Laporan Tugas 2
Interpretasi dan Pengolahan Citra

Disusun oleh:

13519008 Ronggur Mahendra

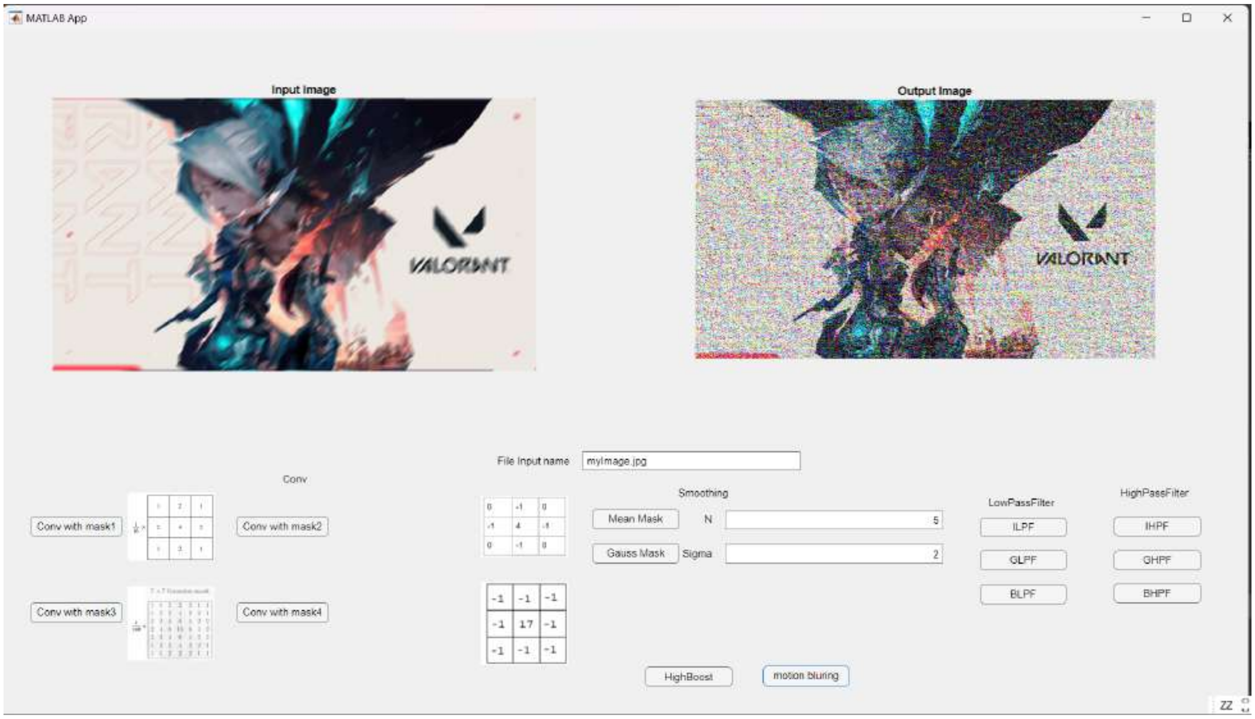
PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

1. Screenshot GUI



File Input name: myImage.jpg

N: 5

Sigma: 2

1	2	1
2	4	2
1	2	1

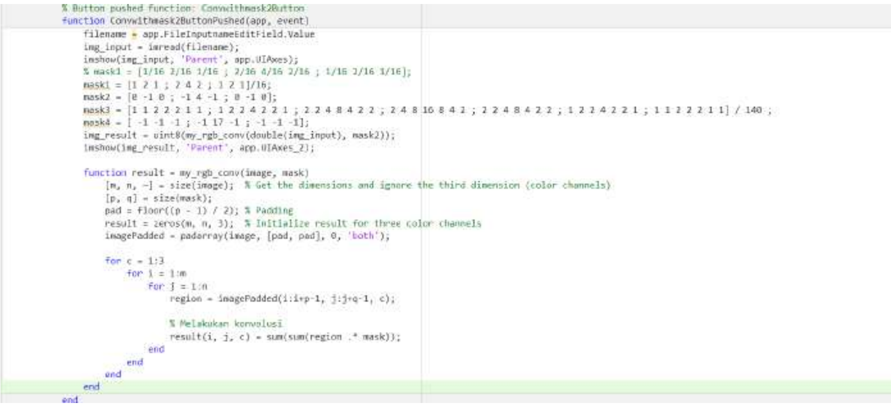
0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	17	-1
-1	-1	-1

2. Program

a. Konvolusi

i. Kode Program



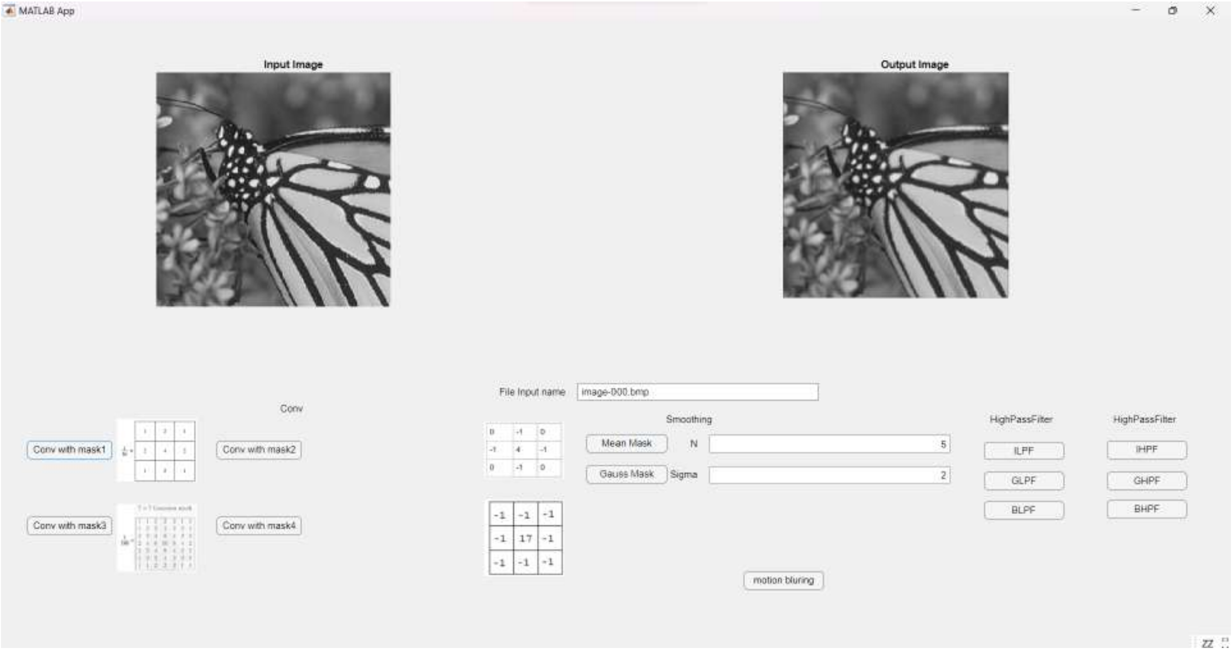
```
% Button pushed function: Convwithmask2Button
function Convwithmask2ButtonPushed(app, event)
    filename = app.FileInputnameEditField.Value
    img_input = imread(filename);
    imshow(img_input, 'Parent', app.UIAxes);
    % mask1 = [1/16 2/16 1/16 ; 2/16 4/16 2/16 ; 1/16 2/16 1/16];
    mask1 = [1 2 1 ; 2 4 2 ; 1 2 1]/16;
    mask2 = [0 -1 0 ; -1 4 -1 ; 0 -1 0];
    mask3 = [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] / 140 ;
    mask4 = [ -1 -1 -1 ; -1 17 -1 ; -1 -1 -1];
    img_result = uint8(my_rgb_conv(double(img_input), mask2));
    imshow(img_result, 'Parent', app.UIAxes_2);

    function result = my_rgb_conv(image, mask)
        [m, n, ~] = size(image);  % Get the dimensions and ignore the third dimension (color channels)
        [p, q] = size(mask);
        pad = floor((p - 1) / 2); % Padding
        result = zeros(m, n, 3);  % Initialize result for three color channels
        imagePadded = padarray(image, [pad, pad], 0, 'both');

        for c = 1:3
            for i = 1:m
                for j = 1:n
                    region = imagePadded(i:i+p-1, j:j+q-1, c);

                    % Melakukan konvolusi
                    result(i, j, c) = sum(sum(region .* mask));
                end
            end
        end
    end
end
```

Cat : dengan parameter masukan disesuaikan dengan tombol pada GUI

ii. Hasil Eksekusi

1. Mask 1

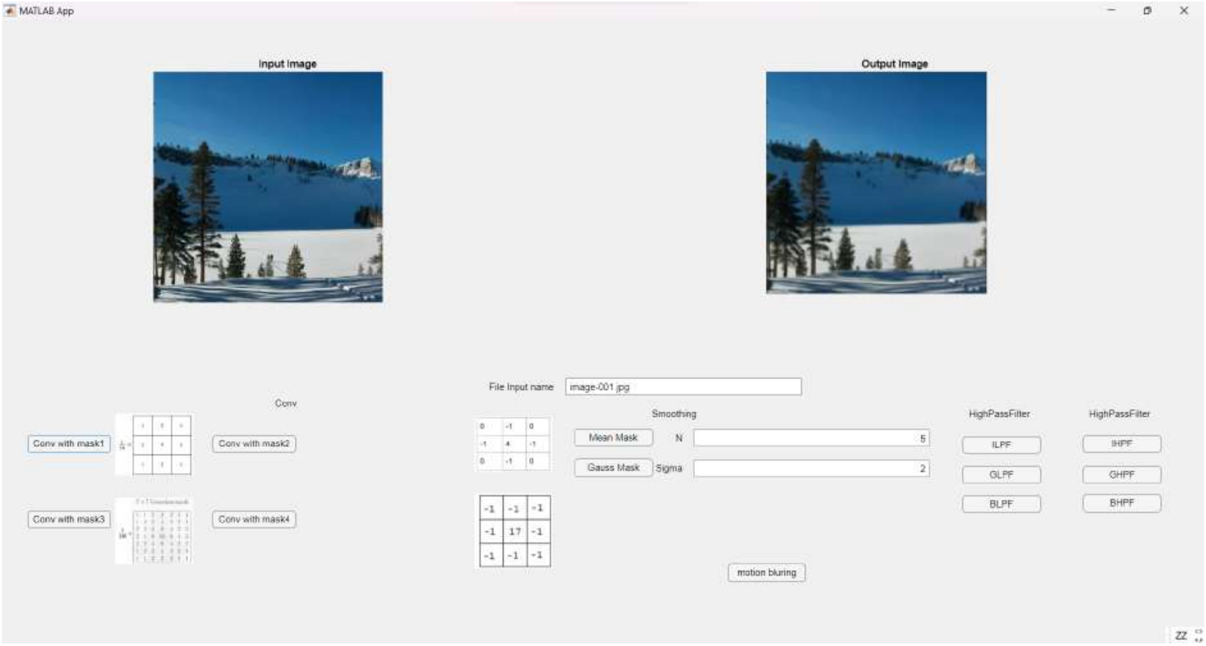
$$\frac{1}{16} \times \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline\end{array}$$



File Input name: image-000.bmp

N: 5

Sigma: 2



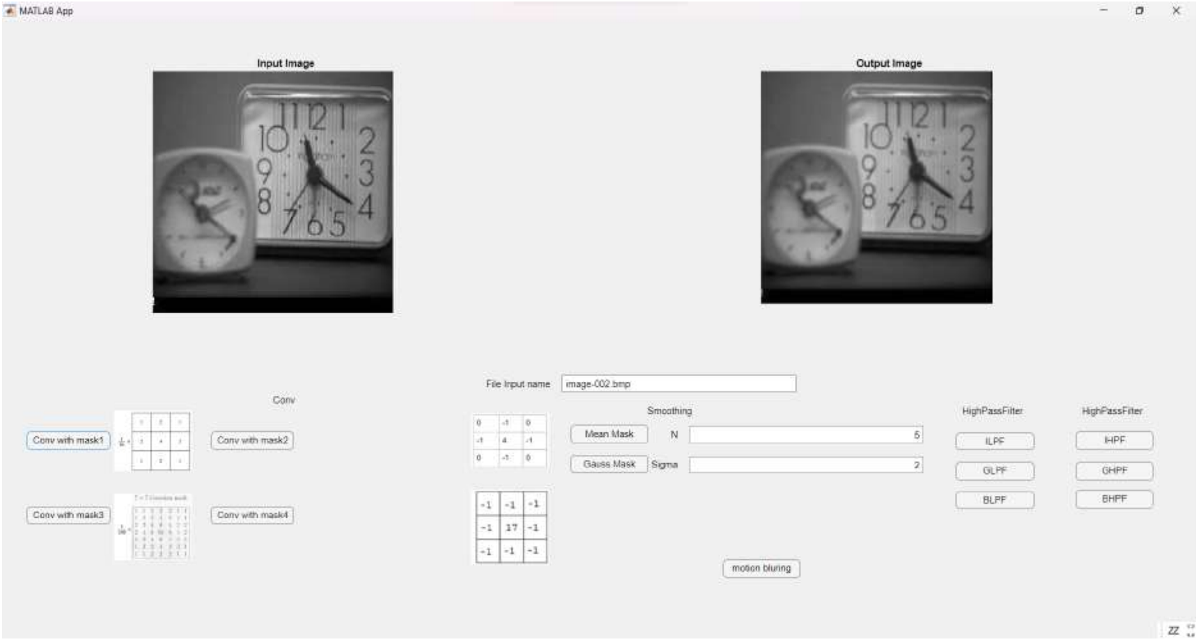
File Input name: image-001.jpg

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



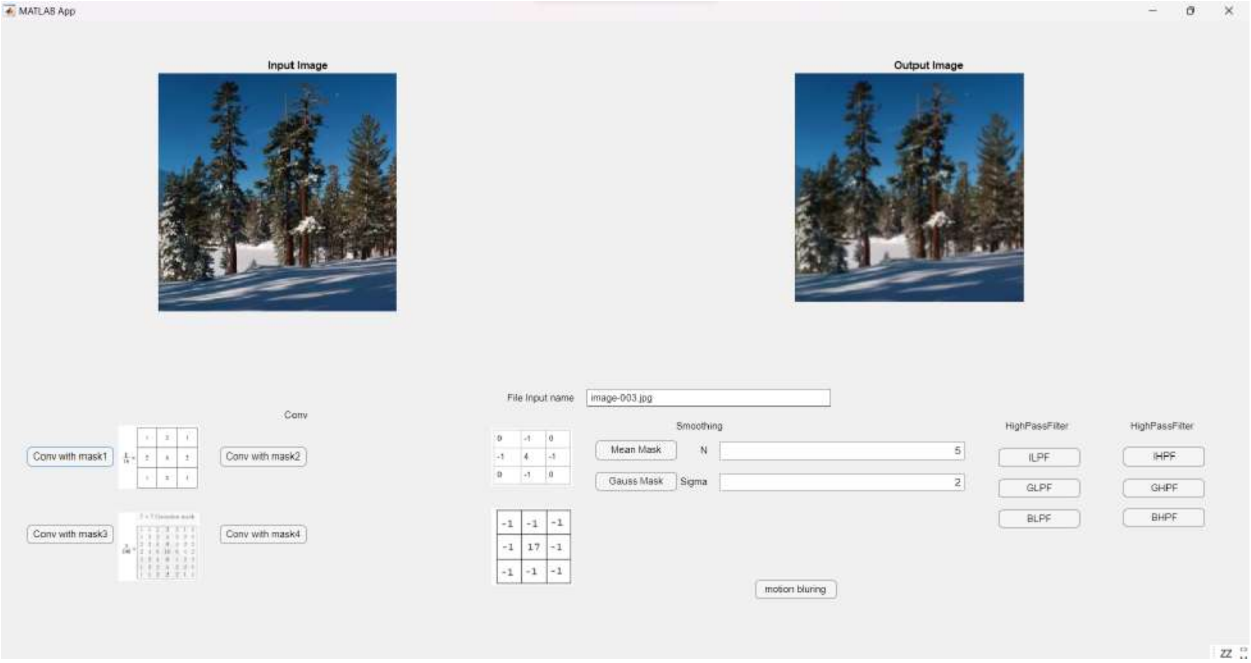
File Input name: image-002.bmp

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



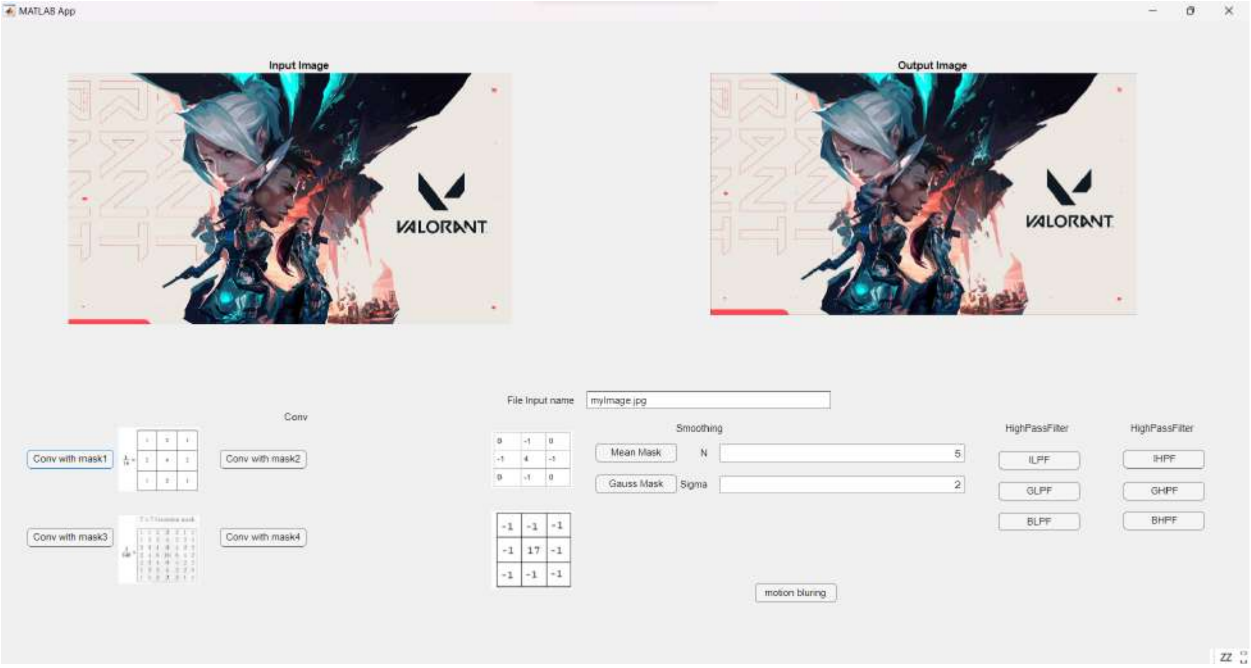
File Input name: image-003.jpg

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



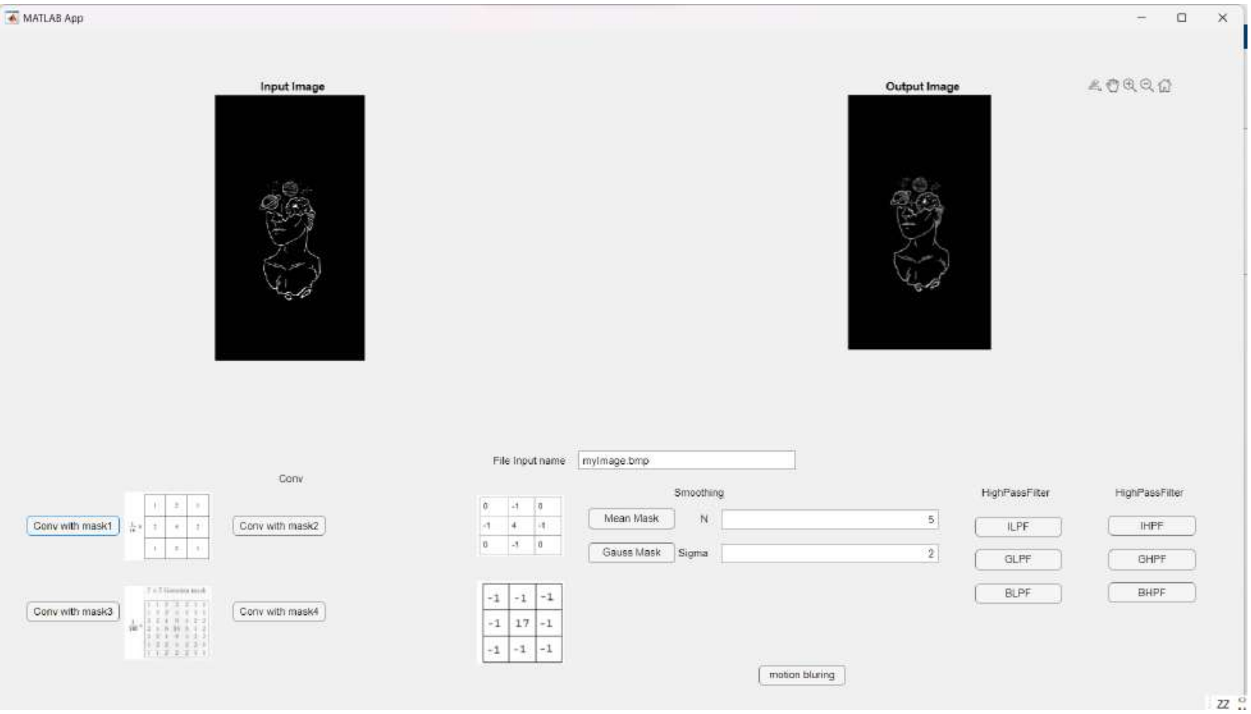
File Input name: myImage.jpg

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



File Input name: myImage.bmp

N: 5

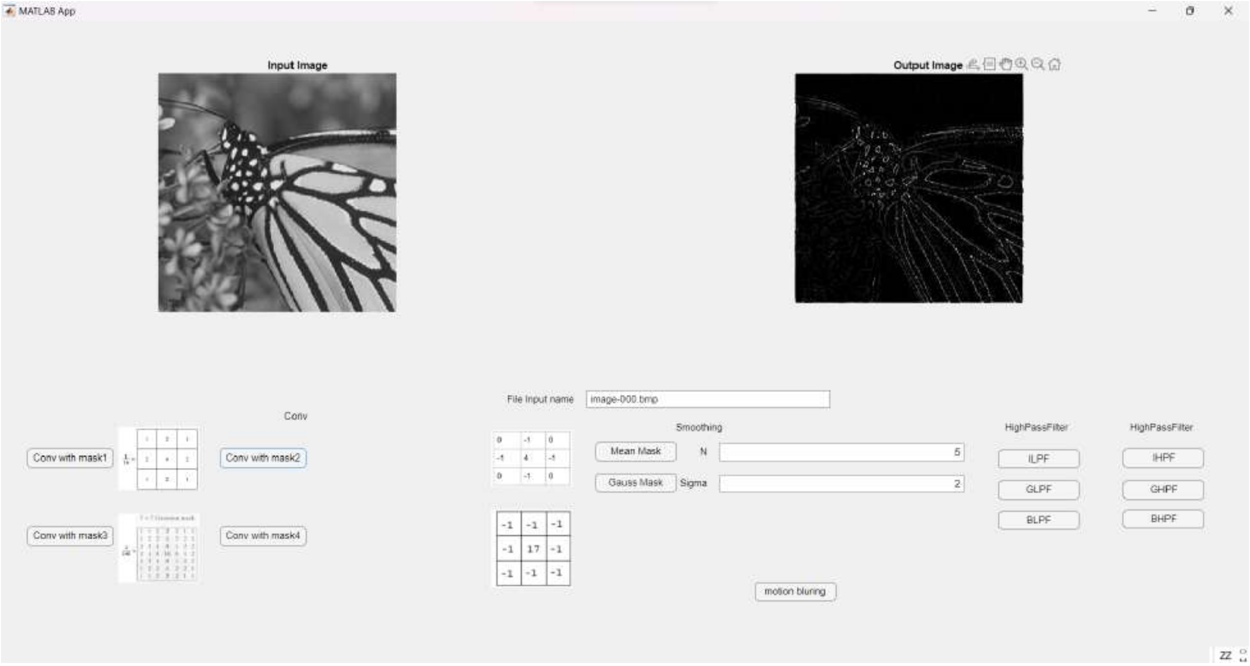
Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1

2. Mask 2

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



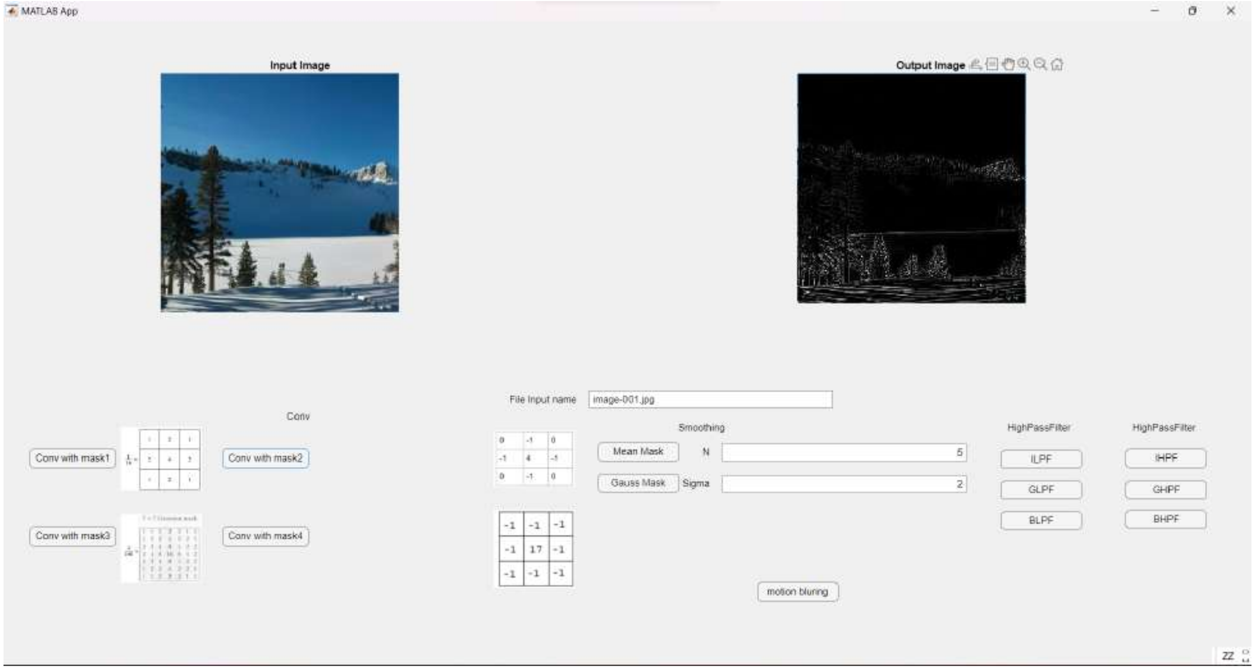
File Input name: image-000.bmp

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



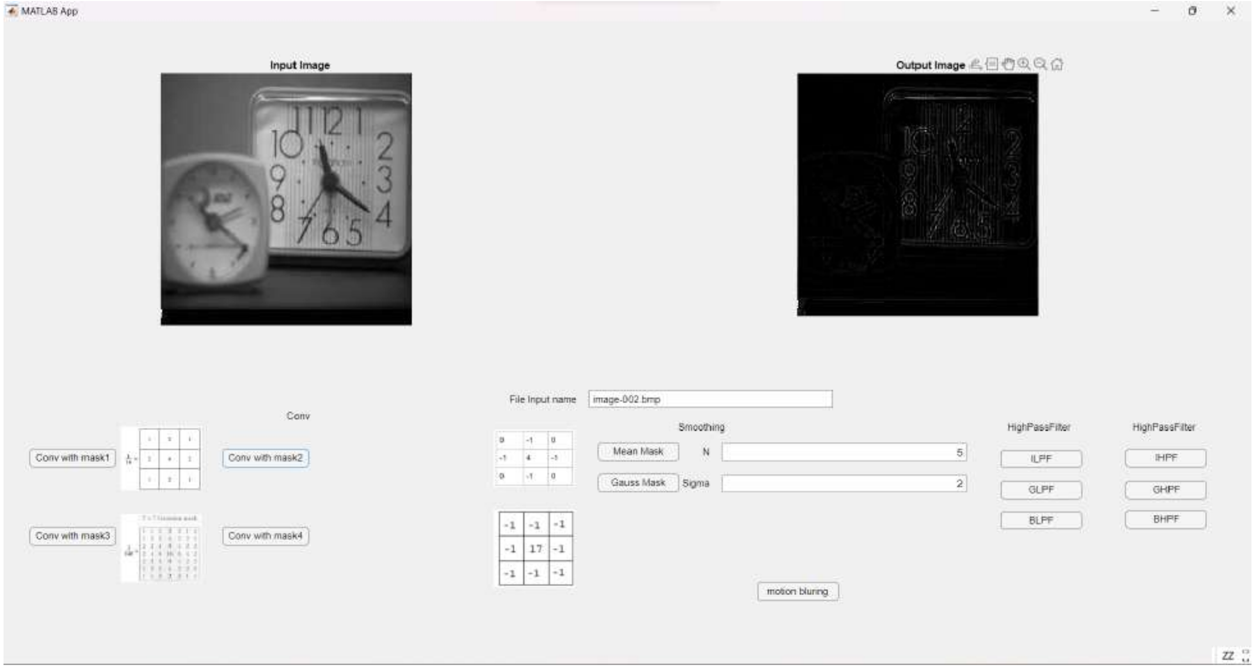
File Input name: image-001.jpg

N: 5

Sigma: 2

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

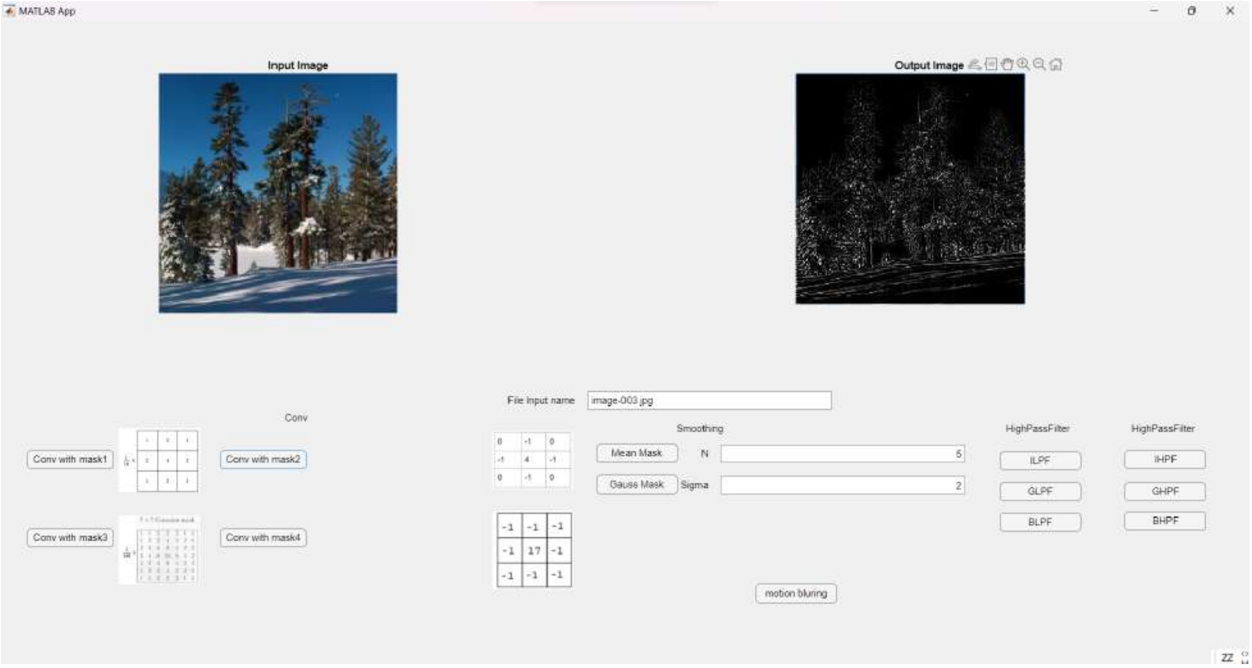
-1	-1	-1
-1	17	-1
-1	-1	-1



File Input name: image-002.bmp

N: 5

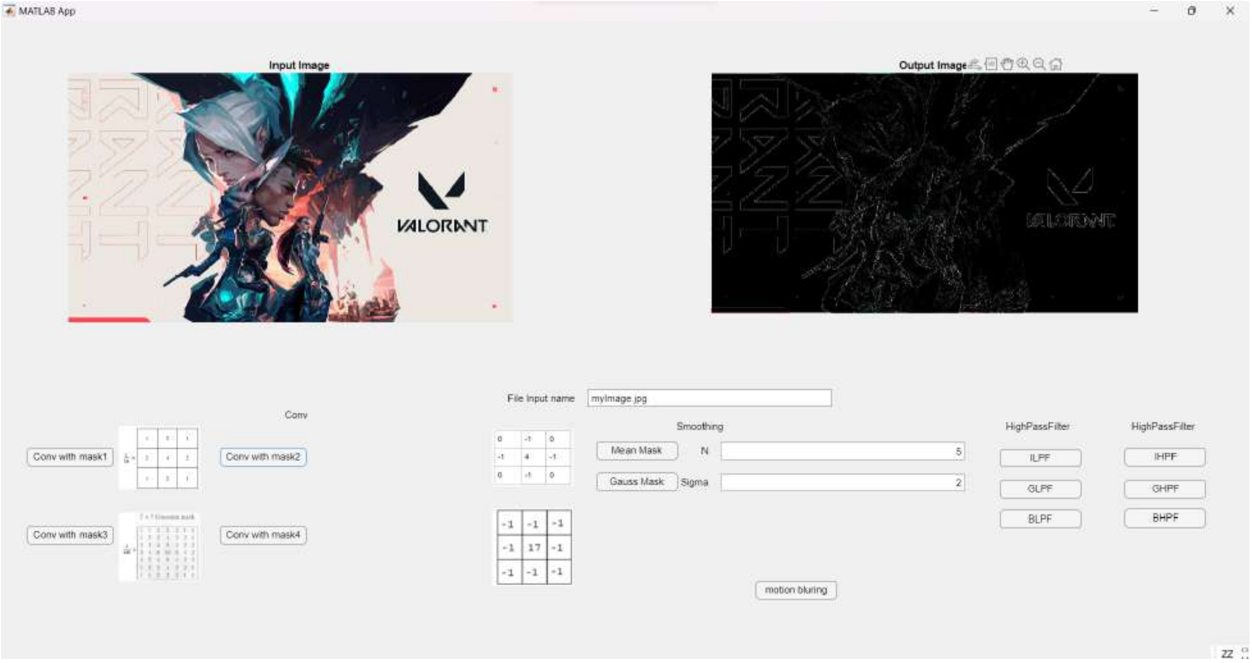
Sigma: 2



File Input name: image-003.jpg

N: 5

Sigma: 2



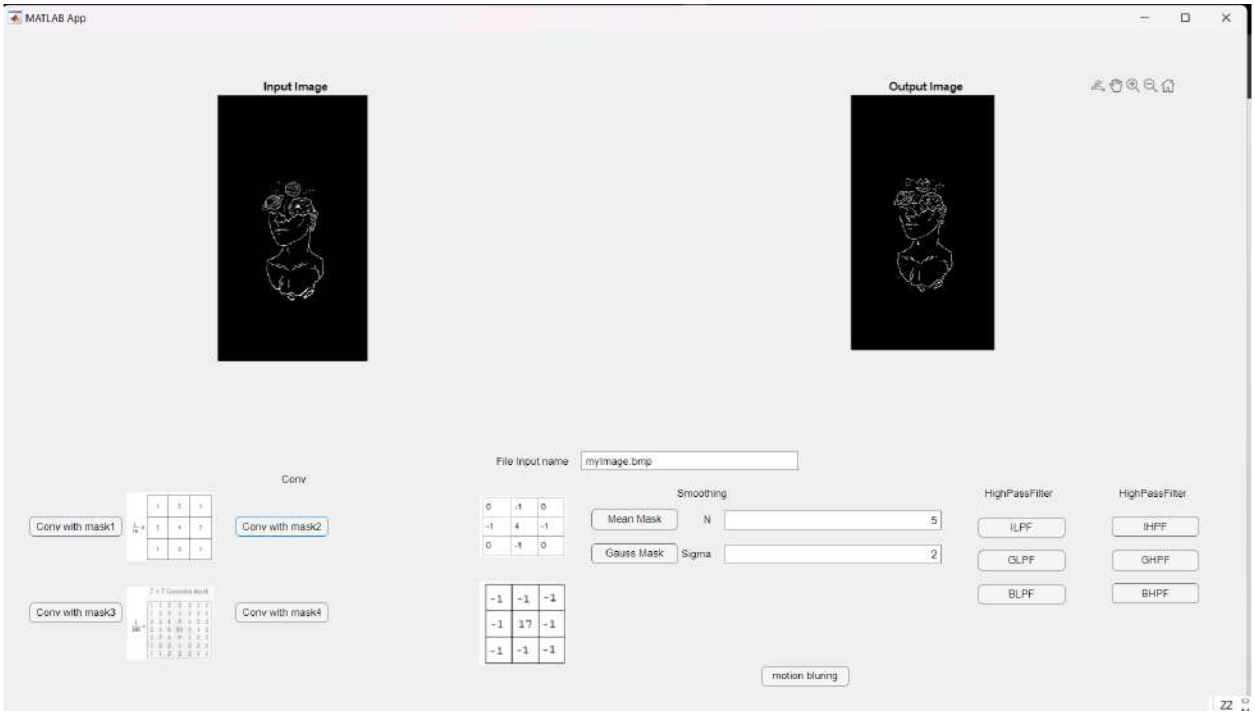
File Input name: myImage.jpg

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



File Input name: myImage.bmp

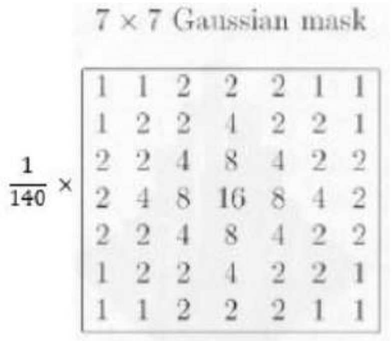
N: 5

Sigma: 2

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

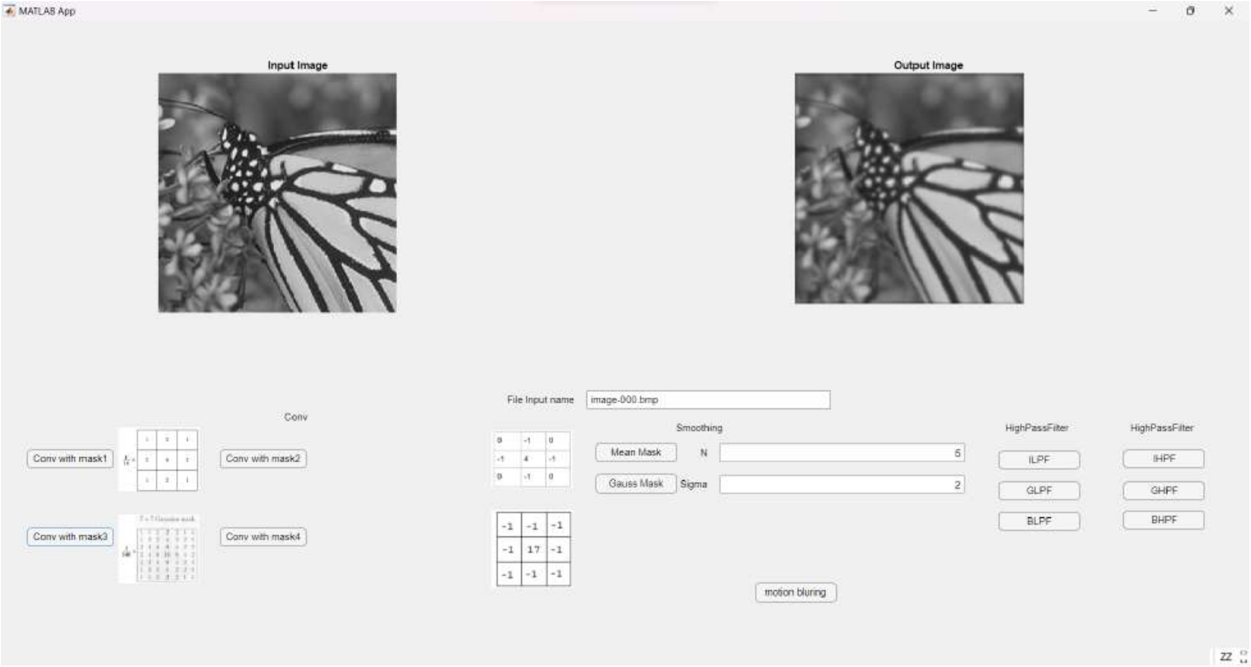
-1	-1	-1
-1	17	-1
-1	-1	-1

3. Mask 3



7 × 7 Gaussian mask

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



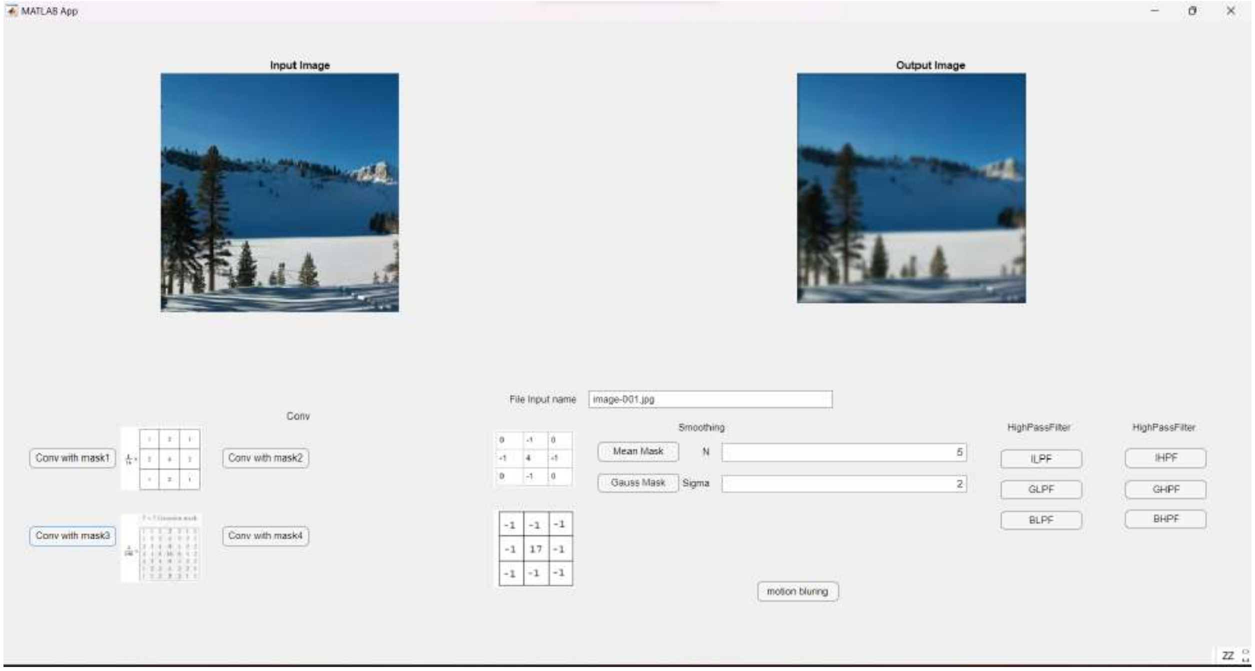
File Input name image-000.bmp

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

N 5

Sigma 2

-1	-1	-1
-1	17	-1
-1	-1	-1



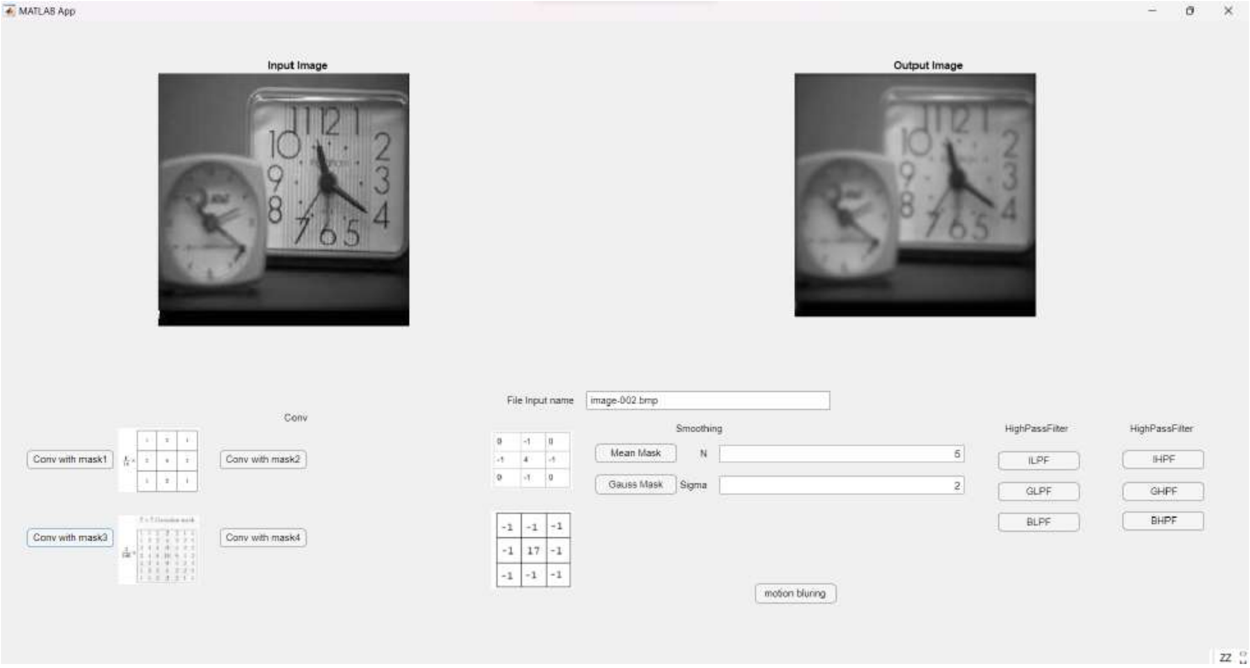
File Input name: image-001.jpg

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



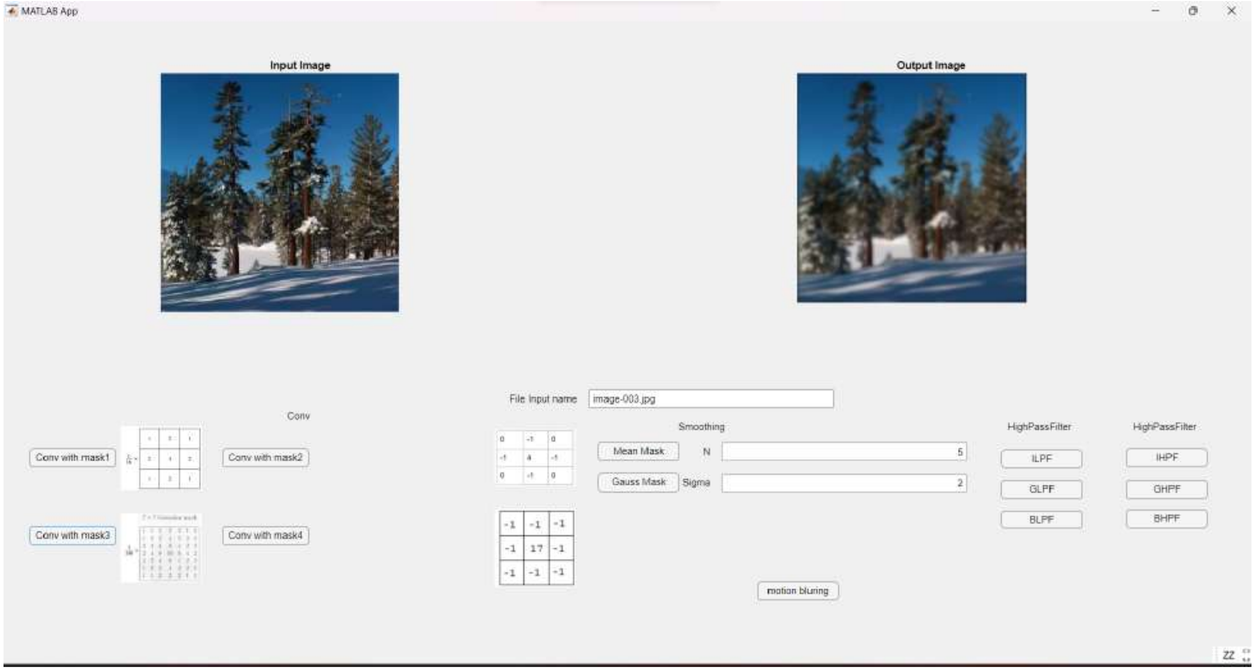
File Input name: image-002.bmp

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



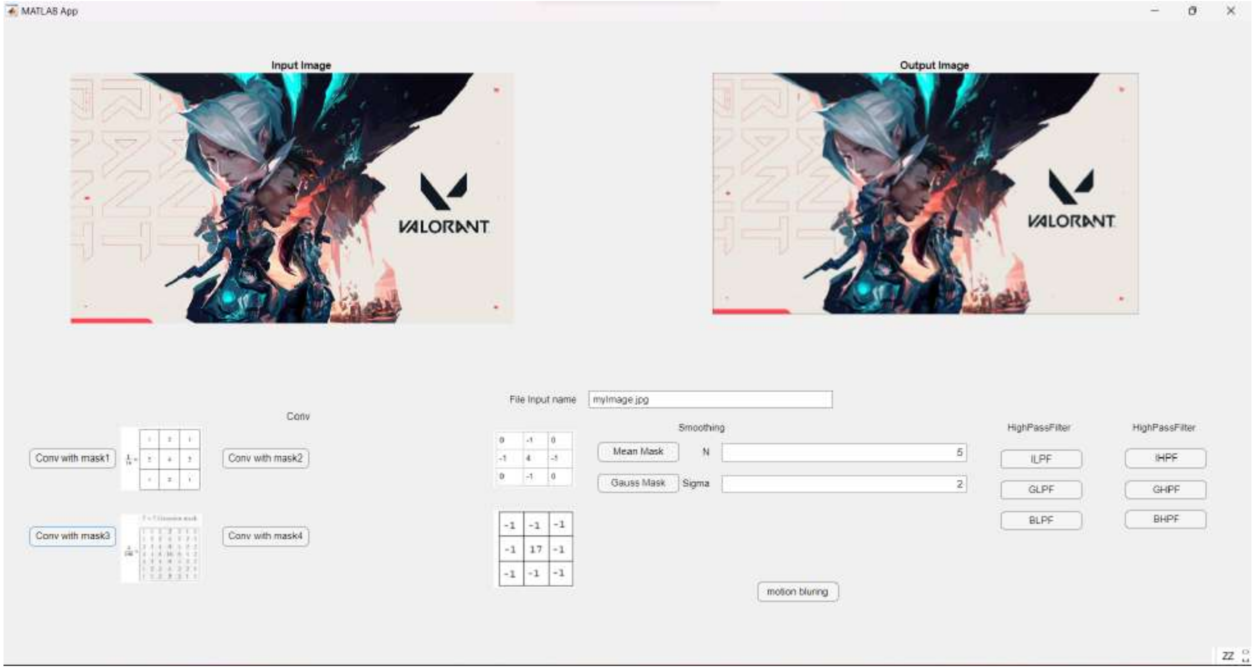
File Input name image-003.jpg

N 5

Sigma 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



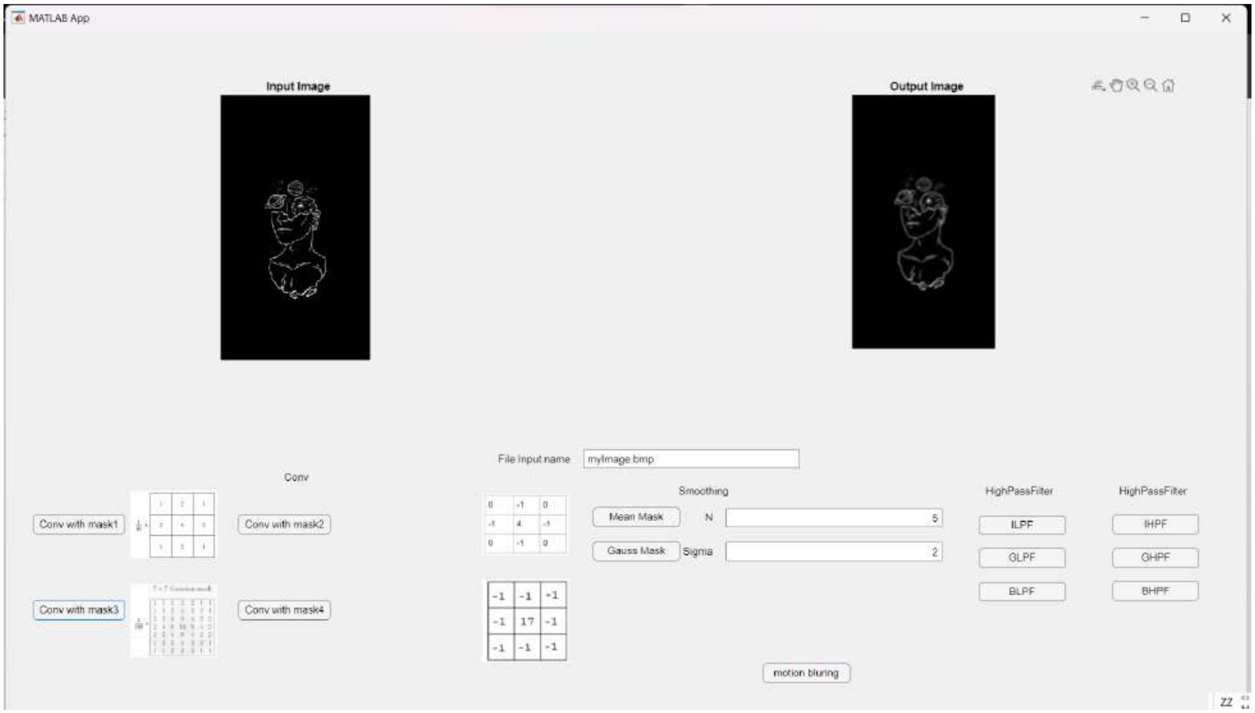
File Input name myImage.jpg

N 5

Sigma 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



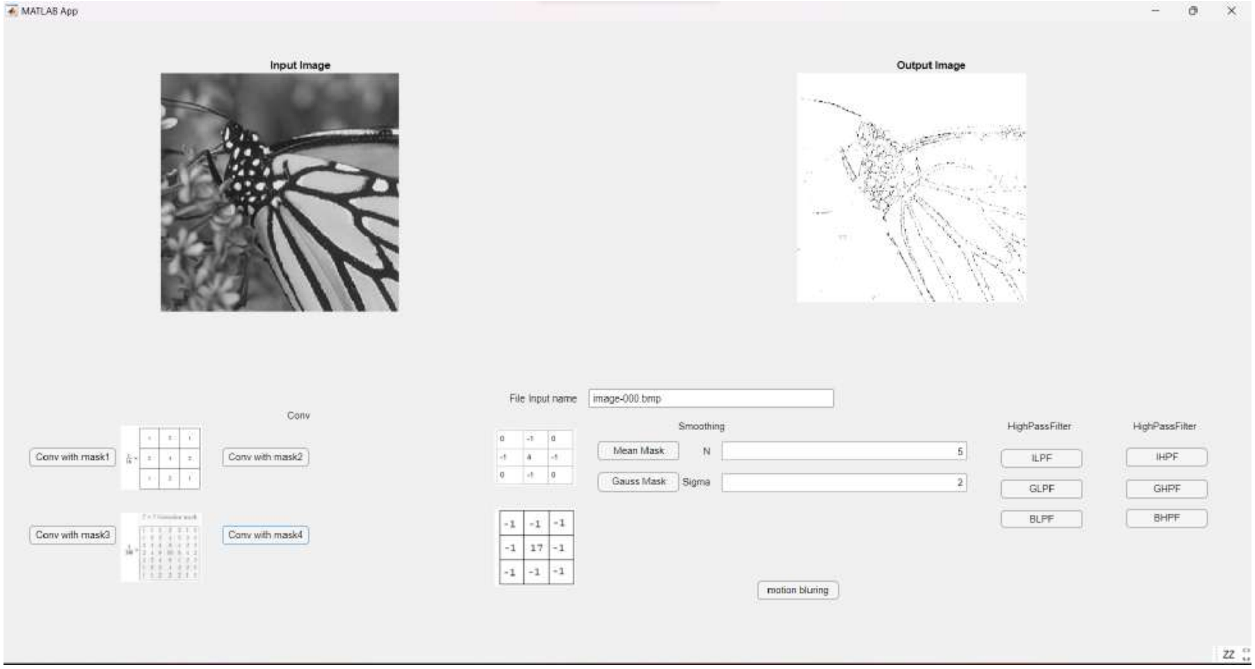
File Input name: myImage.bmp

N: 5

Sigma: 2

4. Mask 4

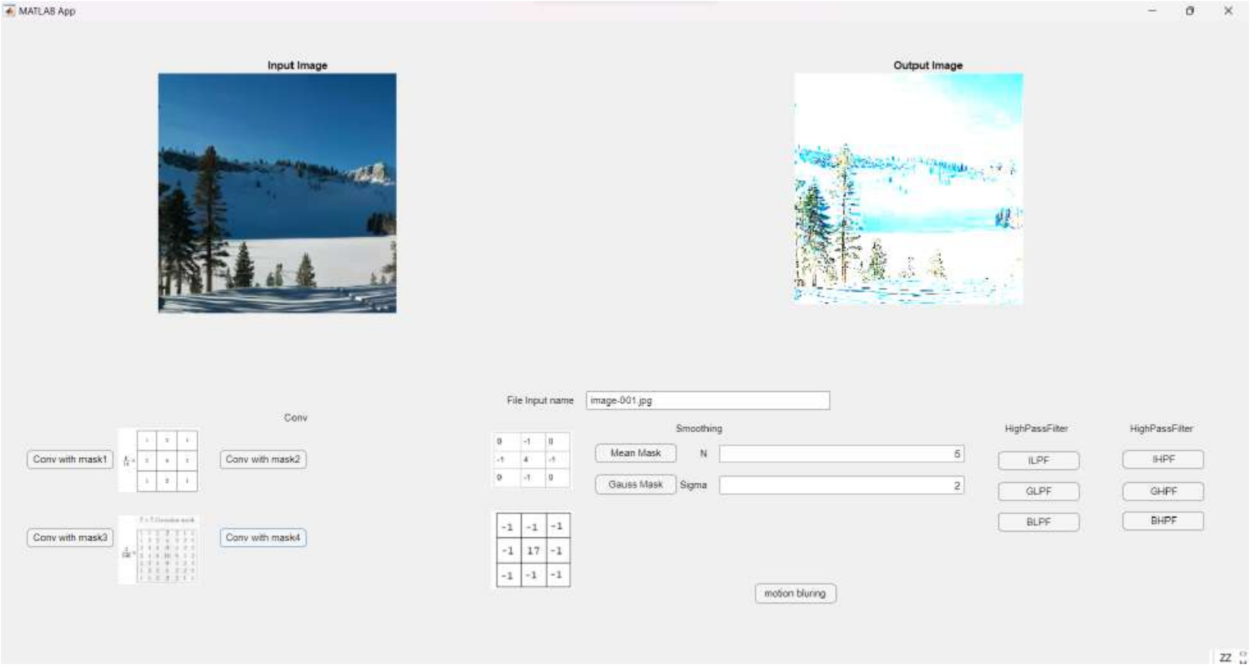
-1	-1	-1
-1	17	-1
-1	-1	-1



File Input name: image-000.bmp

N: 5

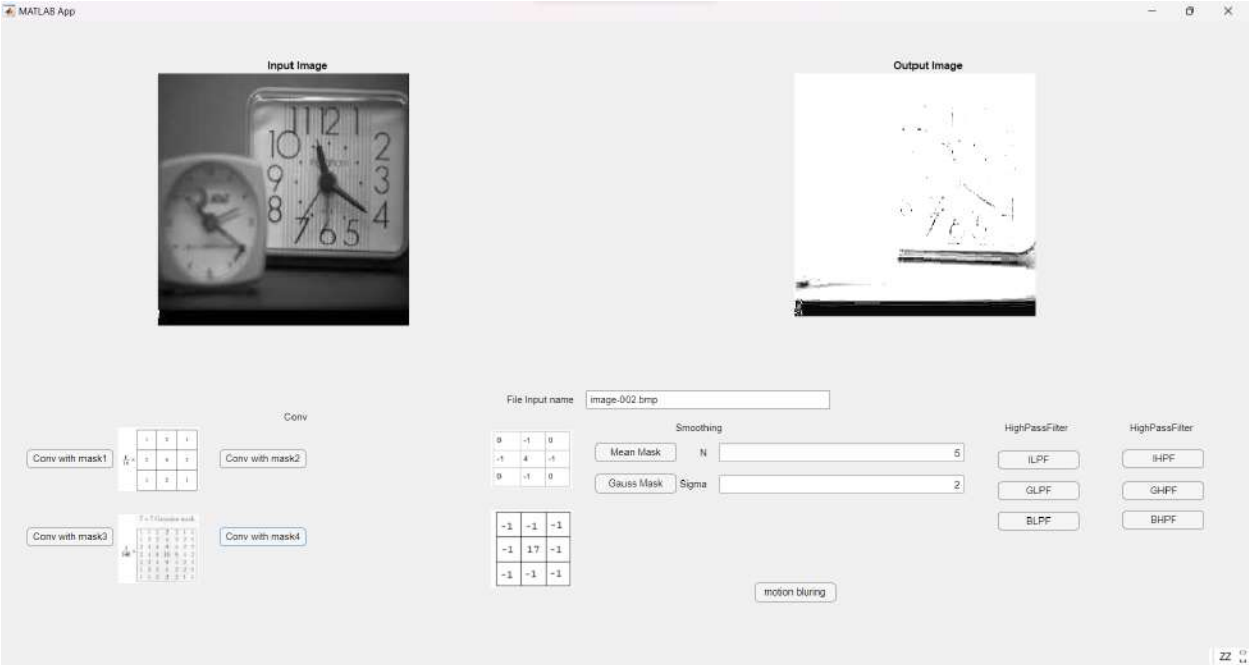
Sigma: 2



File Input name: image-001.jpg

N: 5

Sigma: 2



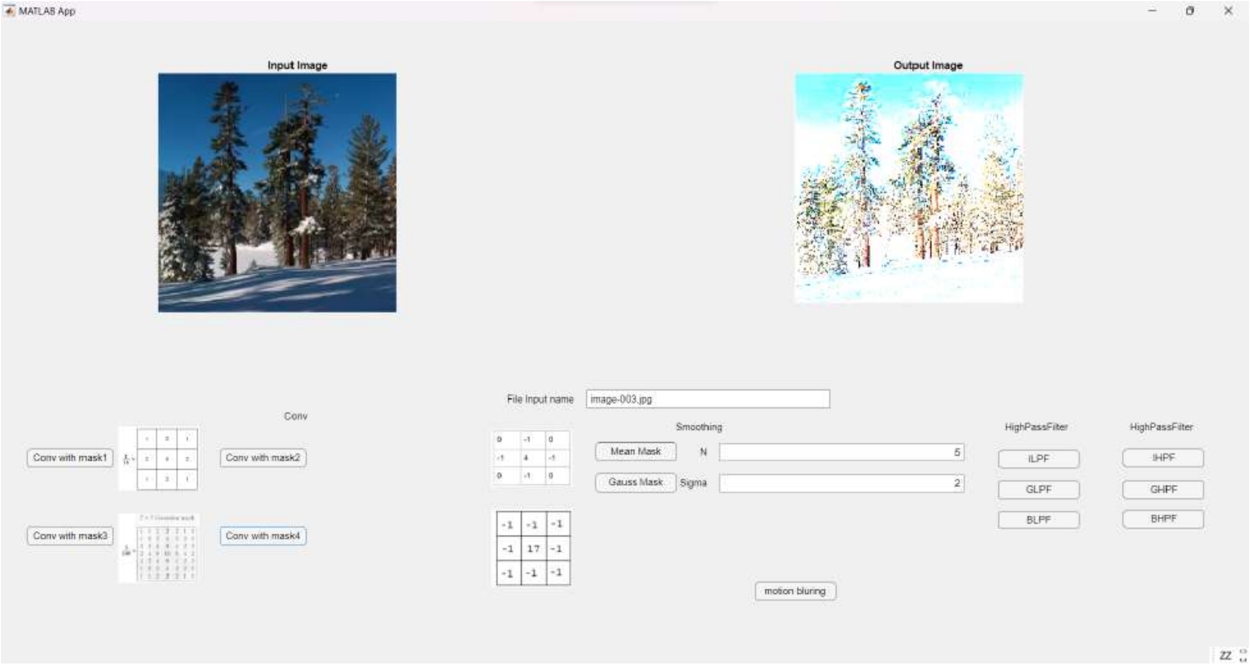
File Input name: image-002.bmp

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



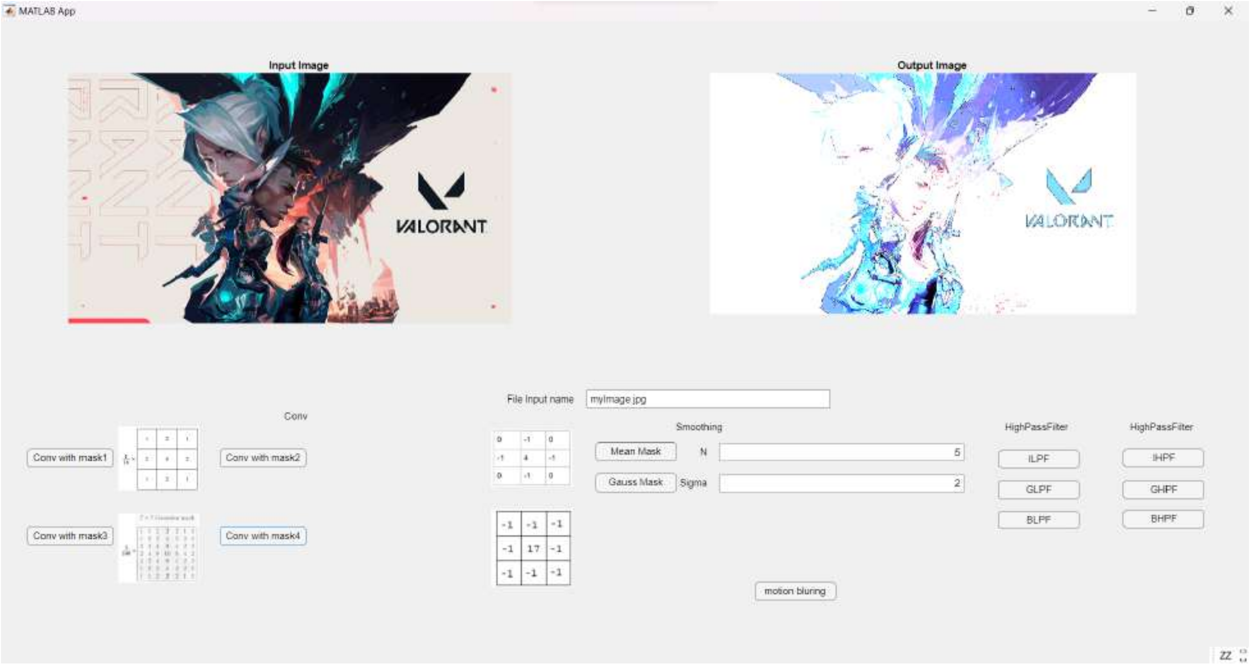
File Input name: image-003.jpg

N: 5

Sigma: 2

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

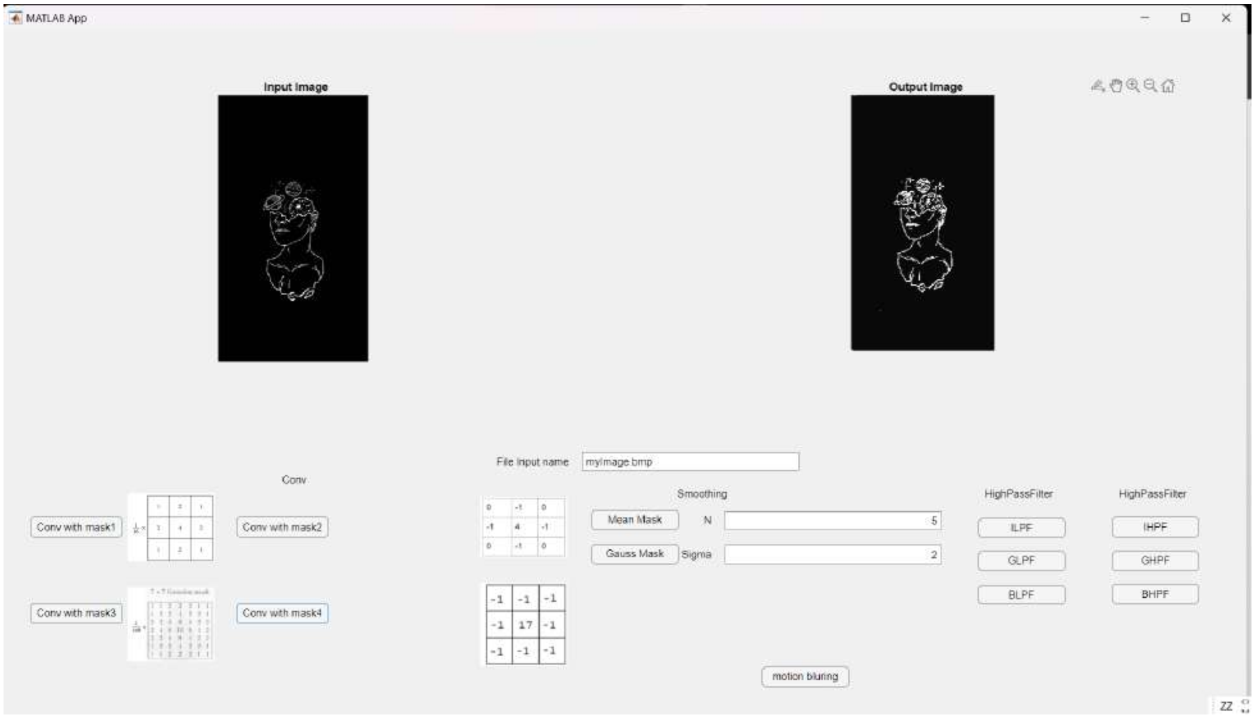
-1	-1	-1
-1	17	-1
-1	-1	-1



File Input name: myImage.jpg

N: 5

Sigma: 2



File Input name: myImage.bmp

N: 5

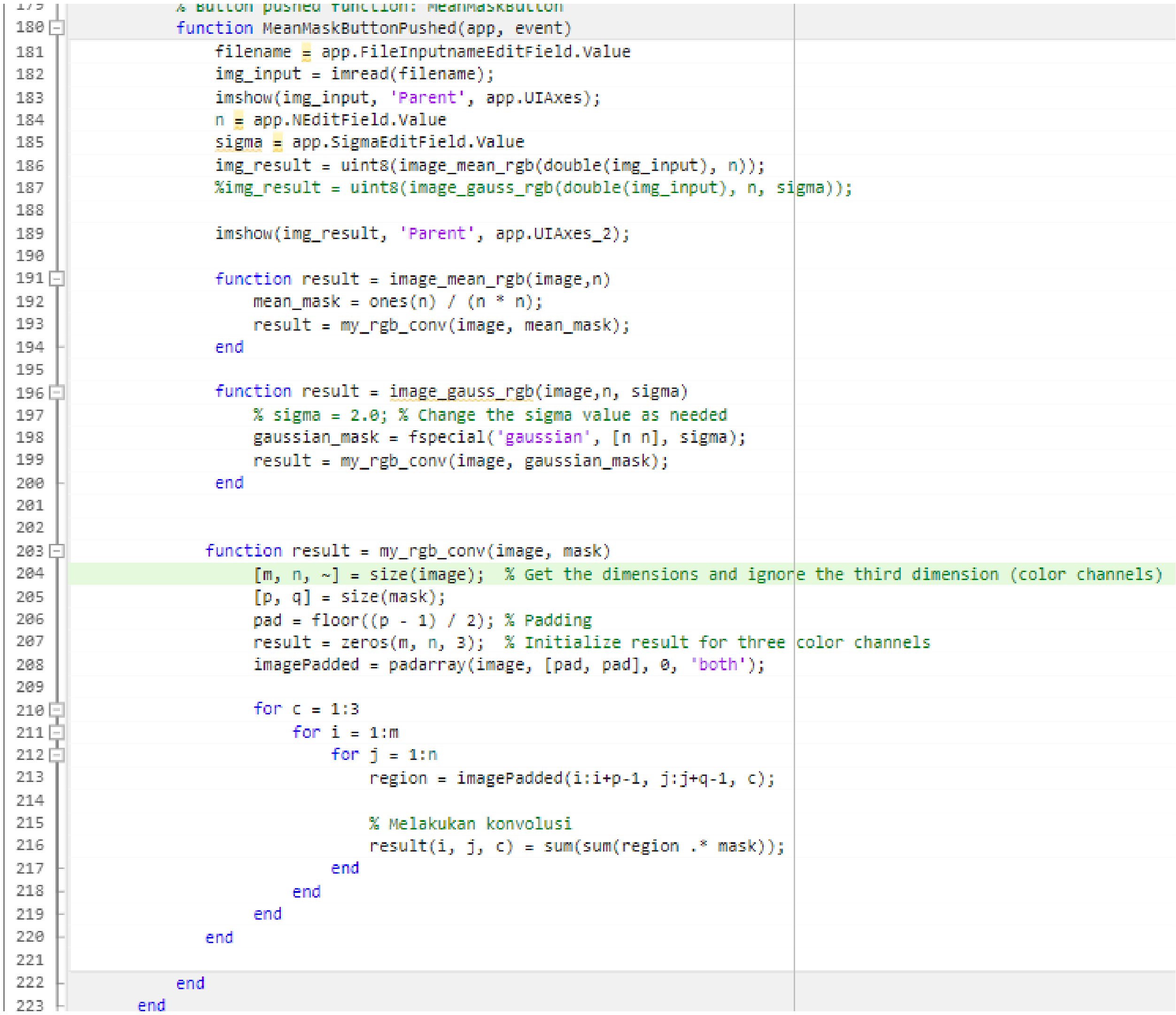
Sigma: 2

iii. Analisa

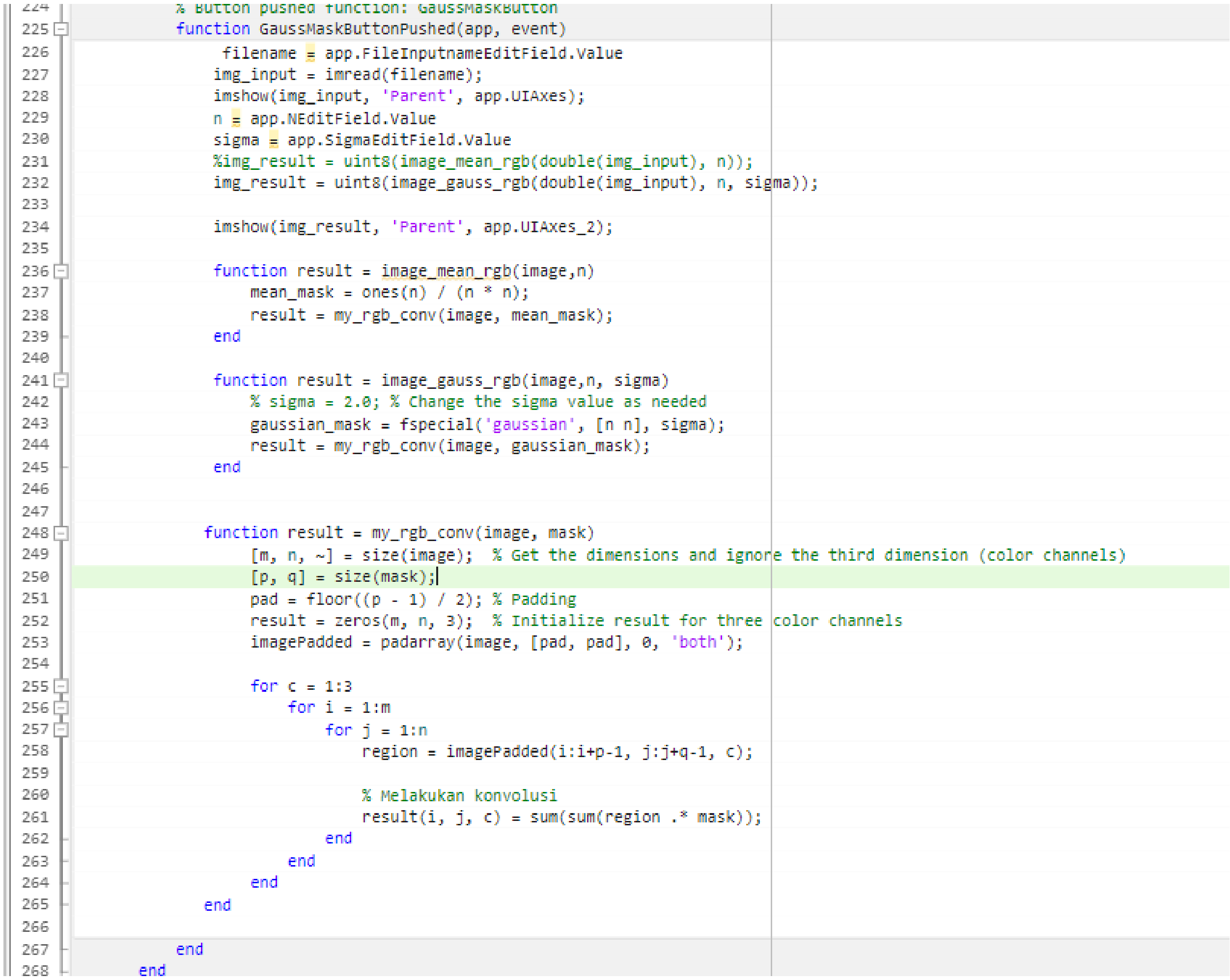
- Mask 1 adalah mask yang menghasilkan efek blurring
- Mask 2 adalah mask yang menghasilkan efek edge detection 4 sisi
- Mask 3 adalah mask gaussian dan menghasilkan efek blur
- Mask 4 adalah masil yang menghasilkan efek edge detection namun diperlukan edge dari semua arah

b. Smoothing dan LowPass Filter

i. Kode Program



```
    % Button pushed function: MeanMaskButton
    function MeanMaskButtonPushed(app, event)
        filename = app.FileInputnameEditField.Value
        img_input = imread(filename);
        imshow(img_input, 'Parent', app.UIAxes);
        n = app.NEditField.Value
        sigma = app.SigmaEditField.Value
        img_result = uint8(image_mean_rgb(double(img_input), n));
        %img_result = uint8(image_gauss_rgb(double(img_input), n, sigma));

        imshow(img_result, 'Parent', app.UIAxes_2);

        function result = image_mean_rgb(image,n)
            mean_mask = ones(n) / (n * n);
            result = my_rgb_conv(image, mean_mask);
        end

        function result = image_gauss_rgb(image,n, sigma)
            % sigma = 2.0; % Change the sigma value as needed
            gaussian_mask = fspecial('gaussian', [n n], sigma);
            result = my_rgb_conv(image, gaussian_mask);
        end


       function result = my_rgb_conv(image, mask)
            [m, n, ~] = size(image);  % Get the dimensions and ignore the third dimension (color channels)
            [p, q] = size(mask);
            pad = floor((p - 1) / 2); % Padding
            result = zeros(m, n, 3);  % Initialize result for three color channels
            imagePadded = padarray(image, [pad, pad], 0, 'both');

            for c = 1:3
                for i = 1:m
                    for j = 1:n
                        region = imagePadded(i:i+p-1, j:j+q-1, c);

                        % Melakukan konvolusi
                        result(i, j, c) = sum(sum(region .* mask));
                    end
                end
            end
       end

    end
end
```

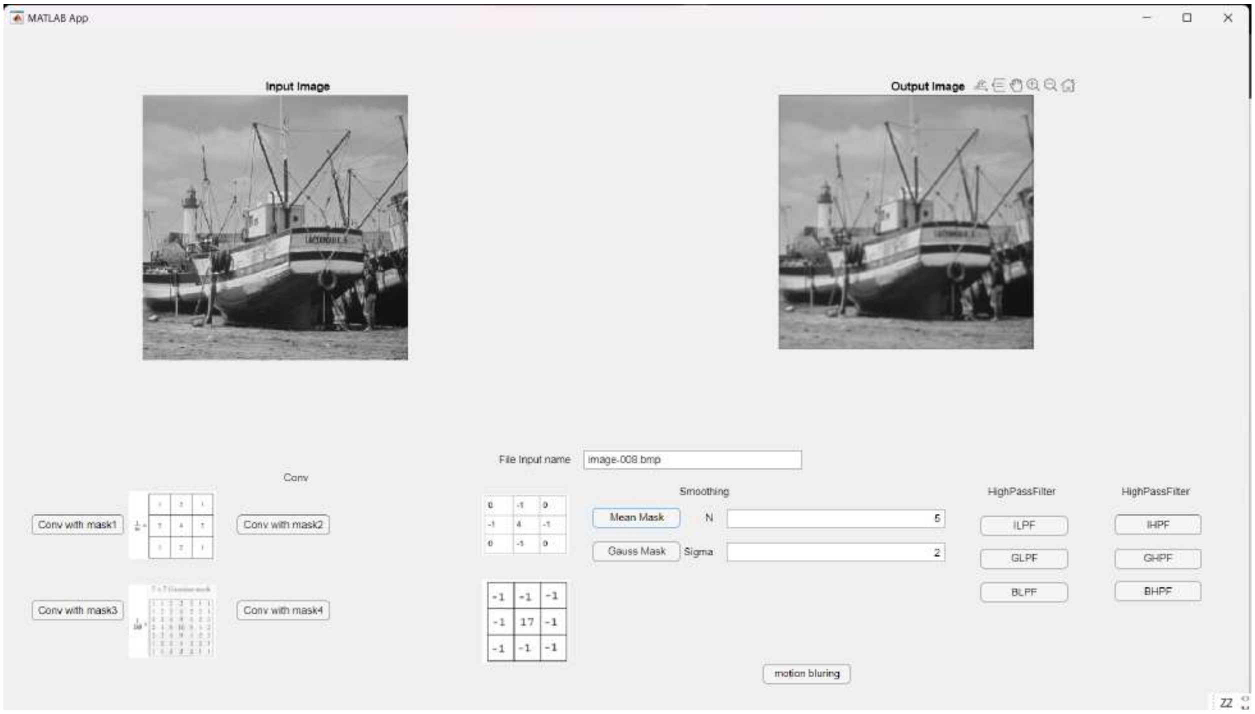


```
        % Button pushed function: GaussMaskButton
        function GaussMaskButtonPushed(app, event)
             filename = app.FileInputnameEditField.Value
            img_input = imread(filename);
            imshow(img_input, 'Parent', app.UIAxes);
            n = app.NEditField.Value
            sigma = app.SigmaEditField.Value
            %img_result = uint8(image_mean_rgb(double(img_input), n));
            img_result = uint8(image_gauss_rgb(double(img_input), n, sigma));

            imshow(img_result, 'Parent', app.UIAxes_2);

            function result = image_mean_rgb(image,n)
                mean_mask = ones(n) / (n * n);
                result = my_rgb_conv(image, mean_mask);
            end

            function result = image_gauss_rgb(image,n, sigma)
                % sigma = 2.0; % Change the sigma value as needed
                gaussian_mask = fspecial('gaussian', [n n], sigma);
                result = my_rgb_conv(image, gaussian_mask);
            end


           function result = my_rgb_conv(image, mask)
                [m, n, ~] = size(image);  % Get the dimensions and ignore the third dimension (color channels)
                [p, q] = size(mask);
                pad = floor((p - 1) / 2); % Padding
                result = zeros(m, n, 3);  % Initialize result for three color channels
                imagePadded = padarray(image, [pad, pad], 0, 'both');

                for c = 1:3
                    for i = 1:m
                        for j = 1:n
                            region = imagePadded(i:i+p-1, j:j+q-1, c);

                            % Melakukan konvolusi
                            result(i, j, c) = sum(sum(region .* mask));
                        end
                    end
                end
           end

        end
    end
```

