Aim:

To study and perform point-point processing operations used in image enhancement.

1. Digital Negative
2. Log Transformation
3. Gamma-factor Transformation
4. Bit Plane Slicing
5. Grey Level Slicing

1) Digital Negative

Input Image:

Code:

```
import cv2
import numpy as np
img=cv2.imread("bone.jpg",0)
negative_img = 255 - img
img=cv2.resize(img,(255,255))
negative_img=cv2.resize(negative_img,(255,255))
cv2.imshow("Original Image", img)
cv2.imshow("Negative Image", negative_img)

cv2.waitKey()
```

Output:

2) Log Transformation

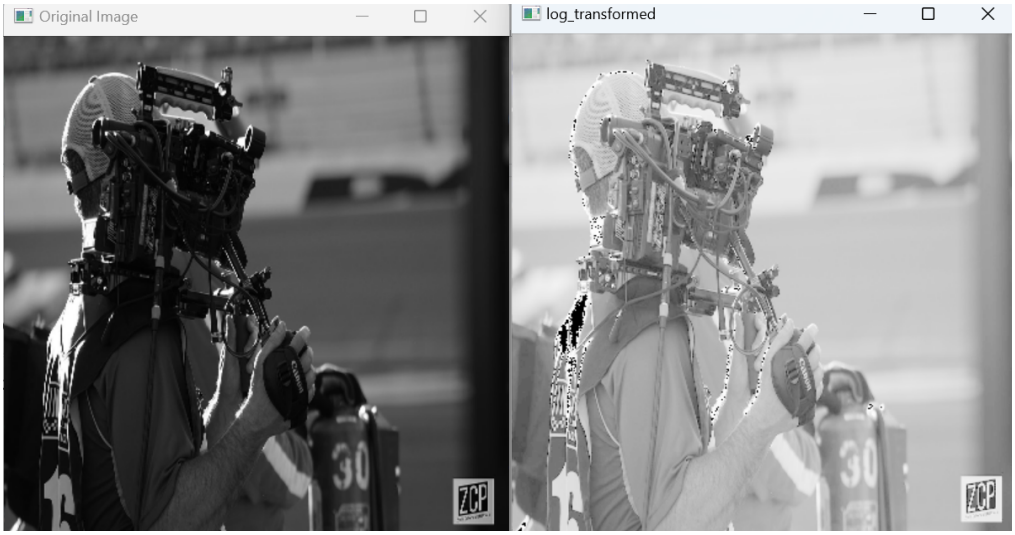
Input Image:

Code:

```
import cv2
import numpy as np

# Open the image.
img = cv2.imread('cameraman.jpg',cv2.IMREAD_GRAYSCALE)
# Apply log transform.
img=cv2.resize(img,(400,400))
c = 255/(np.log(1 + np.max(img)))
log_transformed = c * np.log(1 + img)


# Specify the data type.
log_transformed = np.array(log_transformed, dtype = np.uint8)
```

```
cv2.imshow("Original Image",img)
cv2.imshow('log_transformed', log_transformed)
cv2.waitKey()
```

Output:



3) Gamma Factor Transformation

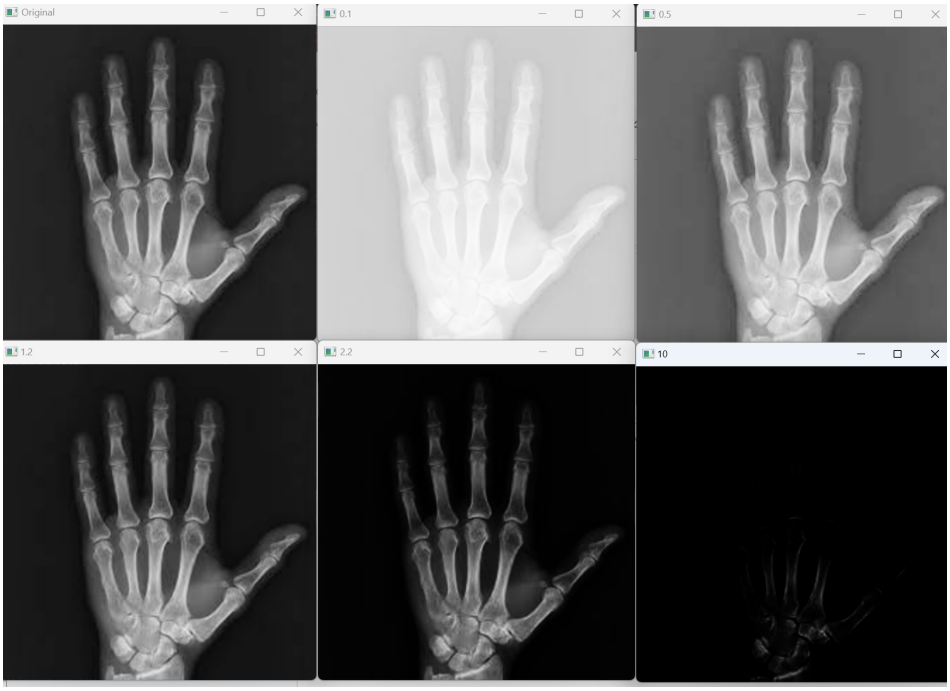
Input Image:

Code:

```
import cv2
import numpy as np

img = cv2.imread('bone.jpg',0)
img=cv2.resize(img,(400,400))
```

```
cv2.imshow("Original",img)
# Trying 4 gamma values.
for gamma in [0.1, 0.5, 1.2, 2.2,10]:
    gamma_corrected = np.array(255*(img / 255) ** gamma, dtype = np.uint8)
    cv2.imshow(f"{gamma}", gamma_corrected)

cv2.waitKey()
```

Output:



4) Bit Plane Slicing

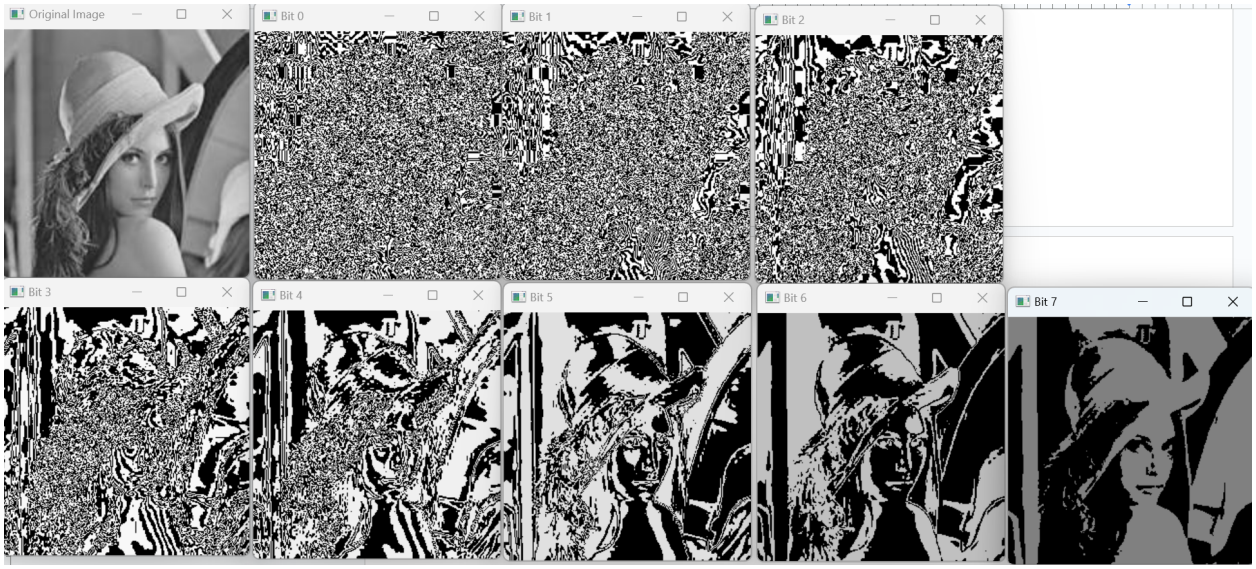
Input Image:

Code:

```
import numpy as np
import cv2

# Load the Lena image
lena = cv2.imread('lena.jpg', cv2.IMREAD_GRAYSCALE)
lena=cv2.resize(lena,(256,256))

# Bit plane slicing function
def bit_plane_slice(img, bit):
    # Create a mask with the specified bit position
    mask = 2**bit

    # Apply the mask to the image
    img_bit = np.bitwise_and(img, mask)

    # Convert the image to 8-bit for display
    img_bit = np.uint8(img_bit * 255)

    return img_bit

# Apply bit plane slicing to the Lena image
bit0 = bit_plane_slice(lena, 0)
bit1 = bit_plane_slice(lena, 1)
bit2 = bit_plane_slice(lena, 2)
bit3 = bit_plane_slice(lena, 3)
bit4 = bit_plane_slice(lena, 4)
bit5 = bit_plane_slice(lena, 5)
bit6 = bit_plane_slice(lena, 6)
bit7 = bit_plane_slice(lena, 7)

# Display the original image and the bit planes
cv2.imshow('Original Image', lena)
cv2.imshow('Bit 0', bit0)
cv2.imshow('Bit 1', bit1)
cv2.imshow('Bit 2', bit2)
cv2.imshow('Bit 3', bit3)
cv2.imshow('Bit 4', bit4)
cv2.imshow('Bit 5', bit5)
cv2.imshow('Bit 6', bit6)
cv2.imshow('Bit 7', bit7)
```

cv2.waitKey(0)

Output:



5) Grey Level Slicing

Input Image:

Code:

```
import cv2
import numpy as np
img = cv2.imread("bone.jpg", cv2.IMREAD_GRAYSCALE)

low = 100
high = 200
```

```
img=cv2.resize(img,(400,400))
# Creating a mask to identify pixels within the threshold range
mask = cv2.inRange(img, low, high)

# Apply the mask to the original image to get the grey level sliced image
sliced_img = cv2.bitwise_and(img, img, mask=mask)

# Display the original and grey level sliced images side by side
cv2.imshow("Original Image", img)
cv2.imshow("Grey Level Sliced Image", sliced_img)
cv2.waitKey(0)
```

Output: