**Batch: A-3 Roll No.: 16010122104**

**Experiment No. 5**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Title:** Implement the following point processing techniques in spatial domain:   * Image Negative. * Thresholding. * Gray level slicing with and without background * Bit plane slicing |

**Objective:** To learn & understand point processing techniques.

**Expected Outcome of Experiment:**

|  |  |
| --- | --- |
| **CO** | **Outcome** |
| **CO4** | Design & implement algorithms for digital image enhancement, segmentation & restoration. |

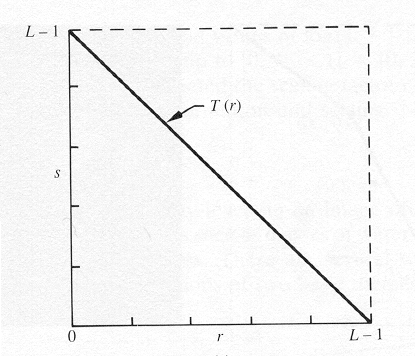
**Books/ Journals/ Websites referred:**

1. http://www.mathworks.com/support/
2. www.math.mtu.edu/~msgocken/intro/intro.html.
3. R. C.Gonsales R.E.Woods, “Digital Image Processing”, Second edition, Pearson Education
4. S.Jayaraman, S Esakkirajan, T Veerakumar “Digital Image Processing “Mc Graw Hill.
5. S.Sridhar,”Digital Image processing”, oxford university press, 1st edition."

**Pre Lab/ Prior Concepts:**

**Image Negative:**

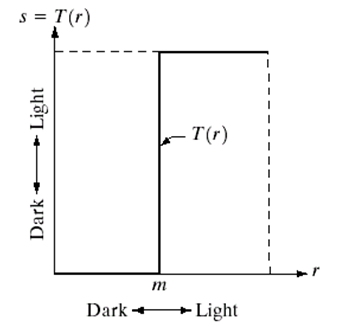
Negative images are useful for enhancing white or grey detail embedded in dark regions of an image. Image negatives are obtained by using the transformation function s=T(r).



[0,L-1] is the range of gray levels

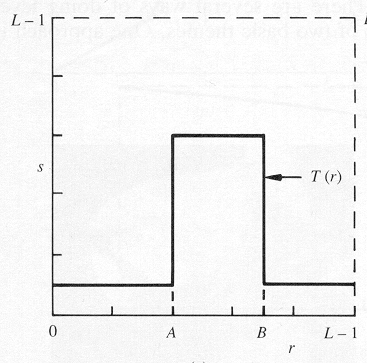
*S= L-*1*-r*

**Thresholding**

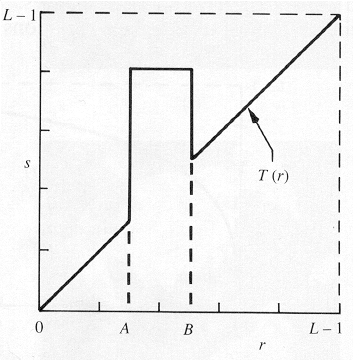
From a [grayscale](https://en.wikipedia.org/wiki/Grayscale) image, thresholding can be used to create [binary images](https://en.wikipedia.org/wiki/Binary_image). The simplest thresholding methods replace each pixel in an image with a black pixel if the image intensity  is less than some fixed constant T or a white pixel if the image intensity is greater than that constant. ****

**Gray Level Slicing**

To highlight a specific range of gray levels in an image (e.g. to enhance certain features). One way is to display a high value for all gray levels in the range of interest and a low value for all other gray levels (binary image).

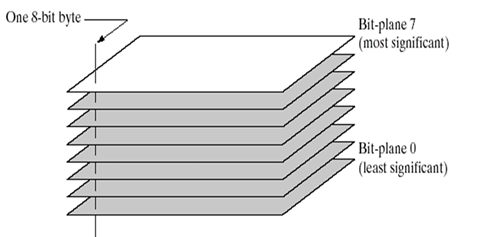


The second approach is to brighten the desired range of gray levels but preserve the background and gray-level tonalities in the image:



**Bit plane slicing**

Bit plane slicing is used to highlight the contribution made to the total image appearance by specific bits. Assuming that each pixel is represented by 8 bits, the image is composed of 8 1-bit planes. Plane 0 contains the least significant bit and plane 7 contains the most significant bit. Only the higher order bits (top four) contain visually significant data. The other bit planes contribute the more subtle details. Plane 7 corresponds exactly with an image thresholded at gray level 128.



|  |
| --- |
|  |

**Implementation steps with screenshots:**

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Function to read an image

def read\_image(image\_path):

    image = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)

    if image is None:

        raise FileNotFoundError(f"Image not found at the path: {image\_path}")

    return image

# 1. Image Negative

def image\_negative(image):

    return 255 - image

# 2. Thresholding

def thresholding(image, threshold\_value=128):

    \_, thresh\_image = cv2.threshold(image, threshold\_value, 255, cv2.THRESH\_BINARY)

    return thresh\_image

# 3. Gray Level Slicing with Background

def gray\_level\_slicing\_with\_background(image, lower\_bound, upper\_bound):

    sliced\_image = np.zeros\_like(image)

    sliced\_image[(image >= lower\_bound) & (image <= upper\_bound)] = 255

    return sliced\_image

# 4. Gray Level Slicing without Background

def gray\_level\_slicing\_without\_background(image, lower\_bound, upper\_bound):

    sliced\_image = np.zeros\_like(image)

    sliced\_image[(image >= lower\_bound) & (image <= upper\_bound)] = image[(image >= lower\_bound) & (image <= upper\_bound)]

    return sliced\_image

# 5. Bit Plane Slicing

def bit\_plane\_slicing(image):

    bit\_planes = []

    for i in range(8):

        bit\_plane = (image >> i) & 1

        bit\_plane = bit\_plane \* 255  # Convert to 0-255 range

        bit\_planes.append(bit\_plane)

    return bit\_planes

# Function to display images

def show\_images(images, titles):

    for i, img in enumerate(images):

        plt.subplot(1, len(images), i + 1)

        plt.imshow(img, cmap='gray')

        plt.title(titles[i])

        plt.axis('off')

    plt.show()

# Main function to process the image

def process\_image(image):

    while True:

        print("\nChoose an operation:")

        print("1. Image Negative")

        print("2. Thresholding")

        print("3. Gray Level Slicing with Background")

        print("4. Gray Level Slicing without Background")

        print("5. Bit Plane Slicing")

        print("6. Exit")

        choice = input("Enter your choice (1-6): ")

        if choice == '1':

            negative\_image = image\_negative(image)

            show\_images([image, negative\_image], ['Original Image', 'Negative Image'])

        elif choice == '2':

            threshold\_value = int(input("Enter threshold value (0-255): "))

            thresholded\_image = thresholding(image, threshold\_value)

            show\_images([image, thresholded\_image], ['Original Image', 'Thresholded Image'])

        elif choice == '3':

            lower\_bound = int(input("Enter lower bound for slicing (0-255): "))

            upper\_bound = int(input("Enter upper bound for slicing (0-255): "))

            sliced\_with\_bg = gray\_level\_slicing\_with\_background(image, lower\_bound, upper\_bound)

            show\_images([image, sliced\_with\_bg], ['Original Image', 'Gray Level Slicing with Background'])

        elif choice == '4':

            lower\_bound = int(input("Enter lower bound for slicing (0-255): "))

            upper\_bound = int(input("Enter upper bound for slicing (0-255): "))

            sliced\_without\_bg = gray\_level\_slicing\_without\_background(image, lower\_bound, upper\_bound)

            show\_images([image, sliced\_without\_bg], ['Original Image', 'Gray Level Slicing without Background'])

        elif choice == '5':

            bit\_planes = bit\_plane\_slicing(image)

            for i in range(0, 8, 2):  # Displaying only a few significant planes

                plt.subplot(1, 4, (i // 2) + 1)

                plt.imshow(bit\_planes[i], cmap='gray')

                plt.title(f'Bit Plane {i}')

                plt.axis('off')

            plt.show()

        elif choice == '6':

            print("Exiting...")

            break

        else:

            print("Invalid choice. Please enter a number between 1 and 6.")

# Upload an image

def upload\_image():

    image\_path = input("Enter the path to the image: ")

    return image\_path

# Main code to run the program

image\_path = upload\_image()  # Upload the image

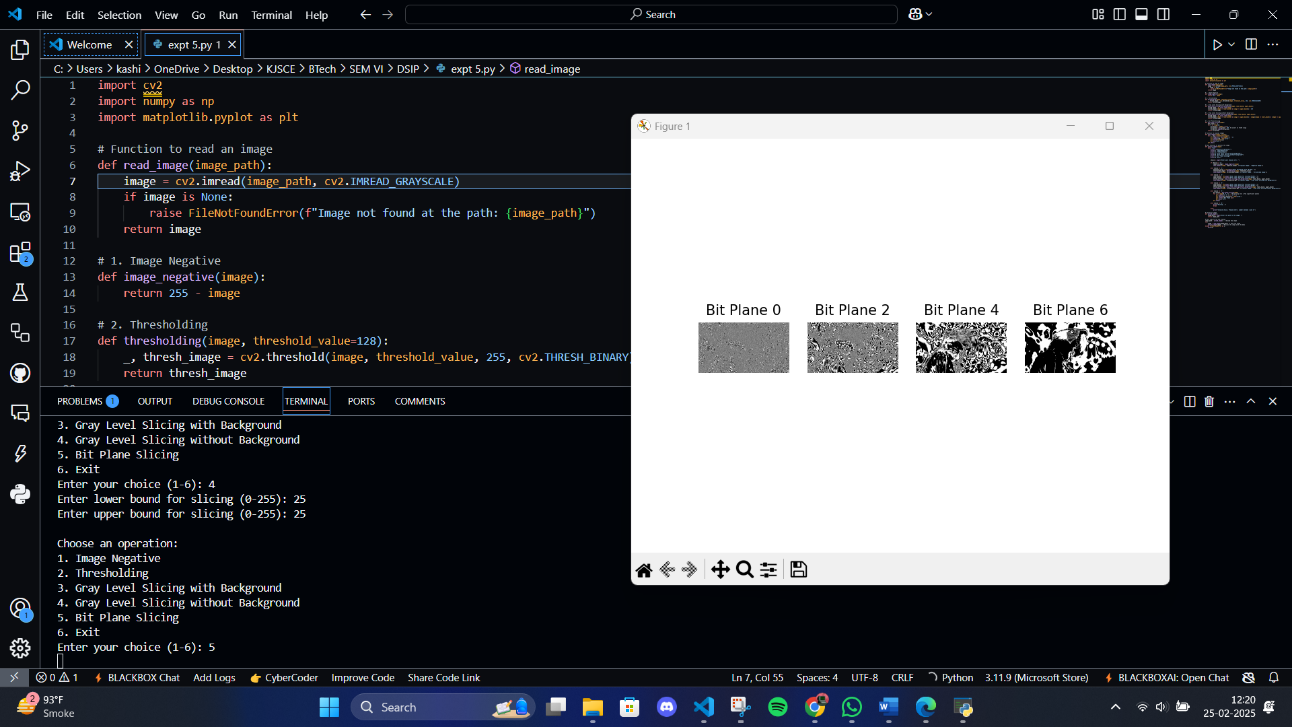
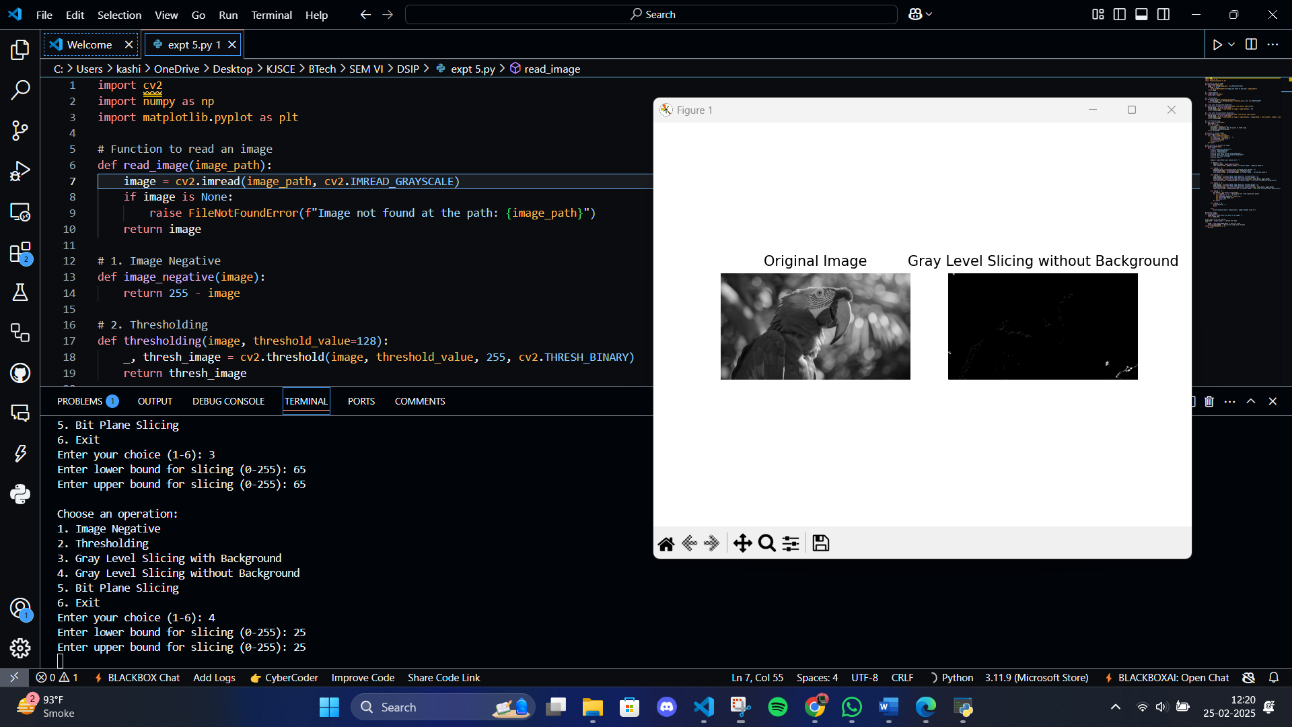
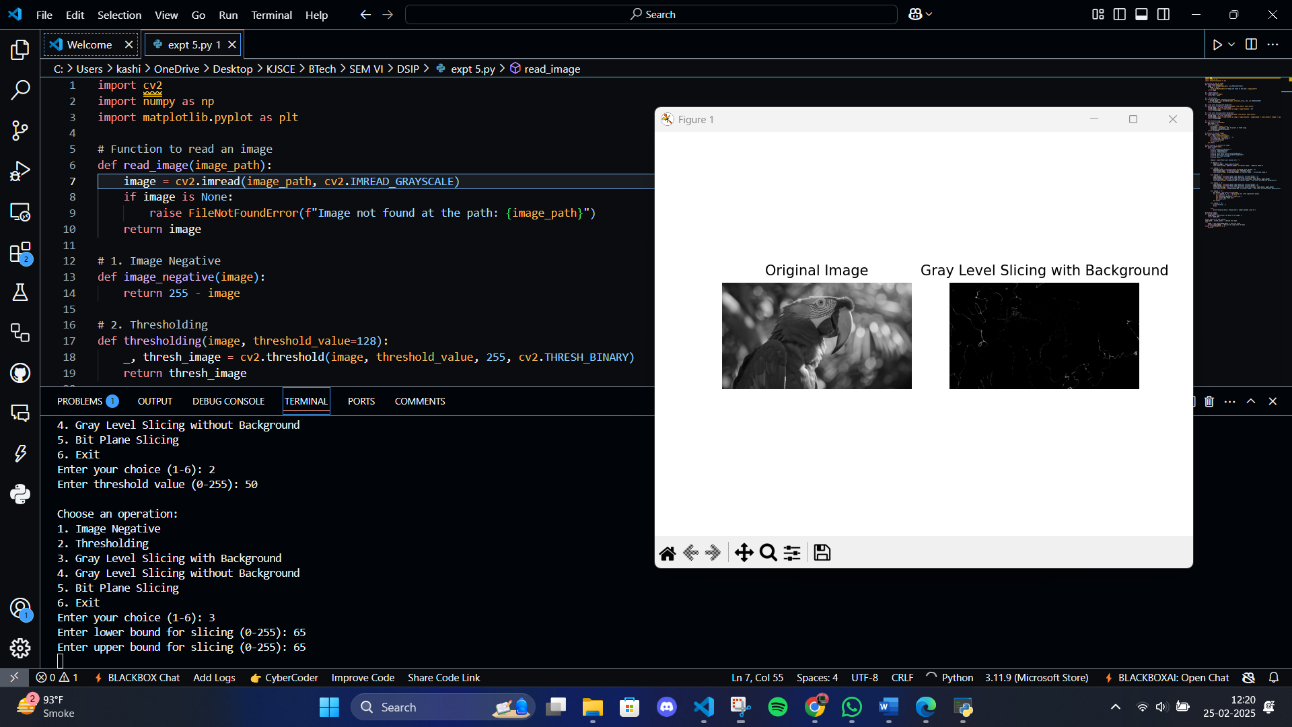
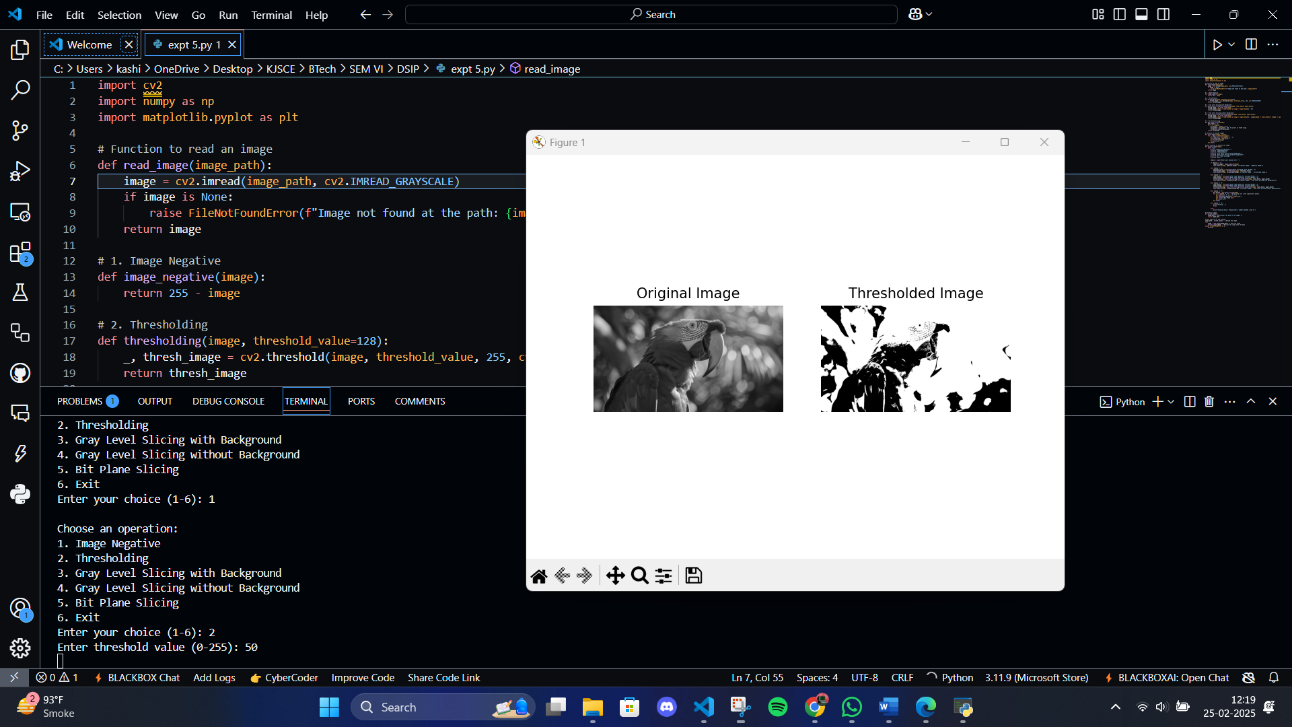
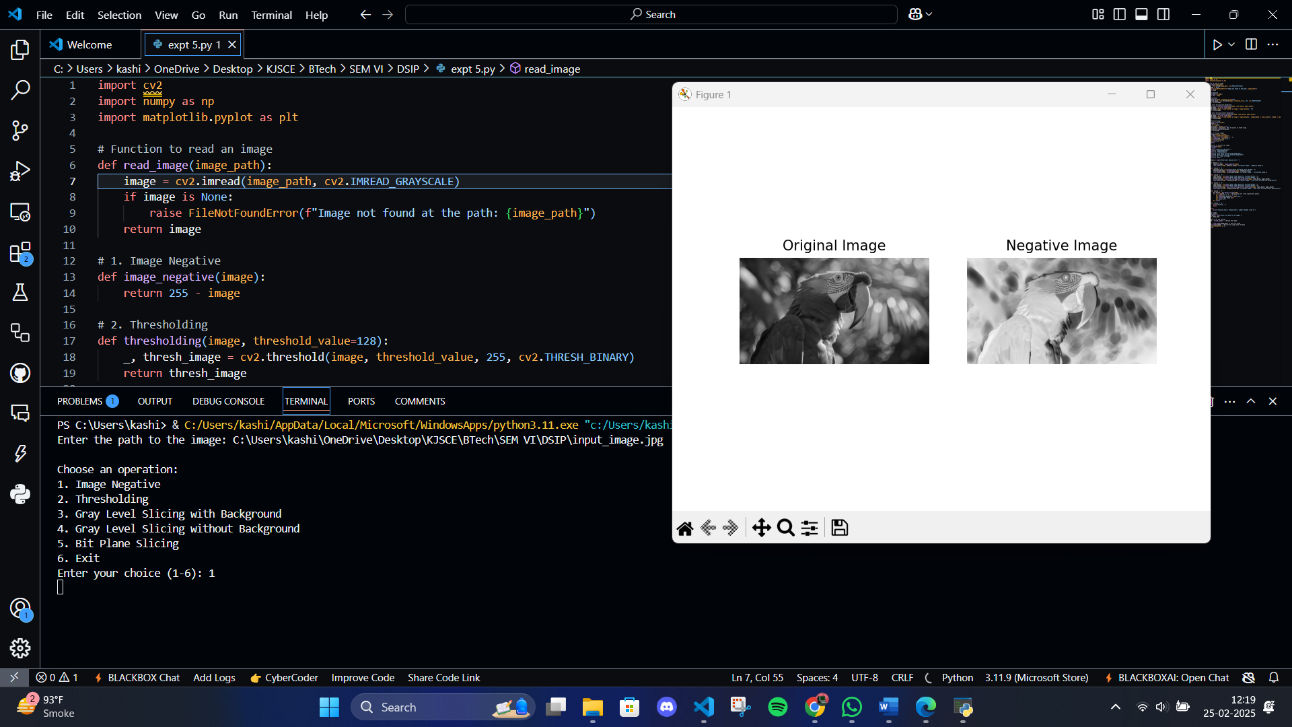
try:

    image = read\_image(image\_path)  # Read the image

    process\_image(image)  # Process the image with the menu

except FileNotFoundError as e:

    print(e)

****

**Conclusion:-**

Point processing techniques in the spatial domain, such as Image Negative, Thresholding, Gray Level Slicing, and Bit Plane Slicing, enable manipulation of pixel values for enhancing features, segmenting regions, and analysing image details, making them essential tools in image processing for various applications.

**Date: 25/02/2025 Signature of faculty in-charge**

**Post Lab Descriptive Questions**

1. Explain the role of bit plane slicing in achieving Steganography concept.

**Ans:**

Bit plane slicing plays a crucial role in achieving steganography by:

1. Providing a method to embed hidden data within specific bit planes of an image.
2. Allowing data to be hidden in the least significant bit (LSB) planes, which results in minimal visual distortion to the original image.
3. Enabling the embedding of large amounts of data. For example, in a 512x512 image, one plane can embed up to 262,144 characters.
4. Offering flexibility in choosing which bit planes to use for data hiding, balancing between capacity and imperceptibility.
5. Supporting the separation of an image into its constituent bit planes, which is useful for analyzing the relative importance of each bit and for image compression.
6. Explain the use of gray level slicing.

**Ans:**

Gray level slicing is used for:

1. Highlighting specific ranges of gray levels in an image while suppressing or maintaining other levels.
2. Enhancing contrast in specific intensity ranges of an image.
3. Emphasizing particular features or regions of interest in medical imaging (e.g., X-rays, CT scans) and satellite imagery.
4. Implementing two approaches:
   * Without background: Displaying high values for the desired range and low values for all other levels.
   * With background: Brightening the desired range while preserving original gray levels in other regions.
5. Performing band-pass filtering equivalent operations on image intensities.
6. Manipulating groups of intensity levels up to a specific range, which is particularly useful in medical and satellite image processing.