Cat : apabila tombol gauss yang di tekan akan menggunakan fungsi image_gauss

ii. Hasil Eksekusi

1. Mean Filter



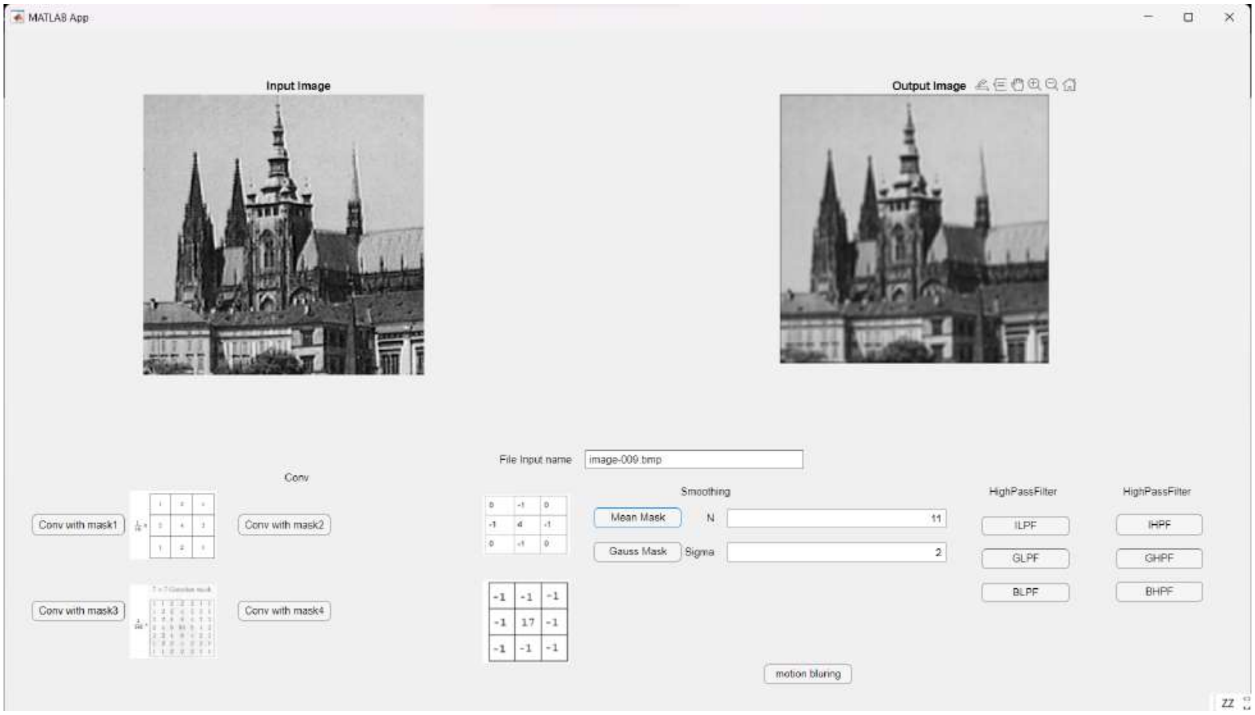
File Input name: image-008.bmp

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



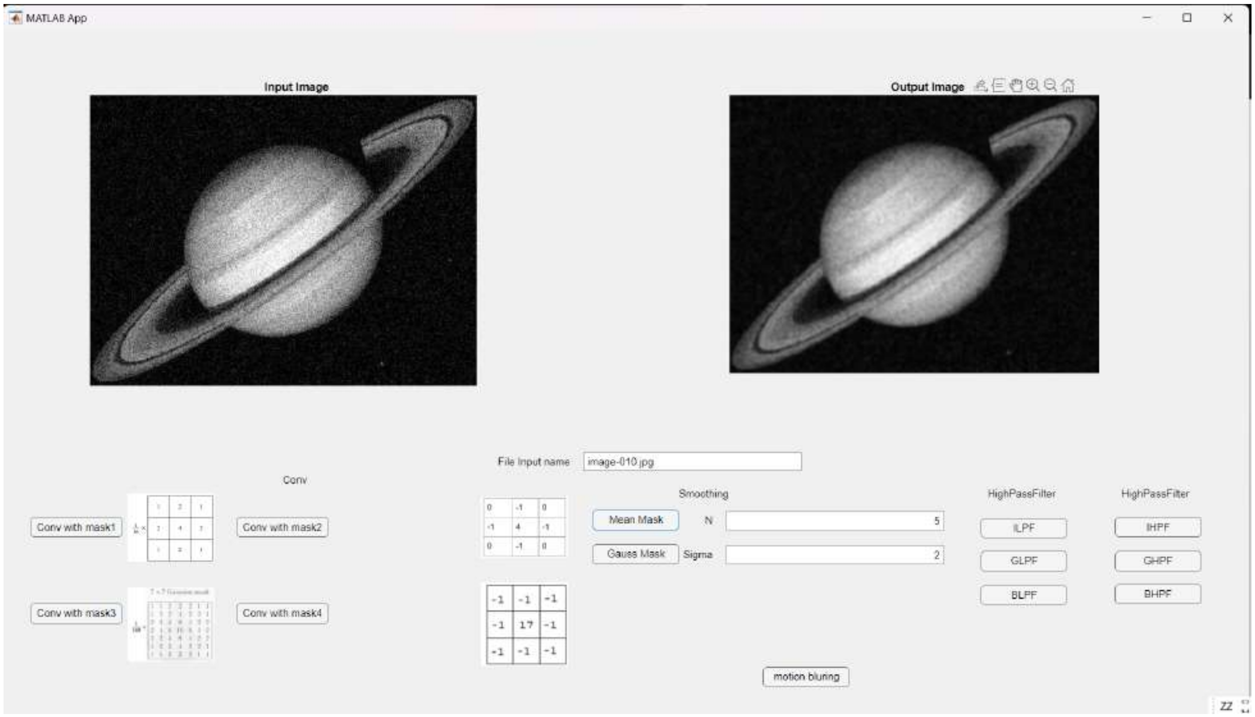
File Input name: image-009.bmp

N: 11

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



File Input name: image-010.jpg

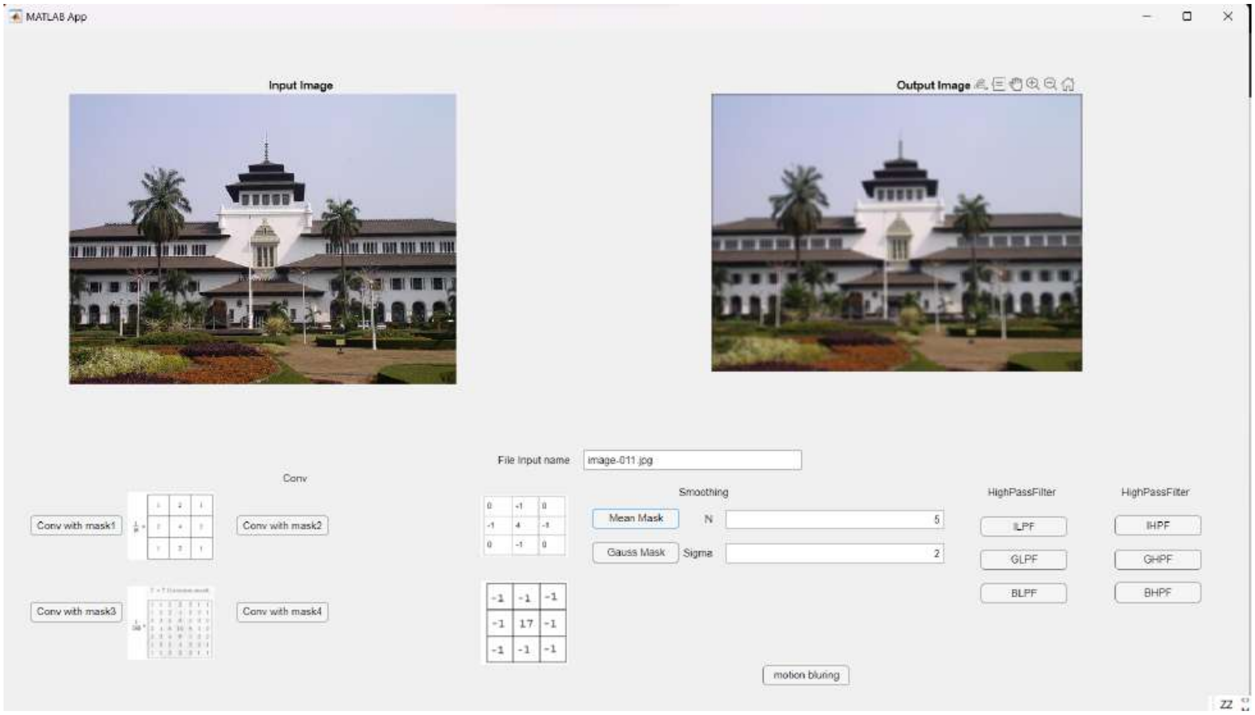
N: 5

Sigma: 2

1	2	1
2	4	2
1	2	1

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

-1	-1	-1
-1	17	-1
-1	-1	-1



File Input name: image-011.jpg

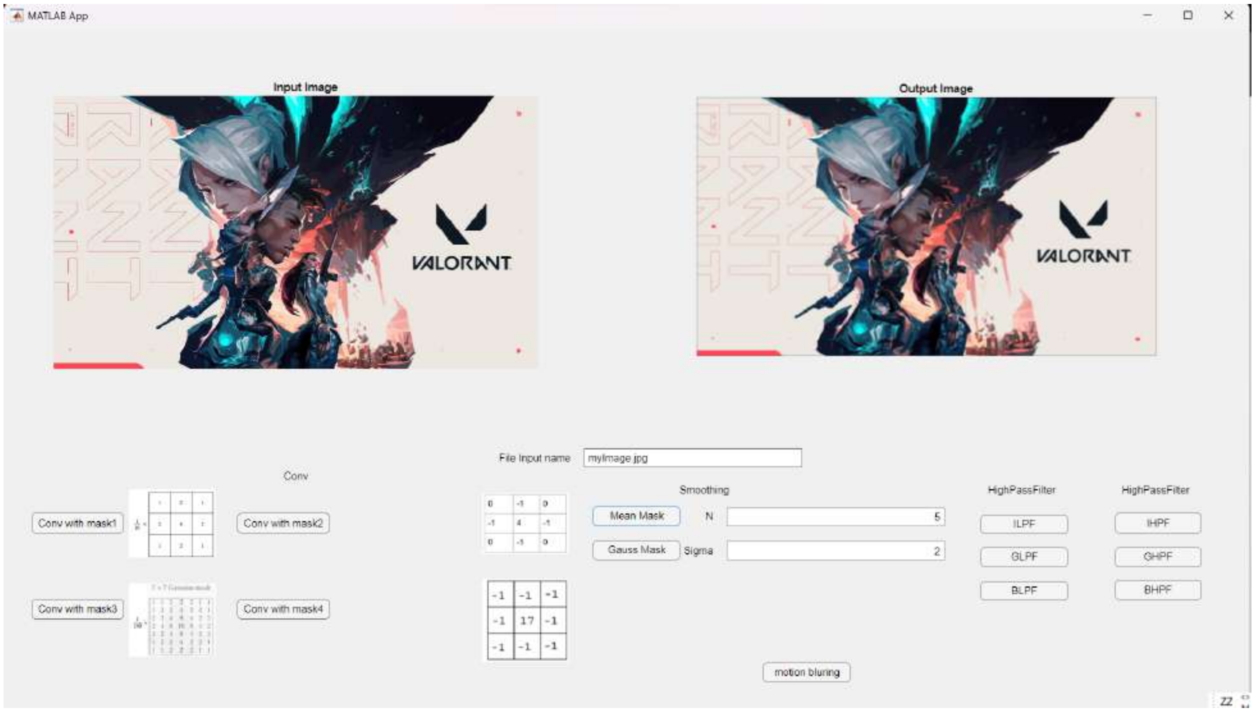
N: 5

Sigma: 2

1	2	1
2	4	2
1	2	1

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

-1	-1	-1
-1	17	-1
-1	-1	-1



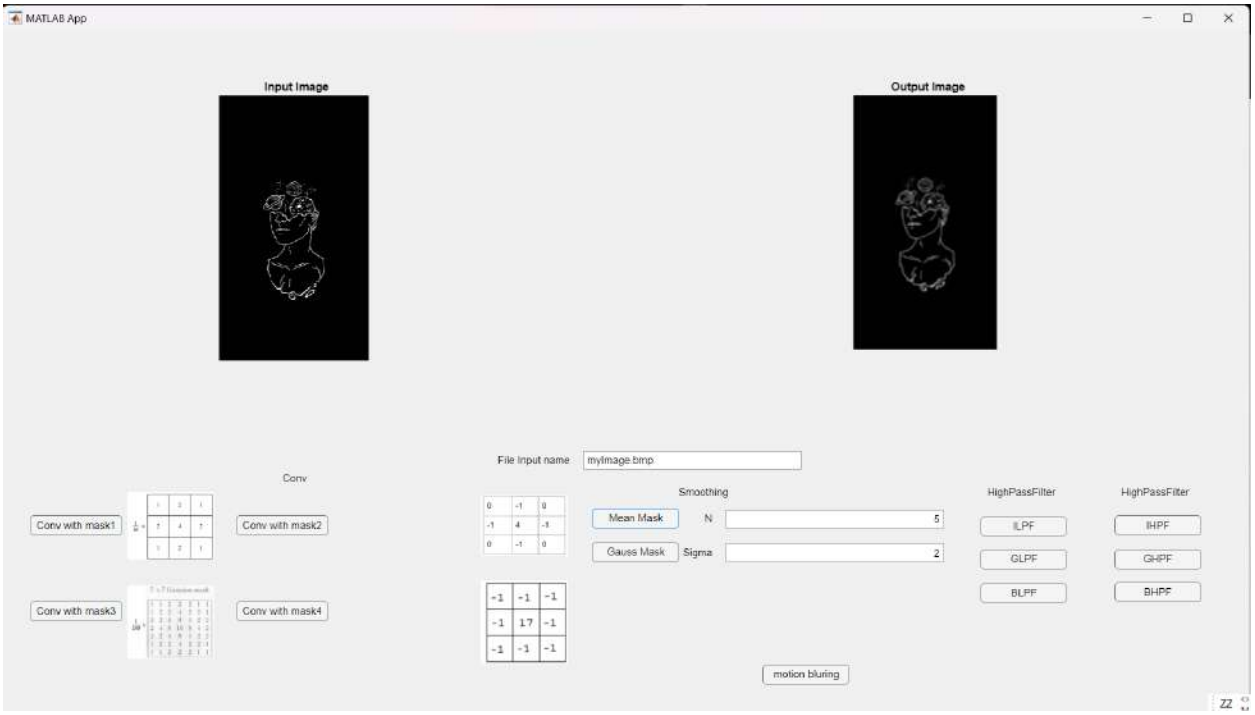
File Input name: myImage.jpg

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



File Input name: myImage.bmp

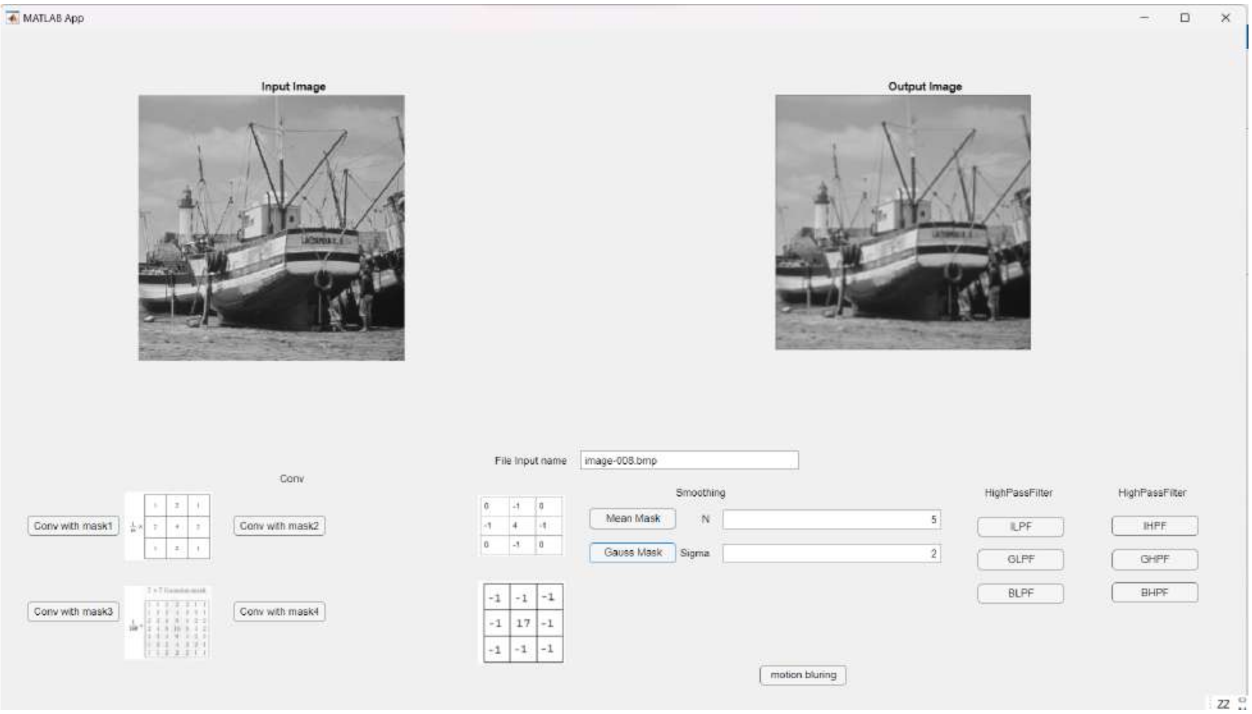
N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1

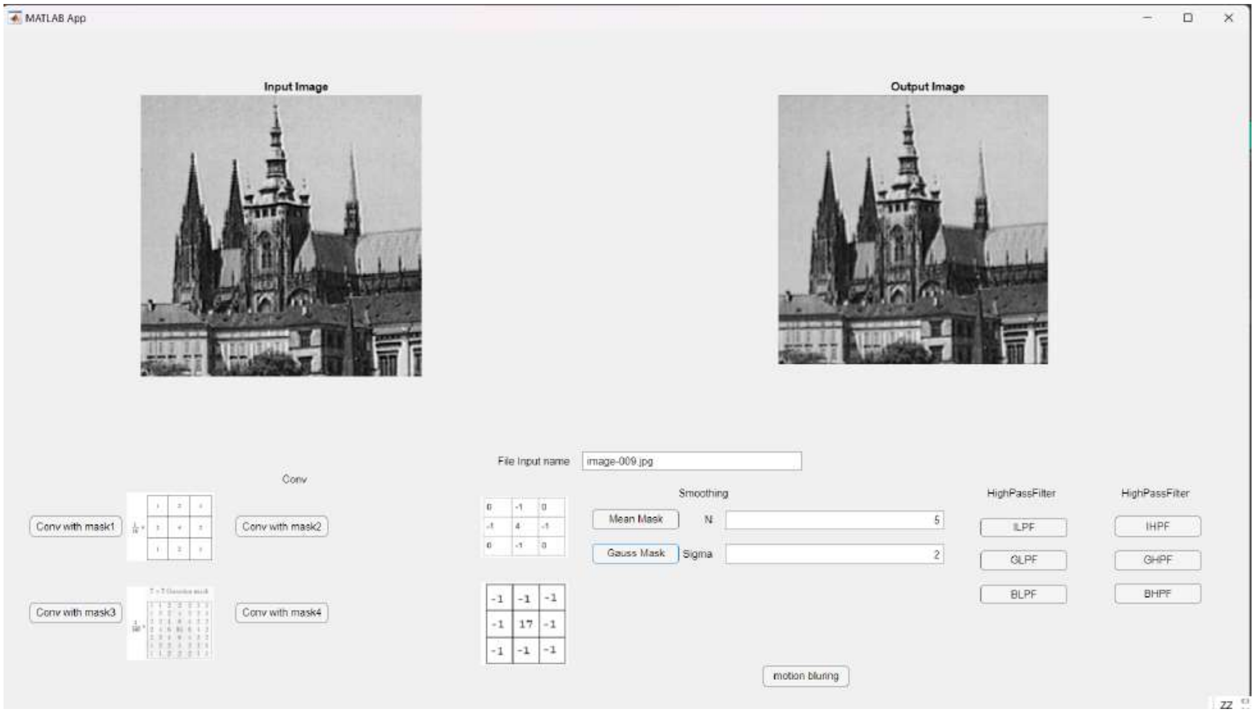
2. Gauss Filter



File Input name: image-008.bmp

N: 5

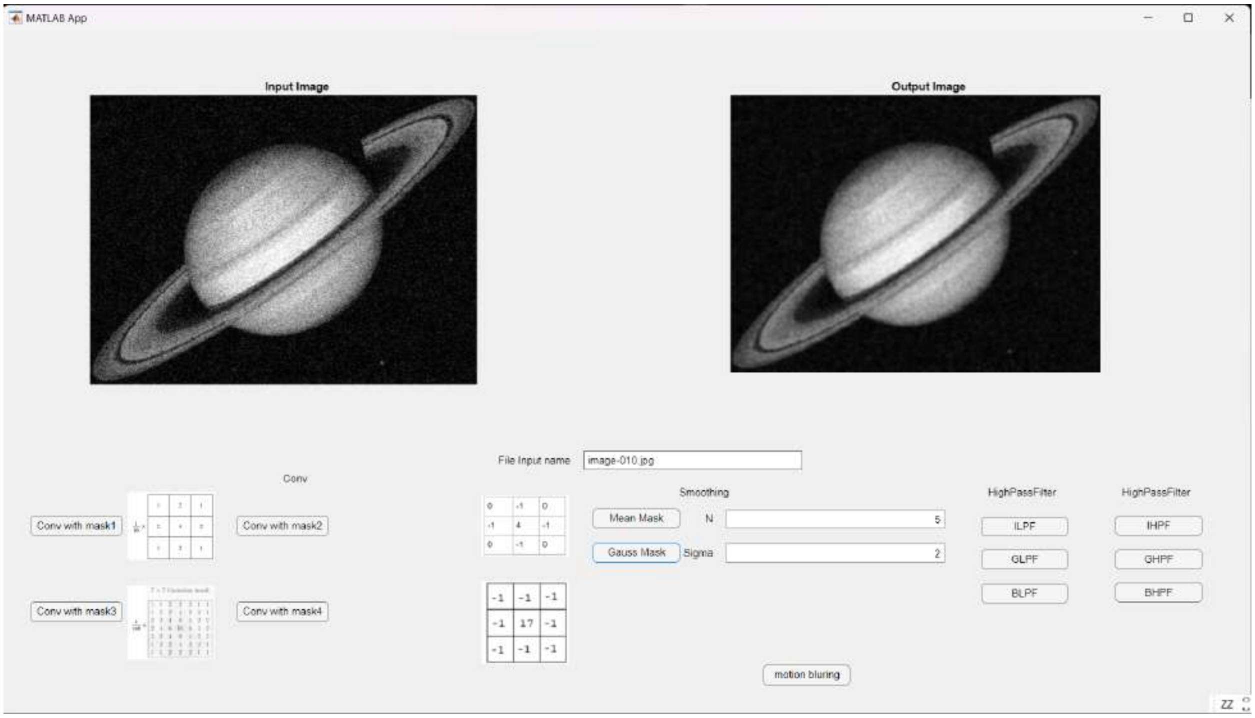
Sigma: 2



File Input name: image-009.jpg

N: 5

Sigma: 2



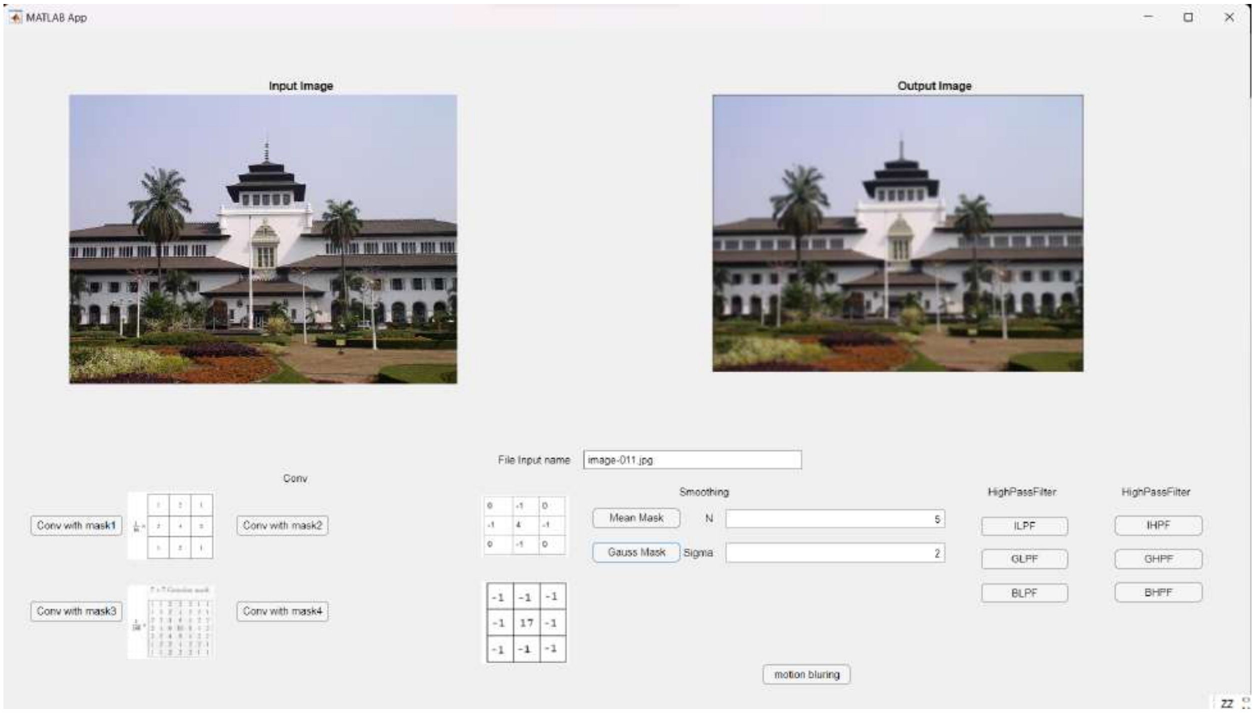
File Input name: image-010.jpg

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



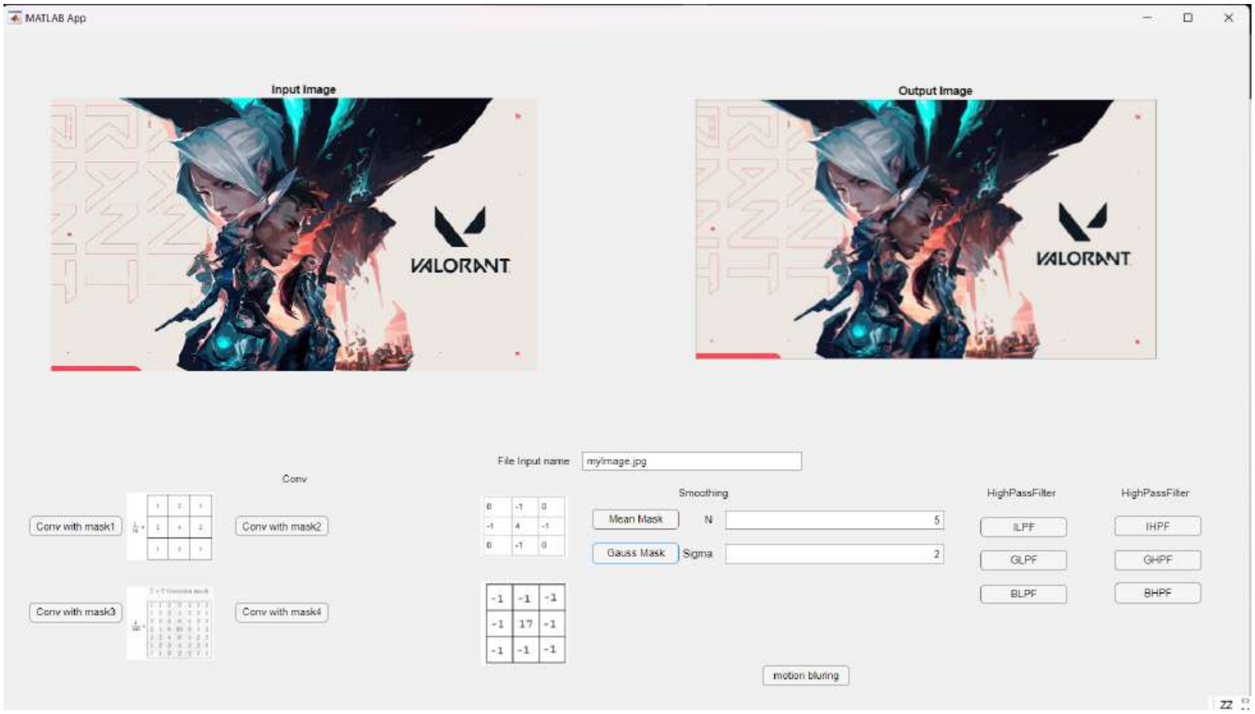
File Input name: image-011.jpg

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



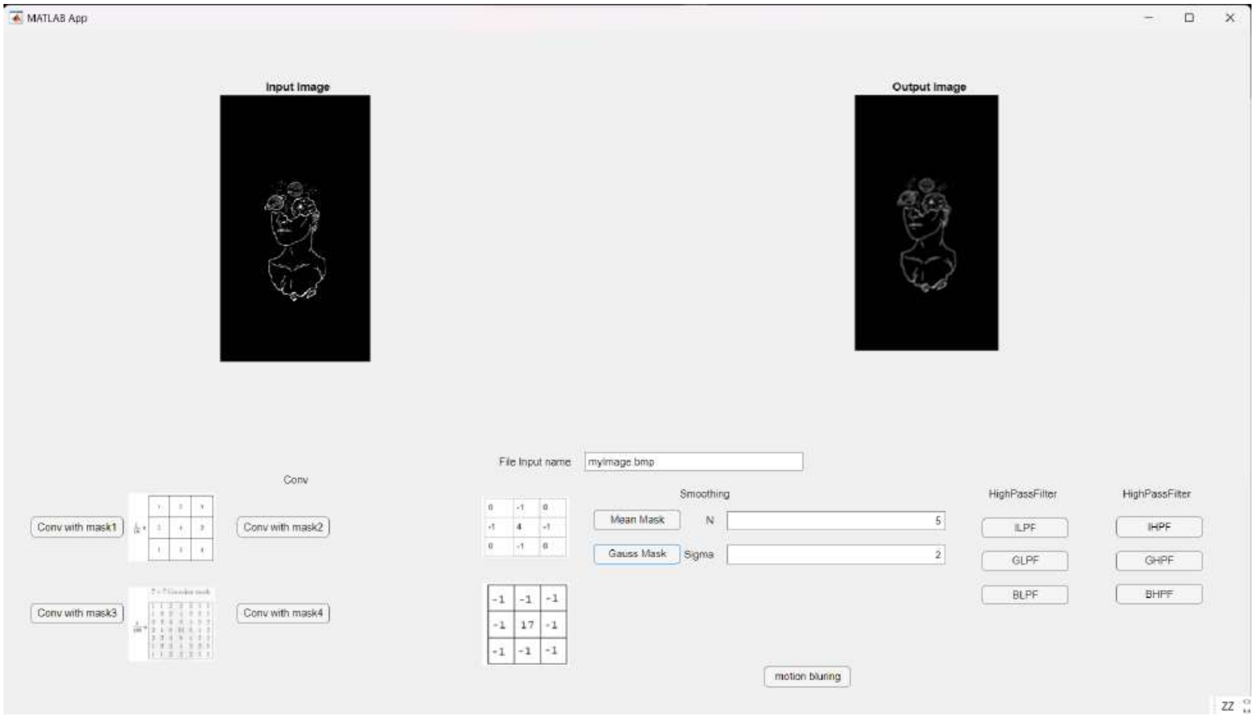
File Input name: myImage.jpg

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



File Input name: myImage.bmp

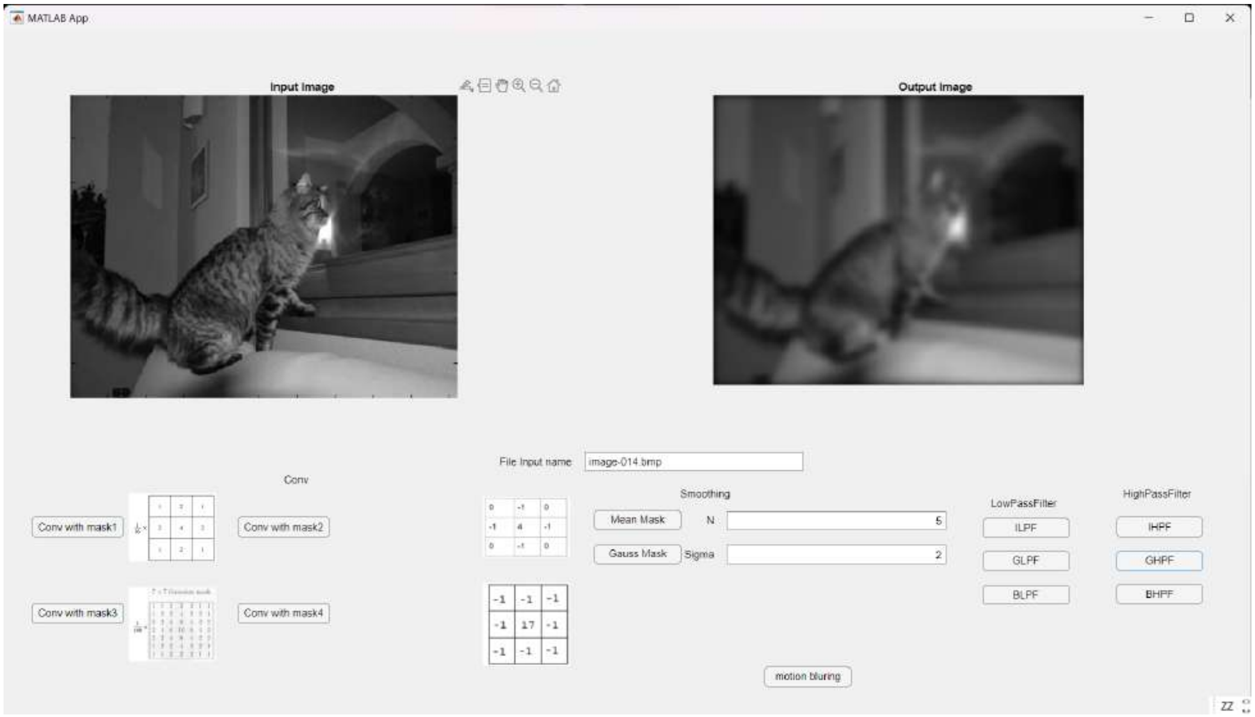
N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1

