

Tugas 2 IF4073 Pemrosesan Citra Digital

Penapisan Citra dalam Ranah Spasial, Ranah Frekuensi, dan Restorasi Citra



Disusun Oleh:

13521042 Kevin John Wesley Hutabarat

13521059 Arleen Chrysantha Gunardi

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
Oktober 2024**

Daftar Isi

Daftar Isi.....	1
Daftar Gambar.....	2
1. GUI.....	6
2. Penjelasan Program.....	6
2.1. Konvolusi.....	6
2.2. Pelembutan Citra pada Ranah Spasial.....	10
2.3. Pelembutan Citra pada Ranah Frekuensi: Low-pass Filter.....	25
2.4. High-pass Filter.....	41
2.5. Image Brightening.....	50
2.6. Penghilangan Derau Salt and Pepper.....	53
2.7. Penghilangan Derau Periodik.....	67
2.8. Motion Blurring dan Dekonvolusi.....	72
3. Lampiran.....	78

Daftar Gambar

Gambar 1.1 GUI Program.....	6
Gambar 2.1 Konvolusi citra kupu-kupu dengan n = 3 (a).....	7
Gambar 2.2 Konvolusi citra kupu-kupu dengan n = 7 (c).....	8
Gambar 2.3 Konvolusi citra jam dengan n = 3 (b).....	8
Gambar 2.4 Konvolusi citra kupu-kupu dengan n = 5 (e).....	8
Gambar 2.5 Konvolusi citra Barbara dengan n =5 (e).....	9
Gambar 2.6 Konvolusi citra pemandangan salju dengan n = 3 (d).....	9
Gambar 2.7 Konvolusi citra pemandangan salju dengan n = 5.....	9
Gambar 2.8 Konvolusi citra pemandangan pohon dengan n = 5 (e).....	9
Gambar 2.9 Konvolusi citra pemandangan pohon dengan n = 7 (b).....	10
Gambar 2.10 Konvolusi citra peppers dengan n = 3 (a).....	10
Gambar 2.11 Mean filter citra boat dengan n = 5.....	11
Gambar 2.12 Mean filter citra boat dengan n = 7.....	12
Gambar 2.13 Gaussian filter citra boat dengan n = 5 dan $\sigma = 1.5$	12
Gambar 2.14 Gaussian filter citra boat dengan n = 7 dan $\sigma = 1.5$	13
Gambar 2.15 Mean filter citra bangunan dengan n = 5.....	13
Gambar 2.16 Mean filter citra bangunan dengan n = 7.....	14
Gambar 2.17 Gaussian filter citra bangunan dengan n = 5 dan $\sigma = 1.5$	14
Gambar 2.18 Gaussian filter citra bangunan dengan n = 7 dan $\sigma = 1.5$	15
Gambar 2.19 Mean filter citra Planet Saturnus dengan n = 5.....	15
Gambar 2.20 Mean filter citra Planet Saturnus dengan n = 9.....	16
Gambar 2.21 Gaussian filter citra Planet Saturnus dengan n = 5 dan $\sigma = 1.5$	16
Gambar 2.22 Gaussian filter citra Planet Saturnus dengan n = 9 dan $\sigma = 1.5$	17
Gambar 2.23 Mean filter citra ginjal dengan n = 5.....	17
Gambar 2.24 Gaussian filter citra ginjal dengan n = 5 dan $\sigma = 1.5$	18
Gambar 2.25 Mean filter citra Gedung Sate dengan n = 3.....	18
Gambar 2.26 Mean filter citra Gedung Sate dengan n = 7.....	19
Gambar 2.27 Gaussian filter citra Gedung Sate dengan n = 3 dan $\sigma = 1.5$	19
Gambar 2.28 Gaussian filter citra Gedung Sate dengan n = 7 dan $\sigma = 1.5$	20
Gambar 2.29 Gaussian filter citra Gedung Sate dengan n = 7 dan $\sigma = 13$	20
Gambar 2.30 Mean filter citra kupu-kupu dengan n = 5.....	21
Gambar 2.31 Gaussian filter citra kupu-kupu dengan n = 3 dan $\sigma = 1.5$	21
Gambar 2.32 Gaussian filter citra kupu-kupu dengan n = 10 dan $\sigma = 1.5$	22
Gambar 2.33 Mean filter citra galaksi dengan n = 3.....	22
Gambar 2.34 Mean filter citra galaksi dengan n = 5.....	23
Gambar 2.35 Gaussian filter citra galaksi dengan n = 5 dan $\sigma = 50$	23

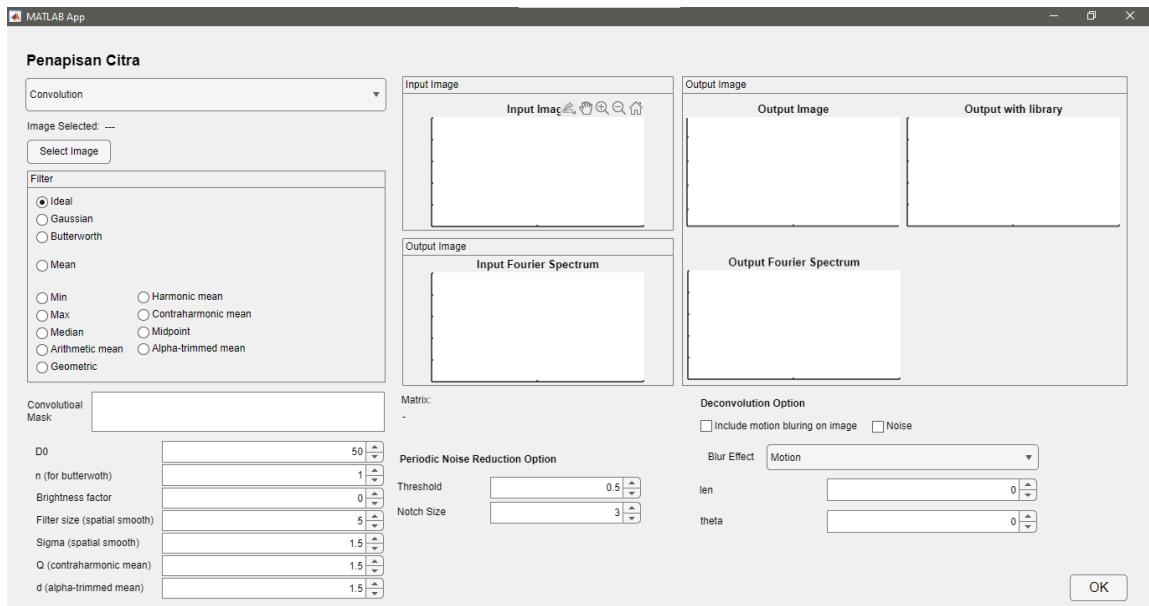
Gambar 2.36 Gaussian filter citra galaksi dengan $n = 5$ dan $\sigma = 1.5$	24
Gambar 2.37 Mean filter citra caster dengan $n = 5$	24
Gambar 2.38 Gaussian filter citra caster dengan $n = 5$ dan $\sigma = 1.5$	25
Gambar 2.39 ILPF citra boat dengan $d_0 = 50$	27
Gambar 2.40 ILPF citra boat dengan $d_0 = 25$	28
Gambar 2.41 GLPF citra boat dengan $d_0 = 50$	28
Gambar 2.42 BLPF citra boat dengan $d_0 = 50$ dan $n = 1$	29
Gambar 2.43 ILPF citra bangunan dengan $d_0 = 60$	29
Gambar 2.44 GLPF citra bangunan dengan $d_0 = 60$	30
Gambar 2.45 BLPF citra bangunan dengan $d_0 = 60$ dan $n = 1$	30
Gambar 2.46 BLPF citra bangunan dengan $d_0 = 60$ dan $n = 3$	31
Gambar 2.47 Mean filter citra Planet Saturnus dengan $d_0 = 65$	31
Gambar 2.48 Gaussian filter citra Planet Saturnus dengan $d_0 = 65$	32
Gambar 2.49 BLPF citra Planet Saturnus dengan $d_0 = 65$ dan $n = 1$	32
Gambar 2.50 ILPF citra galaksi dengan $d_0 = 40$	33
Gambar 2.51 GLPF citra galaksi dengan $d_0 = 40$	33
Gambar 2.52 BLPF citra galaksi dengan $d_0 = 40$ dan $n = 1$	34
Gambar 2.53 ILPF citra Gedung Sate dengan $d_0 = 50$	34
Gambar 2.54 GLPF citra Gedung Sate dengan $d_0 = 50$	35
Gambar 2.55 BLPF citra Gedung Sate dengan $d_0 = 50$ dan $n = 1$	35
Gambar 2.56 ILPF citra kupu-kupu dengan $d_0 = 45$	36
Gambar 2.57 GLPF citra kupu-kupu dengan $d_0 = 45$	36
Gambar 2.58 BLPF citra kupu-kupu dengan $d_0 = 45$ dan $n = 10$	37
Gambar 2.59 ILPF citra galaksi dengan $d_0 = 55$	37
Gambar 2.60 GLPF citra galaksi dengan $d_0 = 55$	38
Gambar 2.61 BLPF citra galaksi dengan $d_0 = 55$ dan $n = 5$	38
Gambar 2.62 ILPF citra caster dengan $d_0 = 25$	39
Gambar 2.63 GLPF citra caster dengan $d_0 = 25$	39
Gambar 2.64 BLPF citra caster dengan $d_0 = 25$ dan $n = 3$	40
Gambar 2.65 IHPF citra kucing dengan $D_0 = 50$	43
Gambar 2.66 GHPF citra kucing dengan $D_0 = 50$	43
Gambar 2.67 BHPF citra kucing dengan $D_0 = 50$ dan $n = 3$	44
Gambar 2.68 IHPF citra bangunan dengan $D_0 = 40$	44
Gambar 2.69 GHPF citra bangunan dengan $D_0 = 40$	45
Gambar 2.70 BHPF citra bangunan dengan $D_0 = 40$ dan $n = 3$	45
Gambar 2.71 IHPF citra Einstein dengan $D_0 = 60$	46
Gambar 2.72 GHPF citra Einstein dengan $D_0 = 60$	46
Gambar 2.73 BHPF citra Einstein dengan $D_0 = 60$ dan $n = 3$	47

Gambar 2.74 IHPF citra papan elektronik dengan D0 = 100.....	47
Gambar 2.75 GHPF citra papan elektronik dengan D0 = 100.....	48
Gambar 2.76 BHPF citra papan elektronik dengan D0 = 70 dan n = 3.....	48
Gambar 2.77 IHPF citra waterfall dengan d0 = 50.....	49
Gambar 2.78 GHPF citra waterfall dengan D0 = 50.....	49
Gambar 2.79 BHPF citra waterfall dengan D0 = 60 dan n = 4.....	50
Gambar 2.80 Image brightening citra terowongan.....	52
Gambar 2.81 Image brightening citra bunga.....	52
Gambar 2.82 Penghilangan derau salt and pepper citra bangunan dengan min filter.....	54
Gambar 2.83 Penghilangan derau salt and pepper citra bangunan dengan max filter.....	55
Gambar 2.84 Penghilangan derau salt and pepper citra bangunan dengan median filter.....	55
Gambar 2.85 Penghilangan derau salt and pepper citra bangunan dengan arithmetic mean filter	55
Gambar 2.86 Penghilangan derau salt and pepper citra bangunan dengan geometric mean filter	56
Gambar 2.87 Penghilangan derau salt and pepper citra bangunan dengan harmonic mean filter.	56
Gambar 2.88 Penghilangan derau salt and pepper citra bangunan dengan contraharmonic mean filter dengan Q = 1.5.....	56
Gambar 2.89 Penghilangan derau salt and pepper citra bangunan dengan contraharmonic mean filter dengan Q = -1.5.....	57
Gambar 2.90 Penghilangan derau salt and pepper citra bangunan dengan midpoint filter.....	57
Gambar 2.91 Penghilangan derau salt and pepper citra bangunan dengan alpha-trimmed mean filter dengan d = 1.....	57
Gambar 2.92 Penghilangan derau salt and pepper citra jam dengan min filter.....	58
Gambar 2.93 Penghilangan derau salt and pepper citra jam dengan max filter.....	58
Gambar 2.94 Penghilangan derau salt and pepper citra jam dengan median filter.....	58
Gambar 2.95 Penghilangan derau salt and pepper citra jam dengan arithmetic mean filter.....	59
Gambar 2.96 Penghilangan derau salt and pepper citra jam dengan geometric mean filter.....	59
Gambar 2.97 Penghilangan derau salt and pepper citra jam dengan harmonic mean filter.....	59
Gambar 2.98 Penghilangan derau salt and pepper citra jam dengan contraharmonic mean filter dengan Q = 1.5.....	60
Gambar 2.99 Penghilangan derau salt and pepper citra jam dengan midpoint filter.....	60
Gambar 2.100 Penghilangan derau salt and pepper citra jam dengan alpha-trimmed mean filter dengan d = 1.....	60
Gambar 2.101 Penghilangan derau salt and pepper citra garam dengan min filter.....	61
Gambar 2.102 Penghilangan derau salt and pepper garam dengan max filter.....	61
Gambar 2.103 Penghilangan derau salt and pepper citra garam dengan median filter.....	61
Gambar 2.104 Penghilangan derau salt and pepper citra garam dengan arithmetic mean filter...	62
Gambar 2.105 Penghilangan derau salt and pepper citra garam dengan geometric mean filter....	62
Gambar 2.106 Penghilangan derau salt and pepper citra garam dengan harmonic mean filter....	62
Gambar 2.107 Penghilangan derau salt and pepper citra garam dengan contraharmonic mean	

filter dengan Q = 1.5.....	63
Gambar 2.108 Penghilangan derau salt and pepper citra garam dengan midpoint filter.....	63
Gambar 2.109 Penghilangan derau salt and pepper citra garam dengan alpha-trimmed mean filter dengan d = 1.....	63
Gambar 2.110 Penghilangan derau salt and pepper citra kupu-kupu dengan min filter.....	64
Gambar 2.111 Penghilangan derau salt and pepper kupu-kupu dengan max filter.....	64
Gambar 2.112 Penghilangan derau salt and pepper kupu-kupu dengan median filter.....	64
Gambar 2.113 Penghilangan derau salt and pepper citra kupu-kupu dengan arithmetic mean filter	
64	
Gambar 2.114 Penghilangan derau salt and pepper citra kupu-kupu dengan geometric mean filter	
65	
Gambar 2.115 Penghilangan derau salt and pepper citra kupu-kupu dengan harmonic mean filter.	
65	
Gambar 2.116 Penghilangan derau salt and pepper citra kupu-kupu dengan contraharmonic mean filter dengan Q = 1.5.....	65
Gambar 2.117 Penghilangan derau salt and pepper citra kupu-kupu dengan midpoint filter.....	66
Gambar 2.118 Penghilangan derau salt and pepper citra kupu-kupu dengan alpha-trimmed mean filter dengan d = 1.....	66
Gambar 2.119 Penghilangan derau periodik citra Lenna (2).....	69
Gambar 2.120 Penghilangan derau periodik citra peralatan lab.....	69
Gambar 2.121 Penghilangan derau periodik citra perbesaran citra 6-2.....	70
Gambar 2.122 Penghilangan derau periodik citra angsa.....	70
Gambar 2.123 Penghilangan derau periodik citra bangunan.....	71
Gambar 2.124 Penghilangan derau periodik citra astronot di bulan.....	71
Gambar 2.125 Penghilangan derau periodik citra mobil.....	72
Gambar 2.126 Motion blurring dan dekonvolusi citra balon udara.....	75
Gambar 2.127 Motion blurring dan dekonvolusi citra mawar (7-4.tif).....	76
Gambar 2.128 Motion blurring dan dekonvolusi citra kupu-kupu.....	76
Gambar 2.129 Motion blurring dan dekonvolusi citra mammoth (7-5.jpg).....	77
Gambar 2.130 Dekonvolusi citra anak.....	77

1. GUI

Berikut adalah tangkapan layar untuk tampilan GUI.



Gambar 1.1 GUI Program

2. Penjelasan Program

2.1. Konvolusi

Berikut merupakan kode program fungsi untuk melakukan konvolusi citra f berukuran sembarang ($M \times N$) dengan *mask* berukuran $n \times n$.

```
function outputImage = convolution(image, convMatrix)
    [rows, cols, colorChannels] = size(image);
    [rowsConv, colsConv] = size(convMatrix);

    outputImage = zeros(rows, cols, 'uint8');
    for i = ((rowsConv+1)/2):(rows-((rowsConv-1)/2))
        for j = ((colsConv+1)/2):(cols-((colsConv-1)/2))
            for k = 1:colorChannels
                temp = 0;
                startIdxRow = i-(rowsConv+1)/2;
                startIdxCols = j-(colsConv+1)/2;
                for a = 1:rowsConv
                    for b = 1:colsConv
                        temp = temp + image(a + startIdxRow, b +
startIdxCols, k)*convMatrix(a,b);
                    end
                end

                if temp < 0
                    outputImage(i,j,k) = 0;
                elseif temp > 255
```

```

        outputImage(i,j,k) = 255;
    else
        outputImage(i,j,k) = uint8(temp);
    end
end
end
end

```

Inti dari teknik konvolusi adalah mengalikan citra dengan suatu *convolution mask* berukuran $m \times n$. Operasi perkalian ini dilakukan dengan mengiterasi setiap piksel di gambar awal dengan mengabaikan piksel-piksel yang berada di bagian pinggir citra. Terdapat beberapa convolution mask yang digunakan dalam pengujian, yaitu

a. $[0 \ -1 \ 0; \ -1 \ 4 \ -1; \ 0 \ -1 \ 0]$ ($n=3$)

b. $\frac{1}{16}[1 \ 2 \ 1; \ 2 \ 4 \ 2; \ 1 \ 2 \ 1]$ ($n=3$)

$$\frac{1}{140} \times \begin{bmatrix} 1 & 1 & 2 & 2 & 2 & 1 & 1 \\ 1 & 2 & 2 & 4 & 2 & 2 & 1 \\ 2 & 2 & 4 & 8 & 4 & 2 & 2 \\ 2 & 4 & 8 & 16 & 8 & 4 & 2 \\ 2 & 2 & 4 & 8 & 4 & 2 & 2 \\ 1 & 2 & 2 & 4 & 2 & 2 & 1 \\ 1 & 1 & 2 & 2 & 2 & 1 & 1 \end{bmatrix}$$

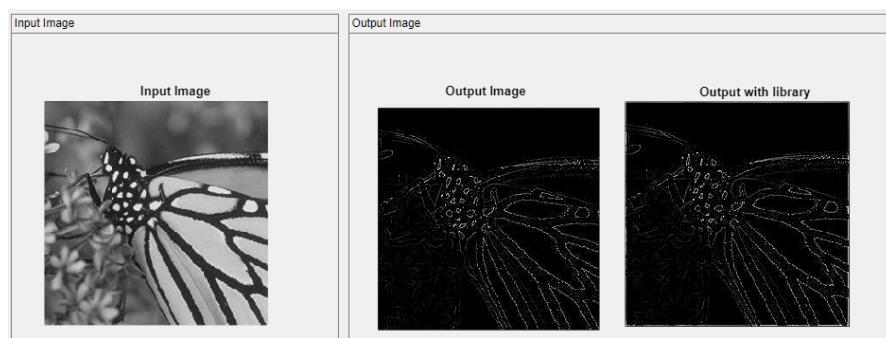
c. $(n=7)$

d. $[-1 \ -1 \ -1; \ -1 \ 17 \ -1; \ -1 \ -1 \ -1]$ ($n=3$)

e. $\frac{1}{25}[1 \ 1 \ 1 \ 1 \ 1; \ 1 \ 1 \ 1 \ 1 \ 1; \ 1 \ 1 \ 1 \ 1 \ 1; \ 1 \ 1 \ 1 \ 1 \ 1; \ 1 \ 1 \ 1 \ 1 \ 1]$ ($n=5$)

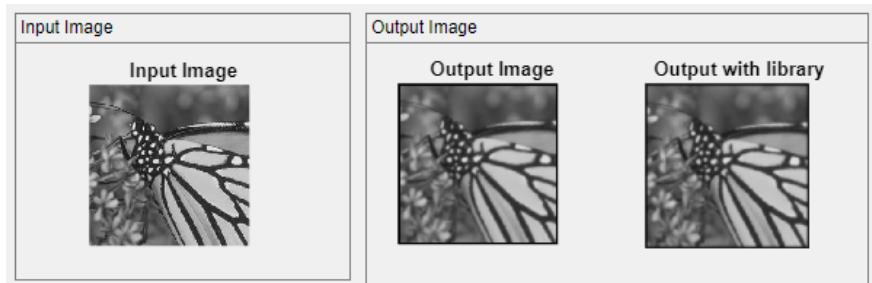
Berikut merupakan contoh hasil eksekusi program.

Citra *grayscale* kupu-kupu (1-1.jpg) dengan $n = 3$ (a):



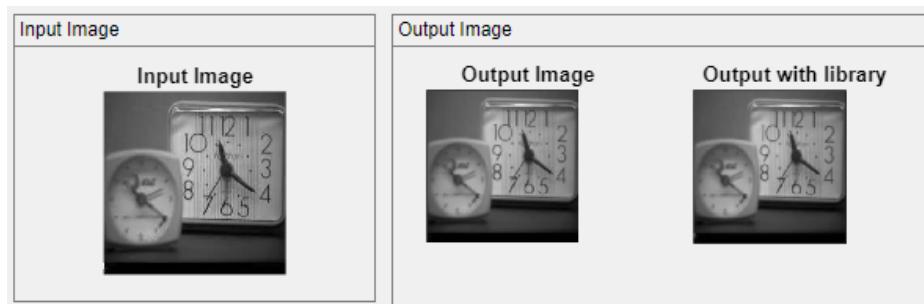
Gambar 2.1 Konvolusi citra kupu-kupu dengan $n = 3$ (a)

Citra *grayscale* kupu-kupu (1-1.jpg) dengan $n = 7$ (c):



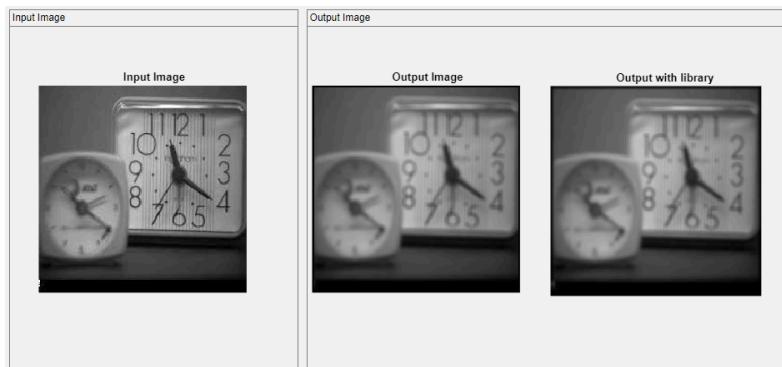
Gambar 2.2 Konvolusi citra kupu-kupu dengan $n = 7$ (c)

Citra *grayscale* jam (1-3.jpg) dengan $n = 3$ (b):



Gambar 2.3 Konvolusi citra jam dengan $n = 3$ (b)

Citra *grayscale* jam (1-3.jpg) dengan $n = 5$ (e):



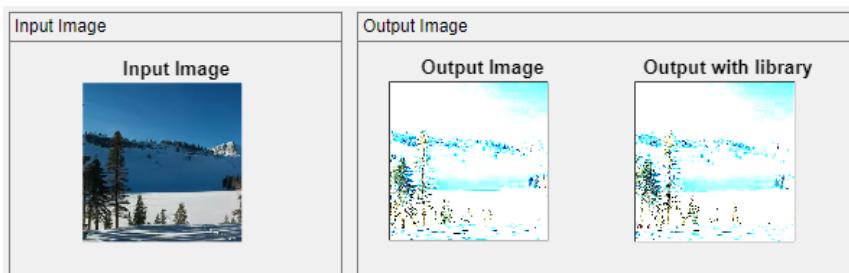
Gambar 2.4 Konvolusi citra kupu-kupu dengan $n = 5$ (e)

Citra grayscale Barbara (1-5.jpg) dengan $n = 5$ (e):



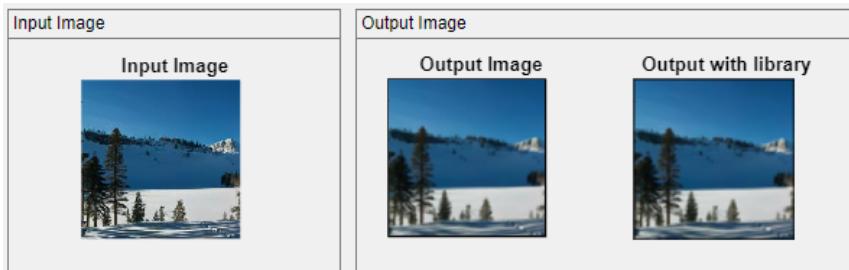
Gambar 2.5 Konvolusi citra Barbara dengan $n = 5$ (e)

Citra berwarna pemandangan salju (1-2.jpg) dengan $n = 3$ (d):



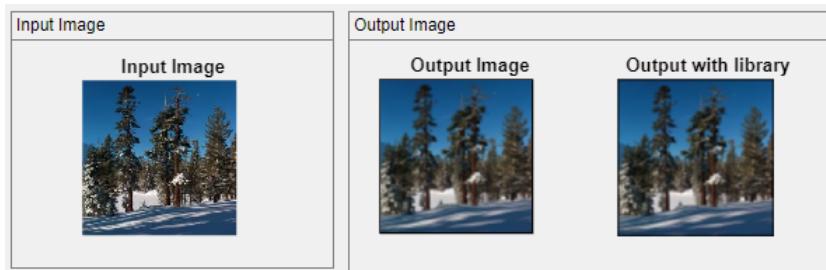
Gambar 2.6 Konvolusi citra pemandangan salju dengan $n = 3$ (d)

Citra berwarna pemandangan salju (1-2.jpg) dengan $n = 5$ (e):



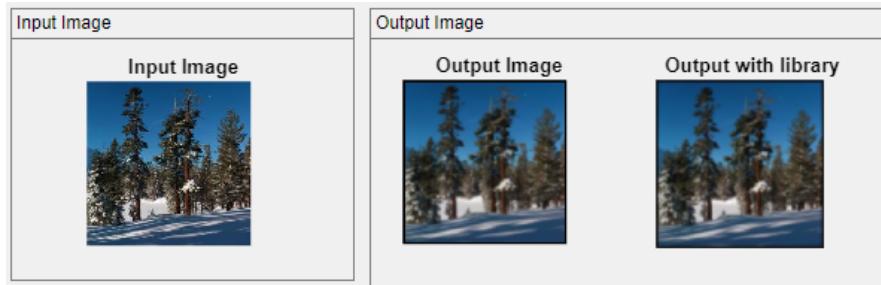
Gambar 2.7 Konvolusi citra pemandangan salju dengan $n = 5$

Citra berwarna pemandangan pohon (1-4.jpg) dengan $n = 5$ (e):



Gambar 2.8 Konvolusi citra pemandangan pohon dengan $n = 5$ (e)

Citra berwarna pemandangan pohon (1-4.jpg) dengan $n = 7$ (b):



Gambar 2.9 Konvolusi citra pemandangan pohon dengan $n = 7$ (b)

Citra berwarna peppers (1-6.jpg) dengan $n = 3$ (a)



Gambar 2.10 Konvolusi citra peppers dengan $n = 3$ (a)

Analisis:

Gambar yang dihasilkan adalah gambar yang sudah dikalikan dengan suatu *convolution mask*. Dapat dilihat bahwa semakin besar dimensi matriks dari *convolution mask*, akan semakin jelas pula efek yang diberikan. Gambar yang dikalikan dengan Dapat dilihat juga bahwa hasil dari kode yang diimplementasikan sendiri tidak jauh berbeda dari hasil dengan *library*. Dari hasil konvolusi juga dapat dilihat bahwa semakin besar dimensi matriks konvolusi, akan semakin lebar garis hitam di sekeliling citra.

2.2. Pelembutan Citra pada Ranah Spasial

Berikut merupakan kode program fungsi untuk melakukan *image smoothing* atau *blurring* pada citra dalam ranah spasial menggunakan *mean filter* $n \times n$ dan *gaussian filter* $n \times n$.

```
function smoothedImage = spatialSmooth(image, filter, filterSize, sigma)
    % Membuat mean filter n x n berdasarkan jenis filter
    if lower(filter) == "mean"
        kernel = ones(filterSize) / (filterSize * filterSize);
```

```

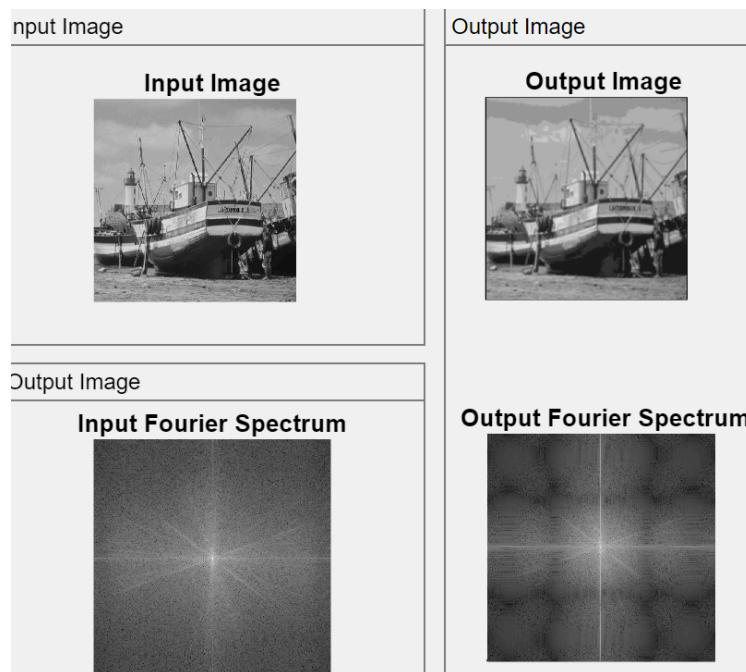
elseif lower(filter) == "gaussian"
    [x, y] = meshgrid(-(filterSize-1)/2:(filterSize-1)/2,
    -(filterSize-1)/2:(filterSize-1)/2);
    kernel = exp(-(x.^2 + y.^2) / (2 * sigma^2));
    kernel = kernel / sum(kernel(:)); % Normalisasi filter
end
% Melakukan konvolusi
smoothedImage = convolution(image, kernel);
end

```

Fungsi untuk melakukan *smoothing* citra pada ranah spasial terdapat pada fungsi `spatialSmooth` dengan parameter citra (`image`), jenis *filter*-nya (`filter`, *mean* atau *gaussian*), ukuran *filter* (*n* atau `filterSize`), dan derajat pelembutan (`sigma`).

