PRACTICAL: 01

Aim: To study and implement basic commands of MATLAB required for digital image processing techniques, and various image file formats.

Objective: To study and implement basic MATLAB commands used in digital image processing, including reading, displaying, transforming, and analyzing images, and to understand different image file formats supported by MATLAB.

Procedure:

1. **clc**
   Syntax:

   ```
   clc
   ```

   Description:
   Clears the Command Window (output screen), but does not affect variables or figures.

   Example:

   ```
   clc
   ```

2. **clear all**
   Syntax:

   ```
   clear all
   ```

   Description:
   Removes all variables from the workspace, freeing up memory.

   Example:

   ```
   clear all
   ```


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

3. closeall

Syntax:

```
close all
```

Description:
Closes all open figure windows.

Example:

```
close all
```

4. imread()

Syntax:

```
img = imread('filename.ext');
```

Description:
Reads an image file into MATLAB as a matrix.

Example:

```
img = imread('trusty.png');
```

5. imwrite()

Syntax:

```
imwrite(img, 'trusty.jpg');
```

Description:
Saves the image to a file.

Example:

```
imwrite(img, 'output_gray.jpg');
```


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

6. imshow()

Syntax:

```
imshow(img);
```

Description:
Displays an image in a figure window.

Example:

```
imshow(img);
```

Output:

7. title()

Syntax:

```
title('Rating')
```

Description:
Adds a title above the current image or plot.

Example:

```
imshow(img);
title('Rating');
```


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Output:

8. subplot()

Syntax:

```
subplot(m, n, p)
```

Description:
Divides the figure into an m × n grid and activates the pth section for plotting.

Example:

```
subplot(1,2,1); imshow(img);
subplot(1,2,2); imshow(rgb2gray(img));
```

Output:

9. figure

Syntax:

```
figure
```

Description:
Opens a new figure window for displaying plots/images.

Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Example:

```
figure; imshow(img);
```

Output:

10. imhist()

Syntax:

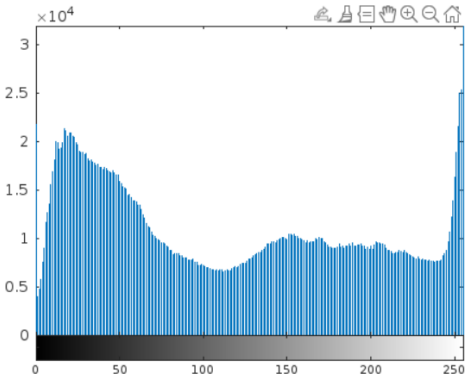
```
imhist(img);
```

Description:
Displays the histogram of a grayscale image.

Example:

```
imhist(img)
```

Output:



Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

11. addpath()

Syntax:

```
addpath('folder_path')
```

Description:
Adds a directory to MATLAB's search path to access files from that folder.

Example:

```
addpath("D:\files\practicals\Practical-1'");
```

12. imfinfo()

Syntax:

```
info = imfinfo('trusty.jpg')
```

Description:
Returns metadata about the image file, such as resolution, color type, and size.

Example:

```
addpath("D:\files\practicals\Practical-1'");
```

Output:

```
struct with fields:

           Filename: '/MATLAB Drive/tweeload_gio6rgik.jpg'
        FileModDate: '26-Jun-2025 09:49:37'
           FileSize: 199537
             Format: 'jpg'
      FormatVersion: ''
              Width: 1200
             Height: 803
           BitDepth: 24
          ColorType: 'truecolor'
    FormatSignature: ''
    NumberOfSamples: 3
       CodingMethod: 'Huffman'
      CodingProcess: 'Sequential'
            Comment: {}
  AutoOrientedWidth: 1200
 AutoOrientedHeight: 803
```


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

13. imresize()

Syntax:

```
resizedImg = imresize(img, scale);
```

Description:
Resizes the image by a scale factor.

Example:

```
halfImg = imresize(img, 0.5);          % 50% of original size
imshow(halfImg);
```

Output:

14. rgb2gray()

Syntax:

```
grayImg= rgb2gray(img);
```

Description:
Converts a color image to grayscale.

Example:

```
grayImg = rgb2gray(img);
imshow(grayImg);
```


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Output:

15. imadjust()

Syntax:

```
newimg = imadjust(img);
```

Description:
Enhances contrast of the image.

Example:

```
gray_img= rgb2gray(img);            % Convert to grayscale
```

Output:

Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

16. edge()

Syntax:

```
edges = edge(grayImg, 'method');
```

Description:
Detects edges in the image using a specific method (`'sobel'`, `'canny'`, etc.).

Example:

```
gray_img= rgb2gray(img);              % Convert to grayscale
edges = edge(gray_img, 'sobel');      % Apply Sobel edge detection
```

Output:

17. imfilter()

Syntax:

```
filteredImg = imfilter(img, h);
```

Description:
Applies a filter to an image.

Example:

```
h = fspecial('average', [5 5]);
smoothImg = imfilter(gray_img, h);
imshow(smoothImg);
```


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Output:

18. imnoise()

Syntax:

```
noisyImg = imnoise(img, 'type');
```

Description:
Addsnoise to the image (`'salt & pepper'`, `'gaussian'`, etc.).

Example:

```
noisy = imnoise(gray_img, 'salt & pepper');
imshow(noisy);
```

Output:

Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

19. medfilt2()

Syntax:

```
filtered = medfilt2(grayImg);
```

Description:
Applies median filtering to remove noise.

Example:

Output:

20. imbinarize()

Syntax:

```
bw = imbinarize(grayImg);
```

Description:
Converts a grayscale image to binary (black and white).

Example:

```
ngrayImg = rgb2gray(img);
bw =imbinarize(ngrayImg);
imshow(bw);
```


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Output:

21. size()

Syntax:

```
[rows, cols] = size(img);
```

Description:
Returns the number of rows and columns in the image.

Example:

```
[r, c, d] = size(img);
fprintf('Rows: %d, Columns: %d, Channels: %d\n', r, c, d);
```

Output:

```
Rows: 803, Columns: 1200, Channels: 3
```


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Commands in brief

Command	Syntax	Description	Example
clc	clc	Clears command window	clc
clear all	clear all	Removes all variables from workspace	clear all
close all	close all	Closes all figure windows	close all
imread	imread('file.jpg')	Reads an image from file	img = imread('peppers.png');
title	title('text')	Adds title to current plot	title('Original Image');
subplot	subplot(m, n, p)	Creates subplots in figure	subplot(2,2,1);
figure	figure	Opens new figure window	figure; imshow(img);
imshow	imshow(img)	Displays an image	imshow(img);

Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Command	Syntax	Description	Example
double	double(var)	Converts data to double precision	A = double(img);
imagesc	imagesc(data)	Scales and displays matrix as image	imagesc(grayImg);
addpath	addpath('folder')	Adds folder to MATLAB path	addpath('C:\images')
clf	clf	Clears current figure window	clf
imwrite	imwrite(img, 'file.jpg')	Saves image to file	imwrite(img, 'output.jpg');
imfinfo	imfinfo('file.jpg')	Displays image file information	info = imfinfo('peppers.png');
who	who	Lists variables in workspace	who
whos	whos	Lists variables with details	whos

Conclusion:

- Basic MATLAB commands for image processing were successfully implemented. We learned how to read, display, convert, resize, and save images, and understood different image formats. This forms the foundation for further image processing techniques.

Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

PRACTICAL-02

AIM: To study the effect of down sampling and quantization techniques on the grayscale image.

Objective: The objective of the experiment is to,

- sampling and quantization techniques on the grayscale image.
- Display effect of down sampling and quantization with help of the MATLAB code.