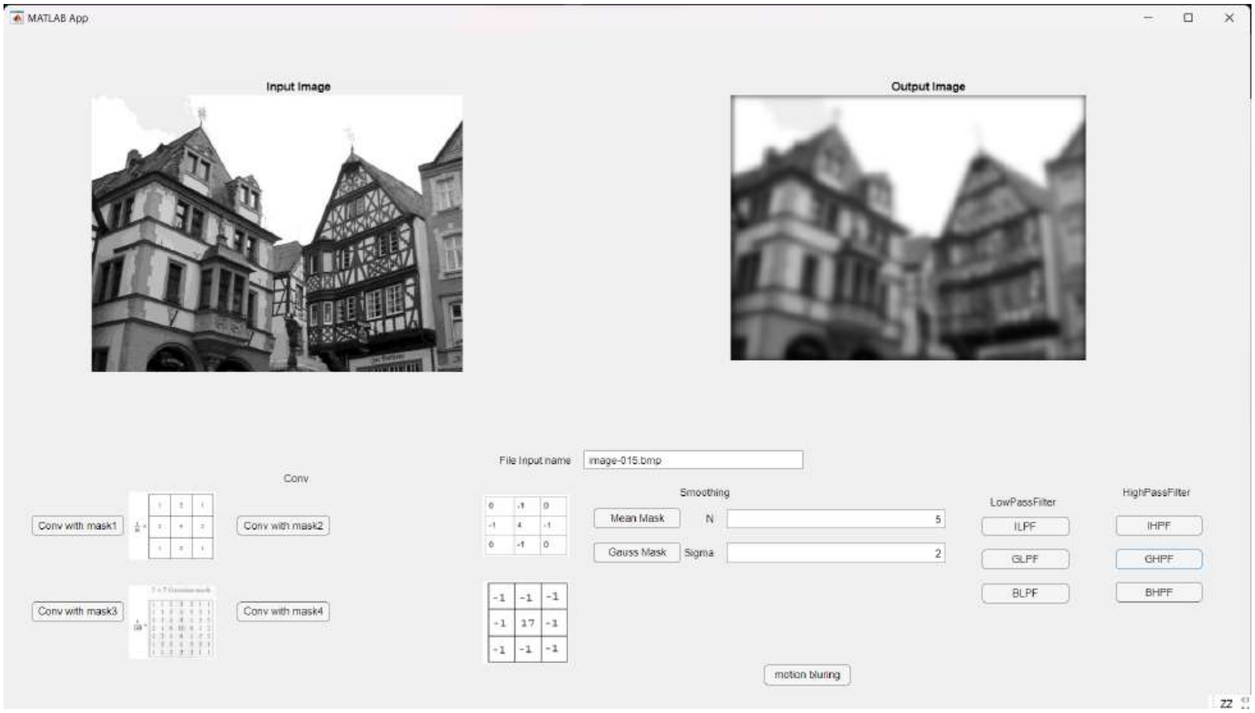
3. Low Pass Filter



File Input name: image-014.bmp

N: 5

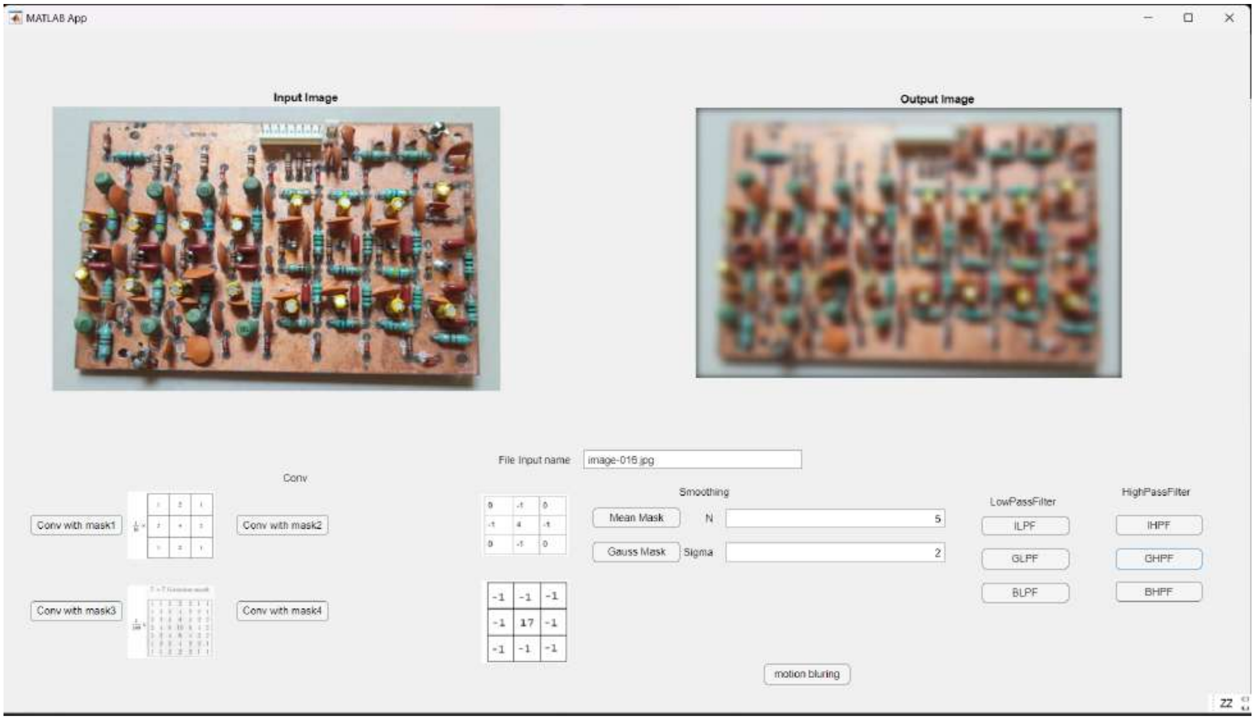
Sigma: 2



File Input name: image-015.bmp

N: 5

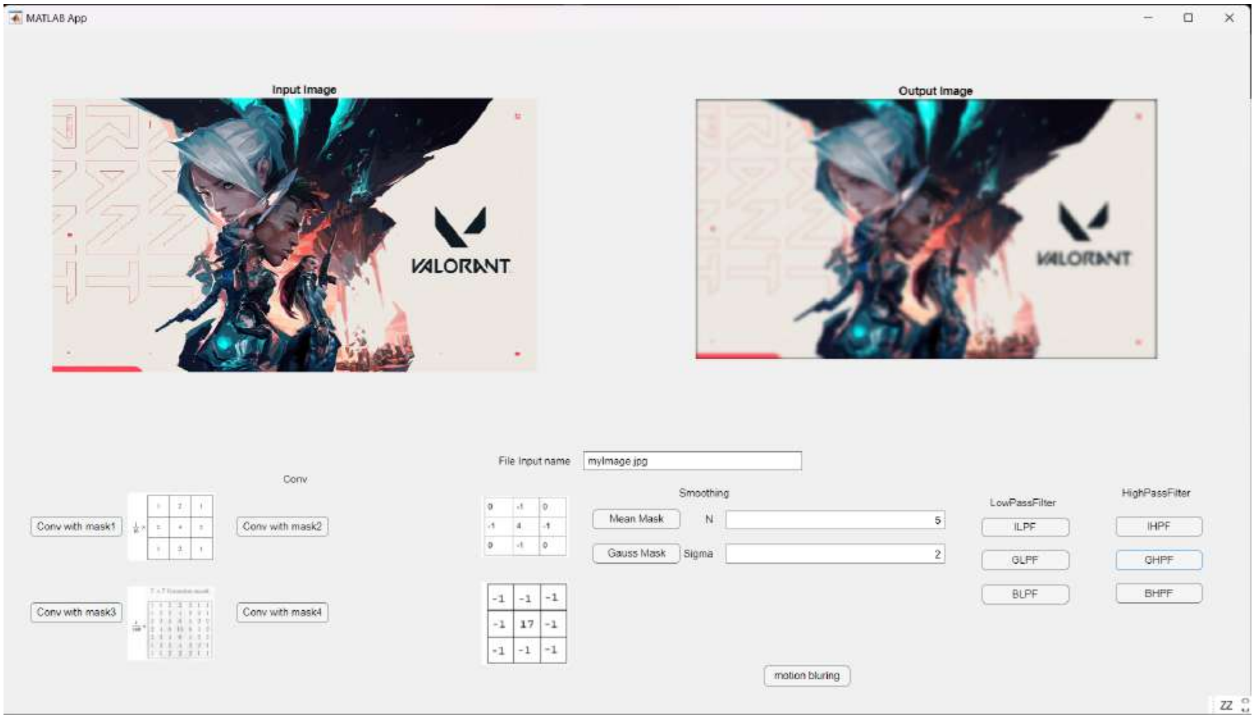
Sigma: 2



File Input name: image-016.jpg

N: 5

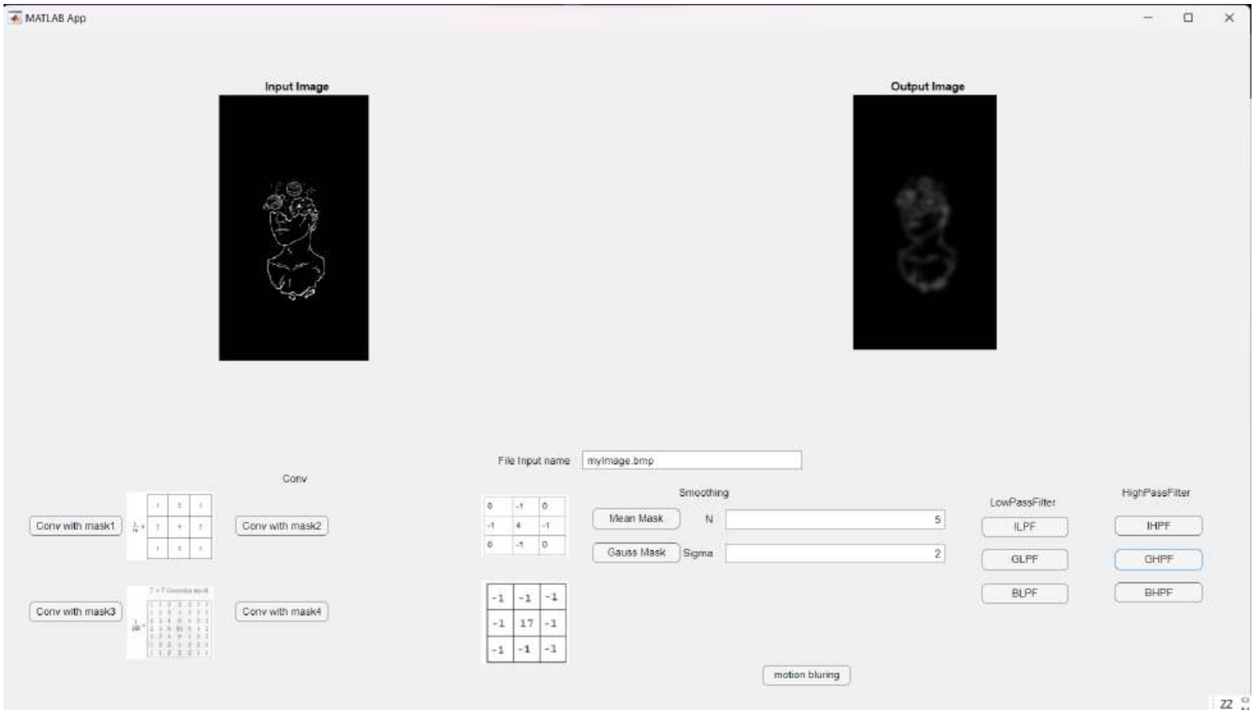
Sigma: 2



File Input name: myImage.jpg

N: 5

Sigma: 2



File Input name: myImage.bmp

N: 5

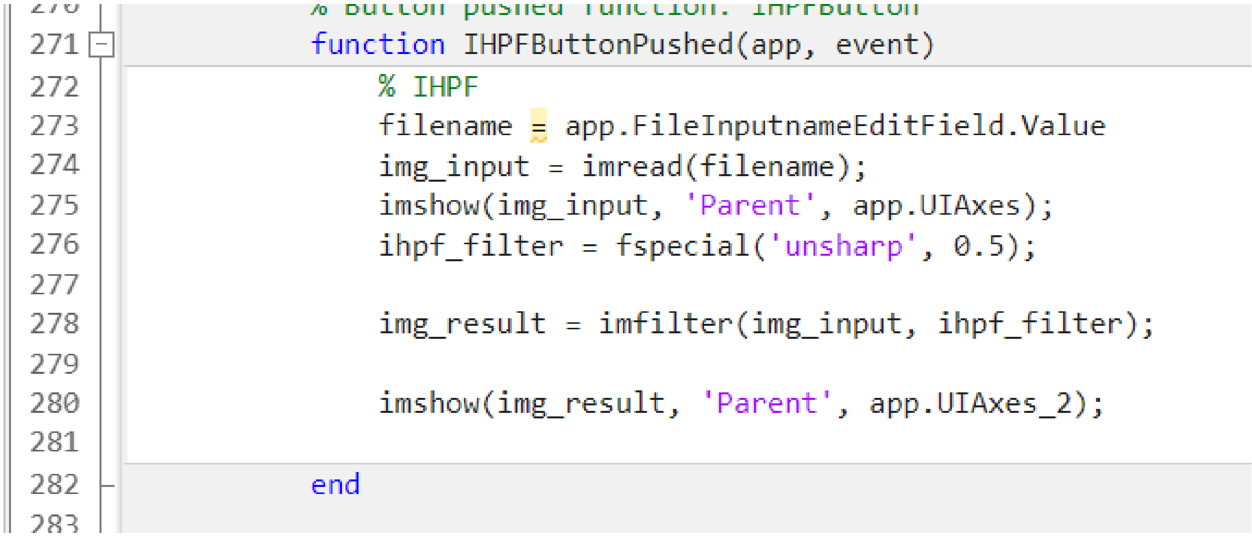
Sigma: 2

iii. analisa

Kedua Mean mask, Gauss mask, dan low pass filter memiliki efek blurring namun pada kernel gauss pixel tengah kenel yang juga pixel asli pada posisi tersebut memiliki efek yang lebih besar daripada pixel di sekitarnya. Karena efek blurring ini juga kernel ini dapat menghilangkan beberapa jenis noise

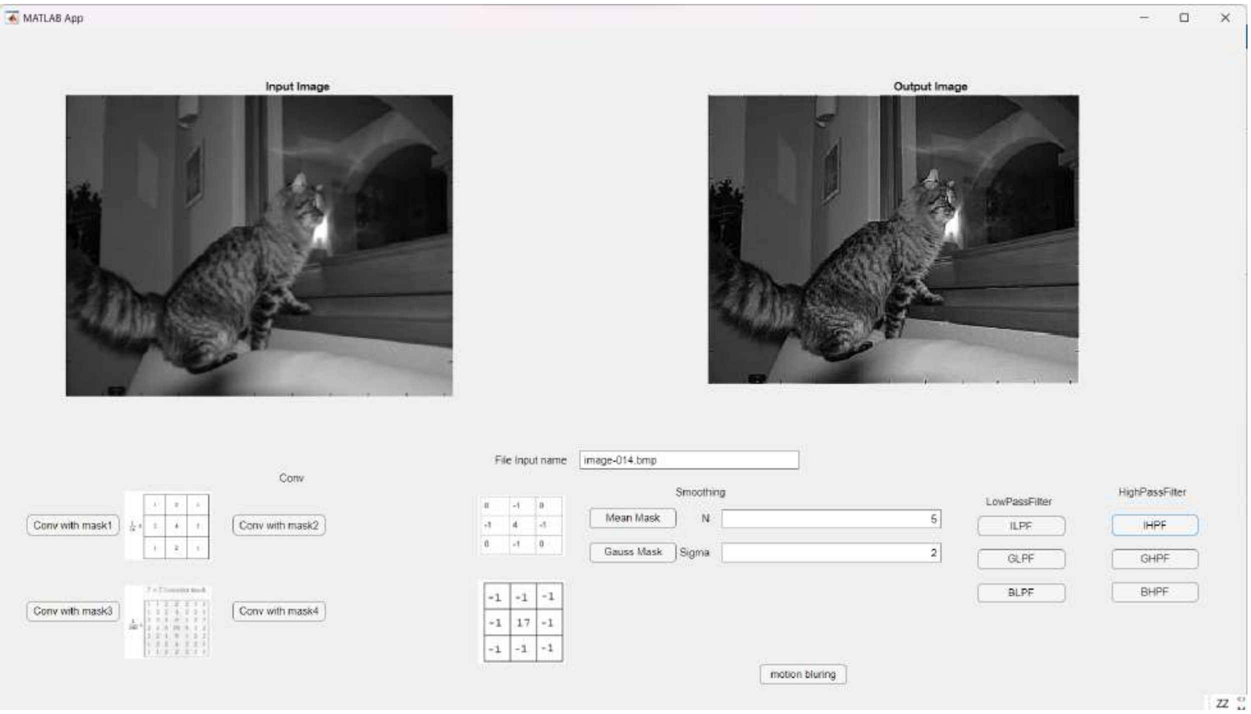
c. Penapisan citra dalam ranah frekuensi (high-pass filter IHPF, GHPF, dan BHPF)

i. Kode Program



```
        % Button pushed function: IHPFButton
        function IHPFButtonPushed(app, event)
            % IHPF
            filename = app.FileInputnameEditField.Value
            img_input = imread(filename);
            imshow(img_input, 'Parent', app.UIAxes);
            ihpf_filter = fspecial('unsharp', 0.5);

            img_result = imfilter(img_input, ihpf_filter);

            imshow(img_result, 'Parent', app.UIAxes_2);

        end

```

ii. Hasil Eksekusi



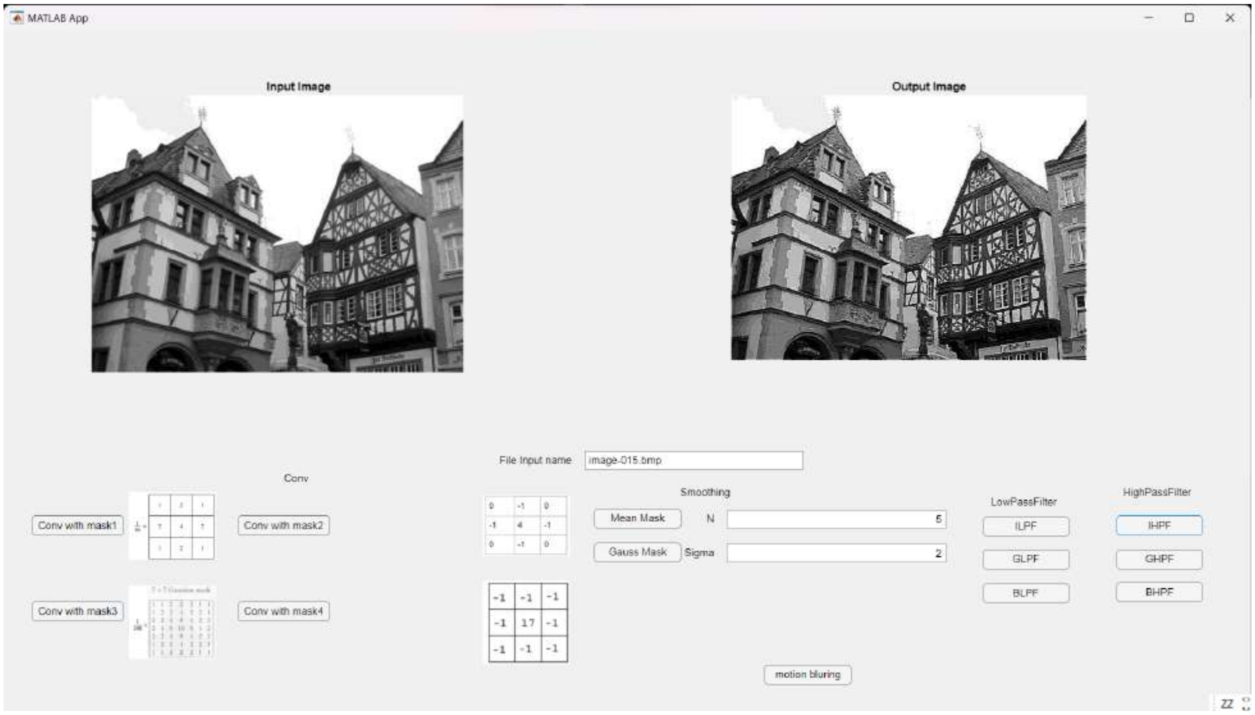
File Input name: image-014.bmp

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



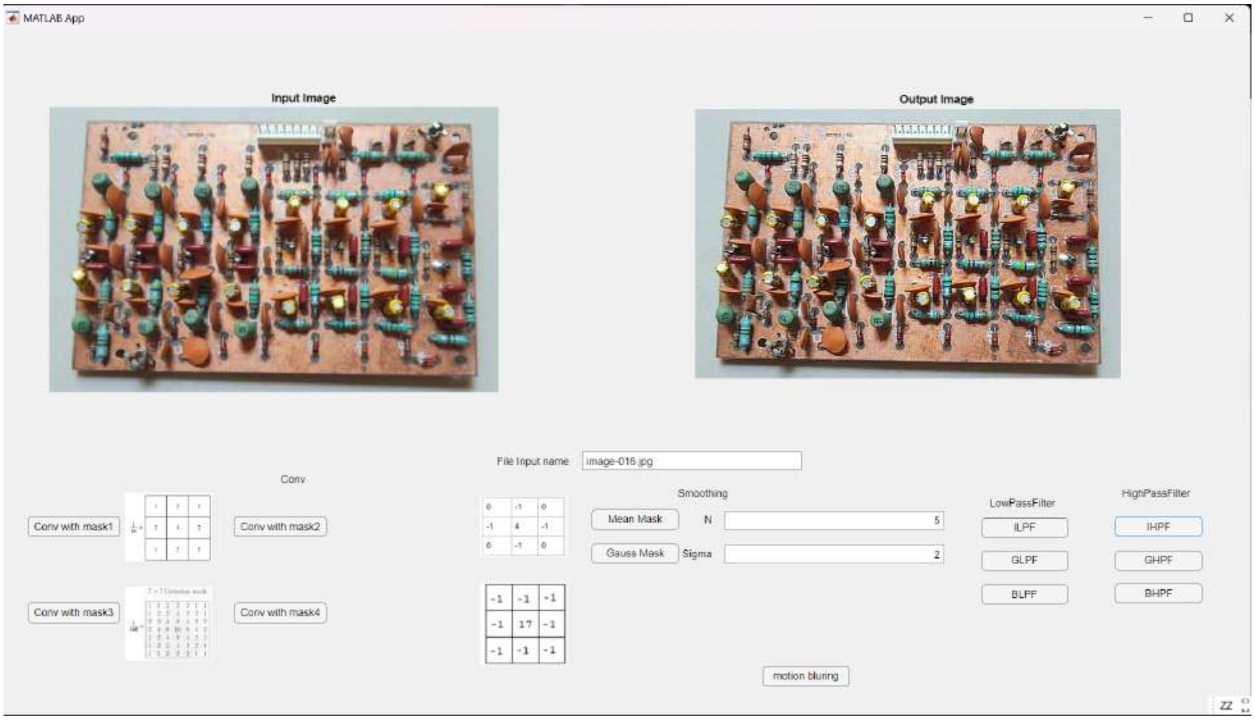
File Input name: image-015.bmp

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



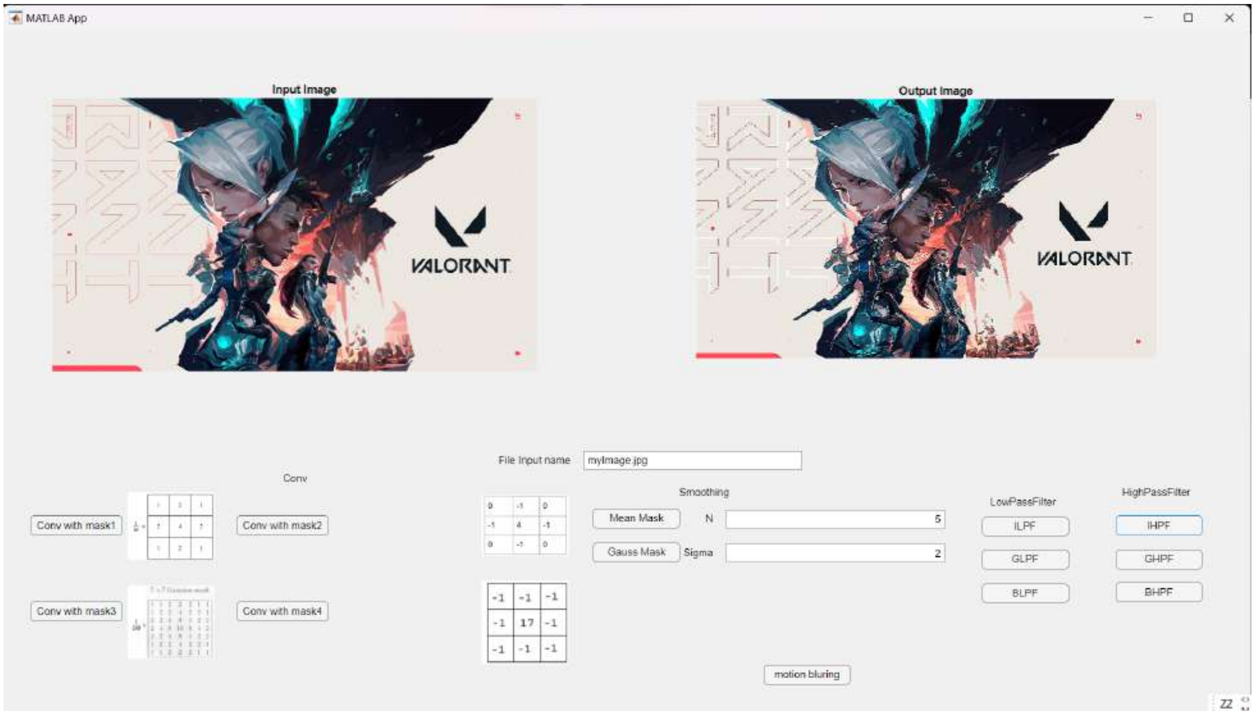
File Input name: image-016.jpg

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



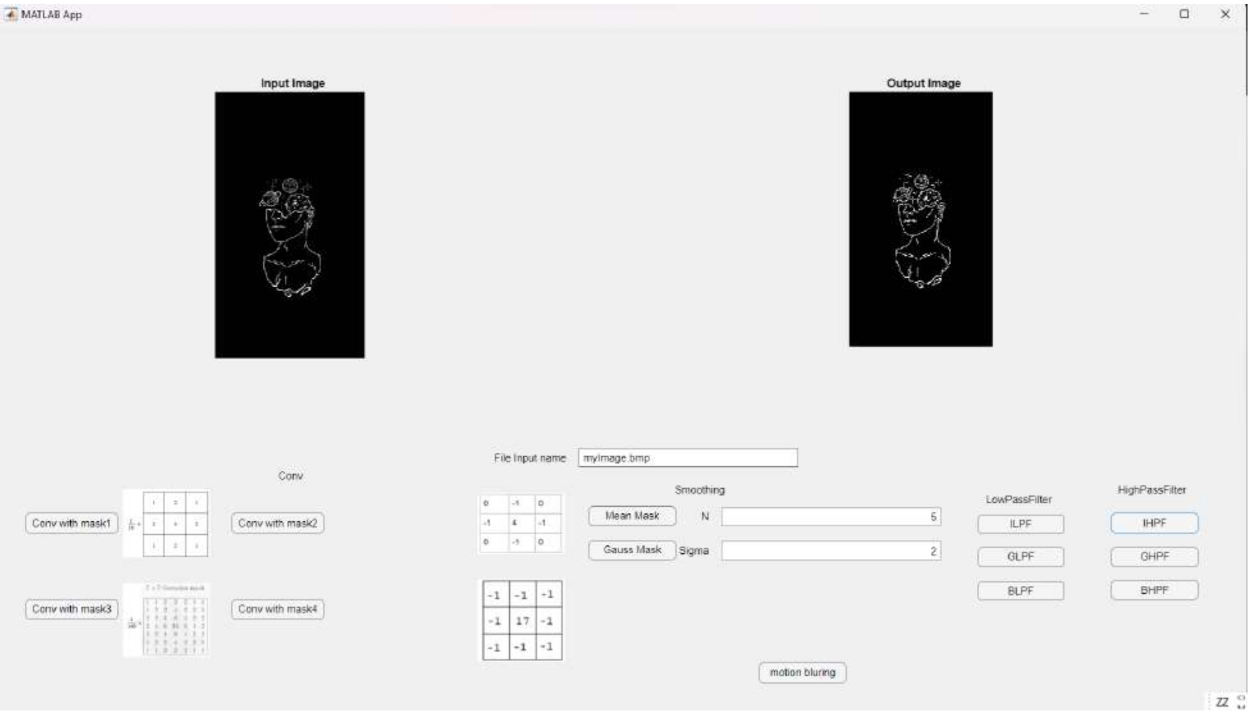
File Input name: myImage.jpg

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



File Input name: myImage.bmp

N: 5

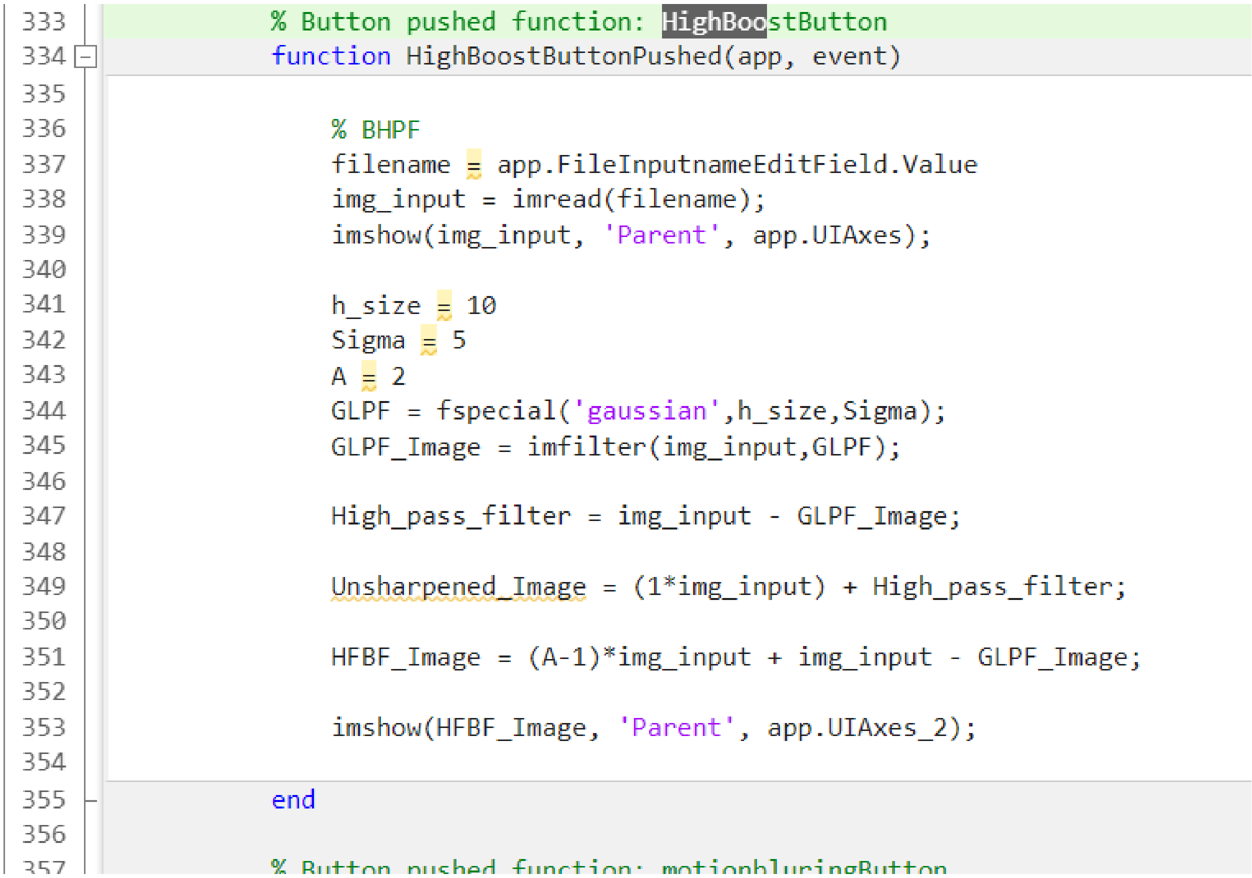
Sigma: 2

1. Analisa

   High Pass Filter (IHPF, GHPF, BHPF) memiliki efek edge/image sharpening. Dengan BHPF dengan threshold yang lebih gradual memiliki hasil yang paling bagus