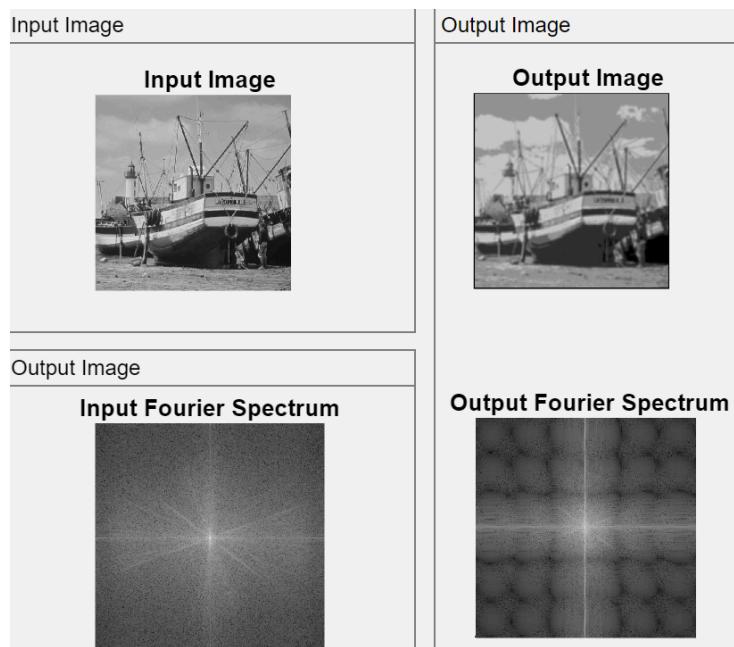
Berikut merupakan contoh hasil eksekusi program.

Mean filter citra *grayscale boat* (2-1.jpg) dengan *n* = 5:



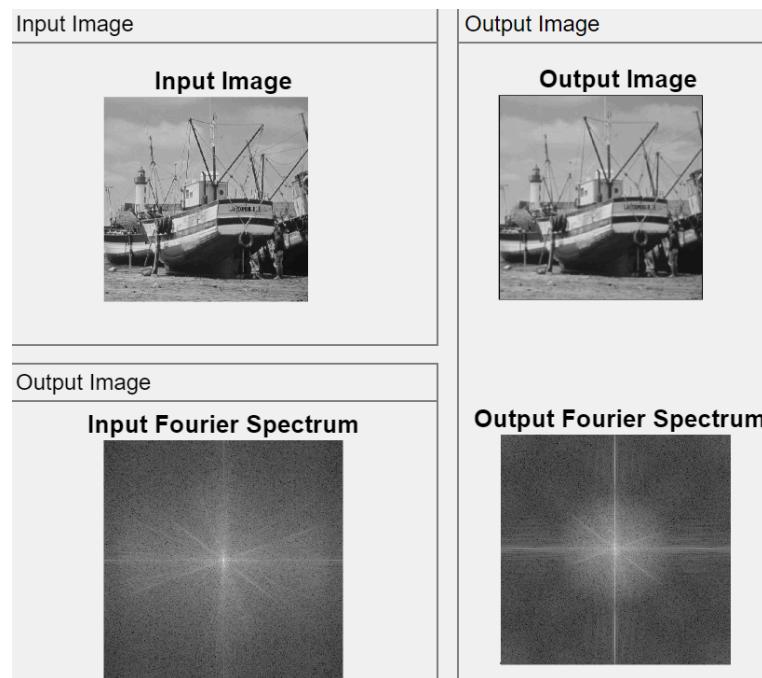
Gambar 2.11 *Mean filter* citra *boat* dengan *n* = 5

Mean filter citra grayscale boat (2-1.jpg) dengan $n = 7$:



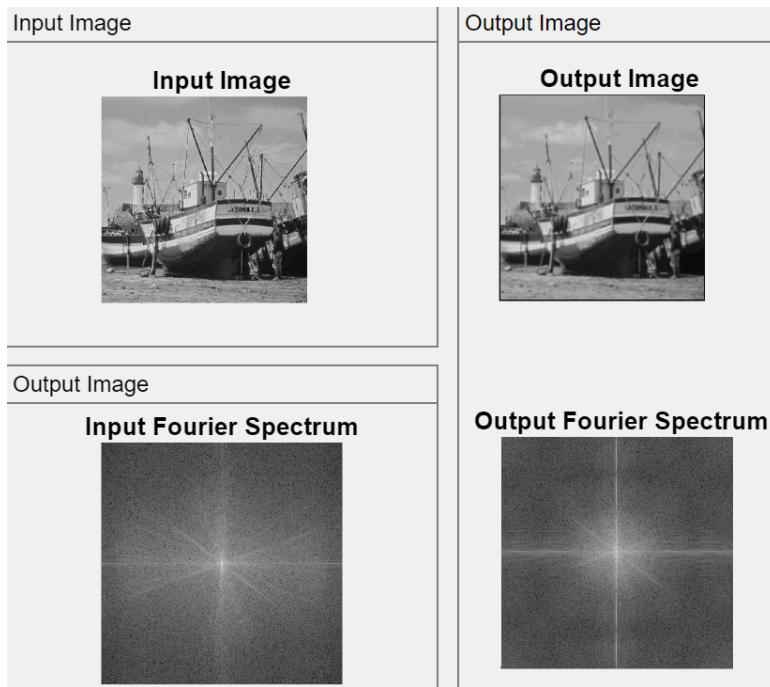
Gambar 2.12 *Mean filter* citra boat dengan $n = 7$

Gaussian filter citra grayscale boat (2-1.jpg) dengan $n = 5$ dan $\sigma = 1.5$:



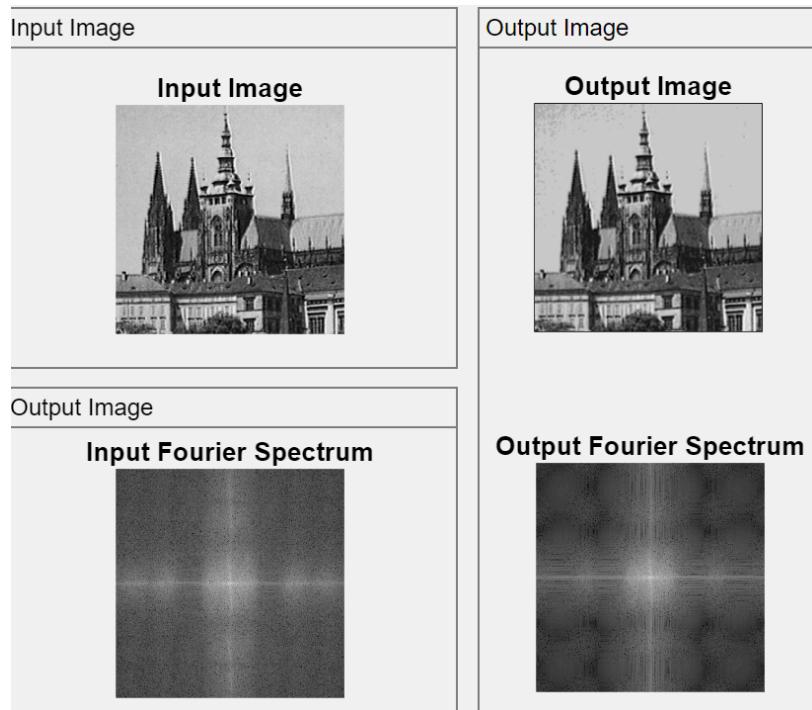
Gambar 2.13 *Gaussian filter* citra boat dengan $n = 5$ dan $\sigma = 1.5$

Gaussian filter citra grayscale boat (2-1.jpg) dengan $n = 5$ dan $\sigma = 1.5$:



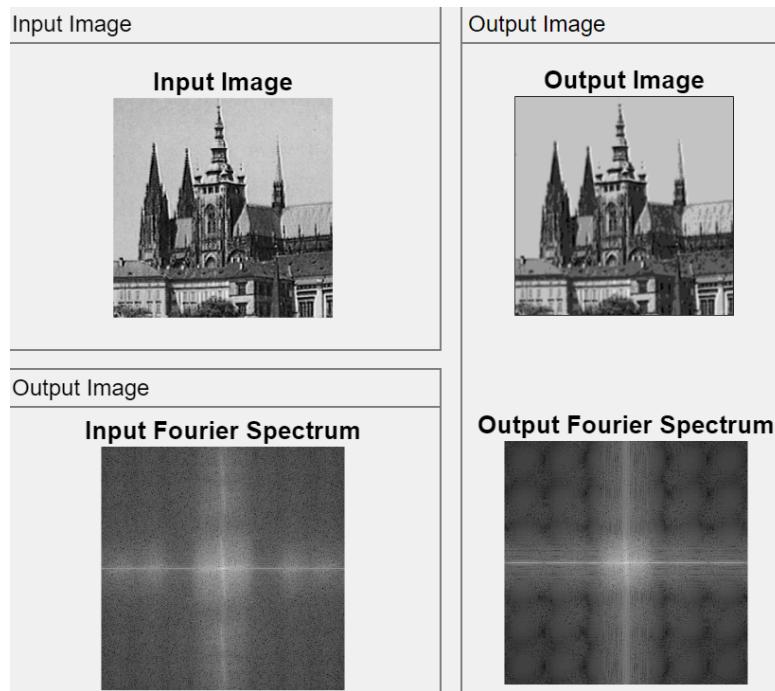
Gambar 2.14 *Gaussian filter* citra boat dengan $n = 7$ dan $\sigma = 1.5$

Mean filter citra grayscale bangunan (2-2.jpg) dengan $n = 5$:



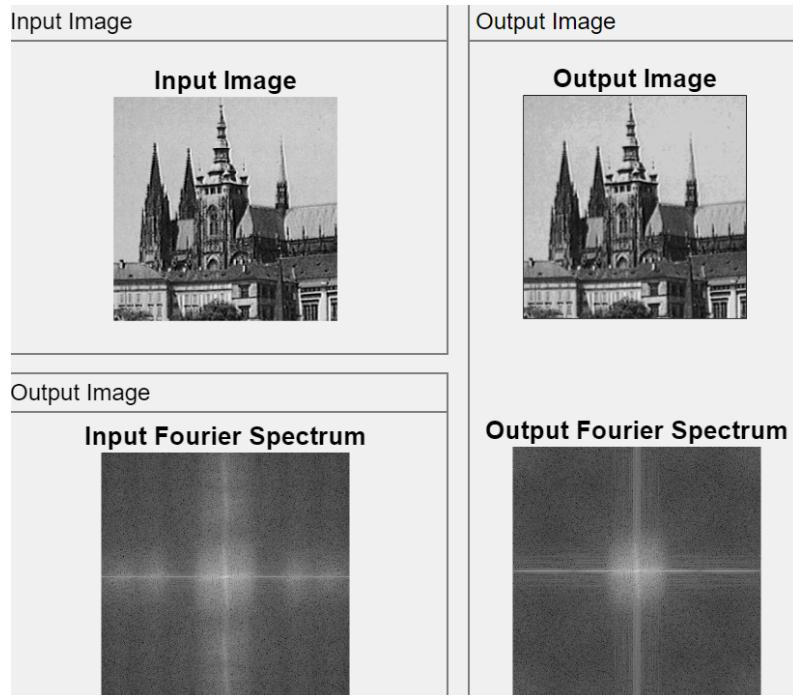
Gambar 2.15 *Mean filter* citra bangunan dengan $n = 5$

Mean filter citra grayscale bangunan (2-2.jpg) dengan $n = 7$:



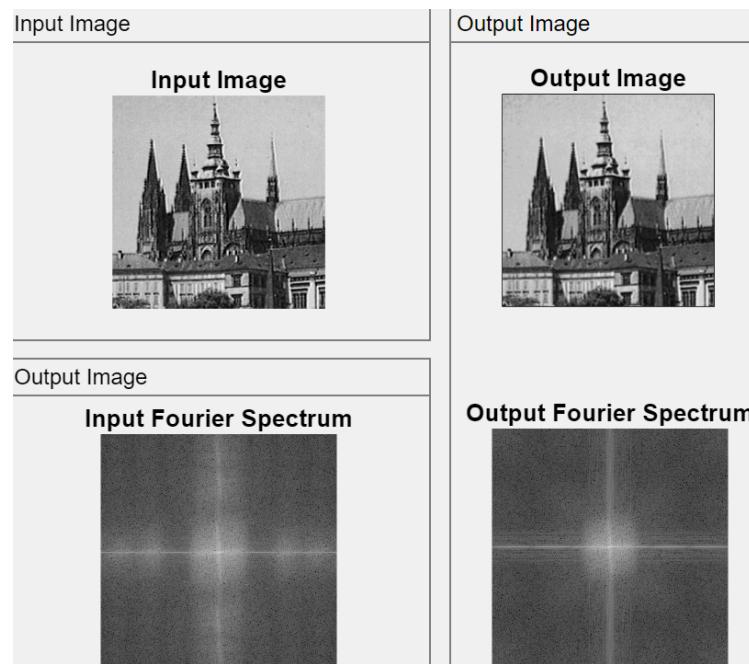
Gambar 2.16 *Mean filter* citra bangunan dengan $n = 7$

Gaussian filter citra grayscale bangunan (2-2.jpg) dengan $n = 5$ dan $\sigma = 1.5$:



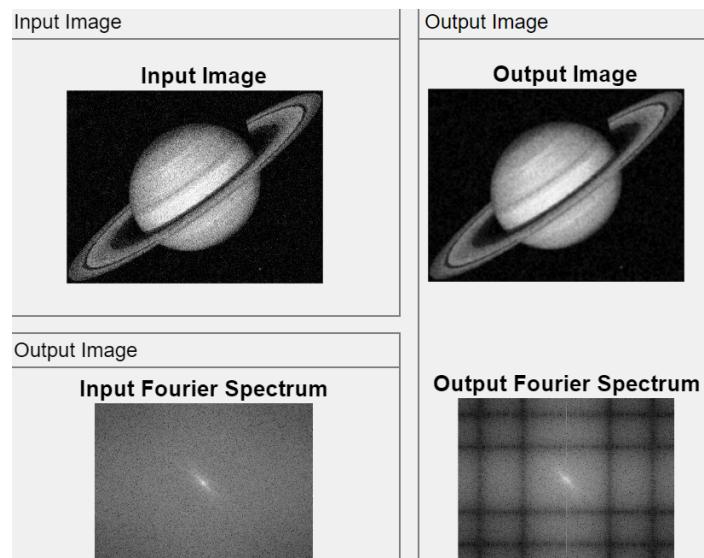
Gambar 2.17 *Gaussian filter* citra bangunan dengan $n = 5$ dan $\sigma = 1.5$

Gaussian filter citra grayscale bangunan (2-2.jpg) dengan $n = 7$ dan $\sigma = 1.5$:



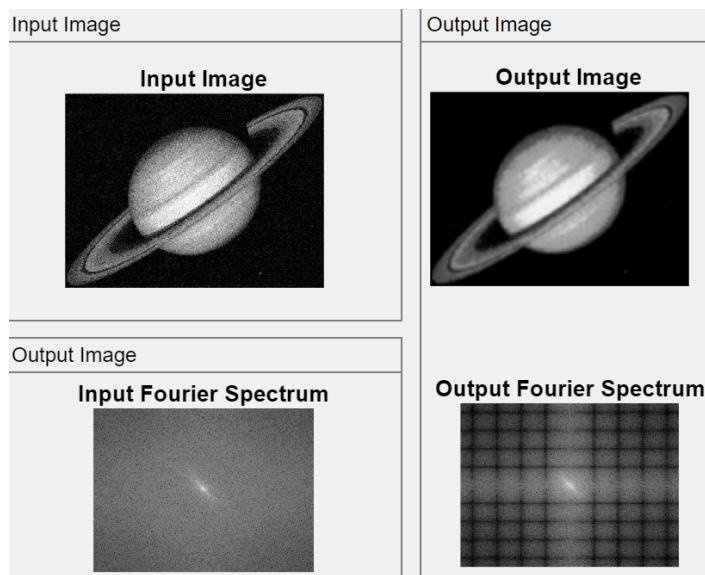
Gambar 2.18 *Gaussian filter* citra bangunan dengan $n = 7$ dan $\sigma = 1.5$

Mean filter citra grayscale Planet Saturnus (2-3.jpg) dengan $n = 5$:



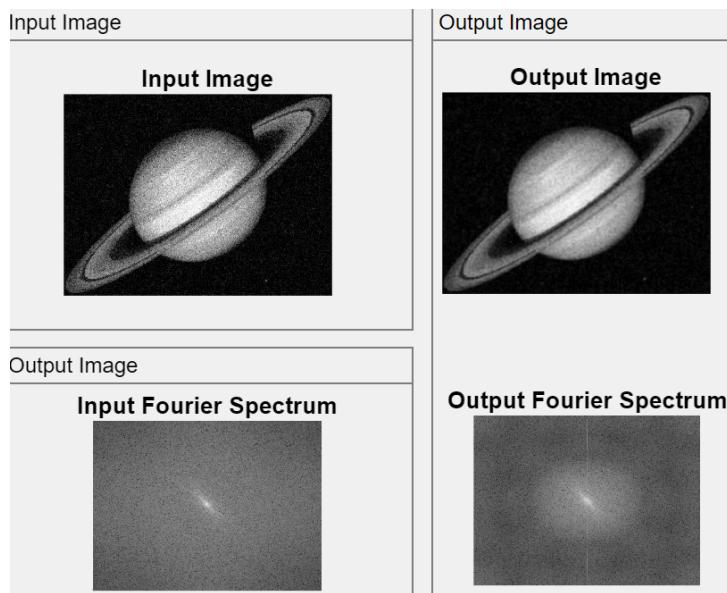
Gambar 2.19 *Mean filter* citra Planet Saturnus dengan $n = 5$

Mean filter citra grayscale Planet Saturnus (2-3.jpg) dengan $n = 9$:



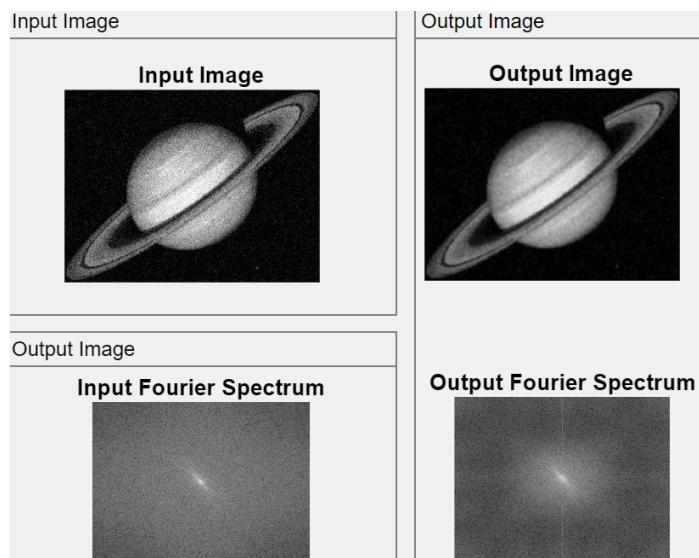
Gambar 2.20 *Mean filter* citra Planet Saturnus dengan $n = 9$

Gaussian filter citra grayscale Planet Saturnus (2-3.jpg) dengan $n = 5$ dan $\sigma = 1.5$:



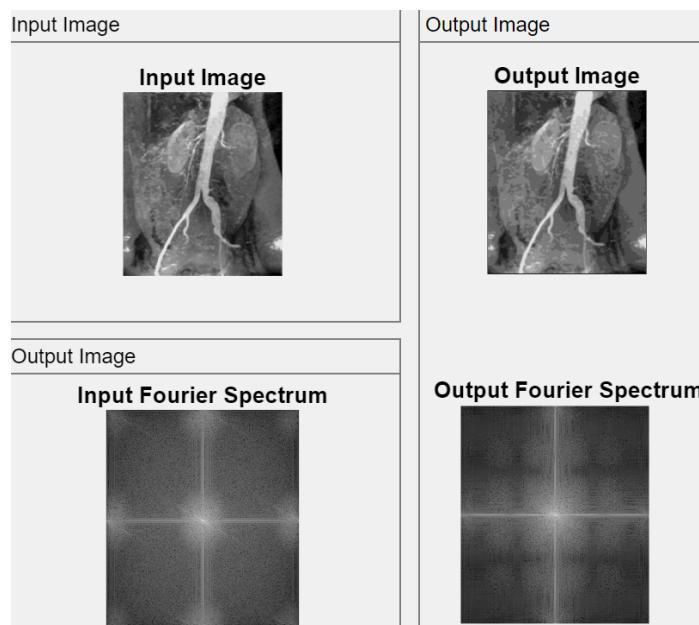
Gambar 2.21 *Gaussian filter* citra Planet Saturnus dengan $n = 5$ dan $\sigma = 1.5$

Gaussian filter citra grayscale Planet Saturnus (2-3.jpg) dengan $n = 9$ dan $\sigma = 1.5$:



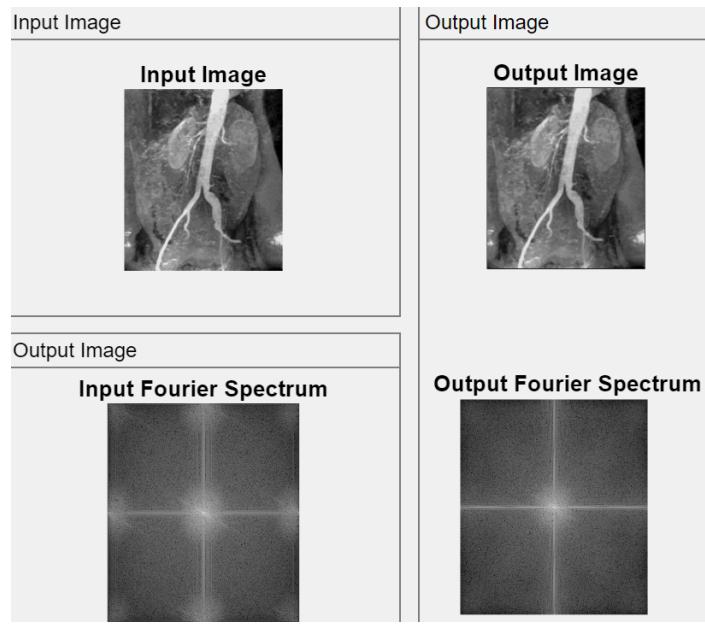
Gambar 2.22 *Gaussian filter* citra Planet Saturnus dengan $n = 9$ dan $\sigma = 1.5$

Mean filter citra grayscale ginjal (kidney.bmp) dengan $n = 5$:



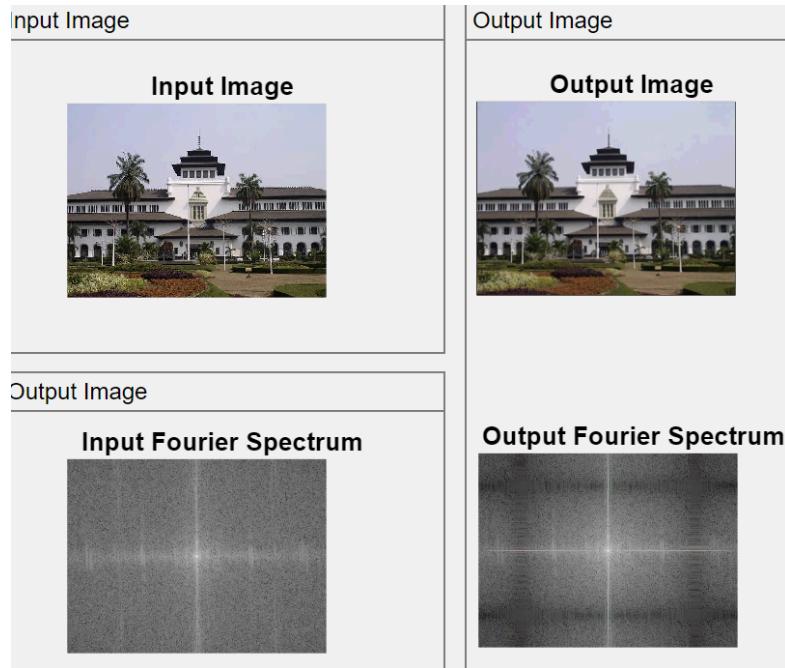
Gambar 2.23 *Mean filter* citra ginjal dengan $n = 5$

Gaussian filter citra grayscale ginjal (kidney.bmp) dengan $n = 5$ dan $\sigma = 1.5$:



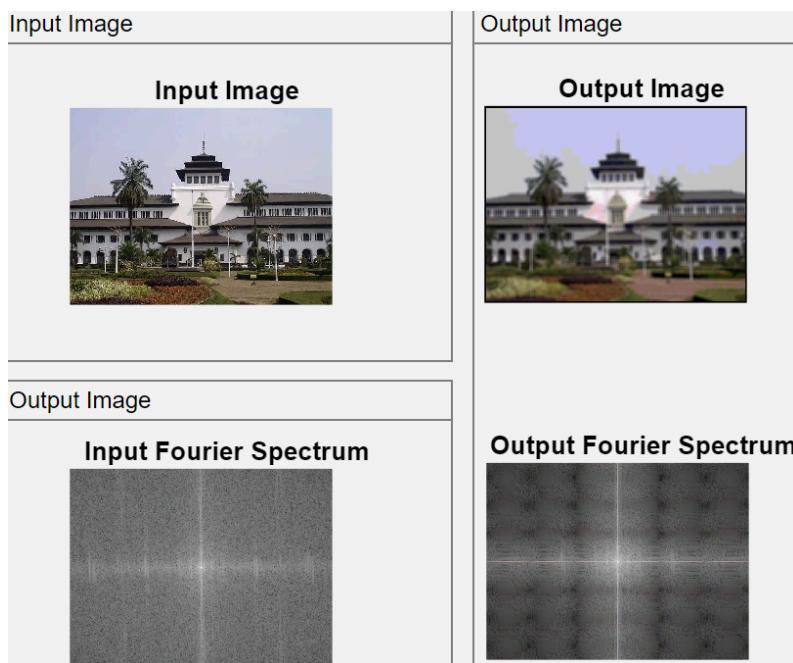
Gambar 2.24 *Gaussian filter* citra ginjal dengan $n = 5$ dan $\sigma = 1.5$

Mean filter citra berwarna Gedung Sate (2-4.jpg) dengan $n = 3$:



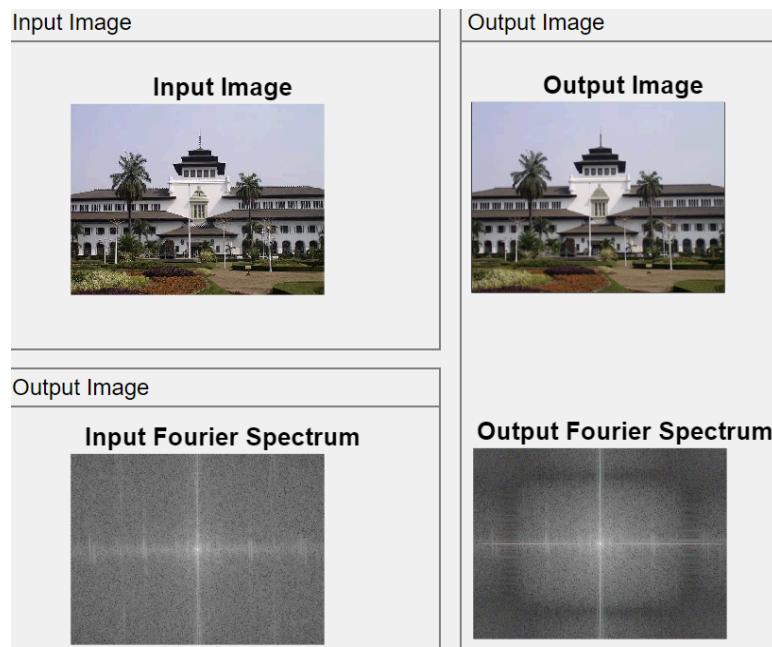
Gambar 2.25 *Mean filter* citra Gedung Sate dengan $n = 3$

Mean filter citra berwarna Gedung Sate (2-4.jpg) dengan $n = 7$:



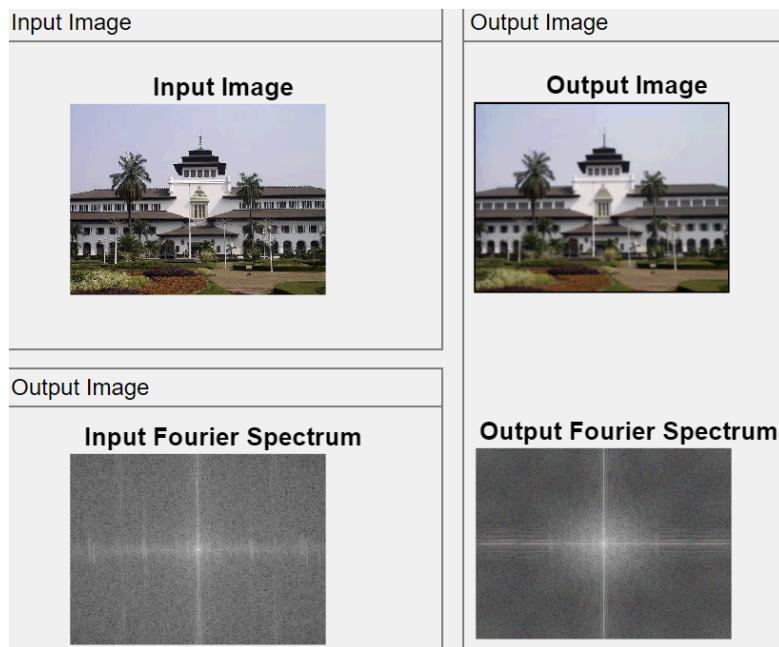
Gambar 2.26 *Mean filter* citra Gedung Sate dengan $n = 7$

Gaussian filter citra berwarna Gedung Sate (2-4.jpg) dengan $n = 3$ dan $\sigma = 1.5$:



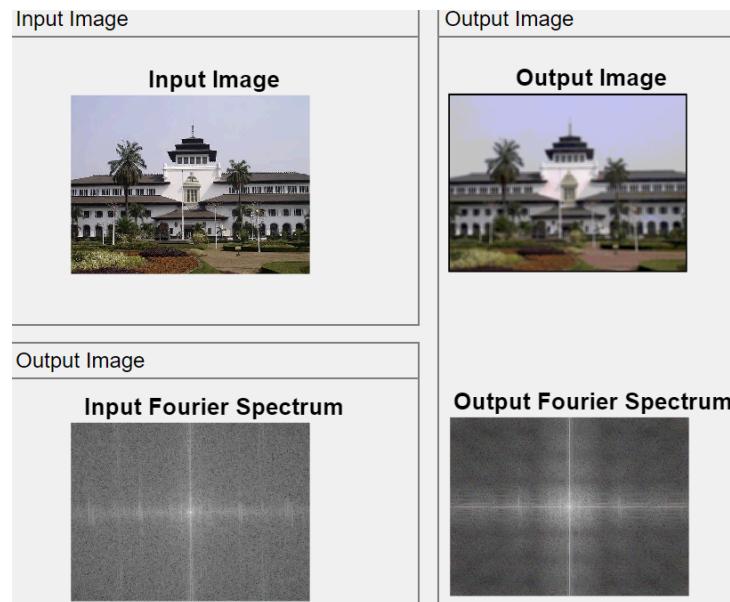
Gambar 2.27 *Gaussian filter* citra Gedung Sate dengan $n = 3$ dan $\sigma = 1.5$

Gaussian filter citra berwarna Gedung Sate (2-4.jpg) dengan $n = 7$ dan $\sigma = 1.5$:



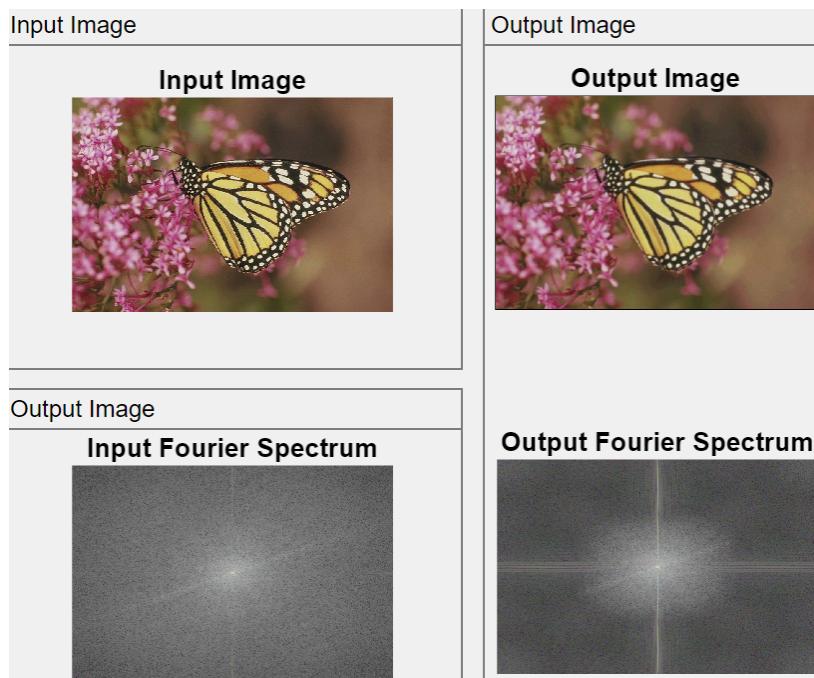
Gambar 2.28 *Gaussian filter* citra Gedung Sate dengan $n = 7$ dan $\sigma = 1.5$

Gaussian filter citra berwarna Gedung Sate (2-4.jpg) dengan $n = 7$ dan $\sigma = 13$:



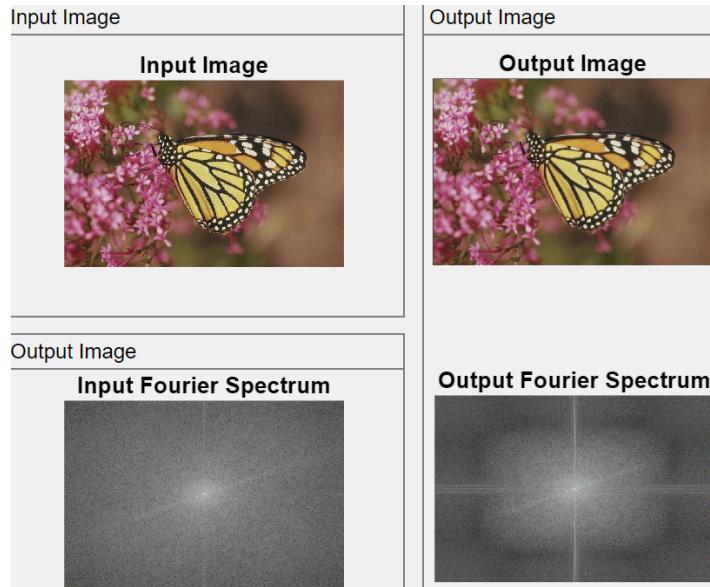
Gambar 2.29 *Gaussian filter* citra Gedung Sate dengan $n = 7$ dan $\sigma = 13$

Mean filter citra berwarna kupu-kupu (2-5.jpg) dengan $n = 5$:



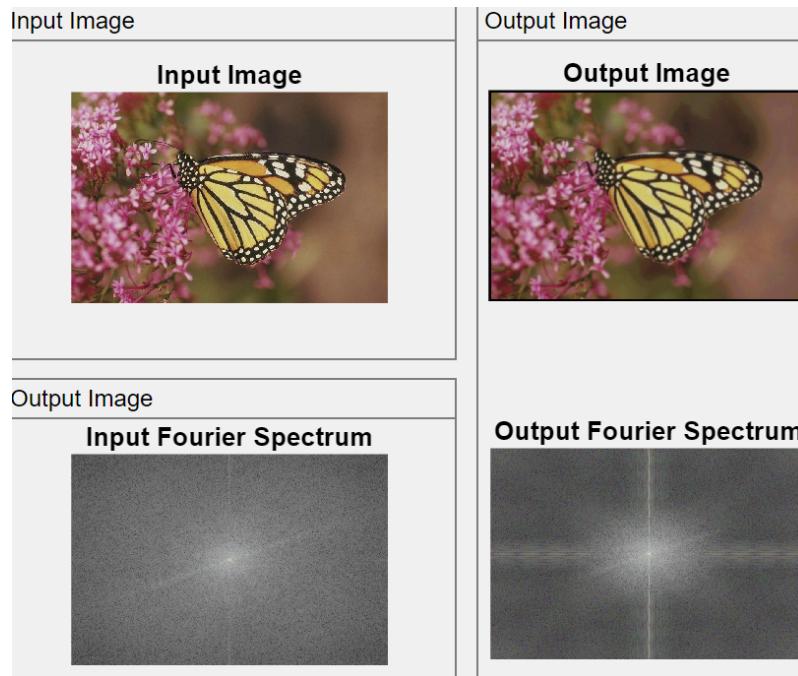
Gambar 2.30 *Mean filter* citra kupu-kupu dengan $n = 5$

Gaussian filter citra berwarna kupu-kupu (2-5.jpg) dengan $n = 3$ dan $\sigma = 1.5$:



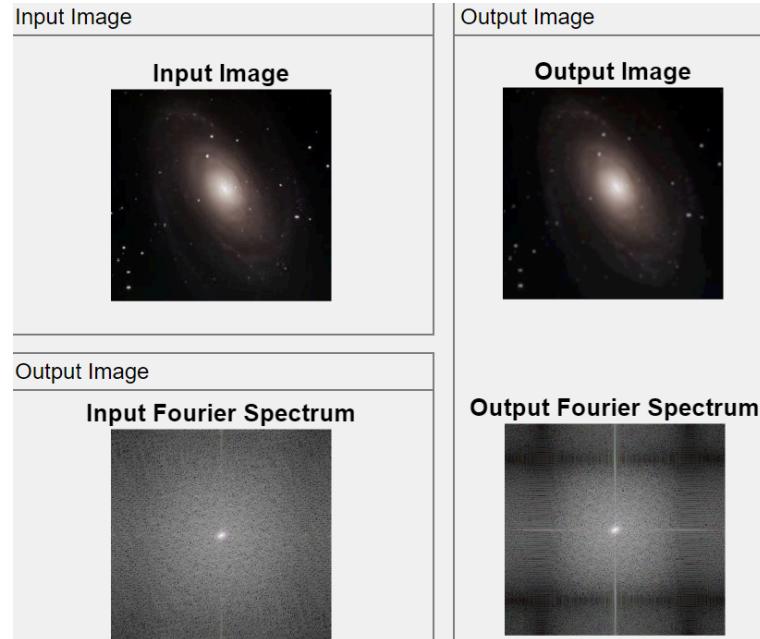
Gambar 2.31 *Gaussian filter* citra kupu-kupu dengan $n = 3$ dan $\sigma = 1.5$

Gaussian filter citra berwarna kupu-kupu (2-5.jpg) dengan $n = 10$ dan $\sigma = 1.5$:



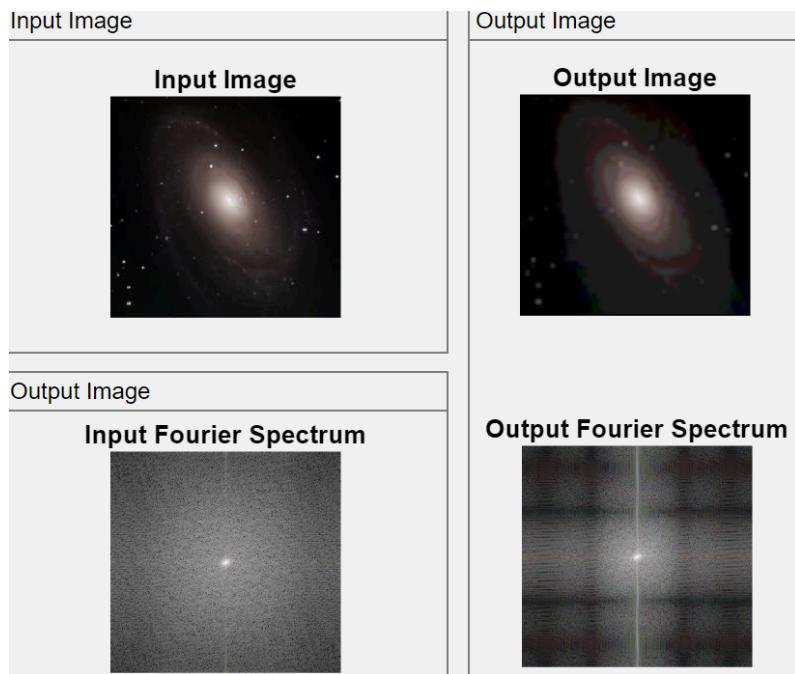
Gambar 2.32 *Gaussian filter* citra kupu-kupu dengan $n = 10$ dan $\sigma = 1.5$

Mean filter citra berwarna galaksi (2-6.jpg) dengan $n = 3$:



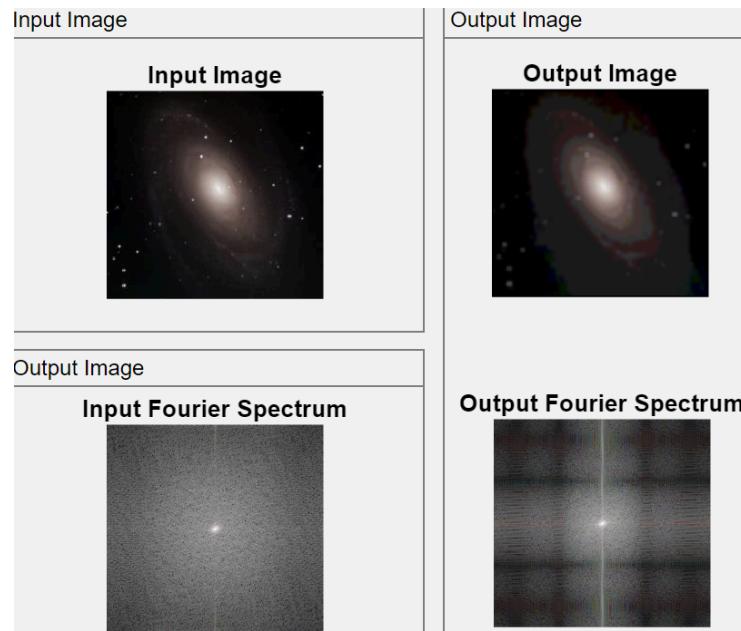
Gambar 2.33 *Mean filter* citra galaksi dengan $n = 3$

Mean filter citra berwarna galaksi (2-6.jpg) dengan $n = 5$:



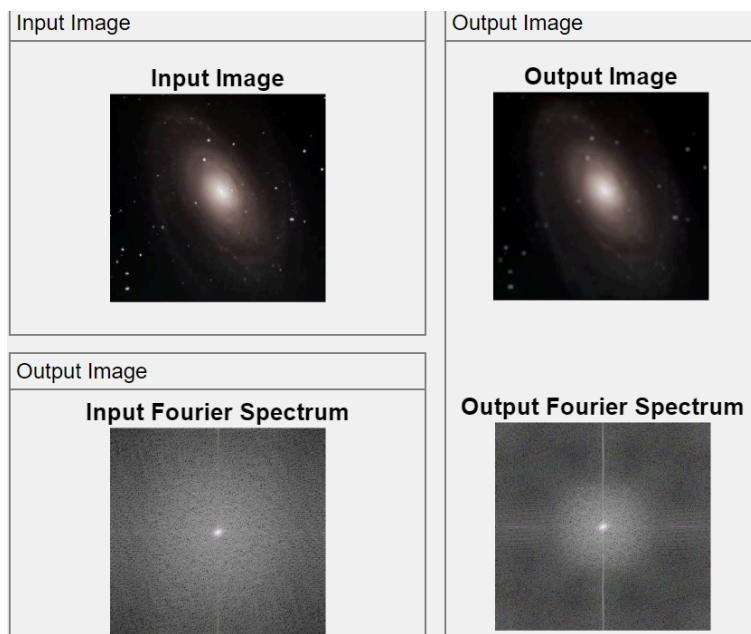
Gambar 2.34 *Mean filter* citra galaksi dengan $n = 5$

Gaussian filter citra berwarna galaksi (2-6.jpg) dengan $n = 5$ dan $\sigma = 50$:



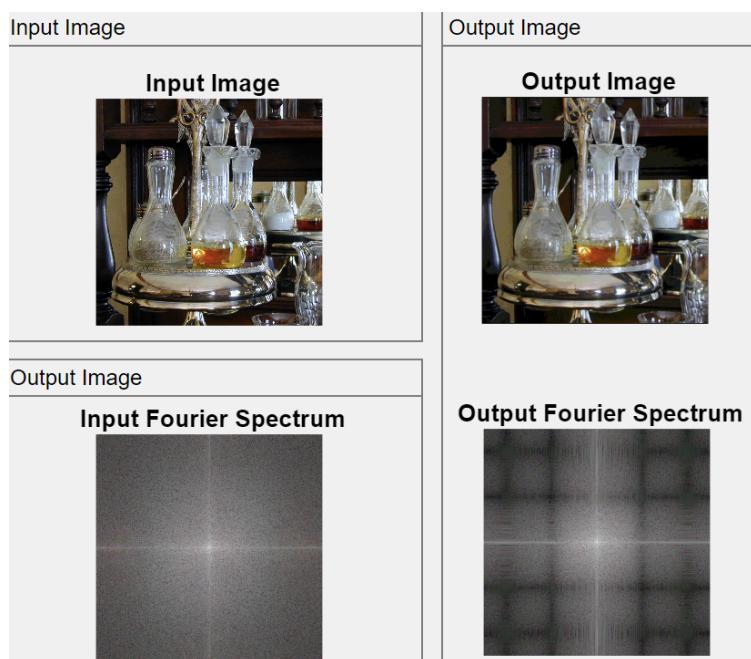
Gambar 2.35 *Gaussian filter* citra galaksi dengan $n = 5$ dan $\sigma = 50$

Gaussian filter citra berwarna galaksi (2-6.jpg) dengan $n = 5$ dan $\sigma = 1.5$:



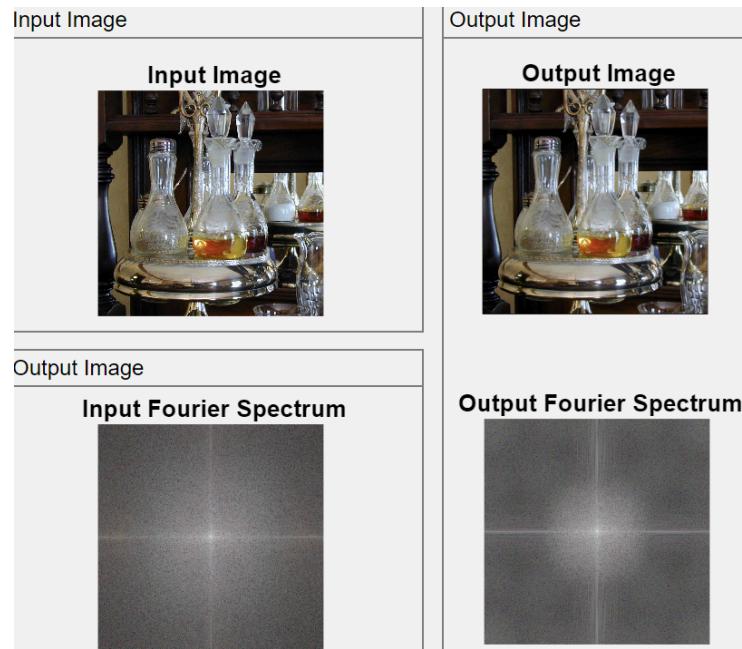
Gambar 2.36 *Gaussian filter* citra galaksi dengan $n = 5$ dan $\sigma = 1.5$

Mean filter citra berwarna *caster* (caster_stand_origin.bmp) dengan $n = 5$:



Gambar 2.37 *Mean filter* citra *caster* dengan $n = 5$

Gaussian filter citra berwarna *caster* (*caster_stand_origin.bmp*) dengan $n = 5$ dan $\sigma = 1.5$:



Gambar 2.38 *Gaussian filter* citra *caster* dengan $n = 5$ dan $\sigma = 1.5$

Analisis:

Pelembutan citra dengan *mean filter* merata-ratakan nilai piksel dalam lingkungan yang ditentukan sehingga dapat membuat citra menjadi kabur dan mengurangi derau. Sebaliknya, *filter Gaussian* menggunakan *kernel* yang memberikan bobot lebih pada piksel pusat dan bobot lebih sedikit pada piksel yang lebih jauh, sehingga memungkinkan transisi yang lebih halus antara nilai piksel. Kedua filter ini efektif dalam mencapai tingkat kaburnya sambil membantu mempertahankan struktur keseluruhan citra.

Semakin besar ukuran *filter* (n), maka efek kabur pada citra akan semakin kuat dan signifikan. Sementara itu, pada *filter Gaussian*, semakin besar derajat pelembutan (σ), maka *kernel* akan semakin lebar dan transisi nilai piksel semakin lembut dan efek kaburnya semakin halus.

2.3. Pelembutan Citra pada Ranah Frekuensi: Low-pass Filter

Berikut merupakan kode program untuk melakukan *image smoothing* atau *blurring* pada citra dalam ranah frekuensi dengan *low-pass filter* ILPF, GLPF, dan BLPF.

```

function outputImage = lowPassFilter(image, filter, d0, nInput)
    [rows, cols, colorChannels] = size(image);
    disp(filter);
    % Menentukan parameter padding, biasanya P = 2*rows dan Q = 2*cols
    P = 2*rows;
    Q = 2*cols;
    tempImage = zeros([P Q colorChannels]);
    image = im2double(image);
    fp = zeros([P Q colorChannels]);
    % Melakukan padding dengan nilai 0 untuk pixel di luar image awal
    for c = 1:colorChannels
        for i = 1 : P
            for j = 1 : Q
                if i <= rows && j <= cols
                    fp(i,j,c) = image(i,j,c);
                else
                    fp(i,j,c) = 0;
                end
            end
        end
    end
    % Pemrosesan untuk tiap kanal warna
    for c = 1:colorChannels
        currImg = fp(:,:,c);
        % Transformasi Fourier
        F = fftshift(fft2(currImg));
        % Cut-off frequency
        D0 = d0;
        % Range dari variabel
        u = 0:(P-1);
        v = 0:(Q-1);
        % Index untuk digunakan pada meshgrid
        idx = find(u > P/2);
        u(idx) = u(idx) - P;
        idy = find(v > Q/2);
        v(idy) = v(idy) - Q;
        % Menghitung meshgrid array
        [V, U] = meshgrid(v, u);
        D = sqrt(U.^2 + V.^2);
        H = 0;
        % Melakukan filtering berdasarkan jenis filter low-pass
        if filter == "Gaussian"
            H = exp(-(D.^2)./(2*(D0.^2)));
        elseif filter == "Ideal"
            H = double(D <= D0);
        elseif filter == "Butterworth"
            n = nInput;
            H = 1 ./ (1 + (D ./ D0).^^(2*n));
        end
        H = fftshift(H);
        G = H .* F;
        G1 = ifftshift(G);
        G2 = real(ifft2(G1));
        tempImage(:,:,:,c) = G2(:,:,:,:,1);
    end
end

```

```

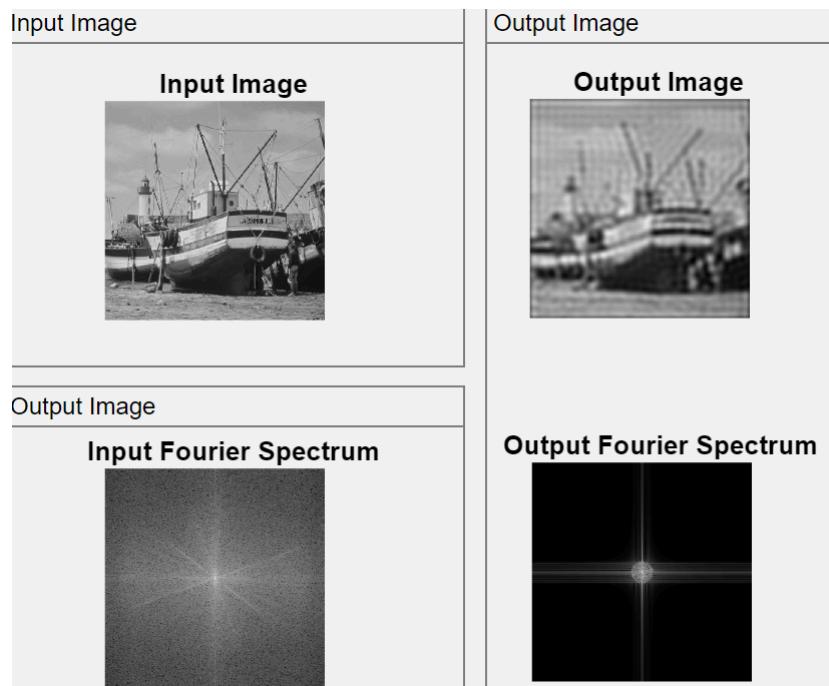
    end
    outputImage = tempImage(1:rows, 1:cols, :);
end

```

Fungsi `lowPassFilter` menerapkan penapis lolos rendah pada citra dengan menerima parameter citra (`image`), jenis penapis (`filter`), frekuensi *cut-off* (`d0`), dan parameter tambahan untuk filter Butterworth, yaitu orde penapis (`nInput`). Pertama, citra ditambahkan *padding* menjadi ukuran dua kali lipat dari ukuran aslinya dengan nilai nol di tepi untuk menghindari *aliasing* saat transformasi *Fourier*. Kemudian, fungsi melakukan transformasi *Fourier* pada setiap saluran warna, menghitung jarak dari pusat frekuensi, dan menerapkan penapis sesuai dengan jenis yang dipilih (*Gaussian*, *Ideal*, atau *Butterworth*). Setelah itu, hasil *filtering* dikembalikan ke ranah spasial menggunakan transformasi invers *Fourier*, dan citra keluaran dipotong kembali ke ukuran asli sebelum dikembalikan.

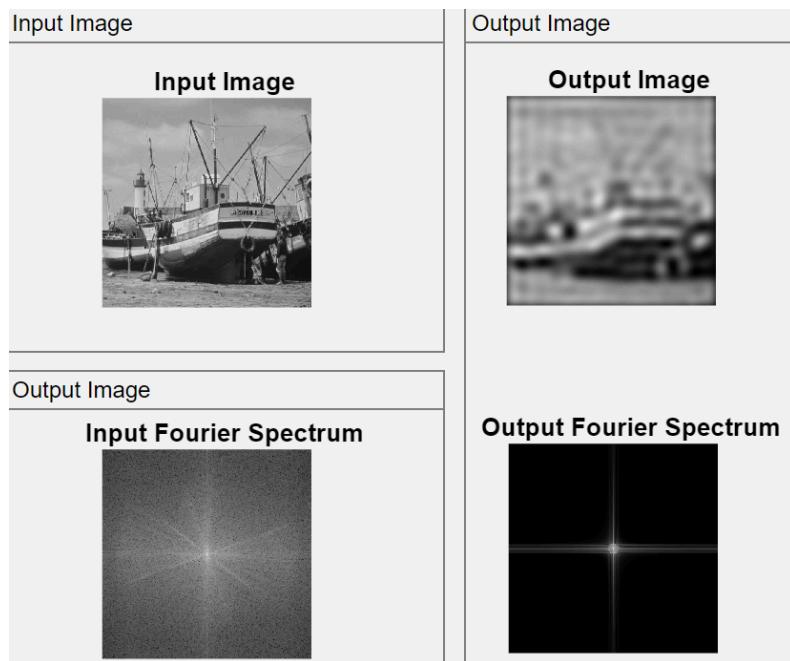
Berikut merupakan contoh hasil eksekusi program.

