Assignment no:5

1. **Aim:** To implement denoising and deblurring in frequency domain using Gray Image.
2. **Software Tool Used:** MATLAB
3. **Theory:**

1. Ideal Pass Filters:

Ideal Low Pass Filter (LPF): A filter that allows all frequency components below a certain cutoff frequency (fc) to pass without attenuation and completely blocks those above fc.

Ideal High Pass Filter (HPF): A filter that blocks all frequency components below a certain cutoff frequency (fc) and allows those above fc to pass without attenuation.

- Ideal Low Pass Filter (LPF):

$$H(f) = \begin{cases} 1, & |f| \leq f_c \\ 0, & |f| > f_c \end{cases}$$

- Ideal High Pass Filter (HPF):

$$H(f) = \begin{cases} 0, & |f| \leq f_c \\ 1, & |f| > f_c \end{cases}$$

In the **frequency domain**, ideal filters have a binary response (1 for passing frequencies and 0 for blocking frequencies). The **time domain** impulse response for ideal LPF is the sinc function: h(t)=sinc(2πfct) which extends infinitely in both directions, making it non-causal and impractical to implement.

2. Gaussian Low and High Pass Filters

Gaussian Low Pass Filter (LPF): A filter whose frequency response follows a Gaussian distribution, allowing a smooth transition between passband and stopband. It attenuates frequencies based on a Gaussian curve centred at 0.

Gaussian High Pass Filter (HPF): Derived from the Gaussian LPF by subtracting the low pass filter response from 1. It smoothly attenuates low frequencies while passing high frequencies.

Gaussian Low Pass Filter (LPF):

$$H(f) = e^{-\frac{f^2}{2\sigma^2}}$$

where σ controls the width of the Gaussian filter; a smaller σ leads to a sharper transition.

Gaussian High Pass Filter (HPF):

$$H(f) = 1 - e^{-\frac{f^2}{2\sigma^2}}$$

Gaussian filters offer a **smooth transition** between passband and stopband without the sharp cutoff of ideal filters. **In the time domain**, Gaussian filters are finite and therefore causal, making them practical to implement.

3. Butterworth Low and High Pass Filters

Butterworth Low Pass Filter (LPF): A filter designed to have a maximally flat frequency response in the passband, with no ripples, and a gradual roll-off into the stopband.

Butterworth High Pass Filter (HPF): Like the LPF, but with attenuation of low frequencies and a gradual increase in response above the cutoff frequency.

Butterworth Low Pass Filter (LPF):

$$H(f) = \frac{1}{\sqrt{1 + \left(\frac{f}{f_c}\right)^{2n}}}$$

where f is the frequency, f_c is the cutoff frequency, and n is the order of the filter. The higher the order, the steeper the roll-off.

Butterworth High Pass Filter (HPF):

$$H(f) = \frac{1}{\sqrt{1 + \left(\frac{f_c}{f}\right)^{2n}}}$$

Filter Order (n): Controls the sharpness of the filter's transition from passband to stopband. Higher-order filters provide a sharper cutoff. **Flat Passband** Butterworth filters have no ripples in the passband, making them ideal for applications where a smooth response is required.

The cutoff frequency in ideal, Gaussian, and Butterworth low/high-pass filters defines the boundary between passing and blocking frequencies:

- **Ideal Filter**: Sharp transition at the cutoff, perfectly passing frequencies below (low-pass) or above (high-pass) the cutoff. However, it causes ringing (Gibbs phenomenon) due to its abrupt nature.
- **Gaussian Filter**: Provides a smooth transition around the cutoff frequency, avoiding ringing, and offering a gradual reduction of frequencies beyond the cutoff.
- **Butterworth Filter**: Balances sharpness and smoothness, with a transition dependent on the filter order. Higher order results in a sharper cutoff, behaving more like the ideal filter.

In both low and high pass, the cutoff determines the extent of frequency components passed or attenuated, affecting image clarity and smoothness.

