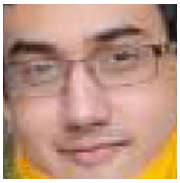
Laporan Tugas 2 IF4073 Interpretasi dan Pengolahan Citra

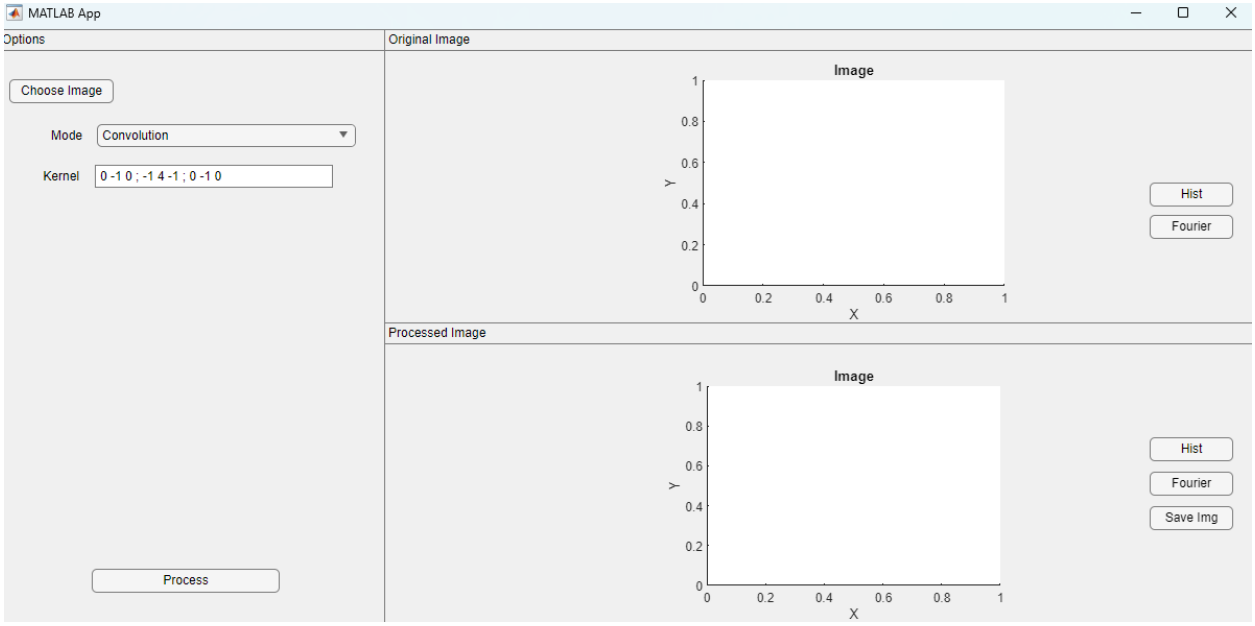
Pembuatan Program Image Restoration dengan Matlab



Oleh
Safiq Faray - 13519145

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023

A. Screenshot GUI



B. Kode Program

a. Program 1 : Convolution

Berikut adalah kode dari program 1.

```
function convolutionImg = convolution(img, kernel, padding)
    %center point of the kernel
    kernelSize = size(kernel);
    kernelSize = kernelSize(1);
    kernelCenter = floor(kernelSize / 2);
    %image dimension
    imgSize = size(img);
    channels = imgSize(3);

    convolutionImg = zeros(imgSize);

    %div factor
    div = sum(kernel, "all");
    if (div == 0)
        div = 1;
    end
    for ch = 1 : channels
        %change img to float
        processedImg = img(:,:,ch);
        %reshape the size of image based on chosen padding
        if (padding == 'fillzero')
            processedImg = padarray(processedImg, [kernelCenter, kernelCenter],
```

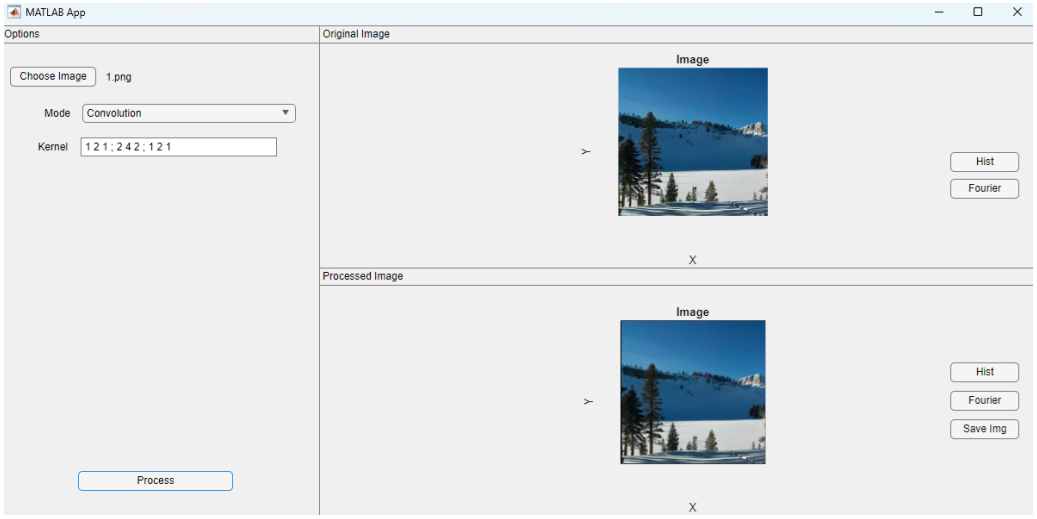
```
0, "both");
        end
        processedImg = im2double(processedImg);
        for i=1:imgSize(1) - kernelCenter - 1
            for j=1:imgSize(2) - kernelCenter - 1
                extractedImage = processedImg(i : i + kernelSize - 1 , j : j +
kernelSize - 1);
                convolutionImg(i + kernelCenter, j + kernelCenter, ch) =
sum(extractedImage .* kernel, "all") / div;
            end
        end
    end
  end
```

Kode dikumpulkan ke dalam class static bernama “ImProcTools”. Proses convolution disini akan menggunakan kernel input dari pengguna. Untuk padding, bisa dipilih antara “fillzero”, yaitu mengisi padding valid dengan 0, atau “ignore” yaitu tidak menggunakan padding sama sekali
Contoh penggunaan adalah sebagai berikut.

Untuk mask ini :

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

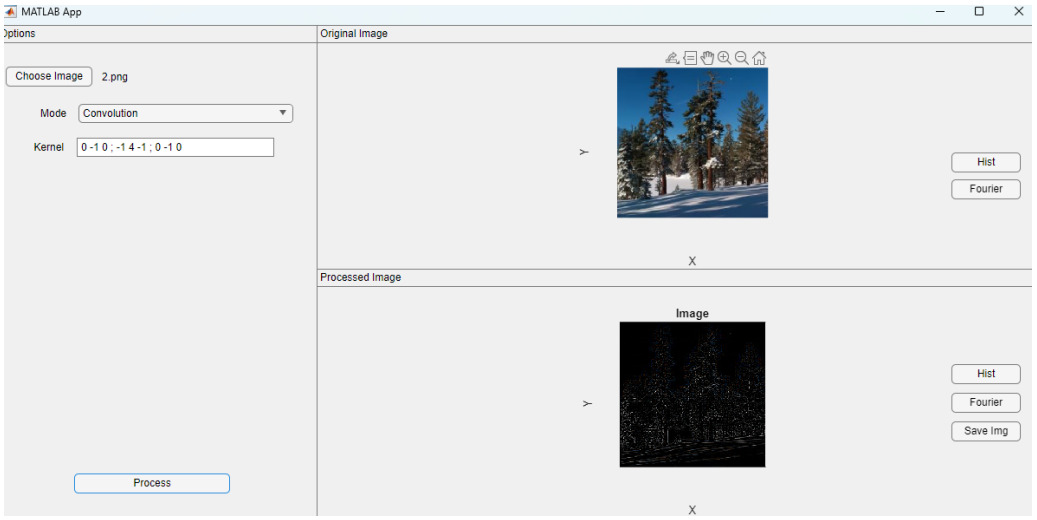
Hasil convolution :



Untuk kernel ini :

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

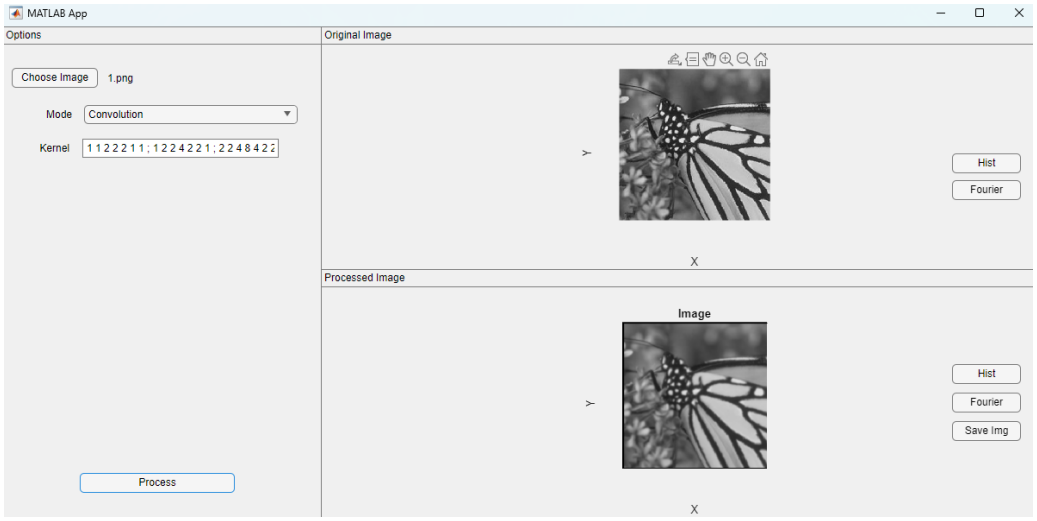
Hasil convolution :



Untuk kernel ini :

7×7 Gaussian mask

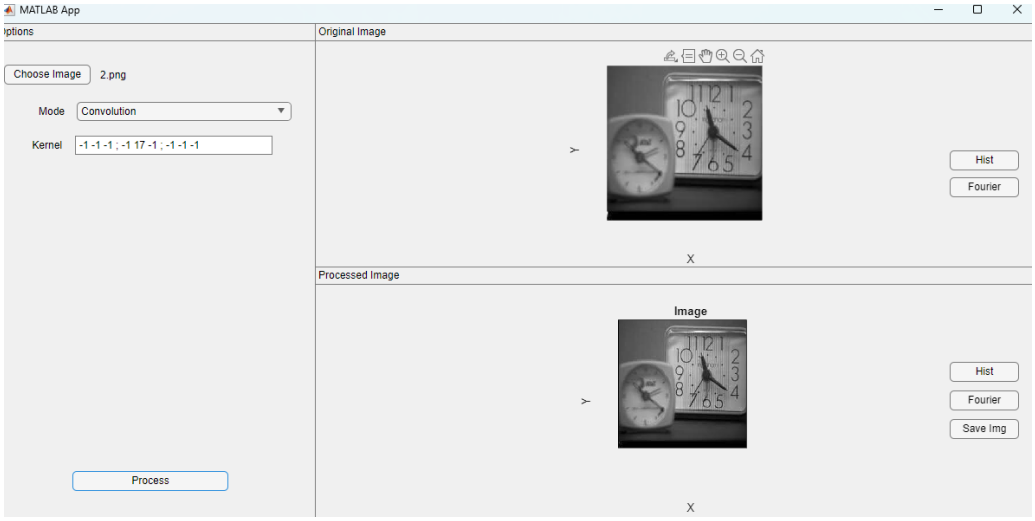
$$\frac{1}{140} \times \begin{bmatrix} 1 & 1 & 2 & 2 & 2 & 1 & 1 \\ 1 & 2 & 2 & 4 & 2 & 2 & 1 \\ 2 & 2 & 4 & 8 & 4 & 2 & 2 \\ 2 & 4 & 8 & 16 & 8 & 4 & 2 \\ 2 & 2 & 4 & 8 & 4 & 2 & 2 \\ 1 & 2 & 2 & 4 & 2 & 2 & 1 \\ 1 & 1 & 2 & 2 & 2 & 1 & 1 \end{bmatrix}$$



Untuk kernel ini :

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

Hasil convolution :



b. Program 2 : Image Smoothing and Blurring

Berikut adalah kode dari Low/High Pass Filter

```
function pfImg = pf(img, h)
        [height, width] = size(img);
        %padding
        pfImg = img;
        pad_h = 2*height;
```

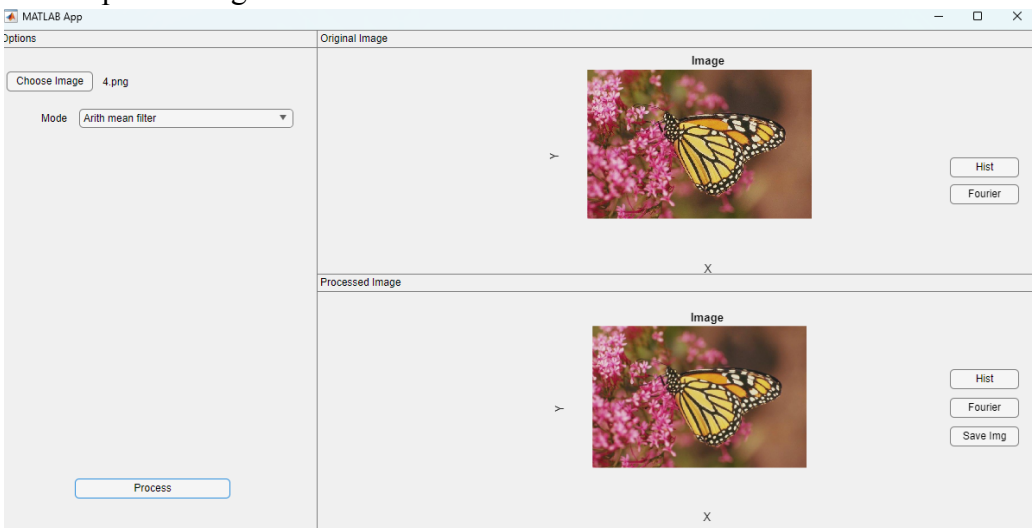
```
    pad_w = 2*width;
    pfImg = padarray(pfImg, [height, width], 0, "post");
    %fourier
    fourier_img = fft2(double(pfImg));

    ilpf_f = h .* fourier_img;
    ilpf_f2 = real(ifft2(ilpf_f));
    pfImg = ilpf_f2(1:height, 1:width);
    pfImg = uint8(pfImg);
end
function ipfImg = ipf(img, d0, high)
    [height, width, channels] = size(img);
    pad_h = height * 2;
    pad_w = width * 2;
    ipfImg = zeros(size(img));
    for ch=1:channels
        u = 0:(pad_h - 1);
        v = 0:(pad_w - 1);
        %indices in meshgrid
        idx = find(u > pad_h / 2);
        u(idx) = u(idx) - pad_h;
        idy = find(v > pad_w / 2);
        v(idy) = v(idy) - pad_w;
        %meshgrid arrays
        [V,U] = meshgrid(v,u);
        d = sqrt(U .^ 2 + V .^ 2);
        h = double(d <= d0);
        if (high)
            h = 1-h;
        end
        h = fftshift(h);
        h = ifftshift(h);
        ipfImg(:,:,ch) = Filter.pf(img(:,:,ch), h);
    end
    ipfImg = uint8(ipfImg);
end
function gpfImg = gpf(img, d0, high)
    [height, width, channels] = size(img);
    pad_h = height * 2;
    pad_w = width * 2;
    gpfImg = zeros(size(img));
    for ch=1:channels
        u = 0:(pad_h - 1);
        v = 0:(pad_w - 1);
        %indices in meshgrid
        idx = find(u > pad_h / 2);
```

```
        u(idx) = u(idx) - pad_h;
        idy = find(v > pad_w / 2);
        v(idy) = v(idy) - pad_w;
        %meshgrid arrays
        [V,U] = meshgrid(v,u);
        d = sqrt(U .^ 2 + V .^ 2);
        h = exp(-(d.^2) ./(2 * (d0^2)));
        if (high)
            h = 1-h;
        end
        h = fftshift(h);
        h = ifftshift(h);
        gpfImg(:,:,ch) = Filter.pf(img(:,:,ch), h);
    end
    gpfImg = uint8(gpfImg);
end
function bpfImg = bpf(img, d0, n, high)
    if (n < 1)
        n = 1;
    end
    [height, width, channels] = size(img);
    pad_h = height * 2;
    pad_w = width * 2;
    bpfImg = zeros(size(img));
    for ch=1:channels
        u = 0:(pad_h - 1);
        v = 0:(pad_w - 1);
        %indices in meshgrid
        idx = find(u > pad_h / 2);
        u(idx) = u(idx) - pad_h;
        idy = find(v > pad_w / 2);
        v(idy) = v(idy) - pad_w;
        %meshgrid arrays
        [V,U] = meshgrid(v,u);
        d = sqrt(U .^ 2 + V .^ 2);
        h = 1 ./ (1 + (d ./d0) .^(2*n));
        if (high)
            h = 1-h;
        end
        h = fftshift(h);
        h = ifftshift(h);
        bpfImg(:,:,ch) = Filter.pf(img(:,:,ch), h);
    end
    bpfImg = uint8(bpfImg);
end
```