ILPF citra *grayscale boat* (2-1.jpg) dengan $d0 = 50$:



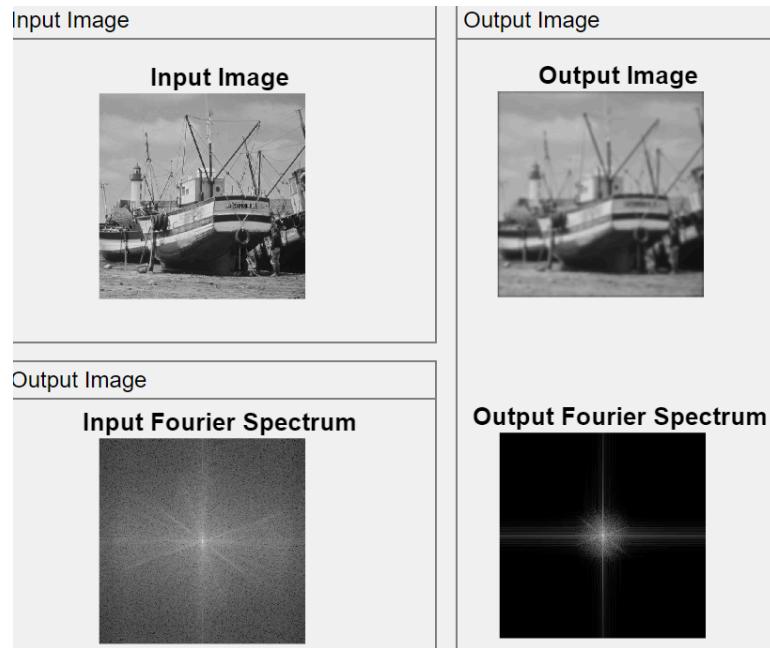
Gambar 2.39 ILPF citra *boat* dengan $d0 = 50$

ILPF citra *grayscale boat* (2-1.jpg) dengan $d_0 = 25$:



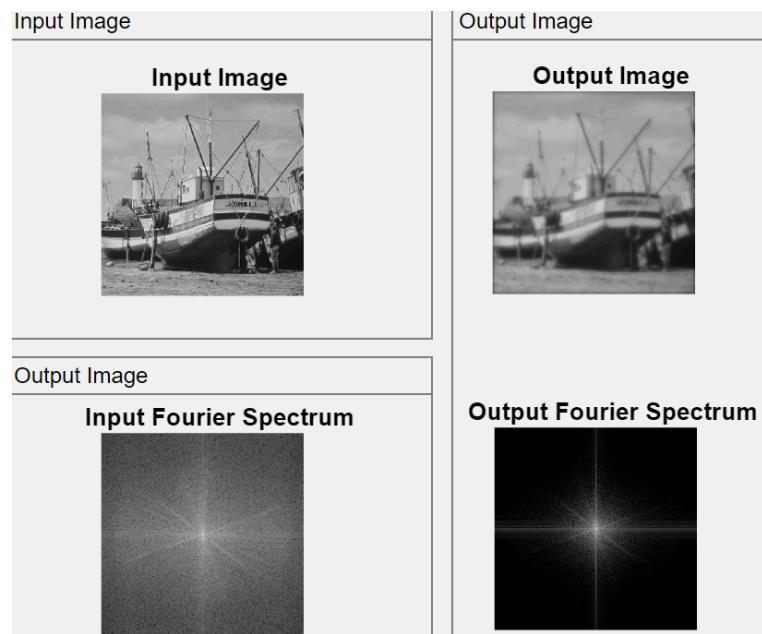
Gambar 2.40 ILPF citra *boat* dengan $d_0 = 25$

GLPF citra *grayscale boat* (2-1.jpg) dengan $d_0 = 50$:



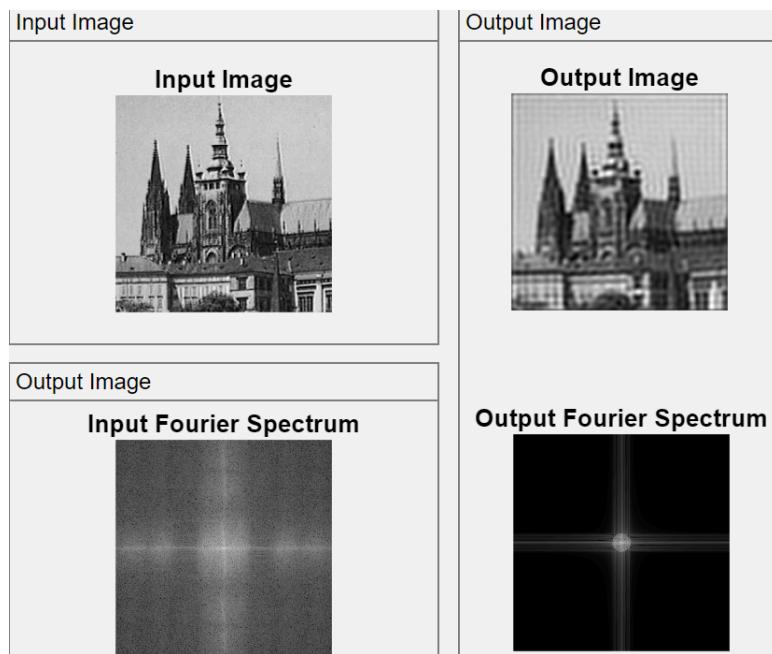
Gambar 2.41 GLPF citra *boat* dengan $d_0 = 50$

BLPF citra *grayscale boat* (2-1.jpg) dengan $d0 = 50$ dan $n = 1$:



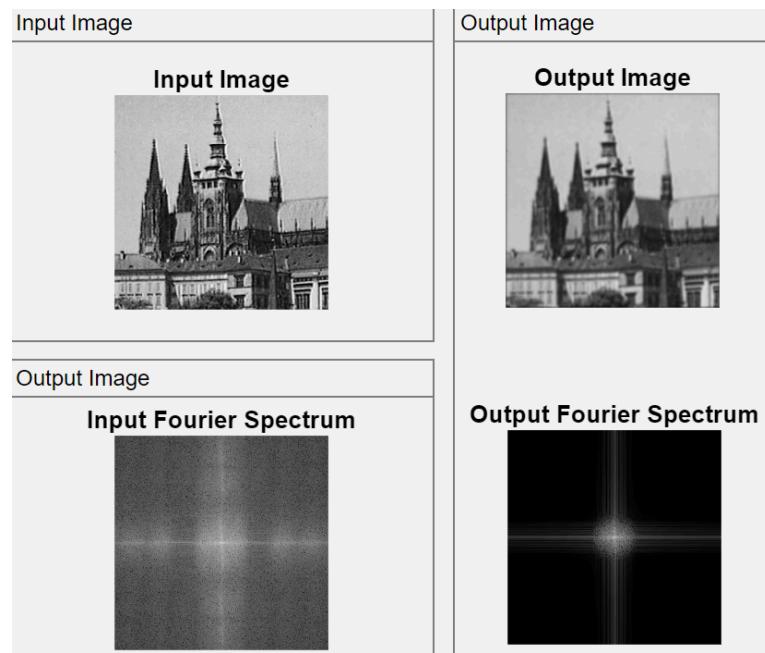
Gambar 2.42 BLPF citra *boat* dengan $d0 = 50$ dan $n = 1$

ILPF citra *grayscale bangunan* (2-2.jpg) dengan $d0 = 60$:



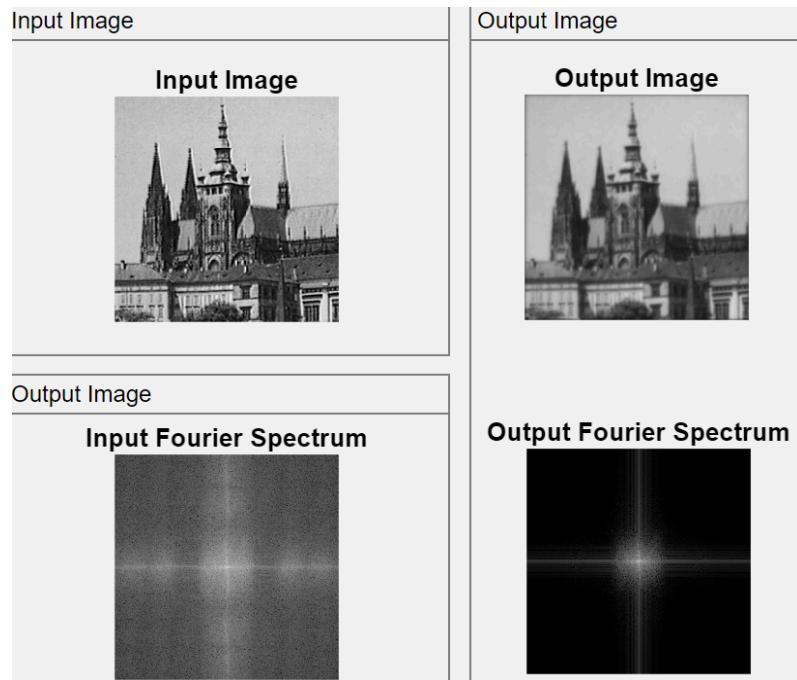
Gambar 2.43 ILPF citra *bangunan* dengan $d0 = 60$

GLPF citra *grayscale* bangunan (2-2.jpg) dengan $d0 = 60$:



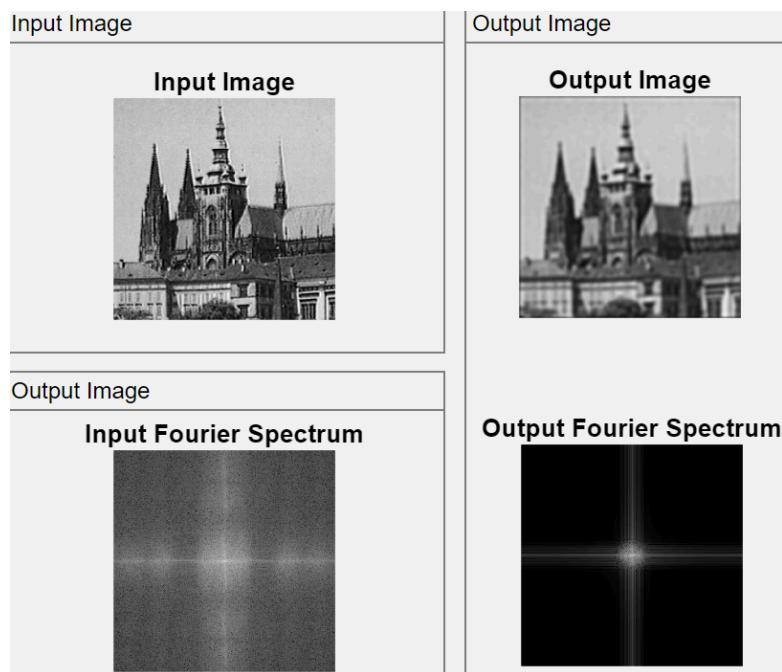
Gambar 2.44 GLPF citra bangunan dengan $d0 = 60$

BLPF citra *grayscale* bangunan (2-2.jpg) dengan $d0 = 60$ dan $n = 1$:



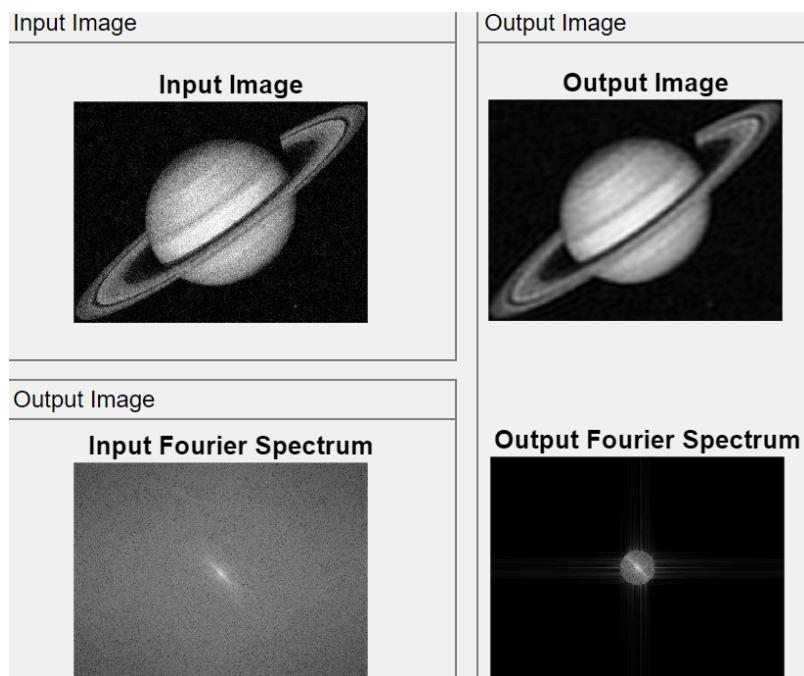
Gambar 2.45 BLPF citra bangunan dengan $d0 = 60$ dan $n = 1$

BLPF citra *grayscale* bangunan (2-2.jpg) dengan $d_0 = 60$ dan $n = 3$:



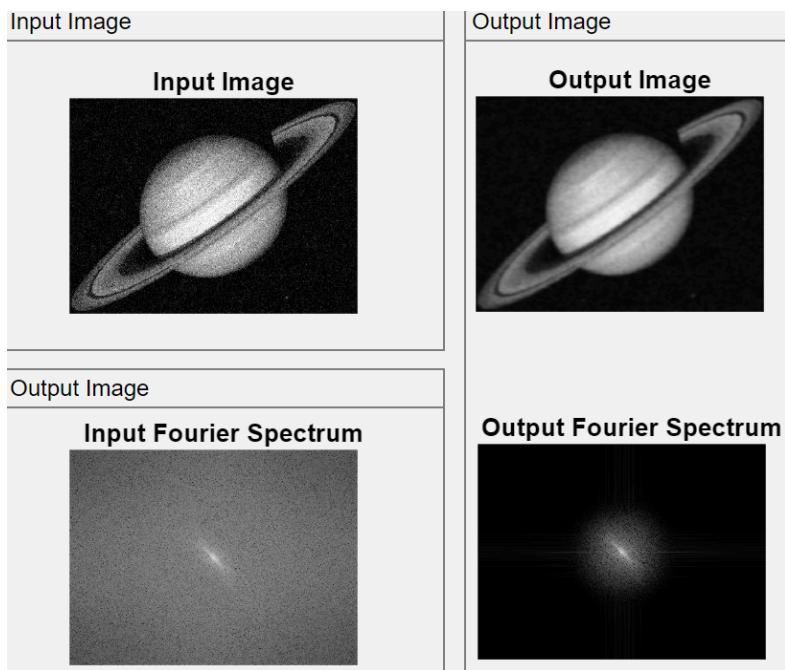
Gambar 2.46 BLPF citra bangunan dengan $d_0 = 60$ dan $n = 3$

ILPF citra *grayscale* Planet Saturnus (2-3.jpg) dengan $d_0 = 65$:



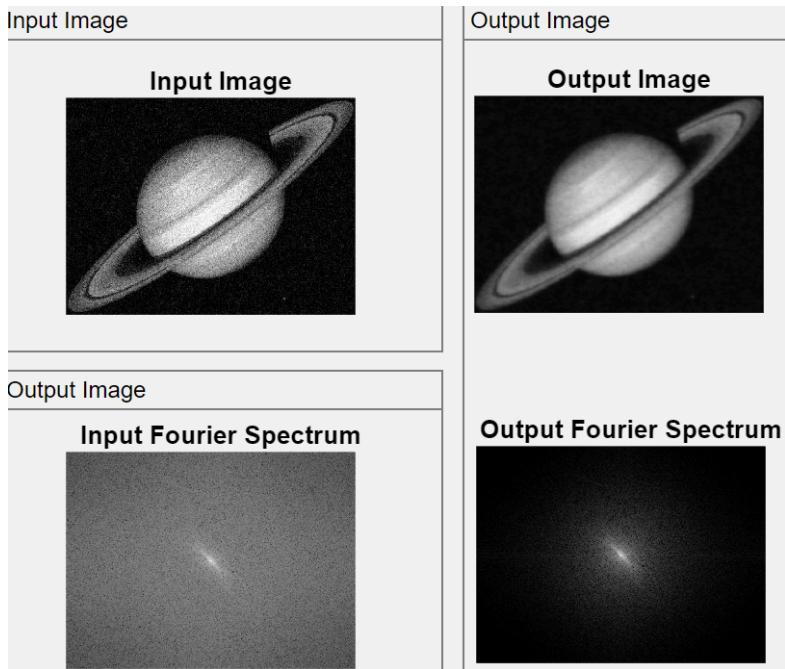
Gambar 2.47 Mean filter citra Planet Saturnus dengan $d_0 = 65$

GLPF citra *grayscale* Planet Saturnus (2-3.jpg) dengan $d0 = 65$:



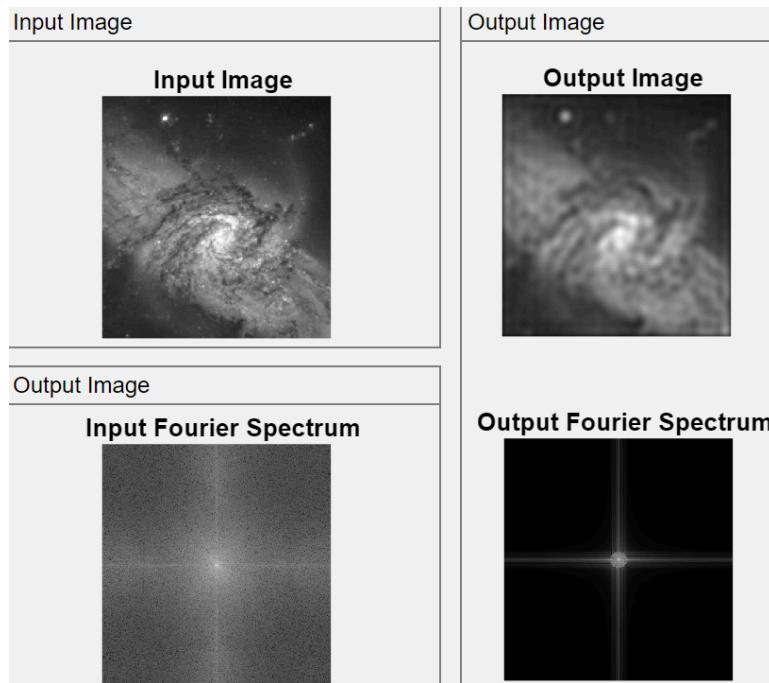
Gambar 2.48 *Gaussian filter* citra Planet Saturnus dengan $d0 = 65$

BLPF citra *grayscale* Planet Saturnus (2-3.jpg) dengan $d0 = 65$ dan $n = 1$:



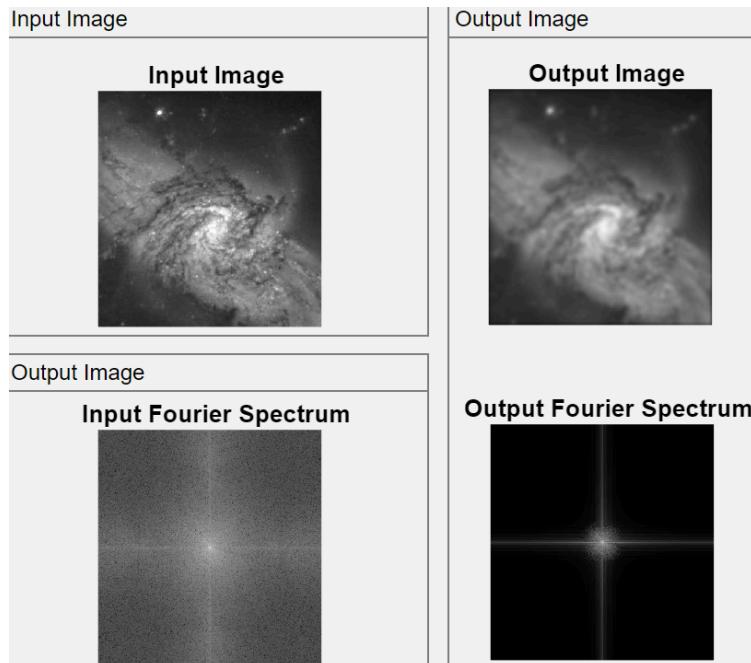
Gambar 2.49 BLPF citra Planet Saturnus dengan $d0 = 65$ dan $n = 1$

ILPF citra *grayscale* galaksi (galaxy_pair_original.jpg) dengan $d0 = 40$:



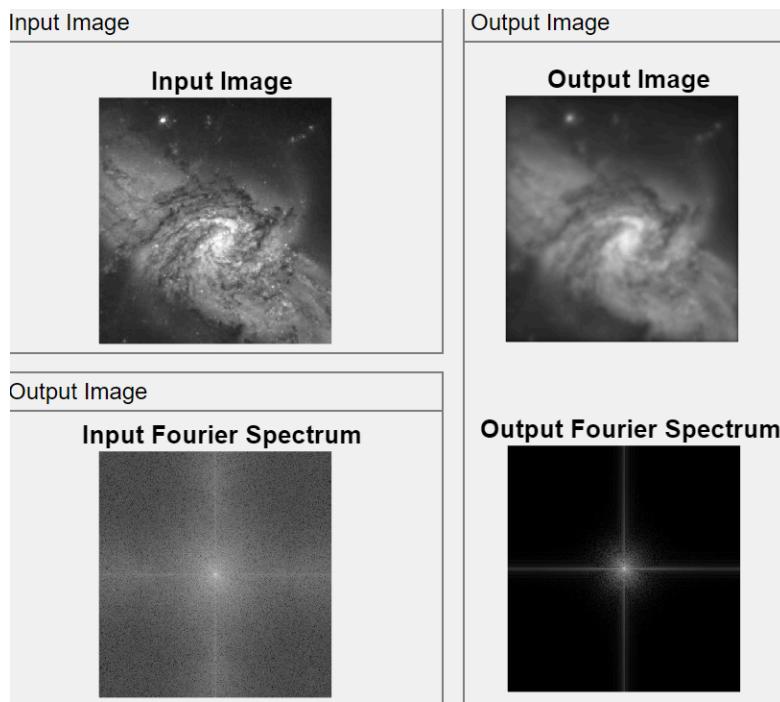
Gambar 2.50 ILPF citra galaksi dengan $d0 = 40$

GLPF citra *grayscale* galaksi (galaxy_pair_original.jpg) dengan $d0 = 40$:



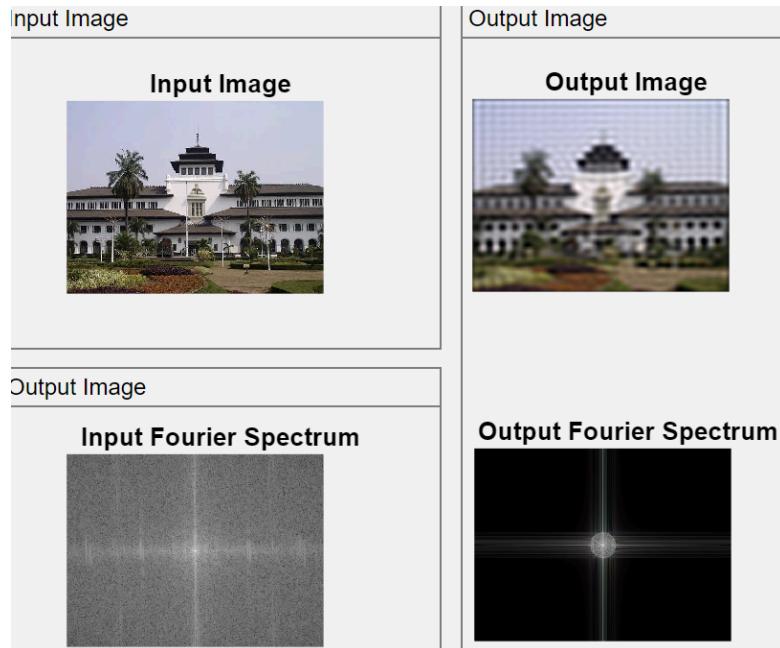
Gambar 2.51 GLPF citra galaksi dengan $d0 = 40$

BLPF citra *grayscale* galaksi (galaxy_pair_original.jpg) dengan $d0 = 40$ dan $n = 1$:



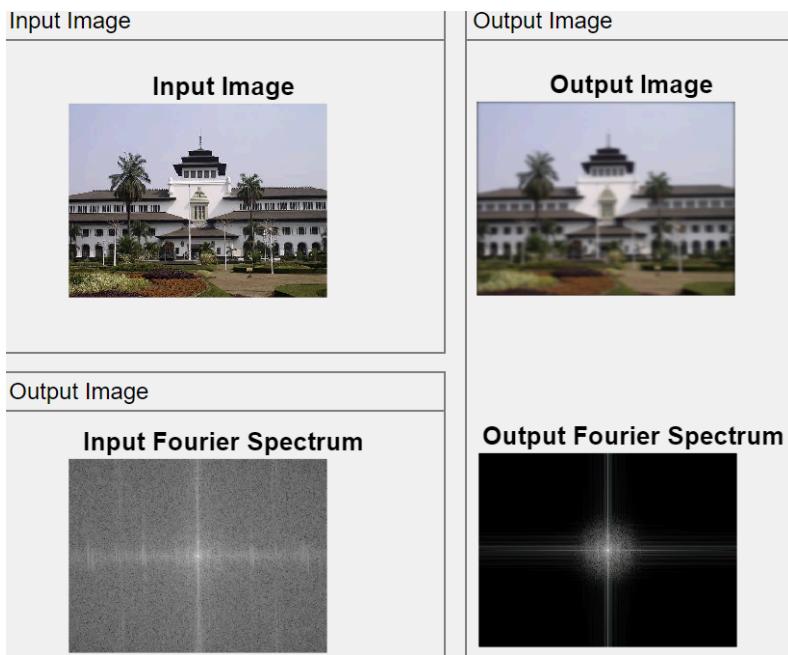
Gambar 2.52 BLPF citra galaksi dengan $d0 = 40$ dan $n = 1$

ILPF citra berwarna Gedung Sate (2-4.jpg) dengan $d0 = 50$:



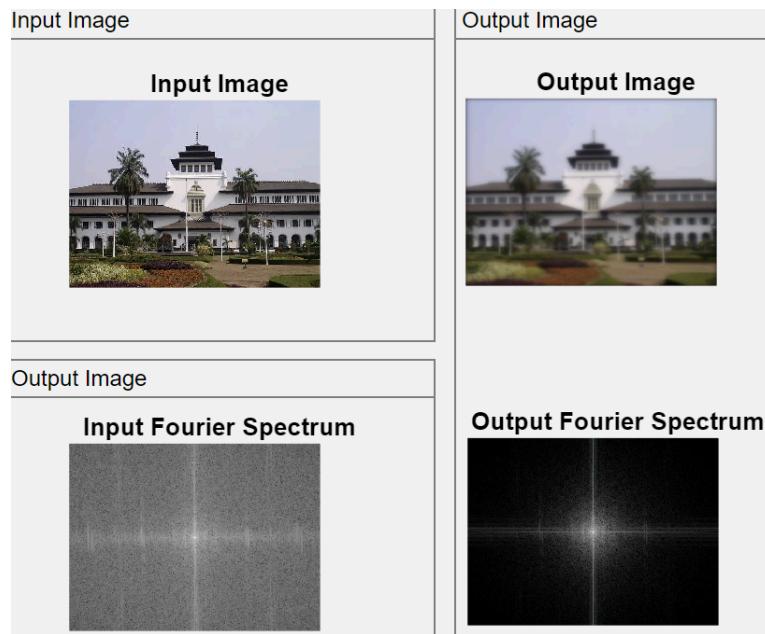
Gambar 2.53 ILPF citra Gedung Sate dengan $d0 = 50$

GLPF citra berwarna Gedung Sate (2-4.jpg) dengan $d0 = 50$:



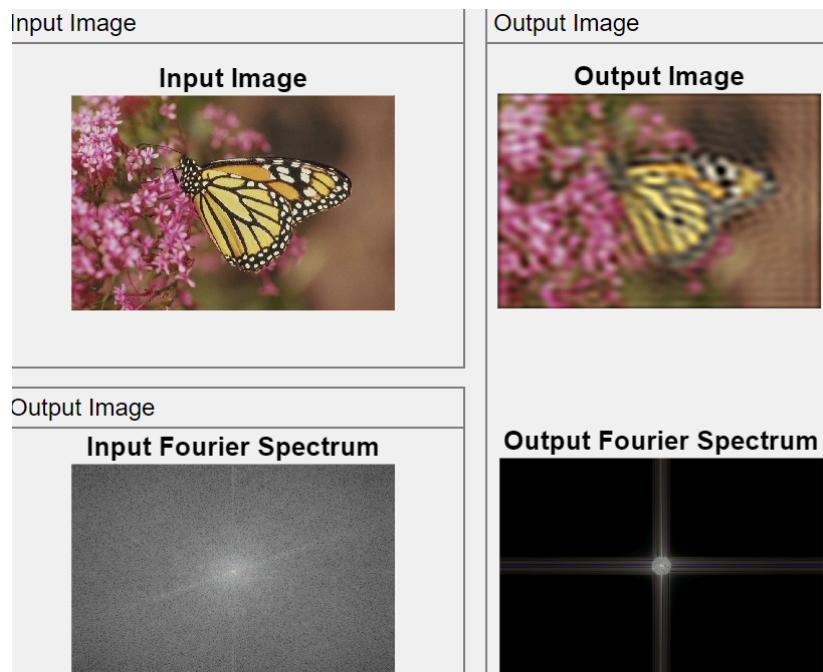
Gambar 2.54 GLPF citra Gedung Sate dengan $d0 = 50$

BLPF citra berwarna Gedung Sate (2-4.jpg) dengan $d0 = 50$ dan $n = 1$:



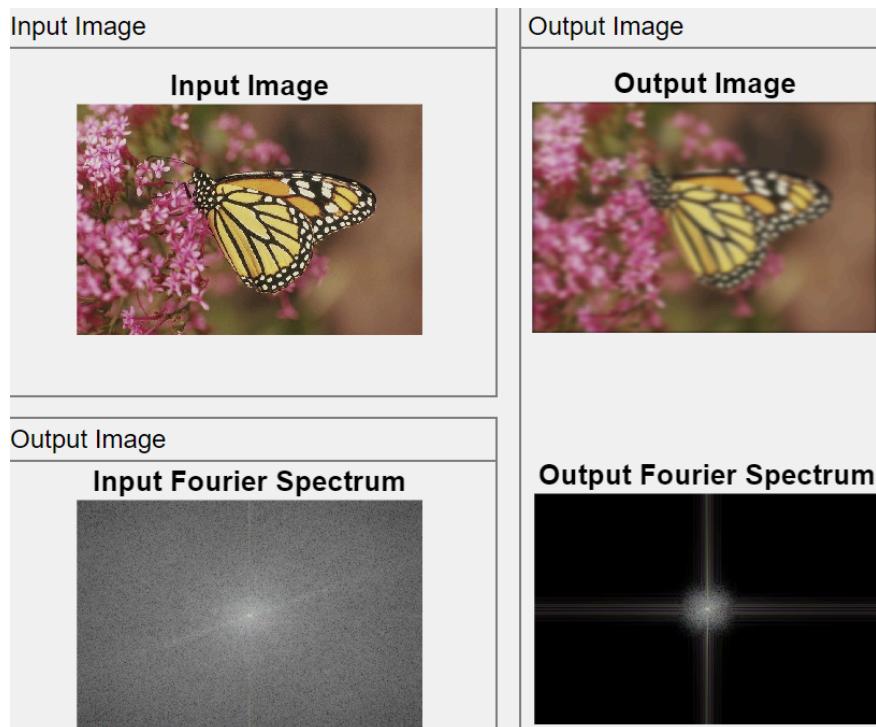
Gambar 2.55 BLPF citra Gedung Sate dengan $d0 = 50$ dan $n = 1$

ILPF citra berwarna kupu-kupu (2-5.jpg) dengan $d0 = 45$:



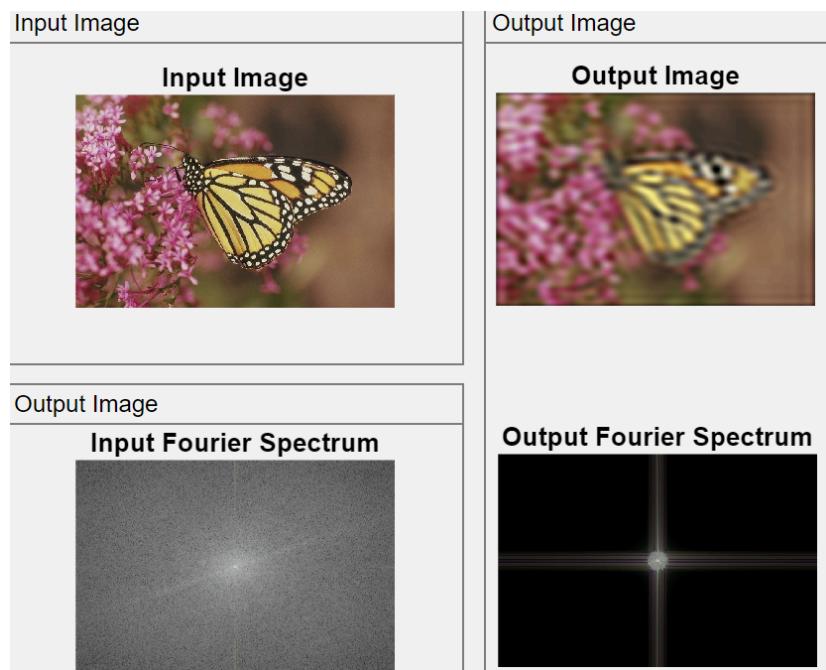
Gambar 2.56 ILPF citra kupu-kupu dengan $d0 = 45$

GLPF citra berwarna kupu-kupu (2-5.jpg) dengan $d0 = 45$:



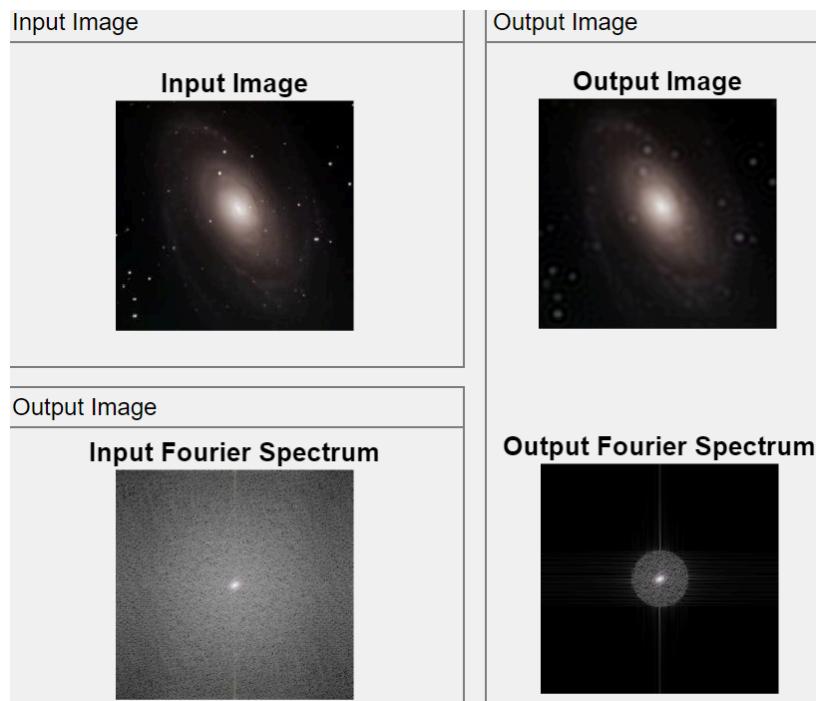
Gambar 2.57 GLPF citra kupu-kupu dengan $d0 = 45$

BLPF citra berwarna kupu-kupu (2-5.jpg) dengan $d0 = 45$ dan $n = 10$:



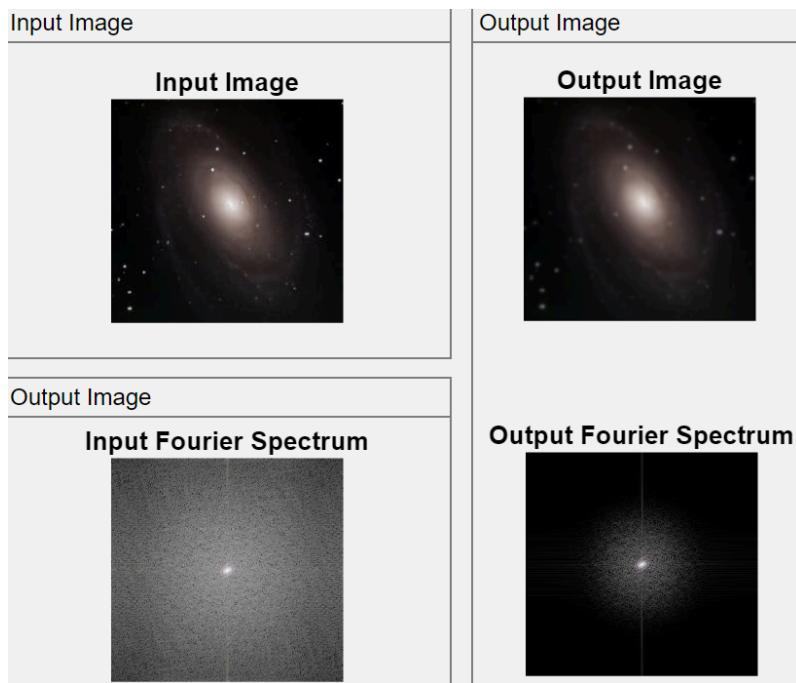
Gambar 2.58 BLPF citra kupu-kupu dengan $d0 = 45$ dan $n = 10$

ILPF citra berwarna galaksi (2-6.jpg) dengan $d0 = 55$:



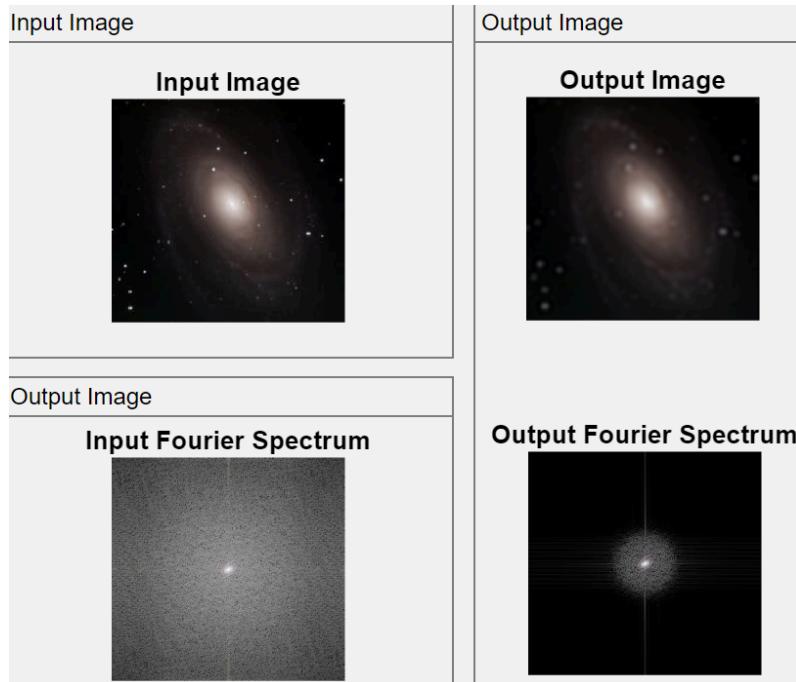
Gambar 2.59 ILPF citra galaksi dengan $d0 = 55$

GLPF citra berwarna galaksi (2-6.jpg) dengan $d0 = 55$:



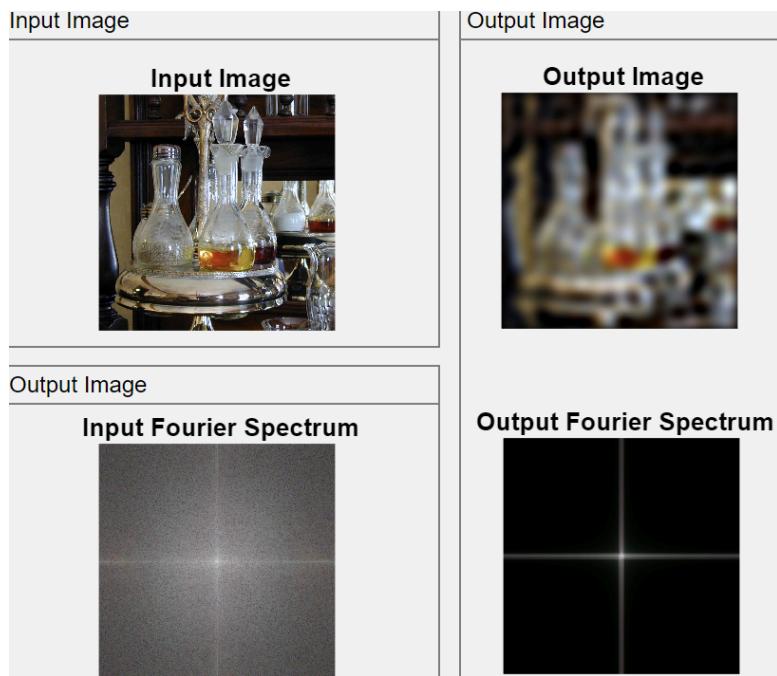
Gambar 2.60 GLPF citra galaksi dengan $d0 = 55$

BLPF citra berwarna galaksi (2-6.jpg) dengan $d0 = 55$ dan $n = 5$:



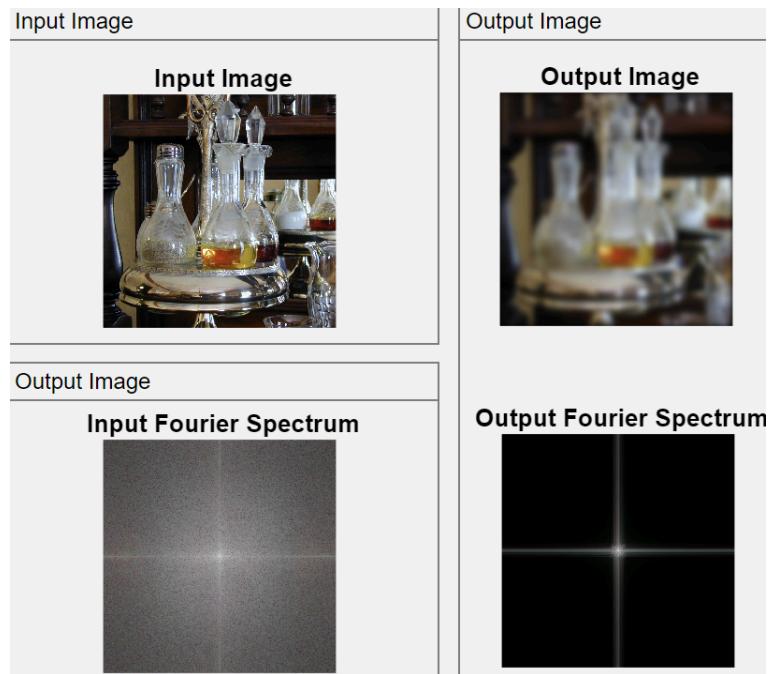
Gambar 2.61 BLPF citra galaksi dengan $d0 = 55$ dan $n = 5$

ILPF citra berwarna *caster* (caster_stand_origin.bmp) dengan $d0 = 25$:



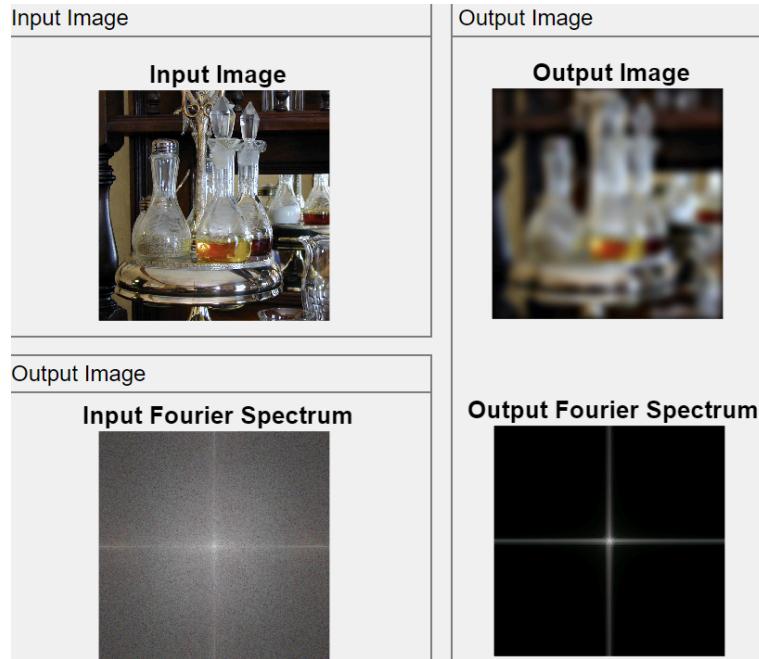
Gambar 2.62 ILPF citra *caster* dengan $d0 = 25$

GLPF citra berwarna *caster* (caster_stand_origin.bmp) dengan $d0 = 25$:



Gambar 2.63 GLPF citra *caster* dengan $d0 = 25$

BLPF citra berwarna *caster* (*caster_stand_origin.bmp*) dengan $d0 = 25$ dan $n = 3$:



Gambar 2.64 BLPF citra *caster* dengan $d0 = 25$ dan $n = 3$

Analisis:

Penapisan lolos rendah dapat secara efektif mengurangi derau dan detail halus pada citra, sehingga menghasilkan citra yang lebih halus dan lebih bersih. Karena penapis ideal (ILPF) memblokir seluruh frekuensi di atas $d0$, maka penapis ideal memiliki transisi yang kasar dan mendadak sehingga menimbulkan *ringing* pada citra. Sementara itu, penapis *Gaussian* (GLPF) dengan nilai frekuensi *cut-off* yang sesuai menghasilkan citra dengan transisi yang lebih lembut antarwarna dan menghilangkan artefak yang tidak diinginkan. Penapis *Butterworth* (BLPF) menghasilkan citra dengan transisi yang lebih halus daripada ILPF tetapi lebih tajam daripada GLPF. Frekuensi rendah pada BLPF dipertahankan dengan cara yang lebih selektif, sehingga memberikan keseimbangan antara ketajaman dan pengurangan noise.

Semakin tinggi nilai frekuensi *cut-off* ($d0$), maka lebih banyak detail yang dipertahankan, namun *noise* semakin sedikit yang dikurangi. Pada penapis *Butterworth*, semakin tinggi nilai orde (n), maka transisi antara frekuensi yang dipertahankan dan diblokir menjadi lebih tinggi (tajam). Oleh karena itu, pemilihan

jenis filter dan parameter yang tepat sangat penting untuk menyeimbangkan antara mengurangi derau dan mempertahankan detail.

2.4. High-pass Filter

Berikut merupakan kode program fungsi untuk melakukan penapisan pada citra dalam ranah frekuensi menggunakan *high-pass filter* IHPF, GHPF, dan BHPF.

```
function outputImage = highPassFilter(image,filter, d0, nInput)
[rows, cols, colorChannels] = size(image);

disp(filter);
% Menentukan parameter padding, biasanya P = 2*rows dan Q = 2*cols
P = 2*rows;
Q = 2*cols;
tempImage = zeros([P Q colorChannels]);
image = im2double(image);
fp = zeros([P Q colorChannels]);
% Melakukan padding dengan nilai 0 untuk pixel di luar image awal
for c = 1:colorChannels
    for i = 1 : P
        for j = 1 : Q
            if i <= rows && j <= cols
                fp(i,j,c) = image(i,j,c);
            else
                fp(i,j,c) = 0;
            end
        end
    end
end
% Pemrosesan untuk tiap kanal warna
for c = 1:colorChannels
    currImg = fp(:,:,c);
    % transformasi fourier
    F = fftshift(fft2(currImg));
    %D0 = 0.05*P; % cut-off frequency
    D0 = d0;
    % range dari variabel
    u = 0:(P-1);
    v = 0:(Q-1);
    % index untuk digunakan pada meshgrid
    idx = find(u > P/2);
    u(idx) = u(idx) - P;
    idy = find(v > Q/2);
    v(idy) = v(idy) - Q;
    % menghitung meshgrid array
    [V, U] = meshgrid(v, u);
    D = sqrt(U.^2 + V.^2);

    H = 0;
    %wiene melakukan filtering berdasarkan jenis filter
    if filter == "Gaussian"
```

```

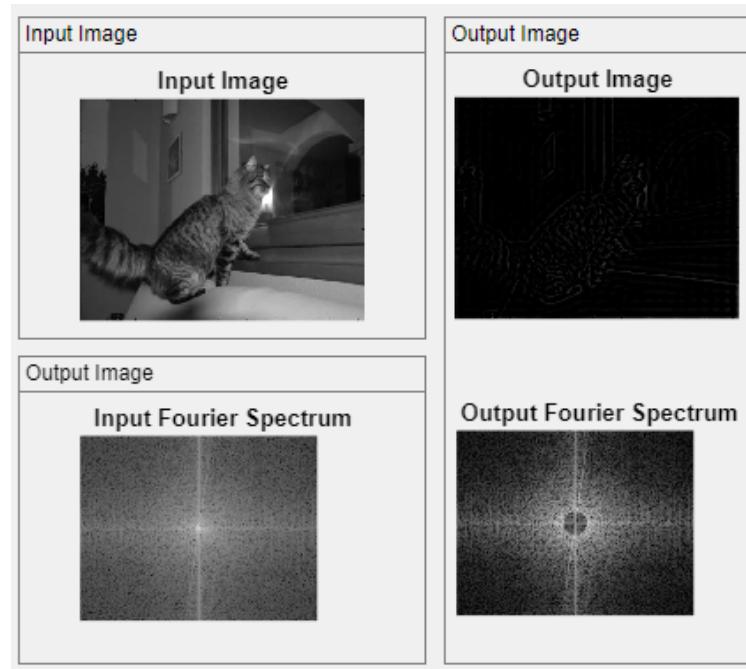
H = 1 - exp(-(D.^2)./(2*(D0^2)));
elseif filter == "Ideal"
    H = 1 - double(D<=50);
elseif filter == "Butterworth"
    % n = 1
    n = nInput;
    H = 1 - 1./(1 + (D./D0).^(2*n));
end
H = fftshift(H);
G = H.*F;
G1 = ifftshift(G);
G2 = real(ifft2(G1));
tempImage(:,:,c) = G2(:,:,1);
end
outputImage = tempImage(1:rows, 1:cols, :);
end

```

Operasi penapisan lolos tinggi dilakukan dengan mengubah citra ke bentuk frekuensi terlebih dahulu dengan *Fast Fourier Transform*. Ukuran citra akan diubah menjadi 4 kali lipatnya terlebih dahulu untuk menyesuaikannya dengan *Fourier Spectrum*. Operasi ini memerlukan penapis, yang dapat dibagi menjadi tiga jenis: ideal, *gaussian*, dan *butterworth*. Filter ini lalu dikalikan dengan spektrum fourier dari citra yang sudah diubah ukurannya. Hasil dari Penapisan lolos tinggi diperoleh dengan memotong kembali citra ke ukuran asalnya.

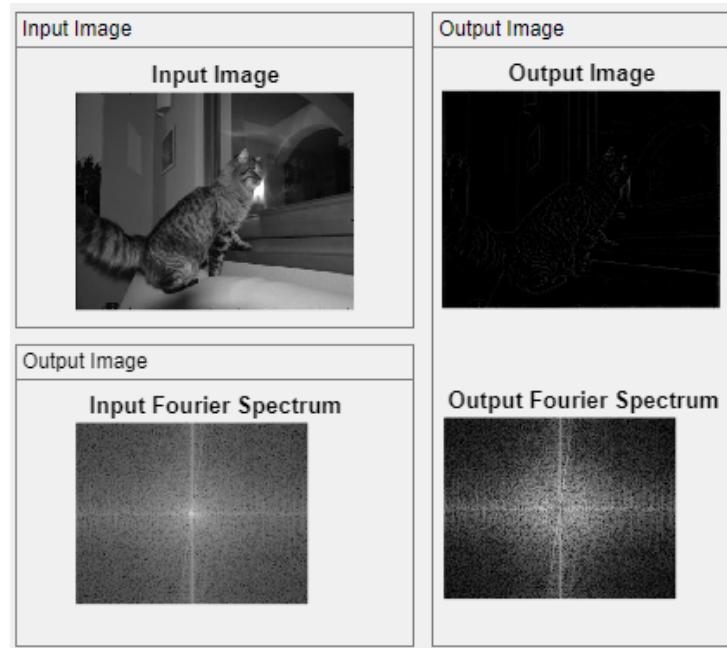
Berikut merupakan contoh hasil eksekusi program.

IHPF citra *grayscale* kucing (3-1.jpg) dengan $D0 = 50$:



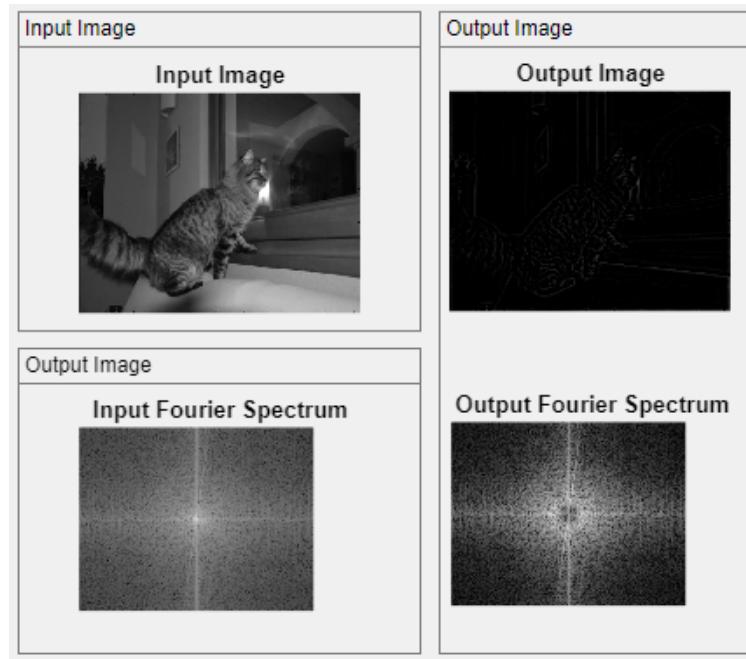
Gambar 2.65 IHPF citra kucing dengan $D0 = 50$

GHPF citra *grayscale* kucing (3-1.jpg) dengan $D0 = 50$:



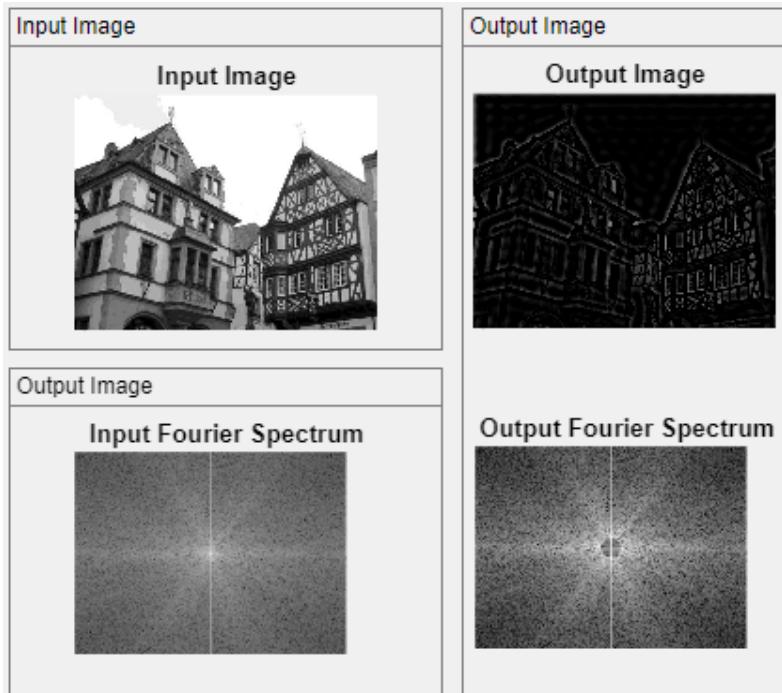
Gambar 2.66 GHPF citra kucing dengan $D0 = 50$

BHPF citra *grayscale* kucing (3-1.jpg) dengan $D0 = 50$ dan $n = 3$:



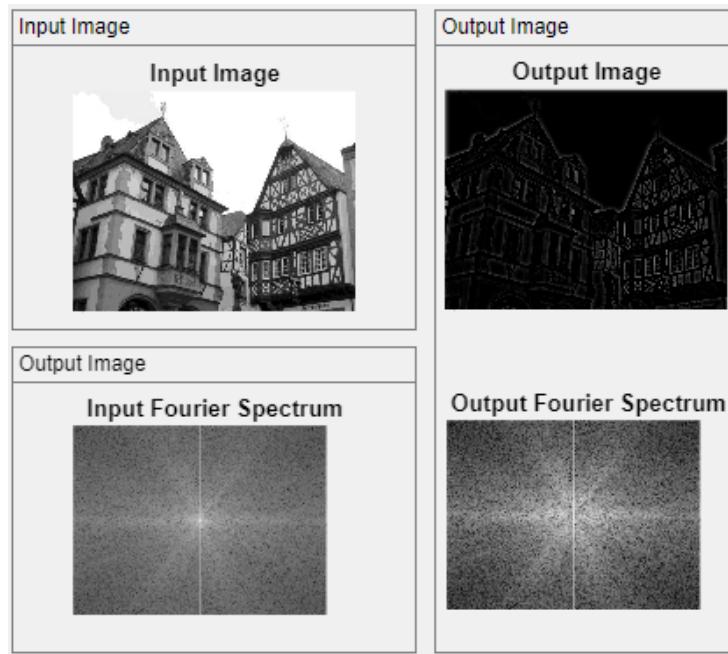
Gambar 2.67 BHPF citra kucing dengan $D0 = 50$ dan $n = 3$

IHPF citra *grayscale* bangunan (3-2.jpg) dengan $D0 = 40$:



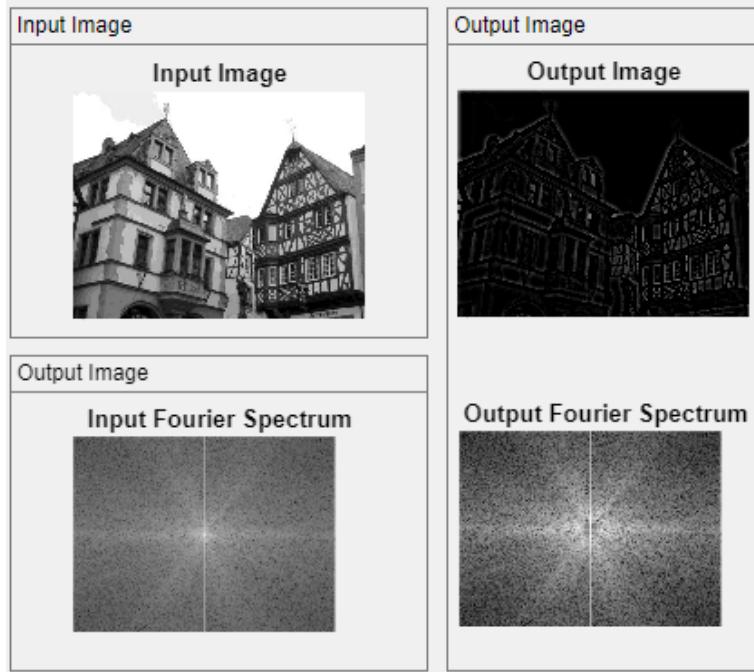
Gambar 2.68 IHPF citra bangunan dengan $D0 = 40$

GHPF citra *grayscale* bangunan (3-2.jpg) dengan $D0 = 40$:



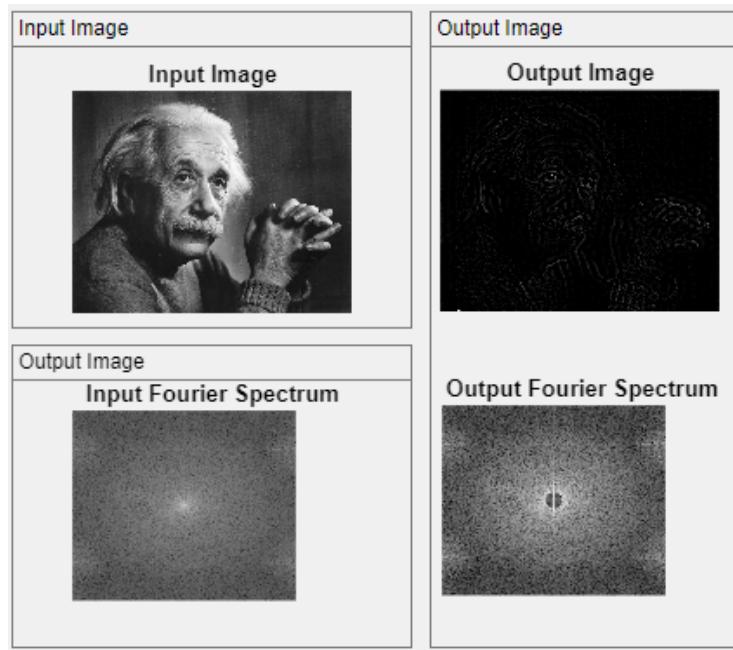
Gambar 2.69 GHPF citra bangunan dengan $D0 = 40$

BHPF citra *grayscale* bangunan (3-2.jpg) dengan $D0 = 40$ dan $n = 3$:



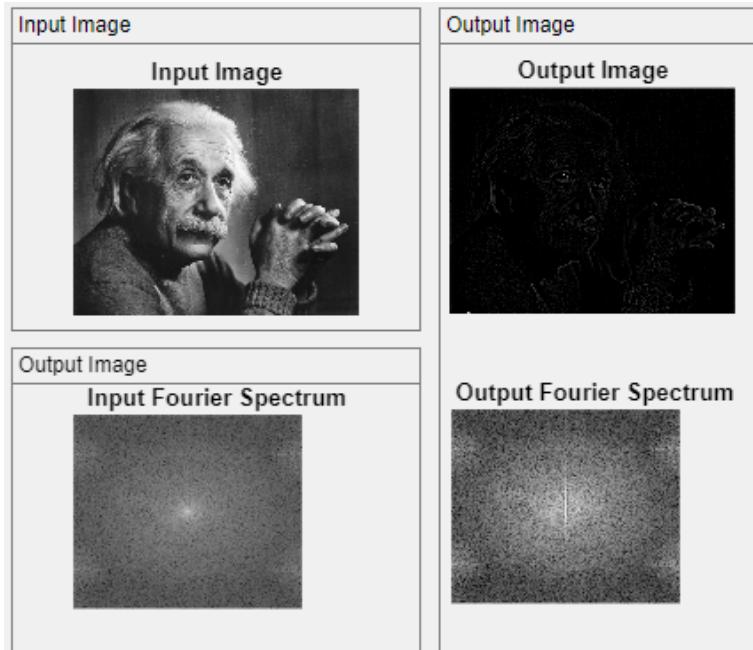
Gambar 2.70 BHPF citra bangunan dengan $D0 = 40$ dan $n = 3$

IHPF citra *grayscale* Einstein (3-4.tif) dengan $D0 = 60$:



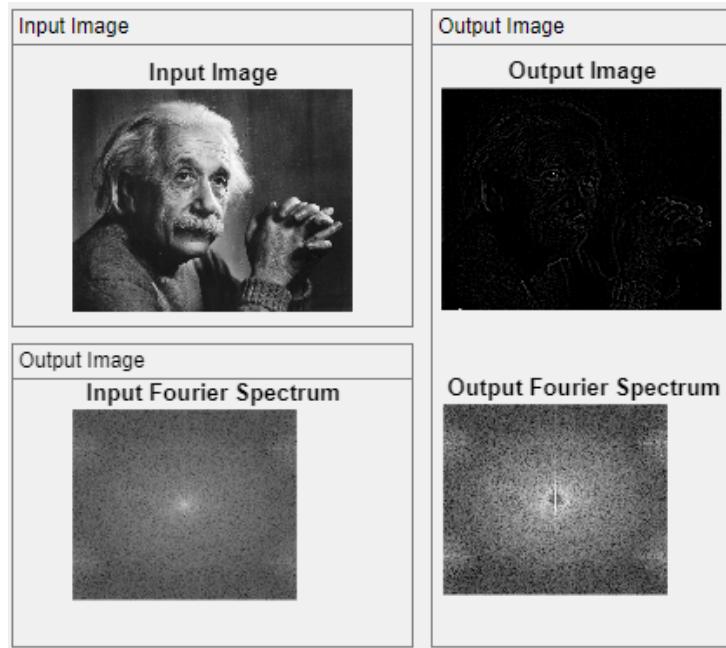
Gambar 2.71 IHPF citra Einstein dengan $D0 = 60$

