

# **LAPORAN TUGAS II**

## **PEMROSESAN CITRA DIGITAL**

### **Penapisan citra dalam ranah spasial, ranah frekuensi, dan restorasi citra**

Diajukan sebagai tugas Mata Kuliah IF4073 Pemrosesan Citra Digital



Disusun Oleh:

Jason Rivalino 13521008

Juan Christopher Santoso 13521116

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
BANDUNG**

**2024**

## DAFTAR ISI

<b>DAFTAR ISI.....</b>	<b>2</b>
<b>DAFTAR GAMBAR.....</b>	<b>3</b>
<b>BAB I</b>	
<b>DESKRIPSI MASALAH.....</b>	<b>5</b>
<b>BAB II</b>	
<b>GRAPHICAL USER INTERFACE (GUI) PROGRAM.....</b>	<b>7</b>
<b>BAB III</b>	
<b>PENJELASAN KODE DAN PENGUJIAN PROGRAM.....</b>	<b>12</b>
I. Kode Program dan Penjelasan.....	12
A. Kode program fungsi utama.....	12
B. Kode program fungsi pendukung (utils).....	20
C. Kode program fungsi operasi matriks.....	23
D. Kode program fungsi penapis Wiener.....	27
II. Hasil Pengujian Program.....	29
A. Pengujian program untuk konvolusi citra.....	29
B. Pengujian program untuk Image Smoothing dengan ranah spasial.....	32
C. Pengujian program untuk Image Smoothing dengan ranah frekuensi Low-Pass Filter.....	34
D. Pengujian program untuk Edge Detection dengan ranah frekuensi High-Pass Filter.....	37
E. Pengujian program untuk Image Brightening dengan ranah frekuensi Homomorphic Filter....	39
F. Pengujian program untuk Noise Filtering.....	40
G. Pengujian program untuk Periodic Noise Restoration.....	43
H. Pengujian program untuk memperbaiki Motion Blur dengan Penapis Wiener.....	46
III. Analisis Cara Kerja Program.....	48
<b>BAB IV</b>	
<b>KESIMPULAN.....</b>	<b>56</b>
<b>DAFTAR PUSTAKA.....</b>	<b>57</b>
<b>LAMPIRAN.....</b>	<b>58</b>

## DAFTAR GAMBAR

Gambar 2.1 Tampilan GUI dari program untuk melakukan proses konvolusi.....	6
Gambar 2.2 Tampilan GUI dari program untuk melakukan proses Image Smoothing dalam ranah spasial dengan memanfaatkan Mean & Gaussian Filter.....	7
Gambar 2.3 Tampilan GUI dari program untuk melakukan proses Image Smoothing dalam ranah frekuensi dengan memanfaatkan Low-Pass Filter.....	7
Gambar 2.4 Tampilan GUI dari program untuk melakukan proses Edge Detection dengan memanfaatkan High-Pass Filter.....	8
Gambar 2.5 Tampilan GUI dari program untuk melakukan proses Image Brightening dengan memanfaatkan Homomorphic.....	8
Gambar 2.6 Tampilan GUI dari program untuk melakukan proses Noise Filtering.....	9
Gambar 2.7 Tampilan GUI dari program untuk melakukan proses Periodic Noise Restoration.....	9
Gambar 2.8 Tampilan GUI dari program untuk melakukan proses dekonvolusi dengan penapis Wiener..	10
Gambar 3.1 Pengujian konvolusi untuk gambar grayscale butterfly.jpg.....	28
Gambar 3.2 Pengujian konvolusi untuk gambar grayscale clock.jpg.....	29
Gambar 3.3 Pengujian konvolusi untuk gambar berwarna snow_mountain.jpg.....	30
Gambar 3.4 Pengujian konvolusi untuk gambar berwarna snow_trees.jpg.....	30
Gambar 3.5 Pengujian Image Smoothing untuk gambar grayscale boat.bmp dengan ranah spasial Mean.	31
Gambar 3.6 Pengujian Image Smoothing untuk gambar grayscale boat.bmp dengan ranah spasial Gaussian.....	31
Gambar 3.7 Pengujian Image Smoothing untuk gambar berwarna gedungSate.png dengan ranah spasial Mean.....	32
Gambar 3.8 Pengujian Image Smoothing untuk gambar berwarna gedungSate.png dengan ranah spasial Gaussian.....	32
Gambar 3.9 Pengujian Image Smoothing untuk gambar grayscale castle.png dengan ranah frekuensi Ideal Low-Pass Filter.....	33
Gambar 3.10 Pengujian Image Smoothing untuk gambar grayscale saturn.png dengan ranah frekuensi Gaussian Low-Pass Filter.....	33
Gambar 3.11 Pengujian Image Smoothing untuk gambar grayscale woman.tif dengan ranah frekuensi Butterworth Low-Pass Filter.....	34
Gambar 3.12 Pengujian Image Smoothing untuk gambar berwarna monarch.bmp dengan ranah frekuensi Ideal Low-Pass Filter.....	34
Gambar 3.13 Pengujian Image Smoothing untuk gambar berwarna galaxy.png dengan ranah frekuensi Gaussian Low-Pass Filter.....	35
Gambar 3.14 Pengujian Image Smoothing untuk gambar berwarna baboon24.bmp dengan ranah frekuensi Butterworth Low-Pass Filter.....	35
Gambar 3.15 Pengujian Edge Detection untuk gambar grayscale cat.png dengan ranah frekuensi Ideal High-Pass Filter.....	36
Gambar 3.16 Pengujian Edge Detection untuk gambar grayscale gedung.bmp dengan ranah frekuensi Gaussian High-Pass Filter.....	36
Gambar 3.17 Pengujian Edge Detection untuk gambar grayscale lake_gray.jpg dengan ranah frekuensi Butterworth High-Pass Filter.....	37

Gambar 3.18 Pengujian Edge Detection untuk gambar berwarna board.png dengan ranah frekuensi Ideal High-Pass Filter.....	37
Gambar 3.19 Pengujian Edge Detection untuk gambar berwarna strawberries_coffee.tif dengan ranah frekuensi Gaussian High-Pass Filter.....	38
Gambar 3.20 Pengujian Image Brightening untuk gambar grayscale cave.png dengan ranah frekuensi Homomorphic Filter.....	38
Gambar 3.21 Pengujian Image Brightening untuk gambar berwarna flower.png dengan ranah frekuensi Homomorphic Filter.....	39
Gambar 3.22 Pengujian Noise Filtering church.jpg dengan derau salt and pepper dan filter min.....	39
Gambar 3.23 Pengujian Noise Filtering church.jpg dengan derau salt and pepper dan filter max.....	40
Gambar 3.24 Pengujian Noise Filtering church.jpg dengan derau gaussian dan filter median.....	40
Gambar 3.25 Pengujian Noise Filtering salt.jpg dengan derau salt and pepper dan filter geometric mean	41
Gambar 3.26 Pengujian Noise Filtering salt.jpg dengan derau gaussian dan filter harmonic mean.....	41
Gambar 3.27 Pengujian Noise Filtering salt.jpg dengan derau gaussian dan filter midpoint.....	42
Gambar 3.28 Pengujian Periodic Noise Restoration tools_noise.jpg dengan menggunakan filter band-reject.....	42
Gambar 3.29 Pengujian Periodic Noise Restoration car_noise.jpg dengan menggunakan filter band-pass	43
Gambar 3.30 Pengujian Periodic Noise Restoration lena_noise.jpg dengan menggunakan filter notch....	43
Gambar 3.31 Pengujian Periodic Noise Restoration building_noise.jpg dengan menggunakan filter band-pass.....	44
Gambar 3.32 Pengujian Periodic Noise Restoration duck_noise.jpg dengan menggunakan filter Notch...	44
Gambar 3.33 Pengujian Periodic Noise Restoration rock_noise.jpg dengan menggunakan filter band-pass.	45
Gambar 3.34 Pengujian program untuk memperbaiki Motion Blur dengan Penapis Wiener untuk gambar camera.bmp.....	45
Gambar 3.35 Pengujian program untuk memperbaiki Motion Blur dengan Penapis Wiener untuk gambar monarch.bmp.....	46
Gambar 3.36 Pengujian program untuk memperbaiki Motion Blur dengan Penapis Wiener untuk gambar bocilCewek.png.....	46
Gambar 3.37 Matriks Mean yang terbentuk dari masukan nMask 3, 5, dan 7.....	48
Gambar 3.38 Matriks Gaussian yang terbentuk dari masukan nMask 3 dan 7.....	49

## **BAB I**

### **DESKRIPSI MASALAH**

Pada tugas 2 dari mata kuliah IF4073 Pemrosesan Citra Digital, terdapat sejumlah program yang harus dibuat dengan menggunakan MATLAB antara lain sebagai berikut:

1. Melakukan konvolusi citra  $f$  berukuran sembarang ( $M \times N$ ) dengan mask berukuran  $n \times n$ , baik pada citra grayscale maupun citra berwarna. Fungsi konvolusi tidak boleh menggunakan fungsi built-in dari Matlab (seperti conv), tetapi dibuat sendiri. Bandingkan hasil fungsi konvolusi buatan anda dengan fungsi konvolusi dari Matlab. Mekanisme untuk konvolusi pixel-pixel pinggir dibebaskan kepada anda. Contoh  $n = 3, 5, 7$ , dsb.
  
2. Membuat program Matlab untuk melakukan image smoothing (pada citra yang mengandung derau) atau blurring pada citra dalam:
  - Ranah spasial menggunakan mean filter  $n \times n$  dan gaussian filter  $n \times n$ . Gunakan fungsi konvolusi yang telah anda kerjakan pada poin 1.
  - Ranah frekuensi menggunakan low-pass filter ILPF, GLPF, dan BLPF.  
Citra yang digunakan adalah citra grayscale dan citra berwarna.
  
3. Membuat program Matlab untuk melakukan penapisan citra dalam ranah frekuensi menggunakan high-pass filter IHPF, GHPF, dan BHPF. Citra yang digunakan adalah citra grayscale dan citra berwarna.
  
4. Membuat program Matlab untuk melakukan penapisan citra dalam ranah frekuensi untuk menghasilkan citra yang lebih terang
  
5. Membuat program Matlab untuk menambahkan derau salt and pepper pada citra (grayscale dan berwarna), lalu lakukan penghilangan derau dengan min filter, max filter, median filter, arithmetic mean filter, geometric filter, harmonic mean filter, contraharmonic mean filter, midpoint filter, alpha-trimmed mean filter. Tidak boleh menggunakan fungsi medfilt2 di dalam Matlab.

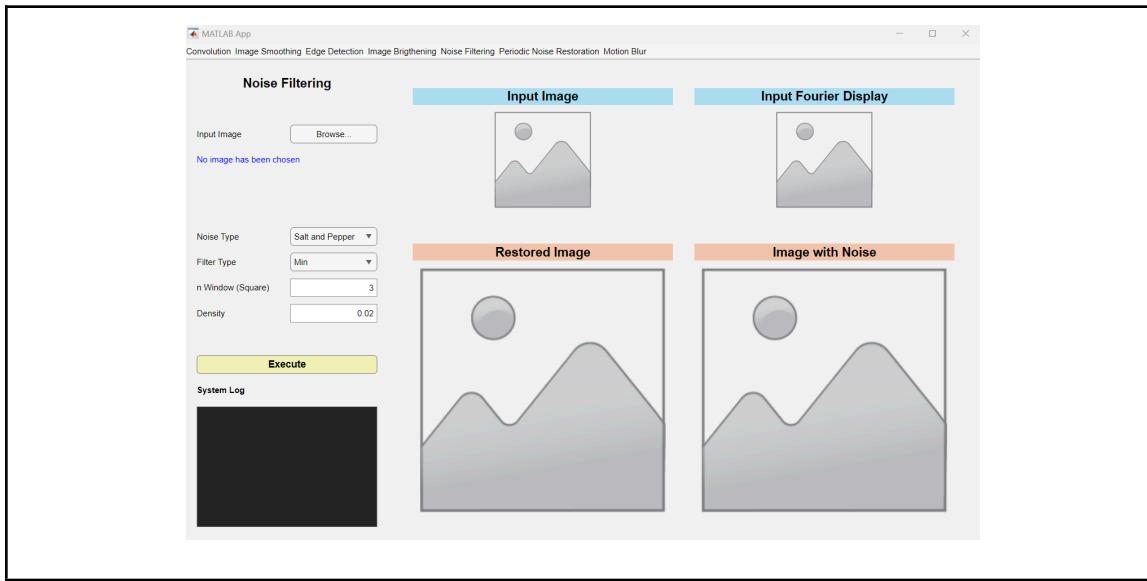
6. Membuat program Matlab untuk menghilangkan derau periodik pada citra yang ada pada gambar
7. Membuat program Matlab untuk melakukan motion bluring pada citra (grayscale dan berwarna), lalu lakukan dekonvolusi pada citra tersebut dengan penapis Wiener. Program penapis Wiener anda buat sendiri (tidak boleh menggunakan fungsi Wiener di dalam Matlab).

## BAB II

### ***GRAPHICAL USER INTERFACE (GUI) PROGRAM***

Berikut merupakan tampilan layar (*screenshot*) untuk GUI dari program yang dibuat oleh kelompok kami dengan menggunakan MATLAB:

#### A. Layar *Convolution*

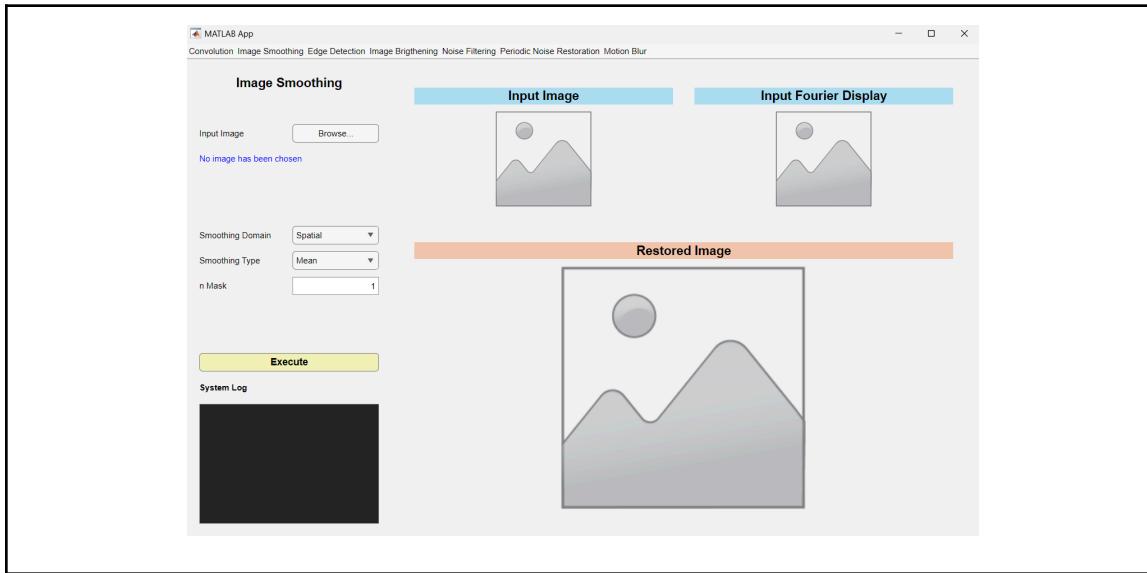


Gambar 2.1 Tampilan GUI dari program untuk melakukan proses konvolusi

Sumber: Dokumen Penulis

## B. Layar *Image Smoothing*

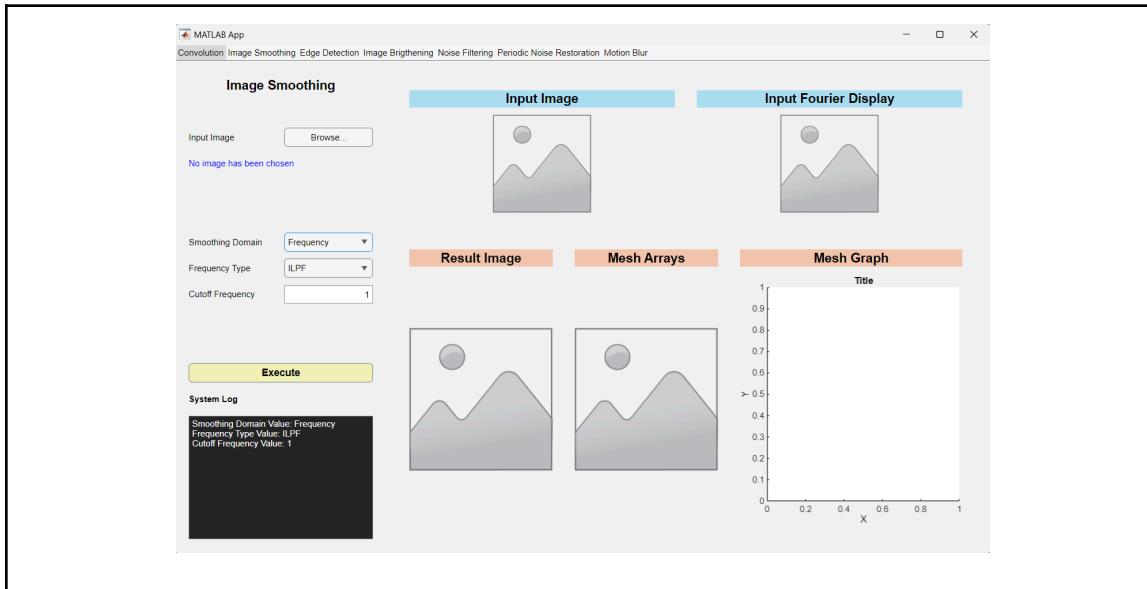
### a. *Image Smoothing* dalam ranah spasial dengan memanfaatkan *Mean & Gaussian Filter*



Gambar 2.2 Tampilan GUI dari program untuk melakukan proses *Image Smoothing* dalam ranah spasial dengan memanfaatkan *Mean & Gaussian Filter*

