PRACTICAL: - 8

AIM: Write MATLAB code to perform Min, Median & Max Filtering on grayscale image.

Software/ Hardware:

Objective: The objective of the experiment is to,

Syntax: Syntax to be covered in this experiment,

Theory:

1. **Min Filtering**: Replaces each pixel in the image with the minimum pixel value in its neighborhood. It is often used to reduce noise in images and preserve edges.

2. **Median Filtering**: Replaces each pixel with the median value in its neighborhood. It effectively removes noise while preserving edges better than mean or min filtering.

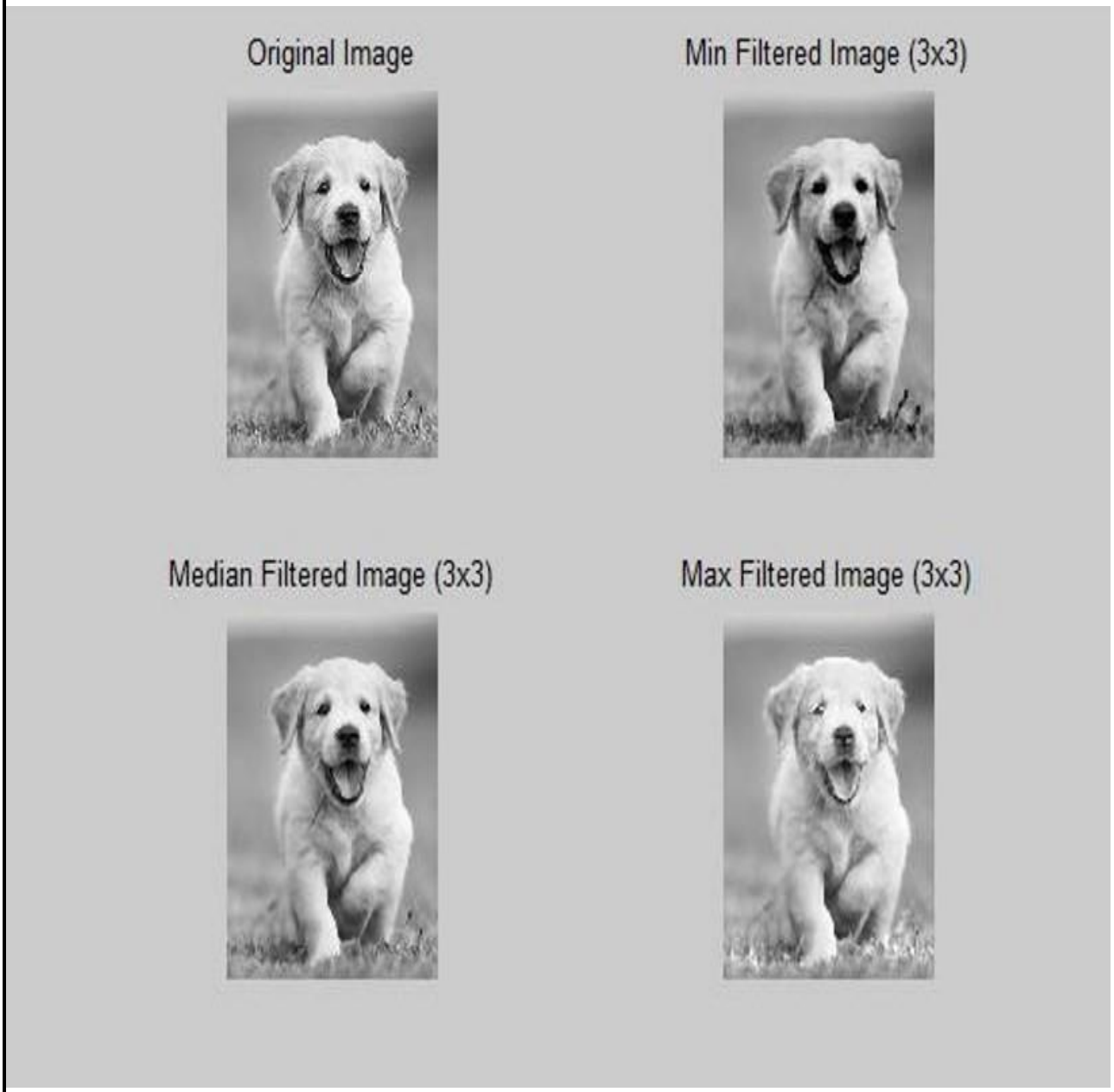
3. **Max Filtering**: Replaces each pixel with the maximum pixel value in its neighborhood. It can be used for tasks like finding the brightest regions in an image.

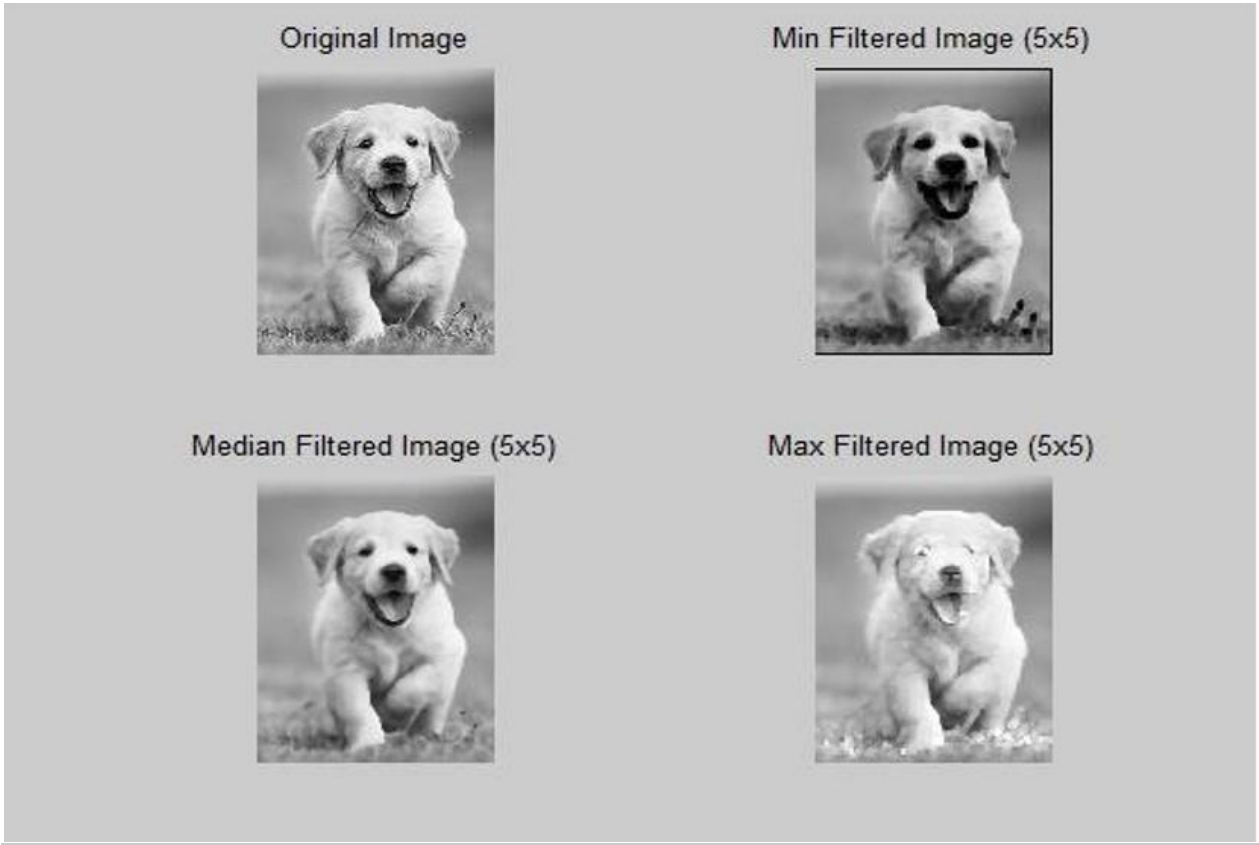
Code:

