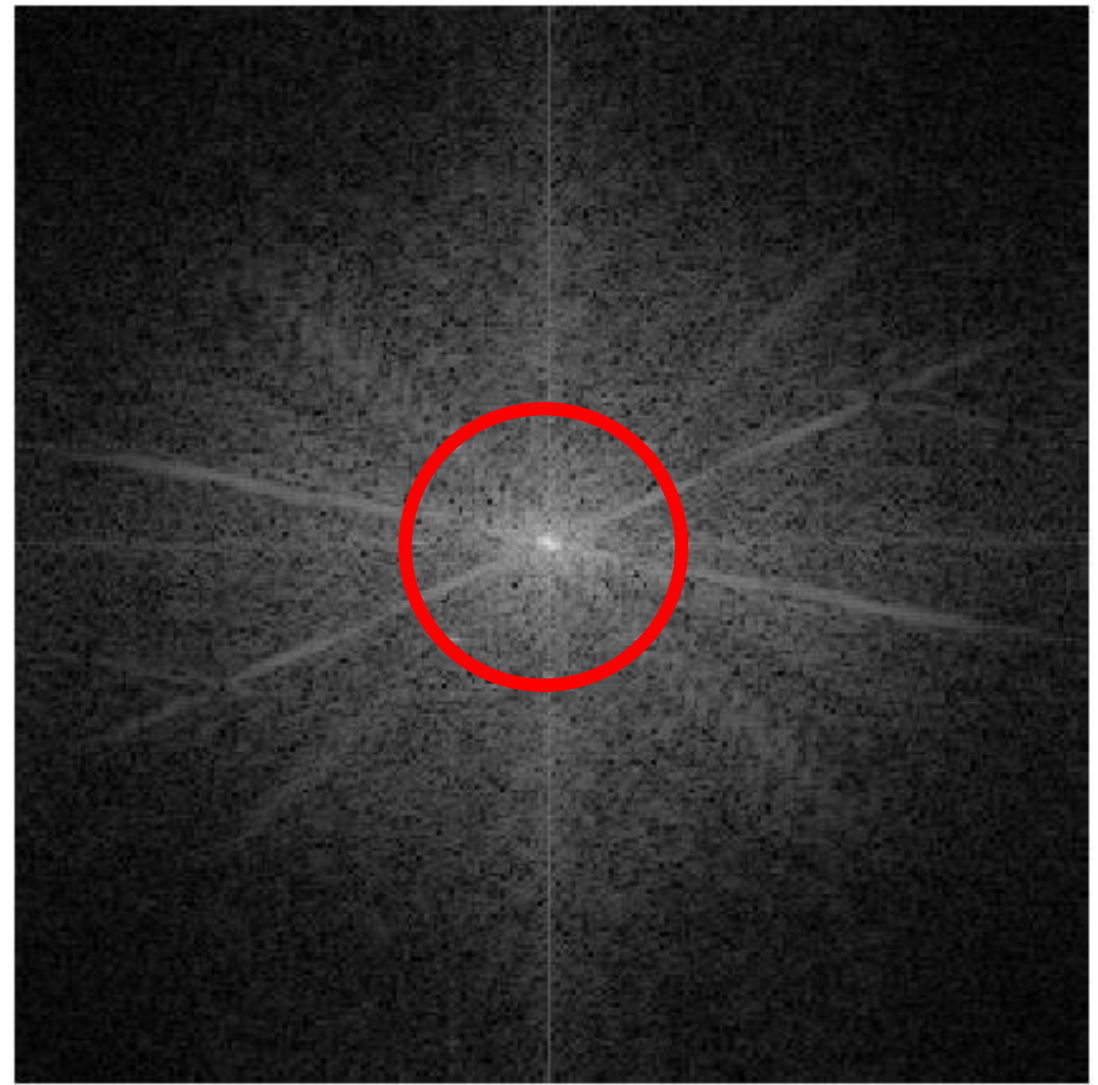
Citra cameraman dengan derau periodik

Fourier spectrum





Titik-titik *spikes*



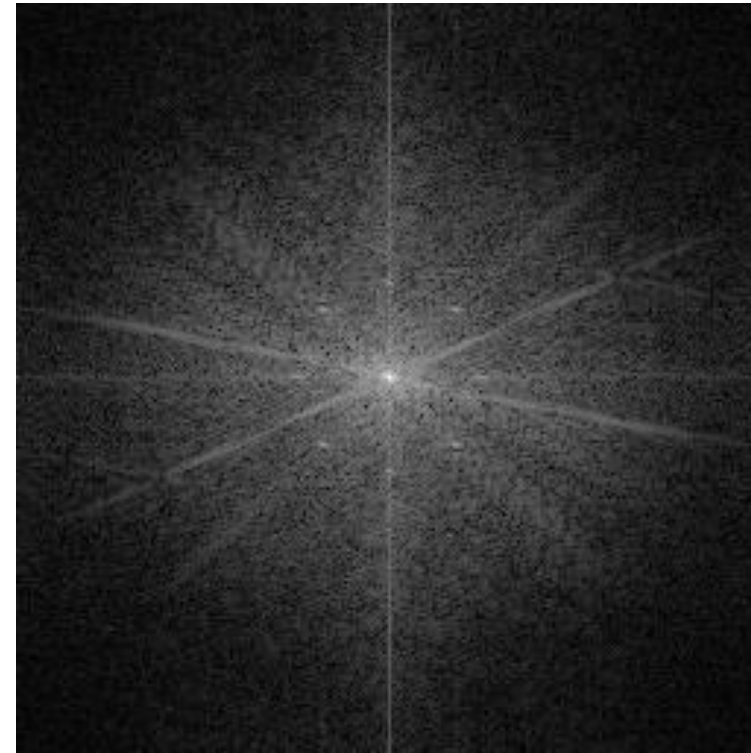
Perkiraan jari-jari lingkaran band filter = 32
($C0 = 32$, $W = 6$)

Hasil run program

original image



Fourier spectrum



Butterworth Bandreject Filter orde $n = 1$



Output image

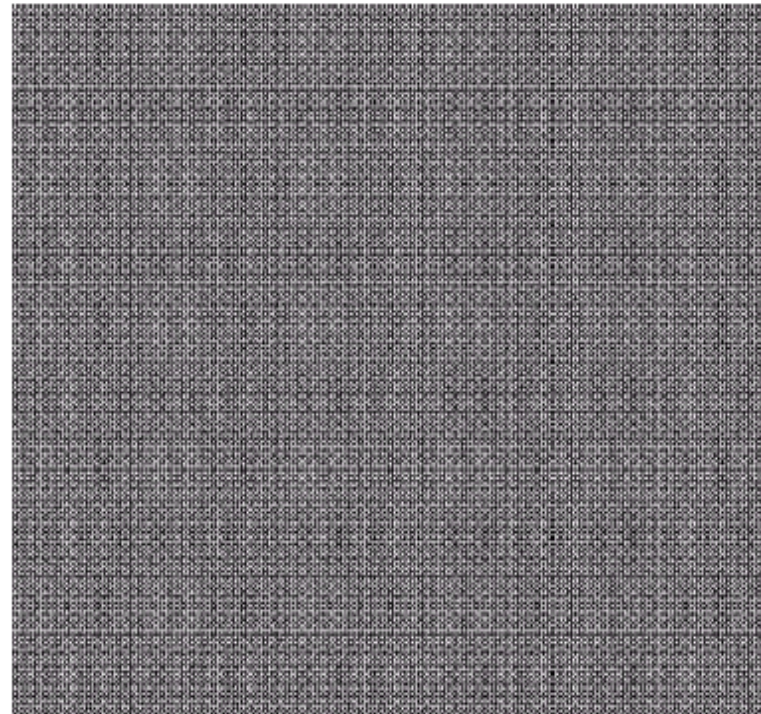


Bandpass Filter

- Band-Pass Filters
 - Operasinya berlawanan dengan *band-reject filter*

$$H_{bp} = 1 - H_{br}(u, v)$$

FIGURE 5.17
Noise pattern of
the image in
Fig. 5.16(a)
obtained by
bandpass filtering.



Contoh kode program dengan *bandpass filter*:

```
function filtered_image = gaussianbpf(I,d0,d1)
% Butterworth Bandpass Filter
% This simple function was written for my Digital Image Processing course
% at Eastern Mediterranean University taught by
% Assoc. Prof. Dr. Hasan Demirel
% for the 2010-2011 Spring Semester
% for the complete report:
% http://www.scribd.com/doc/51981950/HW4-Frequency-Domain-Bandpass-Filtering
%
% Written By:
% Leonardo O. Iheme (leonardo.iheme@cc.emu.edu.tr)
% 24th of March 2011
%
% I = The input grey scale image
% d0 = Lower cut off frequency
% d1 = Higher cut off frequency
%
% The function makes use of the simple principle that a bandpass filter
% can be obtained by multiplying a lowpass filter with a highpass filter
% where the lowpass filter has a higher cut off frequency than the high pass
% filter.
```

```

%
% Usage GAUSSIANBPF(I,DO,D1)
% Example
% ima = imread('grass.jpg');
% ima = rgb2gray(ima);
% filtered_image = gaussianbpf(ima,30,120);
% Gaussian Bandpass Filter
f = double(I);
[nx ny] = size(f);
f = uint8(f);
fftI = fft2(f,2*nx-1,2*ny-1);
fftI = fftshift(fftI);
subplot(2,2,1)
imshow(f,[]);
title('Original Image')
subplot(2,2,2)
fftshow(fftI,'log')
title('Fourier Spectrum of Image')
% Initialize filter.
filter1 = ones(2*nx-1,2*ny-1);
filter2 = ones(2*nx-1,2*ny-1);
filter3 = ones(2*nx-1,2*ny-1);

```

Sumber:

<https://www.mathworks.com/matlabcentral/fileexchange/30947-gaussian-bandpass-filter-for-image-processing>

```

for i = 1:2*nx-1
    for j = 1:2*ny-1
        dist = ((i-(nx+1))^2 + (j-(ny+1))^2)^.5;
        % Use Gaussian filter.
        filter1(i,j) = exp(-dist^2/(2*d1^2));
        filter2(i,j) = exp(-dist^2/(2*d0^2));
        filter3(i,j) = 1.0 - filter2(i,j);
        filter3(i,j) = filter1(i,j).*filter3(i,j);
    end
end
% Update image with passed frequencies
filtered_image = fftI + filter3.*fftI;
subplot(2,2,3)
fftshow(filter3,'log')
title('Frequency Domain Filter Function Image')
filtered_image = ifftshift(filtered_image);
filtered_image = ifft2(filtered_image,2*nx-1,2*ny-1);
filtered_image = real(filtered_image(1:nx,1:ny));
filtered_image = uint8(filtered_image);
subplot(2,2,4)
imshow(filtered_image,[])
title('Bandpass Filtered Image')

```

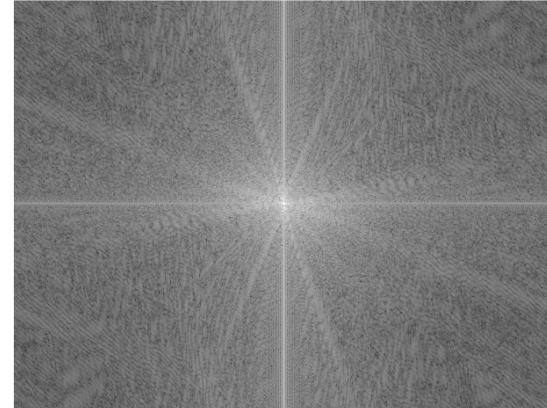
```

function fftshow(f,type)
% Usage: FFTSHOW(F,TYPE)
%
% Displays the fft matrix F using imshow, where TYPE must be one of
% 'abs' or 'log'. If TYPE='abs', then then abs(f) is displayed; if
% TYPE='log' then log(1+abs(f)) is displayed. If TYPE is omitted, then
% 'log' is chosen as a default.
%
% Example:
% c=imread('cameraman.tif');
% cf=fftshift(fft2(c));
% fftshow(cf,'abs')
%
if nargin<2,
    type='log';
end
if (type=='log')
    fl = log(1+abs(f));
    fm = max(fl(:));
    imshow(im2uint8(fl/fm))
elseif (type=='abs')
    fa=abs(f);
    fm=max(fa(:));
    imshow(fa/fm)
else
    error('TYPE must be abs or log.');
```

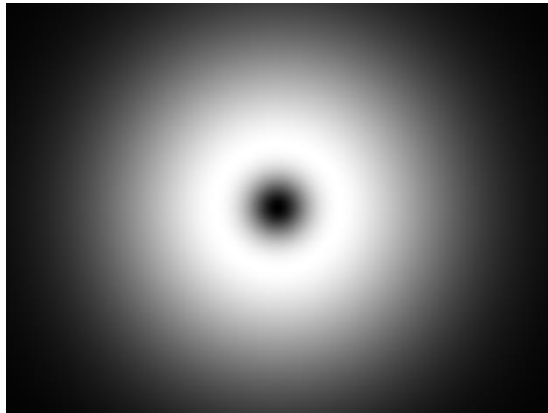
Original Image



Fourier Spectrum of Image



Frequency Domain Filter Function Image

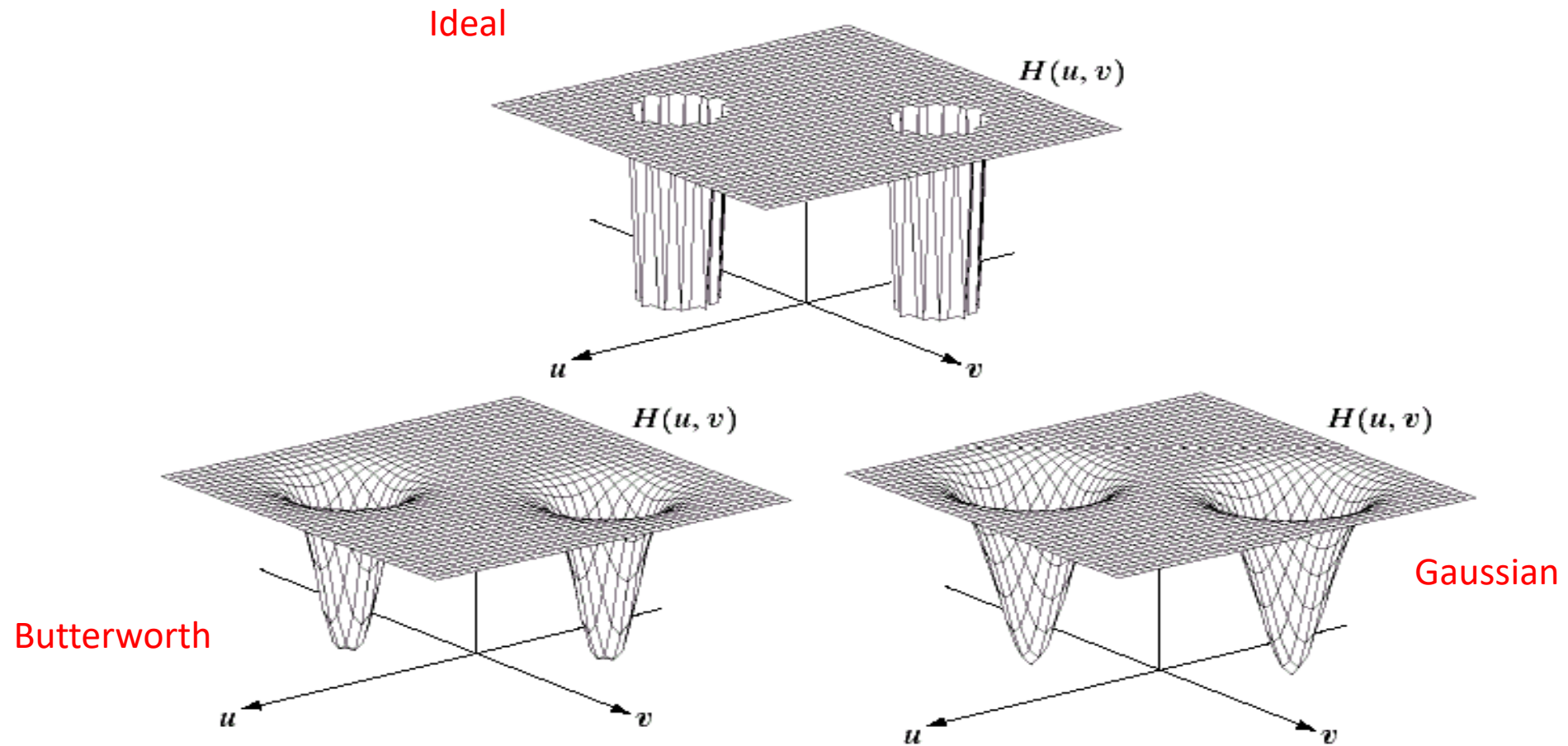


Bandpass Filtered Image



Notch Filters

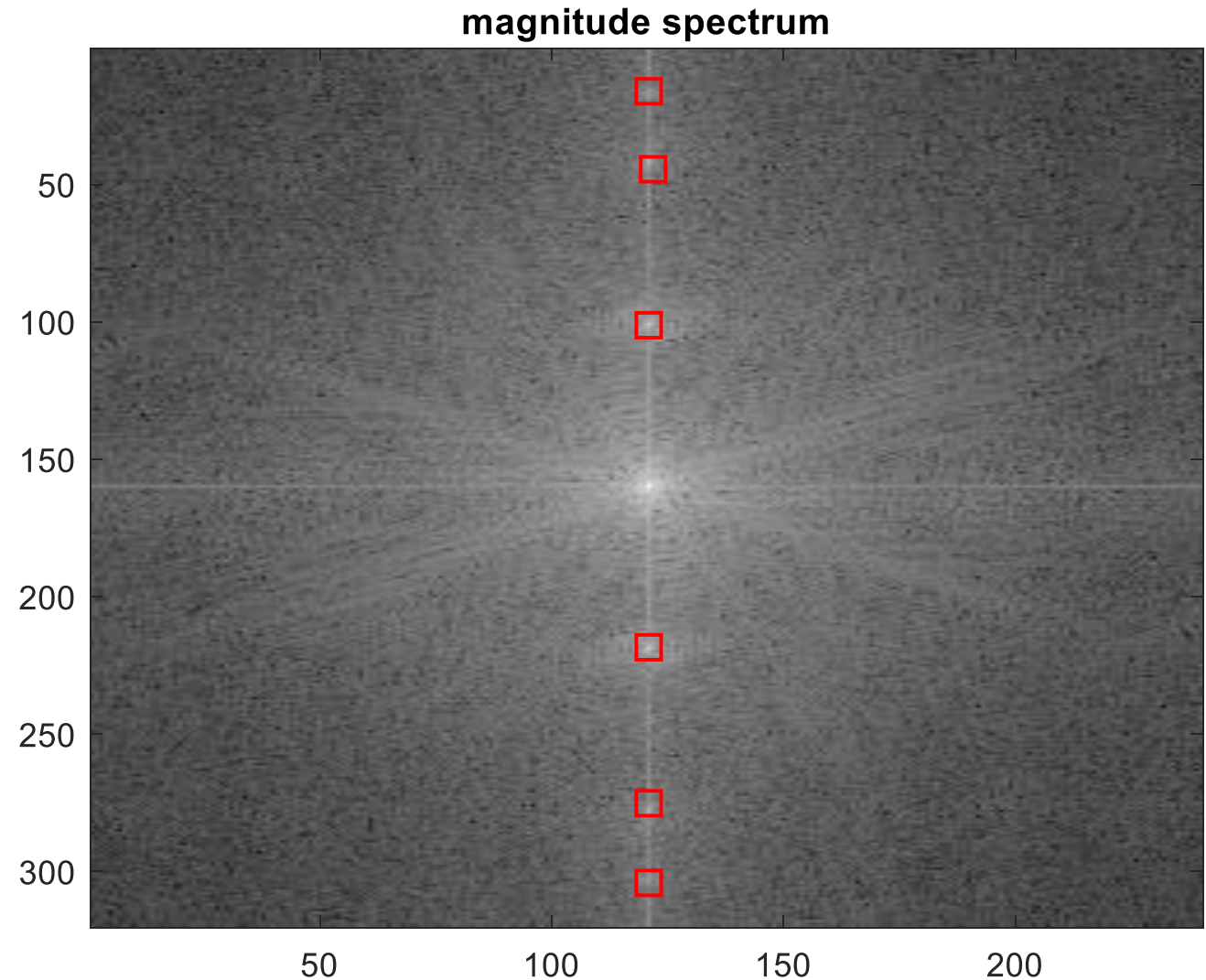
- *Notch Filters*
 - Bertujuan untuk menolak (atau melewatkan) frekuensi pada baris dan kolom yang “spikes”.
 - Perlu diperkirakan di bagian mana lokasi frekuensi yang *spikes*
 - Biasanya muncul berpasangan secara simetris terhadap titik asal (titik tengah persegi panjang)
 - Jika “menolak” maka disebut *notch reject filter*, jika “melewatkan” maka disebut *notch pass filter*.



a
b c

FIGURE 5.18 Perspective plots of (a) ideal, (b) Butterworth (of order 2), and (c) Gaussian notch (reject) filters.

Contoh:



Frekuensi yang mengalami derau ditentukan secara manual, kira-kira pada kolom 115-125, dan pada baris 12-22, 37-47, 96-106, 216-226, 274-284, 298-308.

Program Matlab (Sumber: <https://www.mathworks.com>)

```
I = imread('india.bmp'); % Baca citra
imshow(I);

% Terapkan Fourier Transform
F = fft2(double(I));
F1 = fftshift(F); % Pusatkan FFT

% Tampilkan magnitute spektrum Fourier
F2 = F1;
F2 = abs(F2); % Get the magnitude
F2 = log(F2+1); % Use log
figure, imagesc(100*F2); colormap(gray);
title('magnitude spectrum');
```

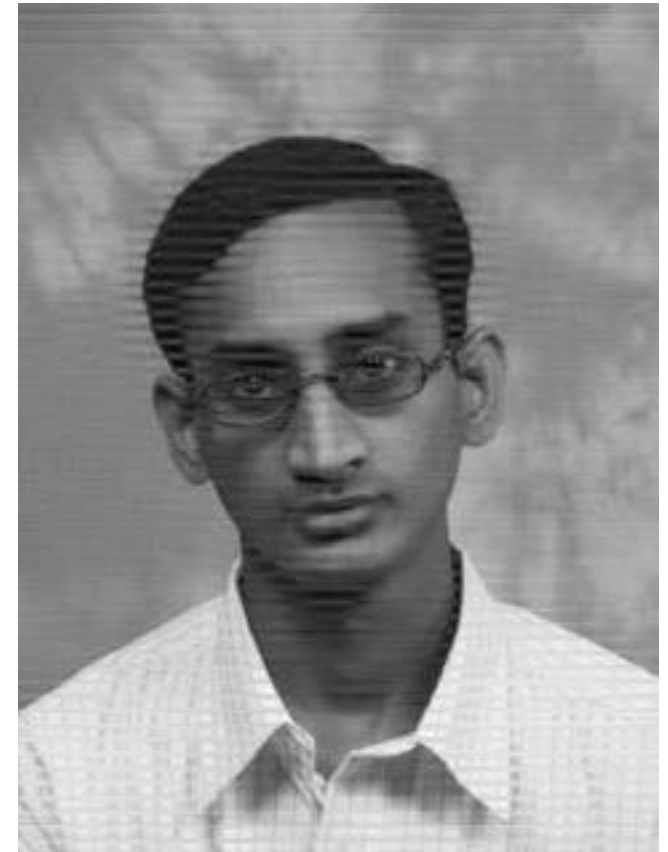
```

% Notch filter, Buang frekuensi yang mengganggu, set jadi 0
for j = 115:125
    for i = 96:106
        F1(i,j) = 0;
    end
    for i = 216:226
        F1(i,j) = 0;
    end
    for i = 274:284
        F1(i,j) = 0;
    end
    for i = 298:308
        F1(i,j) = 0;
    end
    for i = 12:22
        F1(i,j) = 0;
    end
    for i = 37:47
        F1(i,j) = 0;
    end
end
end

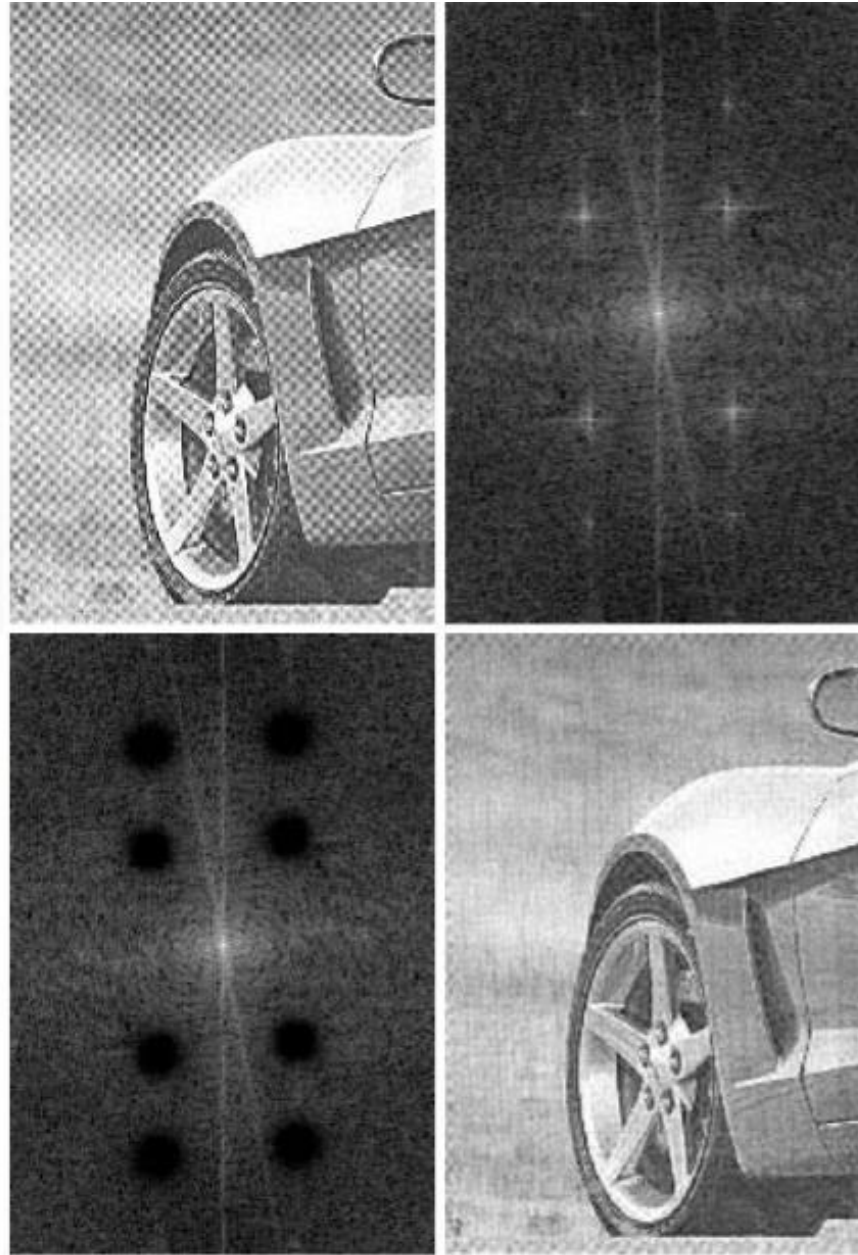
%Kembalikan ke ranah spasial
J = real(ifft2(ifftshift(F1)));
figure, imshow(J,[]);