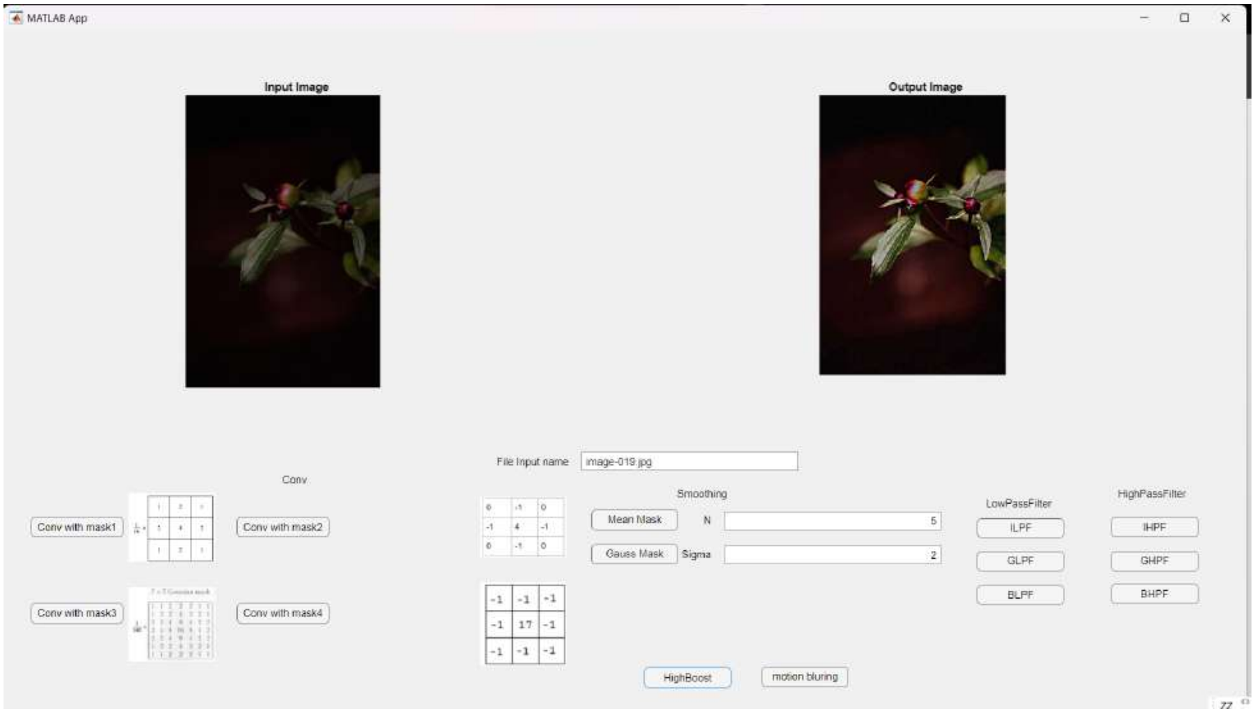
d. High Frequency Boost Filtering (Penapisan citra dalam ranah frekuensi untuk menghasilkan citra yang lebih terang)

   i. Kode Program



```
        % Button pushed function: HighBoostButton
        function HighBoostButtonPushed(app, event)

            % BHPF
            filename = app.FileInputnameEditField.Value
            img_input = imread(filename);
            imshow(img_input, 'Parent', app.UIAxes);

            h_size = 10
            Sigma = 5
            A = 2
            GLPF = fspecial('gaussian',h_size,Sigma);
            GLPF_Image = imfilter(img_input,GLPF);

            High_pass_filter = img_input - GLPF_Image;

            Unsharpened_Image = (1*img_input) + High_pass_filter;

            HFBF_Image = (A-1)*img_input + img_input - GLPF_Image;

            imshow(HFBF_Image, 'Parent', app.UIAxes_2);

        end

        % Button pushed function: motionbluringButton
```

ii. Hasil Eksekusi



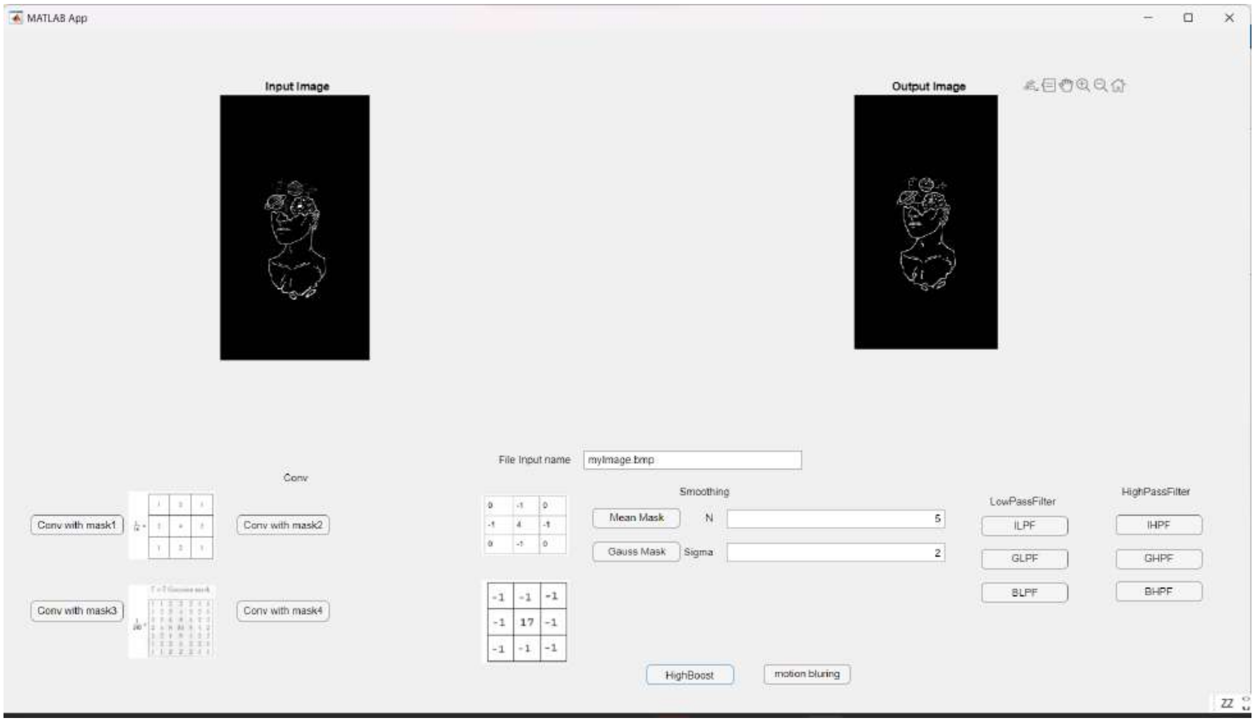
File Input name: image-019.jpg

N: 5

Sigma: 2

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

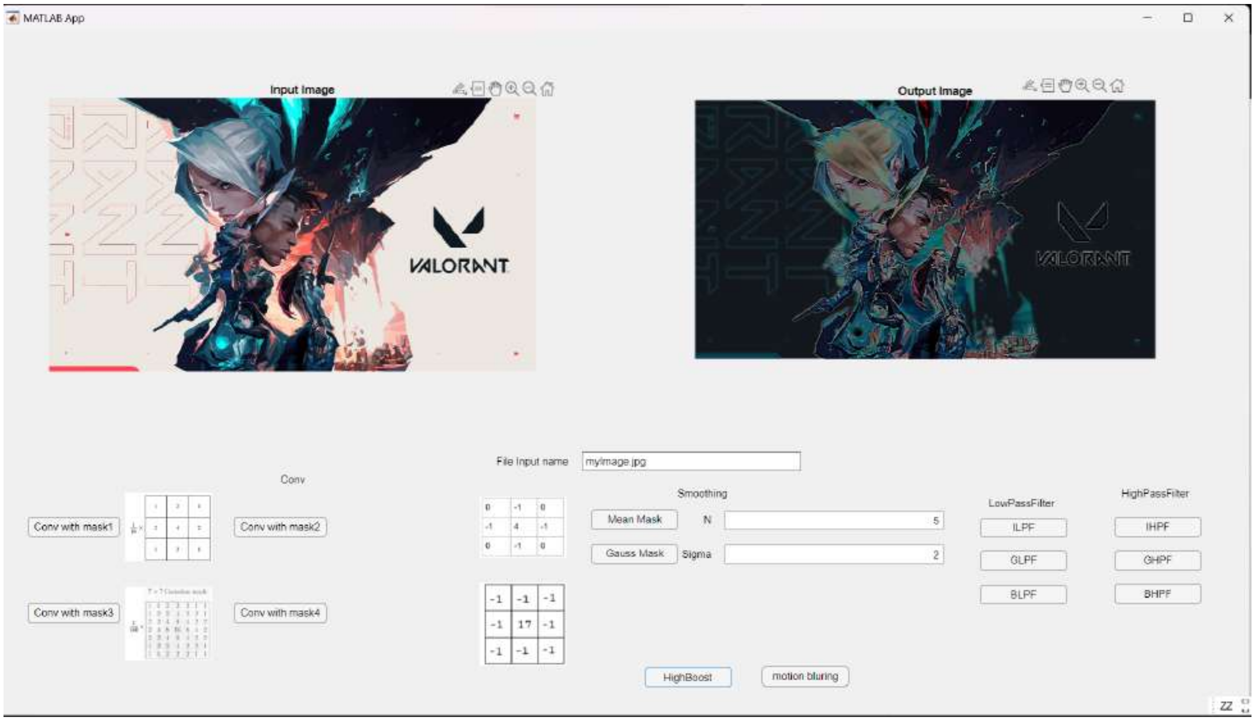
-1	-1	-1
-1	17	-1
-1	-1	-1



File Input name: myImage.bmp

N: 5

Sigma: 2



File Input name: myImage.jpg

N: 5

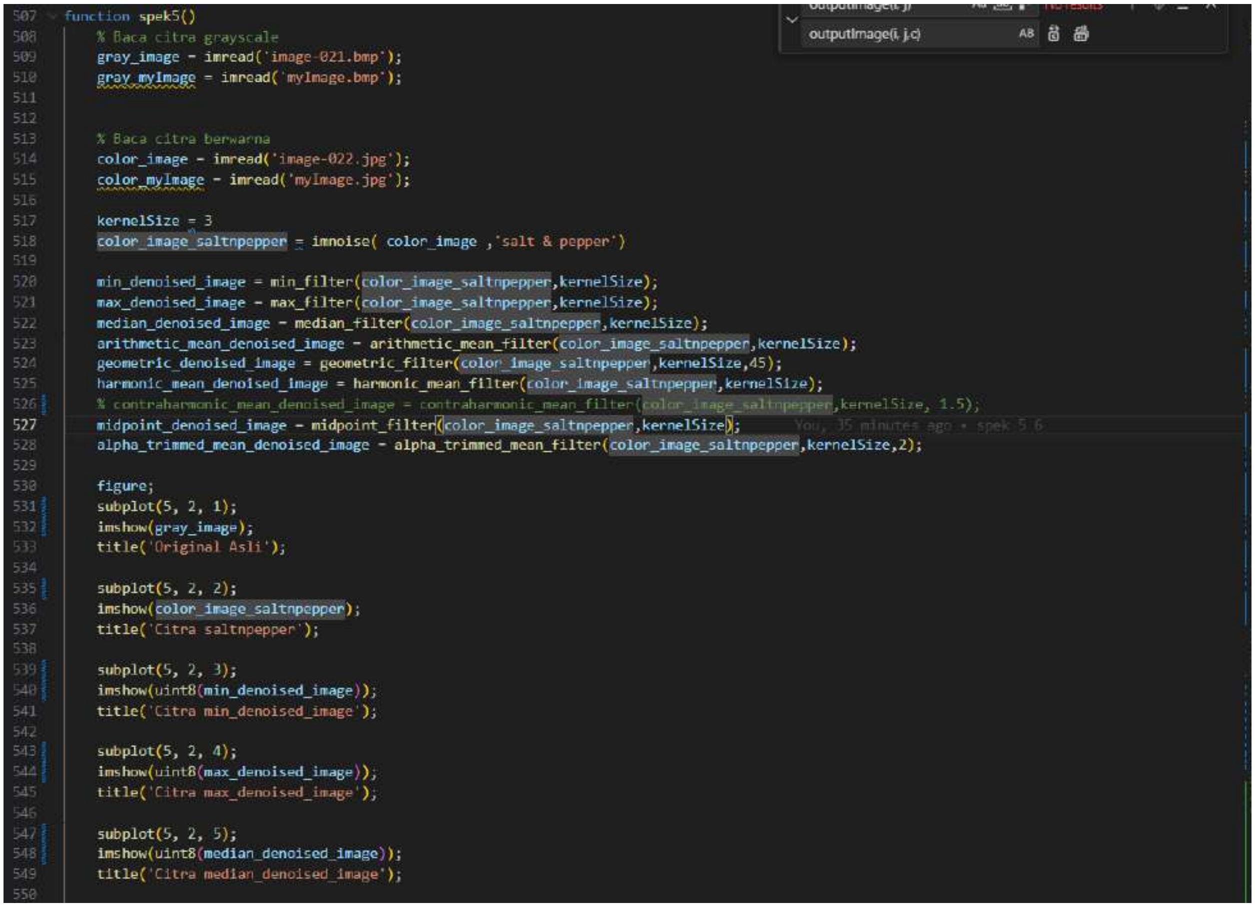
Sigma: 2

iii. Analisa

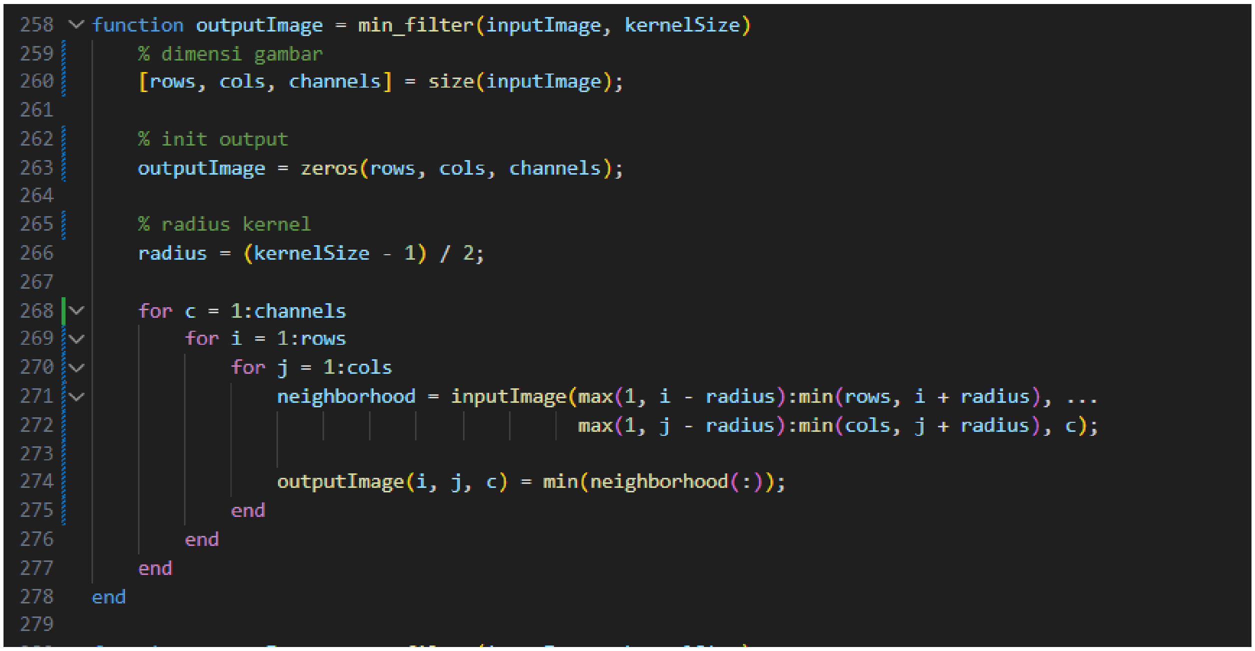
Yang dilakukan pada gambar pada prinsipnya adalah High Frequency Boost Filtering(HFBF) yang memperkuat pixel - pixel dengan frequensi tinggi tanpa mengubah pixel - pixel dengan frekuensi rendah

e. Penghilangan Derau Salt and Pepper

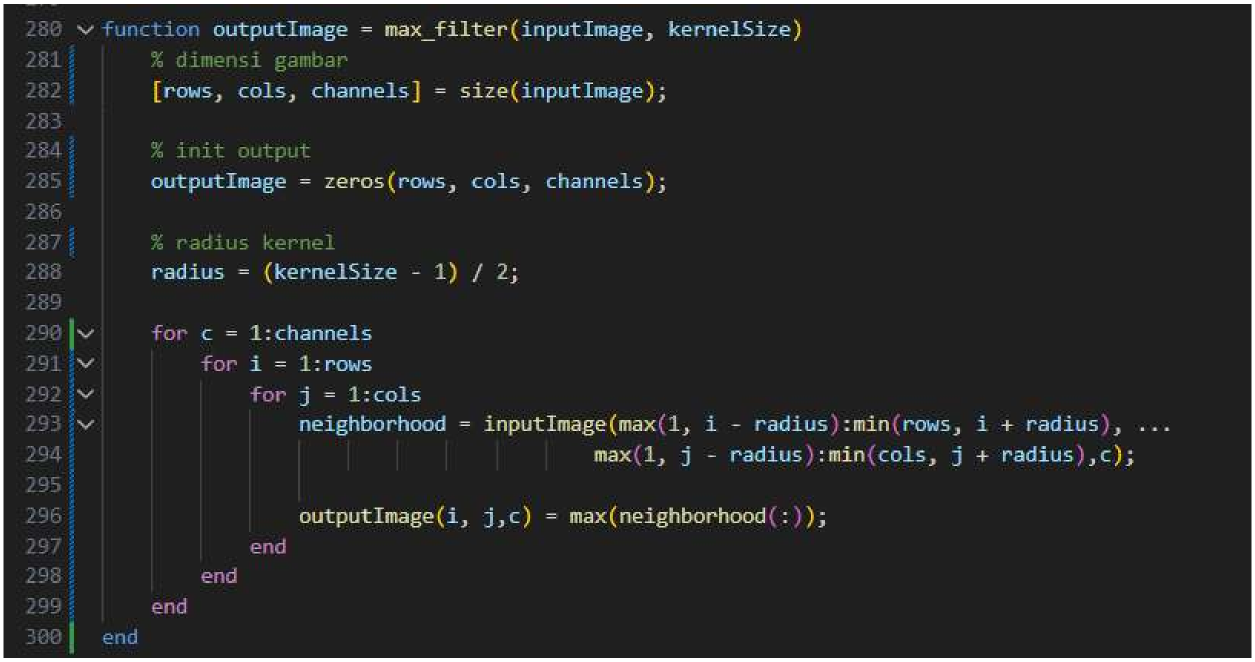
i. Kode Program



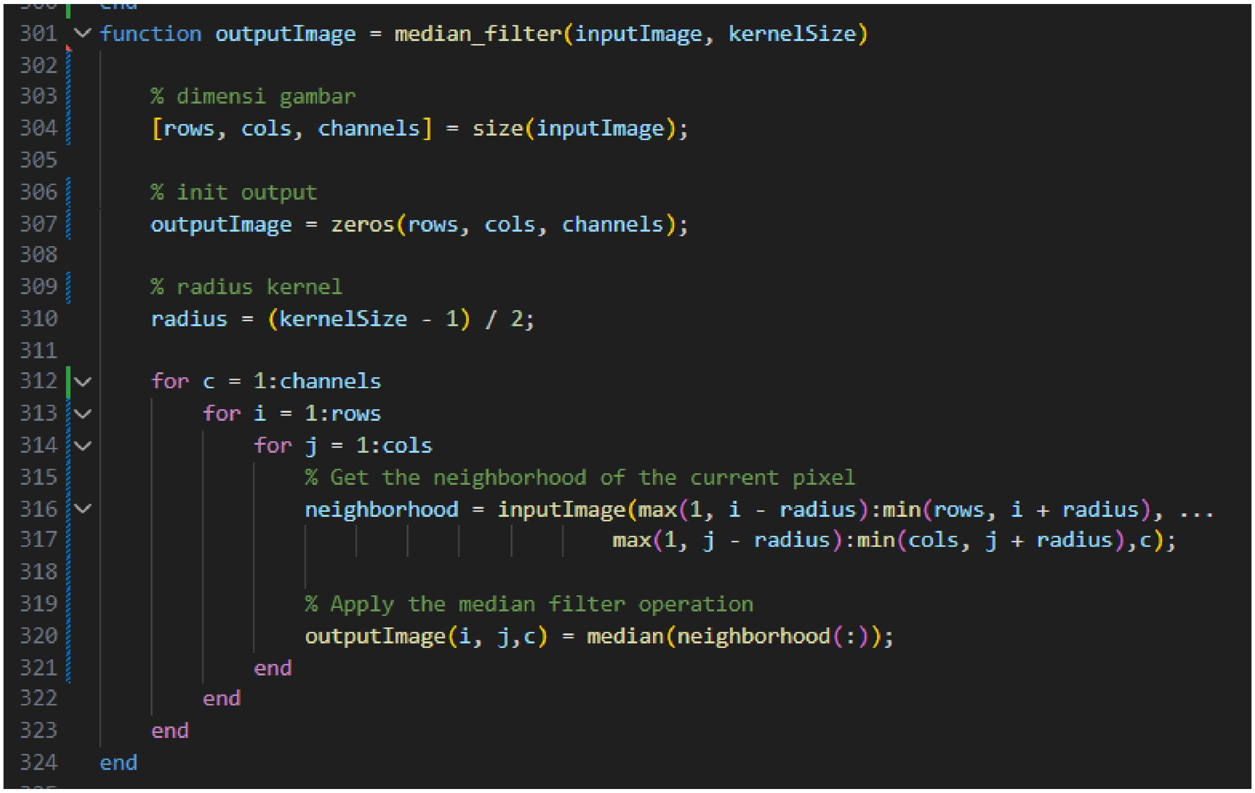
```
function spek5()
    % Baca citra grayscale
    gray_image = imread('image-021.bmp');
    gray_myImage = imread('myImage.bmp');


    % Baca citra berwarna
    color_image = imread('image-022.jpg');
    color_myImage = imread('myImage.jpg');

    kernelSize = 3
    color_image_saltnpepper = imnoise( color_image ,'salt & pepper')

    min_denoised_image = min_filter(color_image_saltnpepper,kernelSize);
    max_denoised_image = max_filter(color_image_saltnpepper,kernelSize);
    median_denoised_image = median_filter(color_image_saltnpepper,kernelSize);
    arithmetic_mean_denoised_image = arithmetic_mean_filter(color_image_saltnpepper,kernelSize);
    geometric_denoised_image = geometric_filter(color_image_saltnpepper,kernelSize,45);
    harmonic_mean_denoised_image = harmonic_mean_filter(color_image_saltnpepper,kernelSize);
    % contraharmonic_mean_denoised_image = contraharmonic_mean_filter(color_image_saltnpepper,kernelSize, 1.5);
    midpoint_denoised_image = midpoint_filter(color_image_saltnpepper,kernelSize);
    alpha_trimmed_mean_denoised_image = alpha_trimmed_mean_filter(color_image_saltnpepper,kernelSize,2);

    figure;
    subplot(5, 2, 1);
    imshow(gray_image);
    title('Original Asli');

    subplot(5, 2, 2);
    imshow(color_image_saltnpepper);
    title('Citra saltnpepper');

    subplot(5, 2, 3);
    imshow(uint8(min_denoised_image));
    title('Citra min_denoised_image');

    subplot(5, 2, 4);
    imshow(uint8(max_denoised_image));
    title('Citra max_denoised_image');

    subplot(5, 2, 5);
    imshow(uint8(median_denoised_image));
    title('Citra median_denoised_image');

```



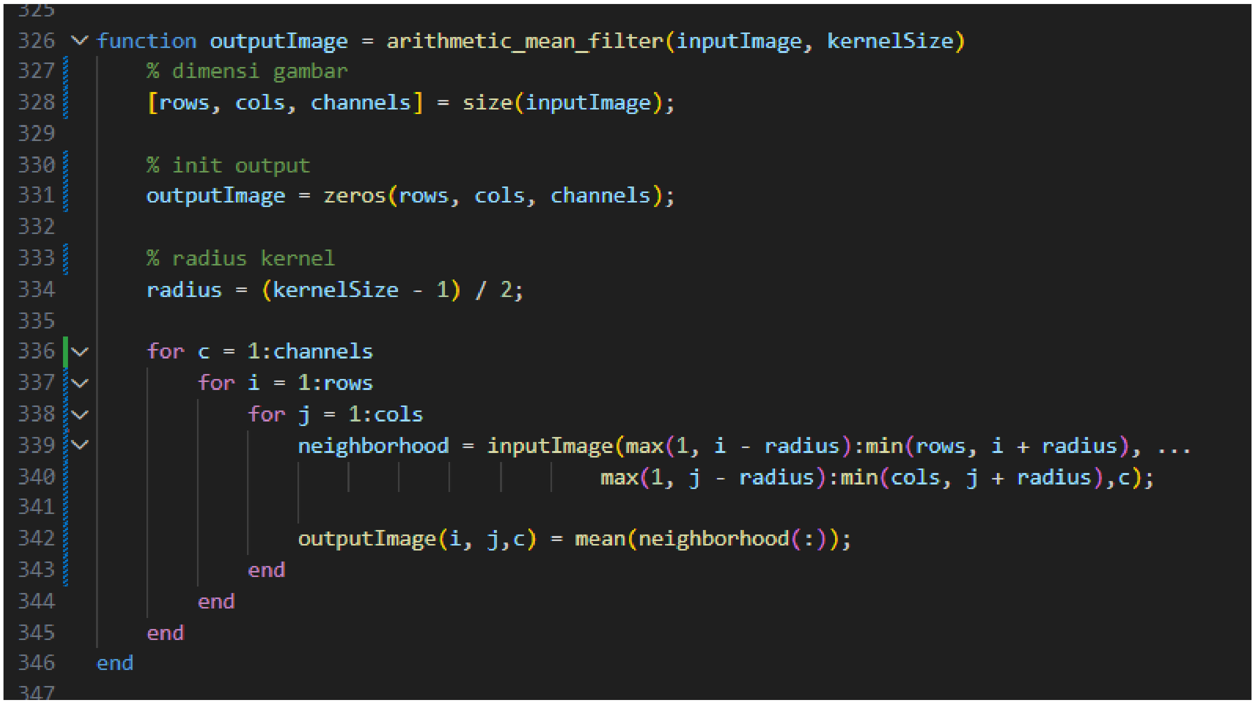
```
function outputImage = min_filter(inputImage, kernelSize)
    % dimensi gambar
    [rows, cols, channels] = size(inputImage);

    % init output
    outputImage = zeros(rows, cols, channels);

    % radius kernel
    radius = (kernelSize - 1) / 2;

    for c = 1:channels
        for i = 1:rows
            for j = 1:cols
                neighborhood = inputImage(max(1, i - radius):min(rows, i + radius), ...
                                          max(1, j - radius):min(cols, j + radius), c);

                outputImage(i, j, c) = min(neighborhood(:));
            end
        end
    end
end

```



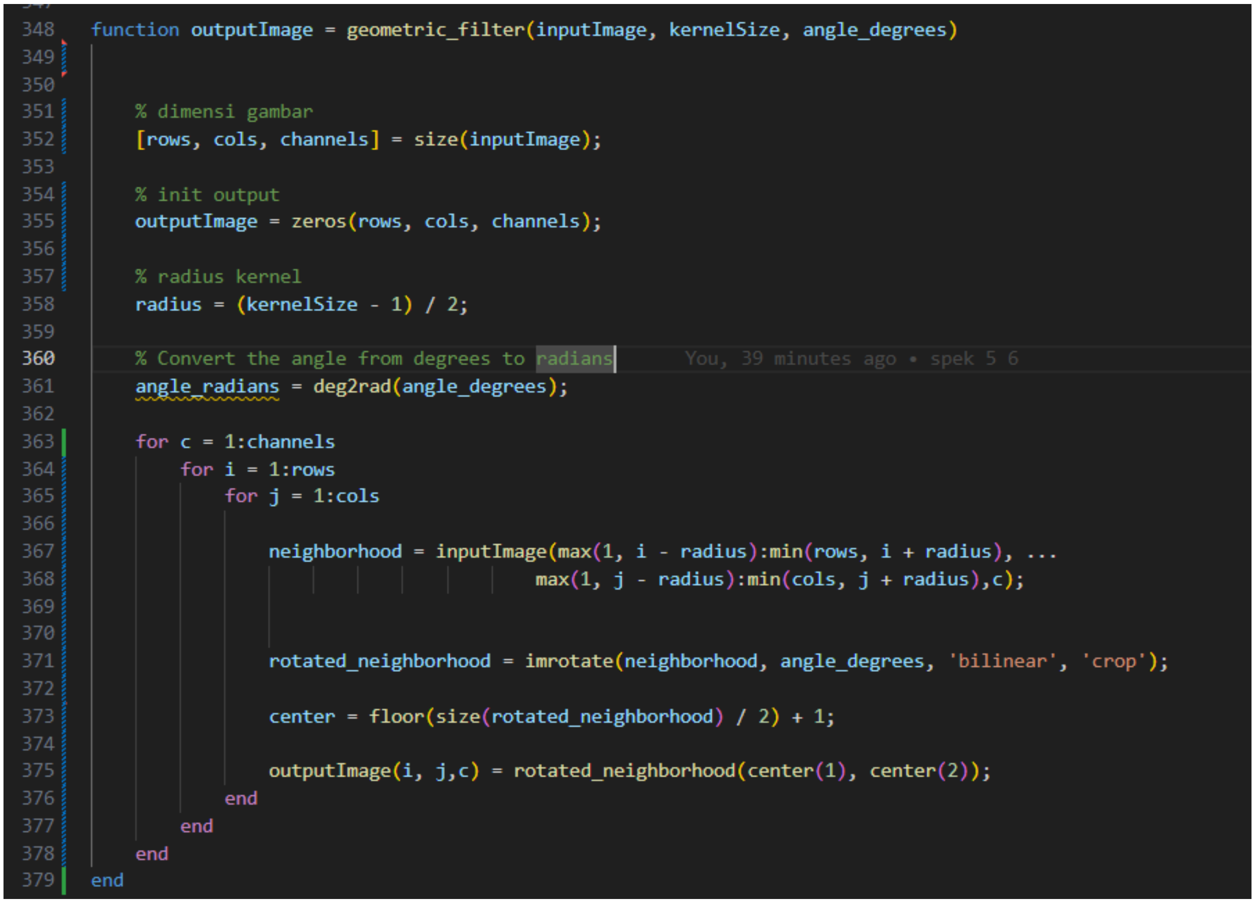
```matlab
function outputImage = max_filter(inputImage, kernelSize)
    % dimensi gambar
    [rows, cols, channels] = size(inputImage);

    % init output
    outputImage = zeros(rows, cols, channels);

    % radius kernel
    radius = (kernelSize - 1) / 2;

    for c = 1:channels
        for i = 1:rows
            for j = 1:cols
                neighborhood = inputImage(max(1, i - radius):min(rows, i + radius), ...
                                        max(1, j - radius):min(cols, j + radius),c);

                outputImage(i, j,c) = max(neighborhood(:));
            end
        end
    end
end
```



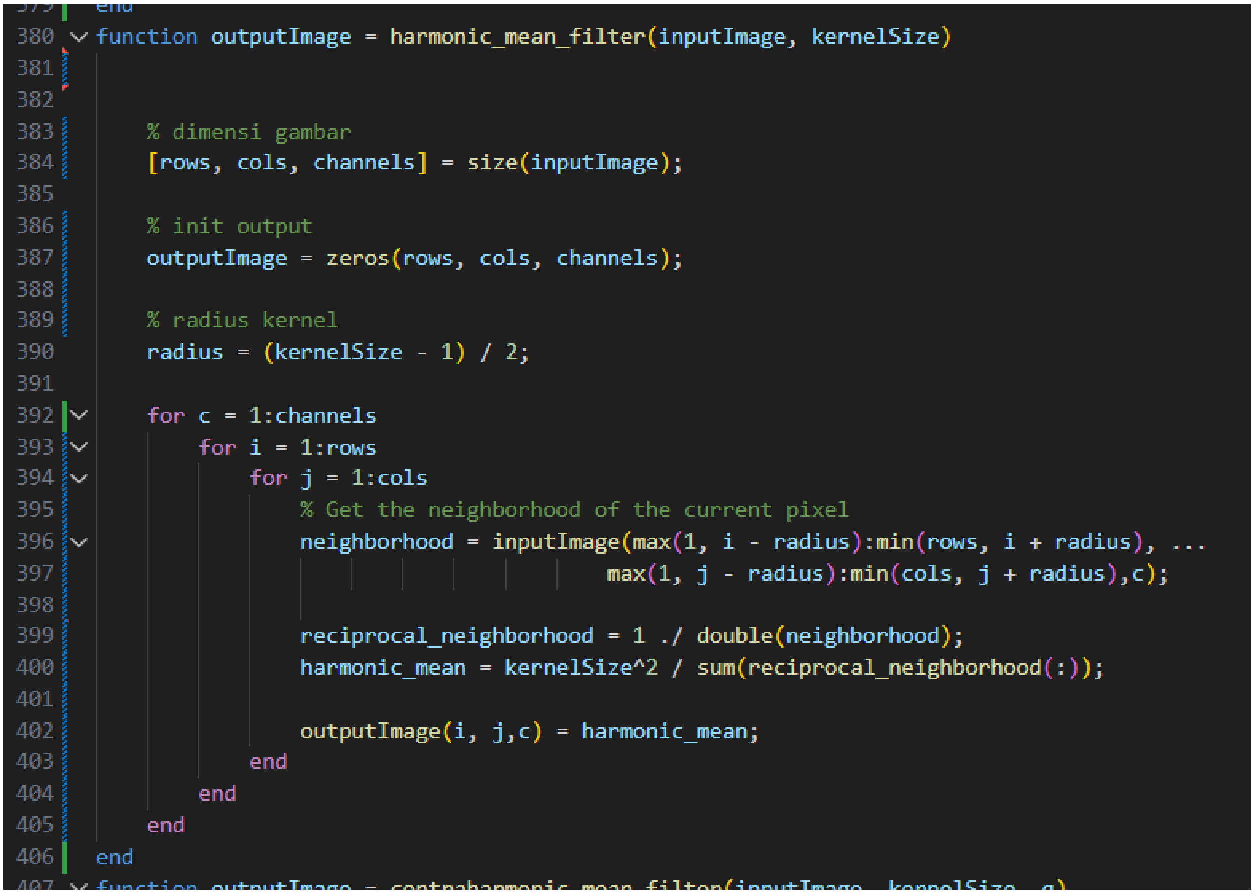
```matlab
function outputImage = median_filter(inputImage, kernelSize)

    % dimensi gambar
    [rows, cols, channels] = size(inputImage);

    % init output
    outputImage = zeros(rows, cols, channels);

    % radius kernel
    radius = (kernelSize - 1) / 2;

    for c = 1:channels
        for i = 1:rows
            for j = 1:cols
                % Get the neighborhood of the current pixel
                neighborhood = inputImage(max(1, i - radius):min(rows, i + radius), ...
                                        max(1, j - radius):min(cols, j + radius),c);

                % Apply the median filter operation
                outputImage(i, j,c) = median(neighborhood(:));
            end
        end
    end
end
```