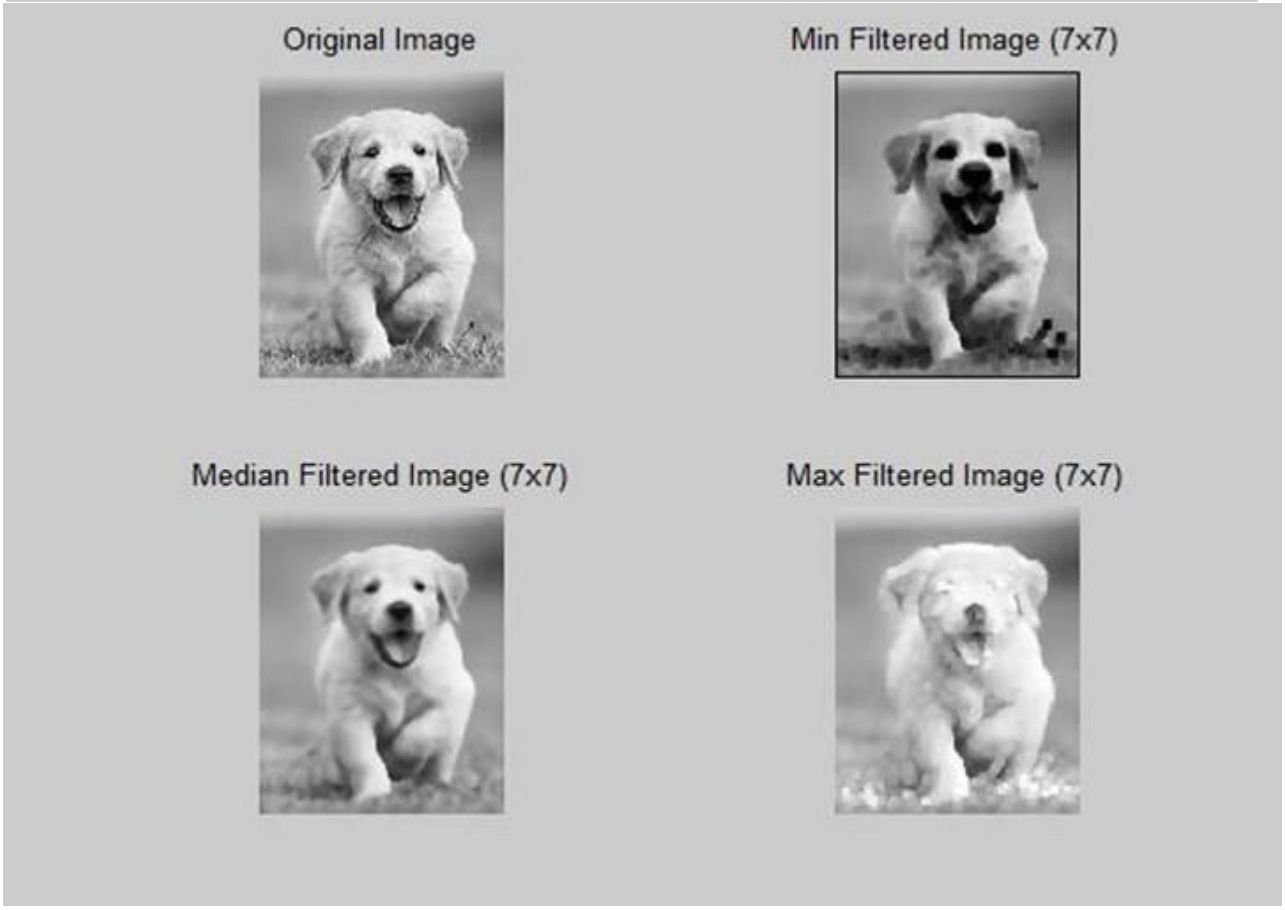
```
clc;
 clear all;
close all;

% Load the grayscale image

image_path = 'C:\Users\Ashok\Desktop\dog.jpg';

original_image = imread(image_path);

original_image = rgb2gray(original_image);
 % Convert to grayscale if needed

% Define filter sizes
filter_sizes = [3, 5, 7];

% You can change these sizes as needed

% Perform Min, Median, and Max filtering for each filter size for i = 1:length(filter_sizes)
% Min filtering
min_filtered_image = ordfilt2(original_image, 1, ones(filter_sizes(i)));

% Median filtering
median_filtered_image = medfilt2(original_image, [filter_sizes(i), filter_sizes(i)]);

% Max filtering
max_filtered_image = ordfilt2(original_image, filter_sizes(i)^2, ones(filter_sizes(i)));
% Display the results
figure;

subplot(2, 2, 1), imshow(original_image), title('Original Image');
```

```
subplot(2, 2, 2), imshow(min_filtered_image), title(['Min Filtered Image (', num2str(filter_sizes(i)), 'x', num2str(filter_sizes(i)), ')']);

subplot(2, 2, 3), imshow(median_filtered_image), title(['Median Filtered Image (', num2str(filter_sizes(i)), 'x', num2str(filter_sizes(i)), ')']);

subplot(2, 2, 4), imshow(max_filtered_image), title(['Max Filtered Image (', num2str(filter_sizes(i)), 'x', num2str(filter_sizes(i)), ')']);

end
```