```

Hasil penapisan frekuensi derau:



Examples:
Notch
Filters (1)

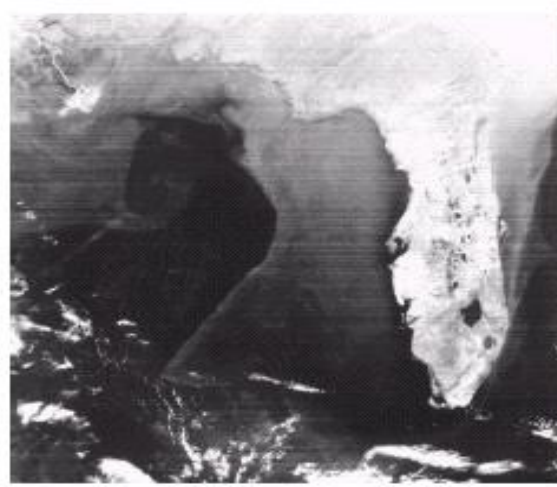


a	b
c	d

FIGURE 4.64

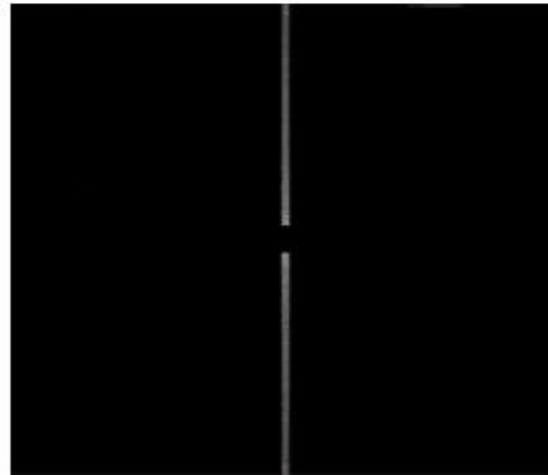
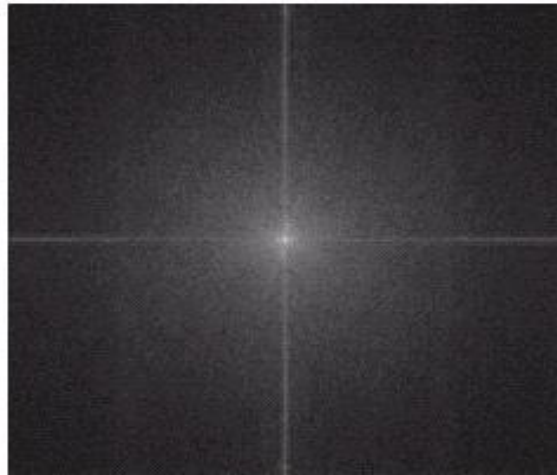
(a) Sampled newspaper image showing a moiré pattern.
(b) Spectrum.
(c) Butterworth notch reject filter multiplied by the Fourier transform.
(d) Filtered image.

A Butterworth notch reject filter $D_0=3$ and $n=4$ for all notch pairs

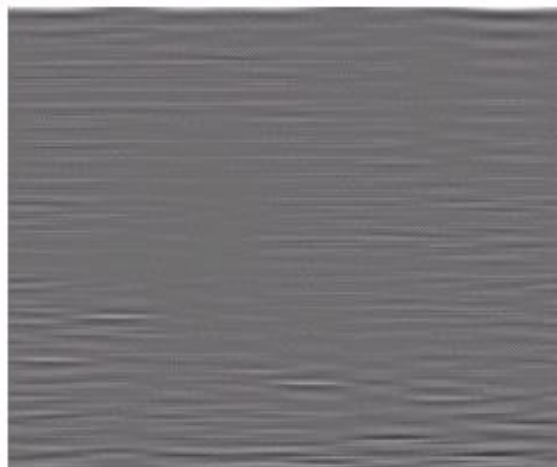


a
b c
d e

FIGURE 5.19 (a) Satellite image of Florida and the Gulf of Mexico (note horizontal sensor scan lines). (b) Spectrum of (a). (c) Notch pass filter shown superimposed on (b). (d) Inverse Fourier transform of filtered image, showing noise pattern in the spatial domain. (e) Result of notch reject filtering. (Original image courtesy of NOAA.)

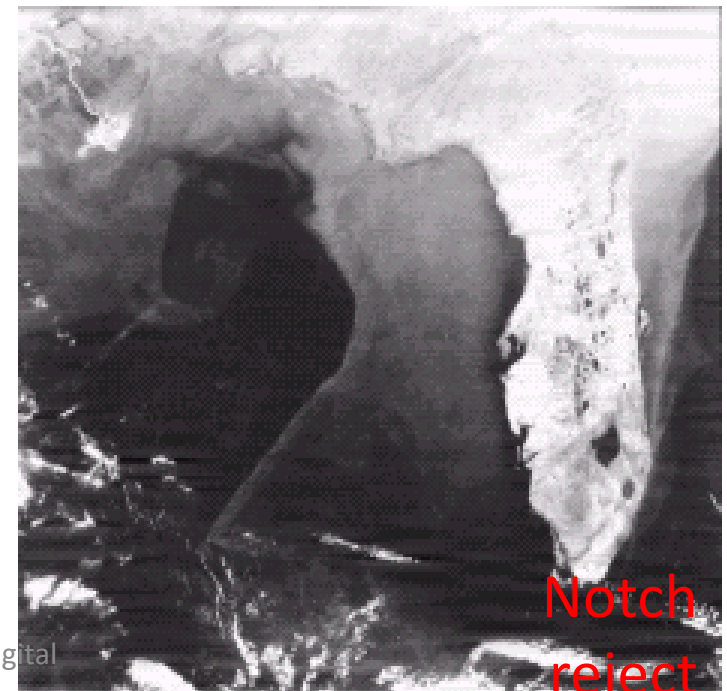
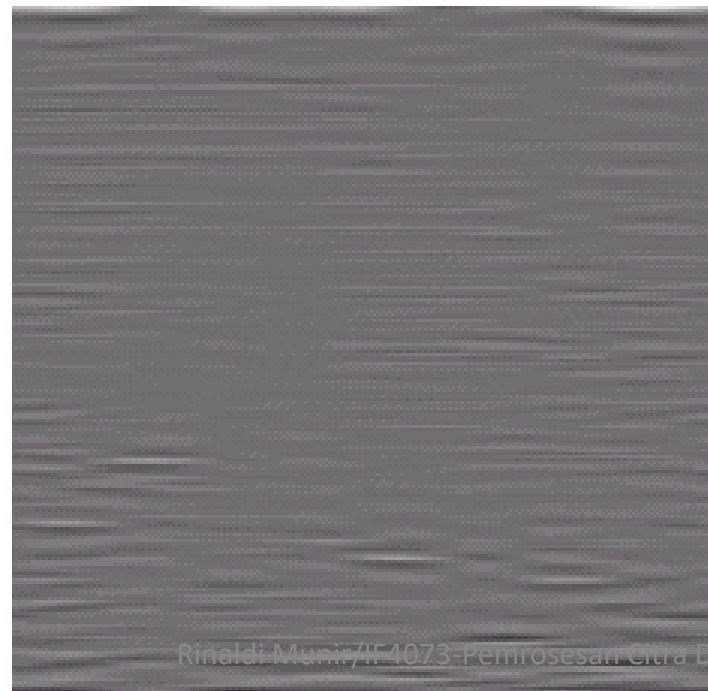
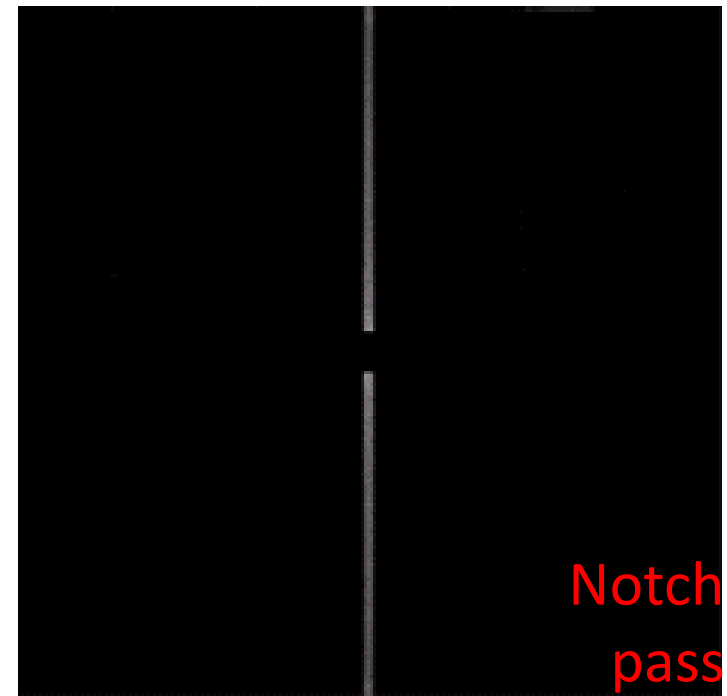
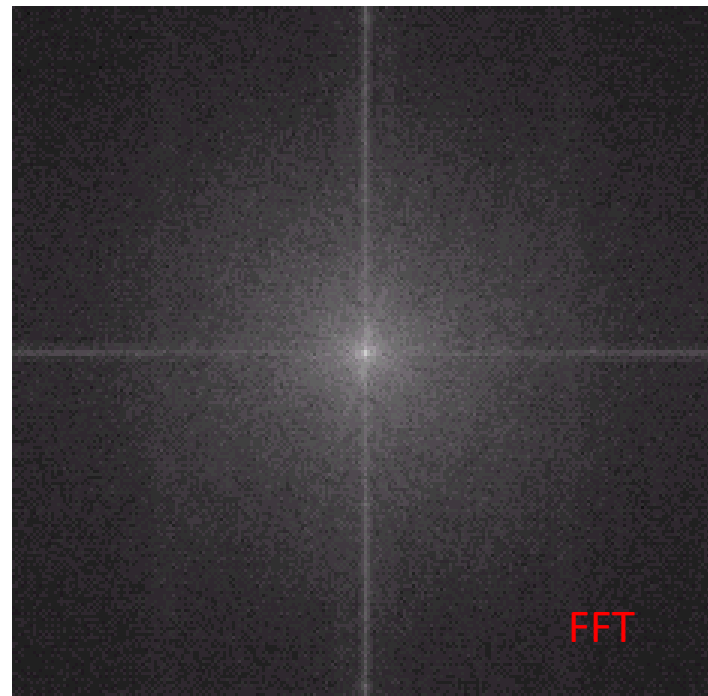
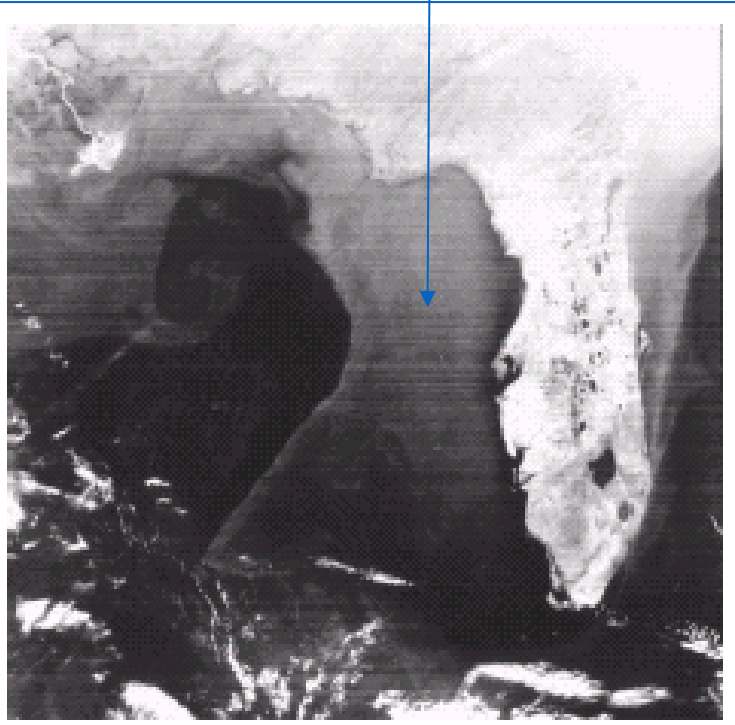


