

## Lab Cycle 1

**1. Install and configure computer vision tool/library/framework such as OpenCV, TensorFlow, or PyTorch.**

### **Downloading and Installing OpenCV:**

OpenCV can be directly downloaded and installed with the use of pip (package manager). To install OpenCV, just go to the command-line and type the following command:

**pip install opencv-python**

To check if OpenCV is correctly installed, just run the following commands to perform a version check:

```
import cv2  
print(cv2.__version__)
```

Output:

```
[Running] python -u "c:\Users\sss\Downloads\Project-Sphere-main (2)\New  
folder\g.py"  
4.12.0  
  
[Done] exited with code=0 in 0.499 seconds
```

### **OpenCV Library**

OpenCV (Open Source Computer Vision Library) is an open-source software toolkit for computer vision and machine learning tasks. Originally developed by Intel, it is now maintained by the OpenCV Foundation and a large community of contributors. OpenCV enables developers to process and analyze visual data such as images and videos with efficiency.

### **Installing TensorFlow**

TensorFlow is a powerful **deep learning framework** developed by Google. While OpenCV is mainly for image processing, TensorFlow lets you **train AI models** to recognize objects, classify images, or detect patterns in data. Some things TensorFlow allows you to do include:

- Image classification (e.g., distinguishing cats from dogs)

G. Narayanamma Institute of Technology and Science  
(Autonomous) (for women)  
Computer Science and Engineering Department  
Computer Vision and Pattern Recognition lab  
(2025-2026 II Semester)

- Object detection (e.g., finding multiple objects in a picture)
- Image segmentation (e.g., separating background from foreground)
- Building neural networks and deep learning pipelines

TensorFlow is widely used in both academia and industry, and it comes with lots of tutorials and pre-trained models, which is great for beginners.

### Installation of TensorFlow

Open your terminal or command prompt.

For beginners, install the CPU version first:

**pip install tensorflow**

### Verifying TensorFlow Installation

Open Python and type:

```
import tensorflow as tf  
print(tf.__version__)
```

### Installing PyTorch

PyTorch is another popular **deep learning library**, developed by Facebook (Meta). Many researchers and developers prefer PyTorch because it is **easy to debug and experiment with**. PyTorch is great for computer vision because it allows you to:

- Build neural networks easily with intuitive syntax
- Train models on custom datasets
- Use pre-trained models for object detection, image classification, and segmentation
- Experiment with research-level computer vision models

PyTorch is very flexible and allows you to work on advanced projects once you are comfortable with the basics.

## Installation of PyTorch

1. Open your terminal or command prompt.

Install PyTorch and some additional libraries for vision:

**pip install torch torchvision torchaudio**

- torch → main PyTorch library
- torchvision → tools specifically for computer vision tasks
- torchaudio → optional, for audio processing

## Verifying PyTorch Installation

Open Python and type:

**import torch**

**print(torch.\_\_version\_\_)**

## 5. Why These Libraries Are Useful

Here's why these three libraries are essential for anyone starting in computer vision:

- **OpenCV:** Great for beginners. Perfect for image and video processing, camera feed analysis, and traditional computer vision tasks.
- **TensorFlow:** Lets you build and train AI models. Ideal for tasks like image classification, object detection, and AI-driven computer vision.
- **PyTorch:** Flexible and research-friendly. Perfect for experimenting with new models, learning deep learning concepts, and working with advanced computer vision datasets.

By combining these tools, you can start with simple image processing in OpenCV, then gradually move to AI-based image analysis with TensorFlow or PyTorch. They complement each other and provide a full toolkit for computer vision projects.

**2. Write programs for the following**

**a) Loading and displaying an image.**

**b) Reading and writing video files.**

**c) Image enhancement.**

**a)**

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

image = cv2.imread("dog_backpack.png")
plt.imshow(image)
(height, width, channels) = image.shape
print("Height", height)
print("Width", width)
print("Channels", channels)
```

**Output:**

<matplotlib.image.AxesImage at 0x22fb092c7f0>



Height 1401  
Width 934  
Channels 3

G. Narayanamma Institute of Technology and Science  
(Autonomous) (for women)  
Computer Science and Engineering Department  
Computer Vision and Pattern Recognition lab  
(2025-2026 II Semester)

**b)**