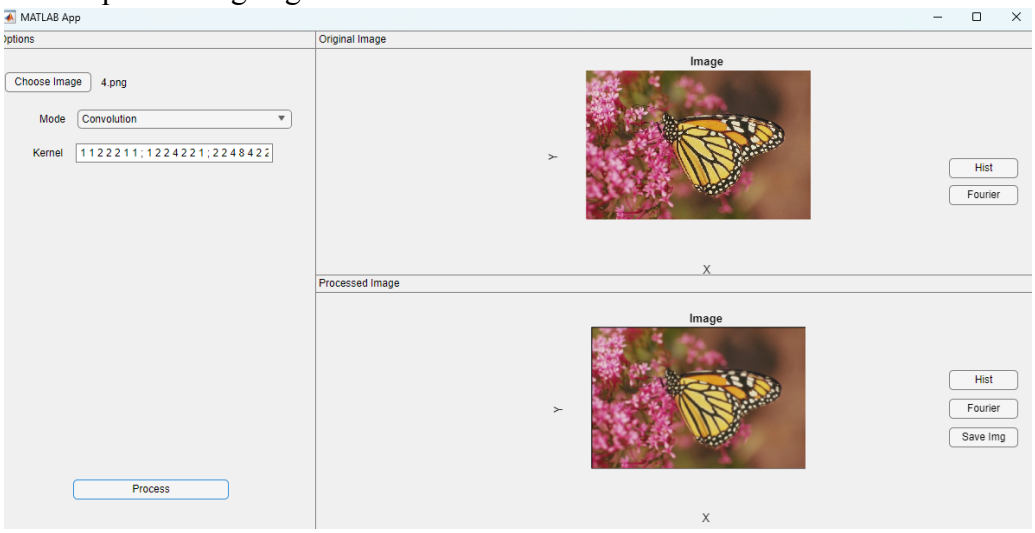
Kode diatas dibuat dalam satu class yaitu “Filter”. Method-method diatas adalah filter dalam ranah frekuensi. Untuk high pass filter, maka parameter method “high” dimasukkan true, jika low pass filter maka dimasukkan false. Untuk mean filter kodenya akan dituliskan di bagian selanjutnya.

