PRACTICAL:5

AIM: Write MATLAB code to perform the bit-plane slicing method on grayscale image. Software/ Hardware:

Objective: The objective of the experiment is to,

Gets basic information of the bit-plane slicing on grayscale image.

Display all techniques on Matlab using Matlab code.

Syntax: Syntax to be covered in this experiment,

Theory:

Bit-plane slicing is a technique used in image processing to decompose an image into its binary bit-planes. Each bit-plane represents a specific bit (0 or 1) of the pixel values in the image. This method is useful for various

Code:

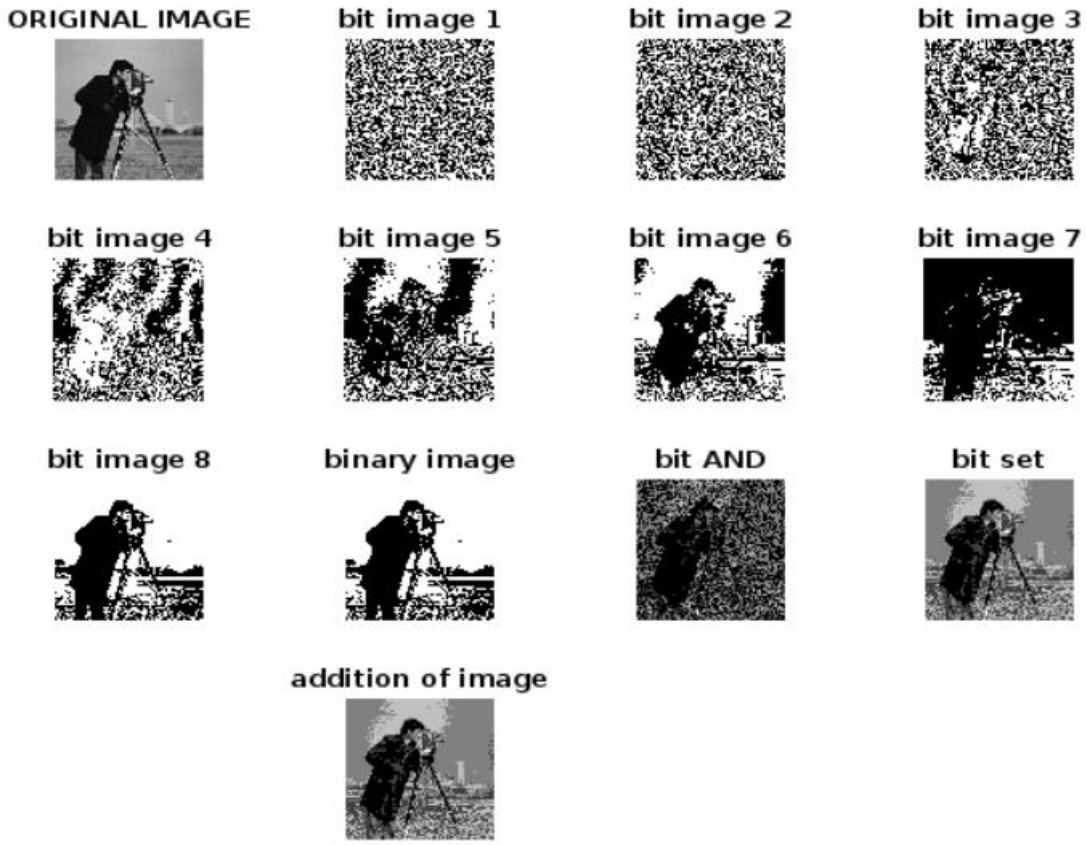
```
clc;
clear all;
close all;
i=imread('CameraMan.tif');

a1=bitget(i,1);
a2=bitget(i,2);
a3=bitget(i,3);
a4=bitget(i,4);
a5=bitget(i,5);
a6=bitget(i,6);
a7=bitget(i,7);
a8=bitget(i,8);
a9=im2bw(i);
figure()
imshow(i);
figure()
subplot(4,4,1),imshow(i),title('ORIGINAL IMAGE');
subplot(4,4,2),imshow(a1, []),title('bit image 1');
subplot(4,4,3),imshow(a2, []),title('bit image 2');
subplot(4,4,4),imshow(a3, []),title('bit image 3');
subplot(4,4,5),imshow(a4, []),title('bit image 4');
subplot(4,4,6),imshow(a5, []),title('bit image 5');
subplot(4,4,7),imshow(a6, []),title('bit image 6');
```