Sumber: Dokumen Penulis

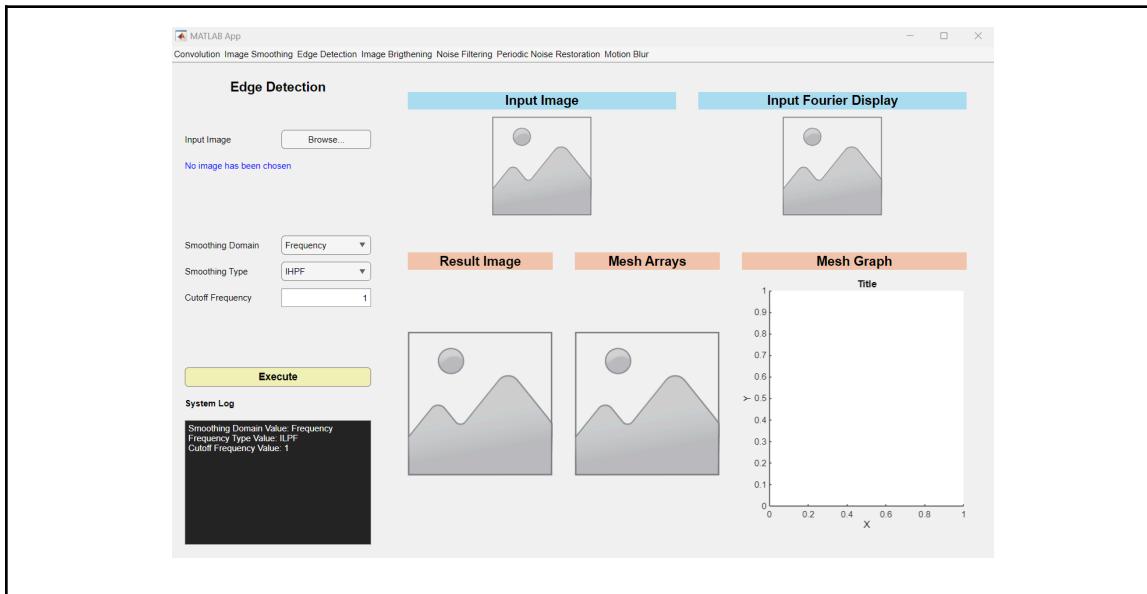
### b. *Image Smoothing* dalam ranah frekuensi dengan memanfaatkan *Low-Pass Filter*



Gambar 2.3 Tampilan GUI dari program untuk melakukan proses *Image Smoothing* dalam ranah frekuensi dengan memanfaatkan *Low-Pass Filter*

Sumber: Dokumen Penulis

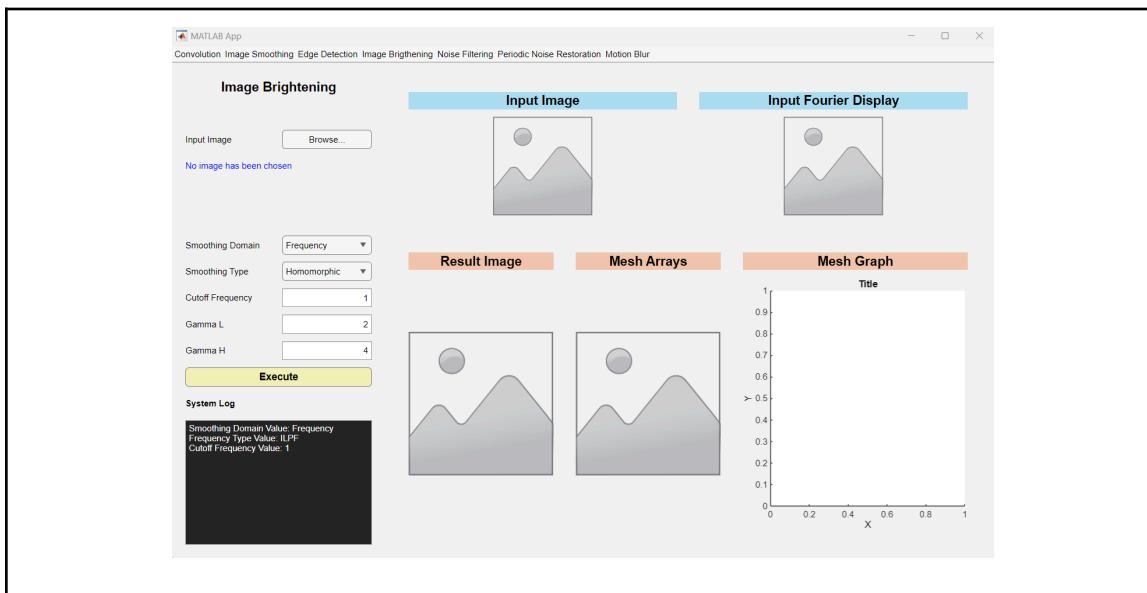
### C. Layar *Edge Detection* dengan memanfaatkan *High-Pass Filter*



Gambar 2.4 Tampilan GUI dari program untuk melakukan proses *Edge Detection* dengan memanfaatkan *High-Pass Filter*

Sumber: Dokumen Penulis

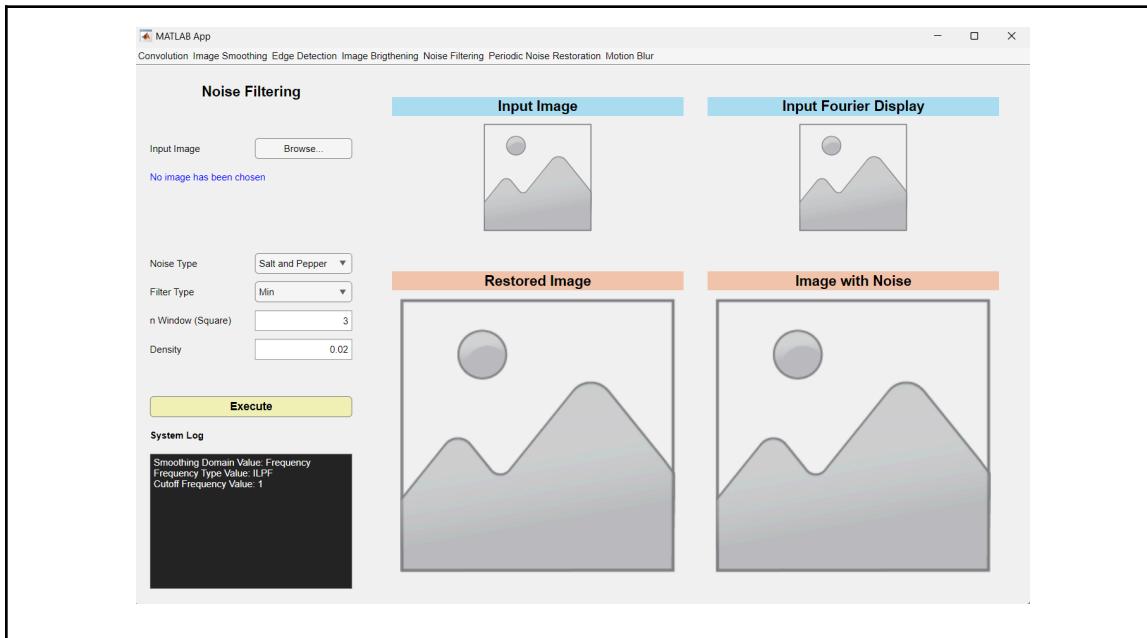
### D. Layar *Image Brightening* dengan memanfaatkan *Homomorphic*



Gambar 2.5 Tampilan GUI dari program untuk melakukan proses *Image Brightening* dengan memanfaatkan *Homomorphic*

Sumber: Dokumen Penulis

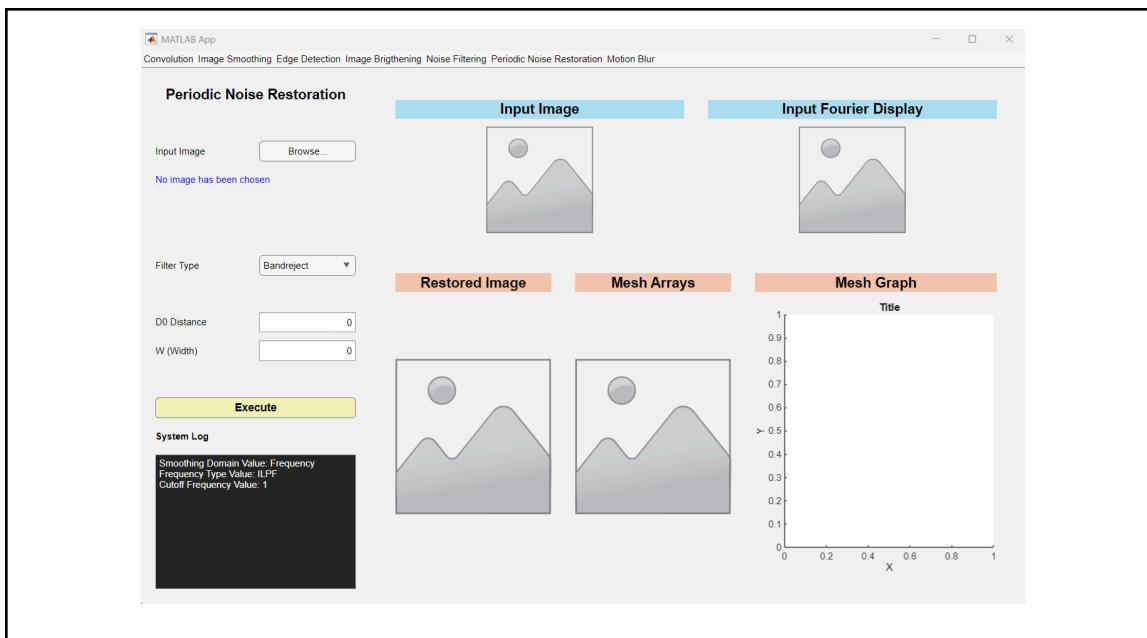
## E. Layar Noise Filtering



Gambar 2.6 Tampilan GUI dari program untuk melakukan proses *Noise Filtering*

Sumber: Dokumen Penulis

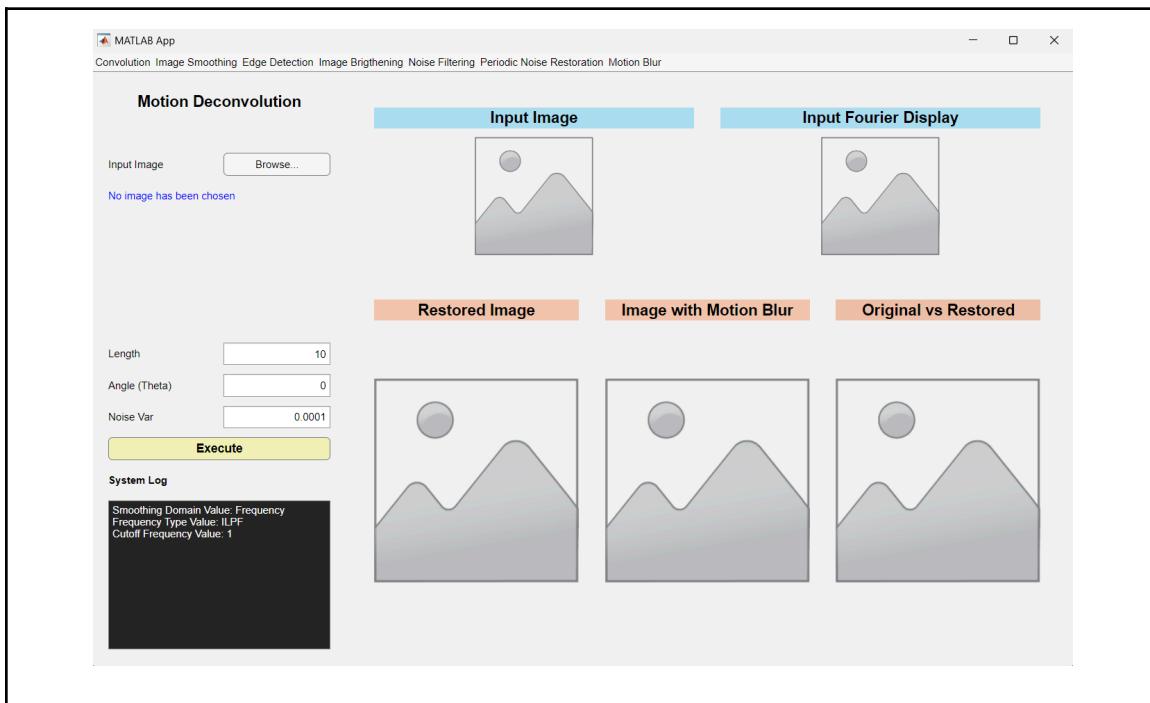
## F. Layar Periodic Noise Restoration



Gambar 2.7 Tampilan GUI dari program untuk melakukan proses *Periodic Noise Restoration*

Sumber: Dokumen Penulis

## G. Layar Wiener Deconvolution



Gambar 2.8 Tampilan GUI dari program untuk melakukan proses dekonvolusi dengan penapis Wiener

Sumber: Dokumen Penulis

## BAB III

### PENJELASAN KODE DAN PENGUJIAN PROGRAM

#### I. Kode Program dan Penjelasan

Untuk pengerjaan kode program, pengerjaan dilakukan dengan membuat berbagai fungsi yang diperlukan untuk menjalankan fungsionalitas program. Berikut merupakan rincian penjabarannya:

##### A. Kode program fungsi utama

###### a. *Convolution*

Fungsi ini digunakan untuk keseluruhan proses konvolusi pada gambar yang bertujuan membantu penapisan citra. Berikut merupakan rincian kodennya:

```
function result_img = convolution(img, mask, n_mask)
    [row, col, num_channels] = size(img);

    pixel_border = floor(n_mask/2);
    sum_mask = sum_mat(mask);

    % Img details
    fprintf("[INFO] An image size [%d, %d] is inputted!\n", row, col);
    result_img = zeros(row, col, num_channels, 'uint8');

    for channel = 1: num_channels
        channel_result = zeros(row, col);
        img_channel = img(:, :, channel); % Each Color channel

        parfor r=1:row
            rowResult = zeros(1, col);

            for c=1:col
                % For border
                % If pixel is in border, set it to black (0)
                if (is_border_pixel(img_channel, r, c, pixel_border))
                    rowResult(c) = 0; % Set to black
                else
                    local_mat = get_local_mat(img_channel, r, c,
n_mask);
                    if(sum_mask == 0)
                        result_pixel = round(dot_product(local_mat,
mask));
                    else
                        result_pixel = round((sum(local_mat.*mask)) / sum_mask);
                    end
                end
            end
            channel_result = channel_result + rowResult;
        end
        result_img(:, :, channel) = channel_result;
    end
end
```

```

        else
            result_pixel = round(dot_product(local_mat,
mask) / sum_mask)
        end
        conv_result = validate_pixel(result_pixel);
        rowResult(c) = conv_result;
    end
end
channel_result(r, :) = rowResult;
result_img(:,:, channel) = channel_result;
end
disp("[FINISHED] Finish processing!");
end

```

b. *image\_smoothing\_spatial*

Fungsi ini digunakan untuk keseluruhan proses penapisan citra pada gambar dengan memanfaatkan ranah spasial antara dengan *Mean* atau *Gaussian Filter*. Berikut merupakan rincian kodennya:

```

function imgMeanFiltered = image_smoothing_spatial(img, spatialType,
nMask, sigma)
    % spatialType = ["Mean", "Gaussian"]
    % Function to apply mean filter smoothing to an image with added
noise

    % Masukkan jumlah nilai untuk pembentukan ukuran matriks mean
    if mod(nMask, 2) == 0
        fprintf("[FAILED] Cannot process even number of n\n");
    else

        % Proses untuk Image Smoothing dengan Spatial Type yang dipilih
        fprintf("[PROCESS] Image Smoothing using " + spatialType + "
Filter\n");

        % Apply Mean filter using convolution
        if spatialType == "Mean"
            mask = generate_mean_matrix(nMask);
        elseif spatialType == "Gaussian"
            mask = generate_gaussian_matrix(nMask, sigma);
        end

        imgMeanFiltered = convolution(double(img), mask, nMask);
    end

```

```
    end  
end
```

c. *image\_smoothing\_frequency*

Fungsi ini digunakan untuk keseluruhan proses penapisan citra pada gambar dengan memanfaatkan ranah frekuensi. Semua jenis penapisan citra dengan ranah frekuensi mulai dari *Low-Pass Filter*, *High-Pass Filter*, hingga *Homomorphic* diproses dengan fungsi ini. Berikut merupakan rincian kodenya:

```
function [image_result, meshArrays] = image_smoothing_frequency(img,  
frequencyType, cutoffFrequency, n, gammaL, gammaH)  
    % frequencyType = ['ILPF', 'GLPF', 'BLPF', 'IHPF', 'GHPF', 'BHPF',  
'Homomorphic']  
    addpath('functions\utils');  
  
    img = im2double(img); % Konversi ke double  
    [M, N, numChannels] = size(img); % Mencari ukuran dan jumlah  
channels dalam gambar  
  
    % Step 1: Mendefinisikan padding parameters (P = 2M and Q = 2N)  
    P = 2 * M;  
    Q = 2 * N;  
  
    % Inisialisasi padded image  
    if numChannels == 1  
        % For grayscale images  
        pad = padarray(img, [M N], 0, 'post');  
    else  
        % Untuk gambar RGB, pad setiap chanel warna secara terpisah  
        pad = zeros(P, Q, 3);  
        for c = 1:3  
            pad(:,:,:,c) = padarray(img(:,:,:,c), [M N], 0, 'post');  
        end  
    end  
  
    % Step 2: Menjalankan fungsi Fourier Transform pada padded image  
    [fourierSpectrum, ~] = fourier_transform(P, Q, numChannels, pad);  
  
    % Step 3: Memproses filter frekuensi berdasarkan kondisi pemilihan  
    meshArrays = filter_processing(frequencyType, P, Q,  
cutoffFrequency, n, gammaL, gammaH);  
  
    % Step 4: Menerapkan filter pada Fourier-transformed image  
    if numChannels == 1
```

```

    % Menerapkan filter pada grayscale image
    convResult = meshArrays .* fourierSpectrum;
else
    % Menerapkan filter pada setiap channel warna dari gambar RGB
    convResult = zeros(P, Q, 3);
    for c = 1:3
        convResult(:,:,:,c) = meshArrays .* fourierSpectrum(:,:,:,:,c);
    end
end

    % Step 5: Menjalankan invers Fourier Transform untuk mendapatkan
filtered image
if numChannels == 1
    fft_result = ifftshift(convResult);
    image_result = real(ifft2(fft_result)); % Inverse FFT untuk
gambar grayscale
else
    fft_result = zeros(P, Q, 3);
    image_result = zeros(P, Q, 3);
    for c = 1:3
        fft_result(:,:,:,c) = ifftshift(convResult(:,:,:,c));
        image_result(:,:,:,c) = real(ifft2(fft_result(:,:,:,c))); % %
Inverse FFT untuk setiap warna channel dari gambar RGB
    end
end

    % Step 6: Crop gambar pada ukuran original
if numChannels == 1
    % Crop for grayscale image
    image_result = image_result(1:M, 1:N);
else
    % Crop each channel of RGB image
    image_result = image_result(1:M, 1:N, :);
end