Berikut adalah contoh image blurring

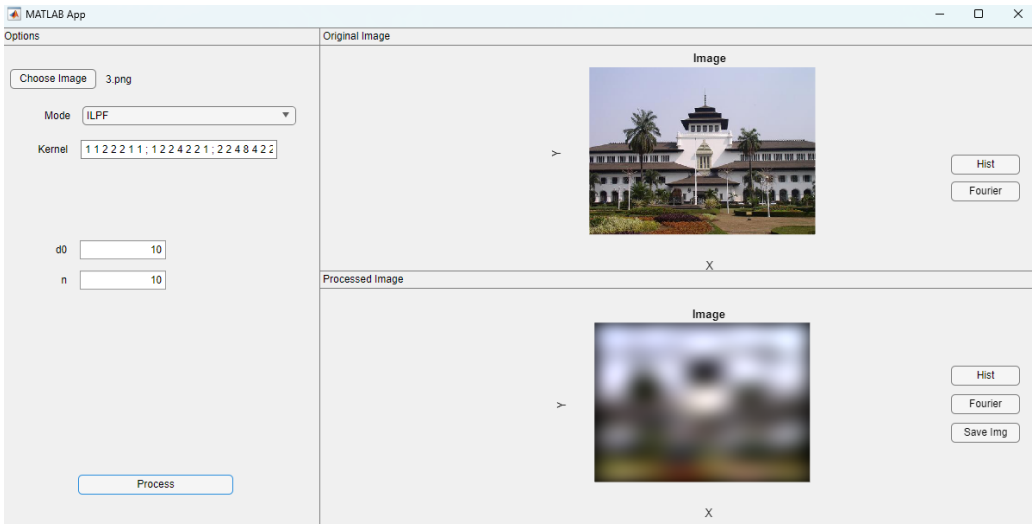
Ranah spasial dengan mean filter



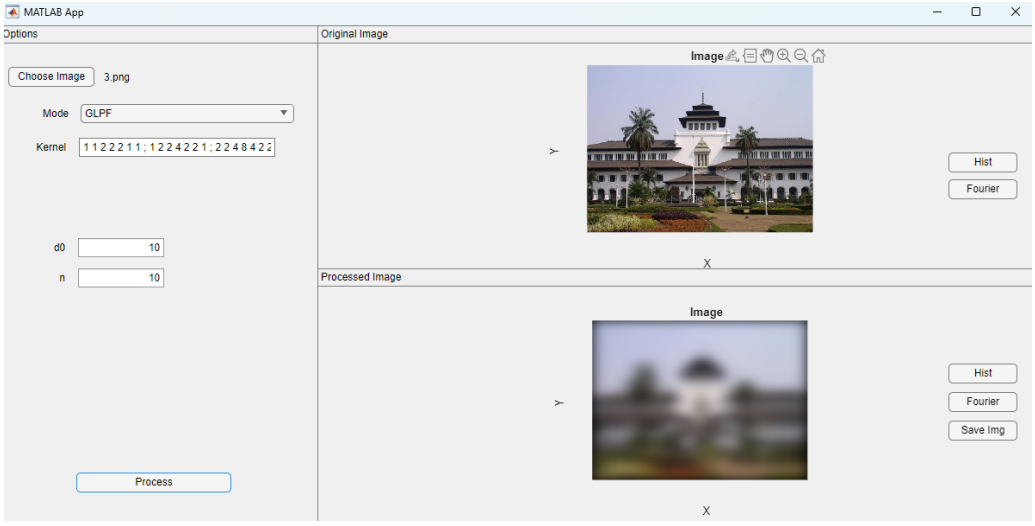
Ranah spasial dengan gaussian mask



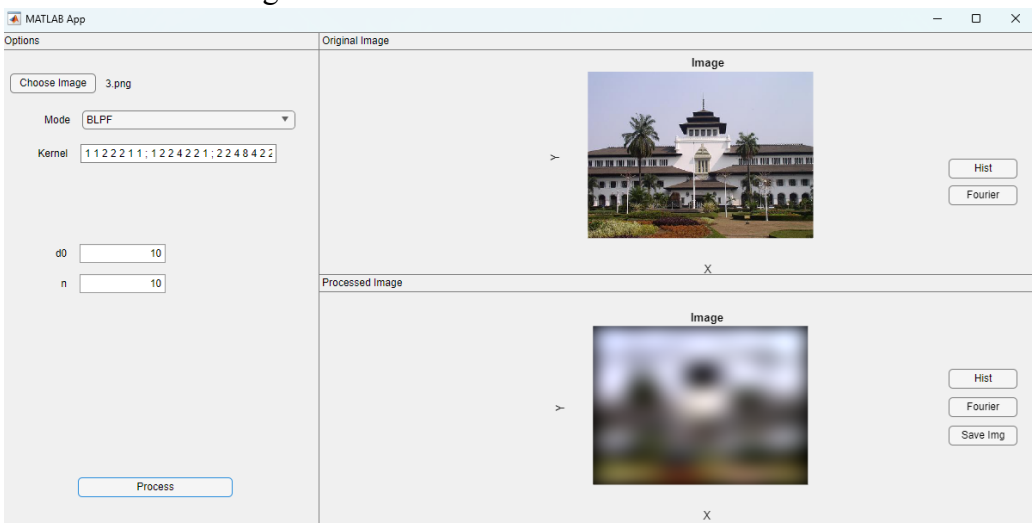
Ranah frekuensi dengan ILPF



Ranah frekuensi dengan GLPF

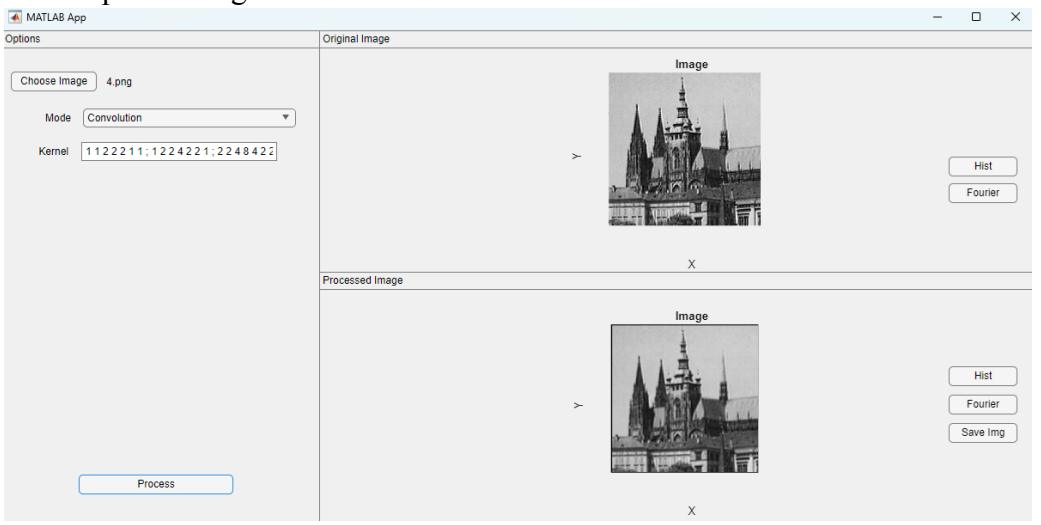


Ranah frekuensi dengan BLPF

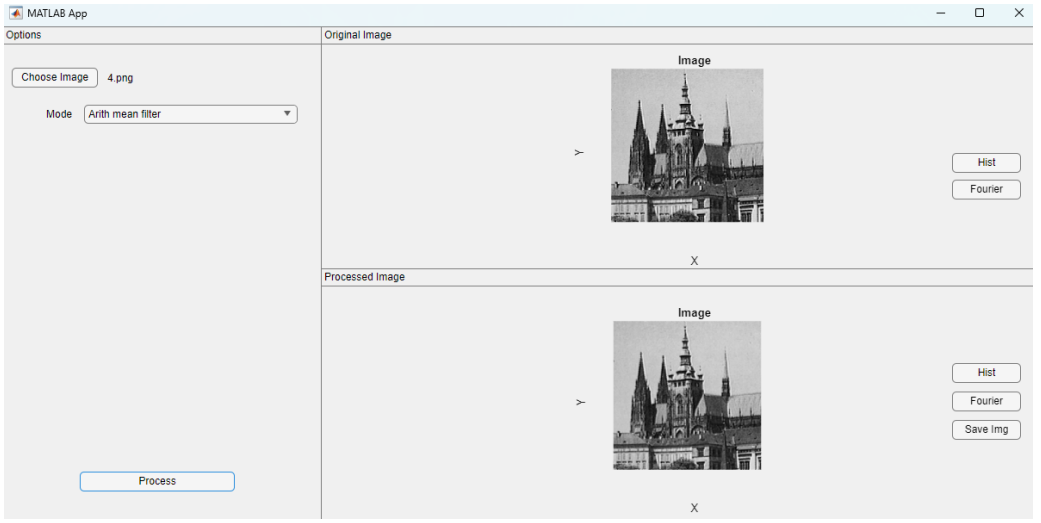


Berikut adalah contoh image smoothing

Ranah spasial dengan Gaussian mask



Ranah spasial dengan mean filter

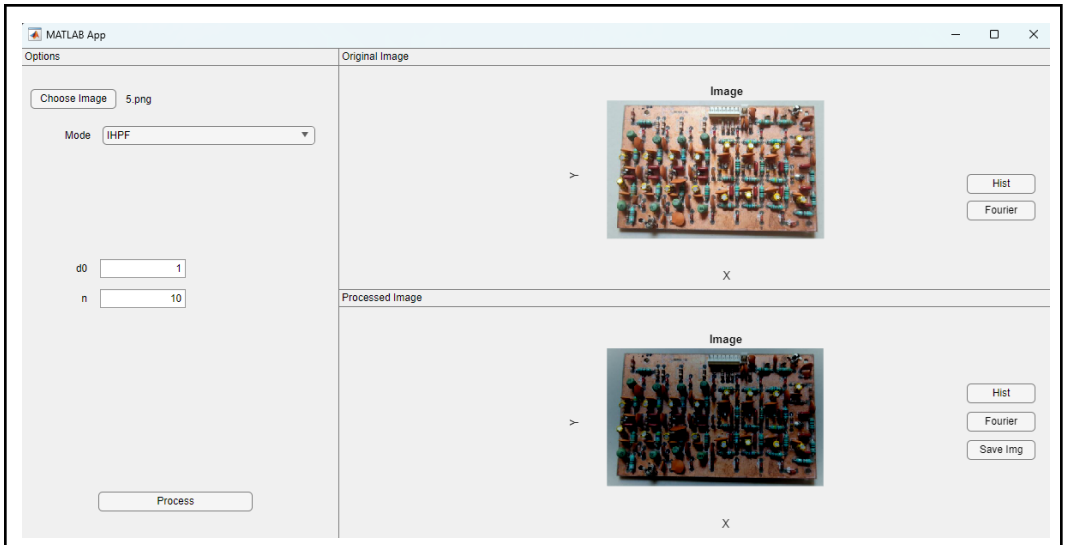


c. Program 3 : High Pass Filters

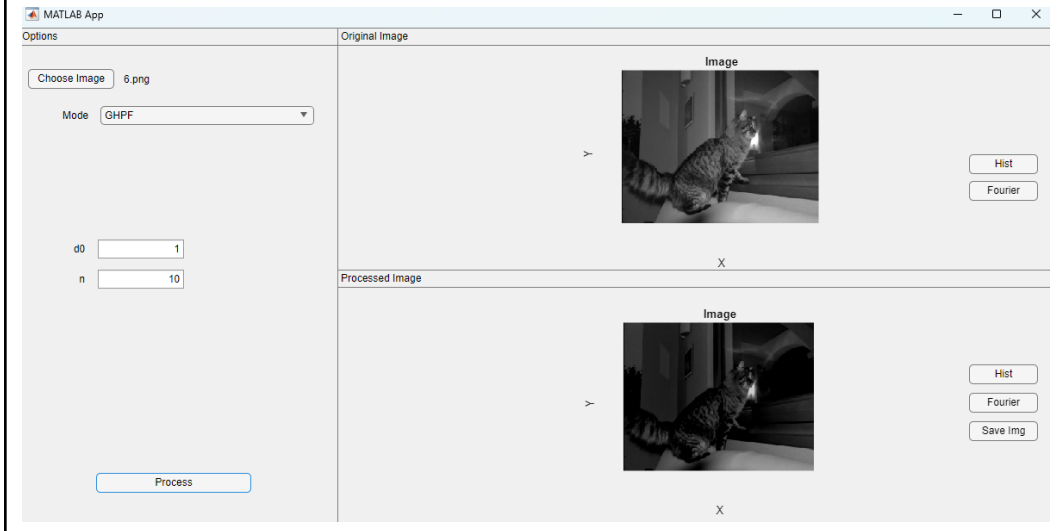
Implementasi program dijadikan satu dengan implementasi program 2, sehingga penjelasan sama.

Berikut adalah contoh penggunaannya.

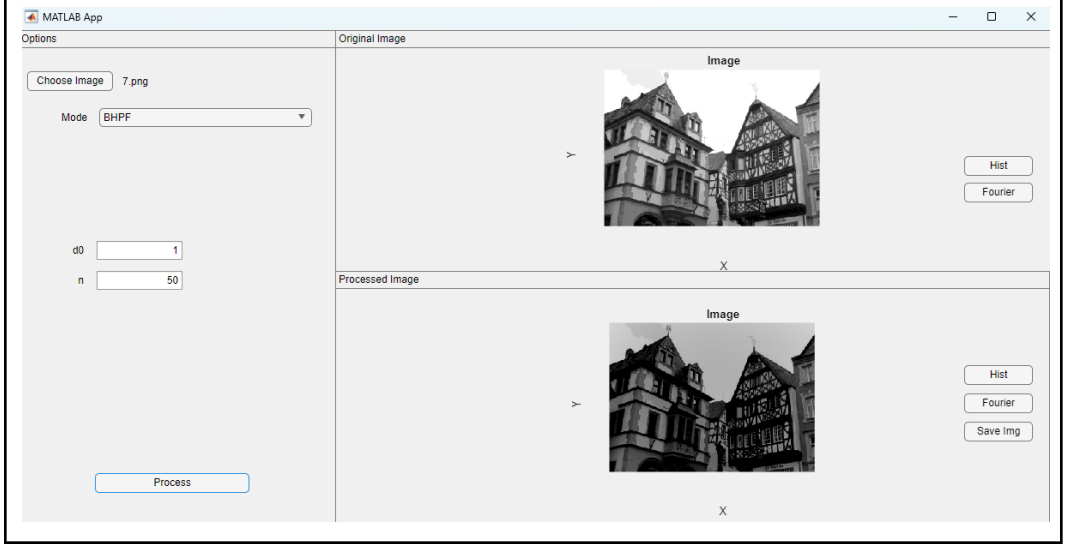
IHPF


MATLAB App
Options
Choose Image
5.png
Mode
IHPF
d0
1
n
10
Process
Original Image
Image
Y
X
Hist
Fourier
Processed Image
Image
Y
X
Hist
Fourier
Save Img