ERP Number : 2 1 0 3 0 3 1 2 4 3 2 8

```
subplot(4,4,8),imshow(a7, []),title('bit image 7');
subplot(4,4,9),imshow(a8, []),title('bit image 8');
subplot(4,4,10),imshow(a9, []),title('bit image');
comb2=imlincomb(1,a5,1,a6,1,a7,1,a8);
bit1=bitand(comb2,i);
bit2=bitset(comb2,8);
subplot(4,4,11),imshow(bit1, []),title('bit and');
subplot(4,4,12),imshow(bit2, []),title('bit set');
subplot(4,4,14),imshow(comb2, []),title('addition of image');
```

INPUT IMAGE:

SIMULATION RESULTS:



CONCLUSION:

Histogram equalization is an effective technique for enhancing the contrast and visual quality of an image. By spreading the pixel intensities over the entire range, it improves the visibility of details, especially in images with low contrast.

EXPERIMENT:06

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

Hardware and Software Used:

- **Hardware:** A computer with MATLAB installed.
- **Software:** MATLAB (preferably R2020b or later) with Image Processing Toolbox.

Objective:

The main objective of this experiment is to visually and numerically verify the convolution theorem, which states that the convolution of two signals in the spatial domain is equivalent to the element-wise multiplication of their Fourier transforms in the frequency domain.

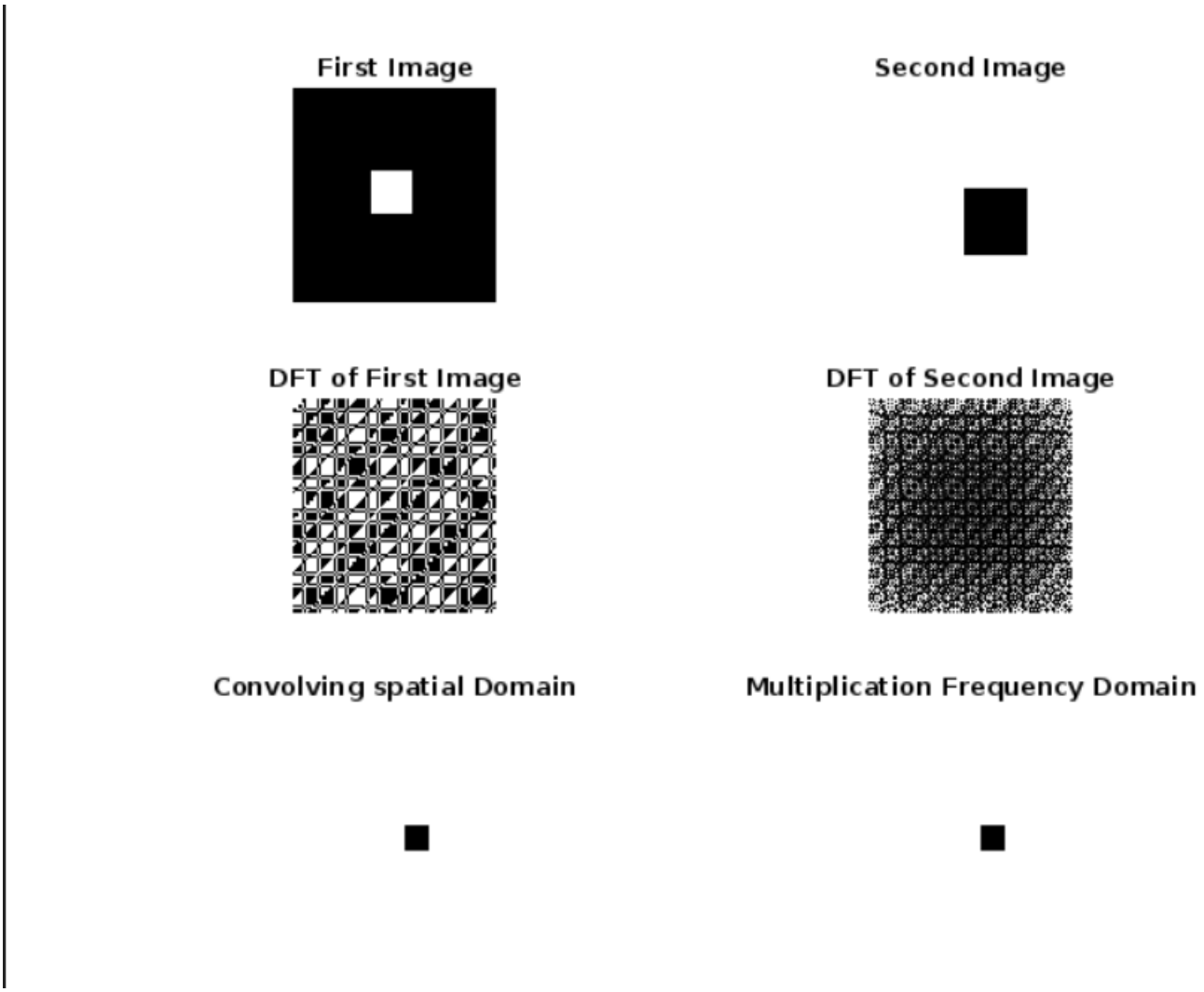
Theory: The convolution theorem states that the convolution of two signals in the time domain is equivalent to the multiplication of their Fourier transforms in the frequency domain.