```

#### d. *noise filtering*

Fungsi ini digunakan untuk melakukan penambahan derau dan penyelesaian derau dalam domain spatial. Penambahan derau dapat bertipe *salt and pepper* dan *gaussian*. Berikut adalah rincian kodennya:

```

function [result_img, img_with_noise] = noise_filtering(img,
noise_type, filter_type, n_window, density, mean, variance)

```

```

if (noise_type == "Salt and Pepper")
    img_with_noise = imnoise(img, 'salt & pepper', density);
else % noise_type == "Gaussian"
    img_with_noise = imnoise(img, 'gaussian', mean, variance);
end

[row, col, num_channels] = size(img);
pixel_border = floor(n_window/2);

% Img details
fprintf("[INFO] An image size [%d, %d] is inputted!\n", row, col);

% Create placeholder for new image
result_img = zeros(row, col, num_channels, 'uint8');

% Create image
for channel = 1: num_channels
    channel_result = zeros(row, col);
    img_channel = img(:, :, channel); % Each Color channel

    parfor r=1:row
        rowResult = zeros(1, col);

        for c=1:col
            % For border
            % If pixel is in border, set it to black (0)
            if (is_border_pixel(img_channel, r, c, pixel_border))
                rowResult(c) = 0; % Set to black
            else
                local_mat = get_local_mat(img_channel, r, c,
n_window);
                conv_result =
validate_pixel(round(noise_filter_process(local_mat, filter_type)));
                rowResult(c) = conv_result;
            end
        end
        channel_result(r, :) = rowResult;
    end
    result_img(:,:, channel) = channel_result;
end
disp("[FINISHED] Finish processing!");

```

e. *blurring\_deconvolution*

Fungsi ini digunakan untuk keseluruhan proses *motion bluring* hingga dekonvolusi gambar pada citra dengan penapis Wiener. Berikut merupakan rincian kodanya:

```
function [image_restored, image_blurred, original_vs_restored] =
blurring_deconvolution(img, len, theta, noise_var)
    % Menjalankan fungsi Motion Blurring
    [blur_processing, image_blurred] = motion.blur(img, len, theta);

    % Menjalankan fungsi untuk Deblurring dengan penapis Weiner
    image_restored = wiener_deconvolution_scratch(img, noise_var,
blur_processing, image_blurred);

    % Konversi image ke dalam bentuk double
    img_converted = im2double(img);
    original_vs_restored = imabsdiff(img_converted, image_restored);
end
```

f. *blurring\_deconvolution*

Fungsi ini digunakan untuk menyelesaikan permasalahan derau dalam domain frekuensi. Berikut adalah rincian kodanya:

```
function [result_img, meshArrays] = periodic_noise_restoration(img,
filter_type, W, D0, D1)

[M, N, n_channel] = size(img);

% Process Fourier
[fourierSpectrum, ~] = fourier_transform(M, N, n_channel, img);

if filter_type == "Bandreject"
    % 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);
n = 1;
H = 1./(1 + ((D.*W)./(D.^2 - D0.^2)).^(2*n));
meshArrays = fftshift(H);

elseif filter_type == "Bandpass"
nx = M;
ny = N;

f = uint8(img);
fftI = fft2(f,2*nx-1,2*ny-1);
fftI = fftshift(fftI);

% 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);

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;
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));
result_img = uint8(filtered_image);
meshArrays = filter3;

else% filter_type == "Notch"

notch_centers = [M/2 + 30, N/2 + 40; M/2 - 30, N/2 - 40];

% Initialize the notch filter with ones
H = ones(M, N);

% Create a notch filter around each noise location
[x, y] = meshgrid(1:N, 1:M);
for i = 1:size(notch_centers, 1)

```

```

    notch_center = notch_centers(i, :);
    D = sqrt((x - notch_center(2)).^2 + (y - notch_center(1)).^2);
    H(D <= D0) = 0; % Set the values within the radius to zero
end
meshArrays = H;
end

if (filter_type == "Bandreject" || filter_type == "Notch")
    if n_channel == 1
        % Menerapkan filter pada grayscale image
        f_transformed = meshArrays .* fourierSpectrum;
    else
        % Menerapkan filter pada setiap channel warna dari gambar RGB
        f_transformed = zeros(M, N, 3);
        for c = 1:3
            f_transformed(:,:,c) = meshArrays .* fourierSpectrum(:,:,:c);
        end
    end

    if n_channel == 1
        fft_result = ifftshift(f_transformed);
        result_img = real(ifft2(fft_result)); % Inverse FFT untuk
        gambar grayscale
    else
        fft_result = zeros(M, N, 3);
        result_img = zeros(M, N, 3);
        for c = 1:3
            fft_result(:,:,c) = ifftshift(f_transformed(:,:,c));
            result_img(:,:,c) = real(ifft2(fft_result(:,:,c))); % Inverse FFT untuk setiap warna channel dari gambar RGB
        end
    end

    result_img = rescale(result_img);
    disp(result_img(1:10, 1:10));
end

```

## B. Kode program fungsi pendukung (*utils*)

### a. *Fourier\_transform*

Fungsi ini digunakan untuk operasi perhitungan untuk transformasi citra dengan memanfaatkan *Fourier Transform*. Fungsi ini bertujuan untuk mengubah fungsi dari ranah spasial ke dalam ranah frekuensi yang selanjutnya akan digunakan untuk pemrosesan *Low-Pass Filter*, *High-Pass Filter*, hingga *Homomorphic*. Berikut merupakan rincian kodennya:

```
function [fourierSpectrum, fourierDisplay] = fourier_transform(P, Q,
numChannels, pad)
    % Menjalankan Fourier Transform pada padded image
    if numChannels == 1
        % Fourier Transform untuk gambar grayscale
        fourierSpectrum = fftshift(fft2(pad));
    else
        % Fourier Transform for setiap channel warna pada gambar
        % RGB
        fourierSpectrum = zeros(P, Q, 3);
        for c = 1:3
            fourierSpectrum(:,:,c) = fftshift(fft2(pad(:,:,:,c)));
        end
    end

    % Perhitungan magnitude spectrum untuk display
    if numChannels == 1
        fourierDisplay = log(1 + abs(fourierSpectrum));
    else
        % Untuk gambar RGB, hitung magnitude untuk setiap channel
        % warna dan rata-ratakan
        fourierDisplay = log(1 + abs(fourierSpectrum(:,:,:,1)) +
abs(fourierSpectrum(:,:,:,2)) + abs(fourierSpectrum(:,:,:,3))) / 3;
    end
```

### b. *filter\_processing*

Fungsi ini digunakan untuk memproses perhitungan pada meshgrid arrays dari penapisan ranah frekuensi berdasarkan tipe filter yang diterima. Hasil dari meshgrid arrays dapat berbeda-beda tergantung dari tipe filter antara *Low-Pass Filter* (*Ideal*, *Gaussian*, *Butterworth*), *High-Pass Filter* (*Ideal*, *Gaussian*, *Butterworth*), hingga *Homomorphic*. Berikut merupakan rincian kodennya:

```

function meshArrays = filter_processing(frequencyType, P, Q,
cutoffFrequency, n, gammaL, gammaH)
    % Designing filter
    u = 0:(P-1);
    idx = find(u > P/2);
    u(idx) = u(idx)-P;
    v = 0:(Q-1);
    idy = find(v > Q/2);
    v(idy) = v(idy)-Q;

    % Buat meshgrid untuk perhitungan jarak dari center
    [V, U] = meshgrid(v, u);
    euclidean = sqrt(U.^2 + V.^2); % Perhitungan jarak dengan Euclidean
distance dari center

    % Set mesh arrays tergantung pada kondisi dari tipe frekuensi
    % =====
    % For image smoothing
    if frequencyType == "ILPF"
        meshArrays = double(euclidean <= cutoffFrequency); % Ideal
Lowpass Filter mask (1 inside cutoff, 0 outside)
    elseif frequencyType == "GLPF"
        meshArrays = exp(-(euclidean.^2) ./ (2 * (cutoffFrequency^2)));
    % Gaussian Lowpass Filter mask
    elseif frequencyType == "BLPF"
        % n = Order of Butterworth filter
        meshArrays = 1 ./ (1 + (euclidean ./ cutoffFrequency).^(2 *
n)); % Butterworth Lowpass Filter mask
    % =====
    % For edge detection
    elseif frequencyType == "IHPF"
        meshArrays = double(euclidean > cutoffFrequency); % Ideal
Highpass Filter mask (1 outside cutoff, 0 inside)
    elseif frequencyType == "GHPF"
        meshArrays = 1 - exp(-(euclidean.^2) ./ (2 *
(cutoffFrequency^2))); % Gaussian Highpass Filter mask
    elseif frequencyType == "BHPF"
        meshArrays = 1 ./ (1 + (cutoffFrequency ./ euclidean).^(2 *
n)); % Butterworth Highpass Filter mask
    % =====
    % For image brightening
    elseif frequencyType == "Homomorphic"
        % n = Order of Butterworth filter
        meshArrays = (gammaH - gammaL) * (1 - exp(-(euclidean.^2) ./ (2 *
(cutoffFrequency^2)))) + gammaL; % Homomorphic Filter mask
    else

```

```

        fprintf("Invalid frequency type\n");
    end

    meshArrays = fftshift(meshArrays); % Shift filter to center the
    frequency components

```

c. *noise\_filter\_processing*

Fungsi ini digunakan untuk memproses perhitungan pada filter pada penyelesaian permasalahan derau domain spatial. Hasil dari fungsi dapat berbeda-beda tergantung dari tipe filter antara *Min*, *Max*, *Median*, *Arithmetic Mean*, *Geometric Mean*, *Harmonic Mean*, *Contraharmonic Mean*, *Midpoint*, dan *Alpha-trimmed Mean*. Berikut merupakan rincian kodennya:

```

function result = noise_filter_process(mat, filter_type)
    len = length(mat) * length(mat);
    % filter_type = ["Min", "Max", "Median", "Arithmetic Mean",
    "Geometric Mean", "Harmonic Mean", "Contraharmonic Mean", "Midpoint",
    "Alpha-trimmed Mean"];
    if(filter_type == "Min")
        result = min(mat(:));
    elseif(filter_type == "Max")
        result= max(mat(:));
    elseif(filter_type == "Median")
        result = median(mat(:));
    elseif(filter_type == "Arithmetic Mean")
        result = mean(mat(:));
    elseif(filter_type == "Geometric Mean")
        result = prod(mat(:))^(1/len);
    elseif(filter_type == "Harmonic Mean")
        mat(mat == 0) = 1; % To prevent devide by 0
        result = len / sum(1 ./ double(mat(:)));
    elseif(filter_type == "Contraharmonic Mean")
        if (sum(mat(:)) == 0)
            result = mean(mat(:)); % To prevent error
        else
            result = sum(mat(:) .^ 2) / sum(mat(:));
        end
    elseif(filter_type == "Midpoint")
        result = 1/2 * ( min(mat(:)) + max(mat(:)) );
    elseif(filter_type == "Alpha-trimmed Mean")
        sorted_mat = sort(mat(:));

```

```

    alpha = 0.2;
    n_delete = round(len * (alpha/2));
    result = mean(sorted_mat(1+n_delete:len-n_delete));
end
end

```

d. *reshape\_mask*

Fungsi ini memiliki tujuan untuk mengubah bentuk *mask* dari input *user* berupa *string* menjadi tipe bentukan matrix. Berikut adalah rincian kodennya:

```

function [mask, n_mask] = reshape_mask(mask_str)
    mask_list = split(mask_str, ' ');
    mask_len = length(mask_list);
    n_mask = round(sqrt(mask_len));

    mask_list = str2double(mask_list);
    mask = reshape(mask_list, n_mask, n_mask);
end

```

## C. Kode program fungsi operasi matriks

a. *display\_mat*

Fungsi ini digunakan untuk menampilkan mask matriks input yang kemudian akan dilakukan proses konvolusi. Berikut merupakan rincian kodennya:

```

function res_string = display_mat(mat)
    res_string = "";
    [row, col] = size(mat);
    for r=1:row
        if (not (r == 1))
            res_string = res_string+ ",";
        end
        for c=1:col
            res_string = res_string + " " + mat(r,c);
        end
    end
end

```

b. *dot\_product*

Fungsi ini digunakan untuk melakukan perkalian secara dot product pada matriks.

Berikut merupakan rincian kodennya:

```
function result = dot_product(mat1, mat2)
    [row1, col1] = size(mat1);
    [row2, col2] = size(mat2);
    mat1 = double(mat1);
    mat2 = double(mat2);
    result = 0;
    if (not (row1 == row2) || not (col1 == col2) )
        disp("Both matrix is not equal in shape!");
        result = [];
    else
        for r=1:row1
            for c=1:col1
                result = result + mat1(r, c) * mat2(r, c);
            end
        end
    end
end
```

c. *get\_local\_mat*

Fungsi ini digunakan untuk mendapatkan matriks nilai warna yang ada pada gambar. Berikut merupakan rincian kodennya:

```
function [local_mat] = get_local_mat(img, row, col, n)
    % n should be an odd number
    pixel_window = floor(n/2);

    local_mat = [];
    for r=row-pixel_window:row+pixel_window
        local_line = img(r, col-pixel_window:col+pixel_window);
        local_mat = [local_mat; local_line];
    end
end
```

d. *is\_border\_pixel*

Fungsi ini digunakan untuk mendapatkan daerah pixel yang menjadi border setelah proses konvolusi. Berikut merupakan rincian kodennya:

```

function result = is_border_pixel(img, r, c, pixel_border)
    [row, col, n_channels] = size(img);
    result = false;
    if (r <= pixel_border || r >= row+1-pixel_border)
        result = true;
    end
    if (c <= pixel_border || c >= col+1-pixel_border)
        result = true;
    end
end

```

e. *sum\_mat*

Fungsi ini digunakan untuk melakukan penjumlahan terhadap semua nilai yang terdapat pada komponen matriks. Berikut merupakan rincian kodennya:

```

function result = sum_mat(mat)
    [row, col] = size(mat);
    result = 0;
    disp(mat);
    for r=1:row
        for c=1:col
            disp(mat(r,c));
            result = result + mat(r,c);
        end
    end
end

```

f. *validate\_pixel*

Fungsi ini digunakan untuk melakukan validasi pada setiap pixel apakah bernilai antara 0-255. Berikut merupakan rincian kodennya:

```

function result = validate_pixel(pixel)
    max_val = 255;
    min_val = 0;
    result = min(pixel, max_val);
    result = max(result, min_val);
end

```

g. *generate\_mean\_matrix*

Fungsi ini digunakan untuk membuat langsung matriks mean yang bernilai 1 / (sum semua nilai pada matriks). Berikut merupakan rincian kodenya:

```
function [mask_mat, n] = generate_mean_matrix(nMask)
    % Set value dari n to nMask
    n = nMask;

    % Membuat nMask x nMask matrix yang terisi dengan angka 1
    mask_mat = ones(nMask);

    % Menghitung jumlah total dari semua elemen dalam matrix (nMask^2)
    total_sum = sum(mask_mat(:));

    % Membagi setiap elemen dari matrix dengan jumlah total untuk
    mendapatkan rata-rata matrix
    mask_mat = mask_mat / total_sum;
end
```

h. *generate\_gaussian\_matrix*

Fungsi ini digunakan untuk membuat langsung matriks dengan kalkulasi Gaussian. Berikut merupakan rincian kodenya:

```
function [mask_mat, n] = generate_gaussian_matrix(nMask, sigma)
    % Generate n x n Gaussian matrix (kernel) dengan standard deviation
    % sigma
    % sigma: standard deviation for the Gaussian function

    % Set value dari n to nMask
    n = nMask;

    % Membuat grid of (x, y) koordinat dengan center di (0, 0)
    half_size = (nMask - 1) / 2;
    [x, y] = meshgrid(-half_size:half_size, -half_size:half_size);

    % Fungsi Gaussian
    mask_mat = (1 / (2 * pi * sigma^2)) * exp(-(x.^2 + y.^2) / (2 *
sigma^2));

    % Normalisasi matrix sehingga total semua elemen menjadi bernilai 1
    mask_mat = mask_mat / sum(mask_mat(:));
end
```

## D. Kode program fungsi penapis Wiener

### a. *Motion blur*

Fungsi ini digunakan untuk proses *motion blurring* untuk membuat gambar input menjadi kabur sebelum dilakukan proses dekonvolusi dengan penapis Wiener. Berikut merupakan rincian kodennya:

```
function [blur_processing, image_blurred] = motion.blur(img, len, theta)
    blur_processing = fspecial('motion', len, theta);
    image_blurred = imfilter(img, blur_processing, 'conv', "circular");
```

