SCHOOL OF COMPUTER SCIENCE AND IT

DEPARTMENT OF CS & IT

BCA PROGRAMME

SEMESTER: V

SUBJECT NAME	: Image Processing
SUBJECT CODE	: 24BCA6IOE02

ACTIVITY 2

MINI PROJECT

Implementation of Spatial, Morphological and Thresholding Operations in Image Processing

Date of Submission: 12th February 2026

Submitted by:

[illegible]

23BCAR0339 – Somnath Gorai

Faculty In-Charge:

Dr. Zion Ramdinthara

Professor

JAIN DEEMED-TO-BE UNIVERSITY | SCHOOL OF COMPUTER SCIENCE AND IT

CERTIFICATE

This is to certify that Ms. /Mr. **Somnath Gorai** has satisfactorily completed Activity 2 prescribed by JAIN (Deemed to be University) for the 6th Semester BCA degree course in the year 2026.

Signature of the Faculty In Charge

Assessment Sheet with Rubrics for Grading & Evaluation

Students have to complete the online courses in the given timeline and submit the report as per format given.

Sr. No.	USN No.	Student Name	Report	Presentation	Viva-Voce	Total
			5 Marks	**5 Marks**	**5 Marks**	**15 Marks**
1	23BCAR0339	Somnath Gorai				

ABSTRACT

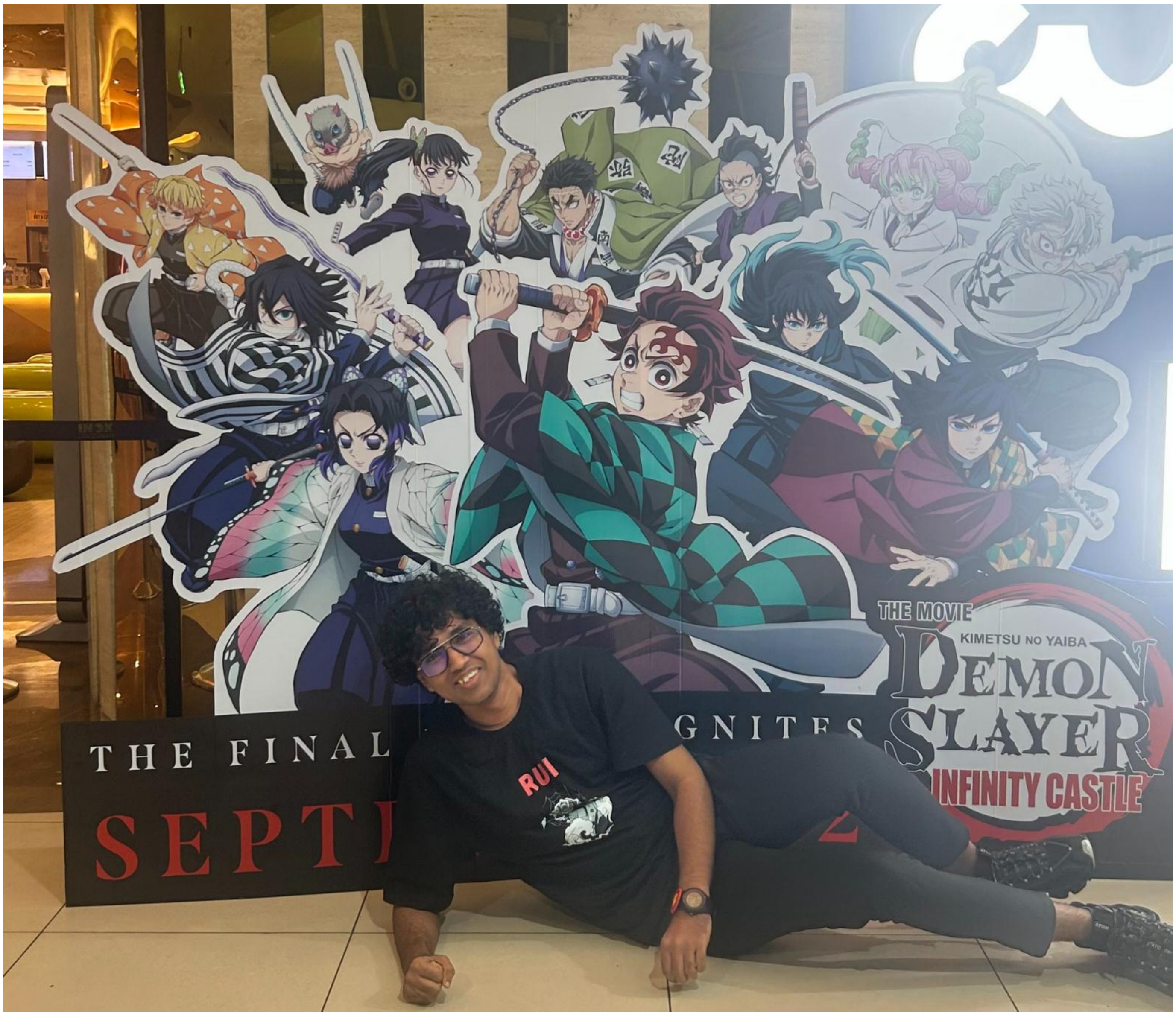
Image processing plays a crucial role in enhancing, analyzing, and interpreting digital images for various real-world applications such as medical imaging, remote sensing, surveillance, and computer vision. The primary objective of this activity is to study and implement fundamental image processing techniques using spatial filtering, morphological operations, feature detection, and segmentation methods. These techniques form the foundation for understanding how digital images can be manipulated and analyzed mathematically and computationally.

In this activity, Gaussian Blur, Image Sharpening, and Unsharp Masking are applied to enhance image quality and control image smoothness. Gaussian Blur is used to reduce noise and suppress high-frequency components by convolving the image with a Gaussian kernel. Image Sharpening enhances edges and fine details by emphasizing high-frequency components, while Unsharp Masking improves clarity by combining the original image with its blurred version. These techniques demonstrate how spatial domain filters influence image appearance and detail.

Morphological image processing operations such as Dilation, Erosion, Opening, and Closing are implemented to analyze and manipulate binary images based on their shapes. Dilation and erosion modify object boundaries, while opening and closing are used for noise removal and gap filling. These operations are particularly effective in processing images containing text and structured objects.
Further, Line Detection and Point Detection techniques are applied using suitable convolution masks to identify structural features within an image. Line detection highlights linear patterns and edges, whereas point detection identifies isolated pixels and sharp intensity variations. These operations are essential for feature extraction and pattern recognition tasks.

Finally, a Global Thresholding technique is implemented to segment an image into foreground and background regions based on a predefined threshold value. This method demonstrates a simple yet effective approach to image segmentation.
All algorithms are implemented using Python with standard development tools such as Spyder, Jupyter Notebook, Google Colab, PyCharm, and Visual Studio Code. The results obtained validate the theoretical concepts and highlight the practical importance of basic image processing techniques.

Input Image:



Above image is the input image that was being used in all the operation performed bellow.

Table of Contents

CHAPTER 1: GAUSSIAN BLUR, SHARPENING AND UNSHARP MASKING

1.1 Gaussian Blur

Gaussian Blur is a linear smoothing filter used to reduce noise and minor details in an image. It works by averaging pixel values with their neighboring pixels using weights derived from the Gaussian distribution. Pixels closer to the center of the kernel have higher influence than distant pixels. This operation suppresses high-frequency components such as noise and sharp edges, resulting in a smoother image.

Kernal:

Gaussian Blur smooths an image using a weighted averaging kernel derived from the Gaussian distribution. The Gaussian kernel used is:

$$G = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Algorithm:

1. Read the input image and convert it to grayscale.
2. Define the Gaussian kernel as given above.
3. Normalize the kernel by dividing each element by 16.
4. Convolve the kernel with the input image.
5. Obtain the smoothed output image.

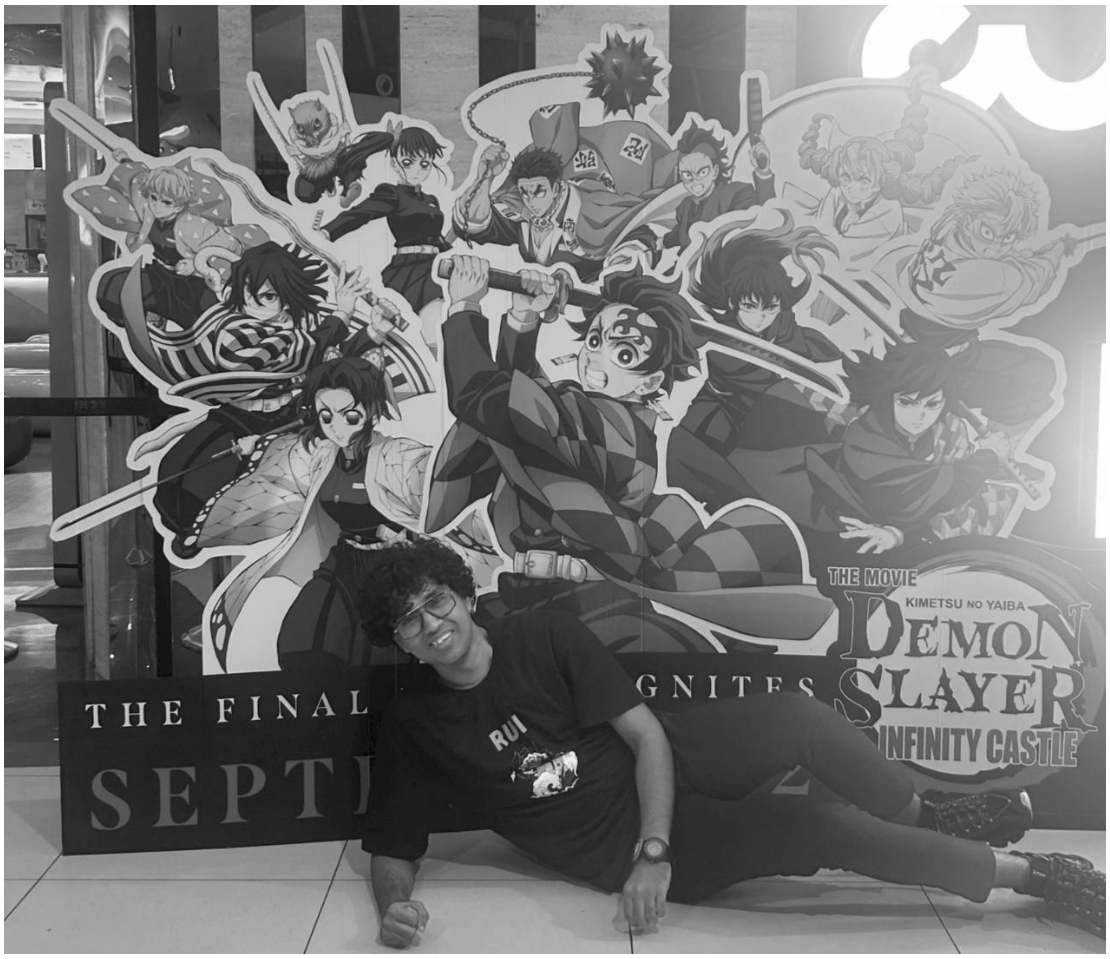
Code:

```
import cv2
import numpy as np
# Read image
img = cv2.imread("image.png", cv2.IMREAD_GRAYSCALE)
# Gaussian kernel as per textbook
kernel = (1/16) * np.array([[1, 2, 1],
                            [2, 4, 2],
                            [1, 2, 1]])
# Apply convolution
gaussian_blur = cv2.filter2D(img, -1, kernel)
# Save output
cv2.imwrite("gaussian_blur.png", gaussian_blur)
```

Code Explanation

- The input image is read in grayscale format.
- A 3×3 Gaussian kernel is defined according to the standard textbook approximation.
- The kernel is normalized by dividing by 16 to preserve image brightness.
- Convolution is performed using filter2D() to smooth the image.
- The resulting blurred image is saved.

Output Image:



Observation:

After applying the Gaussian filter using the 3×3 kernel, the image appears smoother with a noticeable reduction in noise and minor intensity variations. Fine details and sharp edges are slightly softened due to the averaging effect of the filter. The overall brightness of the image is preserved because the kernel is normalized, making Gaussian blur effective for noise reduction without significant loss of image information.

1.2 Image Sharpening

Image sharpening is a spatial domain technique used to enhance edges and fine details in an image. It works by emphasizing high-frequency components such as edges and boundaries, making the image appear clearer and more defined. Sharpening is commonly applied after smoothing operations to restore lost details.

Kernal:

Image sharpening can be achieved using a sharpening mask that highlights intensity differences between a pixel and its neighbors.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Algorithm:

1. Read the input image and convert it to grayscale.
2. Define the sharpening kernel as shown above.
3. Convolve the kernel with the input image.
4. Enhance edges and fine details.
5. Obtain the sharpened output image.

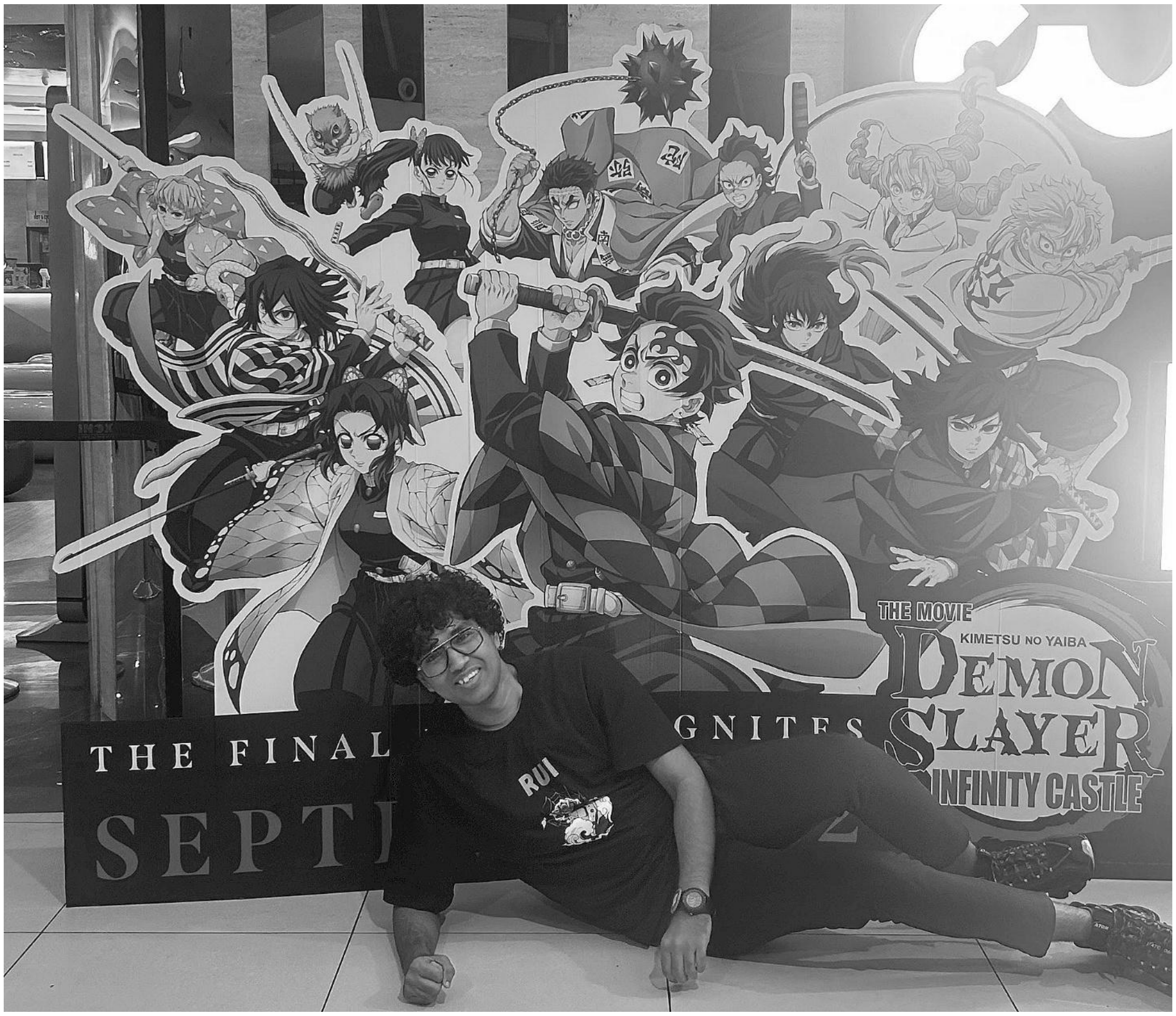
Code:

```
import cv2
import numpy as np

# Read image
img = cv2.imread("image.png", cv2.IMREAD_GRAYSCALE)

# Sharpening kernel
kernel = np.array([[0, -1, 0],
                   [-1, 5, -1],
                   [0, -1, 0]])

# Apply sharpening
sharpened = cv2.filter2D(img, -1, kernel)

# Save output
cv2.imwrite("sharpened.png", sharpened)
```

Code Explanation:

- The input image is read in grayscale format.
- A 3×3 sharpening kernel is defined to enhance edges.
- Convolution is applied using filter2D() to emphasize high-frequency components.
- The sharpened image is saved as the output.

Output Image:



Observation:

Edges and boundaries in the image appear more prominent after sharpening. Fine details are enhanced, making the image clearer compared to the original, though excessive sharpening may slightly increase noise.

CONCLUSION

REFERENCES