Notch pass filter



Horizontal lines of the noise pattern I can be seen

Horizontal
Scan lines



Motion Blur

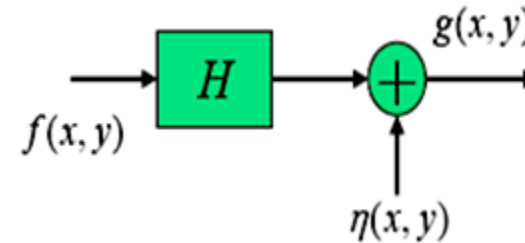
- Derau *motion blur* disebabkan oleh kamera yang bergerak atau objek yang bergerak



- Misalkan $H(u,v)$ adalah fungsi yang menyebabkan citra mengalami degradasi motion blur.

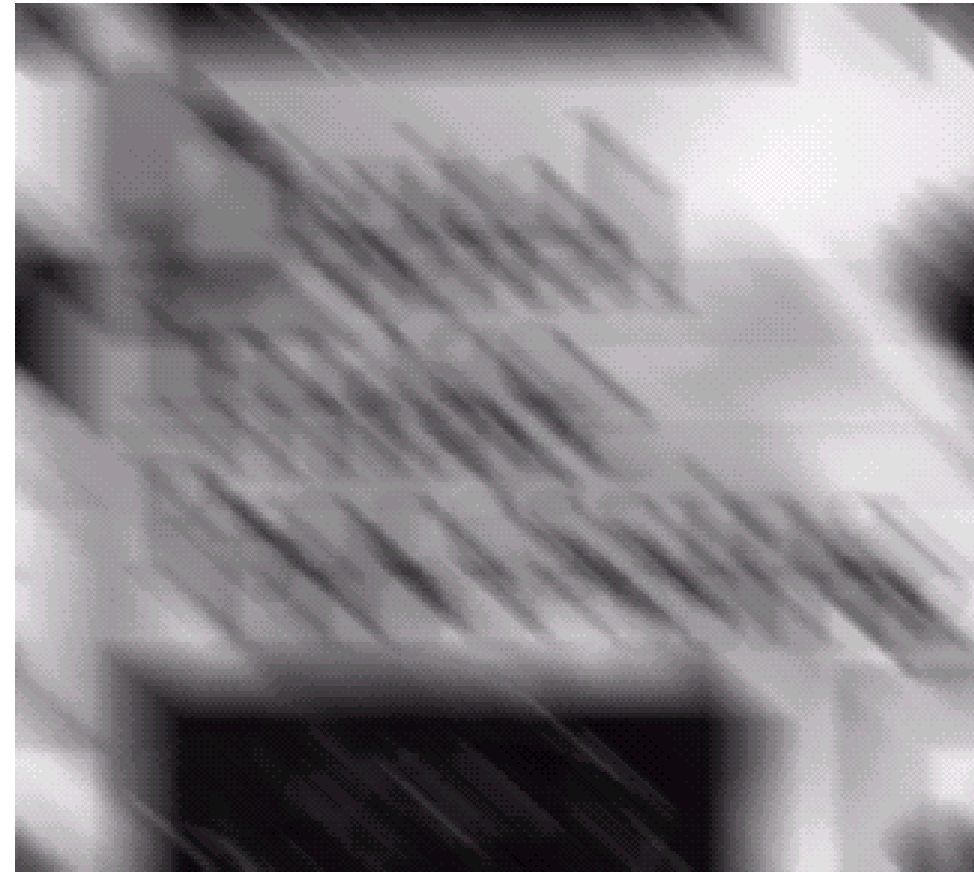
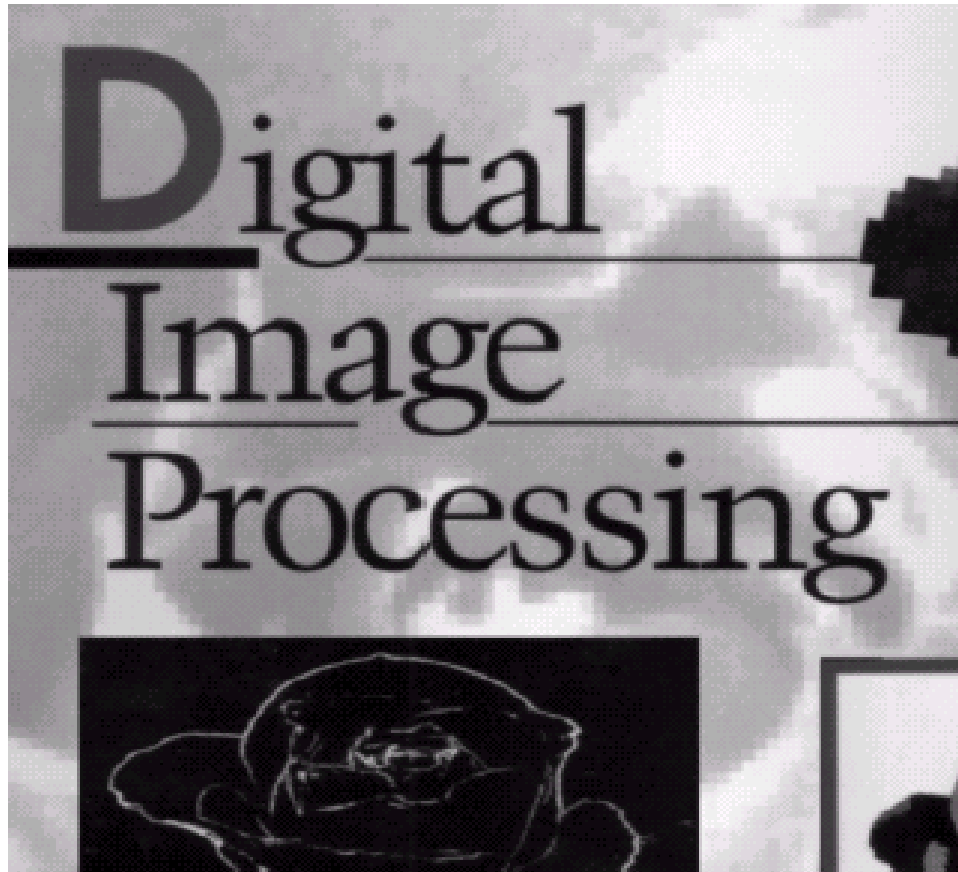
$$g(x,y) = h(x,y) * f(x,y) + \eta(x,y)$$