### b. *wiener\_deconvolution\_scratch*

Fungsi ini digunakan untuk penerimaan gambar berdasarkan kondisi *grayscale* ataupun RGB untuk kemudian dilakukan proses perhitungan dengan penapis Wiener secara manual dan dikembalikan dalam bentuk gambar yang sudah dilakukan proses dekonvolusi. Berikut merupakan rincian kodennya:

```
function wiener_result = wiener_deconvolution_scratch(img, noise_var,
blur_processing, image_blurred)
    % Cek apakah gambar Grayscale atau RGB
    if size(img, 3) == 3
        % Inisialisasi hasil dengan ukuran sama seperti input RGB
        wiener_result = zeros(size(img));

        % Proses tiap warna terpisah
        for c = 1:3
            img_channel = img(:, :, c);
            image_blurred_channel = image_blurred(:, :, c);

            % Apply Wiener deconvolution untuk setiap warna RGB
            wiener_result(:, :, c) = wiener_calculation(img_channel,
noise_var, blur_processing, image_blurred_channel);
        end
    else
        % Jika gambar grayscale, langsung proses
        wiener_result = wiener_calculation(img, noise_var,
blur_processing, image_blurred);
    end

    % Menjalankan Dekonvolusi dengan library Wiener Filter
    % estimated_nsr = noise_var / var(double(img(:))); % Equivalent to
Sη(u,v) / Sf(u,v)
```

```

    wiener_result_lib = deconvwnr(image_blurred, blur_processing,
noise_var);
    subplot(2, 3, 4); imshow(wiener_result_lib); title("Restored Image
(Library)");

    % Menghitung perbedaan absolut antara gambar original dengan gambar
yang diperbaiki (Library)
    subplot(2, 3, 5); imshow(imabsdiff(img, wiener_result_lib));
title('Original VS Restoring Image (Library')

    % Scale gambar ke dalam bentuk gray 0-1
    wiener_result = mat2gray(wiener_result);
end

```

### c. *wiener\_calculation*

Fungsi ini digunakan untuk proses keseluruhan dari perhitungan untuk dekonvolusi gambar dengan penapis Wiener. Berikut merupakan rincian kodenya:

```

function wiener_result_channel = wiener_calculation(img, noise_var,
blur_processing, image_blurred)
    % Menjalankan Dekonvolusi dengan Wiener Filter secara Scratch
    % Menjalankan Fourier transform untuk image dan blur kernel
    % Fourier Transform Image (G(u,v))
    img_fft = fft2(double(image_blurred), size(img, 1), size(img, 2));

    % Padding ukuran dari gambar blur kernel
    pad_size = [size(img, 1) - size(blur_processing, 1), size(img, 2) -
size(blur_processing, 2)];
    blur_processing_padded = padarray(blur_processing, pad_size,
'post');

    % Menjalankan circular shift pada tengah kernel dalam frekuensi
domain
    blur_processing_padded = circshift(blur_processing_padded,
-floor(size(blur_processing) / 2));

    % Fourier transform dari blur kernel (H(u,v))
    blur_fft = fft2(double(blur_processing_padded), size(img, 1),
size(img, 2));

    % Wiener filter dalam frekuensi domain
    H_conj = conj(blur_fft);
    H_abs_squared = abs(blur_fft).^2;

```

```

% Formula Wiener filter
wiener_filter = H_conj ./ (H_abs_squared + noise_var);

% Menjalankan dekonvolusi dengan Wiener filter
restored_fft = img_fft .* wiener_filter;
wiener_result_padded = real(ifft2(restored_fft, size(img, 1),
size(img, 2)));

% Crop padded areas untuk menyamakan dengan ukuran original gambar
wiener_result_channel = wiener_result_padded(1:size(img, 1),
1:size(img, 2));

% Scale gambar ke dalam bentuk gray 0-1
wiener_result_channel = mat2gray(wiener_result_channel);
end

```

## II. Hasil Pengujian Program

### A. Pengujian program untuk konvolusi citra

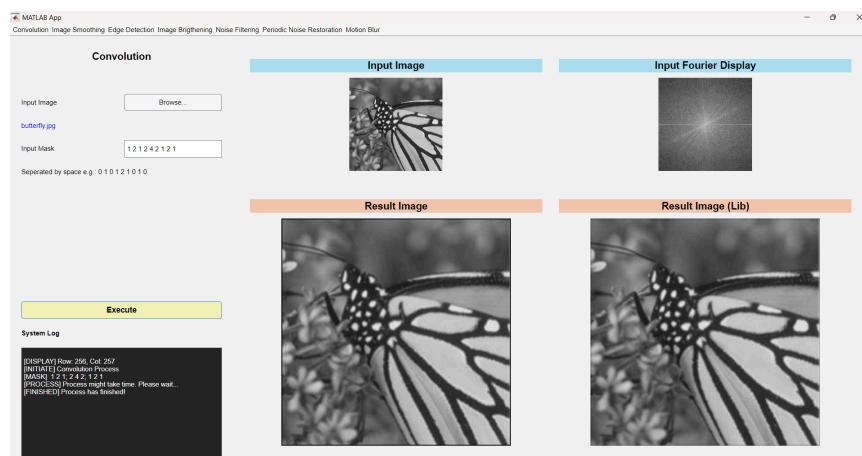
- a. Gambar *grayscale butterfly.jpg*

Matriks Mask:

1 2 1

2 4 2

1 2 1



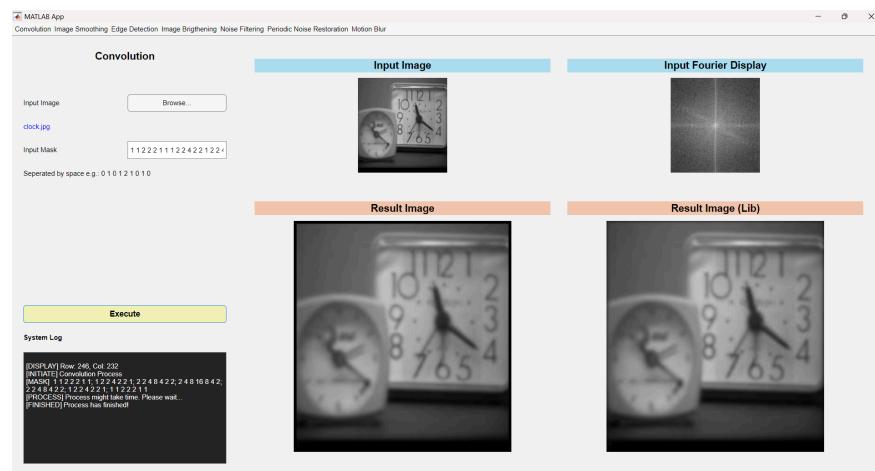
Gambar 3.1 Pengujian konvolusi untuk gambar *grayscale butterfly.jpg*

Sumber: Dokumen Penulis

b. Gambar *grayscale clock.jpg*

Matriks Mask:

```
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
```



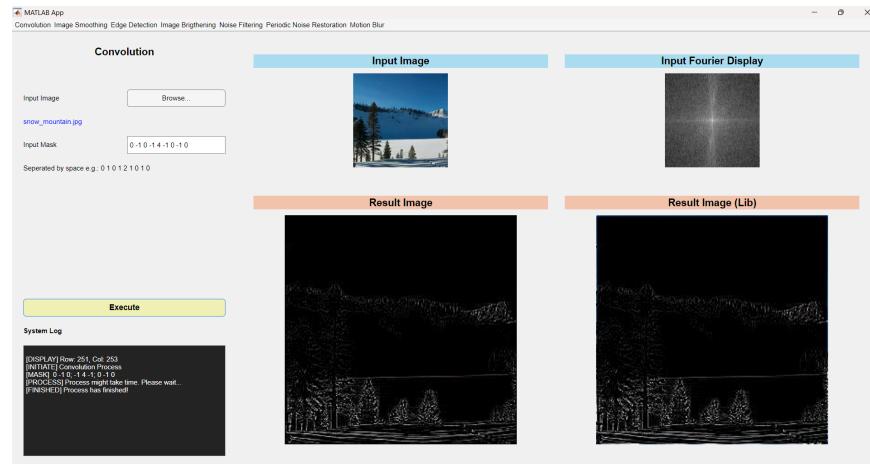
Gambar 3.2 Pengujian konvolusi untuk gambar *grayscale clock.jpg*

Sumber: Dokumen Penulis

c. Gambar berwarna *snow\_mountain.jpg*

Matriks Mask:

```
0 -1 0  
-1 4 -1  
0 -1 0
```



Gambar 3.3 Pengujian konvolusi untuk gambar berwarna *snow\_mountain.jpg*

Sumber: Dokumen Penulis

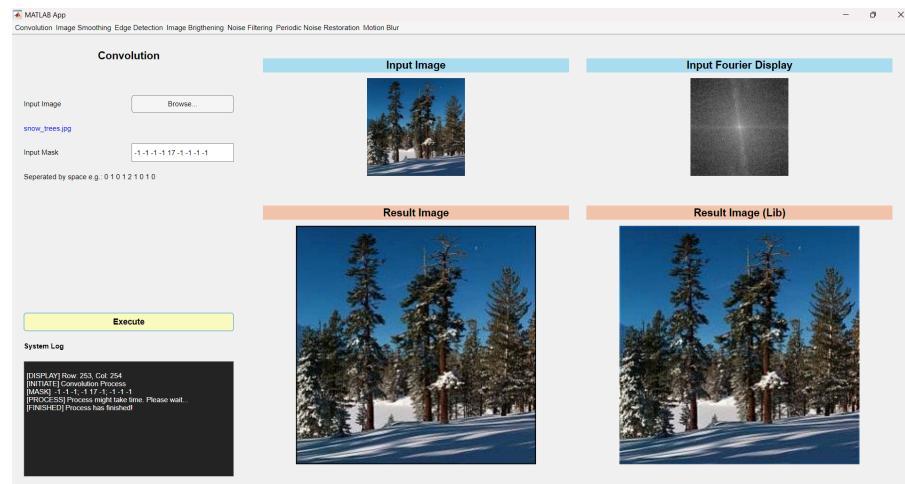
#### d. Gambar berwarna *snow\_trees.jpg*

Matriks Mask:

-1 -1 -1

-1 17 -1

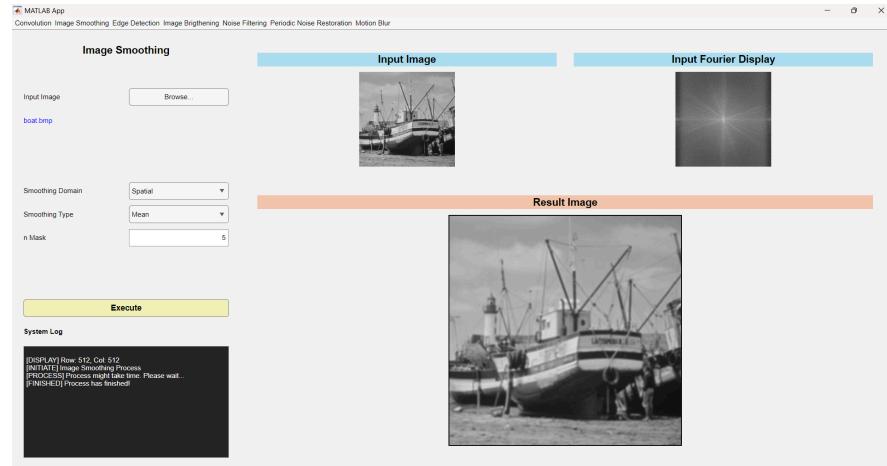
-1 -1 -1



Gambar 3.4 Pengujian konvolusi untuk gambar berwarna *snow\_trees.jpg*

Sumber: Dokumen Penulis

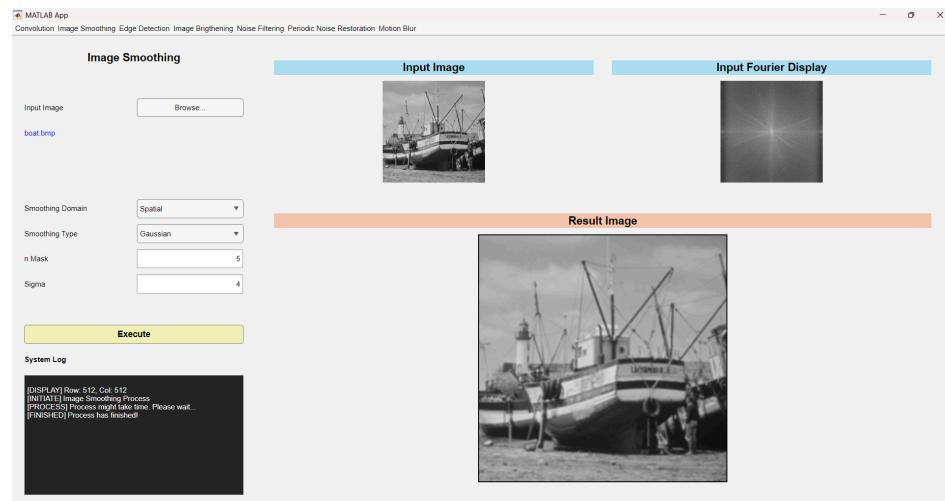
- B. Pengujian program untuk *Image Smoothing* dengan ranah spasial
- Gambar *grayscale boat.bmp* dengan ranah spasial *Mean* (*nMask* = 5)



Gambar 3.5 Pengujian *Image Smoothing* untuk gambar *grayscale boat.bmp*  
dengan ranah spasial *Mean*

Sumber: Dokumen Penulis

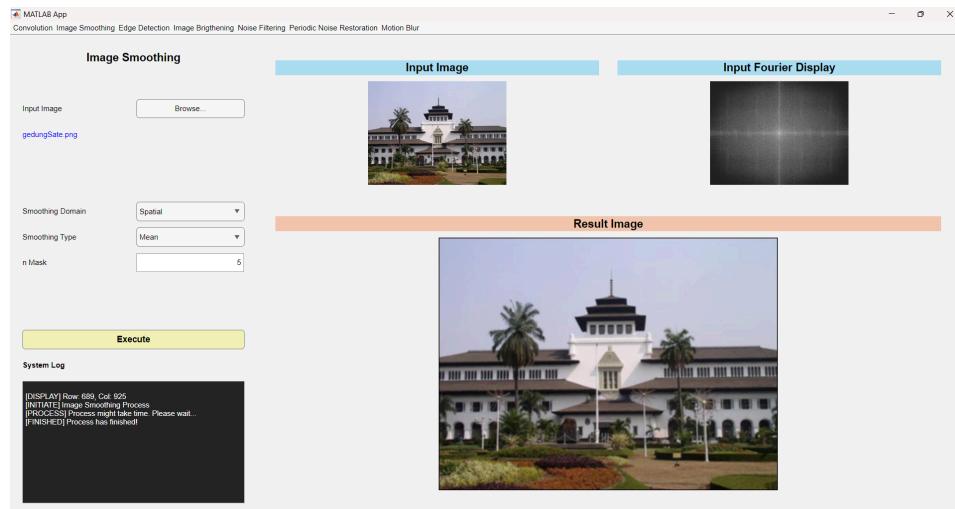
- Gambar *grayscale boat.bmp* dengan ranah spasial *Gaussian* (*nMask* = 5, *sigma* = 4)



Gambar 3.6 Pengujian *Image Smoothing* untuk gambar *grayscale boat.bmp*  
dengan ranah spasial *Gaussian*

Sumber: Dokumen Penulis

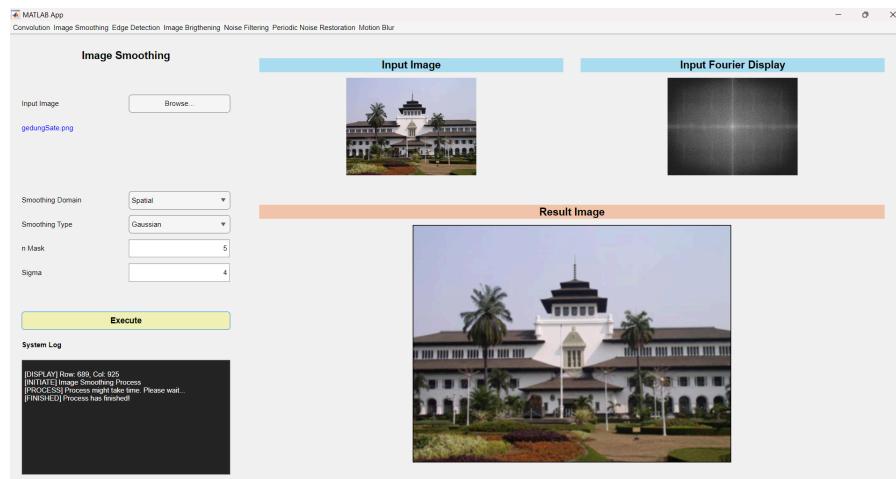
- c. Gambar berwarna *gedungSate.png* dengan ranah spasial *Mean* (*nMask* = 5)



Gambar 3.7 Pengujian *Image Smoothing* untuk gambar berwarna *gedungSate.png*  
dengan ranah spasial *Mean*

Sumber: Dokumen Penulis

- d. Gambar berwarna *gedungSate.png* dengan ranah spasial *Gaussian* (*nMask* = 5, *sigma* = 4)



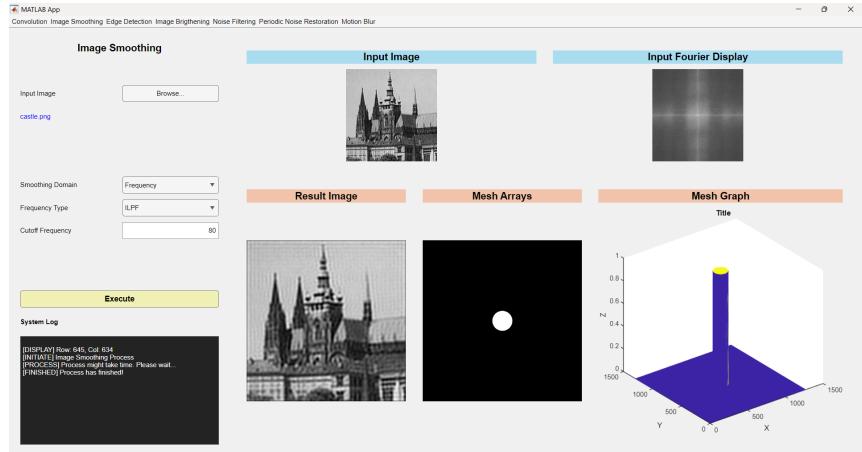
Gambar 3.8 Pengujian *Image Smoothing* untuk gambar berwarna *gedungSate.png*  
dengan ranah spasial *Gaussian*