```
import cv2
import time

import matplotlib.pyplot as plt
cap = cv2.VideoCapture('cctv052x2004080516x01638.avi')
if cap.isOpened() == False:
    print("Error opening the video file. Please check your file path.")
output = cv2.VideoWriter("output.avi", cv2.VideoWriter_fourcc(*'MPEG'), 25, (1080, 1920))

while cap.isOpened():
    ret, frame = cap.read()
    if ret == True:
        cv2.imshow("output", frame)
        if cv2.waitKey(25) & 0xFF == ord('q'):
            break
    else:
        break
print("No. of frames: ", ithFrame)
cap.release()
cv2.destroyAllWindows()
```

**Output:**



**c) Linear.py**

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

img = cv2.imread('image2.jpg', 0)
identity = img.copy()
negative = 255 - img

plt.figure(figsize=(10,4))
plt.subplot(1,3,1)
plt.title("Original Image")
plt.imshow(img, cmap='gray')

plt.axis('off')
plt.subplot(1,3,2)
plt.title("Identity Transformation")
plt.imshow(identity, cmap='gray')

plt.axis('off')
plt.subplot(1,3,3)
plt.title("Negative Transformation")
plt.imshow(negative, cmap='gray')
plt.axis('off')

plt.tight_layout()
plt.show()
```



**logarithmic.py**

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

img = cv2.imread('img.jpg', 0)
img_float = img.astype(np.float32)

c_values = [1, 5, 20]
log_images = []
inv_log_images = []

for c in c_values:
    log_img = c * np.log(1 + img_float)
    log_img = np.uint8(log_img)
    log_images.append(log_img)
    inv_log = c * (np.exp(img_float / 255) - 1)
    inv_log = np.uint8(inv_log)
    inv_log_images.append(inv_log)

plt.figure(figsize=(12, 6))
plt.subplot(2, 4, 1)
plt.imshow(img, cmap='gray')
plt.title('Original')
plt.axis('off')

for i in range(3):
    plt.subplot(2, 4, i + 2)
    plt.imshow(log_images[i], cmap='gray')
    plt.title(f'Log (c={c_values[i]})')
    plt.axis('off')

for i in range(3):
    plt.subplot(2, 4, i + 6)
    plt.imshow(inv_log_images[i], cmap='gray')
    plt.title(f'Inv Log (c={c_values[i]})')
    plt.axis('off')

plt.tight_layout()
plt.show()
```

G. Narayanamma Institute of Technology and Science  
(Autonomous) (for women)  
Computer Science and Engineering Department  
Computer Vision and Pattern Recognition lab  
(2025-2026 II Semester)

Original



Log (c=1)



Log (c=5)



Log (c=20)



Inv Log (c=1)



Inv Log (c=5)



Inv Log (c=20)



G. Narayanamma Institute of Technology and Science  
(Autonomous) (for women)  
Computer Science and Engineering Department  
Computer Vision and Pattern Recognition lab  
(2025-2026 II Semester)

**power.py**

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

img = cv2.imread('img.jpg', 0)
img_norm = img / 255.0
gamma_values = [0.4, 0.8, 1.5, 2.5]
gamma_images = []

for gamma in gamma_values:
    gamma_img = np.power(img_norm, gamma)
    gamma_img = np.uint8(gamma_img * 255)
    gamma_images.append(gamma_img)

plt.figure(figsize=(10, 5))
plt.subplot(1, 5, 1)
plt.imshow(img, cmap='gray')
plt.title('Original')
plt.axis('off')

for i in range(len(gamma_values)):
    plt.subplot(1, 5, i + 2)
    plt.imshow(gamma_images[i], cmap='gray')
    plt.title(f' $\gamma = \{gamma\_values[i]\}$ ')
    plt.axis('off')

plt.tight_layout()
plt.show()
```