Simulation Results: -







Conclusion: -

Min filtering darkens the image by retaining the smallest pixel values, median filtering reduces noise while preserving details, and max filtering brightens the image by retaining the largest pixel values. Each method enhances the image differently based on the filtering approach.

PRACTICAL: -9

AIM: Write MATLAB code to blur the image using Ideal low pass, Butterworth low pass and Gaussian Low pass filter.

Software/ Hardware:

Objective: The objective of the experiment is to,

- The goal is to blur an image using three types of filters: Ideal Low Pass, Butterworth Low Pass, and Gaussian Low Pass.
- Each filter will reduce high-frequency noise and details, making the image smoother.
- This will show how different filters impact the blurring effect in MATLAB.

Syntax: Syntax to be covered in this experiment,

Theory:

1. **Ideal Low Pass Filter**: An ideal low pass filter is a theoretical filter that allows all frequencies below a certain cutoff frequency to pass through unchanged, while completely blocking all frequencies above that cutoff. In practice, it's impossible to create a perfect ideal filter due to issues like Gibbs phenomenon, which causes ringing artifacts in the filtered image.

2. **Butterworth Low Pass Filter**: A Butterworth low pass filter is a type of filter commonly used in signal processing and image processing. It is designed to allow frequencies below a certain cutoff frequency to pass through with minimal attenuation, while attenuating higher frequencies progressively. It provides a smoother roll-off compared to the ideal filter and is defined by its order and cutoff frequency.

3. **Gaussian Low Pass Filter**: A Gaussian low pass filter is a filter where the frequency response is defined by a Gaussian function. It attenuates higher frequencies more than lower frequencies, gradually smoothing the image. It's often used in image processing for noise reduction and blurring while preserving the overall structure of the image. The amount of smoothing is controlled by the standard deviation parameter of the Gaussian function.