Sumber: Dokumen Penulis

### C. Pengujian program untuk *Image Smoothing* dengan ranah frekuensi *Low-Pass Filter*

#### a. Pengujian citra *grayscale*

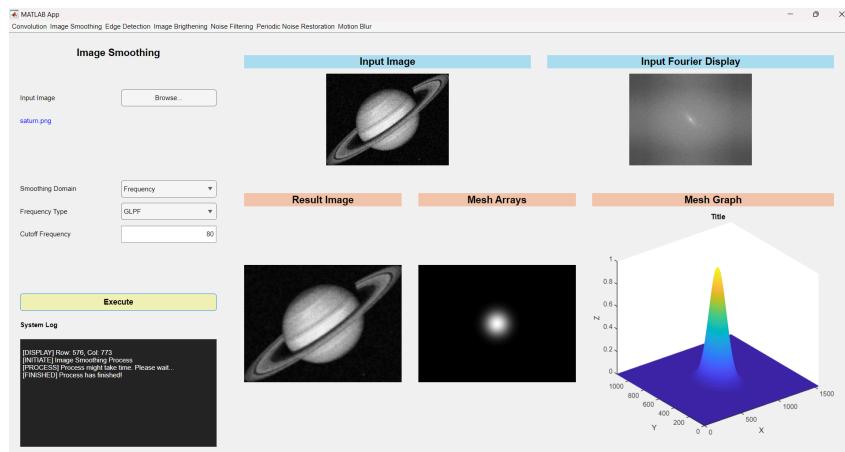
1. Gambar *castle.png* dengan ranah frekuensi *Ideal Low-Pass Filter* (*cutOff* = 80)



Gambar 3.9 Pengujian *Image Smoothing* untuk gambar *grayscale castle.png* dengan ranah frekuensi *Ideal Low-Pass Filter*

Sumber: Dokumen Penulis

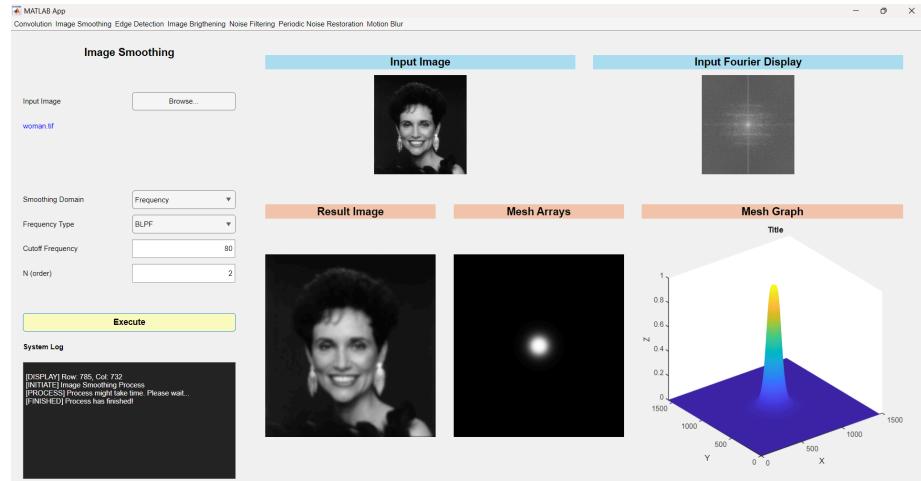
2. Gambar *saturn.png* dengan ranah frekuensi *Gaussian Low-Pass Filter* (*cutOff* = 80)



Gambar 3.10 Pengujian *Image Smoothing* untuk gambar *grayscale saturn.png* dengan ranah frekuensi *Gaussian Low-Pass Filter*

Sumber: Dokumen Penulis

3. Gambar *woman.tif* dengan ranah frekuensi *Butterworth Low-Pass Filter* (cutOff = 80, n = 2)

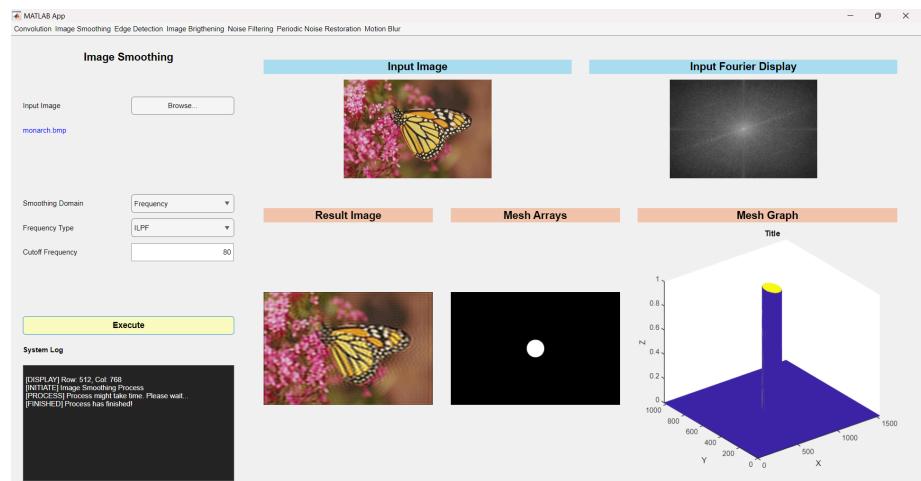


Gambar 3.11 Pengujian *Image Smoothing* untuk gambar *grayscale woman.tif* dengan ranah frekuensi *Butterworth Low-Pass Filter*

Sumber: Dokumen Penulis

## b. Pengujian citra berwarna

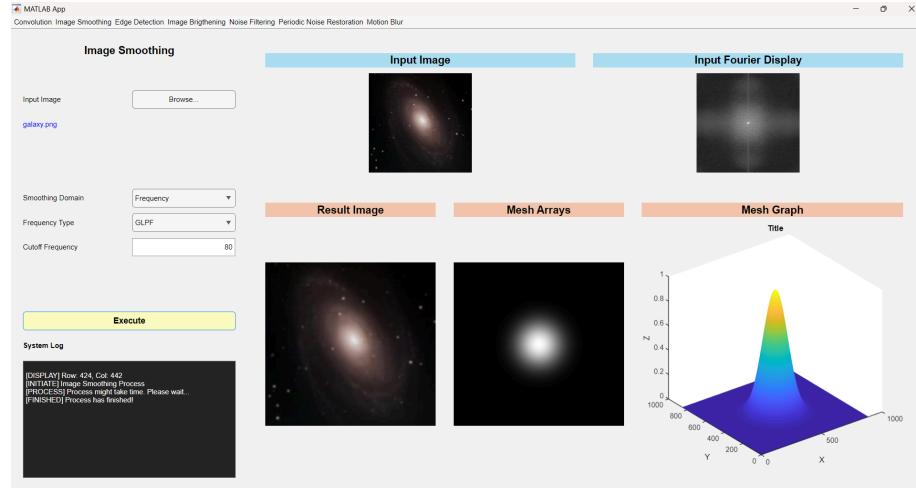
1. Gambar *monarch.bmp* dengan ranah frekuensi *Ideal Low-Pass Filter* (cutOff = 80)



Gambar 3.12 Pengujian *Image Smoothing* untuk gambar berwarna *monarch.bmp* dengan ranah frekuensi *Ideal Low-Pass Filter*

Sumber: Dokumen Penulis

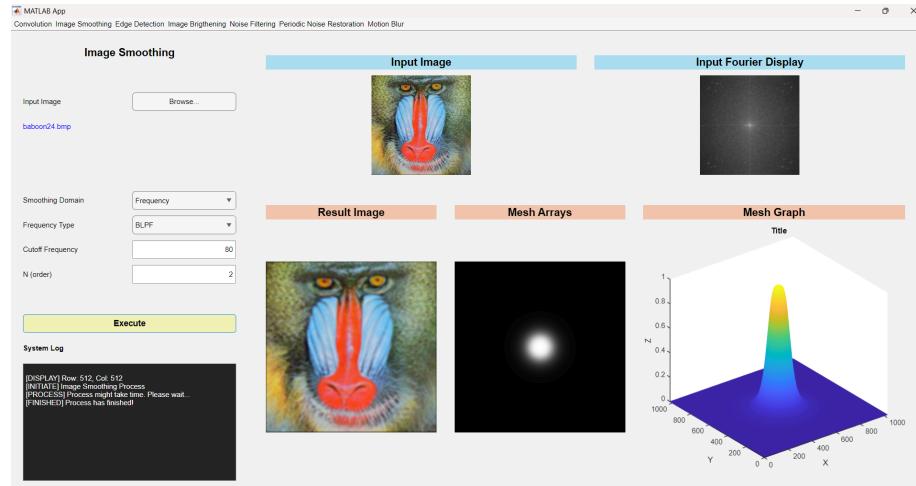
2. Gambar *galaxy.png* dengan ranah frekuensi *Gaussian Low-Pass Filter* (*cutOff = 80*)



Gambar 3.13 Pengujian *Image Smoothing* untuk gambar berwarna *galaxy.png* dengan ranah frekuensi *Gaussian Low-Pass Filter*

Sumber: Dokumen Penulis

3. Gambar *baboon24.bmp* dengan ranah frekuensi *Butterworth Low-Pass Filter* (*cutOff = 80, n = 2*)

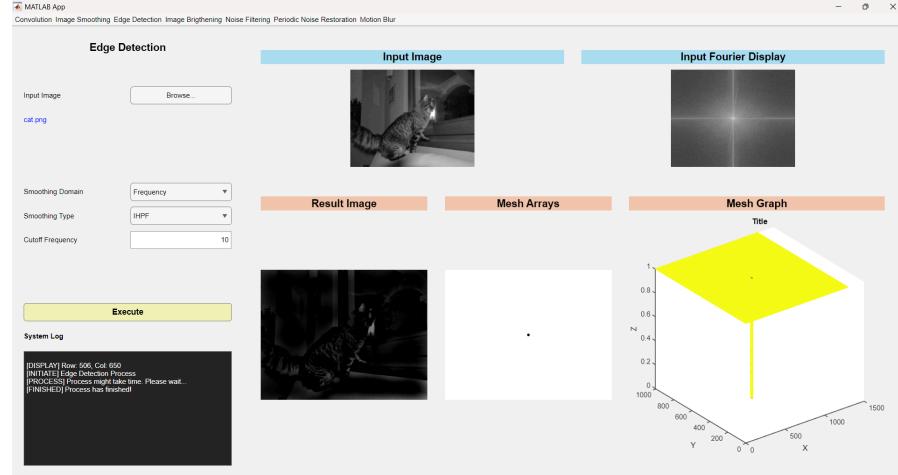


Gambar 3.14 Pengujian *Image Smoothing* untuk gambar berwarna *baboon24.bmp* dengan ranah frekuensi *Butterworth Low-Pass Filter*

Sumber: Dokumen Penulis

D. Pengujian program untuk *Edge Detection* dengan ranah frekuensi *High-Pass Filter*

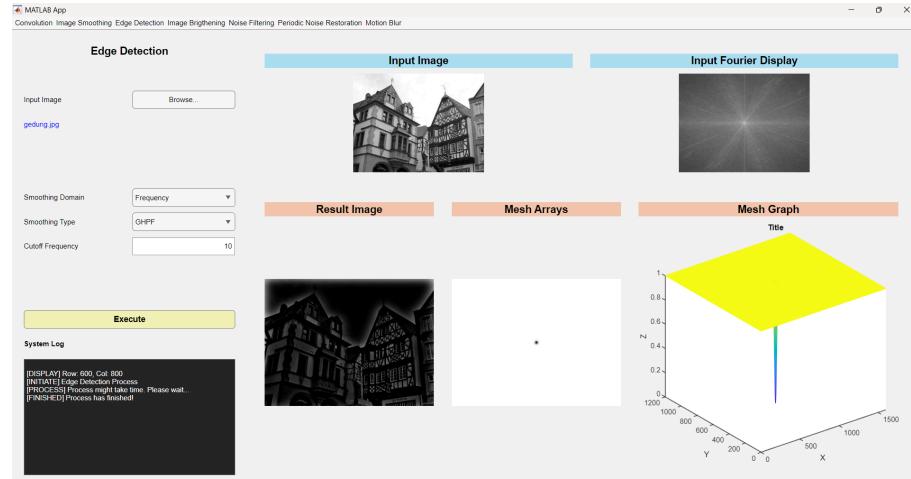
- Gambar *grayscale cat.png* dengan ranah frekuensi *Ideal High-Pass Filter* (*cutOff* = 10)



Gambar 3.15 Pengujian *Edge Detection* untuk gambar *grayscale cat.png* dengan ranah frekuensi *Ideal High-Pass Filter*

Sumber: Dokumen Penulis

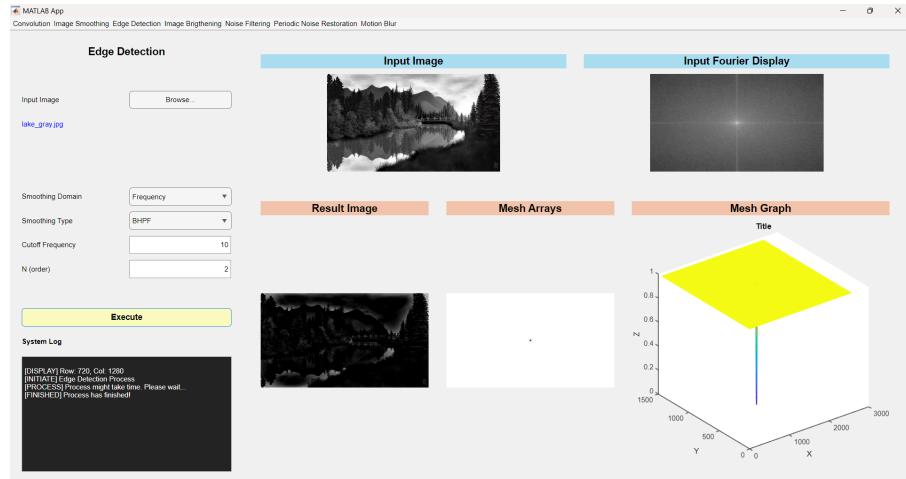
- Gambar *grayscale gedung.bmp* dengan ranah frekuensi *Gaussian High-Pass Filter* (*cutOff* = 10)



Gambar 3.16 Pengujian *Edge Detection* untuk gambar *grayscale gedung.bmp* dengan ranah frekuensi *Gaussian High-Pass Filter*

Sumber: Dokumen Penulis

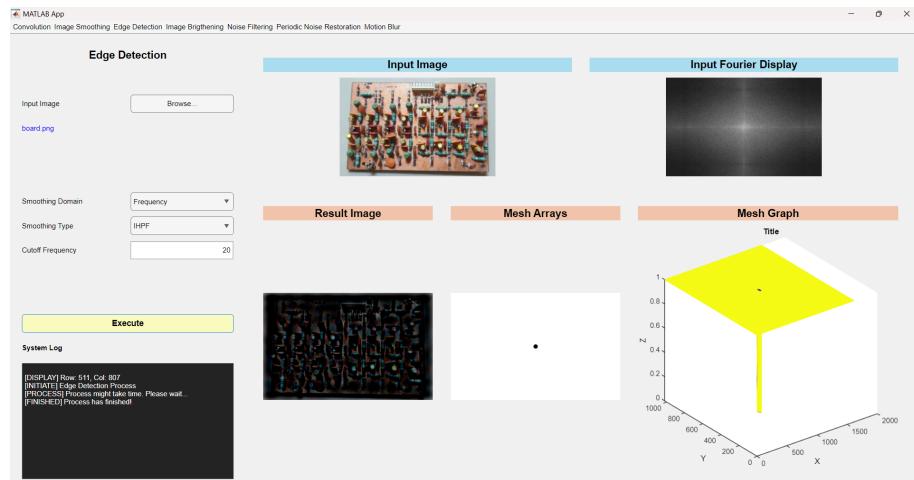
- c. Gambar *grayscale lake\_gray.jpg* dengan ranah frekuensi *Butterworth High-Pass Filter* ( $\text{cutOff} = 10$ ,  $n = 2$ )



Gambar 3.17 Pengujian *Edge Detection* untuk gambar *grayscale lake\_gray.jpg* dengan ranah frekuensi *Butterworth High-Pass Filter*

Sumber: Dokumen Penulis

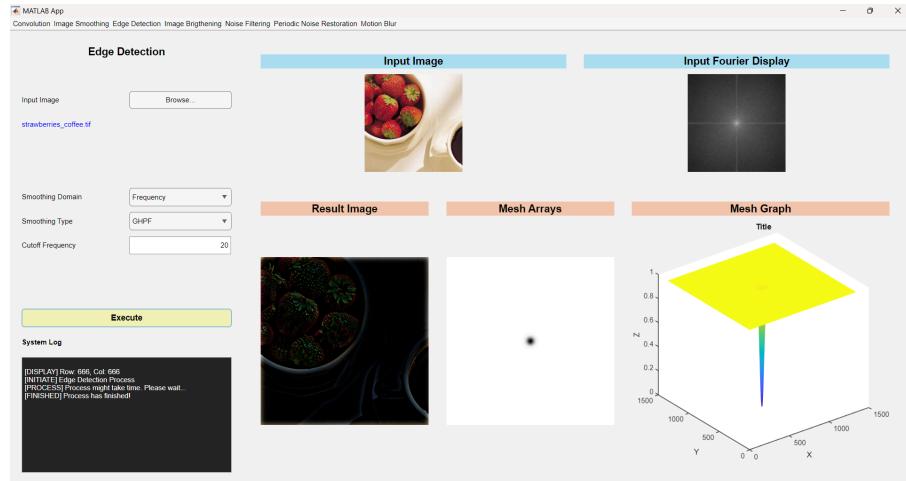
- d. Gambar berwarna *board.png* dengan ranah frekuensi *Ideal High-Pass Filter* ( $\text{cutOff} = 20$ )