GHPF


MATLAB App
Options
Choose Image
6.png
Mode
GHPF
d0
1
n
10
Process
Original Image
Image
Y
X
Hist
Fourier
Processed Image
Image
Y
X
Hist
Fourier
Save Img

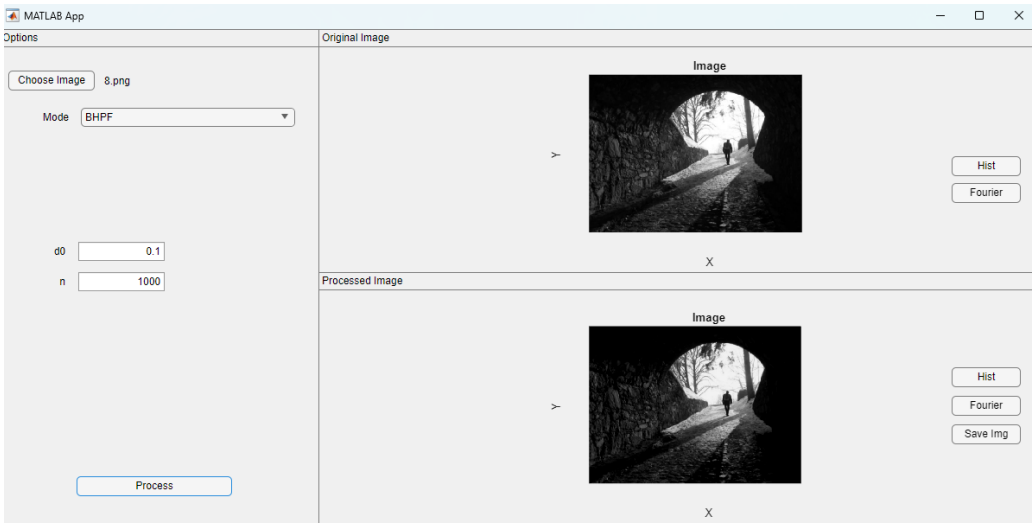
BHPF


MATLAB App
Options
Choose Image
7.png
Mode
BHPF
d0
1
n
50
Process
Original Image
Image
Y
X
Hist
Fourier
Processed Image
Image
Y
X
Hist
Fourier
Save Img

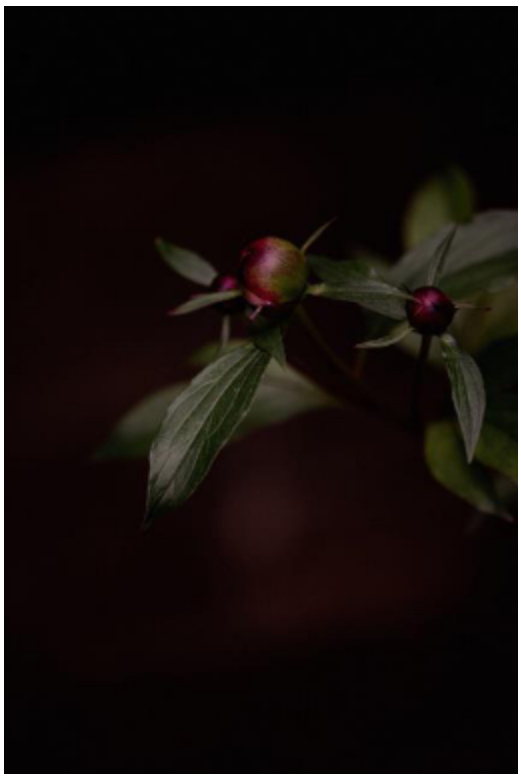
d. Program 4 : Penerangan Citra dalam Ranah Frekuensi

Gambar 1

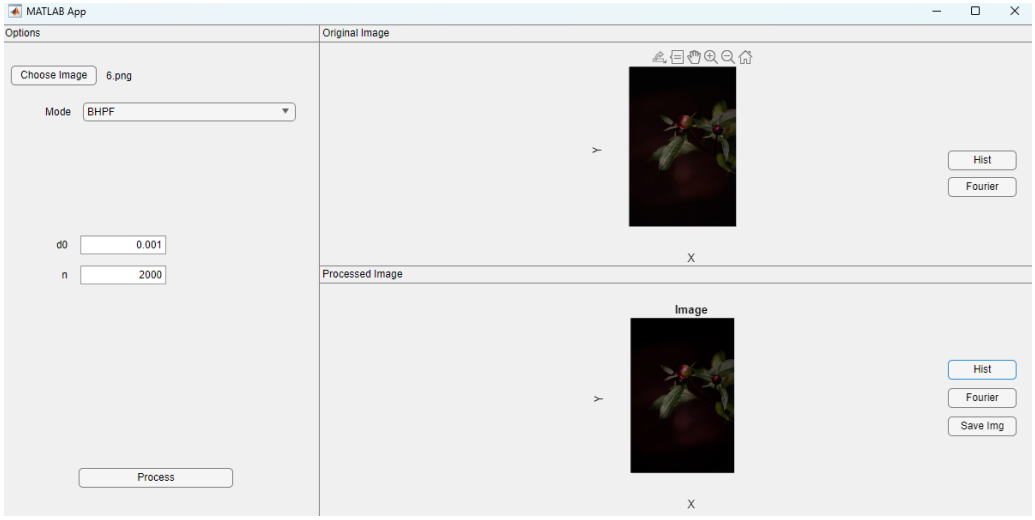
Diterangkan dengan BHPF, nilai d0 kecil dan nilai n besar



Gambar 2



Metode sama seperti gambar sebelumnya



e. Program 5 : Derau Salt and Pepper dan Penghilangan Derau

Berikut adalah implementasi filter pada ranah spasial.

```
function spatialFilterImg = spatialFilter(img, padding, type, kernelSize, d)
    %center point of the kernel
```

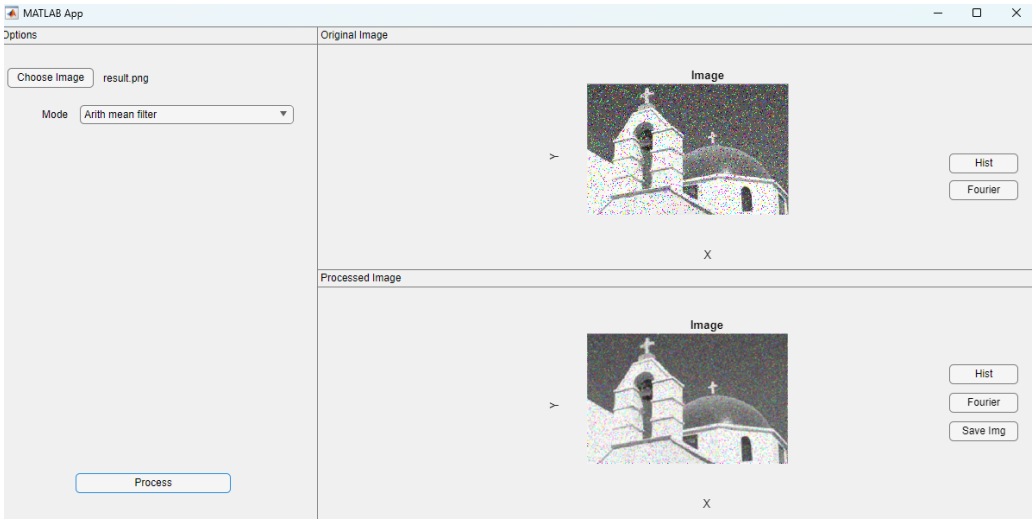
```
    kernelCenter = floor(kernelSize / 2);
    %image dimension
    imgSize = size(img);
    channels = imgSize(3);

    spatialFilterImg = img;

    for ch = 1 : channels
        %change img to float
        processedImg = img(:,:,ch);
        %reshape the size of image based on chosen padding
        if (padding == 'fillzero')
            processedImg = padarray(processedImg, [kernelCenter, kernelCenter],
0, "both");
        end
        processedImg = double(processedImg);
        for i=1:imgSize(1) - kernelCenter - 1
            for j=1:imgSize(2) - kernelCenter - 1
                extractedImage = processedImg(i : i + kernelSize - 1 , j : j +
kernelSize - 1);
                res = 0;
                if (type=="arithmetic_mean")
                    res = mean(extractedImage, "all");
                elseif (type=="geo_mean")
                    res = geomean(extractedImage, "all");
                elseif (type=="harmonic_mean")
                    res = harmmean(extractedImage, "all");
                elseif (type == "max")
                    res = max(extractedImage,[], "all");
                elseif (type == "min")
                    res = min(extractedImage, [], "all");
                elseif (type == "midpoint")
                    res = (max(extractedImage, [], "all") + min(extractedImage, [],
"all")) / 2;
                elseif (type == "alpha_trimmed_mean")
                    d = floor(d/2);
                    trimmedExtractedImage = extractedImage;
                    for i=1:d
                        vectorizedTrim = trimmedExtractedImage(:);
                        [maxVal, maxIndex] = max(vectorizedTrim);
                        [row, col] = ind2sub(size(trimmedExtractedImage),
maxIndex);
                        linearIdx = sub2ind(size(trimmedExtractedImage), row, col);
                        vectorizedTrim(linearIdx) = [];
                        res = mean(vectorizedTrim, "all");
                    end
```

```
            end
            spatialFilterImg(i + kernelCenter, j + kernelCenter, ch) = res;
        end
    end
end
spatialFilterImg = uint8(spatialFilterImg);
end
```