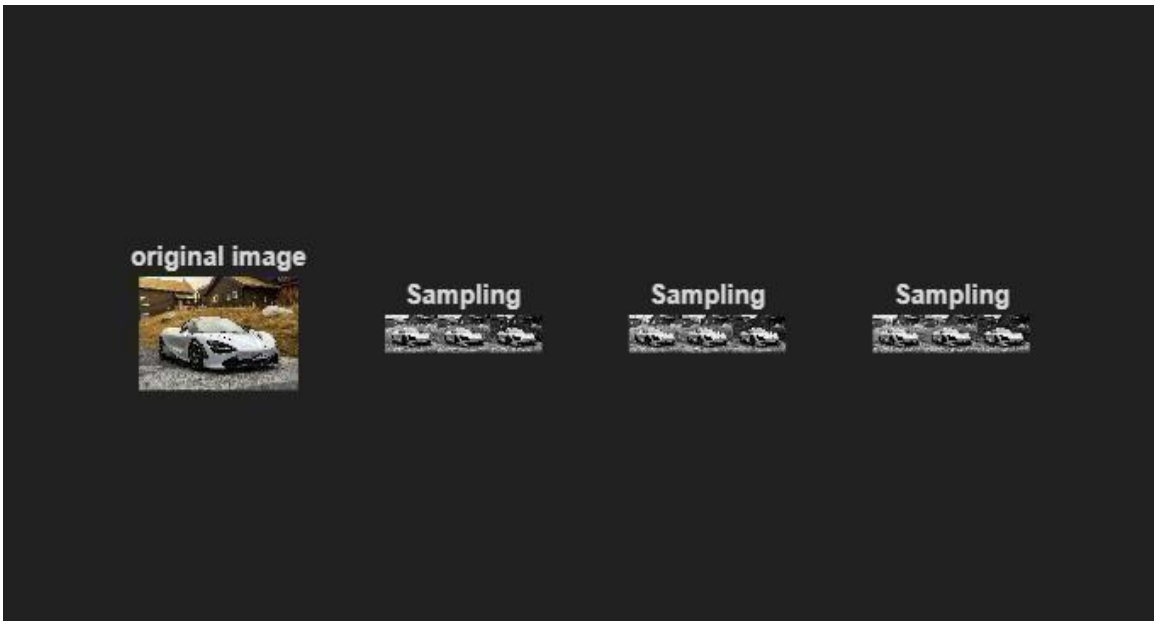
Code:

```
i=imread('car.jpg');
figure();
subplot(1,4,1);
imshow(i);
title('original image');
[r,c]=size(i);
a=i(1:2:r,1:2:c);
subplot(1,4,2);
imshow(a);
title('Sampling');
d=i(1:3:r,1:3:c);
subplot(1,4,3);
imshow(d);
title('Sampling');
b=i(1:4:r,1:4:c);
subplot(1,4,4);
imshow(b);
title('Sampling');
i=imread('car.jpg');
b=imresize(i,0.5,'nearest');
figure();
subplot(1,3,1);
imshow(b);title('nearest');
b1=imresize(i,0.5,'bilinear');
subplot(1,3,2);
imshow(b1);title('bilinear');
b2=imresize(i,0.5,'bicubic');
subplot(1,3,3);
imshow(b2);title('bicubic');
```


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

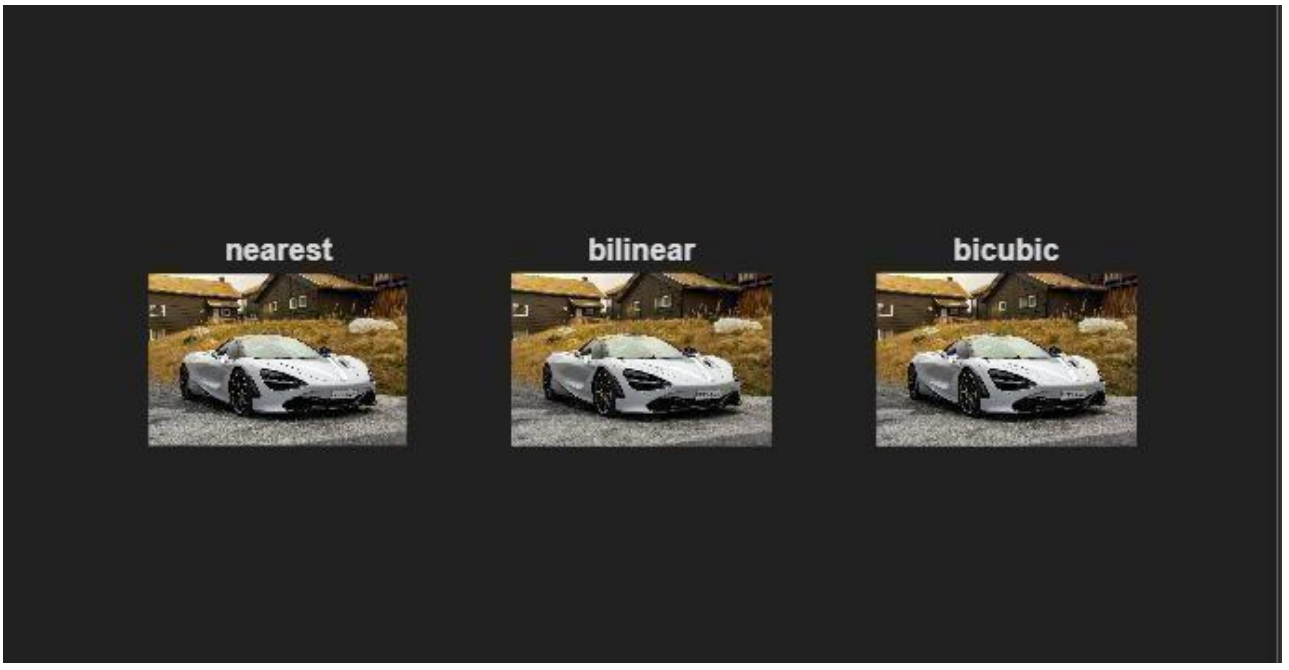
Simulation Results

Input:



Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Output:



Conclusion:

This experiment demonstrated the effects of down sampling and quantization on a grayscale image using MATLAB.
As Sampling rate decreases, image detail and resolution reduce significantly.
Different interpolation methods (nearest, bilinear, bicubic) impact image smoothness and quality during resizing.

Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

PRACTICAL-03

AIM: Write MATLAB code to enhance the visual quality of the image using point processing techniques, namely a) Image negative, b) Logarithmic transformation, and c) Power law transformation and gray level slicing technique.

Objective: The objective of the experiment is to,

➢ Gets basic information of the visual quality of the image using point processing techniques like a) Image negative, b) Logarithmic transformation, c) Power law transformation and d) Gray level slicing technique.

➢ Display all techniques on Matlab using Matlab code.

Theory:

• **Image Negative:** This is an image processing technique where the colors or intensities in an image are inverted. In a grayscale image, this means that lighter areas become darker and vice versa. In a colored image, each color channel is inverted independently.

• **Logarithmic Transformation:** A type of image enhancement technique where the pixel values in an image are transformed using a logarithmic function. This transformation is often used to expand the dynamic range of darker areas in the image while compressing the range of brighter areas.

• **Power Law Transformation:** Also known as gamma correction, this transformation involves raising the pixel values in an image to a power law function. It is used to adjust the contrast and brightness of an image, especially in cases where the image's histogram is concentrated towards either the low or high intensity values.

• **Gray Level Slicing Technique:** This technique is used to highlight specific ranges of pixel intensity values in an image. It involves thresholding the pixel values into different intensity bands or levels. Pixels within a specified range are retained, while others are set to a particular value, typically resulting in enhanced features or segmentation of objects based on their intensity levels.