Gambar 3.18 Pengujian *Edge Detection* untuk gambar berwarna *board.png* dengan ranah frekuensi *Ideal High-Pass Filter*

Sumber: Dokumen Penulis

- e. Gambar berwarna *strawberries\_coffee.tif* dengan ranah frekuensi *Gaussian High-Pass Filter* (*cutOff* = 20)

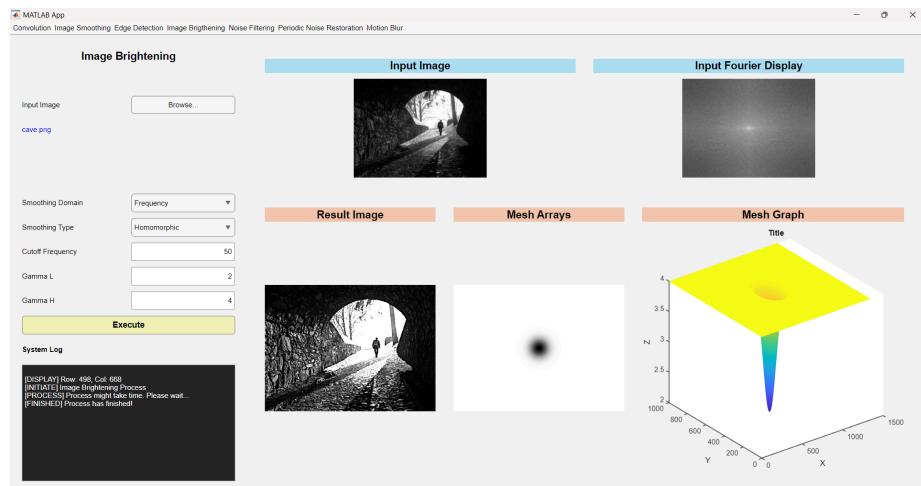


Gambar 3.19 Pengujian *Edge Detection* untuk gambar berwarna *strawberries\_coffee.tif* dengan ranah frekuensi *Gaussian High-Pass Filter*

Sumber: Dokumen Penulis

- E. Pengujian program untuk *Image Brightening* dengan ranah frekuensi *Homomorphic Filter*

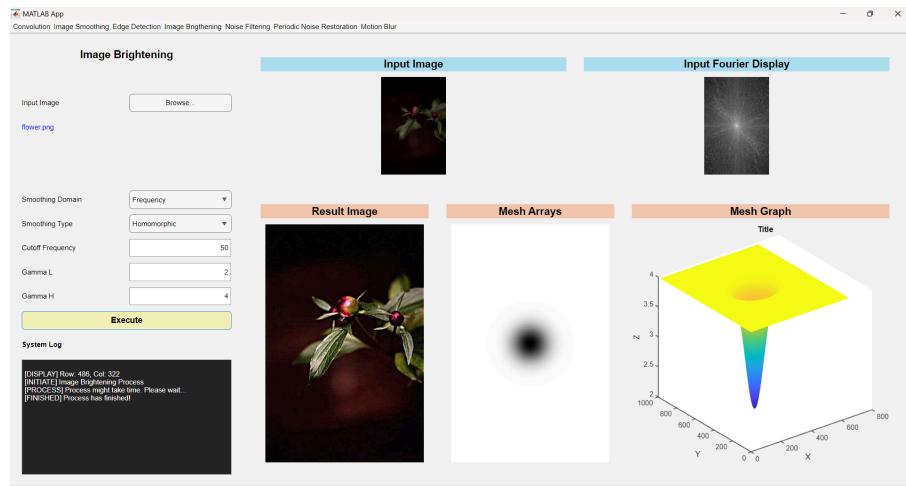
- a. Gambar *grayscale cave.png*



Gambar 3.20 Pengujian *Image Brightening* untuk gambar *grayscale cave.png* dengan ranah frekuensi *Homomorphic Filter*

Sumber: Dokumen Penulis

b. Gambar berwarna *flower.png*

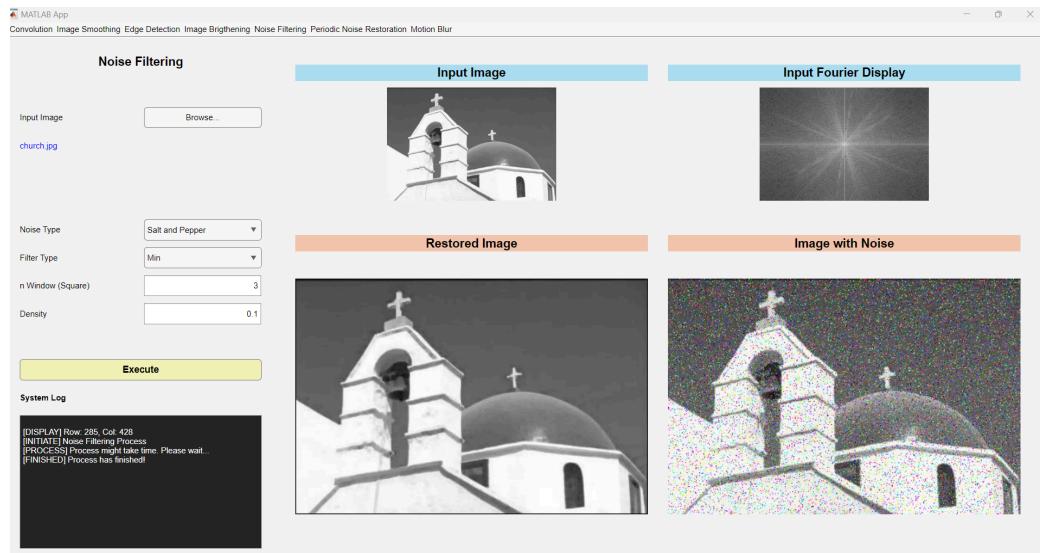


Gambar 3.21 Pengujian *Image Brightening* untuk gambar berwarna *flower.png*  
dengan ranah frekuensi *Homomorphic Filter*

Sumber: Dokumen Penulis

F. Pengujian program untuk *Noise Filtering*

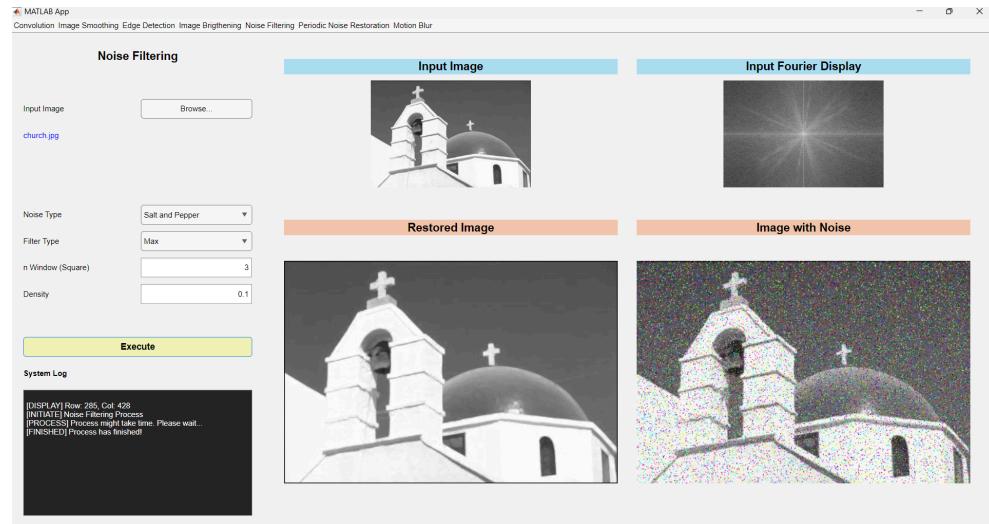
1. Gambar *church.jpg* dengan derau *salt and pepper* dan filter *min*  
( $n = 3$ ,  $density = 0.1$ )



Gambar 3.22 Pengujian *Noise Filtering* *church.jpg* dengan derau *salt and pepper*  
dan filter *min*

Sumber: Dokumen Penulis

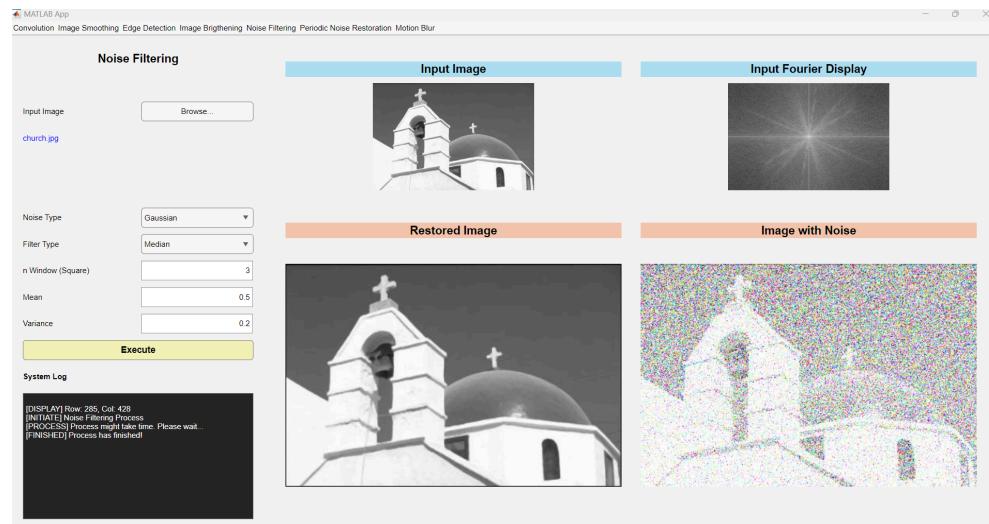
2. Gambar *church.jpg* dengan derau *salt and pepper* dan filter *max*  
 $(n = 3, density = 0.1)$



Gambar 3.23 Pengujian *Noise Filtering* *church.jpg* dengan derau *salt and pepper* dan filter *max*

Sumber: Dokumen Penulis

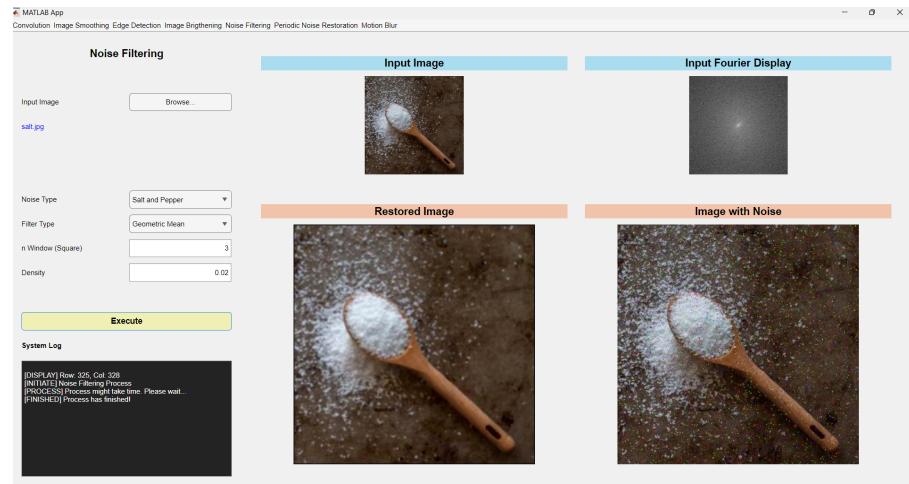
3. Gambar *church.jpg* dengan derau *gaussian* dan filter *median*  
 $(n = 3, mean = 0.5, variance = 0.2)$



Gambar 3.24 Pengujian *Noise Filtering* *church.jpg* dengan derau *gaussian* dan filter *median*

Sumber: Dokumen Penulis

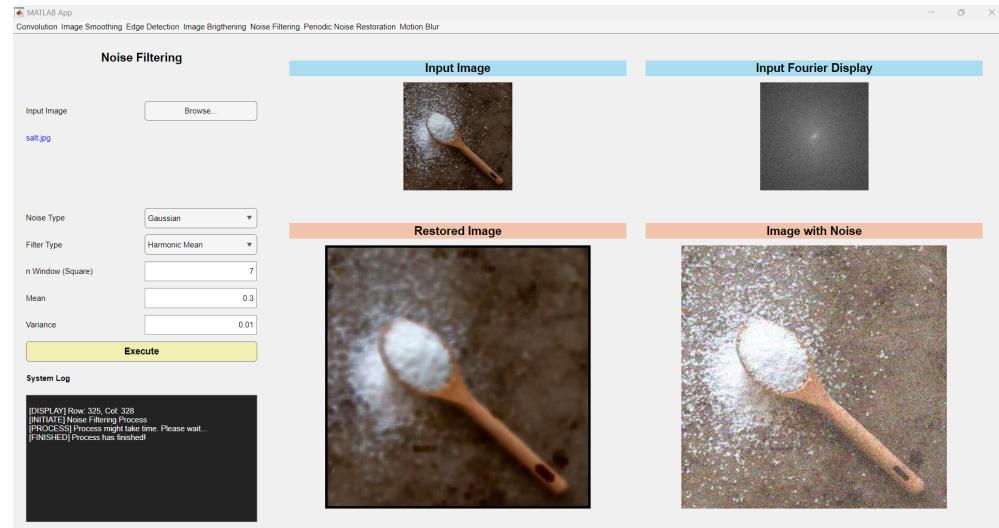
4. Gambar *salt.jpg* dengan derau *salt and pepper* dan filter *geometric mean*  
 $(n = 3, density = 0.02)$



Gambar 3.25 Pengujian *Noise Filtering* *salt.jpg* dengan derau *salt and pepper* dan filter *geometric mean*

Sumber: Dokumen Penulis

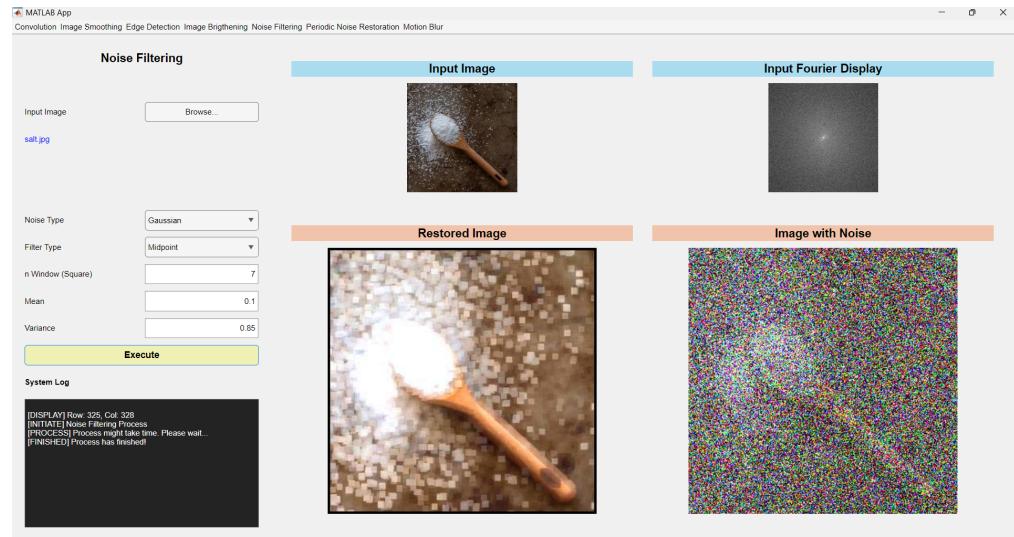
5. Gambar *salt.jpg* dengan derau *gaussian* dan filter *harmonic mean*  
 $(n = 7, mean = 0.3, variance = 0.01)$



Gambar 3.26 Pengujian *Noise Filtering* *salt.jpg* dengan derau *gaussian* dan filter *harmonic mean*

Sumber: Dokumen Penulis

6. Gambar *salt.jpg* dengan derau *gaussian* dan filter *midpoint*  
 $(n = 7, \text{mean} = 0.1, \text{variance} = 0.85)$

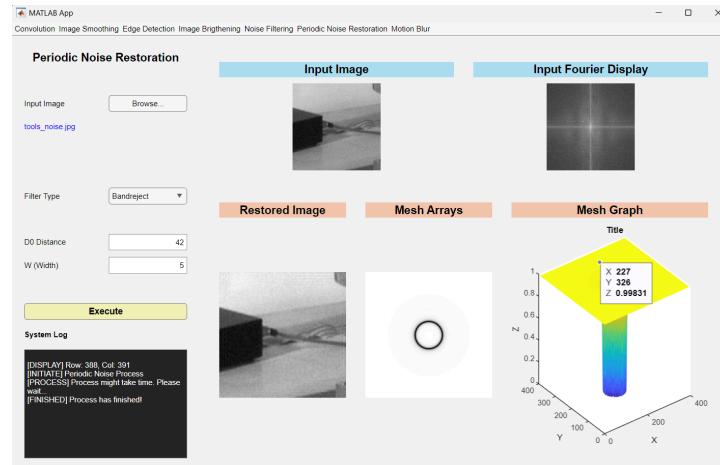


Gambar 3.27 Pengujian *Noise Filtering* *salt.jpg* dengan derau *gaussian* dan filter *midpoint*