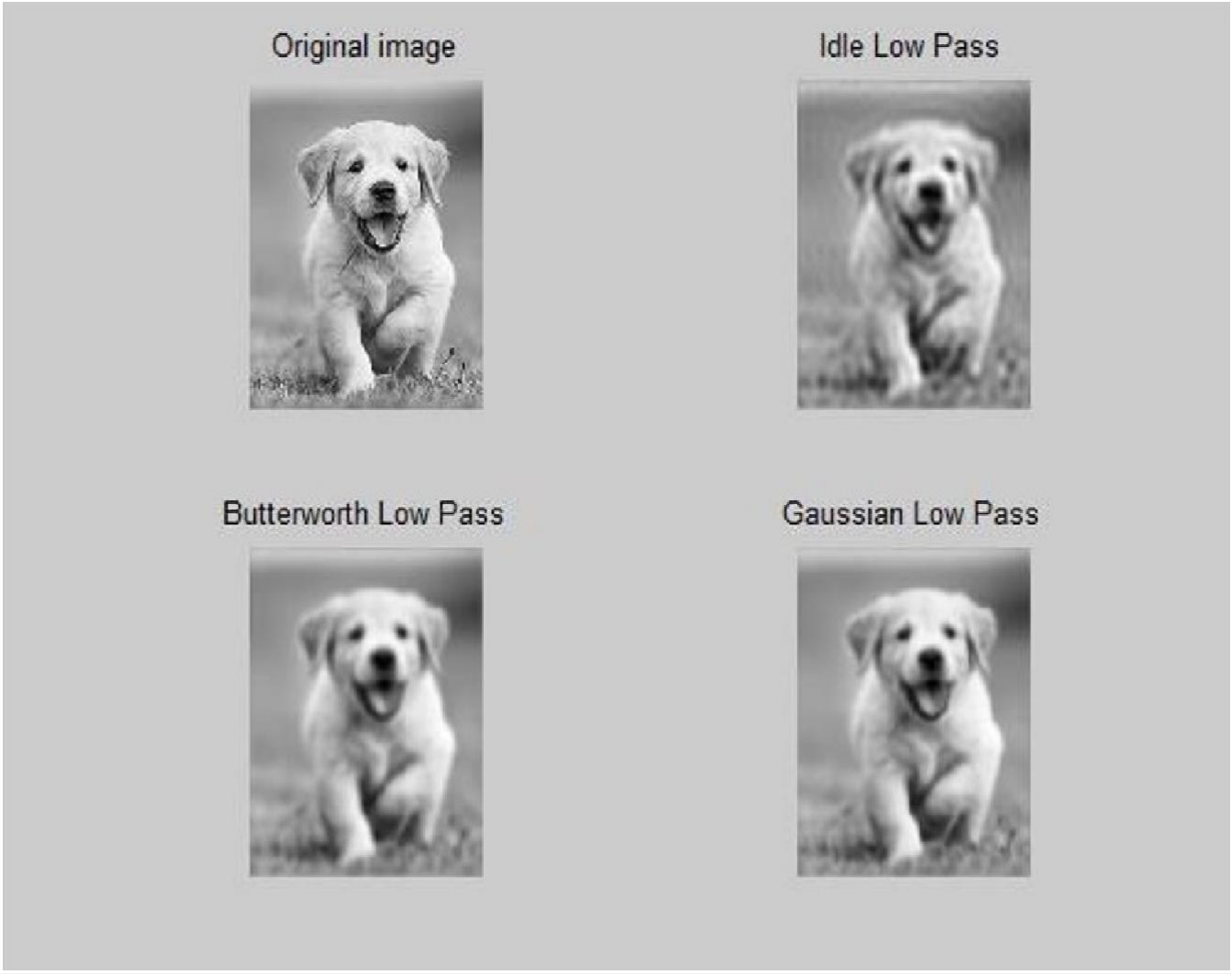
Code:

```
clc;
clear all;
close all;
i= imread('C:\Users\Ashok\Desktop\dog.jpg');
 i=rgb2gray(i);
[M, N] = size(i);
FT_img = fft2(double(i));
% FT_img=fftshift(FT_img);
% Assign Cut-off Frequency D0 = 20;
n=2;
% Designing filter u = 0:(M-1);
idx = find(u>M/2); u(idx) = u(idx)-M;
v = 0:(N-1);
idy = find(v>N/2); v(idy) = v(idy)-N;
[V, U] = meshgrid(v, u);
% Calculating Euclidean Distance D = sqrt(U.^2+V.^2);
Hil = double(D <= D0); Hbl=1./(1 + (D./D0).^(2*n)); a=D0^2;
b=D.^2;
```

```
c=2*a;
Hgl =exp(-(D.^2)/(2*D0.^2)); Gil = Hil.*FT_img;
Gbl = Hbl.*FT_img;
Ggl = Hgl.*FT_img;
% Getting the resultant image by Inverse Fourier Transform output_image1 = real(ifft2(double(Gil)));
output_image2 = real(ifft2(double(Gbl))); output_image3 = real(ifft2(double(Ggl)));
% Displaying Input Image and Output Image figure()
subplot(2, 1, 1), imshow(i),
subplot(2, 1, 2), imshow(output_image1, [ ]);
figure()
subplot(2, 1, 1), imshow(i),
subplot(2, 1, 2), imshow(output_image2, [ ]);
figure()
subplot(2, 1, 1), imshow(i),
subplot(2, 1, 2), imshow(output_image3, [ ]);
figure()
subplot(2,2,1);imshow(i);title('Original image');
subplot(2,2,2);imshow(output_image1,[]);title('IdleLowpass');
subplot(2,2,3);imshow(output_image2,[]);title('ButterworthLowPass');
subplot(2,2,4);imshow(output_image3,[]);title('Gaussian Low Pass');
```

INPUT: -

Simulation Results: -



Conclusion: -

The Ideal Low Pass filter creates a sharp cutoff but may cause ringing artifacts, the Butterworth filter provides smoother transitions and better control, while the Gaussian filter offers the smoothest and most natural blurring effect. Each filter type produces different levels of blurring and smoothness.

PRACTICAL: -10

AIM: - Write MATLAB code to blur the image using Ideal High pass, Butterworth High pass and Gaussian High pass filter**.**

Software/ Hardware: -

Objective: -

The objective of the experiment is to,

- The objective is to blur an image using high pass filters by inverting their effect.
- This involves applying Ideal, Butterworth, and Gaussian High Pass Filters to the image.
- The goal is to observe the blurring effect through these different filters.