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



- Misalkan H adalah fungsi **motion linier uniform**. Gerakan dalam arah x dan dalam arah y adalah $x_o(t)=at/T$ dan $y_o(t)=bt/T$. Misalkan akibat *motion blur* pixel-pixel citra sudah berpindah sejauh a dalam arah x dan b dalam arah y , maka

$$H(u,v) = \frac{T}{\pi(ua + vb)} \sin[\pi(ua + vb)] e^{-j\pi(ua + vb)}$$



a b

FIGURE 5.26 (a) Original image. (b) Result of blurring using the function in Eq. (5.6-11) with $a = b = 0.1$ and $T = 1$.

- Fungsi degradasi H di dalam persamaan

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

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

kadang-kadang dinamakan juga ***point spread spectrum*** (PSF)

Contoh: citra bintang seharusnya tampak seperti pixel tunggal, tetapi yang tertangkap oleh kamera teleskop adalah citra yang diamati menyebar pada beberapa pixel,



Gambar 1.1. *Point Spread Spectrum* pada citra bintang yang ditangkap oleh teleskop. Penjelasan gambar: (a) Citra bintang seharusnya, (b) citra bintang yang diamati akibat distorsi oleh *PSF*.

- Jadi, citra terdegradasi dapat ditulis sebagai:

$$\text{Citra terdegradasi} = \text{citra asli} * \text{PSF} + \text{derau aditif}$$

- Sehingga, pekerjaan mendasar pada proses *deblurring* adalah **dekonvolusi** citra *motion blur* dengan PSF.
- Dekonvolusi adalah proses yang membalikkan efek konvolusi.
- Kualitas citra hasil *deblurring* terutama ditentukan oleh pengetahuan tentang PSF.

- Program Matlab untuk menghasilkan efek *motion blur*

```
I = imread('camera.bmp');  
% the linear motion of a camera by 20 pixels, with an angle of 45 degrees  
% in a counterclockwise direction. Default len = 9 and degree = 0  
H = fspecial('motion', 20, 45);  
MotionBlur = imfilter(I,H,'replicate');  
imshow(MotionBlur);
```

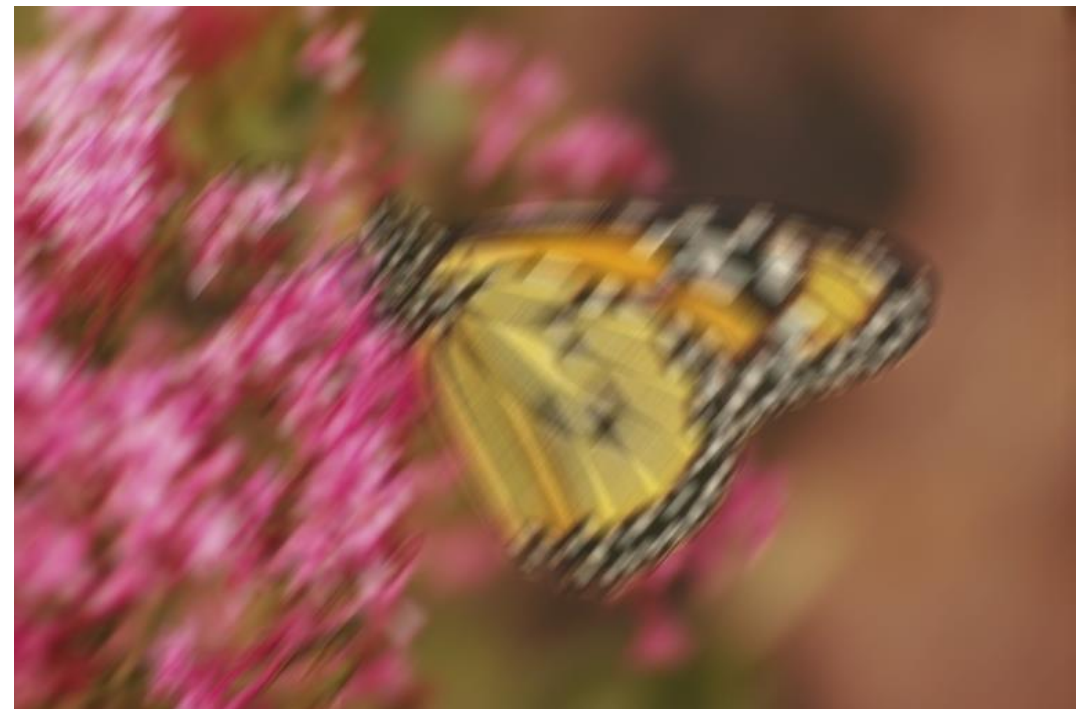
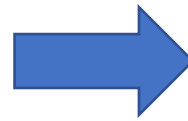


- Bedakan *motion blur* dengan *blur* biasa (misalnya yang dihasilkan oleh penapis *gaussian*):

```
I = imread('camera.bmp');  
H = fspecial('gaussian', 9, 5); %Penapis gaussian 9x9, standard deviasi=2  
Blur = imfilter(I,H,'replicate');  
imshow(Blur);
```



```
I = imread('monarch.jpg');  
% the linear motion of a camera by len = 30 pixels and tetha = 50 degrees  
% in a counterclockwise direction. Deafult len = 9 and tetha = 0  
H = fspecial('motion', 30, 50);  
MotionBlur = imfilter(I,H,'replicate');  
imshow(MotionBlur);
```



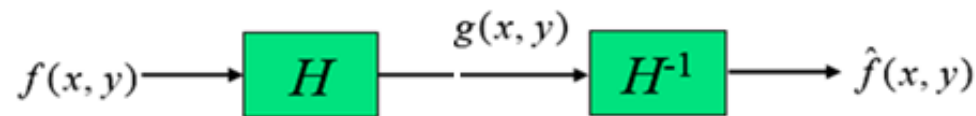
Inverse Filtering

- Misalkan tidak terdapat derau aditif, maka citra degradasi dapat digambarkan sebagai $g(x, y) = f(x, y) * h(x, y)$.

- Dalam ranah frekuensi, maka

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

Estimasi
citra semula



$$\hat{f}(x, y) = g(x, y)h^{-1}(x, y)$$

$$\hat{F}(u, v) = G(u, v)H^{-1}(u, v) = \frac{G(u, v)}{H(u, v)}$$

- Metode ini dinamakan *inverse filtering* atau dekonvolusi

- Jika terdapat derau aditif, maka $g(x, y) = f(x, y) * h(x, y) + n(x, y)$, sehingga

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


Masalah: jika $H = 0$ atau nilai yang sangat kecil

Masalah:

- Nilai $H(u, v)$ yang kecil dapat menyebabkan overflow.
- Jika derau aditif dilibatkan, maka ia akan mendominasi

Solusi:

- Lakukan pembagian hanya pada bagian terbatas bidang (u, v) sejauh radius tertentu dari titik asal.

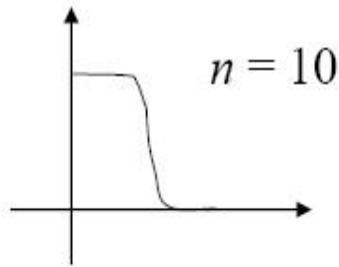
Inverse filtering

Fungsi degradasi

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

- An example

Butterworth filter H_b



a b
c d

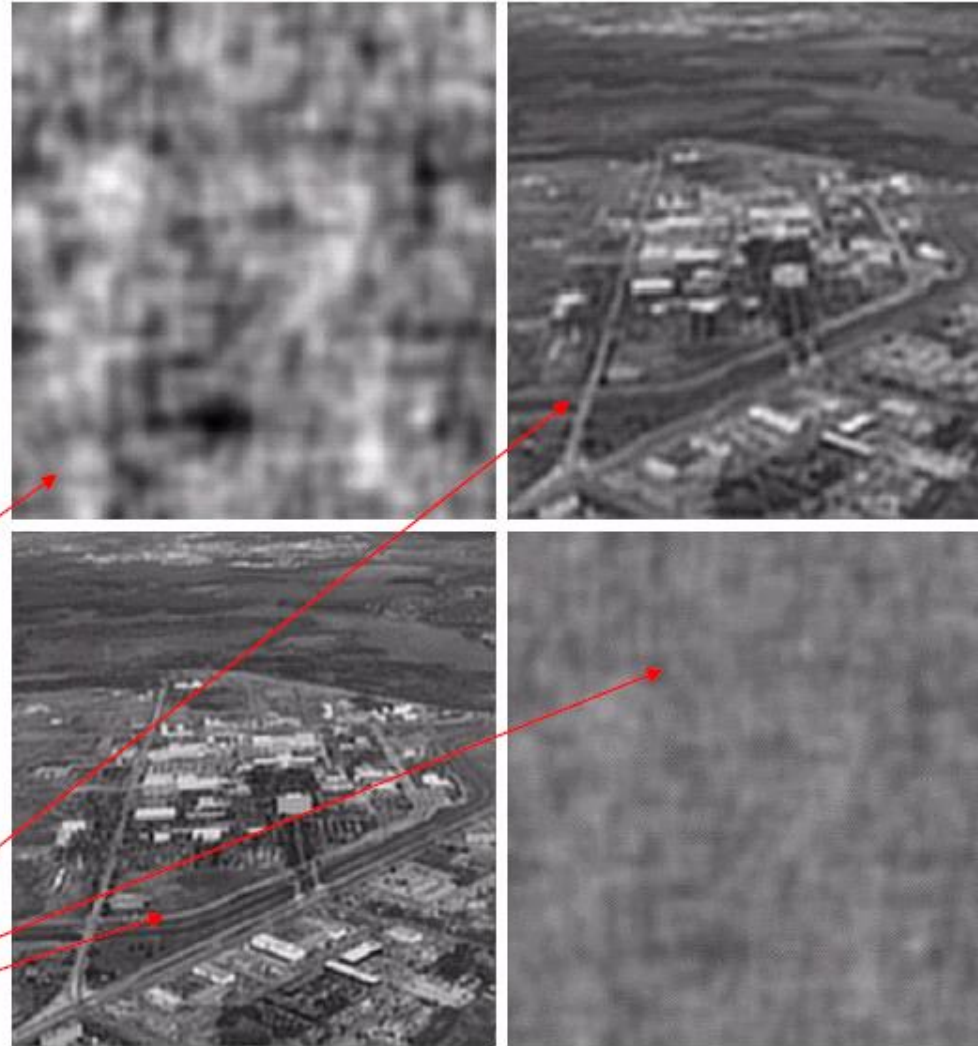
FIGURE 5.27
Restoring
Fig. 5.25(b) with
Eq. (5.7-1).
(a) Result of
using the full
filter. (b) Result
with H cut off
outside a radius of
40; (c) outside a
radius of 70; and
(d) outside a
radius of 85.

$$\frac{G(u, v)}{H(u, v)} = \hat{F}(u, v) \Leftrightarrow \hat{f}(x, y)$$

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

$$\Updownarrow$$

$$\hat{f}(x, y)$$



Penapis Wiener

- Penapis ini ditemukan oleh N. Wiener pada tahun 1942
- Penapis ini efektif bila karakteristik frekuensi citra dan derau aditif diketahui. Jika tidak ada derau aditif, penapis Wiener menjadi penapis yang ideal.
- Penapis Wiener mengestimasi $\hat{f}$ dengan meminimumkan galat, yaitu selisih citra asli dengan citra hasil restorasi:

$$e^2 = E\{ (f - \hat{f})^2 \}$$

- Penapis Wiener memiliki persamaan:

$$H_w(u,v) = \frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)|^2 + S_n(u,v) / S_f(u,v)}$$

$H(u,v)$ = fungsi degradasi

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

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

$S_n(u,v)$ = power spectrum dari derau

$S_f(u,v)$ = power spectrum dari citra degradasi

- Jika derau aditif tidak ada ($N(u,v) = 0$), maka

$$H_w(u,v) = \frac{1}{H(u,v)} \Rightarrow \text{the inverse filter}$$

- Jika $S_n(u,v)$ dan/atau $S_f(u,v)$ tidak diketahui, maka

$$H_w(u,v) \approx \frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)|^2 + K}$$

some constant

Nilai K dapat di-adjust secara intraktif

- Dengan penapis Wiener, citra restorasi dapat diperoleh dengan mengalikan penapis tersebut dengan citra degradasi:

$$\hat{F}(u,v) = H_w(u,v).G(u,v)$$

Program Matlab untuk *deblurring* dengan Wiener Filter (deconvwnr)

```
J = deconvwnr(I,PSF,NSR)
```

PSF is the point-spread function with which I was convolved. NSR is the noise-to-signal power ratio of the additive noise. NSR can be a scalar or a spectral-domain array of the same size as I. Specifying 0 for the NSR is equivalent to creating an ideal inverse filter. (Sumber: Matlab)

```
% Baca citra
I = imread('cameraman.bmp');
imshow(I);
% Kaburkan citra dengan motion blur
LEN = 20; THETA = 30;
PSF = fspecial('motion', LEN, THETA);
blurred = imfilter(I, PSF, 'conv', 'circular');
figure, imshow(blurred);
title('Blurred Image');

% Restorasi citra dengan penapis Wiener
wnr1 = deconvwnr(blurred, PSF, 0); %NSR = 0, tidak ada noise aditif
figure, imshow(wnr1); title('Restored Image');
% Hitung selisih antara citra asli dengan citra hasil restorasi
figure; imshow(imabsdiff(I, blurred));
title('Perbedaan citra asli dengan citra restorasi')
```

Hasil run program:

Original image



Blurred Image



Restored Image



Perbedaan citra asli dengan citra restorasi



Jika ada tambahan *noise* pada citra blur:

```
I = im2double(imread('cameraman.bmp'));
imshow(I);
% Kaburkan citra dengan motion blur
LEN = 21; THETA = 11;
PSF = fspecial('motion', LEN, THETA);
blurred = imfilter(I, PSF, 'conv', 'circular');

% Tambahkan derau aditif yaitu derau gaussian
noise_mean = 0;
noise_var = 0.0001;
blurred_noisy = imnoise(blurred, 'gaussian', noise_mean, noise_var);
figure, imshow(blurred_noisy);
title('Blurred Image with noise');

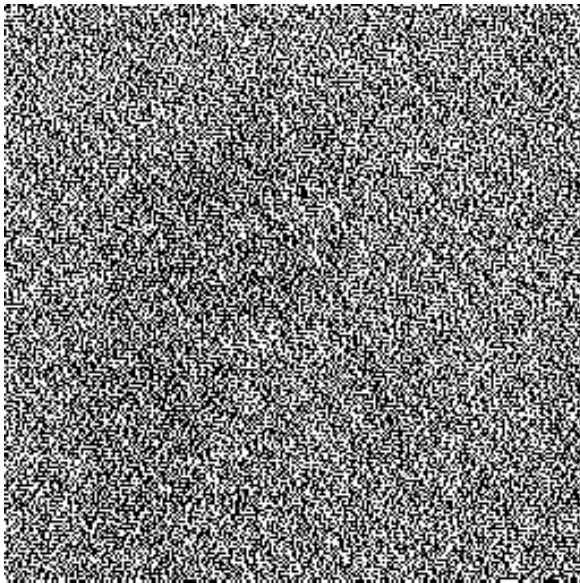
% Restorasi dengan asumsi tanpa ada noise aditif.
estimated_nsr = 0;
wnr2 = deconvwnr(blurred_noisy, PSF, estimated_nsr);
figure, imshow(wnr2)
title('Restoration of Blurred, Noisy Image Using NSR = 0')

% Restorasi dengan asumsi ada derau aditif, estimasi the noise-to-signal-power ratio.
estimated_nsr = noise_var / var(I(:));
wnr3 = deconvwnr(blurred_noisy, PSF, estimated_nsr);
figure, imshow(wnr3)
title('Restoration of Blurred, Noisy Image Using Estimated NSR');
```

Blurred Image with noise



Restoration of Blurred, Noisy Image Using NSR = 0



Restoration of Blurred, Noisy Image Using Estimated NSR



Program *deblurring* untuk citra berwarna

```
I = imread('peppers512.bmp'); % baca citra
figure; imshow(I); title('Citra Lada asli');

% Kaburkan citra dengan motion blur
LEN = 20; % Panjang blur (satuan: pixel)
TETHA = 45; % sudut blur (satuan: derajat)
PSF = fspecial('motion', LEN, TETHA);
Blurred = imfilter(I, PSF, 'circular', 'conv');
figure; imshow(Blurred); title('Citra terdegradasi (motion blur)')

% Restorasi citra dengan penapis Wiener
wnr1 = deconvwnr(Blurred, PSF);
figure; imshow(wnr1);
title('Citra hasil restorasi');

% Hitung selisih antara citra asli dengan citra hasil restorasi
figure; imshow(imabsdiff(I, Blurred));
title('Perbedaan citra asli dengan citra restorasi')
```