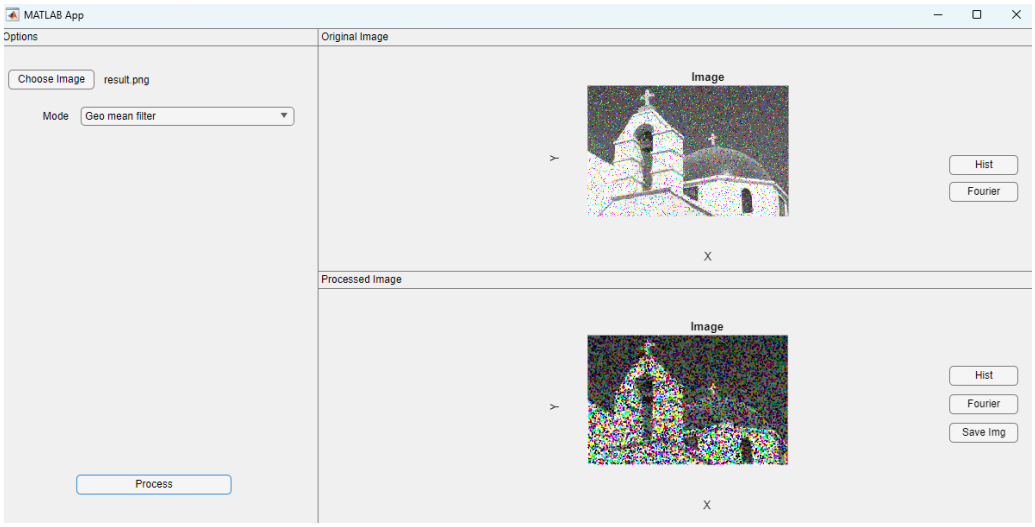
Jenis filter yang diimplementasi adalah arithmetic mean, geometric mean, harmonic mean, max, min, midpoint, dan alpha-trimmed mean.
Berikut adalah contoh penggunaannya.

Gambar 1 dengan salt and pepper

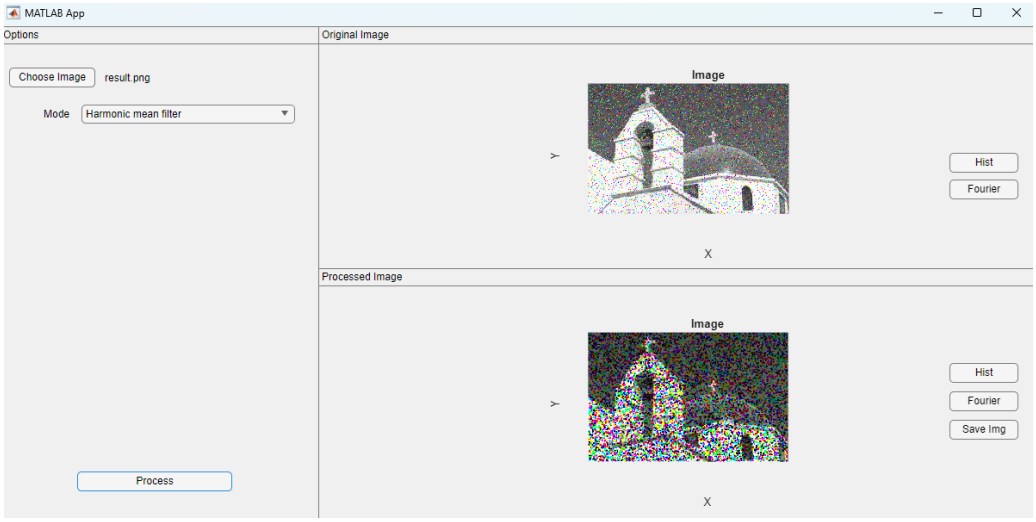
Arithmetic mean :



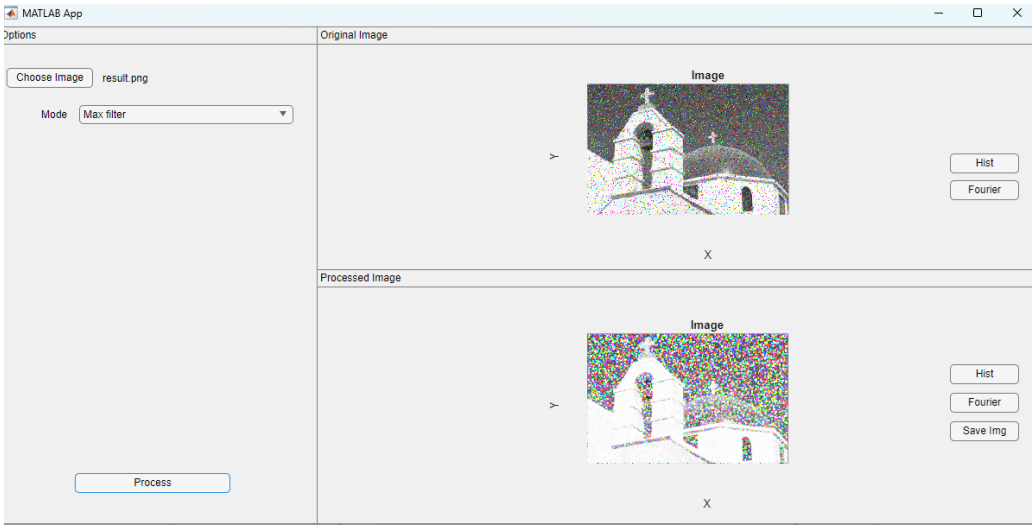
Geometric mean :



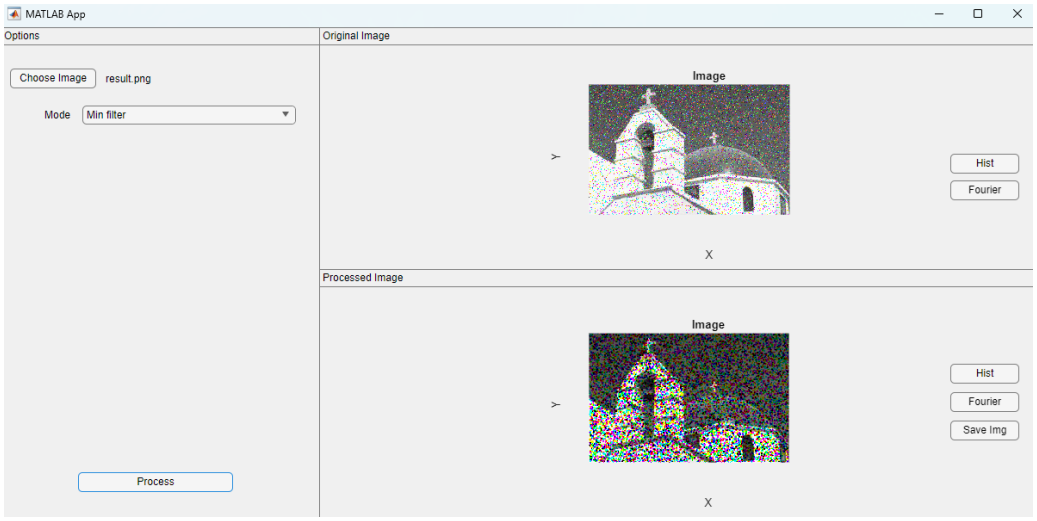
Harmonic mean :