Sumber: Dokumen Penulis

## G. Pengujian program untuk *Periodic Noise Restoration*

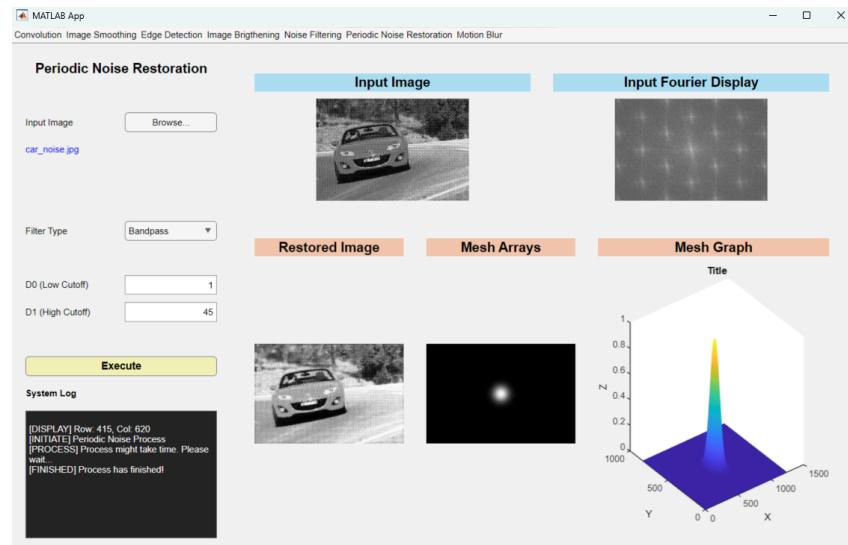
1. Gambar *tools\_noise.jpg* dengan menggunakan filter *band-reject* ( $D_0 = 42$ ,  $W = 5$ )



Gambar 3.28 Pengujian *Periodic Noise Restoration* *tools\_noise.jpg* dengan menggunakan filter *band-reject*

Sumber: Dokumen Penulis

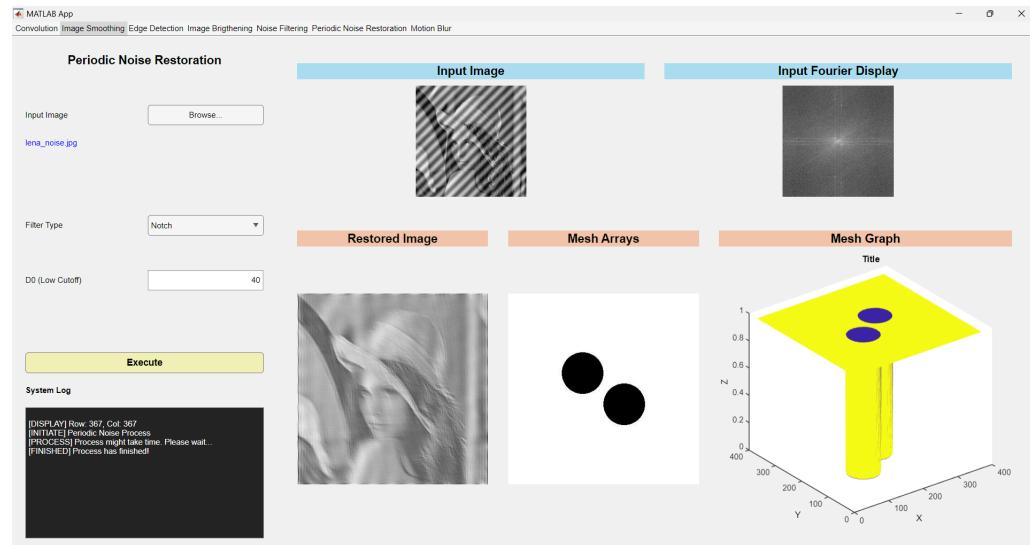
2. Gambar *car\_noise.jpg* dengan menggunakan filter *band-pass* ( $D_0 = 1$ ,  $D_1 = 45$ )



Gambar 3.29 Pengujian *Periodic Noise Restoration car\_noise.jpg* dengan menggunakan filter *band-pass*

Sumber: Dokumen Penulis

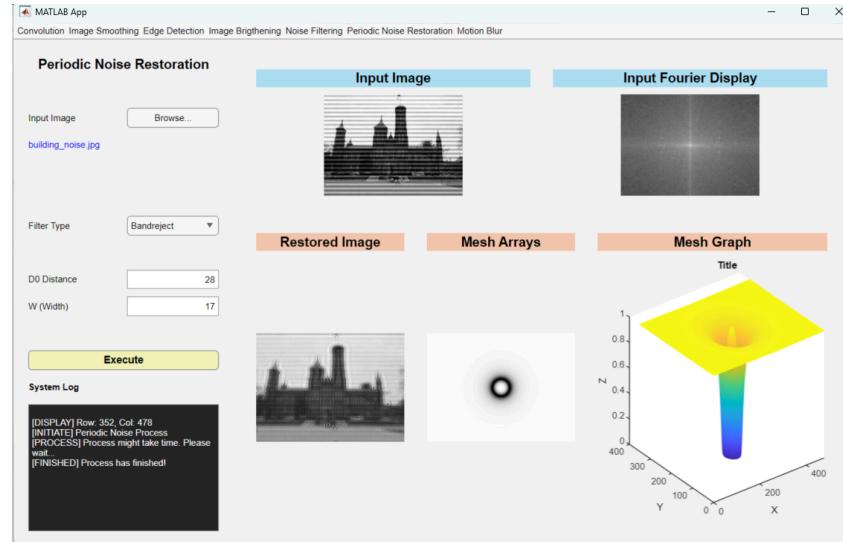
3. Gambar *lena\_noise.jpg* dengan menggunakan filter *notch* ( $D_0 = 40$ )



Gambar 3.30 Pengujian *Periodic Noise Restoration lena\_noise.jpg* dengan menggunakan filter *notch*

Sumber: Dokumen Penulis

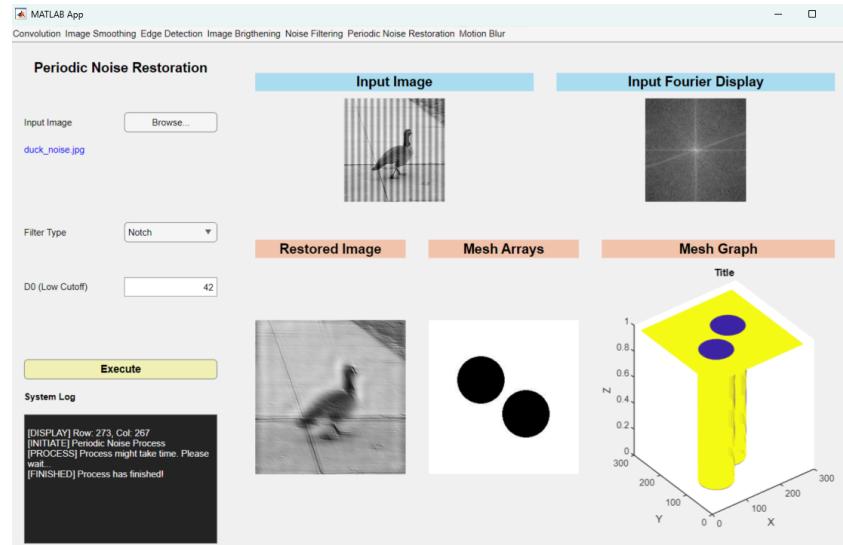
4. Gambar *building\_noise.jpg* dengan menggunakan filter *band-pass* ( $D_0 = 28$ ,  $W = 17$ )



Gambar 3.31 Pengujian *Periodic Noise Restoration building\_noise.jpg* dengan menggunakan filter *band-pass*

Sumber: Dokumen Penulis

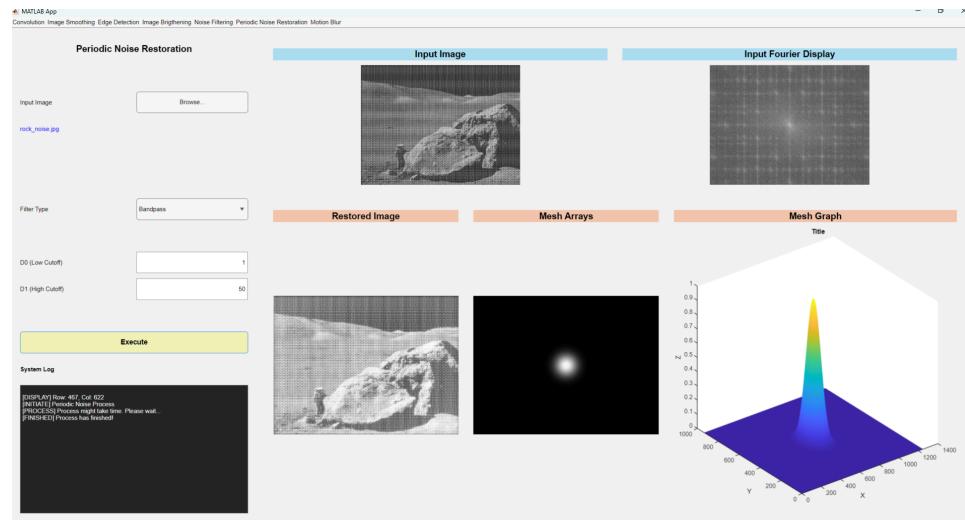
5. Gambar *duck\_noise.jpg* dengan menggunakan filter *Notch* ( $D_0 = 42$ )



Gambar 3.32 Pengujian *Periodic Noise Restoration duck\_noise.jpg* dengan menggunakan filter *Notch*

Sumber: Dokumen Penulis

6. Gambar *rock\_noise.jpg* dengan menggunakan filter *band-pass* ( $D_0 = 1$ ,  $D_1 = 50$ )

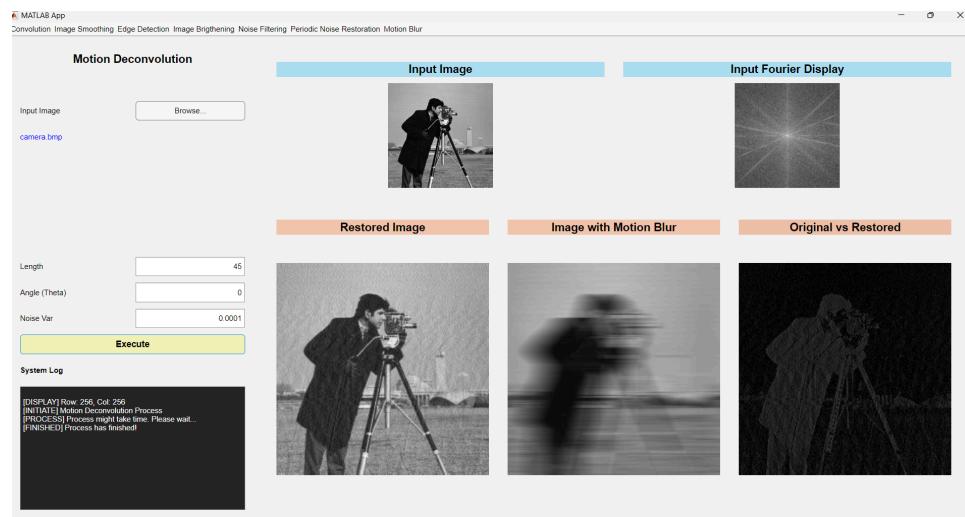


Gambar 3.33 Pengujian *Periodic Noise Restoration* *rock\_noise.jpg* dengan menggunakan filter *band-pass*

Sumber: Dokumen Penulis

#### H. Pengujian program untuk memperbaiki *Motion Blur* dengan Penapis Wiener

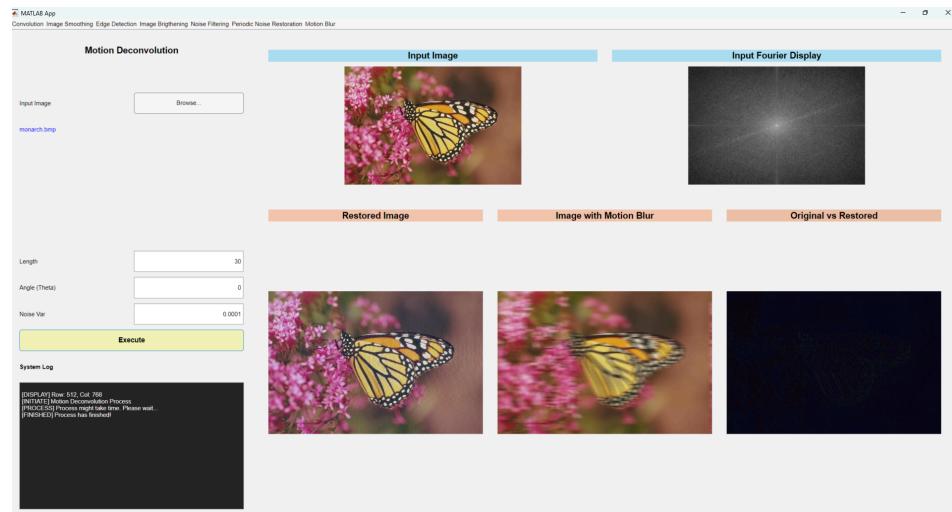
1. Gambar *camera.bmp* ( $length = 45$ ,  $\theta = 0$ ,  $noise var = 0.0001$ )



Gambar 3.34 Pengujian program untuk memperbaiki *Motion Blur* dengan Penapis Wiener untuk gambar *camera.bmp*

Sumber: Dokumen Penulis

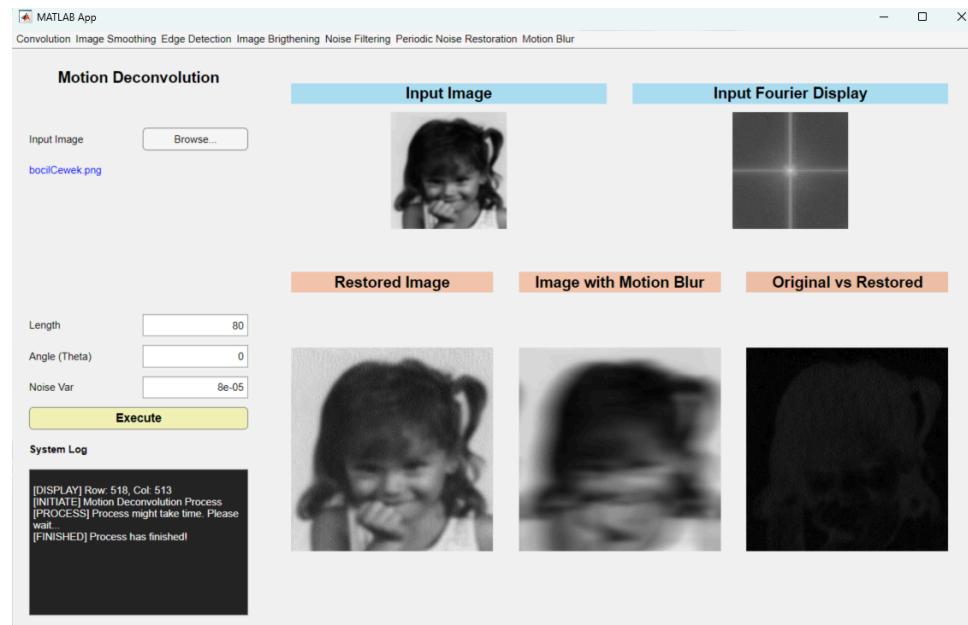
2. Gambar *monarch.bmp* ( $length = 30$ ,  $theta = 0$ ,  $noise var = 0.0001$ )



Gambar 3.35 Pengujian program untuk memperbaiki *Motion Blur* dengan Penapis Wiener untuk gambar *monarch.bmp*

Sumber: Dokumen Penulis

3. Gambar *bocilCewek.png* ( $length = 80$ ,  $theta = 0$ ,  $noise var = 0.00008$ )



Gambar 3.36 Pengujian program untuk memperbaiki *Motion Blur* dengan Penapis Wiener untuk gambar *bocilCewek.png*

Sumber: Dokumen Penulis

### **III. Analisis Cara Kerja Program**

#### A. Analisis cara kerja program untuk proses konvolusi citra

Berikut merupakan langkah-langkah yang dilakukan dalam proses konvolusi citra:

1. Sistem menerima input sebuah citra dari pengguna dan sebuah *mask* yang merupakan matriks  $n \times n$ . *Mask* ini digunakan untuk proses konvolusi kelak. *Mask* dapat memiliki tujuan berbeda-beda tergantung pada nilai-nilai yang dikandung didalamnya.
2. Sistem melakukan analisis pembacaan *mask* sehingga didapatkan nilai *n\_mask*.
3. Sistem melakukan iterasi untuk setiap baris pada citra dan setiap kolom pada setiap baris yang diiterasi.
4. Nilai-nilai pinggir pada matriks diubah menjadi 0 (warna hitam).
5. Nilai matriks yang tidak dianggap pinggir dilakukan konvolusi dengan pembacaan *local matrix* sebesar  $n \times n$ .
6. *Local matrix* dilakukan perhitungan *dot product* dengan *mask* yang ada. Hasil dari *dot product* ini menjadi hasil bagi citra baru yang dihasilkan pada *pixel* tersebut.
7. Setelah semua baris dan kolom selesai diiterasi, sistem mengembalikan citra baru sebagai *output*.