GHPF citra *grayscale* Einstein (3-4.tif) dengan $D0 = 60$:



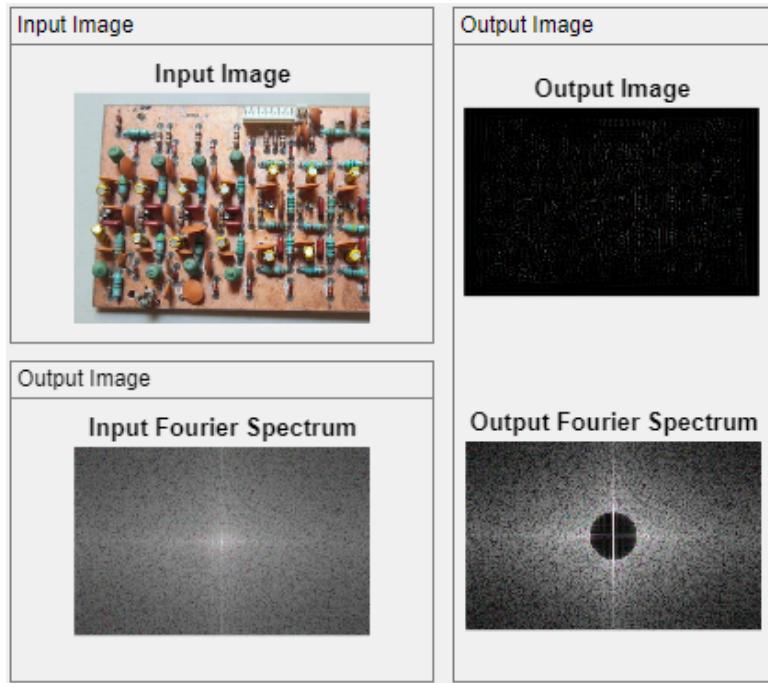
Gambar 2.72 GHPF citra Einstein dengan $D0 = 60$

BHPF citra *grayscale* Einstein (3-4.tif) dengan $D0 = 60$ dan $n = 3$:



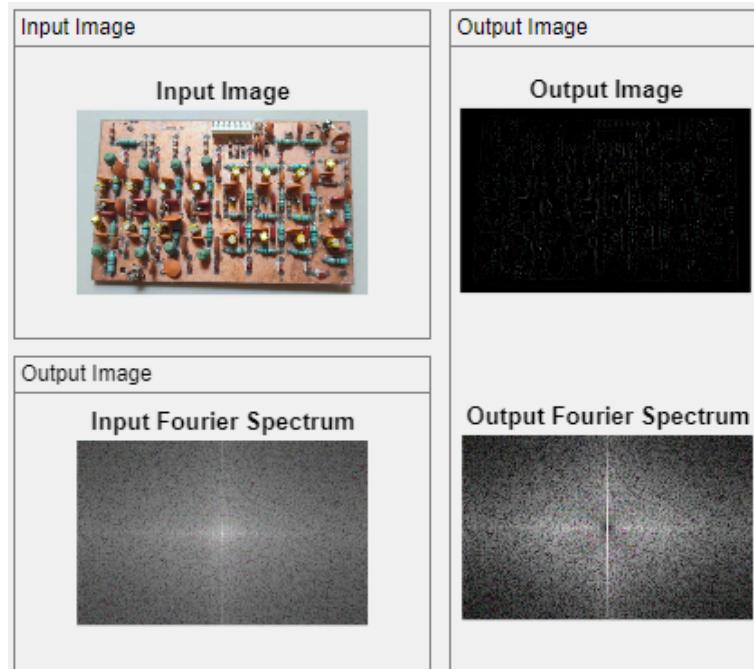
Gambar 2.73 BHPF citra Einstein dengan $D0 = 60$ dan $n = 3$

IHPF citra berwarna papan elektronik (3-3.jpg) dengan $D0 = 100$:



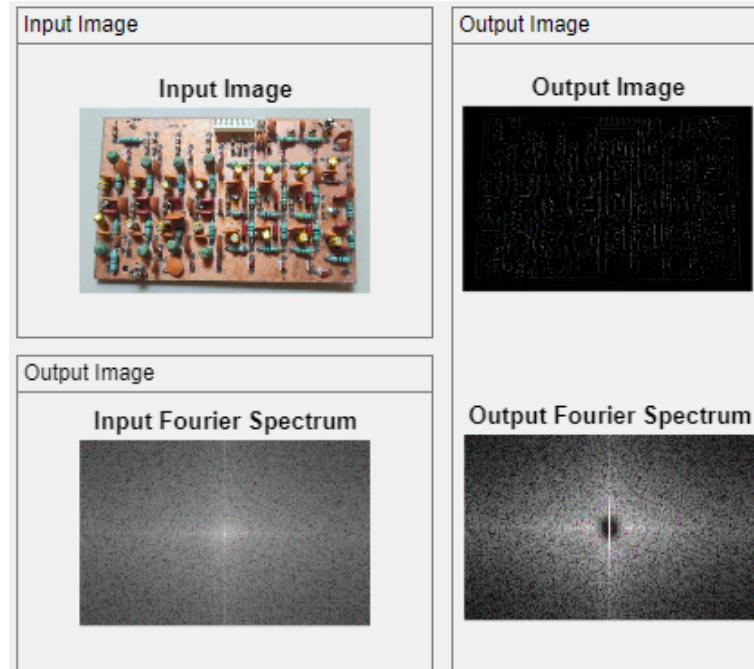
Gambar 2.74 IHPF citra papan elektronik dengan $D0 = 100$

GHPF citra berwarna papan elektronik (3-3.jpg) dengan $D0 = 100$:



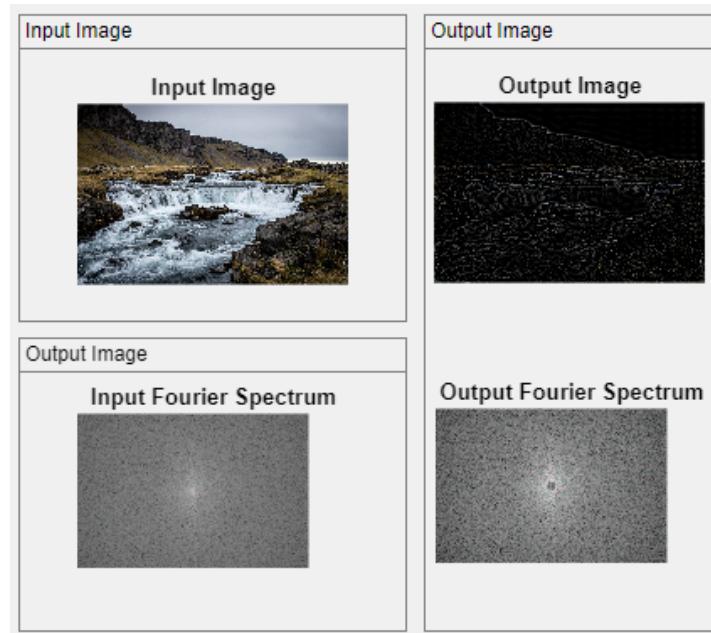
Gambar 2.75 GHPF citra papan elektronik dengan $D0 = 100$

BHPF citra berwarna papan elektronik (3-3.jpg) dengan $D0 = 70$ dan $n = 3$:



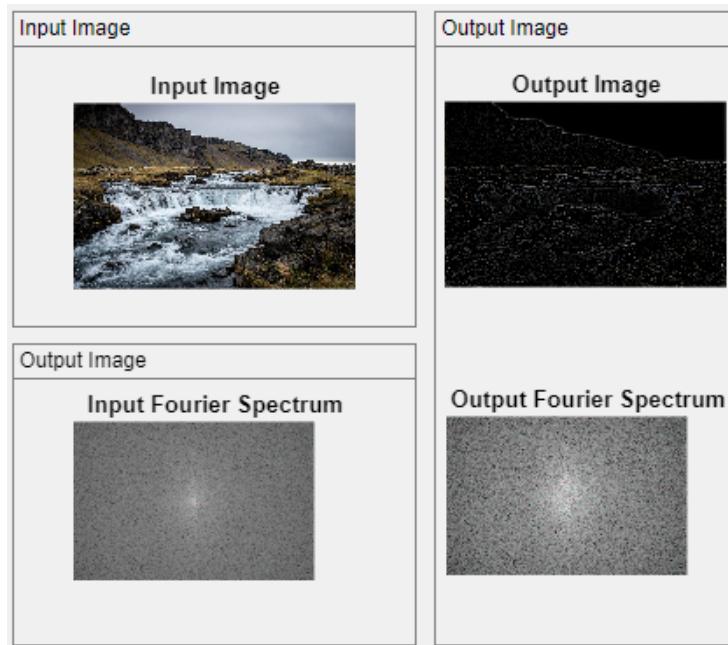
Gambar 2.76 BHPF citra papan elektronik dengan $D0 = 70$ dan $n = 3$

IHPF citra berwarna waterfall (3-5.jpg) dengan $D0 = 50$



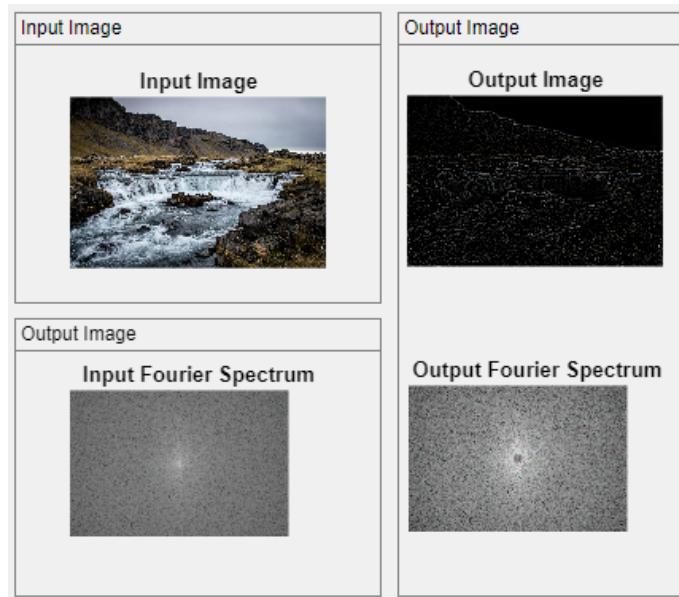
Gambar 2.77 IHPF citra waterfall dengan $d0 = 50$

GHPF citra berwarna waterfall (3-5.jpg) dengan $D0 = 50$



Gambar 2.78 GHPF citra waterfall dengan $D0 = 50$

BHPF citra berwarna waterfall (3-5.jpg) dengan $D0 = 60$ dan $n = 4$



Gambar 2.79 BHPF citra waterfall dengan $D0 = 60$ dan $n = 4$

Analisis:

High-pass filtering adalah mekanisme untuk meloloskan piksel dengan frekuensi tinggi. Frekuensi yang tinggi dapat ditemukan di tepi-tepi objek, di mana terdapat perbedaan yang cukup tinggi antar dua piksel. Secara umum, penapis ini akan menghasilkan citra yang mengandung tepi-tepi dari objek. Saat melakukan penapisan dengan IHPF, penapisan dilakukan dengan mengalikan frekuensi dengan 0 atau 1, tergantung ambang batas yang ditetapkan ($D0$). Alhasil, penapisan dengan filter ini menghasilkan citra yang tidak begitu mulus. Gaussian filter memberikan hasil yang lebih mulus dan garis lebih tegas daripada ideal filter, karena menggunakan fungsi eksponen, sehingga perubahan terjadi secara perlahan. Penapis *butterworth* menghasilkan hasil yang lebih baik daripada penapis ideal ataupun *gaussian* karena fungsinya berbentuk pecahan dan mengandung eksponen.

2.5. *Image Brightening*

Berikut merupakan kode program fungsi untuk melakukan penapisan citra dalam ranah frekuensi untuk menghasilkan citra yang lebih terang.

```
function outputImage = brightening(image, brightnessFactor)
[rows, cols, colorChannels] = size(image);
% Menentukan parameter padding, biasanya P = 2*rows dan Q = 2*cols
```

```

P = 2*rows;
Q = 2*cols;
tempImage = zeros([P Q colorChannels]);
image = im2double(image);
fp = zeros([P Q colorChannels]);
% proses
for c = 1:colorChannels
    for i = 1 : P
        for j = 1 : Q
            if i <= rows && j <= cols
                fp(i,j,c) = image(i,j,c);
            else
                fp(i,j,c) = 0;
            end
        end
    end
end
for c = 1:colorChannels
    currImg = fp(:,:,c);
    % transformasi fourier
    F = fftshift(fft2(currImg));
    radius = 100;                                % Tentukan radius untuk
memodifikasi komponen frekuensi rendah
    centerX = P / 2;
    centerY = Q / 2;
    brightness_factor = brightnessFactor;          % Faktor untuk
meningkatkan kecerahan (sesuaikan nilainya)

    for i = 1:P
        for j = 1:Q
            distance = sqrt((i - centerX)^2 + (j - centerY)^2);
            if distance < radius
                F(i, j) = F(i, j) * brightness_factor;
            end
        end
    end

    F2 = ifftshift(F);
    Freal = real(ifft2(F2));
    tempImage(:,:,c) = Freal(:,:,1);
end
outputImage = tempImage(1:rows, 1:cols, :);
end

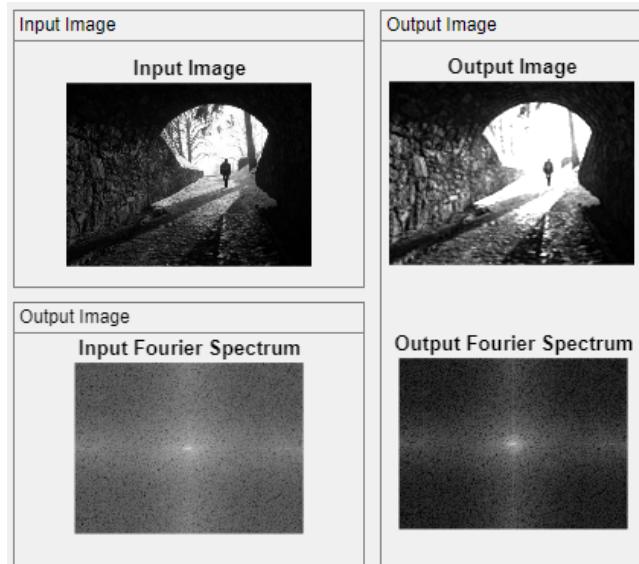
```

Pencerahan citra ini dilakukan dalam ranah frekuensi. Oleh karena itu, citra masukan pengguna akan diubah ke bentuk frekuensi terlebih dahulu dengan *Fast Fourier Transform*. Hasil dari spektrum fourier tersebut kemudian akan dikalikan dengan suatu faktor kecerahan yang dimasukkan oleh pengguna. Hasil dari

perkalian tersebut kemudian dikembalikan ke ranah spasial dengan fungsi ifft2. Nilai yang diambil dari spektrum adalah nilai riil nya saja.

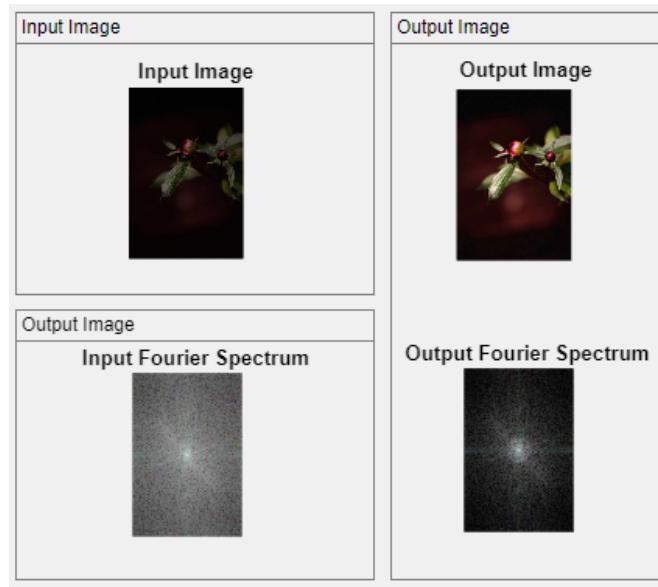
Berikut merupakan contoh hasil eksekusi program.

Image brightening citra grayscale terowongan (4-1.jpg) dengan faktor kecerahan 2.4:



Gambar 2.80 *Image brightening* citra terowongan

Image brightening citra berwarna bunga (4-2.jpg):



Gambar 2.81 *Image brightening* citra bunga

Analisis:

Dari hasil yang ditampilkan, terlihat bahwa gambar keluaran lebih cerah daripada gambar masukan. Dari spektrum *fourier* hasilnya juga dapat dilihat bahwa persebaran frekuensi dari citra sudah berkurang daripada gambar masukan dan menjadi lebih padat. Hal ini juga menunjukkan perbedaan frekuensi yang lebih tajam pada gambar, sehingga kontrasnya lebih tinggi.

2.6. Penghilangan Derau *Salt and Pepper*

Berikut merupakan kode program fungsi untuk melakukan penapisan citra dalam ranah frekuensi untuk menghasilkan citra yang lebih terang.

```
function outputImage = saltPepperNoise(image, density)
    outputImage = imnoise(image, 'salt & pepper', density);
end

function outputImage = noiseFiltering(image, filter, varargin)
    image = double(image);
    [rows, cols, channels] = size(image);
    outputImage = zeros(rows, cols, channels);
    % Ukuran window 3x3
    windowHeight = 3;
    padSize = floor(windowHeight / 2);
    paddedImage = padarray(image, [padSize padSize], 'symmetric');
    % Penapisan untuk tiap kanal warna
    for c = 1:channels
        for i = 1:rows
            for j = 1:cols
                % Ekstrak window untuk channel c
                window = paddedImage(i:i+windowHeight-1, j:j+windowHeight-1,
c);
                window = window(:); % Ubah window jadi vektor
                switch lower(filter)
                    case 'min'
                        outputImage(i, j, c) = min(window);
                    case 'max'
                        outputImage(i, j, c) = max(window);
                    case 'median'
                        outputImage(i, j, c) = median(window);
                    case 'arithmetic mean'
                        outputImage(i, j, c) = mean(window);
                    case 'geometric mean'
                        outputImage(i, j, c) =
prod(window)^(1/numel(window));
                    case 'harmonic mean'
                        outputImage(i, j, c) = numel(window) / sum(1 ./
window);
                    case 'contraharmonic mean'
                        Q = varargin{1}; % Contraharmonic order Q
                end
            end
        end
    end
end
```

```

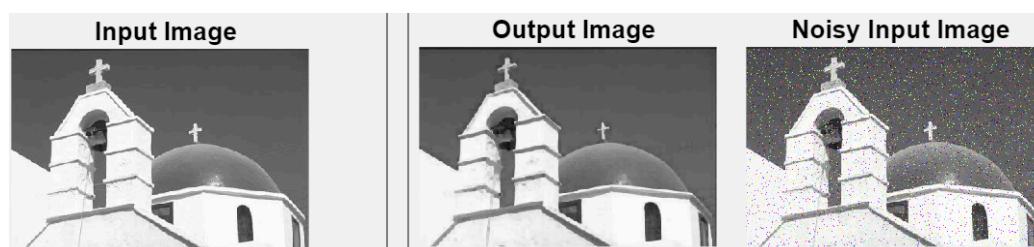
        outputImage(i, j, c) = sum(window.^Q + 1)) /
sum(window.^Q);
    case 'midpoint'
        outputImage(i, j, c) = (min(window) +
max(window)) / 2;
    case 'alpha-trimmed mean'
        d = varargin{1}; % Banyaknya elemen yang
dipangkas
        sortedWindow = sort(window);
        trimmedWindow = sortedWindow(d + 1:end - d);
        outputImage(i, j, c) = mean(trimmedWindow);
    otherwise
        error('Unknown filter type');
    end
end
end
outputImage = uint8(outputImage);
end

```

Fungsi `saltPepperNoise` menambahkan derau *salt and pepper* pada citra. Fungsi `noiseFiltering` menerapkan teknik penapisan pada citra untuk mengurangi derau dengan menerima parameter citra (`image`), jenis penapis (`filter`), dan parameter tambahan, misalnya ordo kontraharmonik Q atau banyaknya elemen yang dipangkas d (melalui `varargin`). Fungsi ini menambahkan *padding* ke citra untuk memudahkan proses penapisan. Kemudian, untuk setiap saluran warna, fungsi mengambil *window* berukuran 3x3 dan menerapkan penapis yang dipilih untuk menghitung nilai baru untuk setiap piksel.

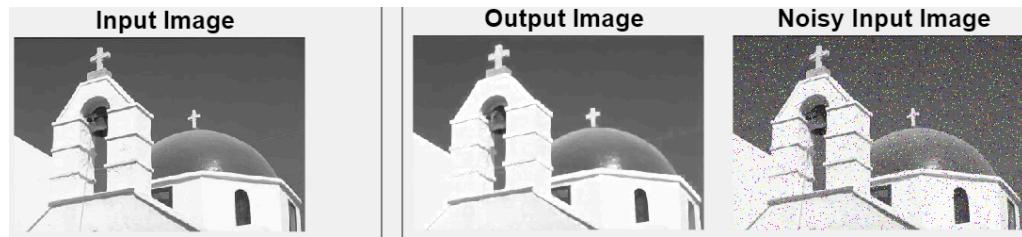
Berikut merupakan contoh hasil eksekusi program.

Penghilangan derau *salt and pepper* citra grayscale bangunan dengan *min filter* (5-1.jpg):



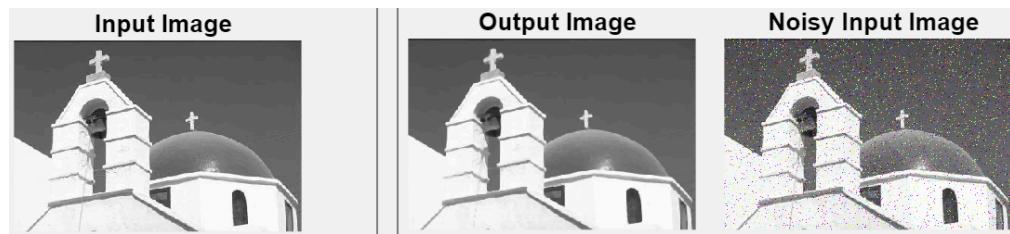
Gambar 2.82 Penghilangan derau *salt and pepper* citra bangunan dengan *min filter*

Penghilangan derau *salt and pepper* citra grayscale bangunan dengan *max filter* (5-1.jpg):



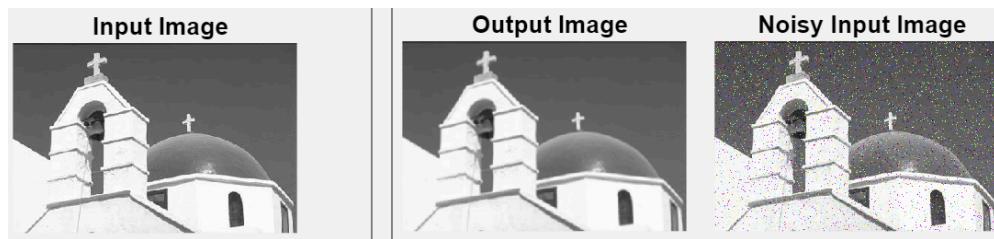
Gambar 2.83 Penghilangan derau *salt and pepper* citra bangunan dengan *max filter*

Penghilangan derau *salt and pepper* citra grayscale bangunan dengan *median filter* (5-1.jpg):



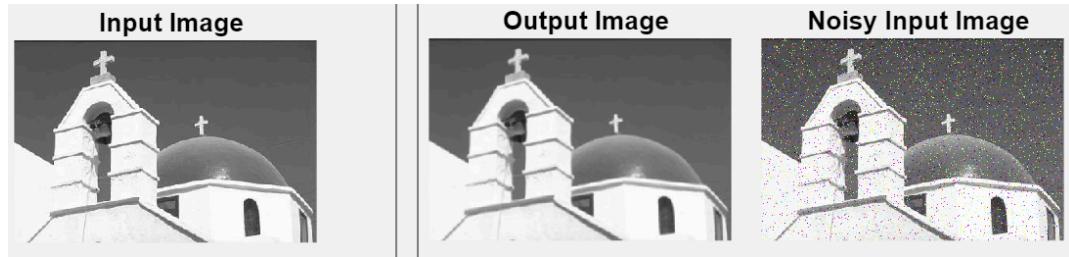
Gambar 2.84 Penghilangan derau *salt and pepper* citra bangunan dengan *median filter*

Penghilangan derau *salt and pepper* citra grayscale bangunan dengan *arithmetic mean filter* (5-1.jpg):



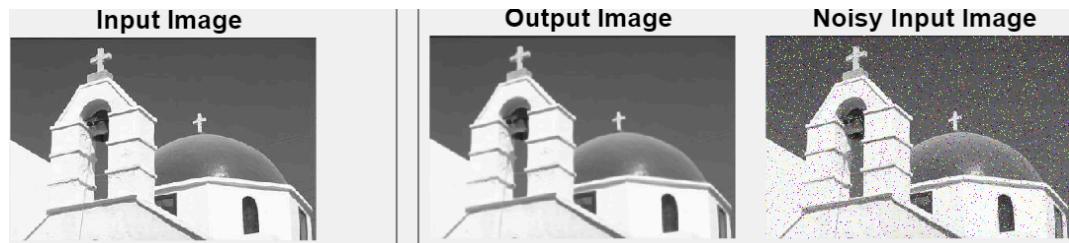
Gambar 2.85 Penghilangan derau *salt and pepper* citra bangunan dengan *arithmetic mean filter*

Penghilangan derau *salt and pepper* citra grayscale bangunan dengan *geometric mean filter* (5-1.jpg):



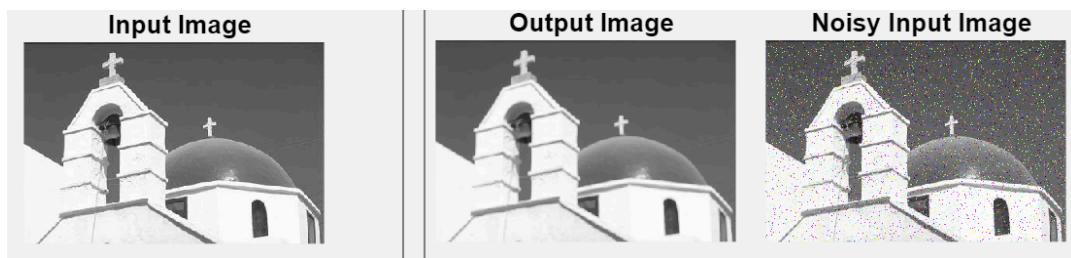
Gambar 2.86 Penghilangan derau *salt and pepper* citra bangunan dengan *geometric mean filter*

Penghilangan derau *salt and pepper* citra grayscale bangunan dengan *harmonic mean filter* (5-1.jpg):



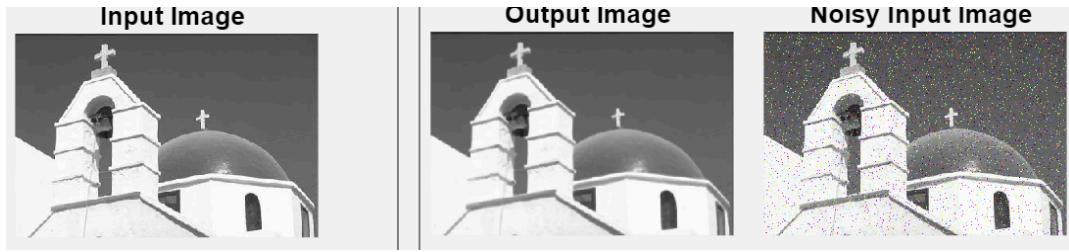
Gambar 2.87 Penghilangan derau *salt and pepper* citra bangunan dengan *harmonic mean filter*

Penghilangan derau *salt and pepper* citra grayscale bangunan dengan *contraharmonic mean filter* dengan $Q = 1.5$ (5-1.jpg):



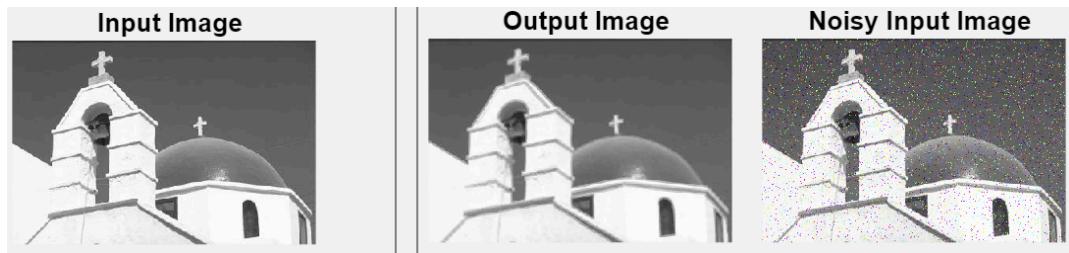
Gambar 2.88 Penghilangan derau *salt and pepper* citra bangunan dengan *contraharmonic mean filter* dengan $Q = 1.5$

Penghilangan derau *salt and pepper* citra grayscale bangunan dengan *contraharmonic mean filter* dengan $Q = -1.5$ (5-1.jpg):



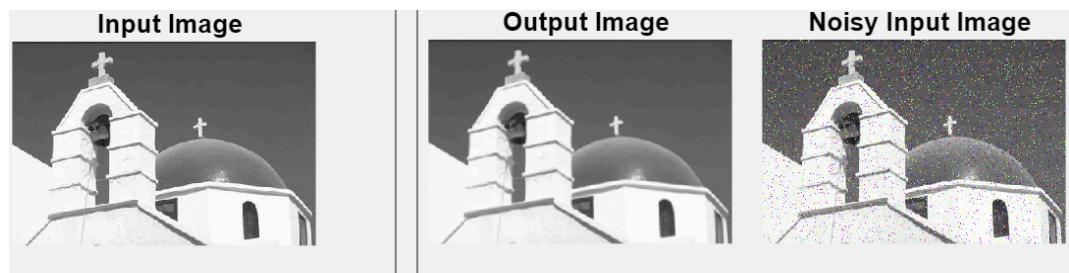
Gambar 2.89 Penghilangan derau *salt and pepper* citra bangunan dengan *contraharmonic mean filter* dengan $Q = -1.5$

