Laporan Tugas 2
IF4073 Interpretasi dan Pengolahan Citra

Konvolusi & Domain Frekuensi

Disusun oleh:

Tanur Rizaldi Rahardjo / 13519214

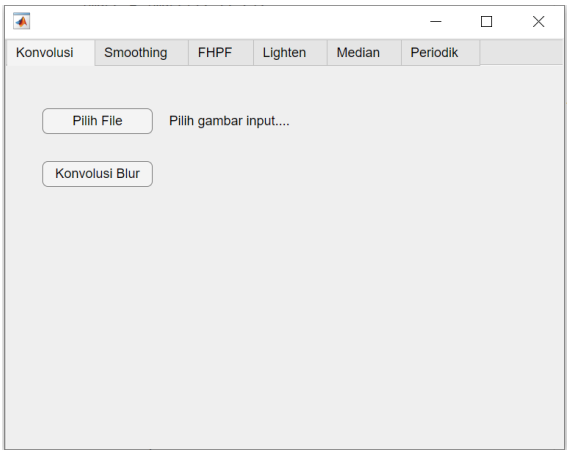
Program Studi Teknik Informatika

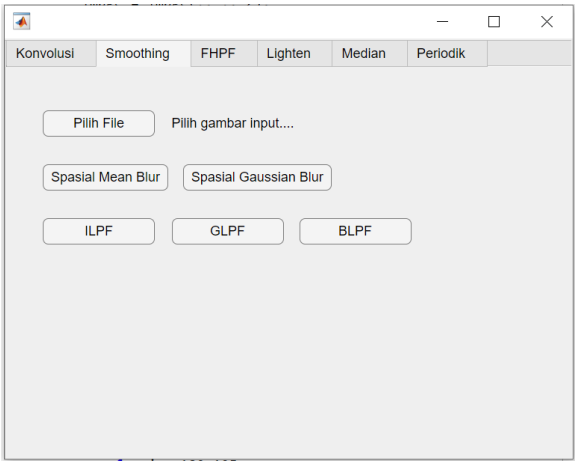
Sekolah Teknik Elektro dan Informatika

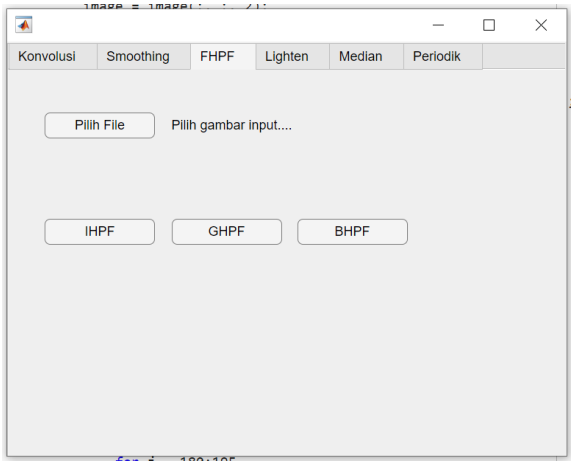
Institut Teknologi Bandung

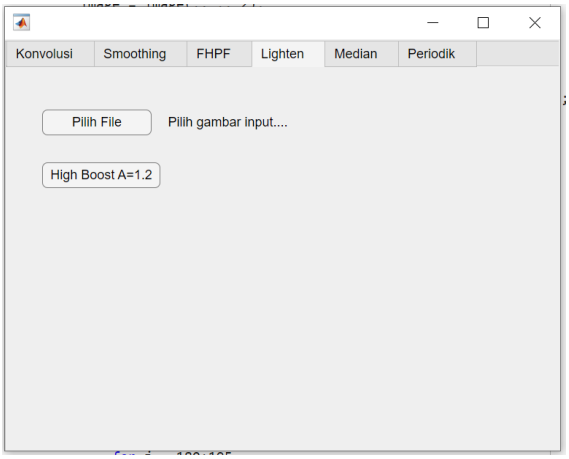
2022

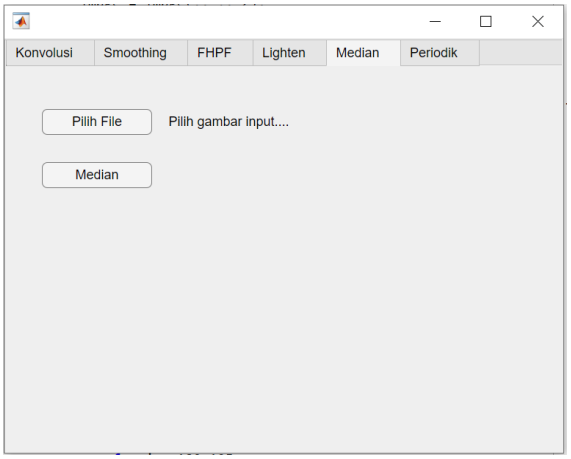
1. *Screenshot* GUI Program

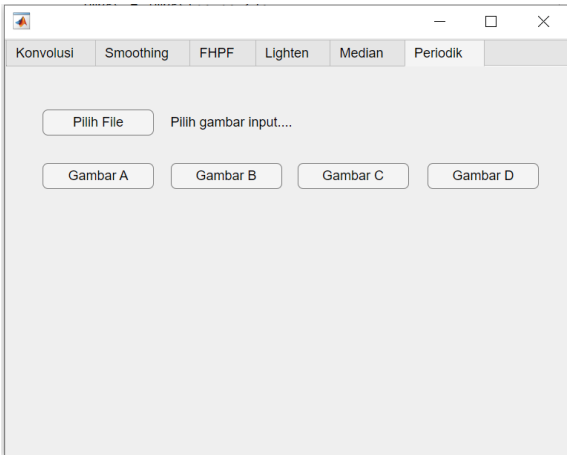












2. Rincian Setiap Program

2.1. Konvolusi

Kode Program

```
function res = Konvolusi(~, image, kernel)
    [numRow, numColumn, numColor] = size(image);
    res = zeros([numRow, numColumn, numColor]);
    [kernelSize, ~] = size(kernel);
    offsetK = floor(kernelSize / 2);

    boundLower = 1 + offsetK;
    rowUpper = numRow - offsetK;
    colUpper = numColumn - offsetK;

    for i = boundLower:rowUpper
        for j = boundLower:colUpper
            for c = 1:numColor
                temp = single(0);
                for a = -offsetK:offsetK
                    for b = -offsetK:offsetK
                        temp = temp + single(image(i+a, j+b, c)) * kernel(1+offsetK+a, 1+offsetK+b);
                    end