Citra Lada asli



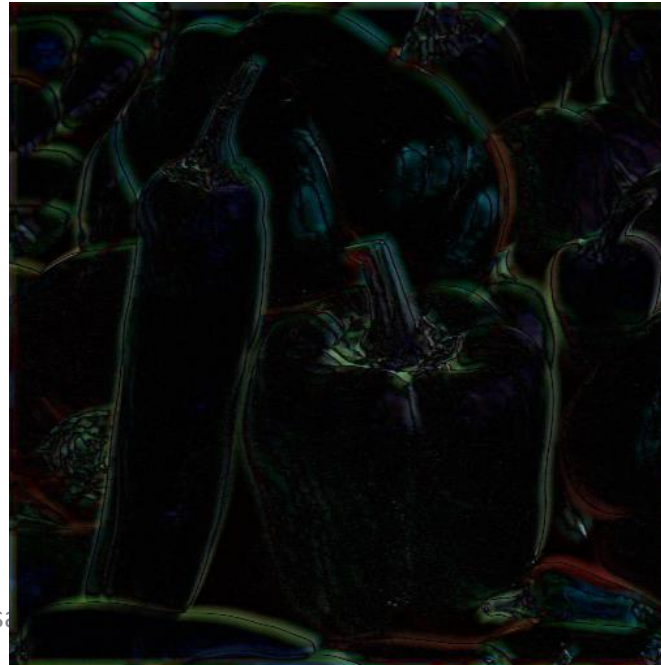
Citra terdegradasi (motion blur)



Restored Image



Perbedaan citra asli dengan citra restorasi



Algoritma Lucky-Richardson

- Algoritma *Lucy-Richardson* (L-R) adalah algoritma restorasi iteratif yang dikembangkan secara independen oleh Richardson (1972) dan Lucy (1974).
- Algoritma ini efektif jika kita mengetahui *PSF* tetapi hanya mengetahui sedikit mengenai derau aditif pada citra.
- Algoritma L-R pada mulanya digunakan untuk merestorasi citra astronomi, sebelum akhirnya digunakan juga secara luas untuk merestorasi sembarang citra yang mengalami kekaburan.

- Algoritma L-R memaksimumkan kemungkinan (*maximum likelihood*) bahwa sebuah citra bila dikonvolusi dengan *PSF* hasilnya adalah instansiasi dari citra kabur, dengan mengasumsikan derau tersebut dengan distribusi *Poisson*.
- Distribusi *Poisson* adalah $p(x) = \frac{e^{-a} a^x}{x!}$ dengan x adalah peubah acak dan a adalah konstanta.
- Esensi dari iterasi adalah sebagai berikut: estimasi ke-($n+1$) dari citra restorasi adalah estimasi ke- n citra restorasi dikali dengan citra koreksi. Persamaan iterasinya adalah

$$\hat{f}_{n+1} = \hat{f}_n \left(\frac{g}{\hat{f}_n * PSF} \right) * reflect(PSF)$$

$$\hat{f}_{n+1} = \hat{f}_n \left(\frac{g}{\hat{f}_n * PSF} \right) * reflect(PSF)$$

yang dalam hal ini,

- operator $*$ menyatakan konvolusi
- $\hat{f} = \hat{f}(x, y)$ menyatakan estimasi citra restorasi
- $g = g(x, y)$ menyatakan citra masukan (yang mengalami degradasi)
- $reflect(PSF)$ menyatakan pencerminan PSF, yaitu

$$reflect((PSF(x, y))) = PSF(-x, -y)$$

- $\left(\frac{g}{\hat{f}_n * PSF} \right) * reflect(PSF)$ menyatakan citra koreksi
- Nilai awal iterasi adalah $\hat{f}_0 = g * PSF$. Kekonvergenan algoritma Lucy-Richardson berarti citra koreksi mendekati satu (*unity*) ketika iterasi bertambah

Program Matlab *deblurring* menggunakan algoritma Lucky-Richardson dengan berbagai jumlah iterasi

```
%baca citra
I = imread('peppers512.bmp');
figure; imshow(I); title('Citra Lada asli');

% Motion-blur kan citra
LEN = 30; % Panjang blur (satuan: pixel)
TETHA = 10; % sudut blur (satuan: derajat)
PSF = fspecial('motion', LEN, TETHA);
Blurred = imfilter(I, PSF, 'circular', 'conv');
figure; imshow(Blurred); title('Citra terdegradasi (motion blur)')

% Restorasi dengan L-R, jumlah iterasi = 5
luc1 = deconvlucy(Blurred, PSF, 5);
figure; imshow(luc1); title('Citra lada terestorasi, jumlah iterasi = 5');
```

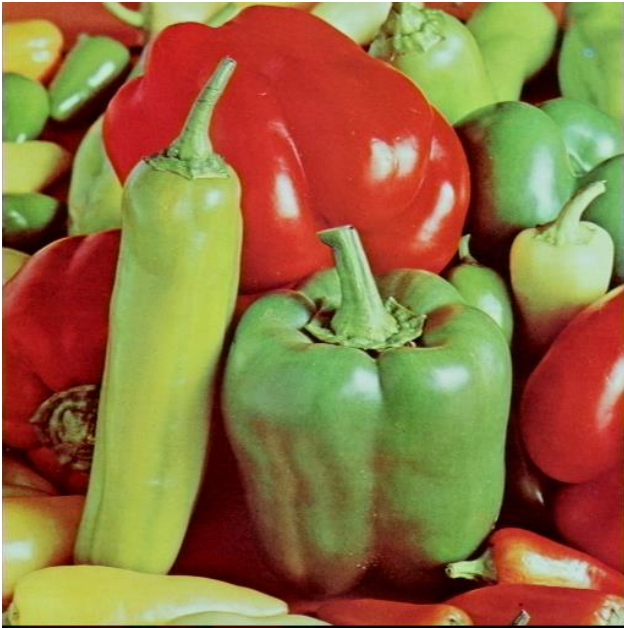
```
% Restorasi dengan L-R, jumlah iterasi = 10
luc1 = deconvlucy(Blurred, PSF, 10);
figure; imshow(luc1); title('Citra lada terestorasi, jumlah iterasi = 10');

% Restorasi dengan L-R, jumlah iterasi = 15
luc1 = deconvlucy(Blurred, PSF, 15);
figure; imshow(luc1); title('Citra lada terestorasi, jumlah iterasi = 15');

% Restorasi dengan L-R, jumlah iterasi = 20
luc1 = deconvlucy(Blurred, PSF, 20);
figure; imshow(luc1); title('Citra lada terestorasi, jumlah iterasi = 20');
```

Hasil run program (setelah slide ini)

Citra Lada asli



Citra terdegradasi (motion blur)



Citra lada terestorasi, jumlah iterasi = 5



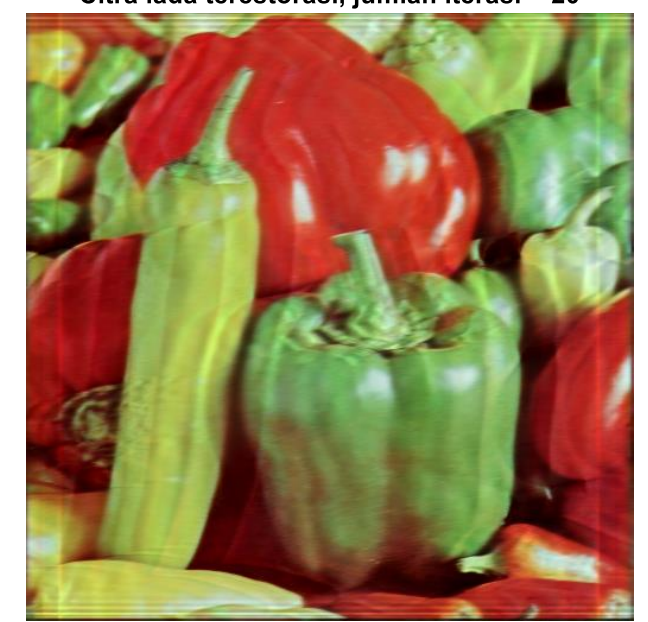
Citra lada terestorasi, jumlah iterasi = 10



Citra lada terestorasi, jumlah iterasi = 15



Citra lada terestorasi, jumlah iterasi = 20



Citra cameraman asli



Citra terdegradasi (motion blur)



Citra lada terestorasi, jumlah iterasi = 5



Citra lada terestorasi, jumlah iterasi = 10



Citra lada terestorasi, jumlah iterasi = 15



Citra lada terestorasi, jumlah iterasi = 20



- Kelemahan algoritma L-R adalah pada jumlah iterasi yang tinggi (seperti diperlihatkan pada gambar dengan jumlah iterasi = 20) dapat memunculkan noda-noda baru yang tidak terdapat pada citra aslinya.
- Noda ini disebut *artifact*. *Artifact* merupakan hasil pengerasan (*noise amplification*) yang meningkat dengan bertambahnya iterasi. *Artifact* tidak terdapat pada citra restorasi hasil penapisan Wiener
- Ini adalah persoalan umum pada semua teknik *maximumlikelihood* (termasuk algoritma *Lucy-Richardson*), yang mencoba mencocokkan data sedekat mungkin dengan citra aslinya.
- Cara yang praktis untuk membatasi pengerasan derau adalah menghentikan iterasi bila citra restorasi muncul dengan derau yang terlalu banyak.

TAMAT