4. Result:

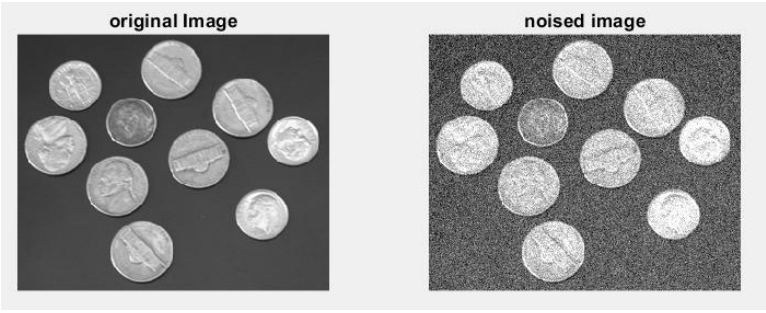

Fig (1) : Original and Gaussian noised image

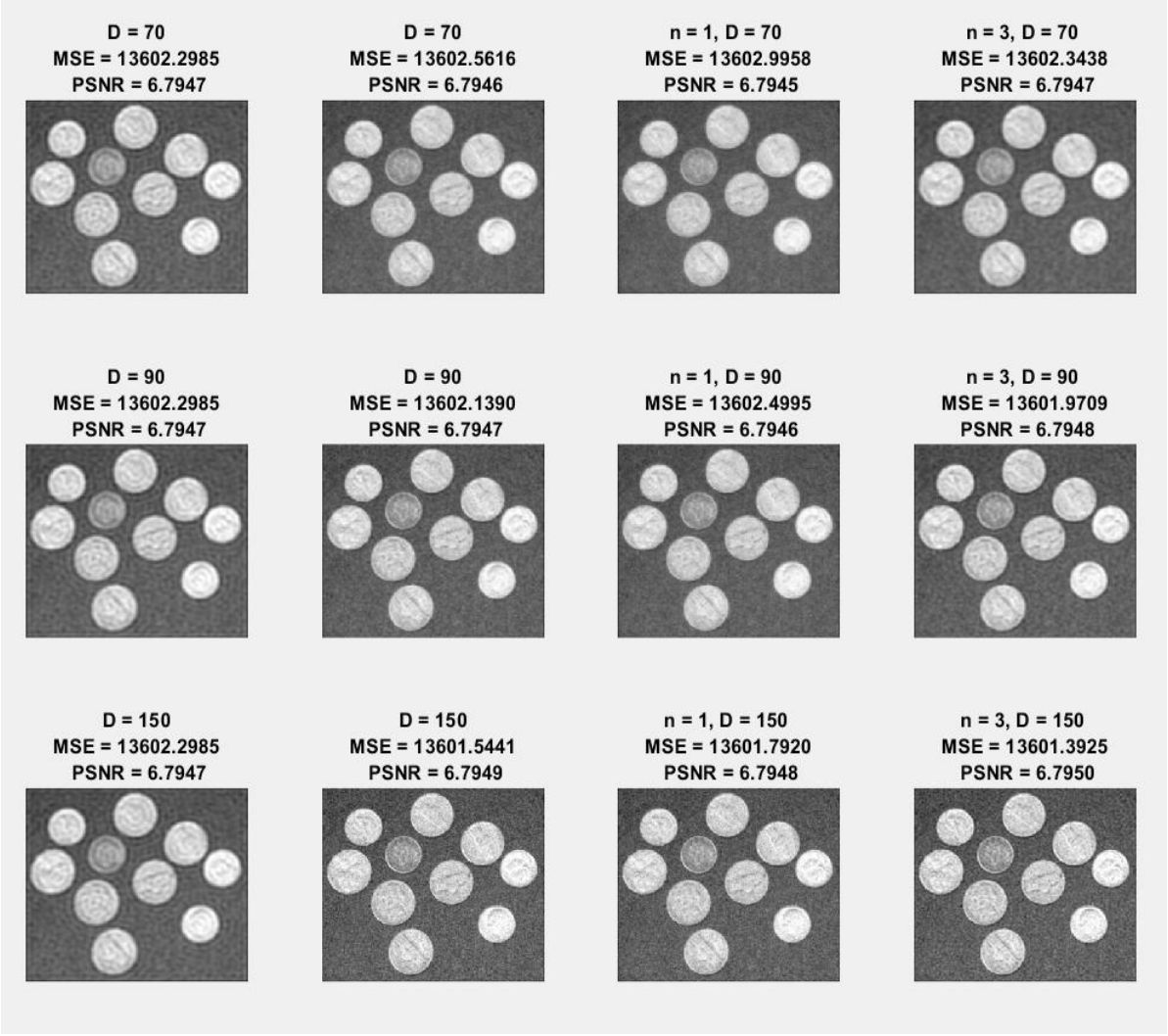

Fig (2): perform the Ideal, Gaussian, Butter-Worth low pass filter for different cutoff frequencies

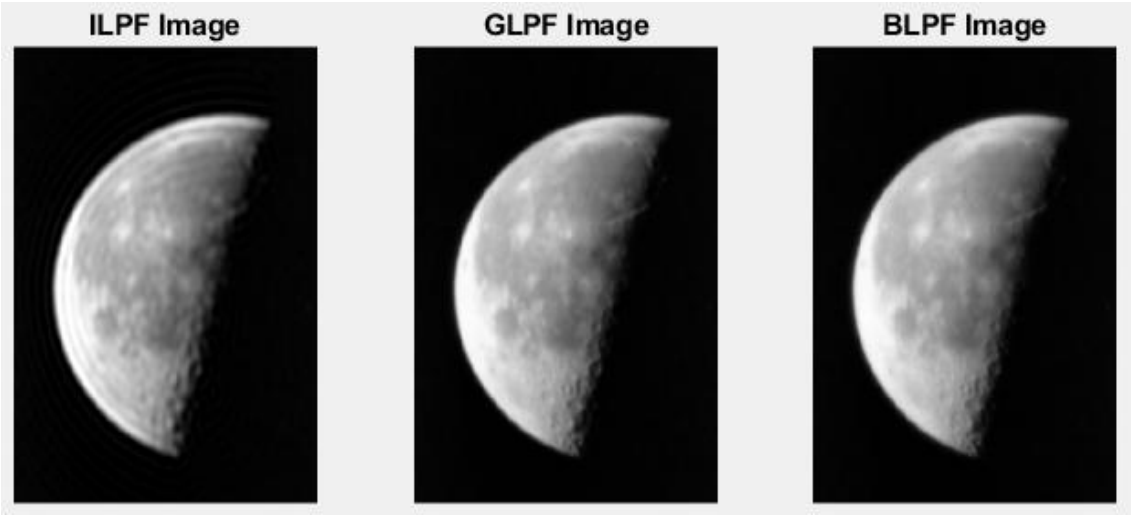

Fig (3) : perform Low pass filter on the Original Image

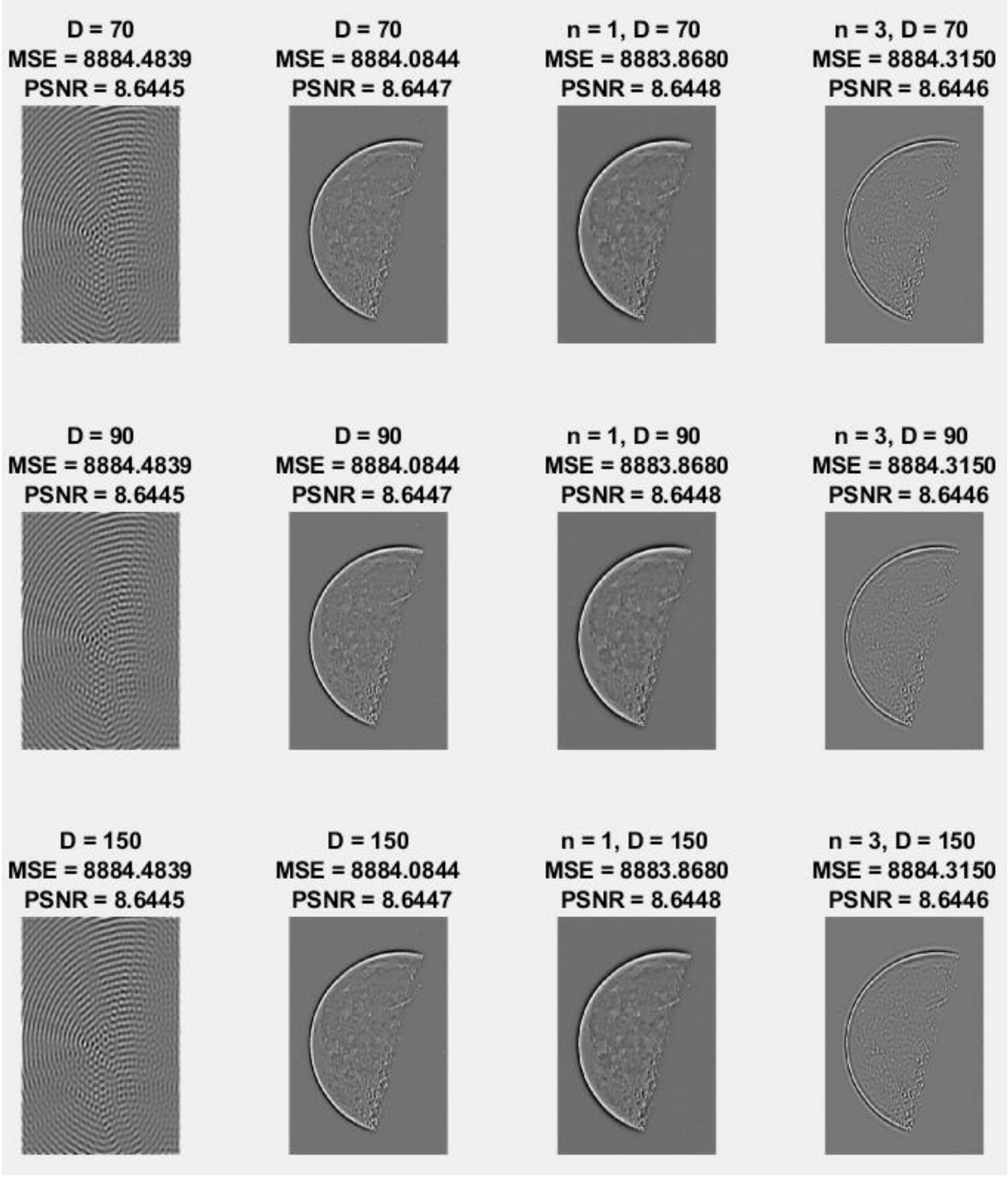

Fig (4): Perform the High Pass filter on the low pass blurred Image

5. Discussion:

Ideal Low and High Pass Filters:

Low Pass Filter (LPF): The ideal low pass filter with a sharp cutoff at 60, 90, and 150 produced a binary filtering effect, completely retaining the low-frequency components and removing the high-frequency noise.

1. At **60**, the image was heavily blurred, with much of the detail being lost, particularly fine textures.
2. At **90**, the image retained more detail compared to 60, but still showed a significant reduction in noise.

3. At **150**, the filter allowed more of the higher frequencies to pass through, resulting in clearer details but also retaining some noise.

High Pass Filter (HPF): The ideal high pass filter emphasizes edges and sharp transitions in the image.

1. At **60**, it highlighted fine edges and small details but left a noisy output, especially with speckle noise.
2. At **90** and **150**, more edges were accentuated with a cleaner image, although higher frequencies brought back some noise components.

Gaussian Low and High Pass Filters:

Low Pass Filter (LPF): The Gaussian filter provides a **smoother transition** between the passband and stopband, making it more effective in reducing both **Gaussian** and **speckle noise** without sharp cutoffs.

1. At **60**, the output was smoother but relatively more blurred than the ideal filter, especially in areas with fine details.
2. At **90**, the balance between noise reduction and detail preservation improved, and fine textures started to reappear.
3. At **150**, the image retained most details but started to reintroduce some noise components, although with less harshness than the ideal filter.

High Pass Filter (HPF): The Gaussian high pass filter smoothed out noise compared to the ideal high pass filter, making the edges less harsh but still pronounced.

1. At **60**, edges were highlighted, but with a cleaner appearance compared to the ideal filter.
2. At **90** and **150**, the edges were smoother, and the noise was less pronounced, maintaining a balance between sharpness and noise suppression.

Butterworth Low and High Pass Filters:

Low Pass Filter (LPF): The Butterworth filter, applied to the **Gaussian-noised image**, offers a compromise between ideal and Gaussian filters. It has a **gradual roll-off**, which smooths out noise while retaining a good amount of detail.

1. At **60**, the image was smoother than the ideal filter but less blurred than the Gaussian filter. Fine details were still lost, but the noise was well controlled.
2. At **90**, it achieved a better balance, suppressing noise while preserving more image detail compared to both the ideal and Gaussian filters.
3. At **150**, the Butterworth filter allowed more high frequencies, so more details were visible, though some noise started reappearing, albeit in a controlled manner.

High Pass Filter (HPF): For the high pass filter, you used the output of the low pass filtered original image. This approach helped retain more high-frequency content while reducing the overall noise.

1. The **60** cutoffs provided a less aggressive noise reduction, focusing more on enhancing finer details in the low pass filtered image.
2. At **90** and **150**, the high pass filtering further sharpened edges and details with a cleaner result compared to the ideal or Gaussian HPF, as the initial low pass filter already reduced the high-frequency noise.

Comparison:

Noise Reduction:

- The **Ideal LPF** provides the most aggressive noise reduction but at the cost of image detail, especially at lower cutoff frequencies like 60.
- The **Gaussian LPF** is smoother and provides better detail retention, particularly in images with Gaussian and speckle noise. It offers a trade-off between sharpness and noise reduction.
- The **Butterworth LPF** offers a middle ground, providing a smoother transition than the ideal filter, with better control over noise than the Gaussian filter.

Edge Preservation:

- The **Ideal HPF** is the most effective at edge enhancement but also enhances noise, especially for images with speckle noise.
- The **Gaussian HPF** provides a more natural edge enhancement by reducing noise and keeping the edges smooth.
- The **Butterworth HPF**, especially when applied after low-pass filtering, strikes a balance by enhancing edges while maintaining control over noise, providing a cleaner result compared to both the ideal and Gaussian HPF.

Filter Behaviour at Different Cutoff Frequencies:

- Lower cutoff frequencies (60) result in more aggressive filtering and smoothing, whereas higher cutoffs (150) allow more high-frequency details but may reintroduce noise.
- **Ideal filters** have the sharpest transitions but may lead to noticeable artifacts.
- **Gaussian filters** provide a smoother, more natural result but can sacrifice sharpness.
- **Butterworth filters** balance detail retention and noise suppression, with smoother results than ideal filters and more control than Gaussian filters.

6. Conclusion:

In conclusion, the application of Ideal, Gaussian, and Butterworth filters demonstrated varying effectiveness in noise reduction and detail preservation across different cutoff frequencies. The Ideal filters provided the sharpest frequency separation but at the cost of introducing artifacts and excessive detail loss. The Gaussian filters, with their smooth transition, offered better noise suppression with less sharpness in edge preservation. The Butterworth filters, striking a balance between the two, delivered superior control over noise and edge details, particularly at mid to high cutoff frequencies. Overall, the Butterworth filter was the most effective in maintaining image quality while reducing noise, making it well-suited for practical image processing tasks.