Code:

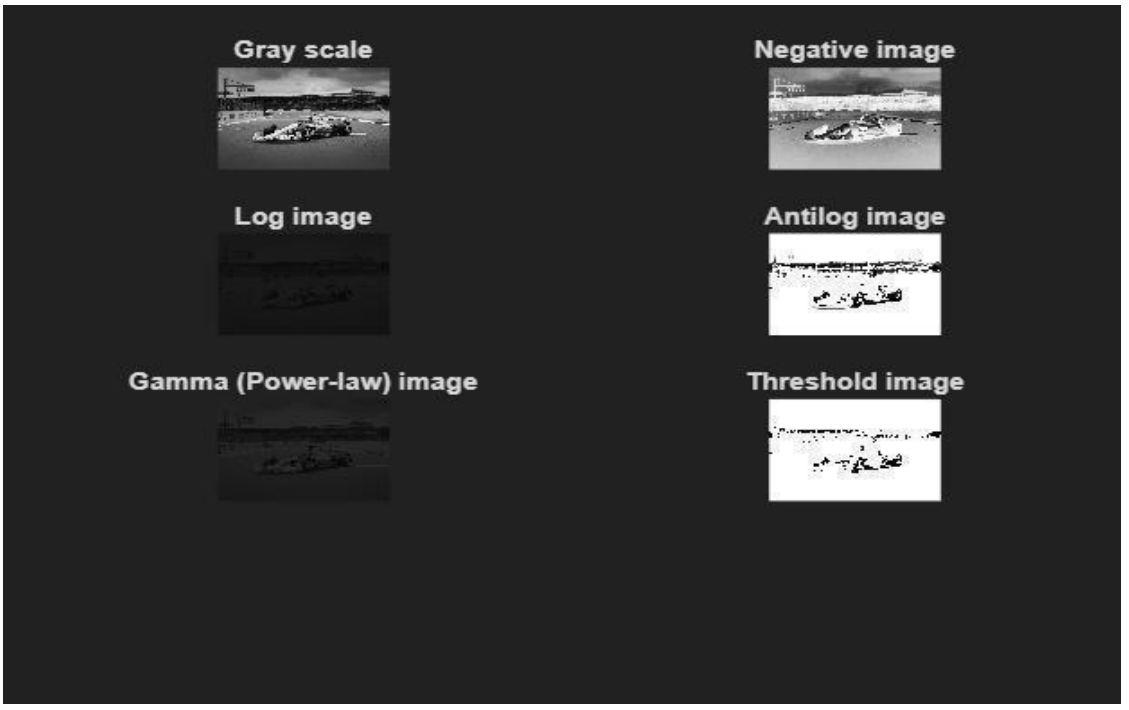
```
clc;
clear all;
close all;
a = imread('f1-.carjpg.jpg');
i = rgb2gray(a);
i = double(i);
s = 255 - i;
```


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

```
figure();
subplot(4,2,1);
imshow(uint8(i)), title('Gray scale');
subplot(4,2,2);
imshow(uint8(s)), title('Negative image');
c = 5;
s1 = c * log(1 + i);
subplot(4,2,3);
imshow(uint8(s1)), title('Log image');
s2 = (exp(i).^(1/c)) - 1;
subplot(4,2,4);
imshow(uint8(s2)), title('Antilog image');
gamma = 0.4;
s3 = c * (i .^ gamma);
subplot(4,2,5);
imshow(uint8(s3)), title('Gamma (Power-law) image');
th = 15;
[r, c] = size(i);
out = i
for x = 1:r
    for y = 1:c
        if (i(x,y) <= th)
            out(x,y)=0;
        else
            out(x,y)=255;
        end
    end
end
subplot(4,2,6);
imshow(uint8(out));
title('Threshold image');
```


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Results:



Conclusion: The experiment showed how point processing techniques can improve the appearance of grayscale images. Each method changes the brightness of pixels in different ways to create special effects.

• **Image negative** flips the light and dark areas, making bright parts easier to see on dark backgrounds.

• **Logarithmic transformation** brings out details in dark areas by making them brighter.

• **Power-law (gamma) transformation** changes contrast and brightness. The amount of change depends on the gamma value used.

• **Gray level slicing** highlights certain brightness levels, which is useful for things like medical images or finding objects, while the background can be kept or removed.

Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

PRACTICAL-04

AIM: Write MATLAB code to display the histogram of the image and enhance the visual quality of the image with the help of histogram equalization technique.

Objective: Understand the concept of image histograms and their significance in image processing.

- Display the histogram of an image using MATLAB.
- Enhance the visual quality of an image using histogram equalization.
- Apply both global and adaptive histogram equalization methods to observe contrast improvement.

Description (Theory in brief):

Image Histogram:

- An image histogram is a graphical representation of the distribution of pixel intensities in an image.
- The x-axis represents intensity values (0–255 for 8-bit images), and the y-axis shows the frequency of each intensity.
- It helps to analyze the contrast, brightness, and intensity spread of the image.

Histogram Equalization:

- Histogram equalization improves image contrast by redistributing the pixel intensities more evenly.
- It uses the **cumulative distribution function (CDF)** of the histogram to remap pixel values.
- The transformation function is:

$$s = T(r) = \frac{L-1}{M \times N} \sum_{j=0}^{r} n_j$$

Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Where:

- L = number of intensity levels (usually 256)
- M×N = total number of pixels
- n_j = number of pixels with intensity value j

Adaptive Histogram Equalization:

- This technique enhances contrast locally in small regions of the image.
- It avoids over-enhancement or artifacts in areas with uniform brightness.

Code:

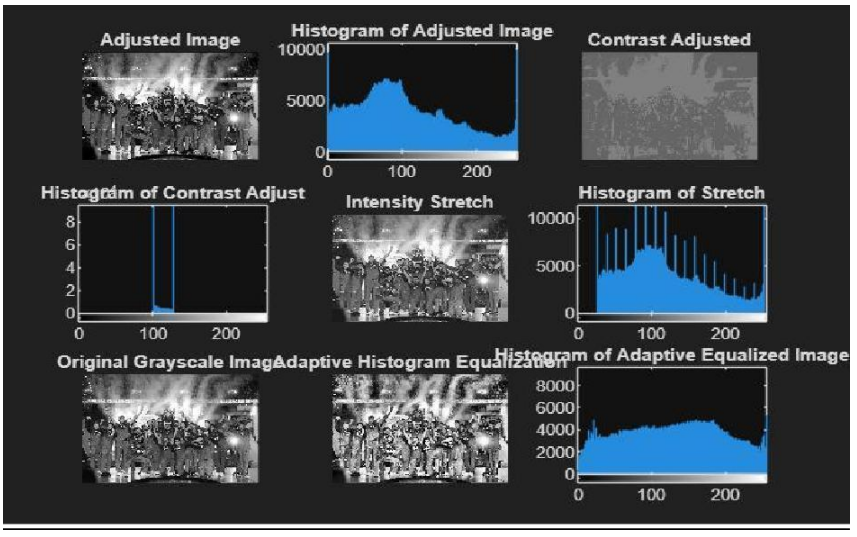
```
clc;
clear all;
close all;
a = imread('tweeload_gio6rgik.jpg');
figure;
imshow(a);
title('Original Color Image');
i = rgb2gray(a);
J = imadjust(i);
figure;
subplot(3,3,1);
imshow(J);
title('Adjusted Image');
subplot(3,3,2);
imhist(J);
title('Histogram of Adjusted Image');
J1 = imadjust(i, [0.4 0.5], [0.4 0.5]);
subplot(3,3,3);
imshow(J1);
title('Contrast Adjusted');
subplot(3,3,4);
imhist(J1);
title('Histogram of Contrast Adjust');
S = imadjust(i, stretchlim(i), [0.10 0.99]);
subplot(3,3,5);
imshow(S);
title('Intensity Stretch');
subplot(3,3,6);
imhist(S);
title('Histogram of Stretch');
AH = adapthisteq(i);
```


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

```
subplot(3,3,7);
imshow(i);
title('Original Grayscale Image');
subplot(3,3,8);
imshow(AH);
title('Adaptive Histogram Equalization');
subplot(3,3,9);
imhist(AH);
title('Histogram of Adaptive Equalized Image');
```

Input Image:

Output:



Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Conclusion:

- The experiment demonstrated how to visualize image histograms and apply histogram equalization in MATLAB.
- Histogram equalization significantly improves image contrast by spreading intensity values across the full dynamic range.
- Adaptive histogram equalization further enhanced local contrast, preserving details in different regions of the image.
- These techniques are useful in enhancing medical images, satellite imagery, and any low-contrast grayscale images.