This property is fundamental in signal processing, allowing for efficient computation and analysis of linear systems. It has applications in various fields such as image processing, audio processing, and communication systems.

CODE:

```
clc;
clear;
close all;
A=zeros(256);
[m,n]=size(A);
for i= 100:150
  for j = 100:150
    A(i,j) = 255;
  end
```

```
end
imshow(A), title("First Image")
B = ones(256);

for i= 120:200
   for j = 120:200
      B(i,j) = 0;
   end
end

imshow(B), title("Second Image")
C=conv2(A,B, "same");
A1= fft2(A);
figure, imshow(A1), title("DFT of first image");
B1= fft2(B);
figure, imshow(B1), title("DFT of second image");
C1=A1.*B1;
D=fftshift(ifft2(C1));

subplot(3,2,1),imshow(A),title("First Image");
subplot(3,2,2),imshow(B),title("Second Image");
subplot(3,2,3),imshow(A1),title("DFT of First Image");
subplot(3,2,4),imshow(B1),title("DFT of Second Image");
subplot(3,2,5),imshow(C),title("Convolving spatial Domain");
subplot(3,2,6),imshow(D),title("Multiplication Frequency Domain");
```



CONCLUSION:

The provided MATLAB code demonstrates the equivalence of convolution in the spatial domain and multiplication in the frequency domain. The results obtained from both methods should be visually similar, and the error between them should be negligible. This experiment provides a practical understanding of the convolution theorem and its applications in image processing.

EXPERIMENT:07

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

Hardware and Software Used:

- **Hardware:** A computer with MATLAB installed.
- **Software:** MATLAB (preferably R2020b or later) with Image Processing Toolbox.

Objective: The objective of the experiment is to,Get basic information of sampling and quantization techniques on grayscale image.Display all techniques on MATLAB using MATLAB code.