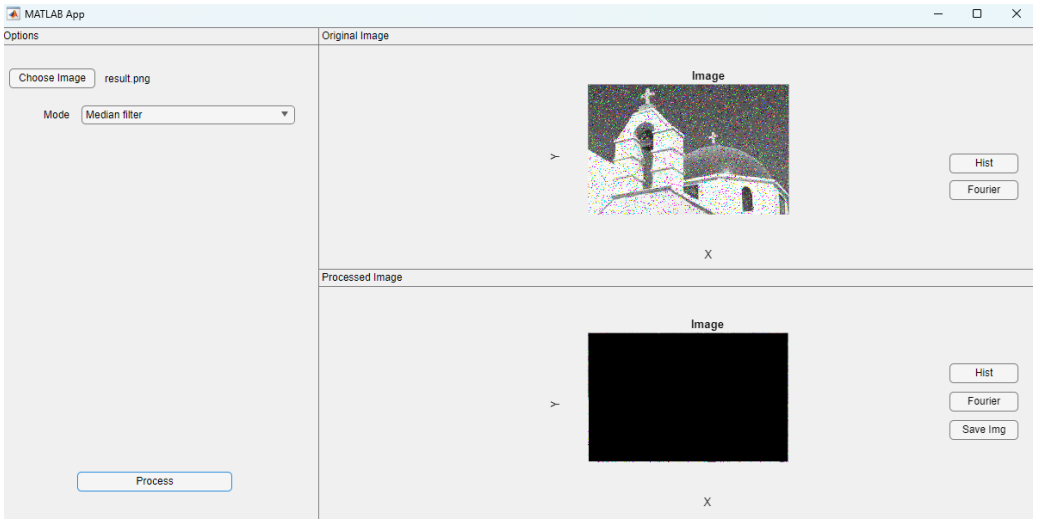
Max filter :



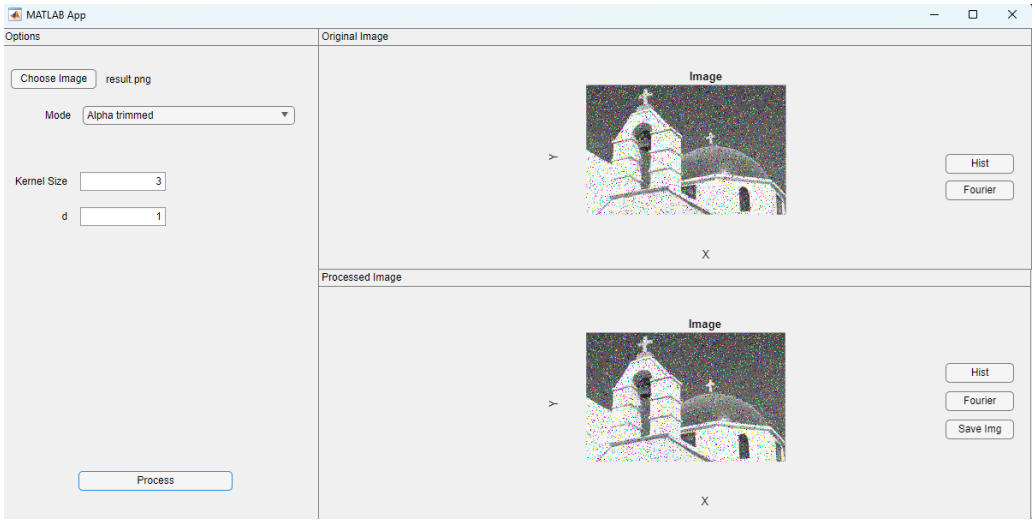
Min filter :



Median filter :

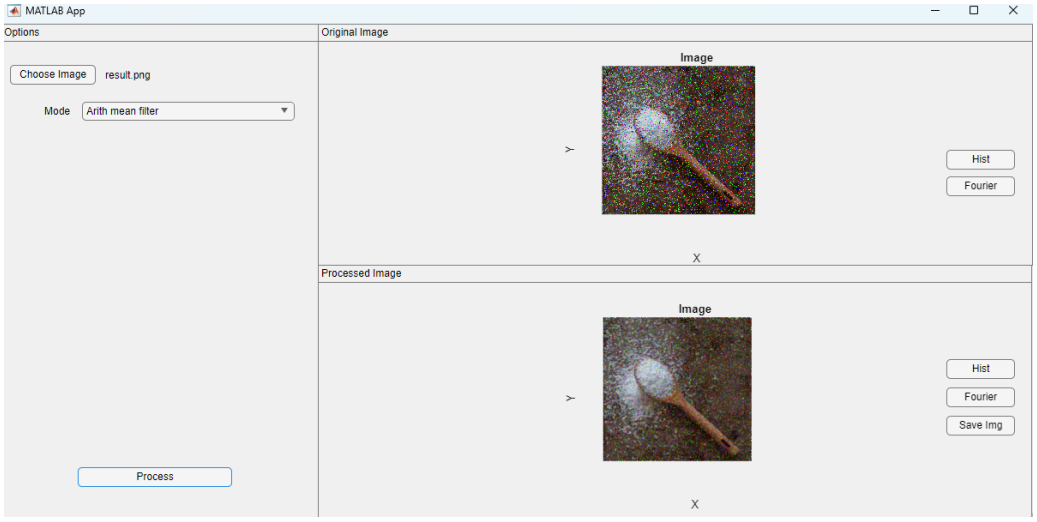


Alpha trimmed mean :

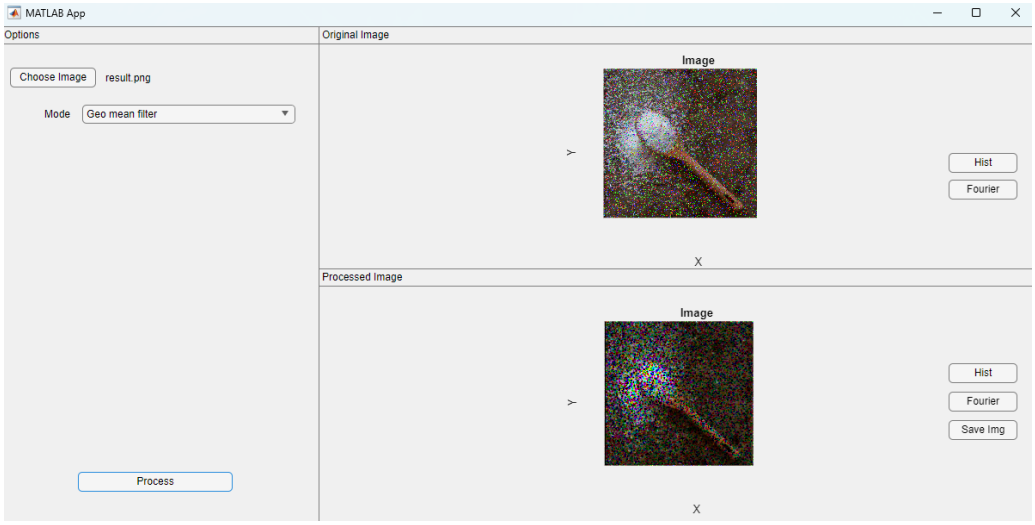


Gambar 2 dengan salt and pepper

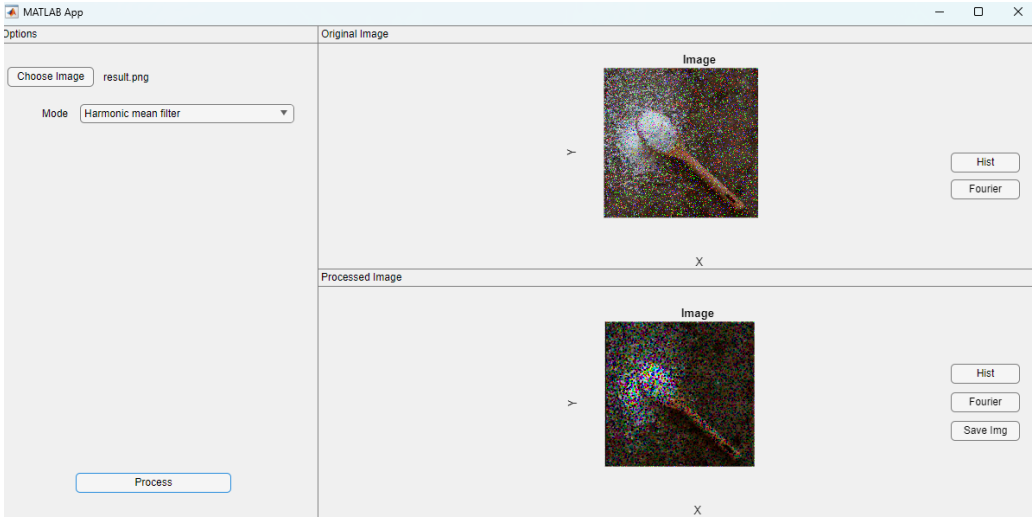
Arithmetic mean :