                end
                res(i, j, c) = temp;
                res(i, j, c) = max(0, min(255, res(i, j, c)));
            end
        end
    end

    res = uint8(res);
end
```

```
    % Button pushed function: KonvolusiBlurButton
    function JalankanKonvolusi(comp, event)
        try
            image = imread(comp.PilihKonvolusiLabel.Text);

            % Initial image
            figure("Name", "Citra Input")
            imshow(image);

            % Konvolusi
            kernel = [0.12 0.12 0.12; 0.12 0.12 0.12; 0.12 0.12 0.12];
            res = comp.Konvolusi(image, kernel);
            figure("Name", "Citra Hasil")
            imshow(res);
        catch e
            msgbox(e.message, "Error", "error");
        end
    end
```

Contoh Hasil Eksekusi

Citra	Hasil Program Buatan Sendiri
	
	

Analisis

Sayangnya, karena keterbatasan waktu, program hanya melakukan satu tipe konvolusi yaitu Box Blur atau Mean Filter 3x3. Kode untuk konvolusi sendiri sudah independen dengan ukuran kernel, tetapi metode input untuk kernel belum ada pada GUI. Kernel Box Blur yang di-hard code dapat dilihat pada screenshot diatas, kernel tersebut dapat diubah ke kernel lain jika diinginkan. Edge handling yang digunakan adalah dibiarkan kosong.