Penghilangan derau *salt and pepper* citra grayscale bangunan dengan *midpoint filter* (5-1.jpg):



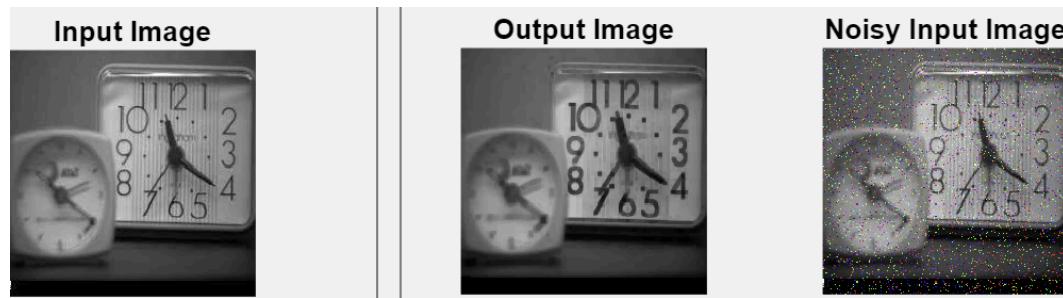
Gambar 2.90 Penghilangan derau *salt and pepper* citra bangunan dengan *midpoint filter*

Penghilangan derau *salt and pepper* citra grayscale bangunan dengan *alpha-trimmed mean filter* (5-1.jpg) dengan $d = 1$:



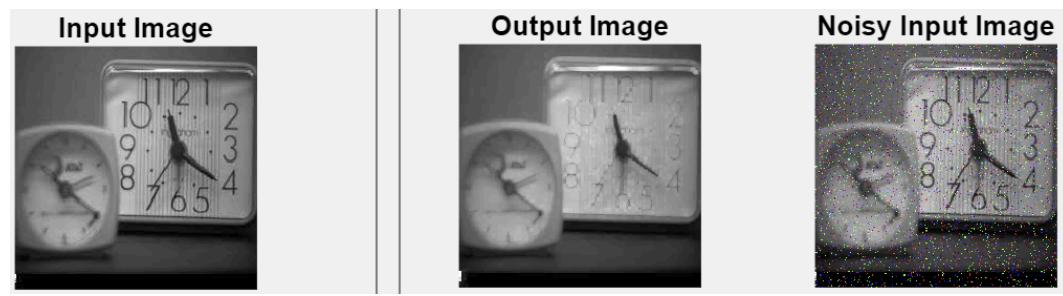
Gambar 2.91 Penghilangan derau *salt and pepper* citra bangunan dengan *alpha-trimmed mean filter* dengan $d = 1$

Penghilangan derau *salt and pepper* citra grayscale jam dengan *min filter* (1-3.jpg):



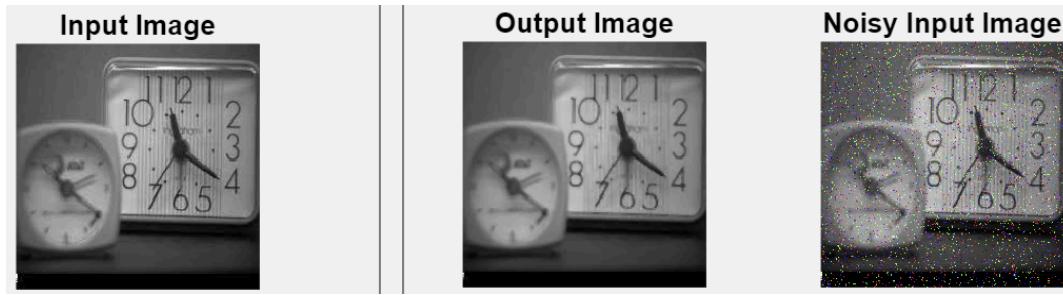
Gambar 2.92 Penghilangan derau *salt and pepper* citra jam dengan *min filter*

Penghilangan derau *salt and pepper* citra grayscale bangunan dengan *max filter* (1-3.jpg):



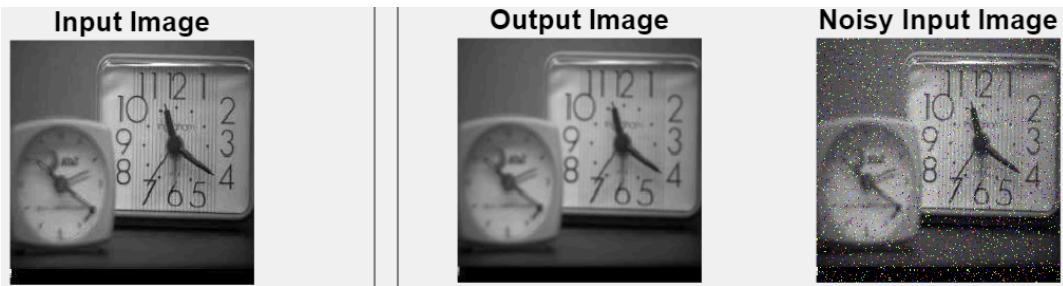
Gambar 2.93 Penghilangan derau *salt and pepper* citra jam dengan *max filter*

Penghilangan derau *salt and pepper* citra grayscale jam dengan *median filter* (1-3.jpg):



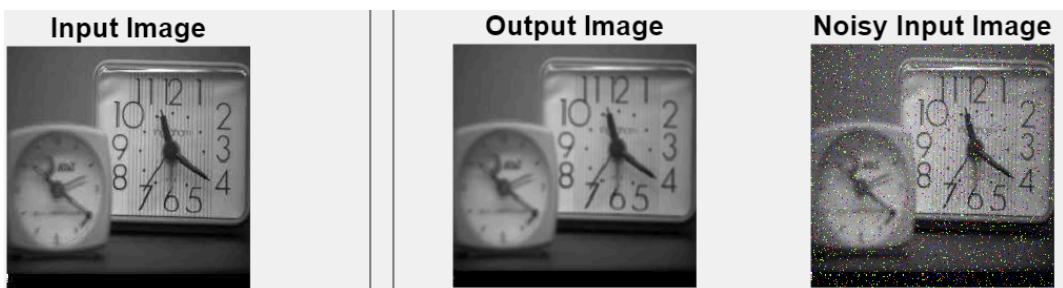
Gambar 2.94 Penghilangan derau *salt and pepper* citra jam dengan *median filter*

Penghilangan derau *salt and pepper* citra grayscale jam dengan *arithmetic mean filter* (1-3.jpg):



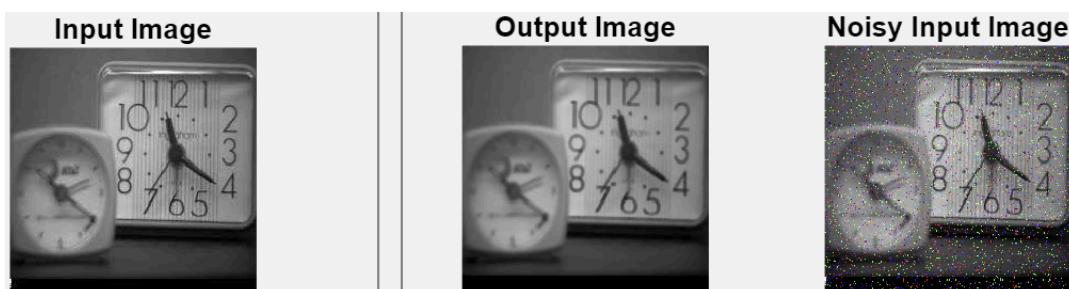
Gambar 2.95 Penghilangan derau *salt and pepper* citra jam dengan *arithmetic mean filter*

Penghilangan derau *salt and pepper* citra grayscale jam dengan *geometric mean filter* (1-3.jpg):



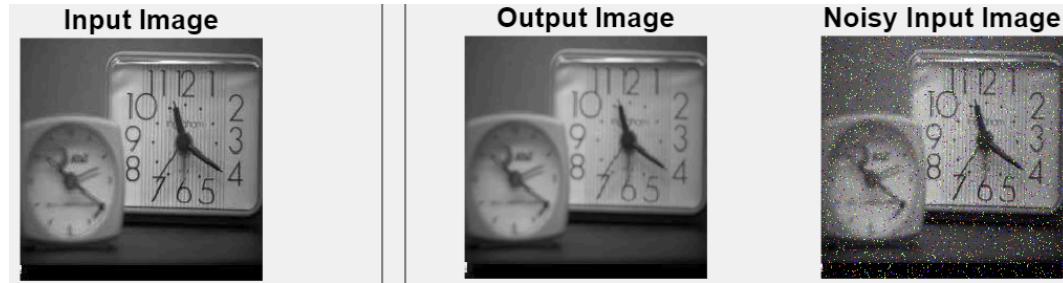
Gambar 2.96 Penghilangan derau *salt and pepper* citra jam dengan *geometric mean filter*

Penghilangan derau *salt and pepper* citra grayscale jam dengan *harmonic mean filter* (1-3.jpg):



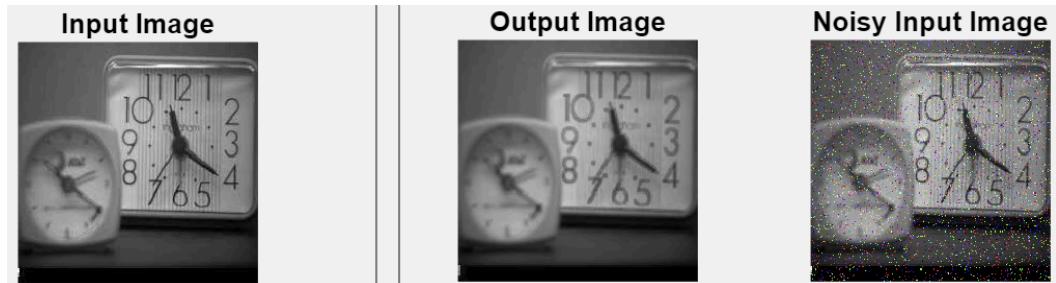
Gambar 2.97 Penghilangan derau *salt and pepper* citra jam dengan *harmonic mean filter*

Penghilangan derau *salt and pepper* citra grayscale jam dengan *contraharmonic mean filter* dengan $Q = 1.5$ (1-3.jpg):



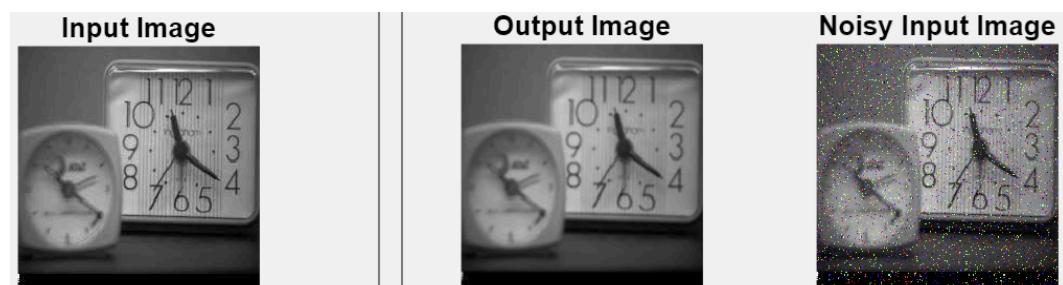
Gambar 2.98 Penghilangan derau *salt and pepper* citra jam dengan *contraharmonic mean filter* dengan $Q = 1.5$

Penghilangan derau *salt and pepper* citra grayscale jam dengan *midpoint filter* (1-3.jpg):



Gambar 2.99 Penghilangan derau *salt and pepper* citra jam dengan *midpoint filter*

Penghilangan derau *salt and pepper* citra grayscale jam dengan *alpha-trimmed mean filter* (1-3.jpg) dengan $d = 1$:



Gambar 2.100 Penghilangan derau *salt and pepper* citra jam dengan *alpha-trimmed mean filter* dengan $d = 1$

Penghilangan derau *salt and pepper* citra berwarna garam dengan *min filter* (5-2.jpg):



Gambar 2.101 Penghilangan derau *salt and pepper* citra garam dengan *min filter*

Penghilangan derau *salt and pepper* citra berwarna garam dengan *max filter* (5-2.jpg):



Gambar 2.102 Penghilangan derau *salt and pepper* garam dengan *max filter*

Penghilangan derau *salt and pepper* citra berwarna garam dengan *median filter* (5-2.jpg):



Gambar 2.103 Penghilangan derau *salt and pepper* citra garam dengan *median filter*

Penghilangan derau *salt and pepper* citra berwarna garam dengan *arithmetic mean filter* (5-2.jpg):



Gambar 2.104 Penghilangan derau *salt and pepper* citra garam dengan *arithmetic mean filter*

Penghilangan derau *salt and pepper* citra berwarna garam dengan *geometric mean filter* (5-2.jpg):



Gambar 2.105 Penghilangan derau *salt and pepper* citra garam dengan *geometric mean filter*

Penghilangan derau *salt and pepper* citra berwarna garam dengan *harmonic mean filter* (5-2.jpg):



Gambar 2.106 Penghilangan derau *salt and pepper* citra garam dengan *harmonic mean filter*

Penghilangan derau *salt and pepper* citra berwarna garam dengan *contraharmonic mean filter* dengan $Q = 1.5$ (5-2.jpg):



Gambar 2.107 Penghilangan derau *salt and pepper* citra garam dengan *contraharmonic mean filter* dengan $Q = 1.5$

Penghilangan derau *salt and pepper* citra berwarna garam dengan *midpoint filter* (5-2.jpg):



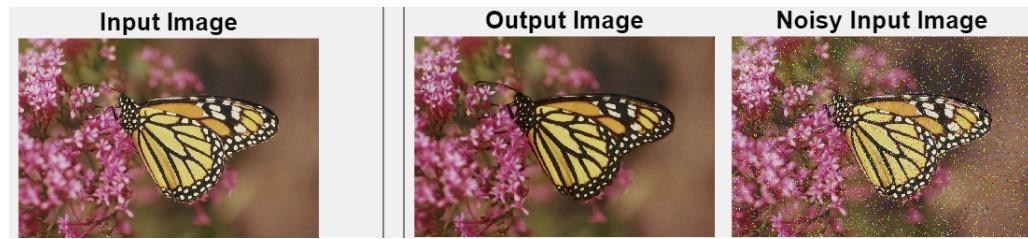
Gambar 2.108 Penghilangan derau *salt and pepper* citra garam dengan *midpoint filter*

Penghilangan derau *salt and pepper* citra berwarna garam dengan *alpha-trimmed mean filter* (5-2.jpg) dengan $d = 1$:



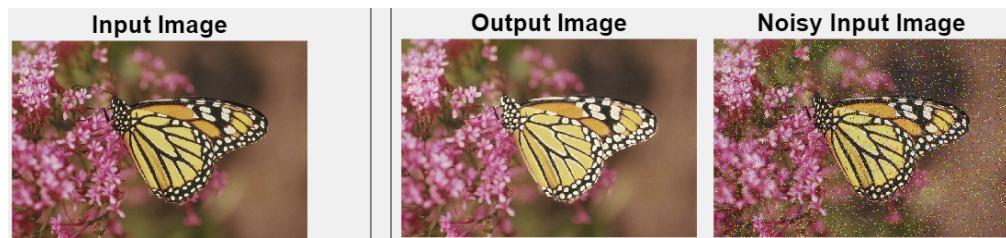
Gambar 2.109 Penghilangan derau *salt and pepper* citra garam dengan *alpha-trimmed mean filter* dengan $d = 1$

Penghilangan derau *salt and pepper* citra berwarna kupu-kupu dengan *min filter* (2-5.jpg):



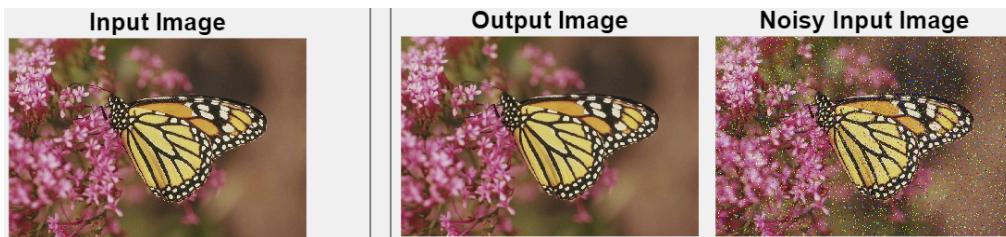
Gambar 2.110 Penghilangan derau *salt and pepper* citra kupu-kupu dengan *min filter*

Penghilangan derau *salt and pepper* citra berwarna kupu-kupu dengan *max filter* (2-5.jpg):



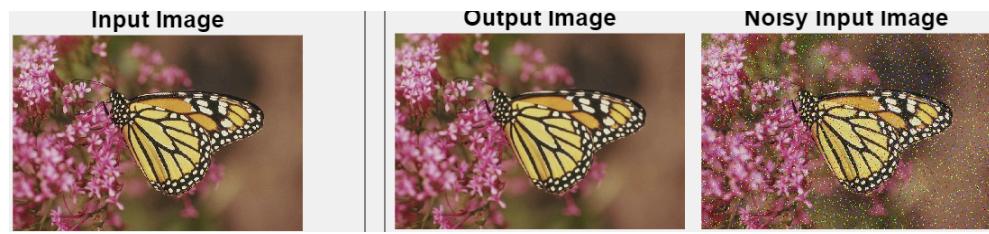
Gambar 2.111 Penghilangan derau *salt and pepper* kupu-kupu dengan *max filter*

Penghilangan derau *salt and pepper* citra berwarna kupu-kupu dengan *median filter* (2-5.jpg):



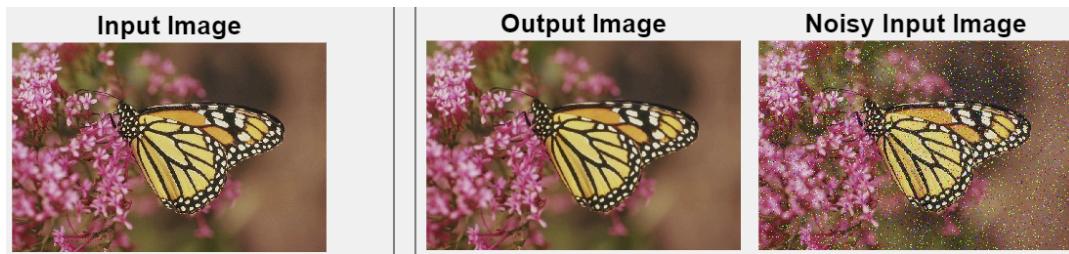
Gambar 2.112 Penghilangan derau *salt and pepper* kupu-kupu dengan *median filter*

Penghilangan derau *salt and pepper* citra berwarna kupu-kupu dengan *arithmetic mean filter* (2-5.jpg):



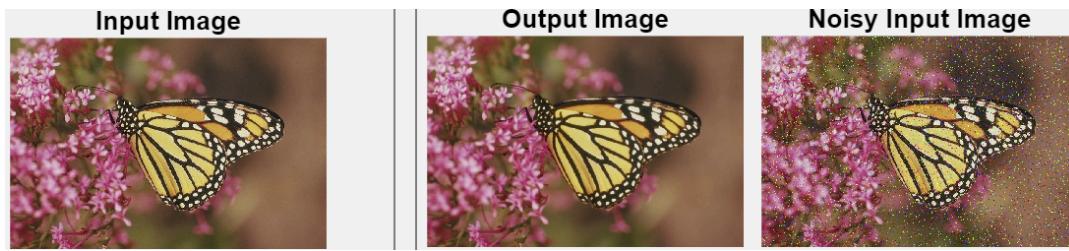
Gambar 2.113 Penghilangan derau *salt and pepper* citra kupu-kupu dengan *arithmetic mean filter*

Penghilangan derau *salt and pepper* citra berwarna kupu-kupu dengan *geometric mean filter* (2-5.jpg):



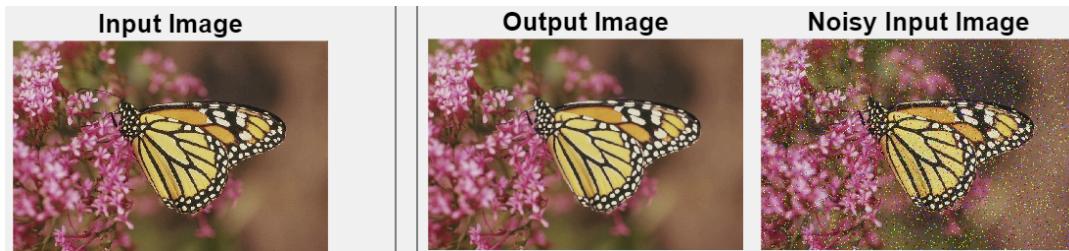
Gambar 2.114 Penghilangan derau *salt and pepper* citra kupu-kupu dengan *geometric mean filter*

Penghilangan derau *salt and pepper* citra berwarna kupu-kupu dengan *harmonic mean filter* (2-5.jpg):



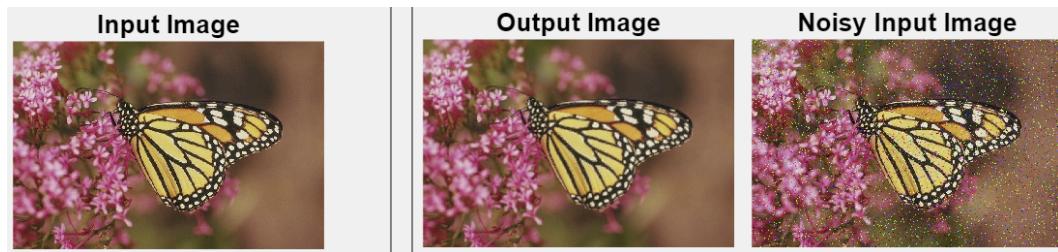
Gambar 2.115 Penghilangan derau *salt and pepper* citra kupu-kupu dengan *harmonic mean filter*

Penghilangan derau *salt and pepper* citra berwarna kupu-kupu dengan *contraharmonic mean filter* dengan $Q = 1.5$ (2-5.jpg):



Gambar 2.116 Penghilangan derau *salt and pepper* citra kupu-kupu dengan *contraharmonic mean filter* dengan $Q = 1.5$

Penghilangan derau *salt and pepper* citra berwarna kupu-kupu dengan *midpoint filter* (2-5.jpg):



Gambar 2.117 Penghilangan derau *salt and pepper* citra kupu-kupu dengan *midpoint filter*

Penghilangan derau *salt and pepper* citra berwarna kupu-kupu dengan *alpha-trimmed mean filter* (2-5.jpg) dengan $d = 1$:



Gambar 2.118 Penghilangan derau *salt and pepper* citra kupu-kupu dengan *alpha-trimmed mean filter* dengan $d = 1$

Analisis:

Setiap jenis penapis yang digunakan dalam fungsi ini memiliki efek yang berbeda pada citra yang dihilangkan deraunya. Penapis median sangat efektif dalam mengurangi derau impulsif (seperti *salt and pepper*) karena menghapus nilai ekstrem dan mempertahankan nilai tengah, sehingga menghasilkan citra yang lebih bersih tanpa mengaburkan detail penting. Di sisi lain, penapis minimum atau maksimum dapat mengurangi derau tetapi mungkin menghilangkan detail di daerah dengan kontras tinggi.

Penapis rata-rata (*mean*) menghilangkan derau dengan cara mengaburkan citra. *arithmetic mean* dan *geometric mean* cocok untuk menghilangkan derau acak (*Gaussian noise*). Penapis *geometric mean* lebih mengaburkan dan detail yang hilangnya lebih sedikit dibandingkan *arithmetic mean*. Sementara itu, penapis *harmonic mean* lebih cocok untuk menghilangkan derau *Gaussian* dan *salt*, tetapi kurang cocok untuk menghilangkan derau *pepper*.

Penapis *contraharmonic mean* tidak dapat meneliminasi derau *salt* dan *pepper* bersamaan. Derau *pepper* dapat dihilangkan ketika Q bernilai positif, sedangkan derau *salt* dapat dihilangkan ketika Q bernilai negatif. Ini cocok untuk derau impulsif.

Penapis *midpoint* mengganti nilai piksel di tengah *window* dengan pembagian antara jumlah nilai maksimum dan nilai minimum dengan 2 di dalam *window*.

Terakhir, penapis *alpha-trimmed mean* menghapuskan beberapa piksel lalu merata-ratakan sisanya. Semakin besar nilai d , semakin banyak elemen yang akan dihilangkan dari *window*, yang dapat mengurangi derau tetapi juga berisiko mengeliminasi informasi penting, terutama jika nilai d terlalu besar dibandingkan dengan ukuran *window*.

2.7. Penghilangan Derau Periodik

Berikut merupakan kode program fungsi untuk menghilangkan derau periodik citra.

```
function outputImage = autoPeriodicNoiseReduction(image, notchSize,
threshold)
    % Konversi ke grayscale jika berwarna
    if size(image, 3) == 3
        image = rgb2gray(image);
    end
    image = double(image);
    % Hitung FFT
    [rows, cols] = size(image);
    imageFFT = fftshift(fft2(image));
    magnitudeFFT = abs(imageFFT);
    % Deteksi spikes dengan threshold
    noisePeaks = magnitudeFFT > threshold * max(magnitudeFFT(:));
    noisePeaks = noisePeaks .* magnitudeFFT; % Mask
    % Inisialisasi notch filter mask dengan 1
    notchFilter = ones(rows, cols);
    sigma = notchSize; % gaussian
    centerRow = floor(rows / 2) + 1;
    centerCol = floor(cols / 2) + 1;
    % Iterasi untuk setiap spike yang terdeteksi
    [peakRows, peakCols] = find(noisePeaks);
    for k = 1:length(peakRows)
        row = peakRows(k);
        col = peakCols(k);
        % Melewati spike di tengah
        if abs(row - centerRow) < notchSize && abs(col - centerCol) <
notchSize
            continue;
        end
        % Buat lingkaran pada posisi spike
        for r = row - notchSize/2:row + notchSize/2
            for c = col - notchSize/2:col + notchSize/2
                if (r - row)^2 + (c - col)^2 <= notchSize^2
                    notchFilter(r, c) = 0;
                end
            end
        end
    end
    outputImage = ifft2(notchFilter * imageFFT);
    outputImage = abs(outputImage);
    outputImage = uint8(outputImage);
end
```

```

    end

    % Menghitung penapisan Gaussian
    for i = 1:rows
        for j = 1:cols
            d = sqrt((i - row)^2 + (j - col)^2);
            notchFilter(i, j) = notchFilter(i, j) * (1 - exp(-d^2 /
(2 * sigma^2)));
        end
    end
    % Masking ke citra
    filteredFFT = imageFFT .* notchFilter;
    % Invers FFT ke ranah spasial
    filteredImage = real(ifft2(ifftshift(filteredFFT)));
    % Normalisasi dengan range dan intensitas aslinya
    filteredImage = filteredImage - min(filteredImage(:));
    filteredImage = filteredImage * (255 / max(filteredImage(:)));

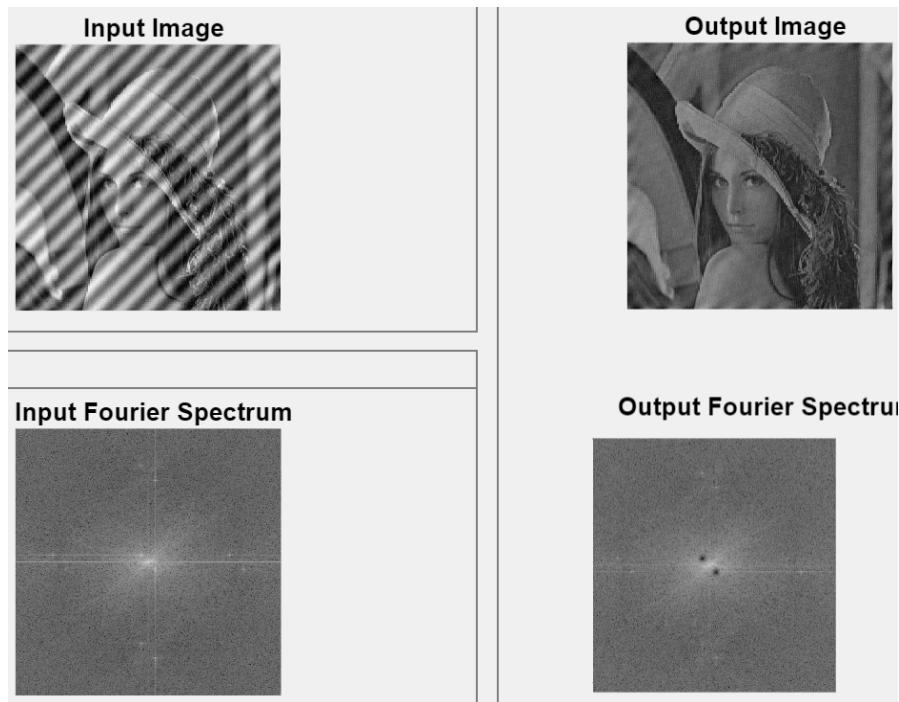
    % Convert to uint8 format for display
    outputImage = uint8(filteredImage);
end

```

Fungsi `autoPeriodicNoiseReduction` mengurangi derau periodik secara otomatis dari sebuah citra menggunakan transformasi *Fourier* dengan menerima parameter citra (`image`), ukuran *notch* (`notchSize`), dan *threshold*. Pertama, citra diubah ke dalam *grayscale* jika citra berwarna, kemudian dihitung transformasi *Fourier*-nya. Dengan menggunakan *threshold* tertentu, program mendeteksi puncak (*spikes*) pada spektrum magnitudo yang menunjukkan adanya derau periodik. Untuk setiap puncak yang terdeteksi, dilakukan penapisan *notch* berbasis *Gaussian* pada titik-titik tertentu untuk mengurangi derau tersebut. Setelah itu, hasil *filter inverse FFT* dikonversi kembali ke ranah spasial, distandarisasi, dan dikembalikan sebagai citra keluaran.

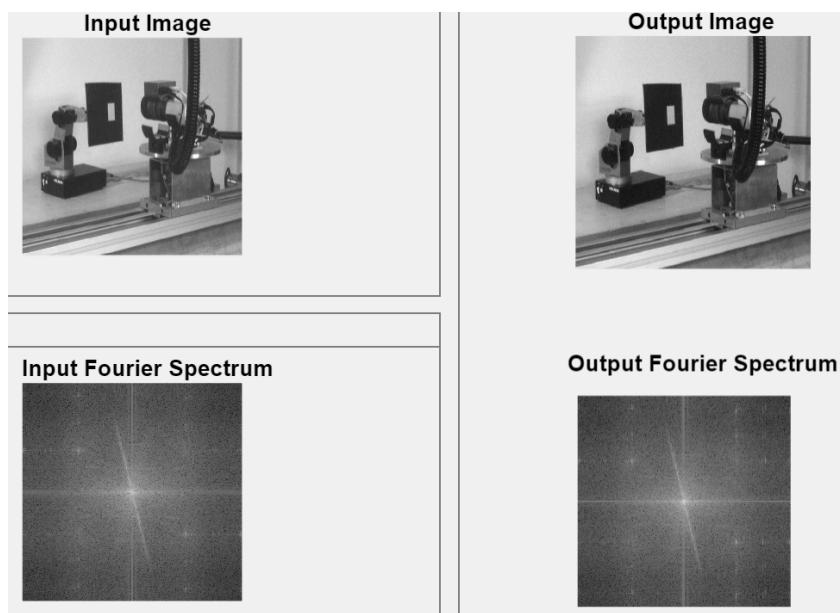
Berikut merupakan contoh hasil eksekusi program.

Penghilangan derau periodik citra *grayscale* Lenna (6-1.jpg) dengan *threshold* 0.04 dan ukuran *notch* 4:



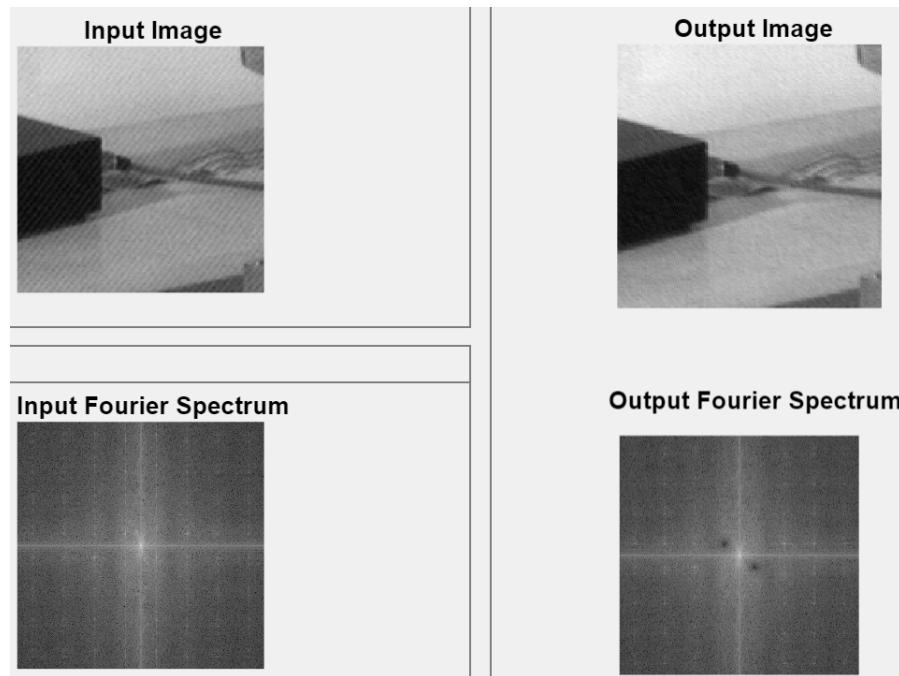
Gambar 2.119 Penghilangan derau periodik citra Lenna (2)

Penghilangan derau periodik citra *grayscale* peralatan lab (6-2.jpg) dengan *threshold* 0.017 dan ukuran *notch* 15:



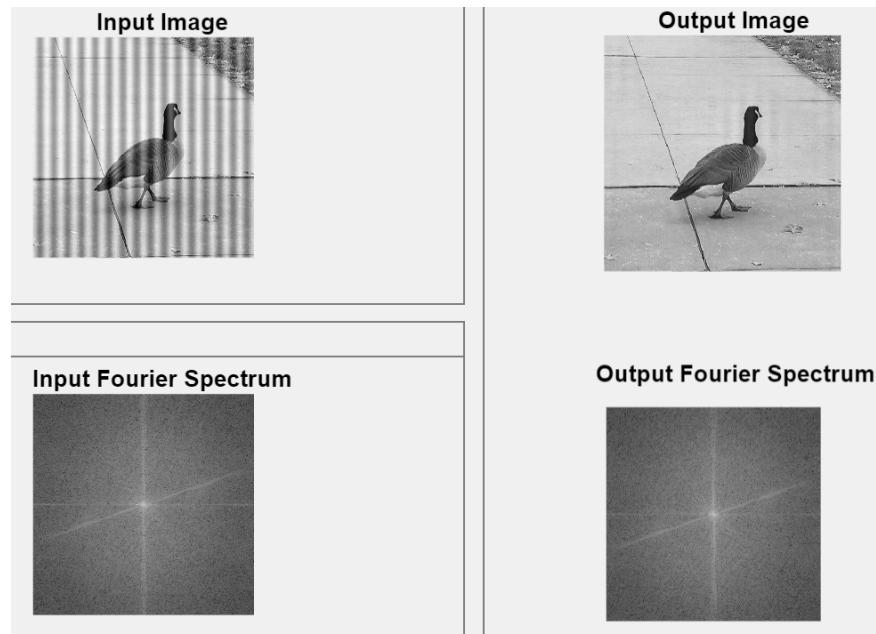
Gambar 2.120 Penghilangan derau periodik citra peralatan lab

Penghilangan derau periodik citra *grayscale* perbesaran dari citra 6-2 (6-3.jpg) dengan *threshold* 0.01 dan ukuran *notch* 15:



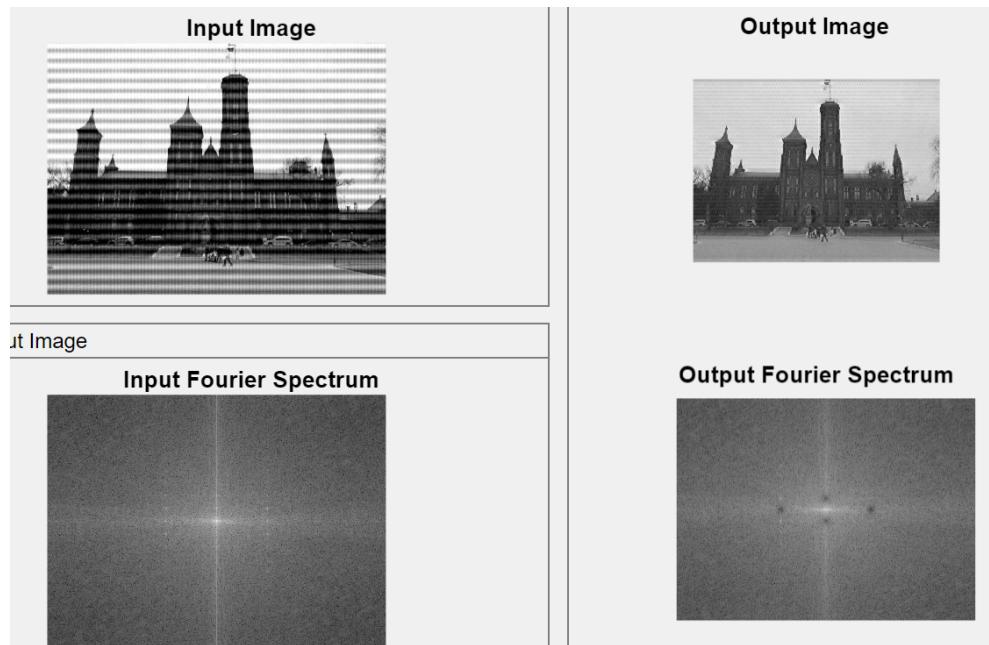
Gambar 2.121 Penghilangan derau periodik citra perbesaran citra 6-2

Penghilangan derau periodik citra *grayscale* angsa (6-4.jpg) dengan *threshold* 0.05 dan ukuran *notch* 2:



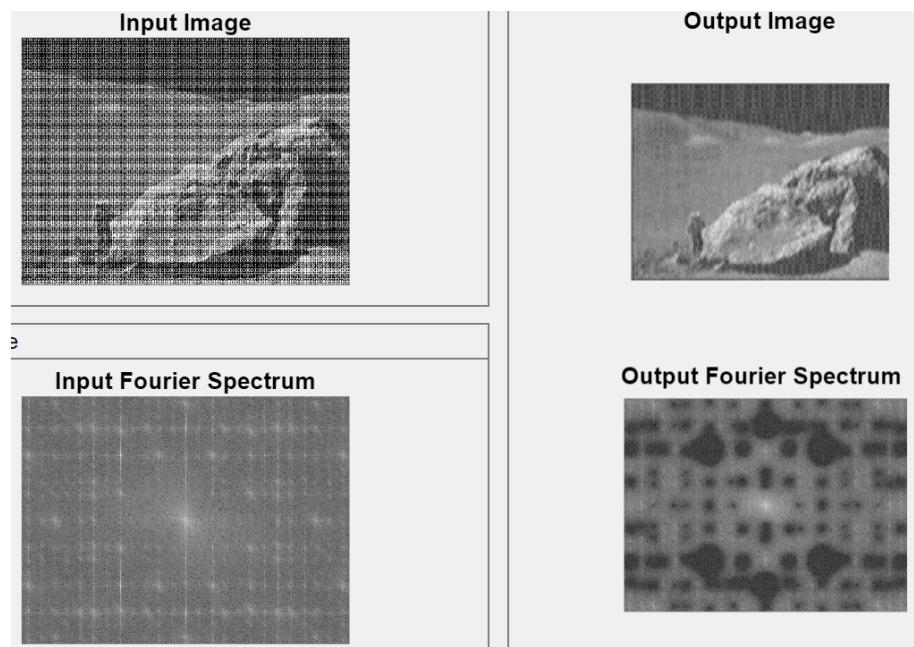
Gambar 2.122 Penghilangan derau periodik citra angsa

Penghilangan derau periodik citra *grayscale* bangunan (6-5.jpg) dengan *threshold* 0.025 dan ukuran *notch* 11:



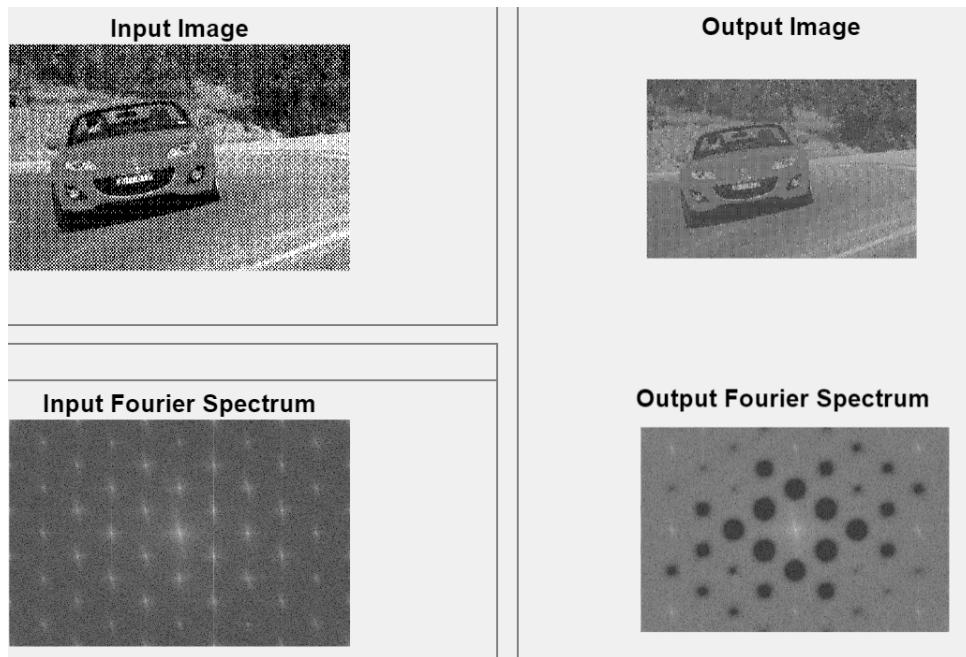
Gambar 2.123 Penghilangan derau periodik citra bangunan

Penghilangan derau periodik citra *grayscale* astronot di bulan (6-6.jpg) dengan *threshold* 0.01 dan ukuran *notch* 17:



Gambar 2.124 Penghilangan derau periodik citra astronot di bulan

Penghilangan derau periodik citra *grayscale* mobil (6-7.jpg) dengan *threshold* 0.014 dan ukuran *notch* 17:



Gambar 2.125 Penghilangan derau periodik citra mobil

Analisis:

Dengan penghilangan derau periodik menggunakan pendekatan *Gaussian*, citra tampak lebih halus, meskipun derau sisa mungkin tetap terlihat jika *filter notch* tidak cukup kuat untuk menapis semua puncak. Penapisan *Gaussian* digunakan karena menghasilkan citra yang lebih halus dibandingkan penapisan ideal.

Semakin besar ukuran *notch*-nya, semakin besar area sekitar *spike* yang terpengaruh oleh *filter*, yang dapat meningkatkan hasil pengurangan derau. Namun, jika ukuran *notch* terlalu besar atau *threshold* tidak optimal, gambar bisa menjadi lebih gelap atau detail penting berkurang karena *filter* menangkap frekuensi yang lebih luas, bahkan yang relevan. Agar hasil lebih optimal, memilih *threshold* yang tepat dan menyesuaikan ukuran *notch* adalah kunci untuk meminimalkan sisa derau periodik tanpa mengorbankan terlalu banyak detail asli gambar.

2.8. *Motion Blurring* dan Dekonvolusi

Berikut merupakan kode program fungsi untuk melakukan *motion blurring* lalu melakukan dekonvolusi pada citra dengan penapis Wiener.

```

function outputImage = deconvolution(image, needBluring, noise, PSF,
LEN, blurEffect)
    %Lakukan bluring
    if needBluring
        blurred = imfilter(image, PSF, 'conv', 'circular');
    else
        blurred = image;
    end

    % Tambahkan noise jika diperlukan
    if noise
        noise_mean = 0;
        noise_var = 0.0001;
        blurred_noisy = imnoise(blurred, 'gaussian', noise_mean,
noise_var);
    else
        blurred_noisy = blurred;
    end
    % Restorasi dengan dekonvolusi Wiener
    % Hitung estimated_nsr berdasarkan ada atau tidaknya noise
    if noise
        signal_variance = var(image(:)); % Hitung variansi sinyal dari
gambar asli
        if signal_variance > 0
            estimated_nsr = noise_var / signal_variance;
        else
            estimated_nsr = noise_var; % Gunakan nilai default jika
variansi sinyal terlalu kecil
        end
    else
        estimated_nsr = 0; % Asumsi tanpa noise
    end

    % Lakukan dekonvolusi dengan fungsi Wiener
    outputImage = wiener(blurred_noisy, PSF,
estimated_nsr,LEN,blurEffect);
end

function outputImage = wiener(image, PSF, k, len, blurEffect)
    temp = zeros(size(image,1), size(image,2), 'double');
    image = im2double(image);
    PSF = im2double(PSF);

    % Ubah ke bentuk frekuensi fourier
    freqImage = fft2(image);
    freqPSF = fft2(PSF, size(image,1), size(image,2));
    % Dekonvolusi Wiener
    wienerFilter = (conj(freqPSF) ./ (abs(freqPSF).^2 + k));
    F = freqImage .* wienerFilter;
    % Invers
    inverseImage = ifft2(F);
    realImage = real(inverseImage);
    img = max(0, min(1, realImage));

```

```

% Memperbaiki pixel yang bergeser
if (strcmp(blurEffect,'Motion'))
    for i = 1:size(img,1)
        for j = 1:size(img,2)
            for k = 1:size(img,3)
                rows = i;
                cols = mod(j + len/2,size(img,2));

                if rows == 0
                    rows = rows + size(img,1);
                end

                if cols == 0
                    cols = cols + size(img,2);
                end

                temp(rows,cols,k) = double(img(i,j,k));

            end
        end
    end
else
    for i = 1:size(img,1)
        for j = 1:size(img,2)
            for k = 1:size(img,3)
                rows = mod(i + floor(len/2),size(img,1));
                cols = mod(j + floor(len/2),size(img,2));

                if rows == 0
                    rows = rows + size(img,1);
                end

                if cols == 0
                    cols = cols + size(img,2);
                end

                temp(rows,cols,k) = double(img(i,j,k));

            end
        end
    end
end
outputImage = temp;
end

function outputImage = motionBluring(image, psf)
    outputImage = imfilter(image,psf,'conv','circular');
end

```

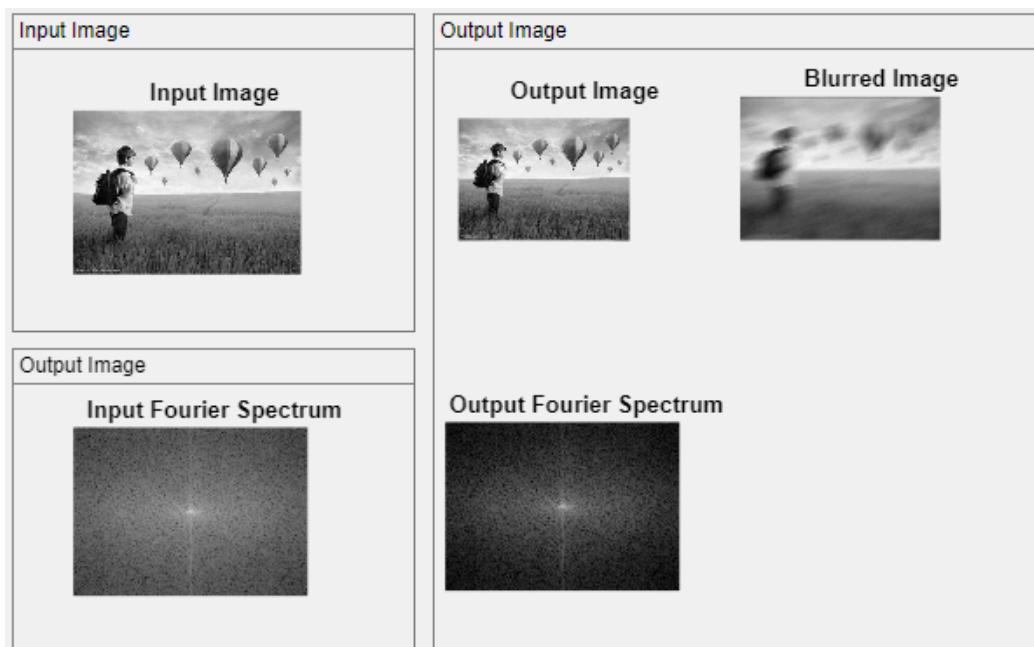
Program dekonvolusi wiener ini memiliki tiga fungsi, yaitu deconvolution, wiener, dan motionBluring. deconvolution adalah fungsi yang mengandung prosedur utama dalam melakukan dekonvolusi suatu citra. Pada fungsi ini, dapat dimasukkan

parameter yang menentukan apakah *blurring* perlu dilakukan terlebih dahulu sebelum dilakukan dekonvolusi. Fungsi yang digunakan untuk melakukan *blurring* ini adalah motionBluring, yang memanfaatkan fungsi imfilter dari MATLAB. Dapat dimasukkan juga parameter *noise*, yang adalah anggapan dari pengguna apakah gambar memiliki noise atau tidak. Saat pengguna memilih untuk melakukan *blurring* terlebih dahulu, citra input akan ditambahkan noise *gaussian*.

Setelah semua proses itu dilakukan, citra dan PSF akan dimasukkan ke dalam fungsi wiener. Di dalam fungsi ini akan diterapkan wiener untuk melakukan dekonvolusi. Proses ini dilakukan dalam ranah frekuensi, sehingga citra perlu diubah ke dalam bentuk spektrum *fourier* dengan Fast Fourier Transform terlebih dahulu.

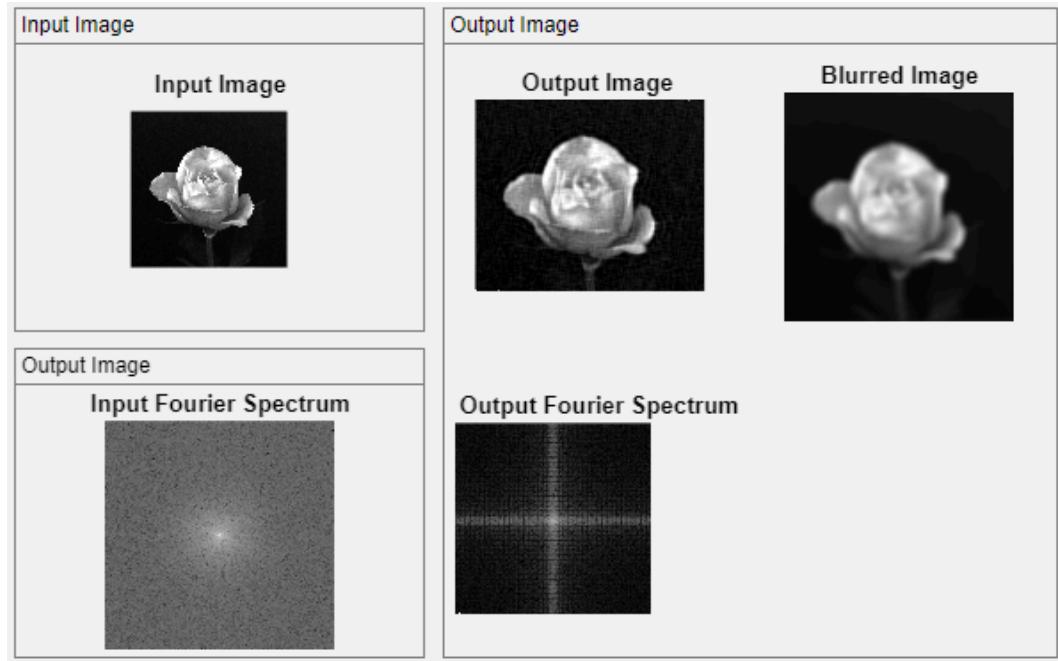
Berikut merupakan contoh hasil eksekusi program.

Motion blurring dan dekonvolusi citra *grayscale* balon udara (7-1.jpg):



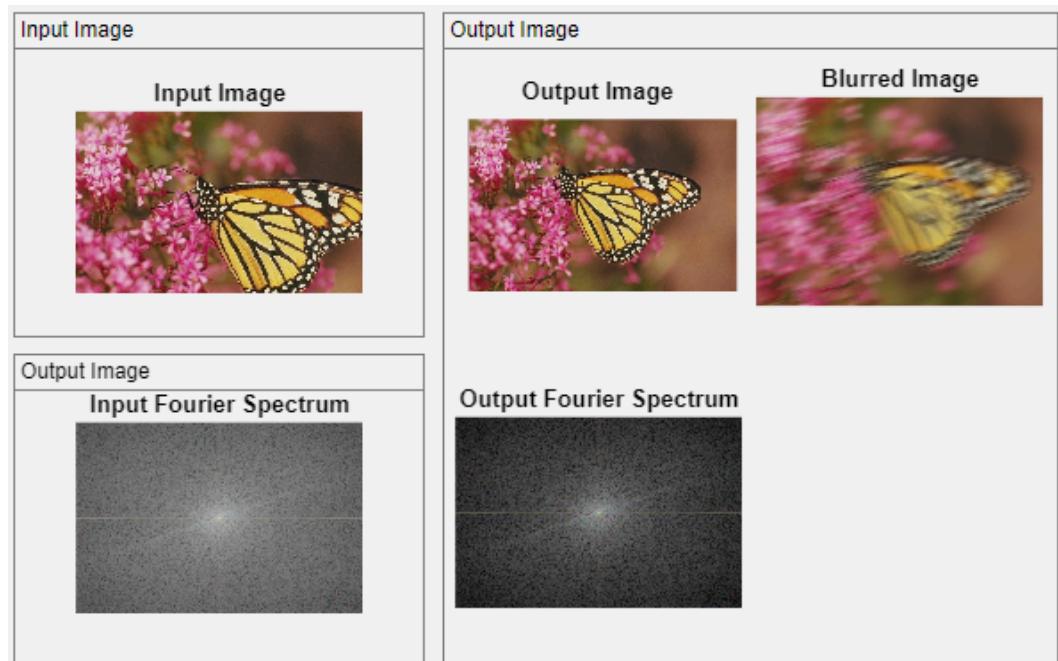
Gambar 2.126 *Motion blurring* dan dekonvolusi citra balon udara

Motion blurring dan dekonvolusi citra grayscale mawar (7-4.tif):



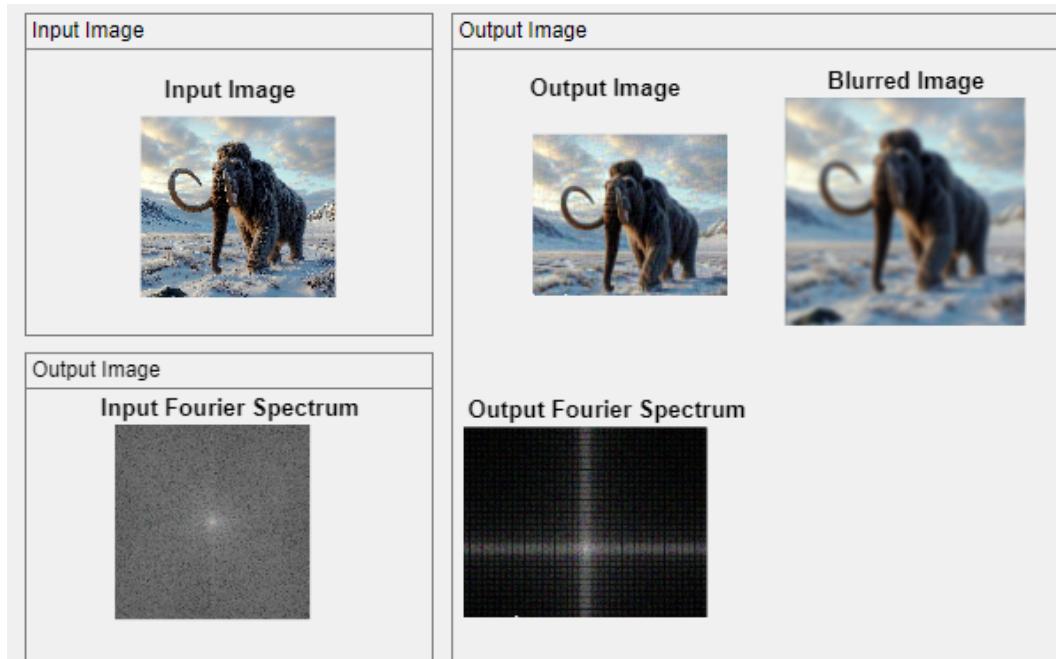
Gambar 2.127 *Motion blurring* dan dekonvolusi citra mawar (7-4.tif)

Motion blurring dan dekonvolusi citra berwarna kupu-kupu (7-2.jpg):



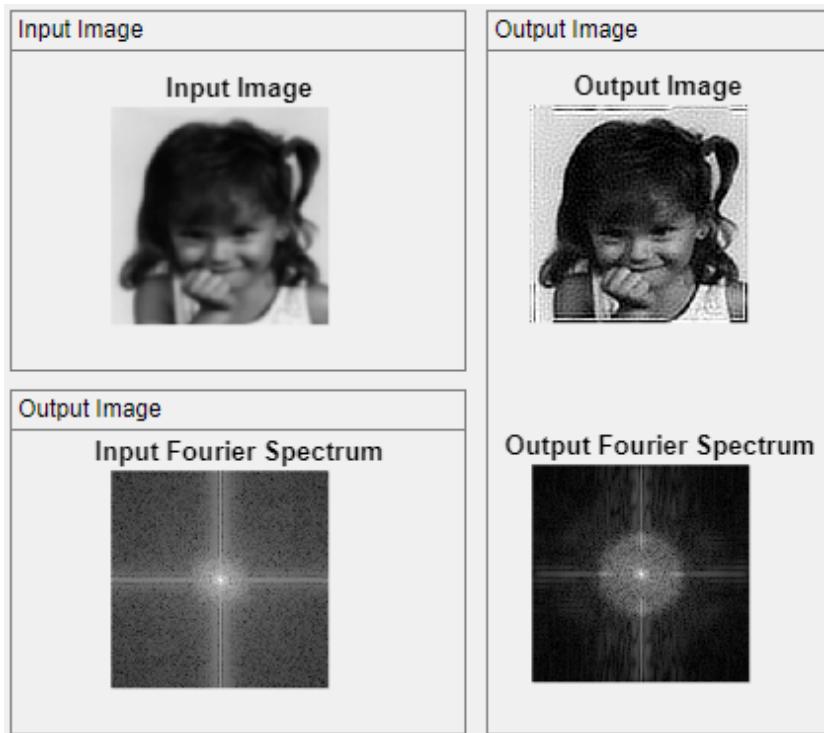
Gambar 2.128 *Motion blurring* dan dekonvolusi citra kupu-kupu

Motion blurring dan dekonvolusi citra berwarna mammoth (7-5.jpg):



Gambar 2.129 *Motion blurring* dan dekonvolusi citra mammoth (7-5.jpg)

Dekonvolusi citra *grayscale* anak (7-3.jpg):



Gambar 2.130 Dekonvolusi citra anak

Analisis:

Dapat dilihat bahwa gambar hasil blur berhasil diubah kembali ke bentuk semula dengan fungsi wiener ini. Awalnya, citra input ditransformasi dengan efek blur *motion* dengan parameter *len* = 44 dan *theta* = 16. Transformasi ini menghasilkan citra blur yang tampak seperti potret gambar yang sedang bergerak. Kemudian dilakukan dekonvolusi dengan penapis wiener. Dekonvolusi ini berhasil mengembalikan citra mendekati citra semula.

Untuk dekonvolusi terhadap citra anak kecil, digunakan ukuran matriks (hsize) 80x80 dan variansi (sigma) 2,6. Diasumsikan gambar tersebut memiliki derau dan mengalami blur *gaussian*. Dengan parameter tersebut, diperoleh citra anak kecil yang blurnya sudah berkurang dan lebih jelas.

3. Lampiran

Berikut merupakan tautan repositori GitHub:

https://github.com/arleenchr/IF4073_Tugas2