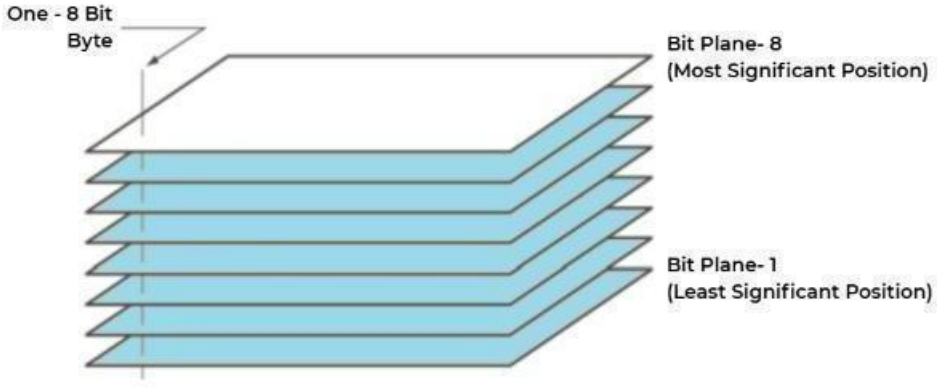
Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

PRACTICAL-05

AIM: Write MATLAB code to perform the bit-plane slicing method on grayscale images.

Objective: To extract and visualize individual bit planes (from bit 1 to bit 8) of a grayscale image to analyze how different bits contribute to the image's overall appearance and information content.

Description (Theory in brief):



- In digital image processing, each pixel of an 8-bit grayscale image is represented using 8 bits (ranging from bit-plane 1 to bit-plane 8). **Bit-plane slicing** involves isolating each of these 8 bits across all pixels to see their individual contributions:
  - **Bit 1 (LSB)** contains the least amount of visual information.
  - **Bit 8 (MSB)** contributes the most to the image's structure.
- This technique is useful for data compression, watermarking, and visual analysis.
- Bit-plane slicing helps in:
  - **Analyzing how different bits contribute** to the image.
  - Performing **compression, watermarking, or encryption**
  - **Reconstructing** an image using only significant bit planes (e.g., 5 to 8).

Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Code:

```
clc;
clear all;
close all;
% Step 1: Load your own colored image
i = imread('cricketT20.jpg');
gray = rgb2gray(i);              % Convert to grayscale
% Step 2: Bit slicing
a1 = bitget(gray, 1);
a2 = bitget(gray, 2);
a3 = bitget(gray, 3);
a4 = bitget(gray, 4);
a5 = bitget(gray, 5);
a6 = bitget(gray, 6);
a7 = bitget(gray, 7);
a8 = bitget(gray, 8);
a9 = im2bw(gray);  % Binary conversion
% Step 3: Grouped Figures
%% --- Figure 1: Original and Grayscale
figure();
subplot(1,2,1), imshow(i), title('Original Color Image');
subplot(1,2,2), imshow(gray), title('Grayscale Image');
%% --- Figure 2: Bit Planes 1 to 8
figure();
subplot(2,4,1), imshow(a1, []), title('Bit Plane 1');
subplot(2,4,2), imshow(a2, []), title('Bit Plane 2');
subplot(2,4,3), imshow(a3, []), title('Bit Plane 3');
subplot(2,4,4), imshow(a4, []), title('Bit Plane 4');
subplot(2,4,5), imshow(a5, []), title('Bit Plane 5');
subplot(2,4,6), imshow(a6, []), title('Bit Plane 6');
subplot(2,4,7), imshow(a7, []), title('Bit Plane 7');
subplot(2,4,8), imshow(a8, []), title('Bit Plane 8');
%% --- Figure 3: Binary & Bitwise Operations
comb2 = imlincomb(1, a5, 1, a6, 1, a7, 1, a8);     % Combine bit 5–8
bit1 = bitand(comb2, gray);                    % Bitwise AND
bit2 = bitset(comb2, 8);                       % Bitwise SET
figure();
subplot(2,2,1), imshow(a9), title('Binary (im2bw)');
subplot(2,2,2), imshow(bit1, []), title('bitand() Result');
subplot(2,2,3), imshow(bit2, []), title('bitset() Result');
subplot(2,2,4), imshow(comb2, []), title('Combined (Bits 5–8)');
```


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Output:

Figure 1:

Figure 2:

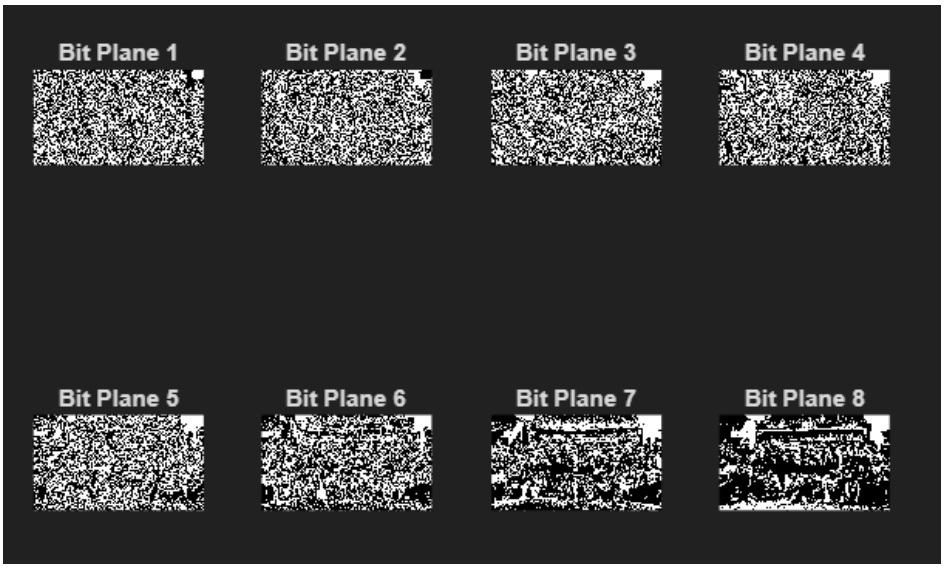


Figure 3:

Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Output Windows You'll See:

Figure: Bit-plane Slicing Results

1. **Original Color Image –** Your selected colored input image.

2. **Grayscale Image –** Converted version used for slicing.

3. **Bit Plane 1 to Bit Plane 8 –**
   - **Bit 1 (Least Significant Bit)** → very noisy, minor details
   - **Bit 8 (Most Significant Bit)** → major structure, most important features
   - Higher bits show clearer parts of the image, lower bits show noise or fine textures

4. **Binary Image –** Thresholded version of the grayscale image using im2bw.

Bitwise Operations Output

5. **bitand() Result** – Highlights shared features between grayscale image and combined bit layers.

6. **bitset() Result** – Alters the 8th bit of the combined image to emphasize its contribution.

7. **Combined (Bit 5–8)** – Logical addition of higher-order bit planes. Shows the reconstructed main structure of the image using only important bits.

Conclusion:

Bit-plane slicing successfully isolates the contribution of each bit in an 8-bit grayscale image. Higher-order bits (e.g., bits 7 and 8) contain most of the significant image information, while lower-order bits contribute to finer details or noise. This method is useful for analyzing image structure, compression, and digital watermarking.

Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

PRACTICAL-06

AIM: Write MATLAB code to demonstrate that the convolution in the spatial domain is equivalent to multiplication in the frequency domain.

Objective: To demonstrate that convolution in the spatial domain is equivalent to multiplication in the frequency domain using MATLAB.

Description (Theory in brief):

- According to the **Convolution Theorem** in signal processing:
  - **Convolution** in the spatial domain (image space) is equivalent to Point-wise multiplication in the frequency domain (Fourier space).
- This is mathematically represented as:

$$f(x,y) * h(x,y) \leftrightarrow F(u,v) \times H(u,v)$$

  - Where**:**
    - * is convolution
    - × is element-wise multiplication
    - **F(u,v)** and **H(u,v)** are the Fourier transforms of the image and filter respectively.
- This property is used to speed up filtering operations using Fast Fourier Transform (FFT).

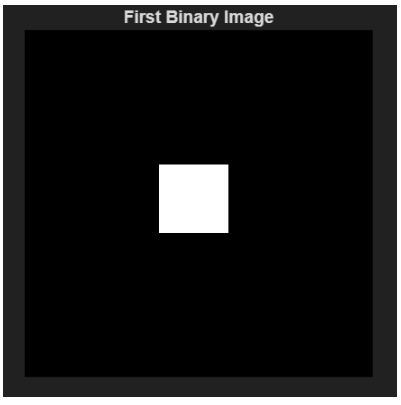
Code:

```
clc;
clear;
closeall;
%% Generating the First Binary Image
A = zeros(256); % create black image
A(100:150, 100:150) = 1; % white square
figure;
imshow(A);
title('First Binary Image');
%% Generating the Second Binary Image
B = zeros(256); % create black image
```

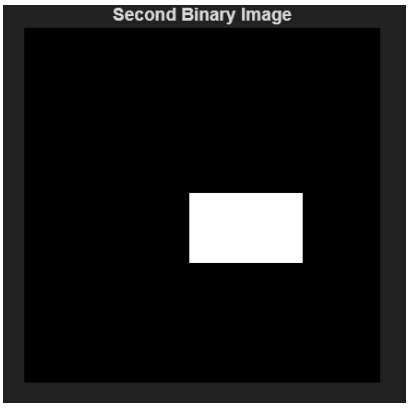

Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

```
B(120:170, 120:200) = 1; % white rectangle
figure;
imshow(B);
title('Second Binary Image');
%% Convolution in Spatial Domain
C = conv2(A, B, 'same');
%% Multiplication in Frequency Domain
A1 = fft2(A);
B1 = fft2(B);
C1 = A1 .* B1;
D = fftshift(ifft2(C1));
%% Display Results
figure;
subplot(3,2,1);
imshow(A);
title('First Binary Image');
subplot(3,2,2);
imshow(B);
title('Second Binary Image');
subplot(3,2,3);
imshow(log(1 + abs(fftshift(A1))), []);
title('DFT of First Image');
subplot(3,2,4);
imshow(log(1 + abs(fftshift(B1))), []);
title('DFT of Second Image');
subplot(3,2,5);
```

First Binary Image :

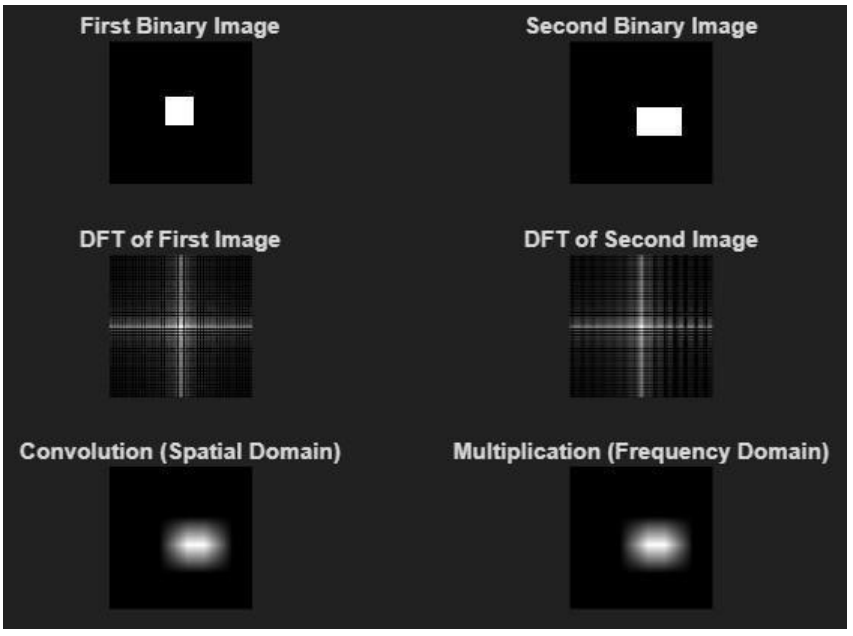


Second Binary Image:



Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Result of First and Second Image:



Output Windows You'll See:

- **First Binary Image**
  - A black 256×256 image with a **white square** in the middle (pixels 100–150 in both x and y directions).

- **Second Binary Image**
  - A black 256×256 image with a **white rectangle** slightly shifted down and right (pixels 120–170 in y-direction, 120–200 in x-direction).

- **DFT of First Image**
  - The **log magnitude spectrum** of the first binary image.
  - Appears as a bright symmetric pattern centered due to the square shape in the spatial domain.

- **DFT of Second Image**
  - The **log magnitude spectrum** of the second binary image.
  - Shows a different bright frequency pattern because of the rectangle shape.

Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

- **Convolution (Spatial Domain)**
  - The **result of sliding the second binary image over the first** and summing the overlap at each position.
  - Appears as a larger bright region due to convolution broadening the shape.
- **Multiplication (Frequency Domain)**
  - The **result of multiplying the frequency spectra** of both images, then converting back to spatial domain.
  - Matches the convolution result, but small numerical differences may appear due to rounding.

Conclusion:

- This experiment validates that:
  - **Spatial Convolution = Frequency Multiplication** (under Fourier Transform)
  - This principle is widely used in **image processing**, **computer vision**, and **digital filtering** to optimize large kernel operations using the Fast Fourier Transform (FFT).

Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

PRACTICAL-07

AIM: Write MATLAB code to restore grayscale images from noisy images with the help of image restoration techniques.

Objective:

- Understand and implement **image restoration techniques** for noise removal.
- Apply **median filtering** to remove impulsive (salt-and-pepper) noise.
- Apply **Wiener filtering** to reduce Gaussian noise.
- Combine both filters for improved noise reduction while preserving details.

Description (Theory in brief):

- **Image Restoration**
  - Image restoration aims to recover an original image from a degraded (noisy) image while preserving important details.
  -

1. Median Filtering

- A **nonlinear** filtering technique.
- Replaces each pixel's value with the **median** of its surrounding neighborhood.
- Effective against **salt-and-pepper noise**.
- Preserves edges better than simple averaging.

2. Wiener Filtering

- A **linear** statistical filter.
- Minimizes the **mean square error** between the restored and original image.
- Adapts to local image variance and mean.
- Works well for **Gaussian noise**.

3. Combined Filtering

- **Step 1:** Median filter removes impulsive noise.
- **Step 2:** Wiener filter reduces Gaussian noise from the median-filtered image.

Results in improved restoration for mixed noise conditions.