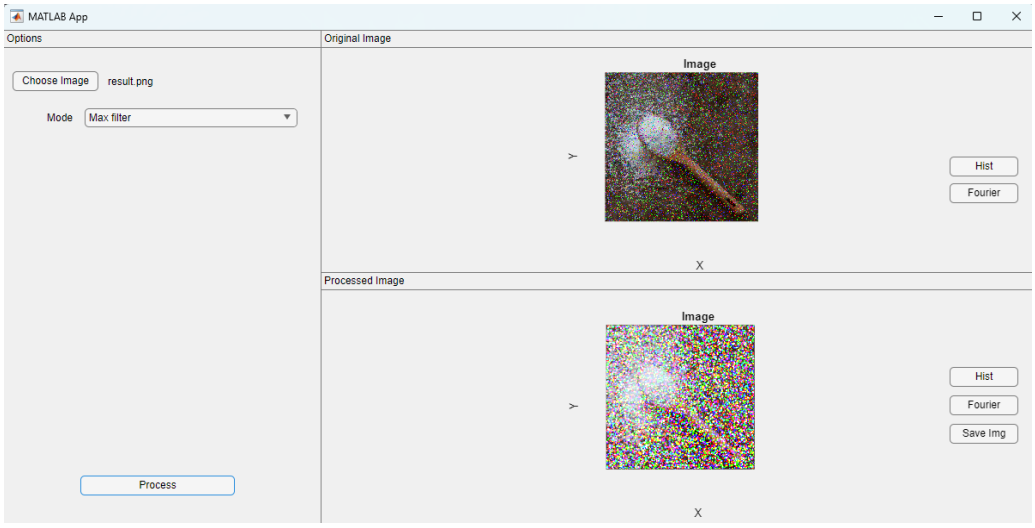
Geometric mean :



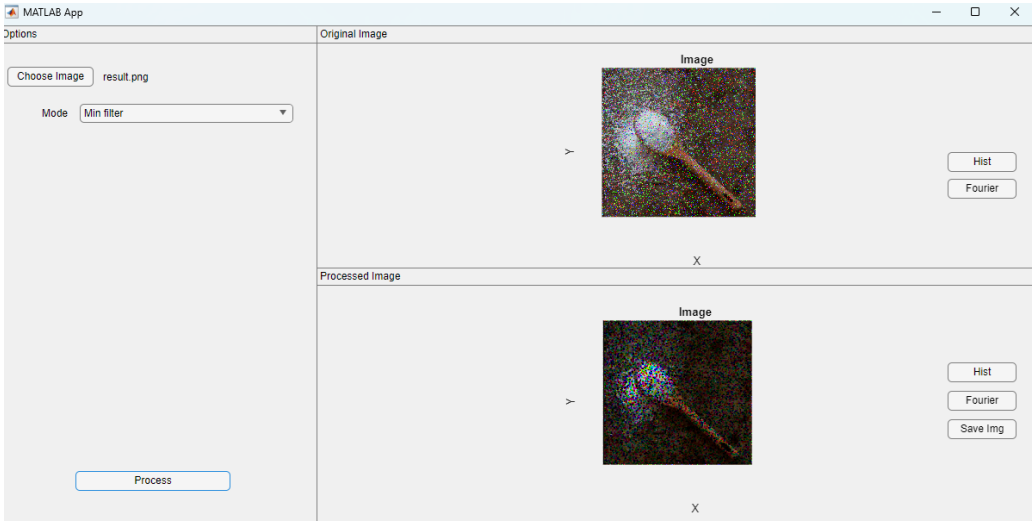
Harmonic mean :



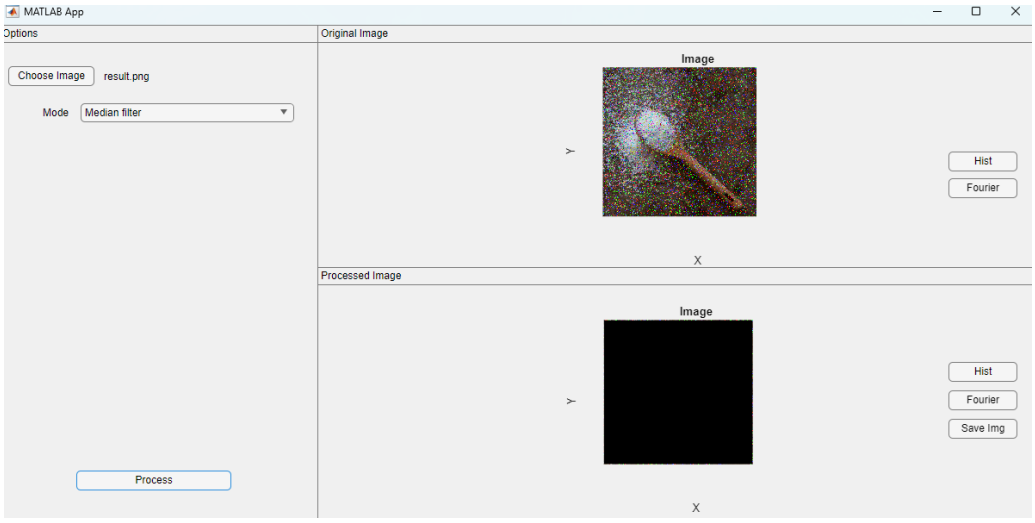
Max filter :



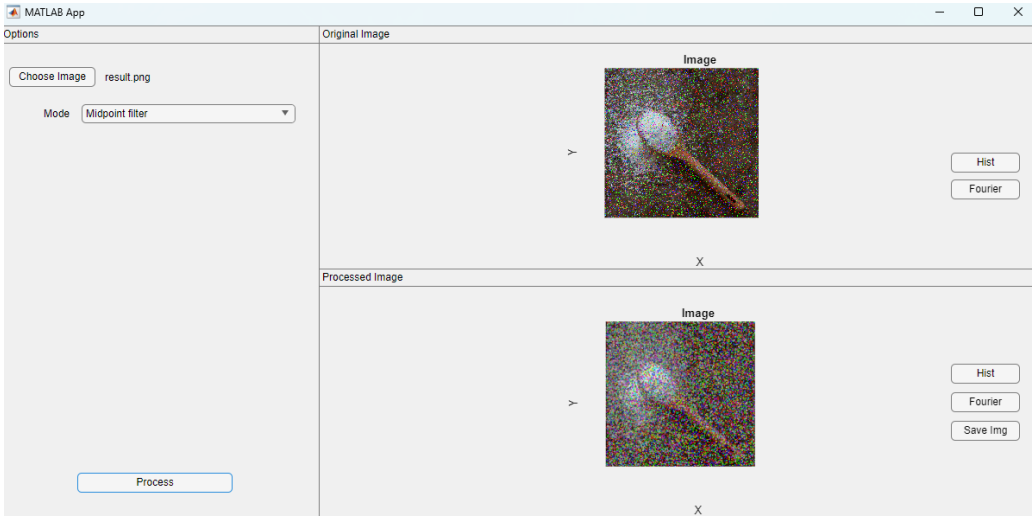
Min filter :



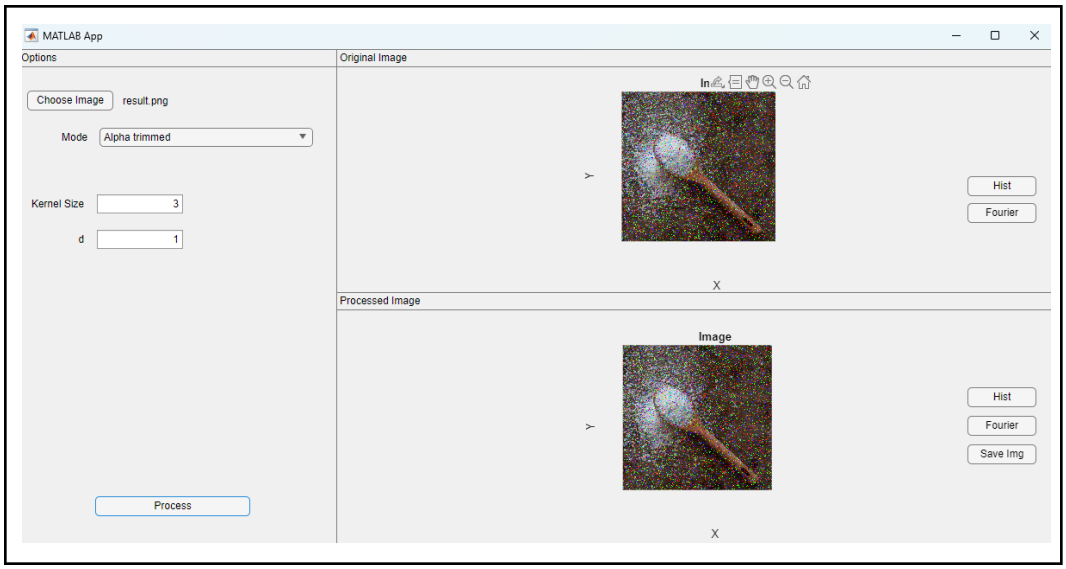
Median filter :



Midpoint filter :



Alpha-trimmed mean :



Dari hasil program yang dibuat, arithmetic mean filter merupakan filter relatif paling bagus untuk memperbaiki derau salt and pepper.

Alamat github : https://github.com/Haruray/tugas2-citra