Syntax Overview:

1. **Reading an Image:**
   image = imread('filename');
   - **Purpose**: Reads an image from a file.
   - **Example**: noisyImage = imread('C:\Users\JAGAN\Desktop\dog.jpg');
2. **Converting to Grayscale:**
   grayImage = rgb2gray(image);
   - **Purpose**: Converts an RGB image to a grayscale image.
   - **Example**: noisyImage = rgb2gray(noisyImage);
3. **Displaying an Image:**
   imshow(image);
   - **Purpose**: Displays an image in a figure window.
   - **Example**:

     ```
     figure;
     imshow(noisyImage);
     title('Noisy Image');
     ```
4. **Applying Median Filtering:**

   filteredImage = medfilt2(image);
   - **Purpose**: Applies a median filter to the image to reduce noise.
   - **Example**: medianFiltered = medfilt2(noisyImage);
5. **Applying Wiener Filtering:**
   filteredImage = wiener2(image, [m n]);

- **Purpose**: Applies Wiener filter to the image for noise reduction.
- **Parameters**:
  - image: The input image.
  - [m n]: Neighborhood size.
- **Example**: wienerFiltered = wiener2(noisyImage, [5 5]);

6. **Saving an Image:**
   imwrite(image, 'filename');

   - **Purpose**: Writes an image to a file.

Theory:

Image restoration techniques are essential for improving the quality of images corrupted by noise. The goal is to recover the original image by reducing or eliminating the noise while preserving important details. Common techniques for restoring grayscale images from noisy images include median filtering and Wiener filtering.

Median Filtering

Median filtering is a nonlinear method that reduces noise by replacing each pixel's value with the median value of its neighboring pixels. It is particularly effective at removing salt-and-pepper noise (impulsive noise) while preserving edges and fine details.

Wiener Filtering

Wiener filtering is a linear method based on statistical principles. It aims to minimize the mean square error between the restored image and the original image. This filter adapts to the local mean and variance of the image, making it effective for reducing Gaussian noise.

Combined Filtering Approach

A combined approach using both median and Wiener filtering can enhance the noise reduction process. By applying median filtering first to remove impulsive noise and then Wiener filtering to reduce Gaussian noise, a more comprehensive noise reduction can be achieved.

Implementation in MATLAB

Using MATLAB, the process of restoring a grayscale image from a noisy image involves the following steps:

1. **Reading the Noisy Image**: Import the noisy grayscale image.
2. **Median Filtering**: Apply median filtering to reduce impulsive noise.
3. **Wiener Filtering**: Apply Wiener filtering to further reduce Gaussian noise.
4. **Combining Filters**: Combine the results of median and Wiener filtering for optimal restoration.
5. **Saving and Displaying**: Save and display the restored image for analysis.

Code:

```
% Read the noisy grayscale image

noisyImage = imread('C:\Users\Jagan\desktop\dog.jpg');

 % Convert to grayscale if the image is not already

if size(noisyImage, 3) == 3

   noisyImage = rgb2gray(noisyImage);
```

```
end

% Display the original noisy image
figure;
imshow(noisyImage);
title('Noisy Image');

% Apply median filtering for noise reduction
medianFiltered = medfilt2(noisyImage);

% Display the median filtered image
figure;
imshow(medianFiltered);
title('Median Filtered Image');

% Apply Wiener filtering for noise reduction
wienerFiltered = wiener2(noisyImage, [5 5]);

% Display the Wiener filtered image
figure;
imshow(wienerFiltered);
title('Wiener Filtered Image');

% Combine median and Wiener filtering
combinedFiltered = wiener2(medianFiltered, [5 5]);

% Display the combined filtered image
figure;
imshow(combinedFiltered);
```

```
title('Combined Median and Wiener Filtered Image');


% Save the restored image
imwrite(combinedFiltered, 'C:\Users\Jagan\Desktop\restored_image.png');
 % Display the restored image
figure;
imshow('C:\Users\Jagan\Desktop\restored_image.png');
title('Restored Image');
```

INPUT IMAGE:

OUT IMAGES:

Median Filtered Image

Wiener Filtered Image

Combined Median and Wiener Filtered Image

ERP Number : 2 1 0 3 0 3 1 2 4 3 2 8

CONCLUSION:

The provided MATLAB code demonstrates various image restoration techniques to remove noise from a grayscale image. The choice of filtering technique depends on the type of noise present in the image. Median filtering is effective for salt-and-pepper noise, while Wiener filtering is suitable for Gaussian noise. Mean filtering can be used to reduce general noise but may introduce blurring. By comparing the results from different techniques, you can select the most appropriate one for your specific application.