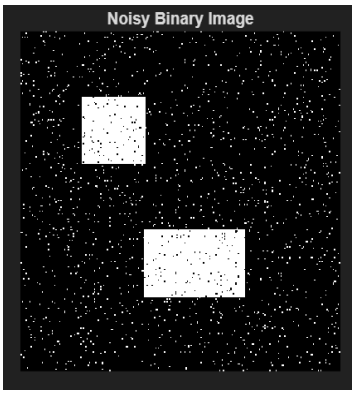
Code:

```
clc;
clear all;
close all;
% 1. Generate a sample binary image (simple shapes)
binaryImage = zeros(256, 256); % create black image
binaryImage(50:100, 50:100) = 1; % white square
```

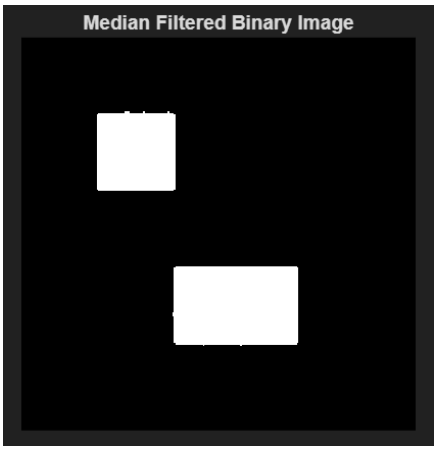

Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

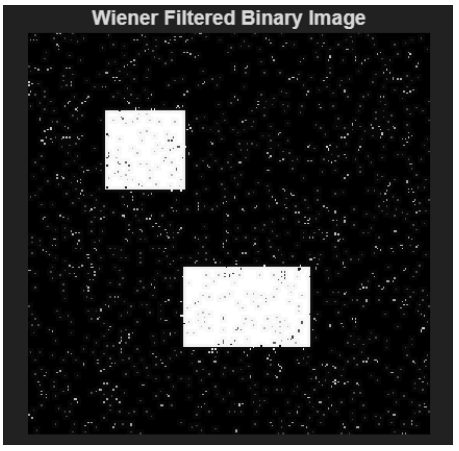
```
binaryImage(150:200, 100:180) = 1; % white rectangle
binaryImage = imnoise(binaryImage, 'salt & pepper', 0.05); % add noise
% 2. Display the noisy binary image
figure;
imshow(binaryImage);
title('Noisy Binary Image');
% 3. Apply Median Filtering
medianFiltered = medfilt2(binaryImage);
figure;
imshow(medianFiltered);
title('Median Filtered Binary Image');
% 4. Apply Wiener Filtering
wienerFiltered = wiener2(binaryImage, [5 5]);
figure;
imshow(wienerFiltered);
title('Wiener Filtered Binary Image');
% 5. Apply Combined Filtering (Median + Wiener)
combinedFiltered = wiener2(medianFiltered, [5 5]);
figure;
imshow(combinedFiltered);
title('Combined Median + Wiener Filtered Binary Image');
% 6. Convert final image back to binary using thresholding
restoredBinary = imbinarize(combinedFiltered);
% 7. Save the restored binary image
imwrite(restoredBinary, 'restored_binary_image.png');
% 8. Display the saved restored binary image
restoredImage = imread('restored_binary_image.png');
figure;
imshow(restoredImage);
title('Restored Binary Image');
```

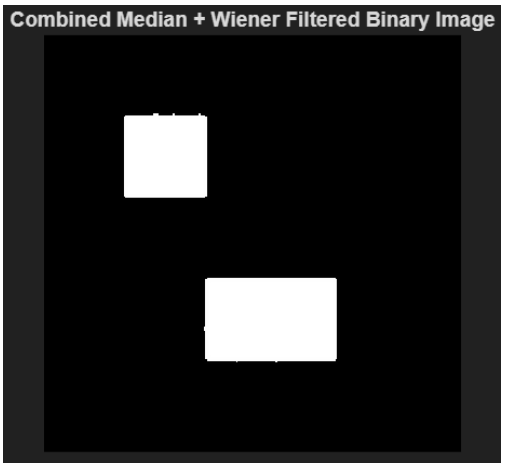
Output:

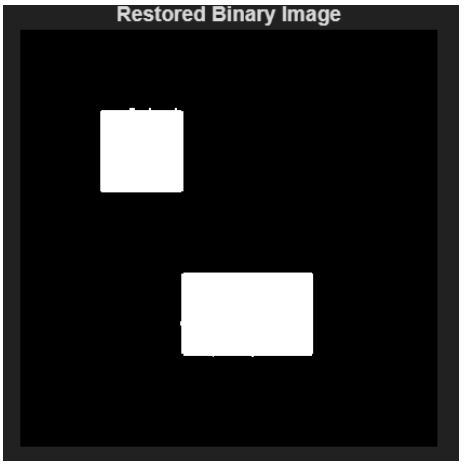


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)









Conclusion:

- **Median filtering** is highly effective for removing salt-and-pepper noise while maintaining edge sharpness.
- **Wiener filtering** works better for Gaussian noise but may cause slight blurring of fine details.
- **The combined approach** successfully reduces both impulsive and Gaussian noise types, producing a cleaner and sharper result.
- This experiment demonstrates that **filter selection depends on the type of noise** present in an image.

Such restoration techniques are widely applicable in **medical imaging, satellite image enhancement, forensic investigations, and historical photograph restoration.**

Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

PRACTICAL-08

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

Objective: To write a MATLAB program to perform Minimum, Median, and Maximum Filtering on a grayscale image and observe the effect of each filtering technique on image noise and details.

Description (Theory in brief):

- **Image Filtering**
  - Image Filtering is a technique used to enhance or modify an image by removing noise or emphasizing certain features.

1. Minimum Filter (Min Filter)

- Replaces each pixel value with the minimum value in its neighborhood (defined by a filter mask).
- Effective in removing salt noise (bright spots) but may cause image darkening.

2. Median Filter

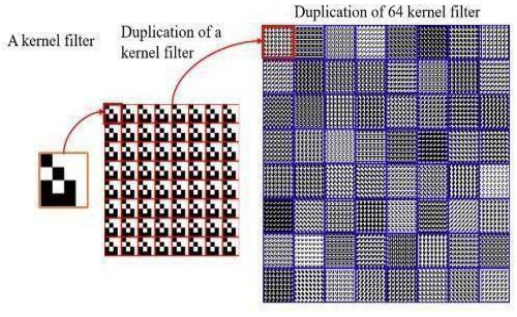
- Replaces each pixel value with the median of the surrounding pixel values.
- Best for removing salt-and-pepper noise while preserving edges.
- Non-linear filtering method.

3. Maximum Filter (Max Filter)

- Replaces each pixel with the maximum value in its neighborhood.
- Removes pepper noise (dark spots) and enhances bright areas.

General Filtering Process:

- A mask (kernel) slides over the image.
- For each position, pixel values under the mask are processed based on the filter type.
- Result is stored in the output image.