2.2. Smoothing

Kode Program

```
        % Button pushed function: SpasialMeanBlurButton
        function JalankanMeanFilter(comp, event)
            try
                image = imread(comp.PilihSmoothingLabel.Text);

                % Initial image
                figure("Name", "Citra Input")
                imshow(image);

                % Konvolusi
                kernel = [0.12 0.12 0.12; 0.12 0.12 0.12; 0.12 0.12 0.12];
                res = comp.Konvolusi(image, kernel);
                figure("Name", "Citra Hasil")
                imshow(res);
            catch e
                msgbox(e.message, "Error", "error");
            end
        end
```

```
    % Button pushed function: SpasialGaussianBlurButton
    function JalankanGauss(comp, event)
        try
            image = imread(comp.PilihSmoothingLabel.Text);

            % Initial image
            figure("Name", "Citra Input")
            imshow(image);

            % Konvolusi
            kernel = [0.0625 0.125 0.0625; 0.125 0.25 0.125; 0.0625 0.125 0.0625];
            res = comp.Konvolusi(image, kernel);
            figure("Name", "Citra Hasil")
            imshow(res);
        catch e
            msgbox(e.message, "Error", "error");
        end
    end
```

```
% Button pushed function: ILPFButton
function JalankanILPF(comp, event)
    try
        image = imread(comp.PilihSmoothingLabel.Text);
        [M, N, ~] = size(image);
        D0 = 50;
        u = 0:(2*M-1);
        v = 0:(2*N-1);

        idx = find(u > M);
        u(idx) = u(idx) - 2*M;
        idy = find(v > N);
        v(idy) = v(idy) - 2*N;

        [V, U] = meshgrid(v, u);
        D = sqrt(U.^2 + V.^2);
        H = double(D <= D0);
        comp.ApplyFrequencyFilter(image, H);
    catch e
        msgbox(e.message, "Error", "error");
    end
end
```

```
% Button pushed function: GLPFButton
function JalankanGLPF(comp, event)
    try
        image = imread(comp.PilihSmoothingLabel.Text);
        [M, N, ~] = size(image);
        D0 = 50;
        u = 0:(2*M-1);
        v = 0:(2*N-1);

        idx = find(u > M);
        u(idx) = u(idx) - 2*M;
        idy = find(v > N);
        v(idy) = v(idy) - 2*N;

        [V, U] = meshgrid(v, u);
        D = sqrt(U.^2 + V.^2);
        H = exp(-(D.^2)./(2*(D0^2)));
        comp.ApplyFrequencyFilter(image, H);
    catch e
        msgbox(e.message, "Error", "error");
    end
end
```

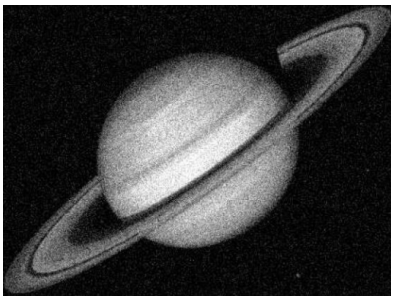
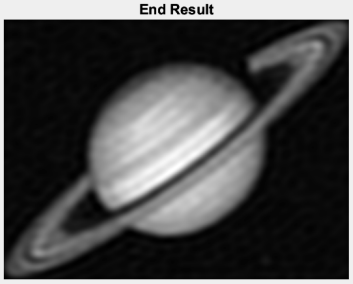
```
% Button pushed function: BLPFButton
function JalankanBLPF(comp, event)
    try
        image = imread(comp.PilihSmoothingLabel.Text);
        [M, N, ~] = size(image);
        D0 = 50;
        u = 0:(2*M-1);
        v = 0:(2*N-1);

        idx = find(u > M);
        u(idx) = u(idx) - 2*M;
        idy = find(v > N);
        v(idy) = v(idy) - 2*N;

        [V, U] = meshgrid(v, u);
        D = sqrt(U.^2 + V.^2);
        n = 2;
        H = 1./(1 + (D./D0).^(2*n));
        comp.ApplyFrequencyFilter(image, H);
    catch e
        msgbox(e.message, "Error", "error");
    end
end
```

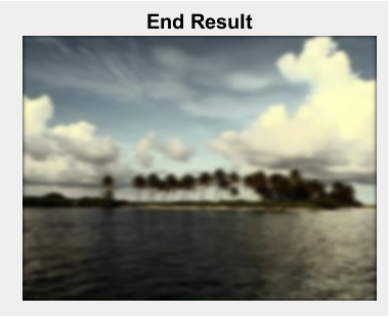
```
function res = ApplyFrequencyFilter(~, image, mask)
    % Initial image
    figure("Name", "Citra Input")
    imshow(image);

    % Pad
    [M, N, c] = size(image);
    padded = uint8(zeros(2*M, 2*N, c));

    for i = 1:2*M
        for j = 1:2*N
            for a = 1:c
              if i <= M && j <= N
                    padded(i, j, a) = image(i, j, a);
              else
                    padded(i, j, a) = 0;
              end
            end
        end
    end
    figure;
    imshow(padded);

    % FT
    Fc = fftshift(fft2(im2double(padded)));
    S2 = log(1+abs(Fc));
    figure, imshow(S2, []);

    F = fft2(im2double(padded));

    % Mask Application
    H = mask;
    H = fftshift(H); figure;imshow(H);title('LPF Ideal Mask');
```

```
    % Mask Application
    H = mask;
    H = fftshift(H); figure;imshow(H);title('LPF Ideal Mask');
    figure, mesh(H);

    % Applying Mask
    H = ifftshift(H);
    FF_f = H.*F;

    FF_f2 = real(ifft2(FF_f));
    figure; imshow(FF_f2); title("Padded Result");

    FF_f2 = FF_f2(1:M, 1:N);
    figure, imshow(FF_f2); title("End Result");

    res = FF_f2;
end
```

Contoh Hasil Eksekusi

Gambar Input	Hasil Eksekusi	Metode Filter
		Spatial Mean Filter / Spatial Box Blur

		Spatial Gaussian Filter
		ILPF
		GLPF
		BLPF

		ILPF
		ILPF
		GLPF

Analisis

Sesuai dengan yang diajarkan dikelas, filter-filter bekerja sesuai dengan deskripsi kelemahan dan kekuatannya. Ringing effect dari ILPF terlihat jelas pada hasil akhir gambar.

2.3. Frekuensi HPF

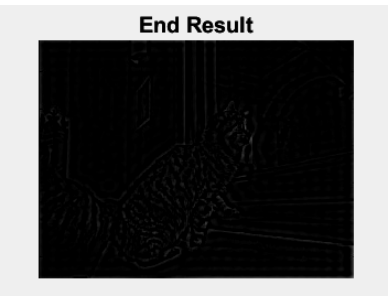
Kode Program

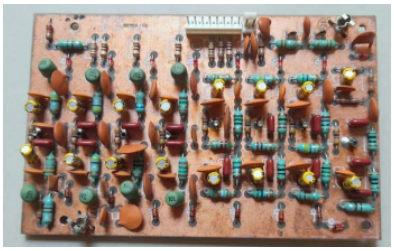
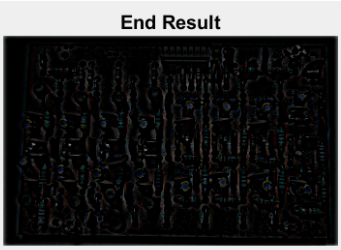
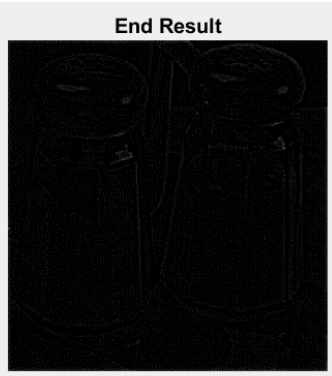
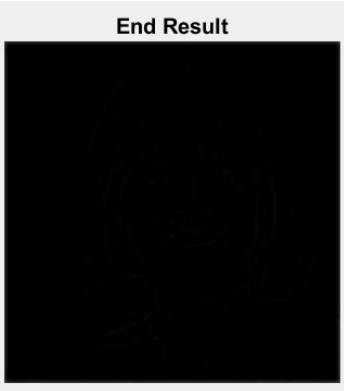
```
        % Button pushed function: IHPFButton
        function JalankanIHPF(comp, event)
            try
                image = imread(comp.PilihFrequencyLabel.Text);
                [M, N, ~] = size(image);
                D0 = 50;
                u = 0:(2*M-1);
                v = 0:(2*N-1);

                idx = find(u > M);
                u(idx) = u(idx) - 2*M;
                idy = find(v > N);
                v(idy) = v(idy) - 2*N;

                [V, U] = meshgrid(v, u);
                D = sqrt(U.^2 + V.^2);
                H = double(D <= D0);
                H = 1 - H;
                comp.ApplyFrequencyFilter(image, H);
            catch e
                msgbox(e.message, "Error", "error");
            end
        end
```

```
% Button pushed function: GHPFButton
function JalankanGHPF(comp, event)
    try
        image = imread(comp.PilihFrequencyLabel.Text);
        [M, N, ~] = size(image);
        D0 = 50;
        u = 0:(2*M-1);
        v = 0:(2*N-1);

        idx = find(u > M);
        u(idx) = u(idx) - 2*M;
        idy = find(v > N);
        v(idy) = v(idy) - 2*N;

        [V, U] = meshgrid(v, u);
        D = sqrt(U.^2 + V.^2);
        H = exp(-(D.^2)./(2*(D0^2)));
        H = 1 - H;
        comp.ApplyFrequencyFilter(image, H);
    catch e
        msgbox(e.message, "Error", "error");
    end
end
```

```
% Button pushed function: BHPFButton
function JalankanBHPF(comp, event)
    try
        image = imread(comp.PilihFrequencyLabel.Text);
        [M, N, ~] = size(image);
        D0 = 50;
        u = 0:(2*M-1);
        v = 0:(2*N-1);

        idx = find(u > M);
        u(idx) = u(idx) - 2*M;
        idy = find(v > N);
        v(idy) = v(idy) - 2*N;

        [V, U] = meshgrid(v, u);
        D = sqrt(U.^2 + V.^2);
        n = 2;
        H = 1./(1 + (D./D0).^(2*n));
        H = 1 - H;
        comp.ApplyFrequencyFilter(image, H);
    catch e
        msgbox(e.message, "Error", "error");
    end
end
```

Contoh Hasil Eksekusi

Gambar Input	Hasil	Metode
		IHPF

		GHPF
		BHPF
		BHPF
		BHPF

Analisa

Untuk bagian ini sangat mirip dengan LPF domain frekuensi pada bagian sebelumnya sehingga menggunakan fungsi ApplyFrequencyFilter sebelumnya tetapi menggunakan mask yang dimodifikasi.

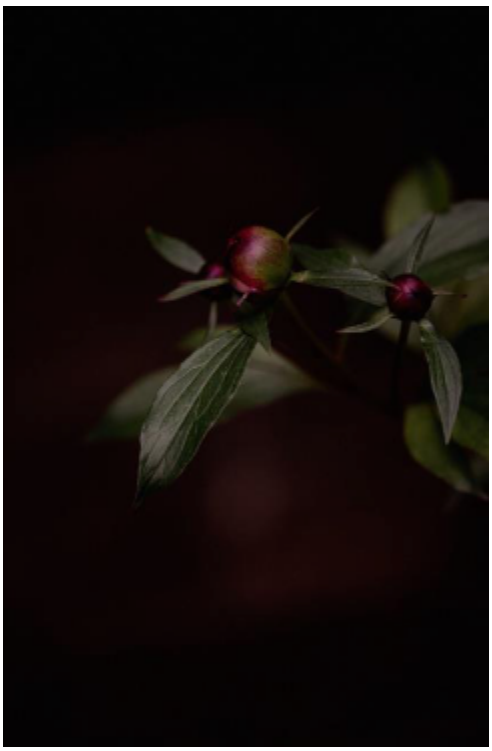
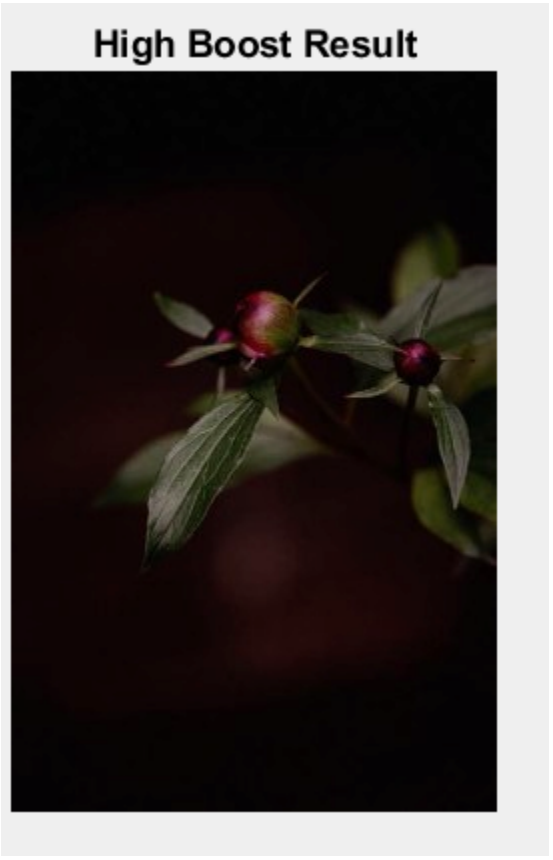
2.4. High Boost Frequency Domain

Kode Program

```
% Button pushed function: HighBoost
function JalankanFreqHB(comp, event)
    try
        image = imread(comp.PilihLightenLabel.Text);
        [M, N, ~] = size(image);
        D0 = 50;
        u = 0:(2*M-1);
        v = 0:(2*N-1);

        idx = find(u > M);
        u(idx) = u(idx) - 2*M;
        idy = find(v > N);
        v(idy) = v(idy) - 2*N;

        [V, U] = meshgrid(v, u);
        D = sqrt(U.^2 + V.^2);
        n = 2;
        H = 1./(1 + (D./D0).^(2*n));
        BLPFSource = comp.ApplyFrequencyFilter(image, H);
        hbimage = 1.2*image - uint8(BLPFSource);

        figure, imshow(hbimage); title("High Boost Result");
    catch e
        msgbox(e.message, "Error", "error");
    end
end
```

Contoh Hasil Eksekusi

Gambar Input	Hasil

<table>
<tr>
<td>  </td>
<td> 
</td>
</tr>
<tr>
<td>  </td>
<td> 
</td>
</tr>
</table>

Analisa

High Boost Frequency Domain tetap menggunakan ApplyFrequencyFilter yang digunakan pada bagian sebelumnya. Untuk metode LPF yang digunakan adalah BLPF dan A = 1.2.

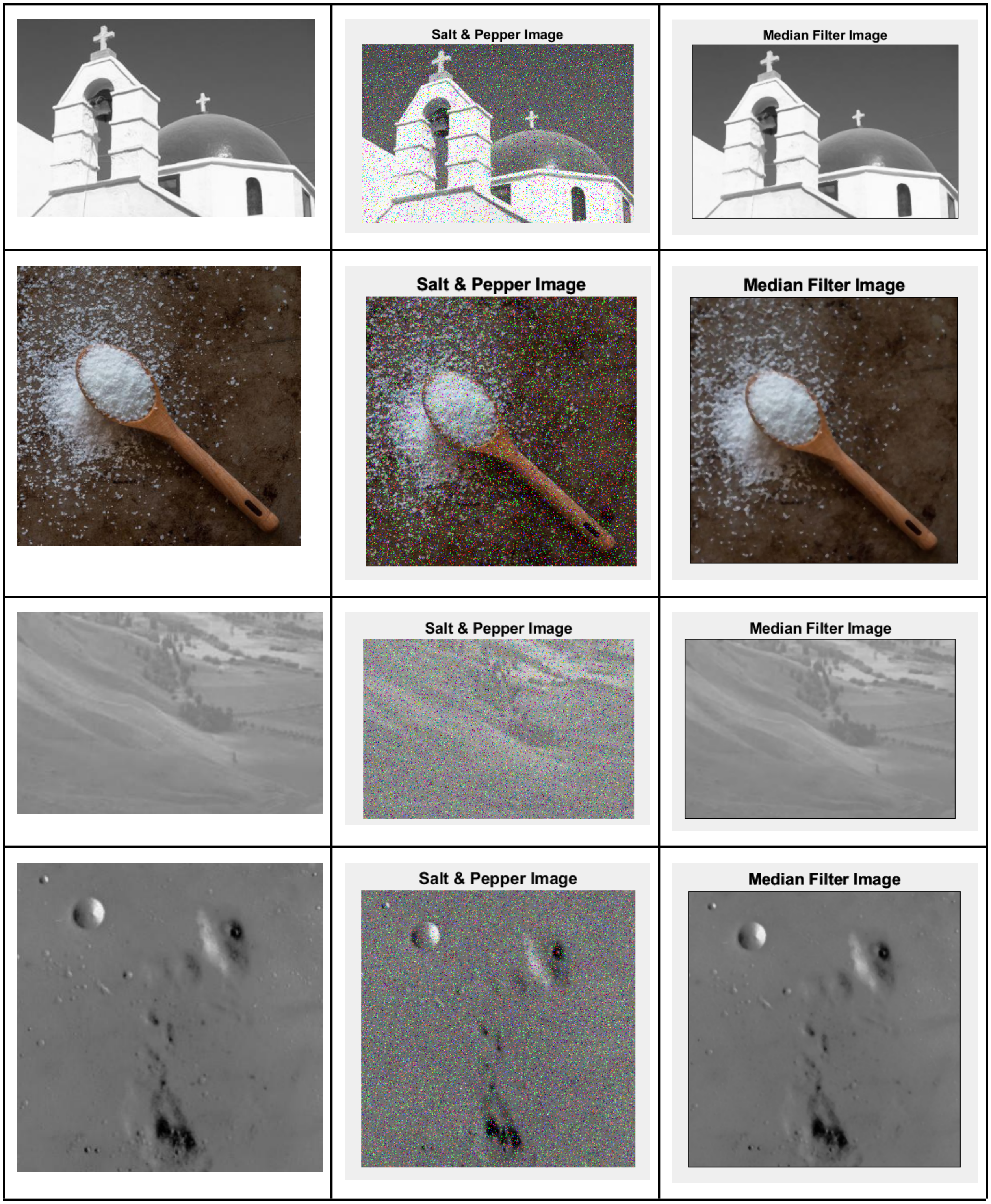
2.5. Median Filter

Kode Program

```
    % Button pushed function: Median
    function JalankanMedian(comp, event)
        try
            image = imread(comp.PilihMedianLabel.Text);
            figure, imshow(image); title("Citra Input");
            noised = imnoise(image, "salt & pepper", 0.1);
            figure, imshow(noised); title("Salt & Pepper Image");

            [row, col, c] = size(image);
            res = zeros([row, col, c]);
            for i = 2:row-1
                for j = 2:col-1
                    for c = 1:c
                        temp = zeros([3, 3]);
                        for a = -1:1
                            for b = -1:1
                                temp(2 + a, 2 + b) = image(i+a, j+b, c);
                            end
                        end
                        res(i, j, c) = median(reshape(temp.', 1, []));
                        res(i, j, c) = max(0, min(255, res(i, j, c)));
                    end
                end
            end
            res = uint8(res);

            figure, imshow(res); title("Median Filter Image");
        catch e
            msgbox(e.message, "Error", "error");
        end
    end
```

Contoh Hasil Eksekusi

Gambar input	Noise	Hasil


Salt & Pepper Image
Median Filter Image
Salt & Pepper Image
Median Filter Image
Salt & Pepper Image
Median Filter Image
Salt & Pepper Image
Median Filter Image

Analisa

Sesuai deskripsi pada kelas, median filter adalah filter non-linear yang cukup powerful untuk menghilangkan noise impuls seperti salt & pepper.

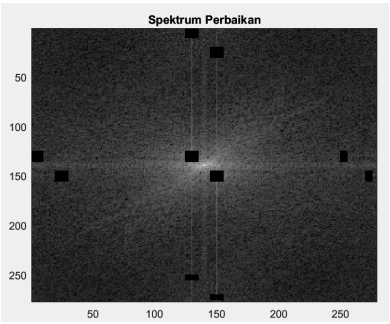
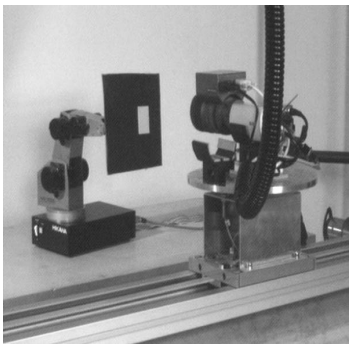
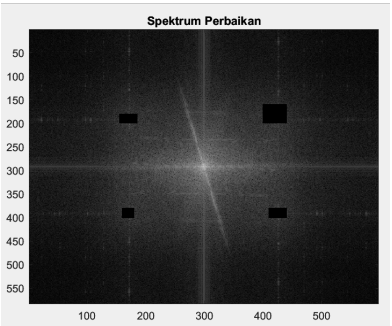
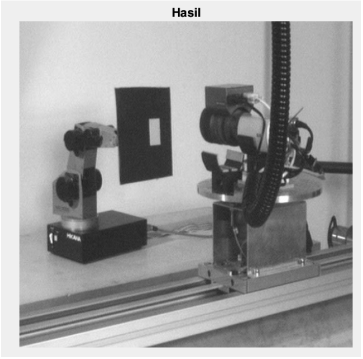
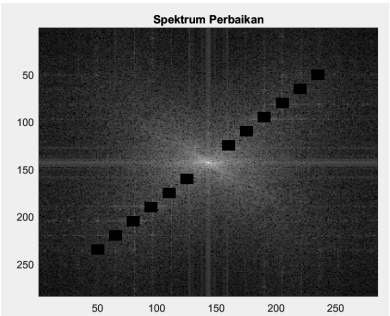
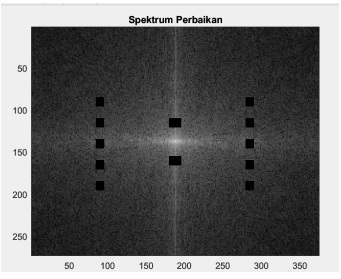
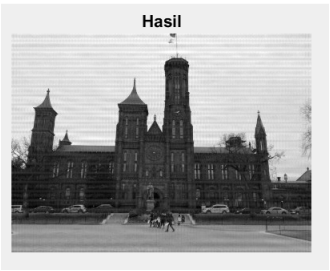
2.6. Periodic Noise

Kode Program

Berikut adalah contoh penghilangan noise periodik, terutama untuk gambar C

```
function JalankanGambarC(comp, event)
    try
        image = imread(comp.PilihPeriodikLabel.Text);
        image = image(:, :, 2);
        figure, imshow(image); title("Citra Input");

        F1 = fftshift(fft2(im2double(image)));
        F2 = log(1+abs(F1));
        figure, imagesc(100*F2); colormap("gray"); title("Spektrum");

        for nudge = 0:5
            for i = 230-nudge*15:240-nudge*15
                for j = 45+nudge*15:55+nudge*15
                    F1(i, j) = 0;
                end
            end
        end

        for nudge = 0:5
            for i = 120-nudge*15:130-nudge*15
                for j = 155+nudge*15:165+nudge*15
                    F1(i, j) = 0;
                end
            end
        end

        F2 = log(1+abs(F1));
        figure, imagesc(100*F2); colormap("gray"); title("Spektrum Perbaikan");

        res = real(ifft2(ifftshift(F1)));
        figure, imshow(res); title("Hasil");
    catch e
        msgbox(e.message, "Error", "error");
    end
```

Contoh Hasil Eksekusi

Gambar Input	Spektrum Hasil Koreksi	Hasil
		
		
		
		

Analisa

Untuk bagian ini, program ditulis secara manual untuk menghitamkan bagian-bagian pada domain frekuensi setelah menganalisanya secara manual.

3. Alamat Github Program

https://github.com/Lock1/IF4073-Image-2