Theory: -

- **Ideal High-Pass Filter: -**
  - The ideal high-pass filter rejects or attenuates low-frequency components (i.e., smooth areas) of an image while preserving or passing high-frequency components (i.e., edges and details).
  - It is characterized by a sharp transition from attenuation to preservation at a specified cutoff frequency. Frequencies above the cutoff are passed, and frequencies below are attenuated.
  - The main drawback of the ideal high-pass filter is that it has infinite support in the frequency domain, resulting in ringing artifacts in the spatial domain.

- **Butterworth High-Pass Filter: -**
  - The Butterworth high-pass filter is a type of high-pass filter with a smoother transition between the passband (where frequencies are preserved) and the stopband (where frequencies are attenuated) compared to the ideal filter.
  - It is characterized by a parameter called the order (n), which determines the rate of roll off from the passband to the stopband. Higher order filters have steeper roll offs.
  - Butterworth filters are preferred over ideal filters when a smoother transition is desired, as they produce fewer ringing artifacts.

- **Gaussian High-Pass Filter: -**
  - The Gaussian high-pass filter attenuates low-frequency components of an image while preserving high-frequency components, similar to other high-pass filters.
  - It is based on the Gaussian function, which has a bell-shaped curve. In the frequency domain, the Gaussian filter reduces the weights of lower frequencies gradually, with a smooth roll off.
  - Gaussian filters are often used for noise reduction and image smoothing but can also be applied as a high-pass filter by subtracting the low-pass Gaussian filter from the original image.

Code:

```
clc;
clear all;
close all;

i = imread('C:\Users\Ashok\Desktop\dog.jpg'); i
= rgb2gray(i);
[M, N] = size(i);
FT_img = fft2(double(i));

% Assign Cut-off Frequency
D0 = 20;
n = 2;

% Designing high-pass filters
u = 0:(M-1);
idx = find(u>M/2);
u(idx) = u(idx)-M;
v = 0:(N-1);
idy = find(v>N/2);
v(idy) = v(idy)-N;
[V, U] = meshgrid(v, u);

% Calculating Euclidean Distance
D = sqrt(U.^2 + V.^2);

% Ideal high-pass filter
Hil = double(D > D0);

% Butterworth high-pass filter
Hbl = 1 ./ (1 + (D./D0).^(2*n));
Hbh = 1 - Hbl;

% Gaussian high-pass filter
Hgl = 1 - exp(-(D.^2) / (2 * D0^2));
Hgh = 1 - Hgl;

% Apply high-pass filters in frequency domain
Gih = Hil .* FT_img;
Gbh = Hbh .* FT_img;
Ggh = Hgh .* FT_img;

% Getting the resultant image by Inverse Fourier Transform
output_image1 = real(ifft2(double(Gih)));
output_image2 = real(ifft2(double(Gbh)));
output_image3 = real(ifft2(double(Ggh)));

% Displaying Input Image and Output Image
figure()
subplot(2, 1, 1), imshow(i), title('Original image');
subplot(2, 1, 2), imshow(output_image1, []), title('Ideal High Pass');
```

```
figure()
subplot(2, 1, 1), imshow(i), title('Original image');
subplot(2, 1, 2), imshow(output_image2, []), title('Butterworth High Pass');

figure()
subplot(2, 1, 1), imshow(i), title('Original image');
subplot(2, 1, 2), imshow(output_image3, []), title('Gaussian High Pass');
```

INPUT : -

Simulation Results:

Fig: -1

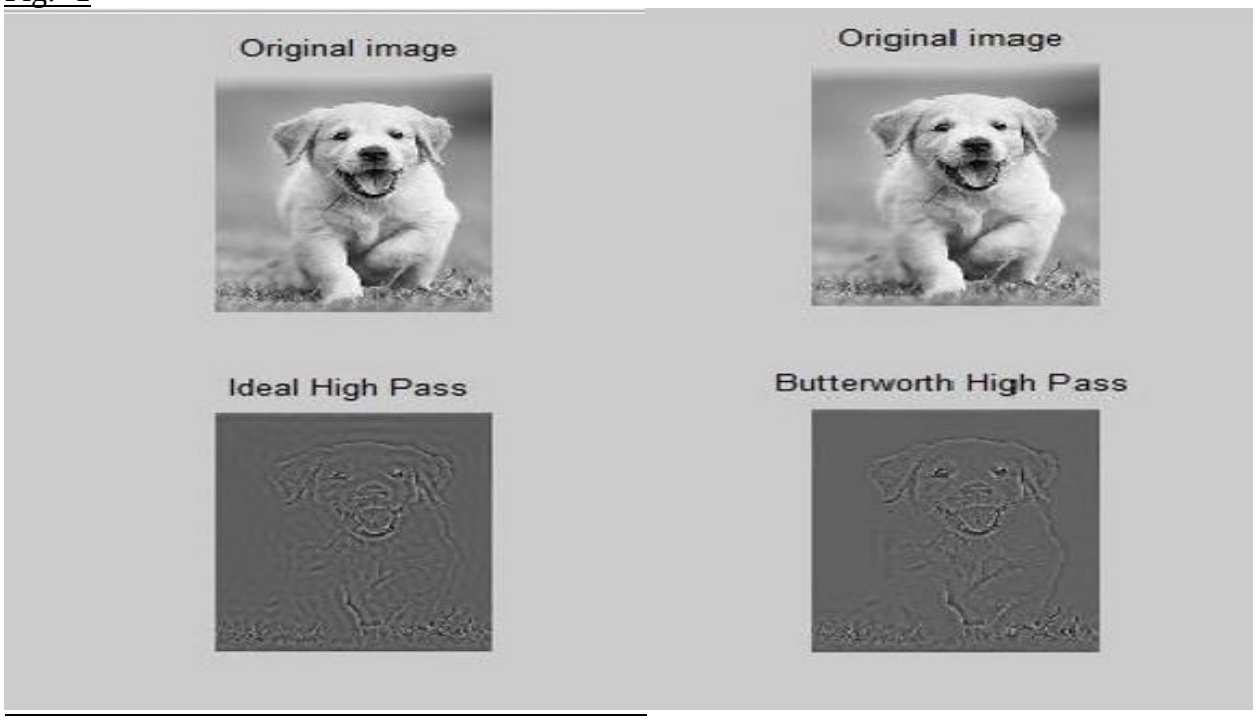


Fig: -2

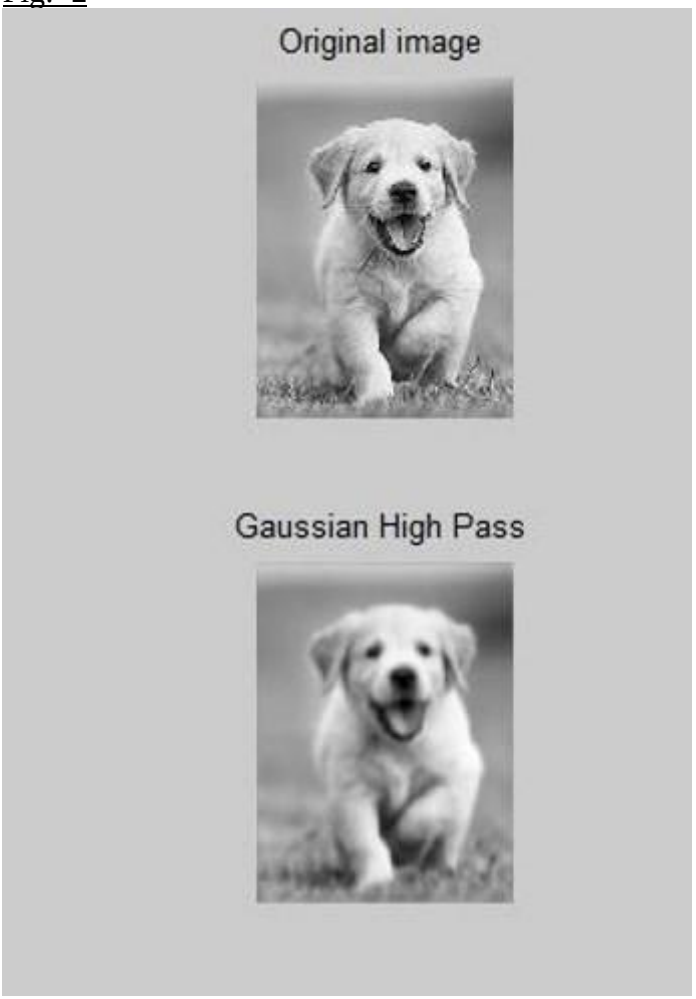
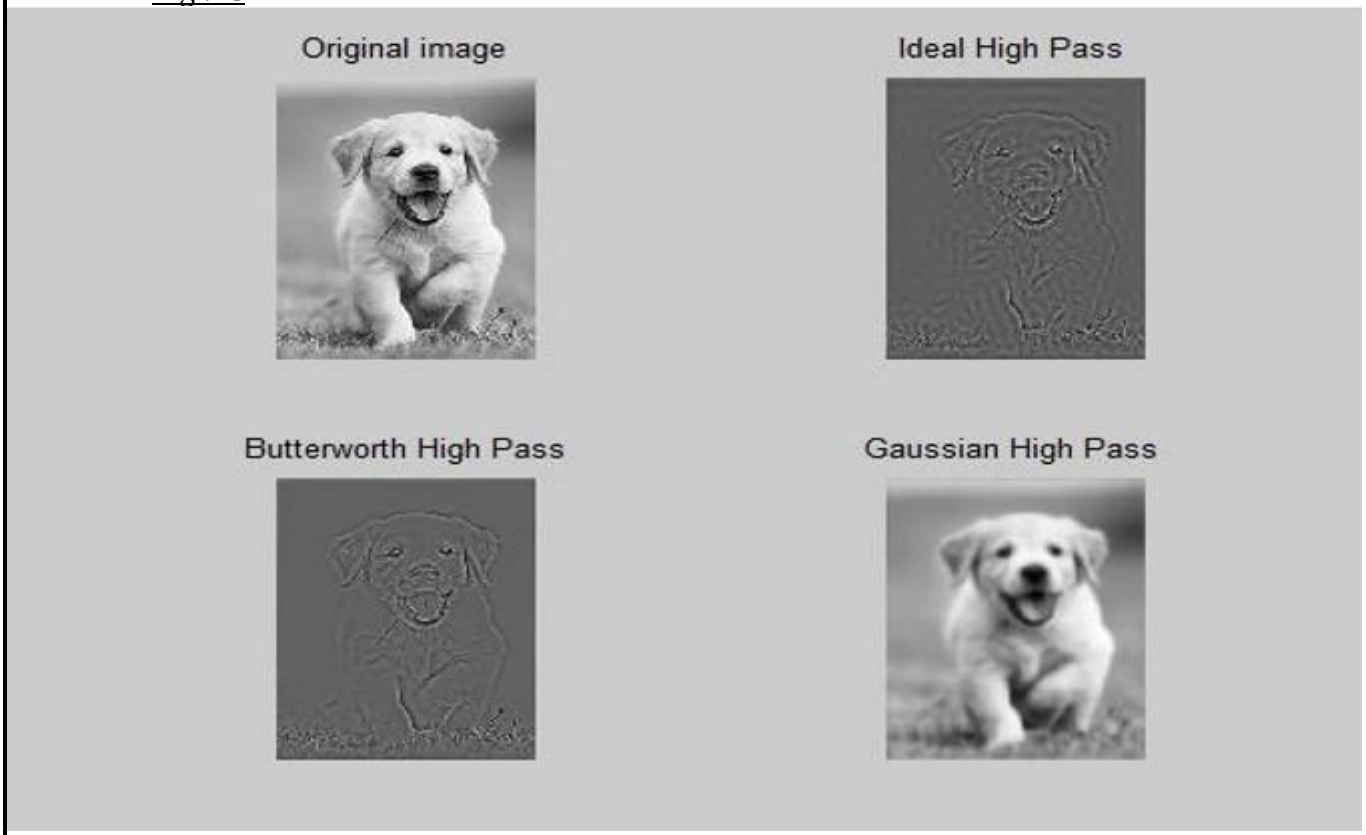

Original image
Gaussian High Pass

Fig : -3


Original image
Ideal High Pass
Butterworth High Pass
Gaussian High Pass

Conclusion: -

Blurring an image using Ideal, Butterworth, and Gaussian High Pass Filters demonstrates that different filters produce varying levels of smoothness and edge retention. The Gaussian filter offers the smoothest transition, while the Ideal filter has the sharpest cutoff. These filters effectively reduce high-frequency components, leading to a blurred effect.