7. CODE:

Low pass filter:

```
I = imread("coins.png");
r = im2double(I);
noise = imnoise(r, "gaussian", 0.1);
[rows, cols] = size(noise);
noise_i = noise .* ((-1) .^ (meshgrid(1:cols, 1:rows) + meshgrid(1:rows, 1:cols).'));
padd_i = padarray(noise_i, [rows, cols], 0, 'post');
dft_mtx = fft2(padd_i);
D_values = [70, 90, 150];
```

```
function process_filter(filter, dft_mtx, I, rows, cols, D, n, subplot_idx)
    Guv = dft_mtx .* filter;
    idft_mtx = real(ifft2(Guv));
    idft_mtx = idft_mtx .* ((-1) .^ (meshgrid(1:2*cols, 1:2*rows) + meshgrid(1:2*rows,
1:2*cols).'));
    filter_img = idft_mtx(1:rows, 1:cols);
    mse = sum((double(filter_img(:)) - double(I(:))).^2) / (rows * cols);
    psnr = 10 * log10(255 * 255 / mse);
    subplot(3, 4, subplot_idx);
    imshow(filter_img);
    if n == 0
        title(sprintf(' D = %d \nMSE = %.4f\n PSNR = %.4f', D, mse, psnr));
    else
        title(sprintf('n = %d, D = %d \nMSE = %.4f\n PSNR = %.4f',n, D, mse, psnr));
    end
end
figure();
for k = 1:3
    D = D_values(k);
    ILP = zeros(2 * rows, 2 * cols);
    GLP = zeros(2 * rows, 2 * cols);
    BLP1 = zeros(2 * rows, 2 * cols);
    BLP3 = zeros(2 * rows, 2 * cols);
    for i = 1:2 * rows
        for j = 1:2 * cols
            Duv = sqrt((i - rows)^2 + (j - cols)^2);
            ILP(i, j) = Duv <= 70;
            GLP(i, j) = exp((-1) * (Duv^2) / (2 * (D^2)));
            BLP1(i, j) = 1 / (1 + (Duv / D)^(2));
            BLP3(i, j) = 1 / (1 + (Duv / D)^(6));
        end
    end
    process_filter(ILP, dft_mtx, I, rows, cols, D, 0, 4 * (k - 1) + 1);
    process_filter(GLP, dft_mtx, I, rows, cols, D, 0, 4 * (k - 1) + 2);
    process_filter(BLP1, dft_mtx, I, rows, cols, D, 1, 4 * (k - 1) + 3);
    process_filter(BLP3, dft_mtx, I, rows, cols, D, 3, 4 * (k - 1) + 4);
end
```

