Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2024-25

Class:	**SE**	**Semester:**	**IV**
Course Code:	**CSL405**	**Course Name:**	**Skills Based Python Programming Lab**

Name of Student:	**Shravani Sandeep Raut**
Roll No. :	**48**
Experiment No.:	**10**
Title of the Experiment:	**Program to demonstrate use of NumPy array for working with images.**
Date of Performance:	**18/03/2025**
Date of Submission:	**25/03/2025**

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty : Mr. Raunak Joshi

Signature :

Date:

Aim: Program to demonstrate use of NumPy array for working with images.

Theory:

NumPy (Numerical Python) is a powerful library in Python used for numerical and scientific computing. It provides support for multi-dimensional arrays and a collection of high-level mathematical functions. NumPy arrays are widely used in image processing because images are essentially multi-dimensional arrays of pixel values.

Introduction to Digital Images:

- **Digital Image:** A digital image is a matrix of pixel values. Each pixel represents the color or intensity at a particular point.
- **Grayscale Image:** Contains only intensity values ranging from 0 (black) to 255 (white). It is represented as a 2D array.
- **Color Image (RGB):** Consists of three color channels — Red, Green, and Blue. It is represented as a 3D array with dimensions (height, width, 3).

Why Use NumPy for Image Processing?

1. **Efficient Storage and Computation:** NumPy arrays store pixel data compactly and allow fast mathematical operations using vectorization.
2. **Ease of Manipulation:** NumPy provides easy-to-use functions for manipulating arrays (reshaping, slicing, and mathematical operations).
3. **Integration with Other Libraries:** NumPy arrays are compatible with popular image processing libraries such as OpenCV, Pillow (PIL), and Matplotlib.

Reading and Displaying Images Using NumPy:

NumPy itself does not support reading or displaying images directly, but it can be combined with other libraries:

- **OpenCV (cv2):** Fast and efficient, widely used in computer vision tasks. Uses BGR color order.
- **PIL (Pillow):** User-friendly and compatible with many image formats.
- **Matplotlib:** Primarily used for data visualization but can also display images. Uses RGB color order.

Applications of NumPy in Image Processing:

- Image manipulation (cropping, resizing, flipping, rotating).
- Color space conversions (BGR to RGB, grayscale).
- Image filtering and enhancement.
- Object detection and image segmentation.
- Machine learning and computer vision tasks.

Implementation:

```
# pip install numpy
# pip install matplotlib
# pip install opencv-python
```

In [8]:

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
```

In [11]:

```
image = cv2.imread('new.jpg')
```

In [12]:

```
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

In [13]:

```
plt.imshow(image_rgb)
plt.title("Original Image")
plt.axis("off")
plt.show()
```

Original Image

In [14]:

```
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
plt.imshow(image_gray, cmap='gray')
plt.title("Grayscale Image")
plt.axis("off")
plt.show()
```

Grayscale Image

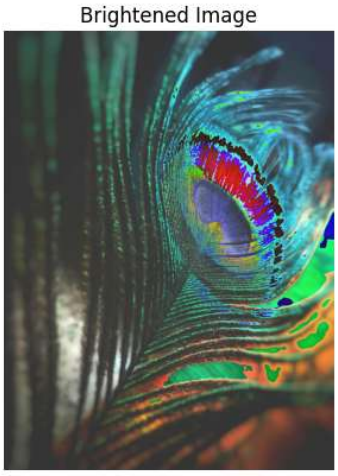
In [15]:

```
cropped_image = image_rgb[50:200, 50:200]
plt.imshow(cropped_image)
plt.title("Cropped Image")
```

```
plt.axis("off")
plt.show()
```

In [16]:

```
bright_image = np.clip(image_rgb + 50, 0, 255)
plt.imshow(bright_image.astype(np.uint8))
plt.title("Brightened Image")
plt.axis("off")
plt.show()
```



In [17]:

```
sobel_x = cv2.Sobel(image_gray, cv2.CV_64F, 1, 0, ksize=5)
sobel_y = cv2.Sobel(image_gray, cv2.CV_64F, 0, 1, ksize=5)
edge_image = np.sqrt(sobel_x**2 + sobel_y**2)
plt.imshow(edge_image, cmap='gray')
plt.title("Edge Detection")
plt.axis("off")
plt.show()
```

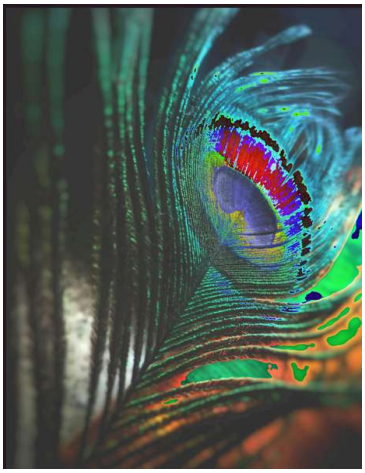
In [18]:

```
cv2.imwrite('modified_image.jpg', cv2.cvtColor(bright_image,
cv2.COLOR_RGB2BGR))
```

Out[18]:

```
True
```

Modified image -



Conclusion : NumPy provides an efficient and flexible way to work with images by treating them as multi-dimensional arrays.