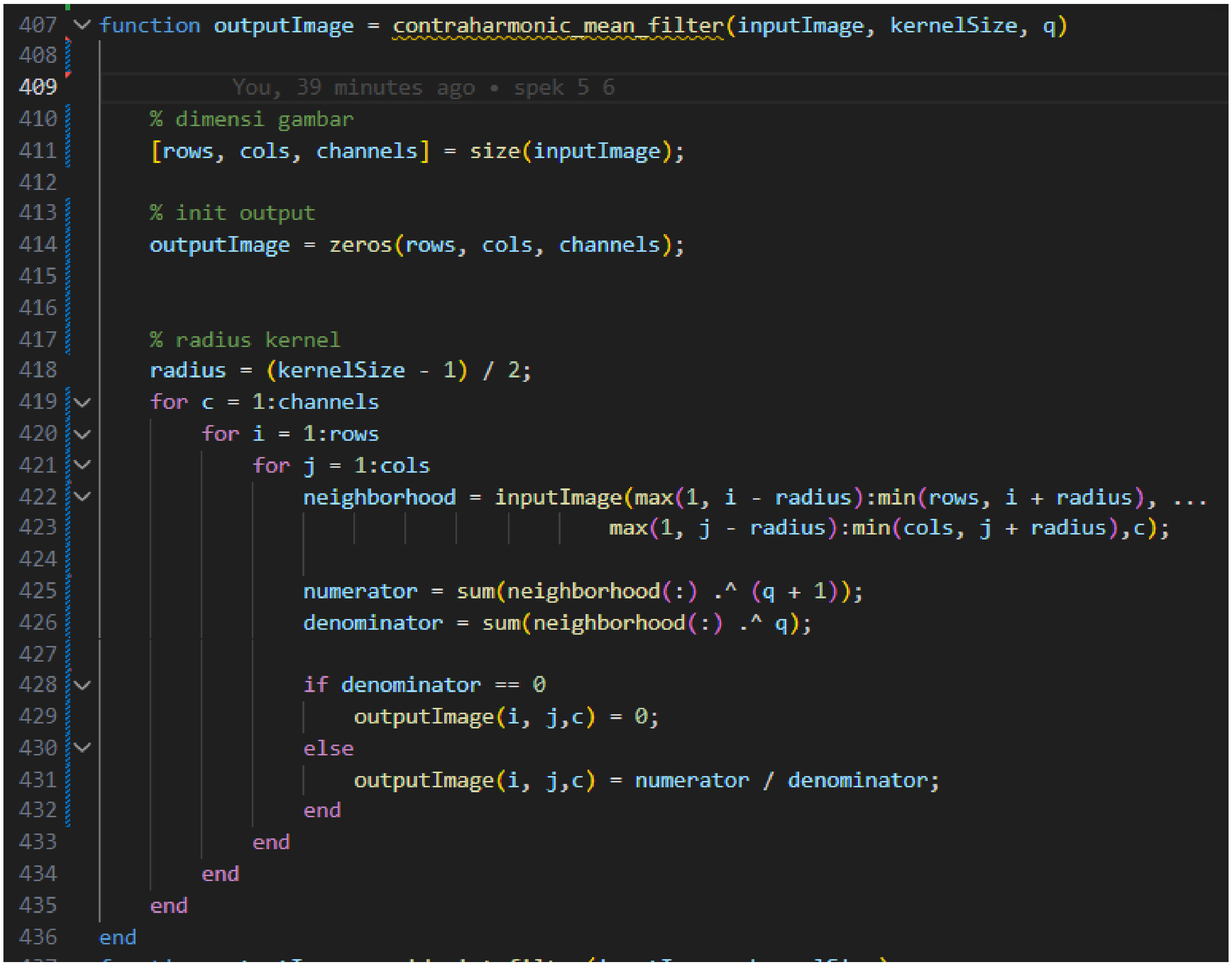
```
function outputImage = arithmetic_mean_filter(inputImage, kernelSize)
    % dimensi gambar
    [rows, cols, channels] = size(inputImage);

    % init output
    outputImage = zeros(rows, cols, channels);

    % radius kernel
    radius = (kernelSize - 1) / 2;

    for c = 1:channels
        for i = 1:rows
            for j = 1:cols
                neighborhood = inputImage(max(1, i - radius):min(rows, i + radius), ...
                                        max(1, j - radius):min(cols, j + radius),c);

                outputImage(i, j,c) = mean(neighborhood(:));
            end
        end
    end
end
```



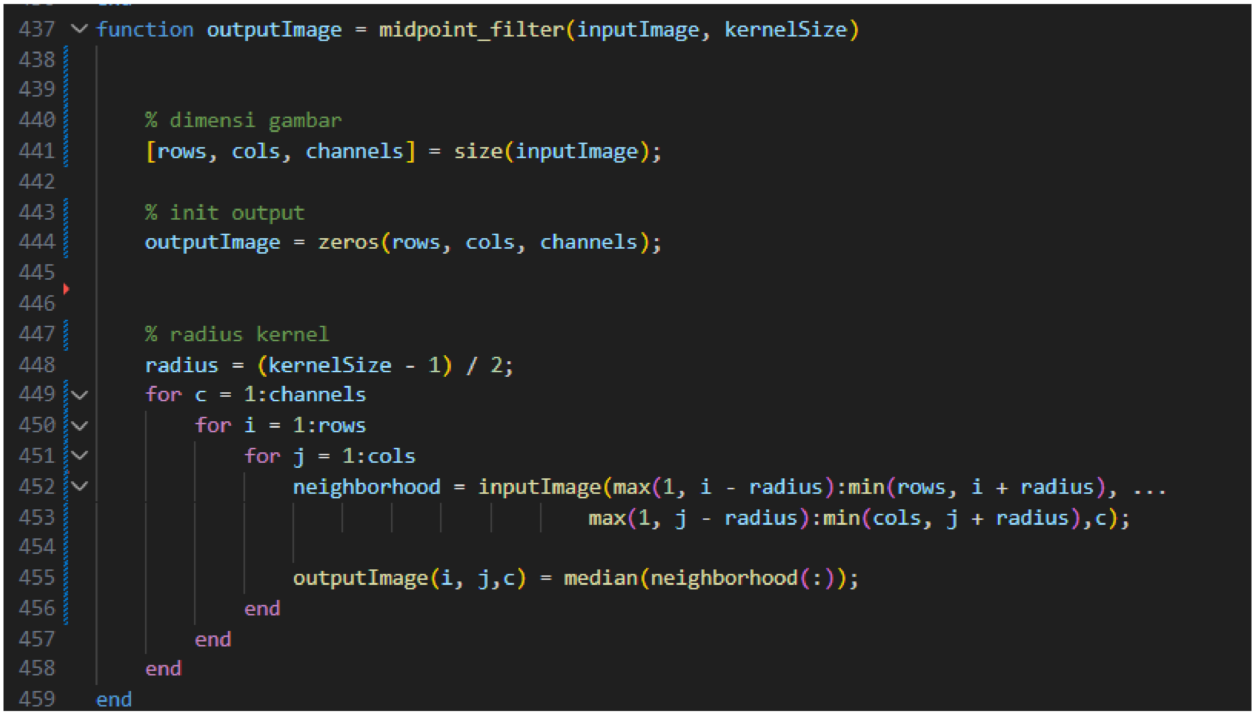
```
function outputImage = geometric_filter(inputImage, kernelSize, angle_degrees)


    % dimensi gambar
    [rows, cols, channels] = size(inputImage);

    % init output
    outputImage = zeros(rows, cols, channels);

    % radius kernel
    radius = (kernelSize - 1) / 2;

    % Convert the angle from degrees to radians
    angle_radians = deg2rad(angle_degrees);

    for c = 1:channels
        for i = 1:rows
            for j = 1:cols

                neighborhood = inputImage(max(1, i - radius):min(rows, i + radius), ...
                                        max(1, j - radius):min(cols, j + radius),c);


                rotated_neighborhood = imrotate(neighborhood, angle_degrees, 'bilinear', 'crop');

                center = floor(size(rotated_neighborhood) / 2) + 1;

                outputImage(i, j,c) = rotated_neighborhood(center(1), center(2));
            end
        end
    end
end
```



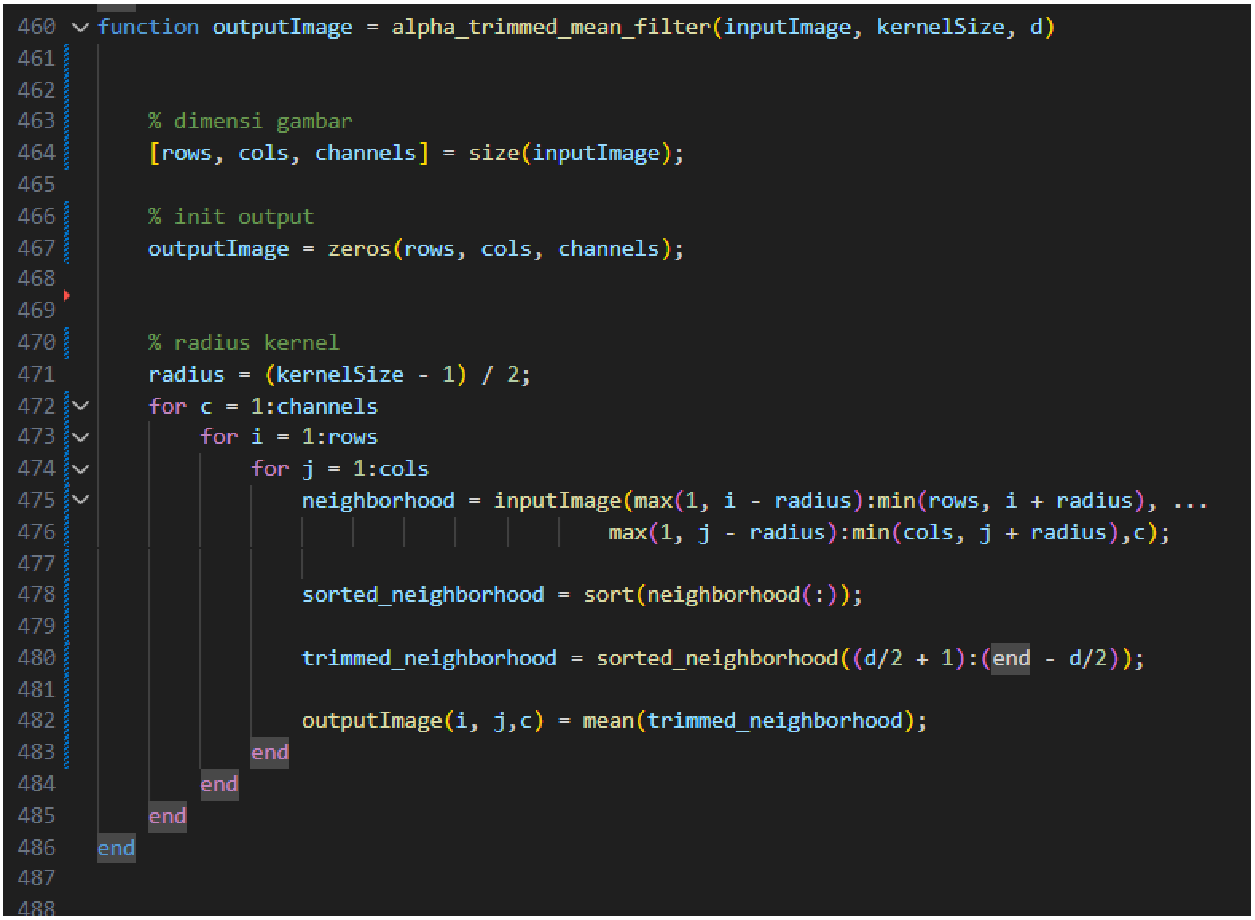
```
end
function outputImage = harmonic_mean_filter(inputImage, kernelSize)


    % dimensi gambar
    [rows, cols, channels] = size(inputImage);

    % init output
    outputImage = zeros(rows, cols, channels);

    % radius kernel
    radius = (kernelSize - 1) / 2;

    for c = 1:channels
        for i = 1:rows
            for j = 1:cols
                % Get the neighborhood of the current pixel
                neighborhood = inputImage(max(1, i - radius):min(rows, i + radius), ...
                                        max(1, j - radius):min(cols, j + radius),c);

                reciprocal_neighborhood = 1 ./ double(neighborhood);
                harmonic_mean = kernelSize^2 / sum(reciprocal_neighborhood(:));

                outputImage(i, j,c) = harmonic_mean;
            end
        end
    end
end
function outputImage = contraharmonic_mean_filter(inputImage, kernelSize, q)
```



```
function outputImage = contraharmonic_mean_filter(inputImage, kernelSize, q)

    You, 39 minutes ago • spek 5 6
    % dimensi gambar
    [rows, cols, channels] = size(inputImage);

    % init output
    outputImage = zeros(rows, cols, channels);


    % radius kernel
    radius = (kernelSize - 1) / 2;
    for c = 1:channels
        for i = 1:rows
            for j = 1:cols
                neighborhood = inputImage(max(1, i - radius):min(rows, i + radius), ...
                                        max(1, j - radius):min(cols, j + radius),c);

                numerator = sum(neighborhood(:) .^ (q + 1));
                denominator = sum(neighborhood(:) .^ q);

                if denominator == 0
                    outputImage(i, j,c) = 0;
                else
                    outputImage(i, j,c) = numerator / denominator;
                end
            end
        end
    end
end
```

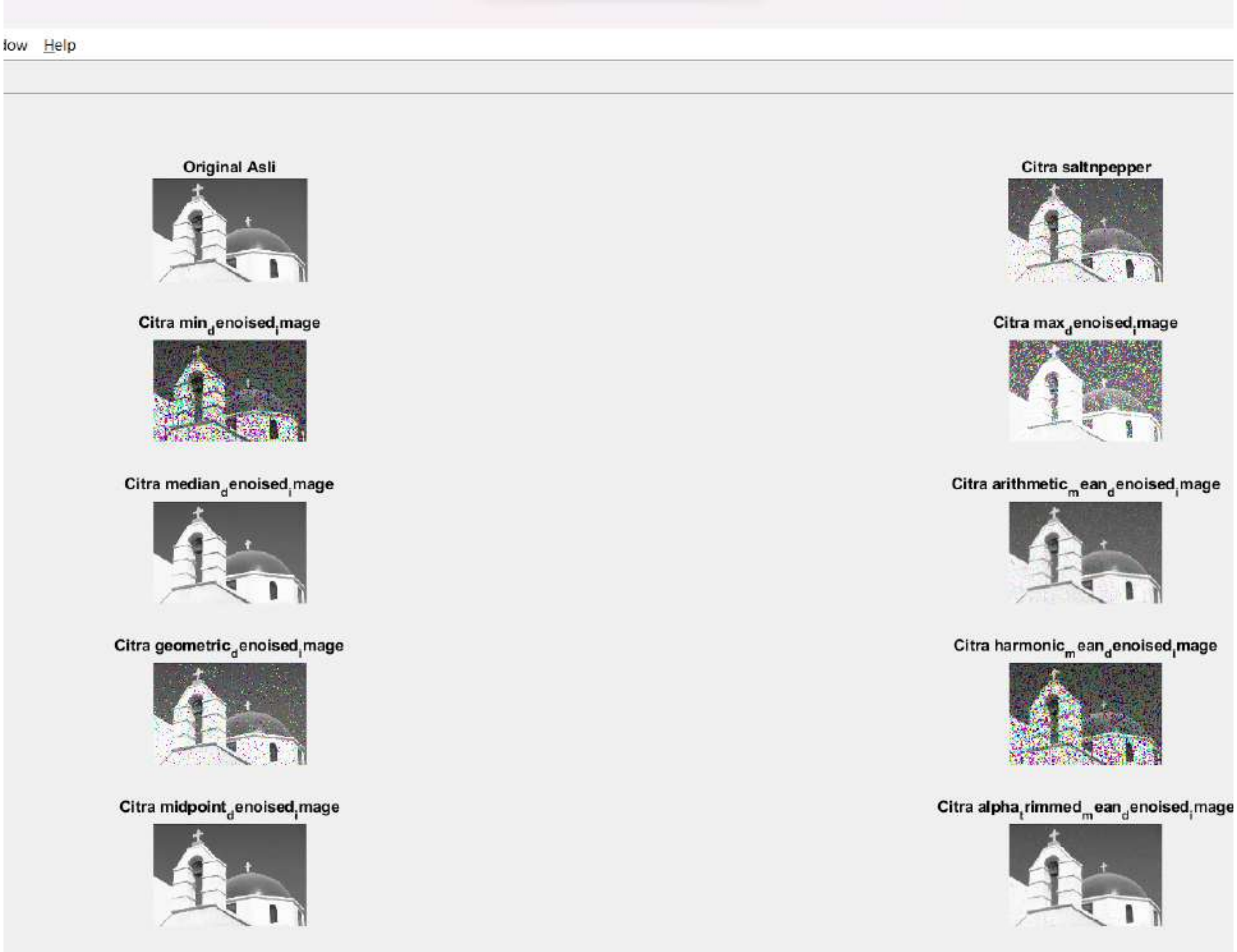


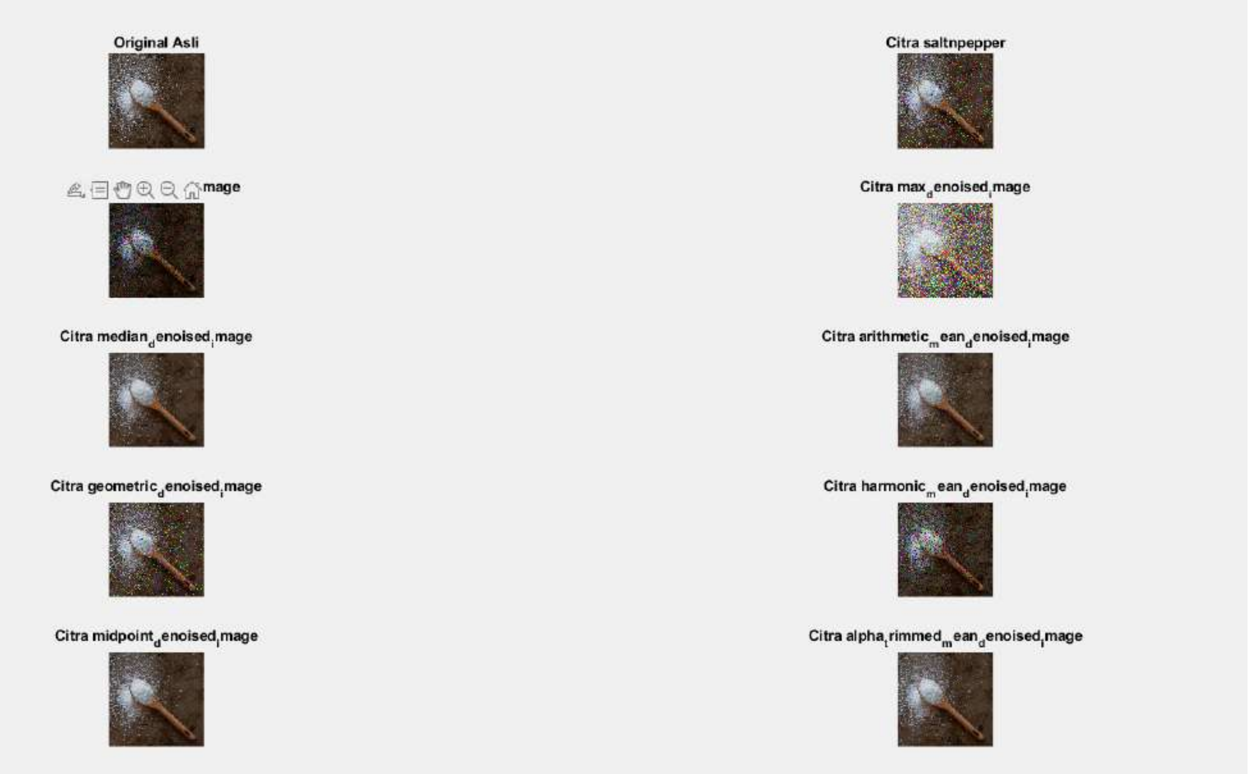
```
function outputImage = midpoint_filter(inputImage, kernelSize)


    % dimensi gambar
    [rows, cols, channels] = size(inputImage);

    % init output
    outputImage = zeros(rows, cols, channels);


    % radius kernel
    radius = (kernelSize - 1) / 2;
    for c = 1:channels
        for i = 1:rows
            for j = 1:cols
                neighborhood = inputImage(max(1, i - radius):min(rows, i + radius), ...
                                        max(1, j - radius):min(cols, j + radius),c);

                outputImage(i, j,c) = median(neighborhood(:));
            end
        end
    end
end
```



```
function outputImage = alpha_trimmed_mean_filter(inputImage, kernelSize, d)


    % dimensi gambar
    [rows, cols, channels] = size(inputImage);

    % init output
    outputImage = zeros(rows, cols, channels);


    % radius kernel
    radius = (kernelSize - 1) / 2;
    for c = 1:channels
        for i = 1:rows
            for j = 1:cols
                neighborhood = inputImage(max(1, i - radius):min(rows, i + radius), ...
                                        max(1, j - radius):min(cols, j + radius),c);

                sorted_neighborhood = sort(neighborhood(:));

                trimmed_neighborhood = sorted_neighborhood((d/2 + 1):(end - d/2));

                outputImage(i, j,c) = mean(trimmed_neighborhood);
            end
        end
    end
end

```

ii. Hasil Eksekusi


low Help
Original Asli
Citra saltnpepper
Citra min$_d$enoised$_i$mage
Citra max$_d$enoised$_i$mage
Citra median$_d$enoised$_i$mage
Citra arithmetic$_m$ean$_d$enoised$_i$mage
Citra geometric$_d$enoised$_i$mage
Citra harmonic$_m$ean$_d$enoised$_i$mage
Citra midpoint$_d$enoised$_i$mage
Citra alpha$_t$rimmed$_m$ean$_d$enoised$_i$mage

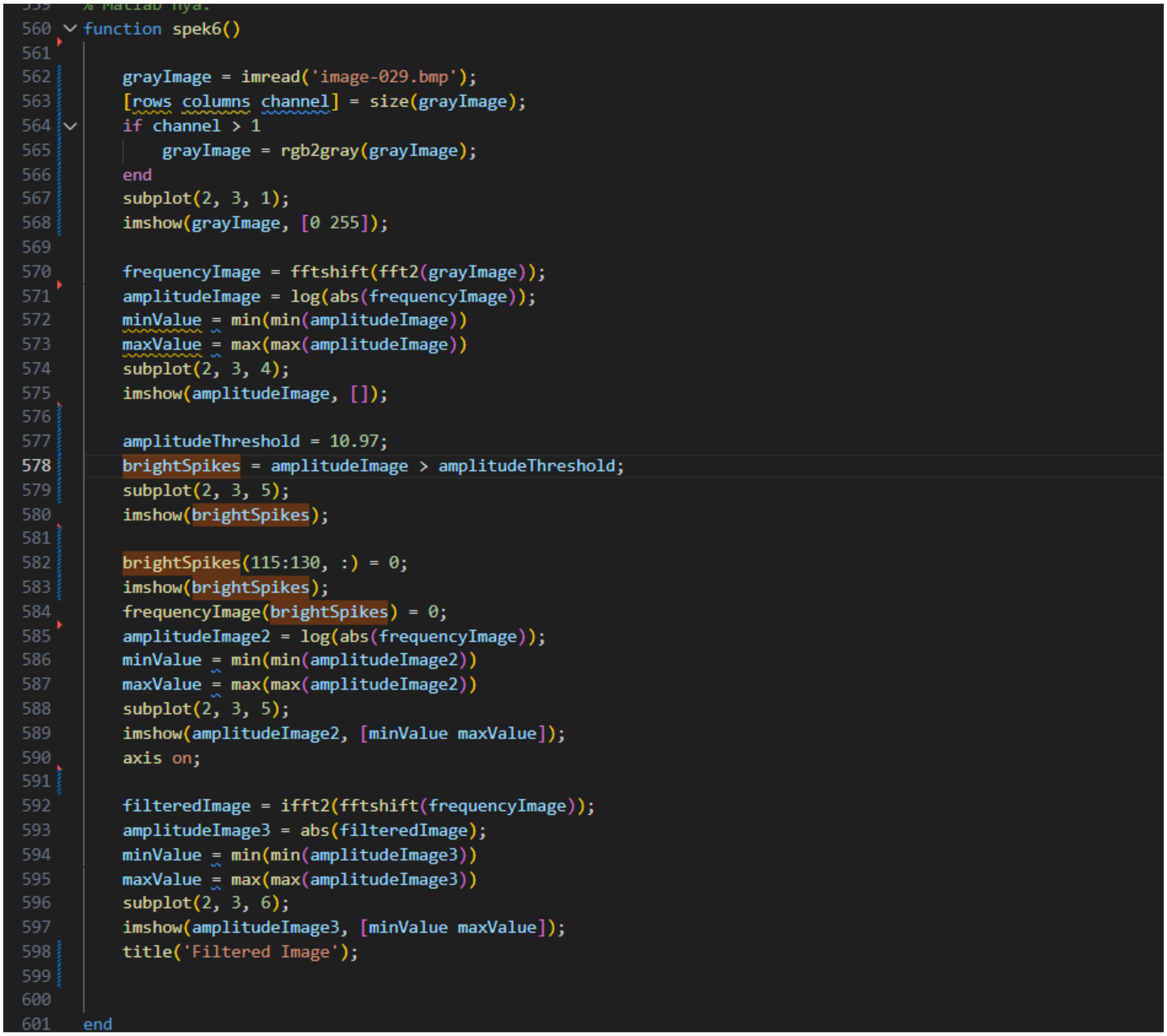

Original Asli
Citra saltnpepper
mage
Citra max$_d$enoised$_i$mage
Citra median$_d$enoised$_i$mage
Citra arithmetic$_m$ean$_d$enoised$_i$mage
Citra geometric$_d$enoised$_i$mage
Citra harmonic$_m$ean$_d$enoised$_i$mage
Citra midpoint$_d$enoised$_i$mage
Citra alpha$_t$rimmed$_m$ean$_d$enoised$_i$mage

iii. Analisa

- Median filter, midpoint, dan alpha rimmed mean berhasil men filter derau salt and paper dengan cukup baik
- Min filter gagal men filter pixel dengan value tinggi
- Max filter gagal men filter pixel dengan value rendah
- Arithmetic Median filter berhasil men filter derau salt and pepper dengan cukup baik
- Geometric filter, dan Harmonic mean gagal memfilter derau salt and pepper

f. Penghilangan Derau Periodik

i. Kode Program



```
% Matlab nya:
function spek6()

    grayImage = imread('image-029.bmp');
    [rows columns channel] = size(grayImage);
    if channel > 1
        grayImage = rgb2gray(grayImage);
    end
    subplot(2, 3, 1);
    imshow(grayImage, [0 255]);

    frequencyImage = fftshift(fft2(grayImage));
    amplitudeImage = log(abs(frequencyImage));
    minValue = min(min(amplitudeImage))
    maxValue = max(max(amplitudeImage))
    subplot(2, 3, 4);
    imshow(amplitudeImage, []);

    amplitudeThreshold = 10.97;
    brightSpikes = amplitudeImage > amplitudeThreshold;
    subplot(2, 3, 5);
    imshow(brightSpikes);

    brightSpikes(115:130, :) = 0;
    imshow(brightSpikes);
    frequencyImage(brightSpikes) = 0;
    amplitudeImage2 = log(abs(frequencyImage));
    minValue = min(min(amplitudeImage2))
    maxValue = max(max(amplitudeImage2))
    subplot(2, 3, 5);
    imshow(amplitudeImage2, [minValue maxValue]);
    axis on;

    filteredImage = ifft2(fftshift(frequencyImage));
    amplitudeImage3 = abs(filteredImage);
    minValue = min(min(amplitudeImage3))
    maxValue = max(max(amplitudeImage3))
    subplot(2, 3, 6);
    imshow(amplitudeImage3, [minValue maxValue]);
    title('Filtered Image');


end
```

ii. Hasil Eksekusi

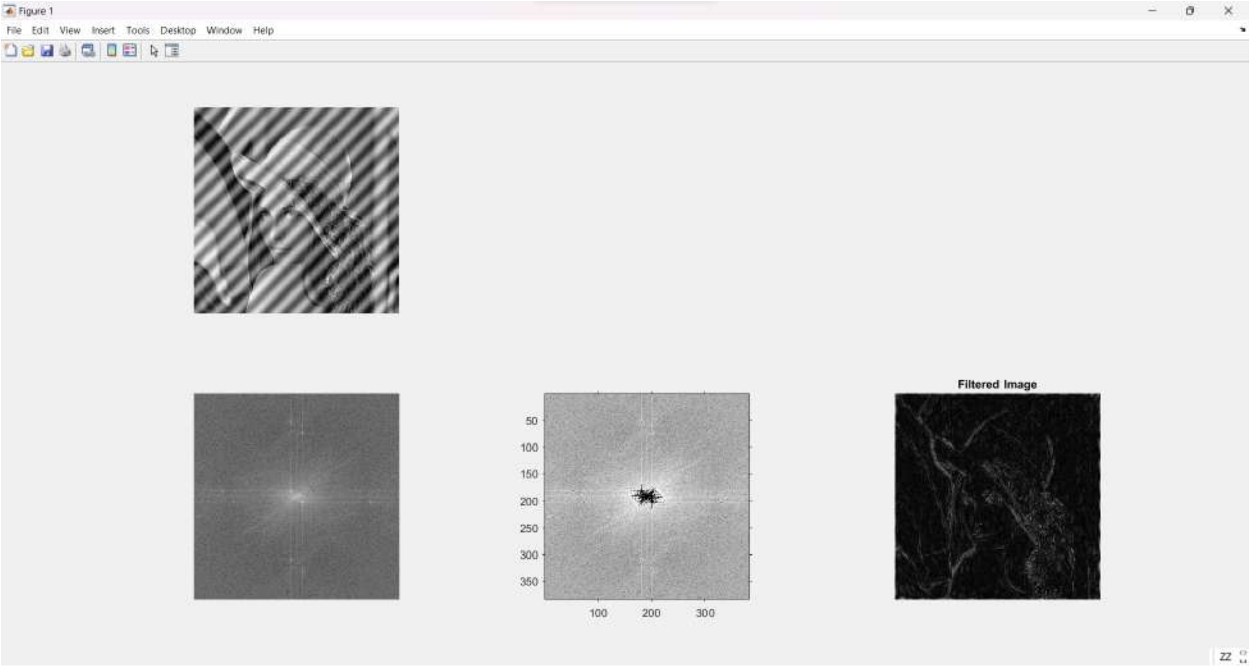
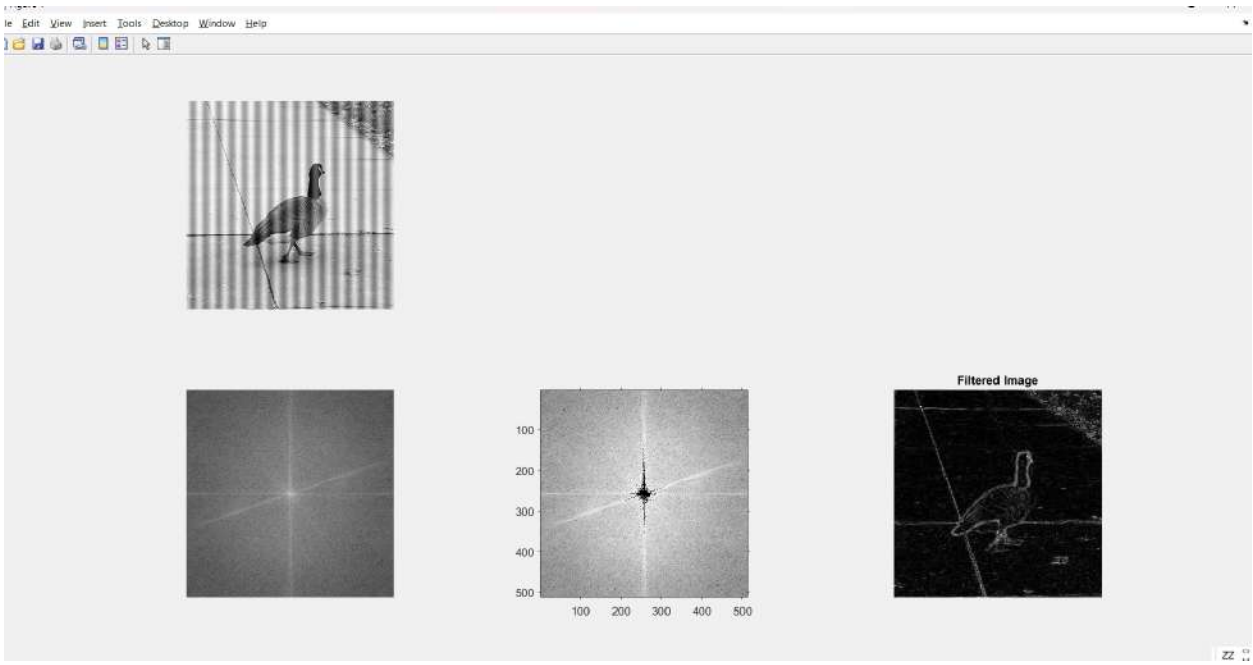
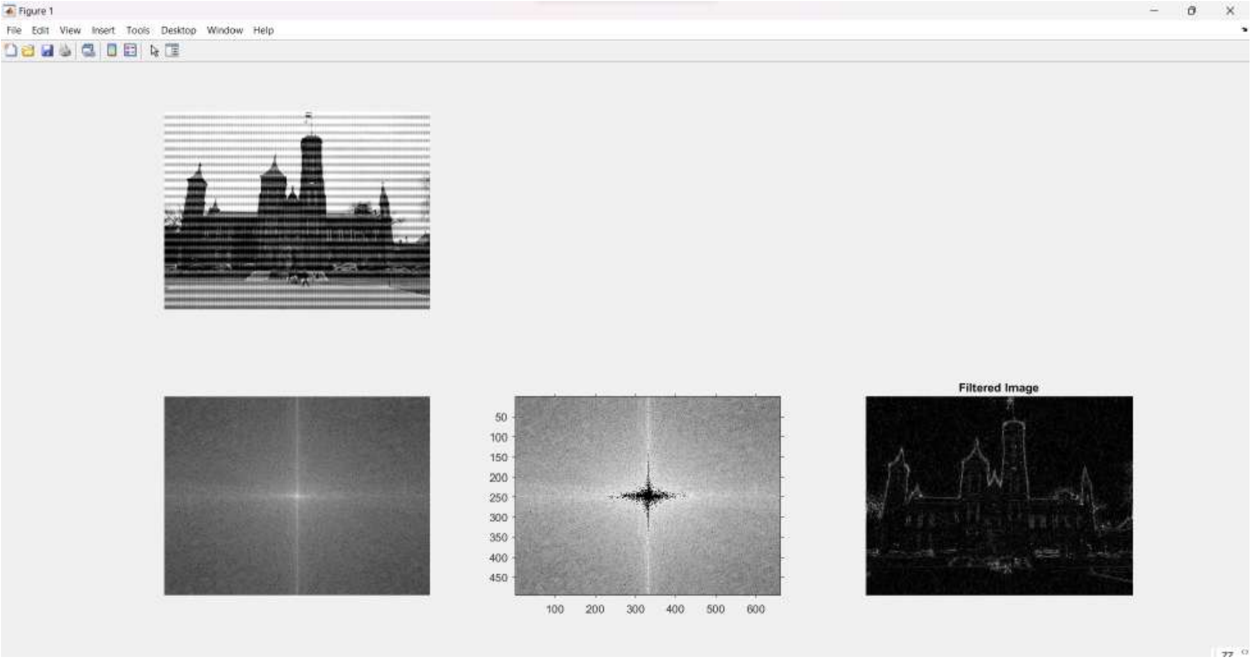


Figure 1
File Edit View Insert Tools Desktop Window Help
50
100
150
200
250
300
350
100
200
300
Filtered Image


le Edit View Insert Tools Desktop Window Help
100
200
300
400
500
100
200
300
400
500
Filtered Image

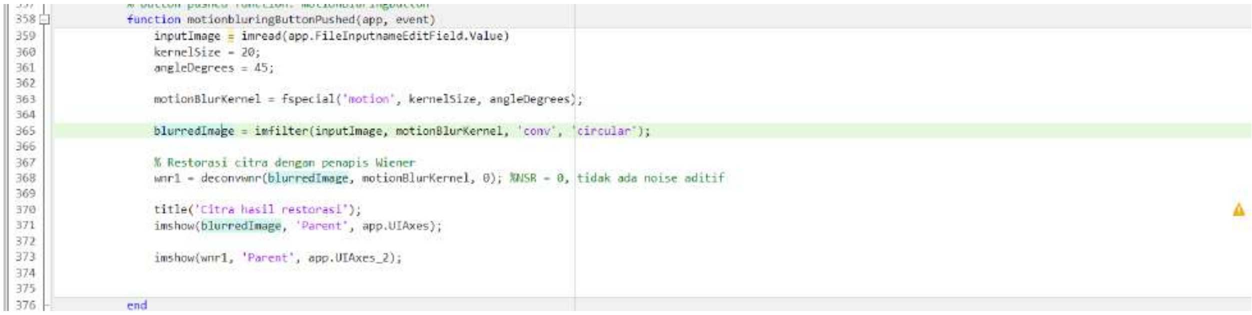


iii. Analisa

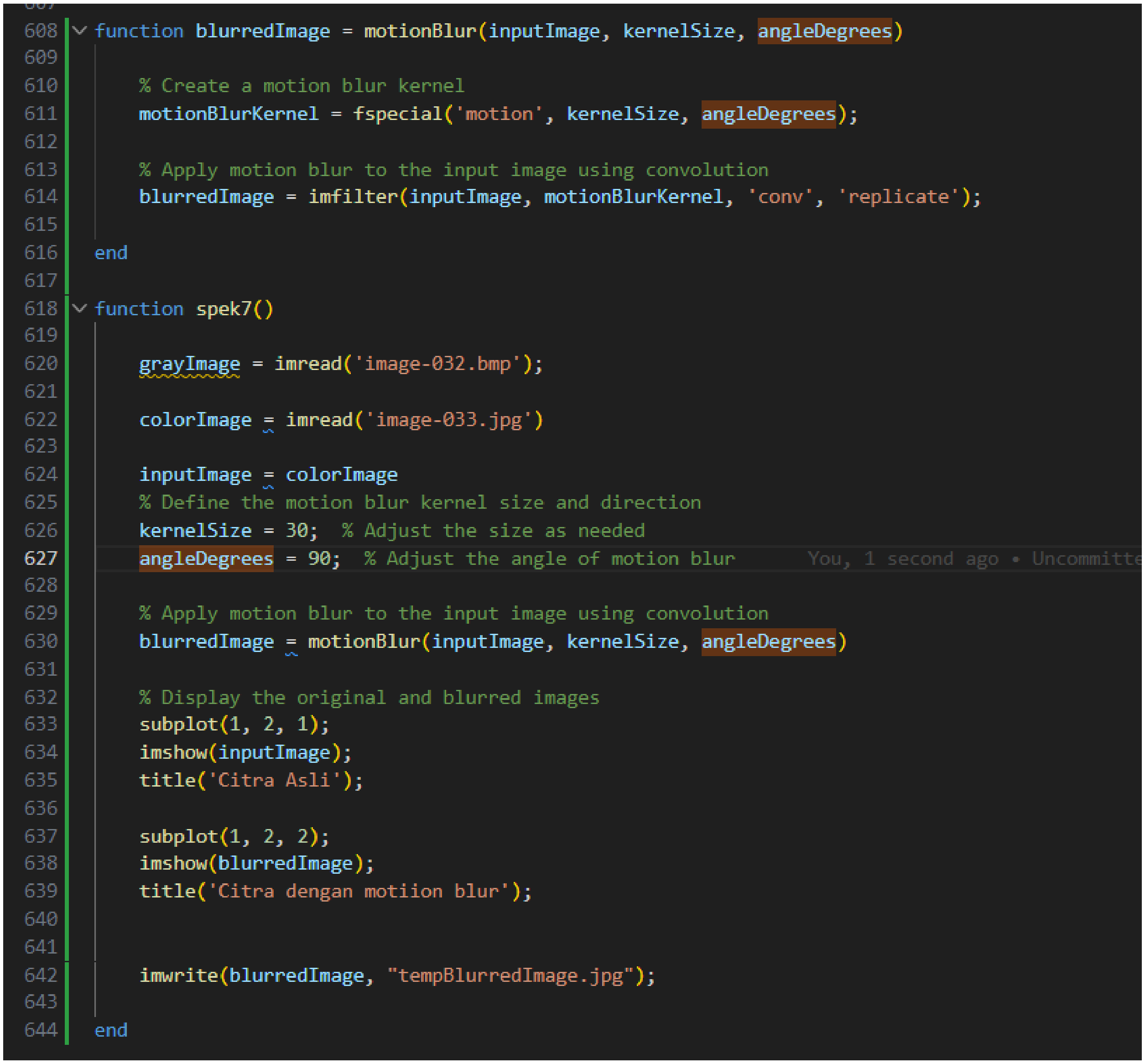
Image masih belum berhasil di filter dengan baik, dikarenakan oleh parameter tuning dari filter masih buruk.

g. Dekonvolusi Motion Blurring

i. Kode Program



```
        % Button pushed function: motionbluringButton
        function motionbluringButtonPushed(app, event)
            inputImage = imread(app.FileInputnameEditField.Value)
            kernelSize = 20;
            angleDegrees = 45;

            motionBlurKernel = fspecial('motion', kernelSize, angleDegrees);

            blurredImage = imfilter(inputImage, motionBlurKernel, 'conv', 'circular');

            % Restorasi citra dengan penapis Wiener
            wnr1 = deconvwnr(blurredImage, motionBlurKernel, 0); %NSR = 0, tidak ada noise aditif

            title('Citra hasil restorasi');
            imshow(blurredImage, 'Parent', app.UIAxes);

            imshow(wnr1, 'Parent', app.UIAxes_2);


        end
```



```
function blurredImage = motionBlur(inputImage, kernelSize, angleDegrees)

    % Create a motion blur kernel
    motionBlurKernel = fspecial('motion', kernelSize, angleDegrees);

    % Apply motion blur to the input image using convolution
    blurredImage = imfilter(inputImage, motionBlurKernel, 'conv', 'replicate');

end

function spek7()

    grayImage = imread('image-032.bmp');

    colorImage = imread('image-033.jpg')

    inputImage = colorImage
    % Define the motion blur kernel size and direction
    kernelSize = 30;  % Adjust the size as needed
    angleDegrees = 90;  % Adjust the angle of motion blur      You, 1 second ago • Uncommitte

    % Apply motion blur to the input image using convolution
    blurredImage = motionBlur(inputImage, kernelSize, angleDegrees)

    % Display the original and blurred images
    subplot(1, 2, 1);
    imshow(inputImage);
    title('Citra Asli');

    subplot(1, 2, 2);
    imshow(blurredImage);
    title('Citra dengan motiion blur');


    imwrite(blurredImage, "tempBlurredImage.jpg");

end
```

ii. Hasil Eksekusi

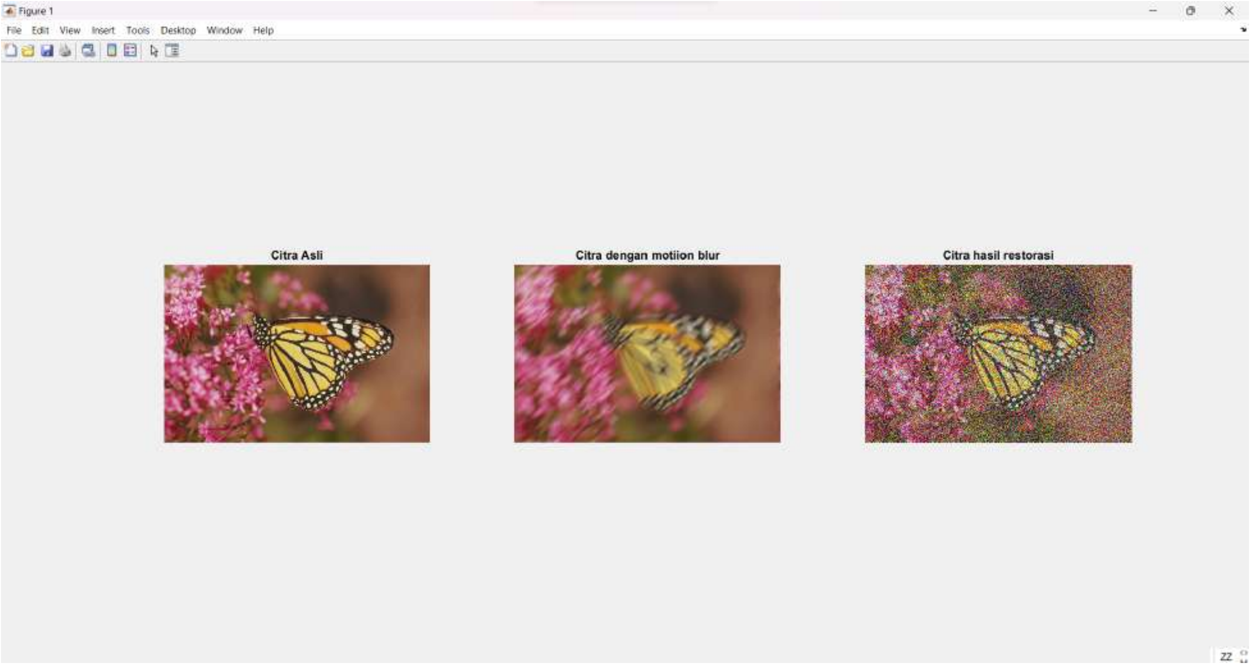
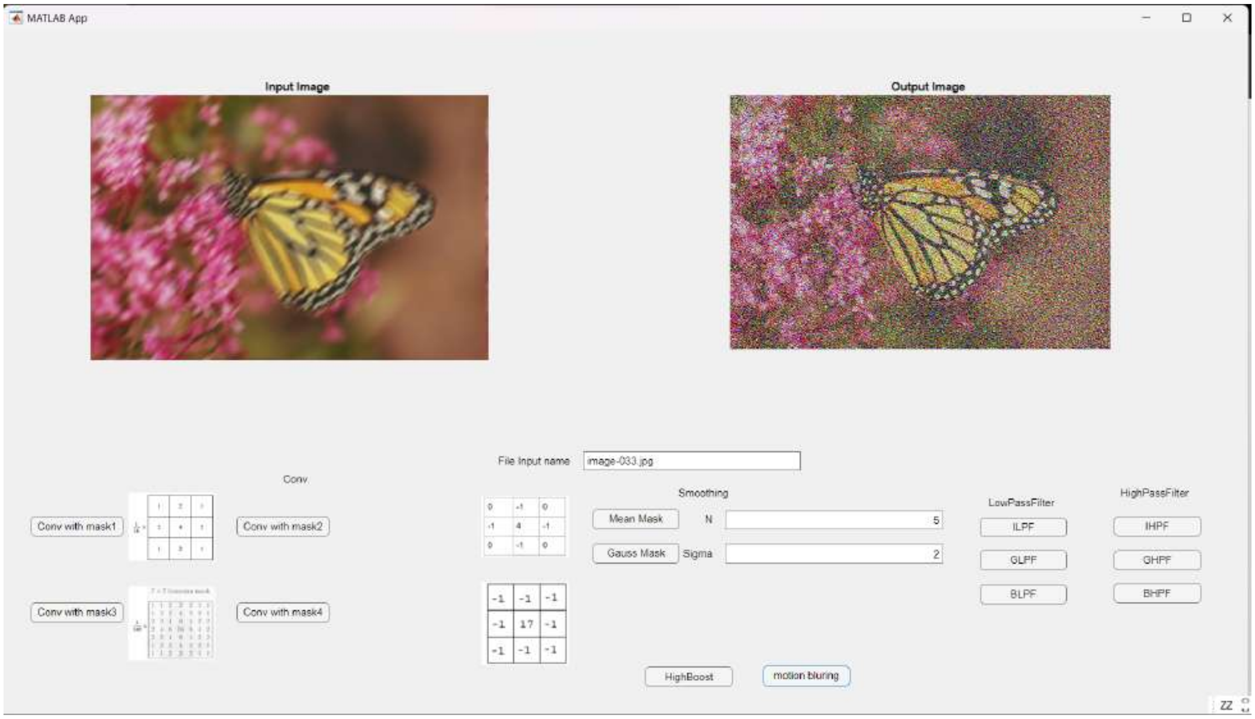


Figure 1
File Edit View Insert Tools Desktop Window Help
Citra Asli
Citra dengan motiion blur
Citra hasil restorasi



File Input name: image-033.jpg

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1

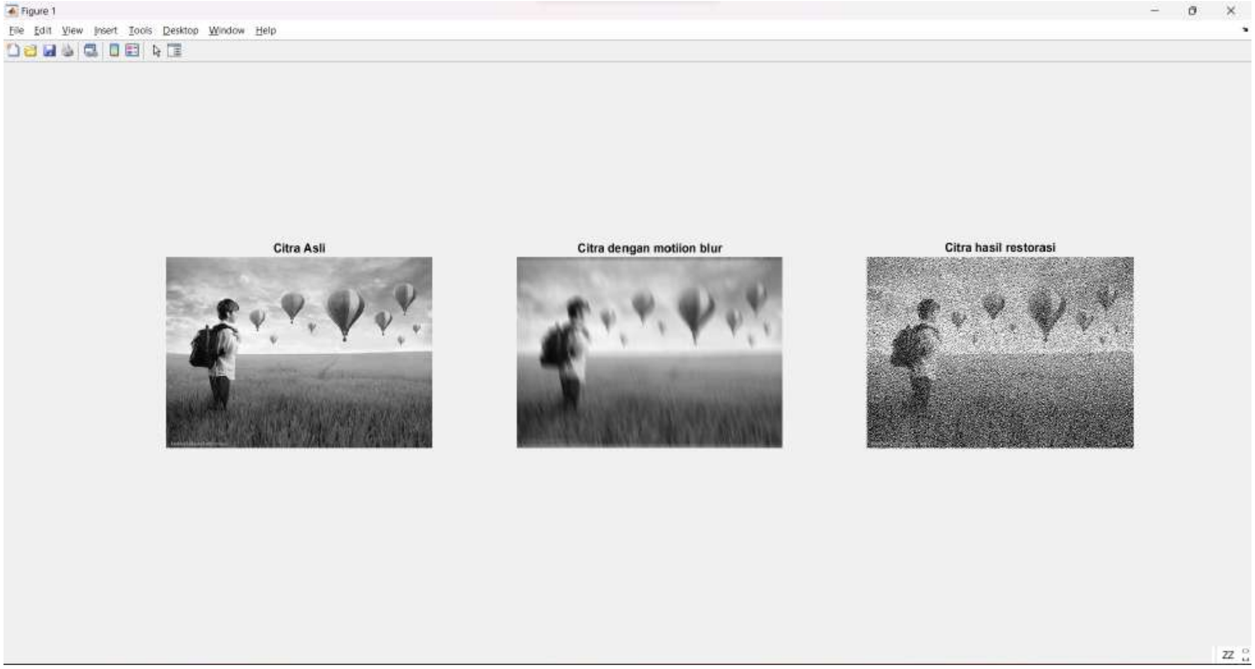
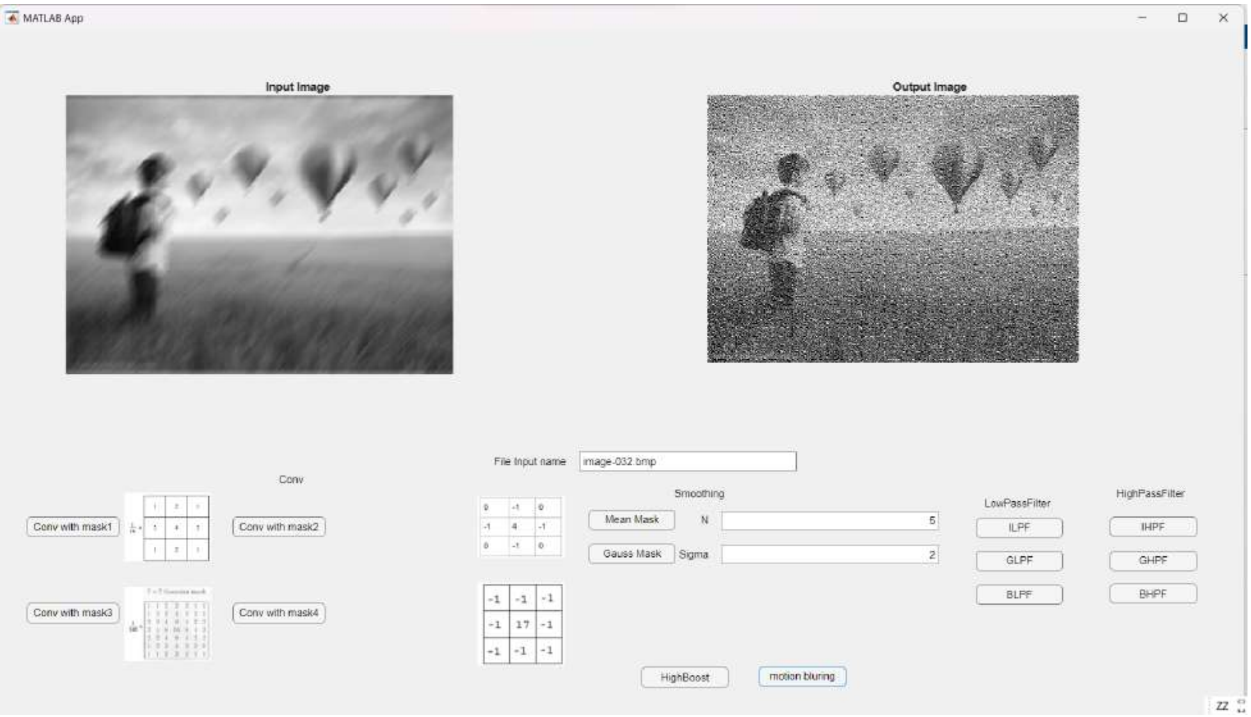


Figure 1
File Edit View Insert Tools Desktop Window Help
Citra Asli
Citra dengan motiion blur
Citra hasil restorasi



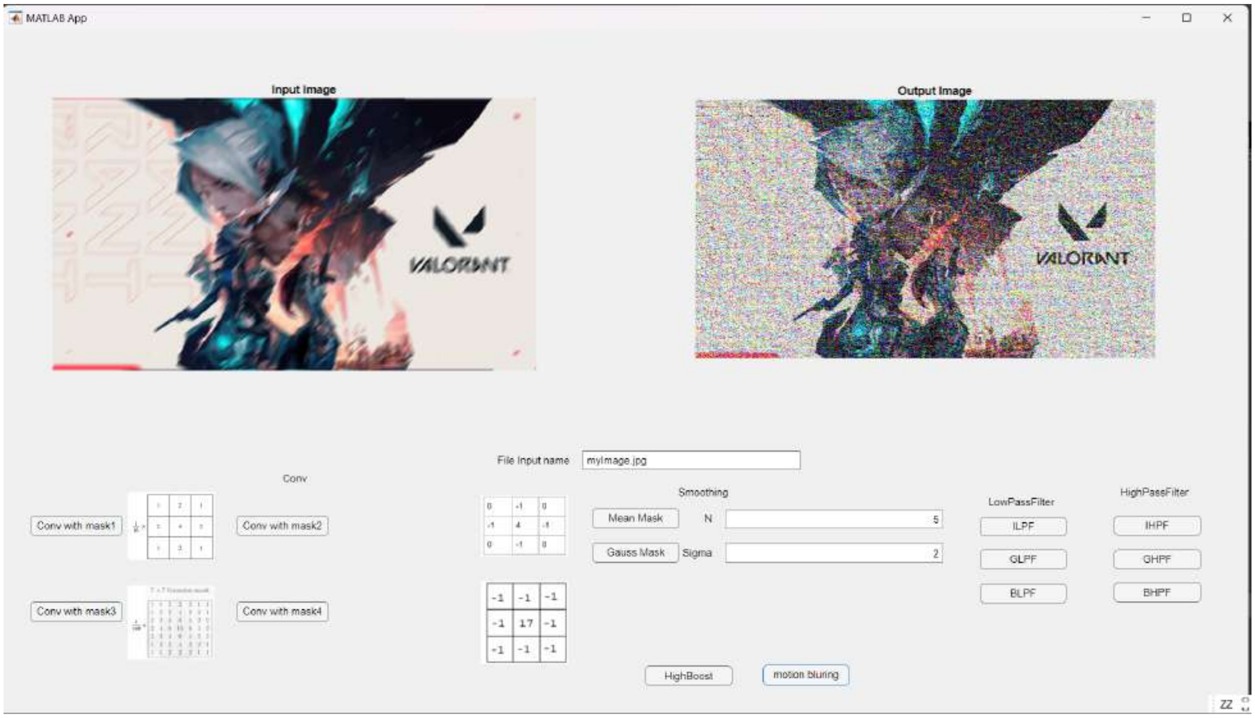
File Input name: image-032.bmp

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



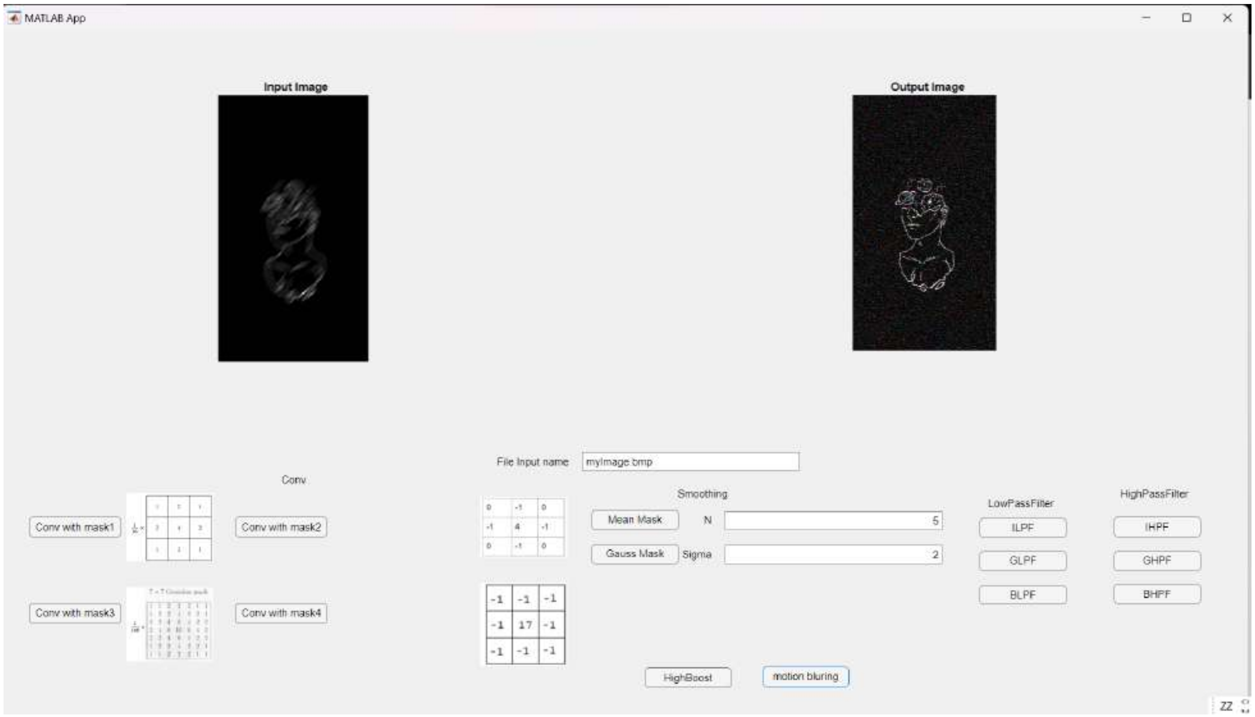
File Input name: myImage.jpg

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1



File Input name: myImage.bmp

N: 5

Sigma: 2

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

-1	-1	-1
-1	17	-1
-1	-1	-1

iii. Analisa

Algoritma berhasil menghilangkan motion blurring walaupun menghasilkan bentuk derau lainya.

3. Alamat Github

https://github.com/ronggurmahendra/Tugas2-Citra-2023.git