High pass filter:

```
I = imread("moon.tif");
r = im2double(I);
[rows, cols] = size(I);
r = r .* ((-1) .^ (meshgrid(1:cols, 1:rows) + meshgrid(1:rows, 1:cols).'));  % Preprocessing the
image
padd_i = padarray(r, [rows, cols], 0, 'post');
dft_mtx = fft2(padd_i);

function process_filter(filter, dft_mtx, I, rows, cols, D, n, subplot_idx)
    Guv = dft_mtx .* filter;
```

```
    idft_mtx = real(ifft2(Guv));
    idft_mtx = idft_mtx .* ((-1) .^ (meshgrid(1:2*cols, 1:2*rows) + meshgrid(1:2*rows,
1:2*cols).'));
    filter_img = idft_mtx(1:rows, 1:cols);
    mse = sum((double(filter_img(:)) - double(I(:))).^2) / (rows * cols);
    psnr = 10 * log10(255 * 255 / mse);
    subplot(3, 4, subplot_idx);
    imshow(filter_img, []);
    if n == 0
        title(sprintf(' D = %d \nMSE = %.4f\n PSNR = %.4f', D, mse, psnr));
    else
        title(sprintf('n = %d, D = %d \nMSE = %.4f\n PSNR = %.4f',n, D, mse, psnr));
    end
end

function process_high_pass(filter, blurr_img, I, rows, cols, c)
    blurr_img = blurr_img .* ((-1) .^ (meshgrid(1:cols, 1:rows) + meshgrid(1:rows, 1:cols).'));
    padd_i = padarray(blurr_img, [rows, cols], 0, 'post');
    dft_mtx = fft2(padd_i);
    D_values = [70, 90, 150];
    for k = 1:3
        D = D_values(k);
        if c == 1
            process_filter(filter, dft_mtx, I, rows, cols, D, 0, 4 * (k - 1) + 1);
        elseif c == 2
            process_filter(filter, dft_mtx, I, rows, cols, D, 0, 4 * (k - 1) + 2);
        elseif c == 3
            process_filter(filter, dft_mtx, I, rows, cols, D, 1, 4 * (k - 1) + 3);
        else
            process_filter(filter, dft_mtx, I, rows, cols, D, 3, 4 * (k - 1) + 4);
        end
    end
end
D = 70; % Low pass cutoff frequency
ILP = zeros(2 * rows, 2 * cols);
GLP = zeros(2 * rows, 2 * cols);
BHP1 = zeros(2 * rows, 2 * cols);
BHP3 = zeros(2 * rows, 2 * cols);
IHP = ones(2*rows,2*cols);
GHP = ones(2*rows,2*cols);
BLP = zeros(2*rows,2*cols);
for i = 1:2 * rows
    for j = 1:2 * cols
        Duv = sqrt((i - rows)^2 + (j - cols)^2);
        ILP(i, j) = Duv <= D; % Ideal low pass
        GLP(i, j) = exp((-1) * (Duv^2) / (2 * (D^2))); % Gaussian low pass
        BLP(i, j) = 1 / (1 + (Duv / D)^(2));
        BHP1(i, j) = 1 / (1 + (D / Duv)^(2)); % Butterworth low pass (n=1)
        BHP3(i, j) = 1 / (1 + ( D/ Duv)^(6)); % Butterworth low pass (n=3)
    end
```

```
end
Guv = dft_mtx .* ILP;
idft_mtx = real(ifft2(Guv));
idft_mtx = idft_mtx .* ((-1) .^ (meshgrid(1:2*cols, 1:2*rows) + meshgrid(1:2*rows,
1:2*cols).'));
ideal_low = idft_mtx(1:rows, 1:cols);

Guv = dft_mtx .* GLP;
idft_mtx = real(ifft2(Guv));
idft_mtx = idft_mtx .* ((-1) .^ (meshgrid(1:2*cols, 1:2*rows) + meshgrid(1:2*rows,
1:2*cols).'));
Gaussian_low = idft_mtx(1:rows, 1:cols);

Guv = dft_mtx .* BLP;
idft_mtx = real(ifft2(Guv));
idft_mtx = idft_mtx .* ((-1) .^ (meshgrid(1:2*cols, 1:2*rows) + meshgrid(1:2*rows,
1:2*cols).'));
butter_low = idft_mtx(1:rows, 1:cols);
figure();
subplot(1,3,1);
imshow(ideal_low);
title("BLPF Image");
subplot(1,3,2);
imshow(Gaussian_low);
title("BLPF Image");
subplot(1,3,3);
imshow(butter_low);
title("BLPF Image");
GHP = GHP-GLP ;
IHP = IHP-ILP ;

process_high_pass(IHP, ideal_low, I, rows, cols,1);
process_high_pass(GHP, Gaussian_low, I, rows, cols,2);
process_high_pass(BHP1, butter_low, I, rows, cols,3);
process_high_pass(BHP3, butter_low, I, rows, cols,4);
```