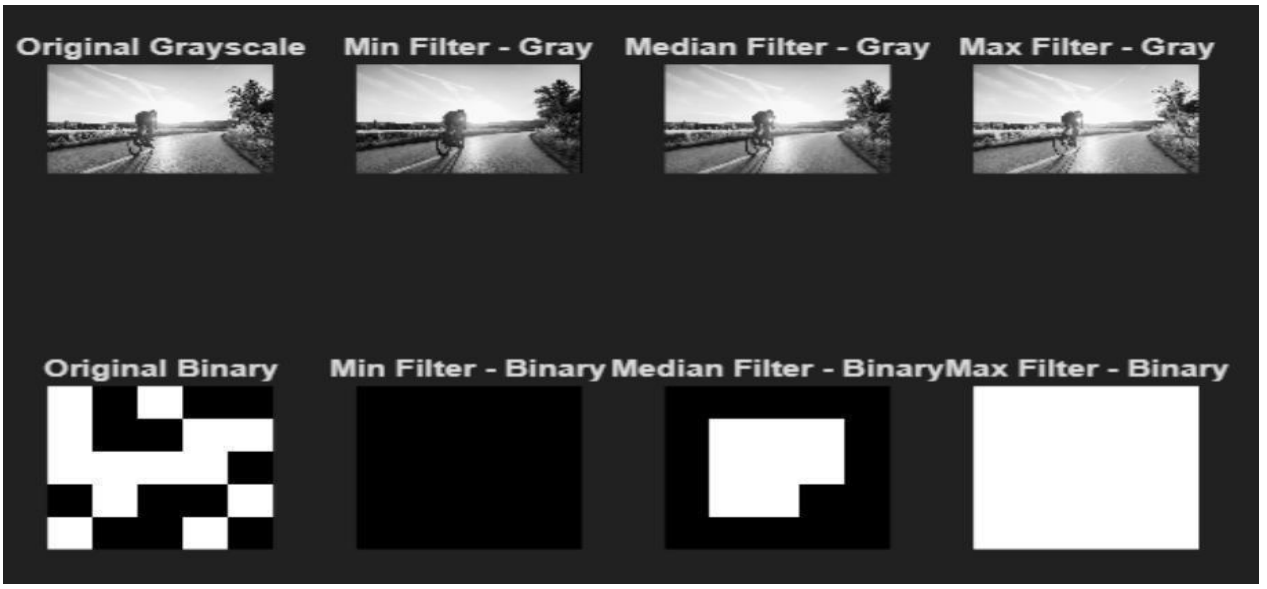
Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Code:

```
clc;
clear;
closeall;
% -------- Grayscale Image --------
i = imread('cycle.jpg');
grayImg=rgb2gray(i);
figure;
subplot(2,4,1), imshow(grayImg), title('Original Grayscale');
minGray = ordfilt2(grayImg, 1, ones(3,3));
medianGray = medfilt2(grayImg, [3 3]);
maxGray = ordfilt2(grayImg, 9, ones(3,3));
subplot(2,4,2), imshow(minGray), title('Min Filter - Gray');
subplot(2,4,3), imshow(medianGray), title('Median Filter - Gray');
subplot(2,4,4), imshow(maxGray), title('Max Filter - Gray');
% -------- Binary Image (0s & 1s) --------
binaryImg = [
    1 0 1 0 0;
    1 0 0 1 1;
    1 1 1 1 0;
    0 1 0 0 1;
    1 0 0 1 0
];
binaryImg = logical(binaryImg); % Convert to logical for display
subplot(2,4,5), imshow(binaryImg), title('Original Binary');
minBinary = ordfilt2(double(binaryImg), 1, ones(3,3));
medianBinary = medfilt2(double(binaryImg), [3 3]);
maxBinary = ordfilt2(double(binaryImg), 9, ones(3,3));
subplot(2,4,6), imshow(minBinary), title('Min Filter - Binary');
subplot(2,4,7), imshow(medianBinary), title('Median Filter - Binary');
subplot(2,4,8), imshow(maxBinary), title('Max Filter - Binary');
```


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Output:



Output Windows You'll See

- **The first row will show:**
  1. Original grayscale image
  2. Minimum filtered grayscale image
  3. Median filtered grayscale image
  4. Maximum filtered grayscale image

- **The second row will show:**
  1. Original binary image (0s & 1s)
  2. Minimum filtered binary image
  3. Median filtered binary image
  4. Maximum filtered binary image

Conclusion:

- **Min filtering** is best for removing bright noise (salt) but may erode object boundaries.
- **Median filtering** offers the best balance, removing salt-and-pepper noise while preserving shapes and edges.
- **Max filtering** is best for removing dark noise (pepper) but can cause objects to grow in size.
- **Binary images**(0s & 1s) also respond to these filters, but the result is purely logical intensity changes rather than tonal variations.

On **grayscale images**, Median filtering provides the most visually pleasing results for mixed noise removal.

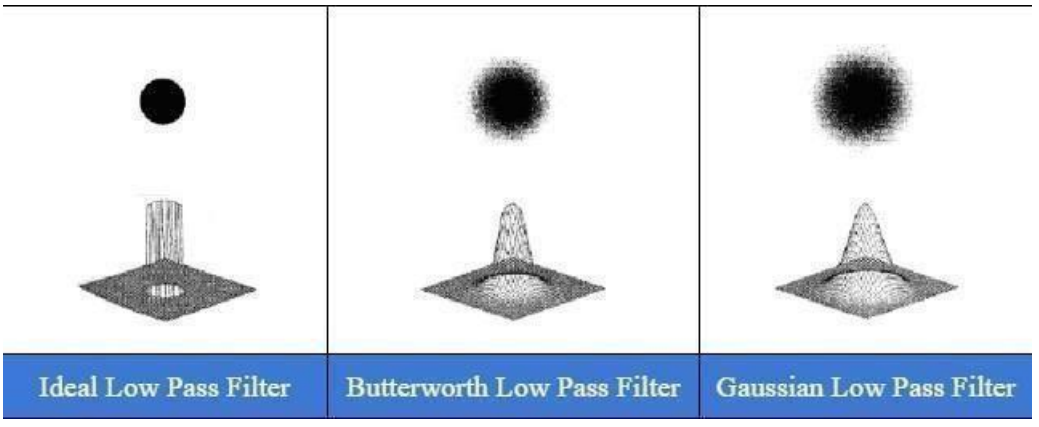
Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

PRACTICAL-09

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

Objective: To demonstrate the effect of frequency domain filtering by applying Ideal Low Pass Filter (ILPF), Butterworth Low Pass Filter (BLPF), and Gaussian Low Pass Filter (GLPF) on a grayscale image, and to analyze how each filter blurs the image by reducing high-frequency components.

Description (Theory in brief):

- **Image Blurring** can be achieved in the frequency domain by applying Low Pass Filters (LPF) that allow low frequencies (smooth regions) to pass and attenuate high frequencies (edges & details).
- **Types of LPFs:**
  - **Ideal LPF (ILPF):** Sharp cutoff; removes all frequencies beyond cutoff. Causes ringing effect.
  - **Butterworth LPF (BLPF):** Smooth transition; order *n* controls sharpness. Less ringing than ILPF.
  - **Gaussian LPF (GLPF):** Smoothest filter; no ringing, Gaussian-shaped frequency response.
  - 

Code:

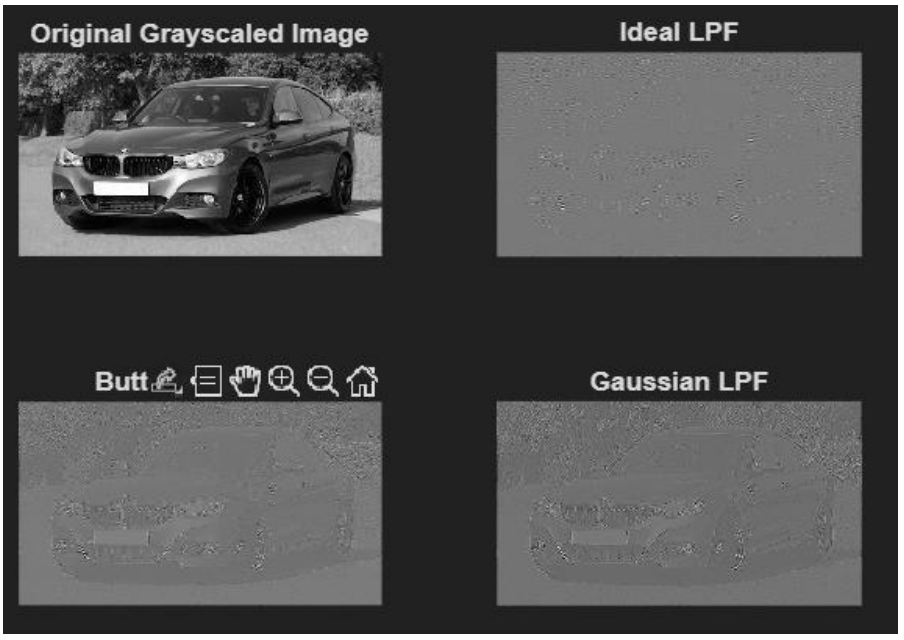
```
clc;
clear all;
close all;
% Load grayscale image
img = imread('car.jpeg');
if size(img,3)==3
    img = rgb2gray(img);
end
```


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

```
figure; imshow(img); title('Original Image');
% FFT of image
F = fft2(double(img));
Fshift = fftshift(F);
% Get image size
[M, N] = size(img);
u = 0:(M-1);
v = 0:(N-1);
idx = find(u > M/2); u(idx) = u(idx) - M;
idy = find(v > N/2); v(idy) = v(idy) - N;
[V,U] = meshgrid(v,u);
D = sqrt(U.^2 + V.^2);    % distance matrix
% Cutoff frequency
D0 = 100;
%% 1. Ideal Low Pass Filter
H_ideal = double(D <= D0);
G_ideal = H_ideal .* Fshift;
ideal_img = real(ifft2(ifftshift(G_ideal)));
ideal_img = mat2gray(ideal_img);
%% 2. Butterworth Low Pass Filter
n = 2; % order
H_butter = 1 ./ (1 + (D./D0).^(2*n));
G_butter = H_butter .* Fshift;
butter_img = real(ifft2(ifftshift(G_butter)));
butter_img = mat2gray(butter_img);
%% 3. Gaussian Low Pass Filter
H_gaussian = exp(-(D.^2) ./ (2*(D0^2)));
G_gaussian = H_gaussian .* Fshift;
gaussian_img = real(ifft2(ifftshift(G_gaussian)));
gaussian_img = mat2gray(gaussian_img);
%% Display results
figure;
subplot(2,2,1); imshow(img); title('Original Grayscaled Image');
subplot(2,2,2); imshow(ideal_img, []); title('Ideal LPF');
subplot(2,2,3); imshow(butter_img, []); title('Butterworth LPF');
subplot(2,2,4); imshow(gaussian_img, []); title('Gaussian LPF');
```


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Output:



Output Windows You'll See

- **Original Image** – the normal grayscale image (before filtering).
- **Ideal Low Pass Filter Output** – image blurred but with ringing/distortion at edges (due to sharp cutoff in frequency domain).
- **Butterworth Low Pass Filter Output** – smoother blur, less distortion compared to Ideal filter.
- **Gaussian Low Pass Filter Output** – most natural blur, smooth and without ringing artifacts.

Conclusion:

- Low Pass Filters are used to remove high-frequency components (edges, noise) and keep low-frequency parts (smooth areas).
- **Ideal LPF** produces strong blur but introduces ringing artifacts due to abrupt cutoff.
- **Butterworth LPF** provides controlled blurring with smoother transition, reducing ringing.
- **Gaussian LPF** gives the best visual quality with natural blur and no ringing.

Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

PRACTICAL-10

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

Objective: To demonstrate the effect of frequency domain filtering by applying Ideal Low Pass Filter (ILPF), Butterworth Low Pass Filter (BLPF), and Gaussian Low Pass Filter (GLPF) on a grayscale image, and to analyze how each filter blurs the image by reducing high-frequency components.

Description (Theory in brief):

High-Pass Filtering in Image Processing

High-pass filters are used to emphasize edges, fine details, and sharp transitions in an image by removing low-frequency components (smooth regions).

1. **Ideal High-Pass Filter (IHPF):**
   - Sharp cutoff at frequency D0.
   - Passes all frequencies higher than D0 and blocks lower ones.
   - Produces ringing (Gibbs effect).

2. **Butterworth High-Pass Filter (BHPF):**
   - Smooth transition between low and high frequencies.
   - Controlled by order n. Higher n → sharper cutoff.
   - Reduces ringing compared to Ideal.

3. **Gaussian High-Pass Filter (GHPF):**
   - Smoothest filter with no ringing.
   - Based on Gaussian function.
   - Preserves edges while minimizing artifacts.

Code:

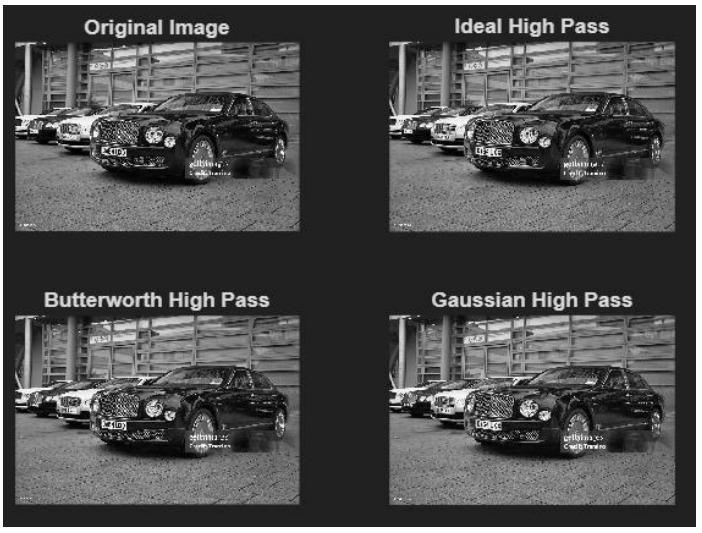
```
clc;
clear all;
close all;
% Read and convert to grayscale
i = imread('bentley.jpg');
i = rgb2gray(i);
[M, N] = size(i);
% Fourier Transform
FT_img = fft2(double(i));
FT_img = fftshift(FT_img); % Center the spectrum
```


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

```
% Cut-off frequency & order
D0 = 20;
n = 2;
% Frequency grid
u = 0:(M-1);
v = 0:(N-1);
idx = find(u > M/2);
u(idx) = u(idx) - M;
idy = find(v > N/2);
v(idy) = v(idy) - N;
[V, U] = meshgrid(v, u);
% Distance matrix
D = sqrt(U.^2 + V.^2);
% High-pass filters
Hil = double(D > D0);                    % Ideal HPF
Hbh = 1 ./ (1 + (D0 ./ D).^(2*n));       % Butterworth HPF
Hgh = 1 - exp(-(D.^2) / (2*(D0^2)));     % Gaussian HPF
% Apply filters
Gih = Hil .* FT_img;
Gbh = Hbh .* FT_img;
Ggh = Hgh .* FT_img;
% Inverse transform
output_image1 = real(ifft2(ifftshift(Gih)));
output_image2 = real(ifft2(ifftshift(Gbh)));
output_image3 = real(ifft2(ifftshift(Ggh)));
% Display results
figure();
subplot(2,2,1); imshow(i); title('Original Image');
subplot(2,2,2); imshow(output_image1, []); title('Ideal High Pass');
subplot(2,2,3); imshow(output_image2, []); title('Butterworth High Pass');
subplot(2,2,4); imshow(output_image3, []); title('Gaussian High Pass');
```


Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)

Output:



Output Windows You'll See

- **Original Image** – unprocessed grayscale image.
- **Ideal High Pass** – strong edges but ringing artifacts around objects.
- **Butterworth High Pass** – smoother edges, less ringing than Ideal.
- **Gaussian High Pass** – cleanest result, enhances edges naturally.

Conclusion:

- High Pass Filters remove **low-frequency components (smooth background)** and preserve **high-frequency components (edges & fine details)**.
- **Ideal HPF** highlights edges but introduces distortion (ringing).
- **Butterworth HPF** balances edge sharpness with fewer artifacts.
- **Gaussian HPF** produces the most natural edge enhancement.

Name:B.Nanda Kishore
Enrollment No: 2203031240124 (Roll no: 50)
Division: 7AI-15 CSE (AI)