#### B. Analisis cara kerja program untuk proses *Image Smoothing* pada ranah spasial

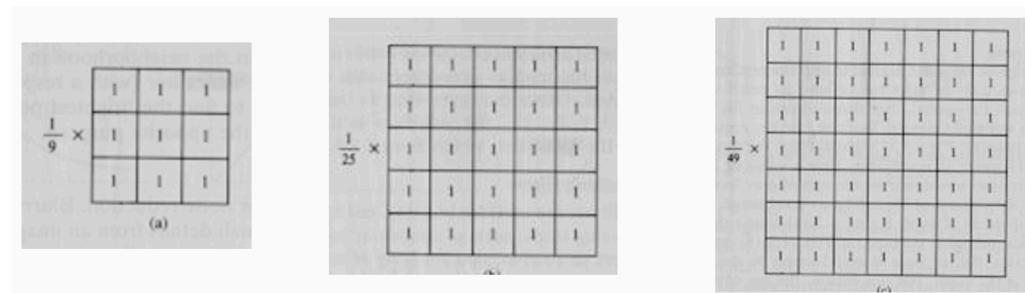
Berikut merupakan langkah-langkah yang dilakukan dalam proses *Image Smoothing* pada ranah spasial:

1. Program menerima input dari pengguna berupa gambar, tipe spasial antara *Mean* atau *Gaussian Filtering*, banyaknya iterasi mask, dan juga menerima nilai sigma khusus untuk *Gaussian Filtering*. Terdapat pengecekan juga bahwa untuk nilai iterasi mask harus bernilai ganjil (1,3,5, dst) dan akan error jika bernilai genap.
2. Setelah menerima berbagai parameter tersebut, program akan mengecek input tipe spasial untuk pembentukan matriks. Terdapat dua kondisi dari pembentukan matriks yaitu sebagai berikut:
  - a. Matriks dari *Mean Filtering*

Proses pembentukan matriks yang ada pada *Mean Filtering* adalah sebagai berikut:

1. Pembentukan ukuran matriks berdasar pada input nMask diawal. Misalkan input sebesar tiga, maka matriks yang terbentuk akan berukuran 3x3 dengan seluruh nilai matriks adalah 1
2. Perhitungan total jumlah semua elemen yang ada pada matriks, misal jika matriks 3x3, maka total nilainya adalah 1x3x3 yaitu 9, jika matriks 5x5, maka total nilainya adalah 1x5x5 yaitu 25
3. Setelahnya, akan dilakukan pembagian terhadap setiap nilai dalam matriks dengan perhitungan total jumlah semua elemen

Untuk contoh dari hasil akhir matriks yang terbentuk adalah sebagai berikut:



Gambar 3.37 Matriks *Mean* yang terbentuk dari masukan nMask 3, 5, dan 7

Sumber:

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/10-Image-Enhancement-Bagian3-2024.pdf>

#### b. Matriks dari *Gaussian Filtering*

Proses pembentukan matriks yang ada pada *Gaussian Filtering* adalah sebagai berikut:

1. Pembentukan ukuran matriks berdasar pada input nMask diawal. Misalkan input sebesar tiga, maka matriks yang terbentuk akan berukuran 3x3
2. Melakukan pembentukan nilai pada matriks Gaussian dengan menentukan titik tengah sebagai koordinat (0,0) dan nilai ditentukan dengan menggunakan rumus Gaussian sebagai berikut

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

- Setelahnya, akan dilakukan pembagian terhadap setiap nilai dalam matriks dengan perhitungan total jumlah semua elemen

Untuk contoh dari hasil akhir matriks yang terbentuk adalah sebagai berikut:

$3 \times 3$ Gaussian mask	$7 \times 7$ Gaussian mask																																																										
$\frac{1}{16} \times$ <table border="1" style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <tr> <td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;">1</td></tr> <tr> <td style="text-align: center;">2</td><td style="text-align: center;">4</td><td style="text-align: center;">2</td></tr> <tr> <td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;">1</td></tr> </table>	1	2	1	2	4	2	1	2	1	$\frac{1}{140} \times$ <table border="1" style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <tr> <td style="text-align: center;">1</td><td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;">2</td><td style="text-align: center;">2</td><td style="text-align: center;">1</td><td style="text-align: center;">1</td></tr> <tr> <td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;">2</td><td style="text-align: center;">4</td><td style="text-align: center;">2</td><td style="text-align: center;">2</td><td style="text-align: center;">1</td></tr> <tr> <td style="text-align: center;">2</td><td style="text-align: center;">2</td><td style="text-align: center;">4</td><td style="text-align: center;">8</td><td style="text-align: center;">4</td><td style="text-align: center;">2</td><td style="text-align: center;">2</td></tr> <tr> <td style="text-align: center;">2</td><td style="text-align: center;">4</td><td style="text-align: center;">8</td><td style="text-align: center;">16</td><td style="text-align: center;">8</td><td style="text-align: center;">4</td><td style="text-align: center;">2</td></tr> <tr> <td style="text-align: center;">2</td><td style="text-align: center;">2</td><td style="text-align: center;">4</td><td style="text-align: center;">8</td><td style="text-align: center;">4</td><td style="text-align: center;">2</td><td style="text-align: center;">2</td></tr> <tr> <td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;">2</td><td style="text-align: center;">4</td><td style="text-align: center;">2</td><td style="text-align: center;">2</td><td style="text-align: center;">1</td></tr> <tr> <td style="text-align: center;">1</td><td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;">2</td><td style="text-align: center;">2</td><td style="text-align: center;">1</td><td style="text-align: center;">1</td></tr> </table>	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
1	2	1																																																									
2	4	2																																																									
1	2	1																																																									
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																																																					

Gambar 3.38 Matriks *Gaussian* yang terbentuk dari masukan nMask 3 dan 7

Sumber:

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/10-Image-Enhancement-Bagian3-2024.pdf>

- Setelah matriks mask sudah ada, kemudian program memanggil fungsi konvolusi dari poin A dengan matriks tersebut untuk mendapatkan hasil citra setelah melalui proses konvolusi
- C. Analisis cara kerja program untuk proses *Image Smoothing* pada ranah frekuensi dengan *Low-Pass Filter*, *High-Pass Filter*, dan penapisan untuk membuat citra lebih terang

Berikut merupakan langkah-langkah yang dilakukan dalam proses *Image Smoothing* pada ranah frekuensi:

- Program menerima sejumlah input wajib dari pengguna yaitu gambar, jenis frekuensi dan nilai cutoff. Ada juga beberapa parameter yang bersifat opsional seperti n yang digunakan untuk *Butterworth Filter* serta gammaL dan gammaH yang digunakan untuk penerangan citra dengan memanfaatkan *Homomorphic Filter*

2. Setelah menerima input, akan dilakukan pendefinisian terhadap padding parameters yang pada program dibuat *default* untuk P bernilai  $2*M$  dan Q bernilai  $2*N$  dengan M dan N merupakan banyaknya pixel yang terdapat pada gambar. Dari padding parameters ini kemudian dilakukan pembentukan citra padding dengan menambahkan pixel-pixel bernilai nol pada  $f(x, y)$ .
3. Berikutnya kemudian menjalankan fungsi Fourier Transform pada padded image. Dalam program, akan dipanggil fungsi `fourier_transform` untuk menjalankan transformasinya. Untuk detail pada fungsi `fourier_transform` sendiri adalah sebagai berikut:
  - a. Proses transformasi akan dibedakan berdasarkan kondisi warna pada gambar apakah *grayscale* ataupun RGB berwarna
  - b. Untuk gambar *grayscale*, prosesnya yaitu melakukan transformasi dengan kode berikut `fftshift(fft2(pad))` dan setelahnya dilakukan perhitungan magnitude spectrum untuk menampilkan kondisi dari Transformasi Fourier dengan rumus

$$fourierDisplay = \log(1 + abs(fourierSpectrum))$$

- c. Untuk gambar RGB berwarna, prosesnya kurang lebih sama seperti dengan *grayscale*, hanya saja untuk transformasi dan perhitungan magnitude dilakukan untuk ketiga jenis warna yaitu *Red*, *Green*, dan *Blue*
4. Setelah melakukan transformasi Fourier, kemudian memanggil fungsi penapis yaitu `filter_processing` berdasarkan kondisi frekuensi yang dipilih. Untuk detail pada fungsi `filter_processing` sendiri adalah sebagai berikut:
  - a. Designing filter dan membuat meshgrid untuk perhitungan jarak dari center dengan menfaatkan rumus *Euclidean*
  - b. Melakukan set mesh tergantung pada tipe frekuensi masukan. Pada program, terdapat tujuh kondisi untuk tipe frekuensi antara lain sebagai berikut:  
Keterangan: D = euclidean, D0 = cutoffFrequency
    - i. *Ideal Low-Pass Filter*

Rumus dari penerapan filter ini adalah sebagai berikut:

$$mesh = double(euclidean <= cutoffFrequency)$$

ii. *Gaussian Low-Pass Filter*

Rumus dari penerapan filter ini adalah sebagai berikut:

$$mesh = e^{(-\frac{euclidean^2}{2 * cutoffFrequency^2})}$$

iii. *Butterworth Low-Pass Filter*

Rumus dari penerapan filter ini adalah sebagai berikut:

$$mesh = \frac{1}{(1 + (\frac{euclidean}{cutoffFrequency})^{2n})}$$

iv. *Ideal High-Pass Filter*

Rumus dari penerapan filter ini adalah sebagai berikut:

$$mesh = double(euclidean > cutoffFrequency)$$

v. *Gaussian High-Pass Filter*

Rumus dari penerapan filter ini adalah sebagai berikut:

$$mesh = 1 - e^{(-\frac{euclidean^2}{2 * cutoffFrequency^2})}$$

vi. *Butterworth High-Pass Filter*

Rumus dari penerapan filter ini adalah sebagai berikut:

$$mesh = \frac{1}{(1 + (\frac{cutoffFrequency}{euclidean})^{2n})}$$

vii. *Homomorphic Filter*

Rumus dari penerapan filter ini adalah sebagai berikut:

$$mesh = (gammaH - gammaL) * mesh GHPF + gammaL$$

5. Selanjutnya yaitu mencari nilai konvolusi dengan ranah frekuensi menggunakan rumus berikut `convResult = meshArrays .* fourierSpectrum` dan perkalian dilakukan berdasarkan kondisi gambar, jika `grayscale` akan dilakukan perkalian untuk satu matriks saja, tetapi jika gambar adalah RGB berwarna, iterasi perkalian akan dilakukan untuk setiap jenis warna

6. Mengambil bagian real dengan menjalankan inverse Fourier Transform untuk mendapatkan filtered image. Kondisi juga sama bahwa inverse untuk gambar *grayscale* akan dilakukan satu iterasi saja dan untuk gambar RGB berwarna akan dilakukan iterasi untuk setiap jenis warna
7. Memotong gambar pada bagian atas untuk mengembalikan pada semula

D. Analisis cara kerja program untuk proses penghilangan derau dengan *Noise Filtering*

Berikut merupakan langkah-langkah yang dilakukan dalam proses *noise filtering* pada citra:

1. Sistem menerima sebuah input berupa citra dari pengguna. Selain citra, sistem juga menerima tipe derau, yang dapat bernilai *salt and pepper* dan *gaussian*, dan tipe filter, yang merupakan algoritma dalam penyelesaian derau yang diberikan.
2. Selanjutnya, sistem memberikan derau/ *noise* ke dalam citra berdasarkan jenis tipe derau yang dimasukkan oleh pengguna. Pengguna juga dapat melakukan konfigurasi terkait *density*, nilai *mean*, dan nilai *variance* yang dapat mempengaruhi hasil derau.
3. Setelah dihasilkan citra dengan derau, sistem akan melakukan penyelesaian untuk menghilangkan derau yang telah ditambahkan kedalam citra tersebut.
4. Sistem melakukan iterasi untuk setiap baris dan setiap kolom untuk setiap barisnya.
5. Untuk *pixel* yang bukan merupakan *pixel* pinggir pada citra, dilakukan pencarian *local matrix*.
6. Sistem mengganti nilai *pixel* tersebut dengan hasil perhitungan dari *local matrix*. Hasil perhitungan ini bergantung pada tipe filter yang dimasukkan oleh pengguna. Filter-filter tersebut antara lain:
  - a. *Min*, mencari nilai minimum,
  - b. *Max*, mencari nilai maksimum,
  - c. *Median*, mencari nilai tengah,
  - d. *Arithmetic mean*, mencari nilai rerata aritmatika,
  - e. *Geometric mean*, mencari nilai rerata geometri,
  - f. *Harmonic mean*, mencari nilai rerata harmonik,
  - g. *Contraharmonic mean*, mencari nilai rerata kontra harmonik,

- h. *Midpoint*, mencari nilai rerata dari nilai *min* dan *max*, dan
    - i. *Alpha-trimmed mean*, mencari nilai rerata dengan membuang nilai ujung pada *local matrix*.
  - 7. Setelah seluruh *pixel* dikalkulasikan dan sistem telah mengiterasi seluruh *pixel* yang ada, sistem akan mengembalikan citra baru yang dihasilkan sebagai *output*.
- E. Analisis cara kerja program untuk proses penghilangan derau dengan *Periodic Noise Restoration*

Serupa dengan poin (D), fungsi ini juga berfungsi untuk menyelesaikan permasalahan akibat derau. Akan tetapi, metode penyelesaian yang dilakukan oleh fungsi ini berada dalam *domain* frekuensi. Berikut merupakan langkah-langkah yang dilakukan dalam proses *periodic noise restoration*:

1. Sistem menerima input berupa citra dari pengguna. Selain itu, sistem juga menerima tipe filter yang akan digunakan dan juga konfigurasi dari filter-filter tersebut.
2. Sistem melakukan transformasi citra tersebut menggunakan transformasi fourier agar dihasilkan tampilan citra tersebut dalam *domain* frekuensi.
3. Pada tampilan hasil *transformasi fourier*, pengguna dapat melihat bagaimana pola yang dibentuk akibat derau pada citra.
4. Pengguna memasukkan konfigurasi berupa W (*width*), D0 (*low cutoff*), dan D1 (*high cutoff*) ke sistem agar penyelesaian derau dapat lebih optimal dilakukan.
5. Sistem melakukan penyelesaian derau tersebut dengan menggunakan tipe filter yang dimasukkan oleh pengguna. Tipe filter tersebut antara lain adalah *band-reject*, *band-pass*, dan *notch*.
6. Sistem menghasilkan sebuah citra baru berdasarkan proses penggunaan filter tersebut dan mengembalikan citra tersebut sebagai *output*.

- F. Analisis cara kerja program untuk proses menghilangkan *Motion Blur* dengan Penapis Wiener

Berikut merupakan langkah-langkah yang dilakukan dalam proses menghilangkan *Motion Blur* dengan Penapis Wiener:

1. Program menerima sejumlah input wajib dari pengguna yaitu gambar, nilai len dan theta untuk pemrosesan penapis, dan noise\_var untuk ukuran derau aditif

2. Setelahnya, gambar akan dibuat blur terlebih dahulu dengan menjalankan fungsi *motion blur* untuk proses blur. Dari program kami, proses *blurring* dibuat secara circular untuk *defaultnya*
3. Setelah terbentuk gambar blur, program kemudian memanggil fungsi untuk menjalankan penapisan Wiener secara scratch dengan perhitungan formula Wiener untuk setiap channel. Kondisi dibedakan berdasarkan gambar jika *grayscale*, iterasi akan dilakukan sekali saja dan untuk gambar RGB berwarna, akan dilakukan iterasi untuk tiap jenis warna
4. Untuk proses perhitungan formula Wiener dilakukan dengan memanggil fungsi *wiener\_calculation* dengan proses sebagai berikut:
  - a. Menjalankan Fourier Transform untuk image dan blur kernel secara terpisah. Image kernel untuk  $G(u,v)$  dan Blur kernel untuk  $H(u,v)$
  - b. Menjalankan padding dan circular shift untuk memastikan ukuran gambar tetap sama setelah proses transformasi Fourier
  - c. Menghitung penapis Wiener dengan formula sebagai berikut

$$H_w(u, v) = \frac{1}{H(u, v)} \cdot \frac{|H(u, v)|^2}{|H(u, v)|^2 + noiseVar}$$

- d. Mendapatkan nilai citra restorasi dengan mengalikan penapis tersebut dengan citra degradasi dengan rumus  $F(u, v) = H_w(u, v) * G(u, v)$
- e. Scale gambar ke dalam bentuk *gray* menggunakan fungsi *mat2gray* untuk mendapatkan hasil akhir

## **BAB IV**

### **KESIMPULAN**

Kesimpulan yang didapat dari penggerjaan tugas ini adalah sebagai berikut:

1. Penapisan citra dengan ranah spasial dapat dilakukan dengan memanfaatkan konvolusi terhadap matriks dengan rumus tertentu seperti *Mean* dan *Gaussian Filter*. Pengaplikasian dari cara ini dapat dilakukan seperti untuk proses *Image Smoothing* untuk penghalusan citra
2. Terdapat banyak implementasi perbaikan citra yang dapat dilakukan dengan penapisan citra ranah frekuensi, antara lain seperti *Image Smoothing* dengan *Low-Pass Filter*, *Edge Detection* dengan *High-Pass Filter*, dan penerangan citra dengan *Homomorphic Filter*
3. Perbaikan derau *noise* dan periodik dapat dilakukan dengan memanfaatkan berbagai macam implementasi filter
4. Citra dengan kondisi *Motion Blur* dapat dilakukan perbaikan dengan memanfaatkan penapis Wiener untuk mengembalikan citra ke dalam kondisi semula

## DAFTAR PUSTAKA

- [1] “Image Enhacement Bagian 3”. Rinaldi Munir. <https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/10-Image-Enhancement-Bagian3-2024.pdf> (Diakses pada tanggal 17 Oktober 2024)
- [2] “Penapisan Citra dalam Ranah Frekuensi”. Rinaldi Munir. <https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/12-Penapisan-Citra-dalam-Ranah-Frekuensi-2024.pdf> (Diakses pada tanggal 17 Oktober 2024)
- [3] “How can I implement a Homomorphic filter in Matlab?” Stackoverflow. <https://stackoverflow.com/questions/45248163/how-can-i-implement-a-homomorphic-filter-in-matlab> (Diakses pada tanggal 20 Oktober 2024)
- [4] “Frequency domain filtering image processing - High pass Sharpening - High boost - Homomorphic Part 2” Youtube. <https://www.youtube.com/watch?v=gZAO9U4UbTI> (Diakses pada tanggal 20 Oktober 2024)
- [5] “Restorasi Citra Bagian 2”. Rinaldi Munir. <https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/14-Restorasi-citra-bagian2-2024.pdf> (Diakses pada tanggal 24 Oktober 2024)

## **LAMPIRAN**

Pranala Github: [https://github.com/Gulilil/IF4073\\_Tugas2](https://github.com/Gulilil/IF4073_Tugas2)