**Output:**



**3. Perform the following operations on an image**

**a) Resize b) Rotation c) Flipping d) Cropping**

```
import cv2
from matplotlib import pyplot as plt
img = cv2.imread("bone.jpg", 0)
resized = cv2.resize(img, (300, 300))
rotated_90 = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
flipped = cv2.flip(img, 1)
cropped = img[50:250, 50:250]

plt.figure(figsize=(10, 6))

plt.subplot(2, 3, 1)
plt.title("Original")
plt.imshow(img, cmap='gray')

plt.subplot(2, 3, 2)
plt.title("Resized")
plt.imshow(resized, cmap='gray')

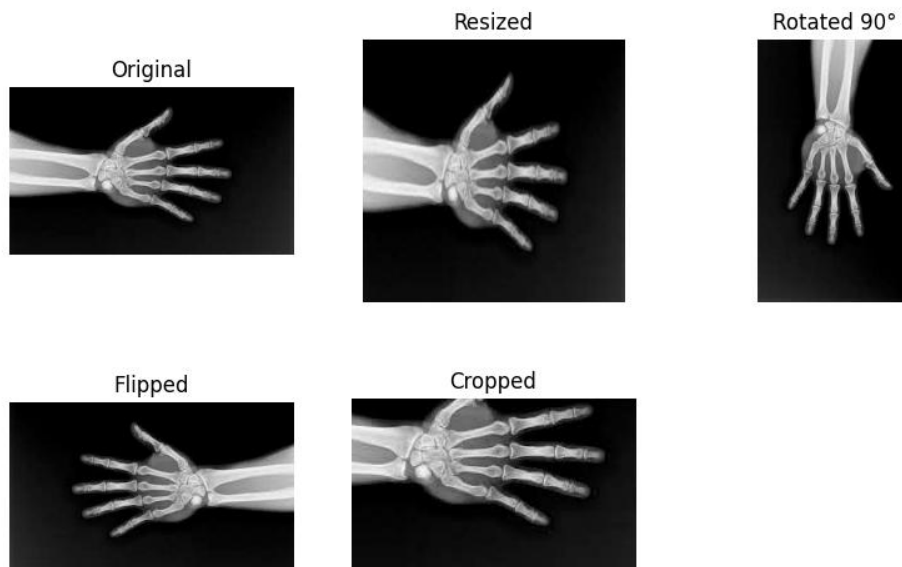
plt.subplot(2, 3, 3)
plt.title("Rotated 90°")
plt.imshow(rotated_90, cmap='gray')

plt.subplot(2, 3, 4)
plt.title("Flipped")
plt.imshow(flipped, cmap='gray')

plt.subplot(2, 3, 5)
plt.title("Cropped")
plt.imshow(cropped, cmap='gray')

plt.show()
```

G. Narayanamma Institute of Technology and Science  
(Autonomous) (for women)  
Computer Science and Engineering Department  
Computer Vision and Pattern Recognition lab  
(2025-2026 II Semester)



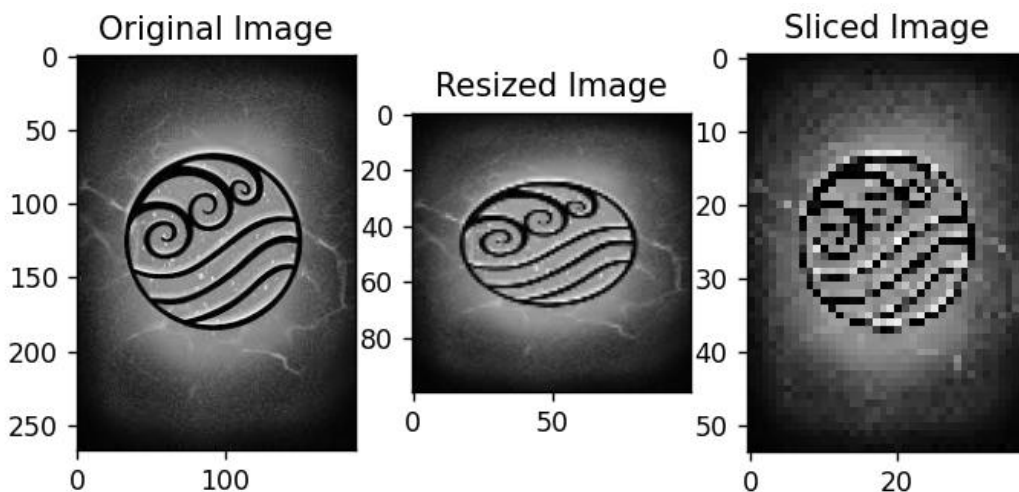
#### 4. Displaying an image with different Spatial resolutions and Intensity resolutions.

##### sampling.py

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

img=cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)
down=cv2.resize(img, (100,100)) #Sampling using resize
d1=img[::5,::5] #Sampling using slicing
plt.subplot(1,3,1)
plt.title('Original Image')
plt.imshow(img, cmap='gray')
plt.subplot(1,3,2)
plt.title('Resized Image')
plt.imshow(down, cmap='gray')
plt.subplot(1,3,3)
plt.title('Sliced Image')
plt.imshow(d1, cmap='gray')
plt.show()
```

##### Output:



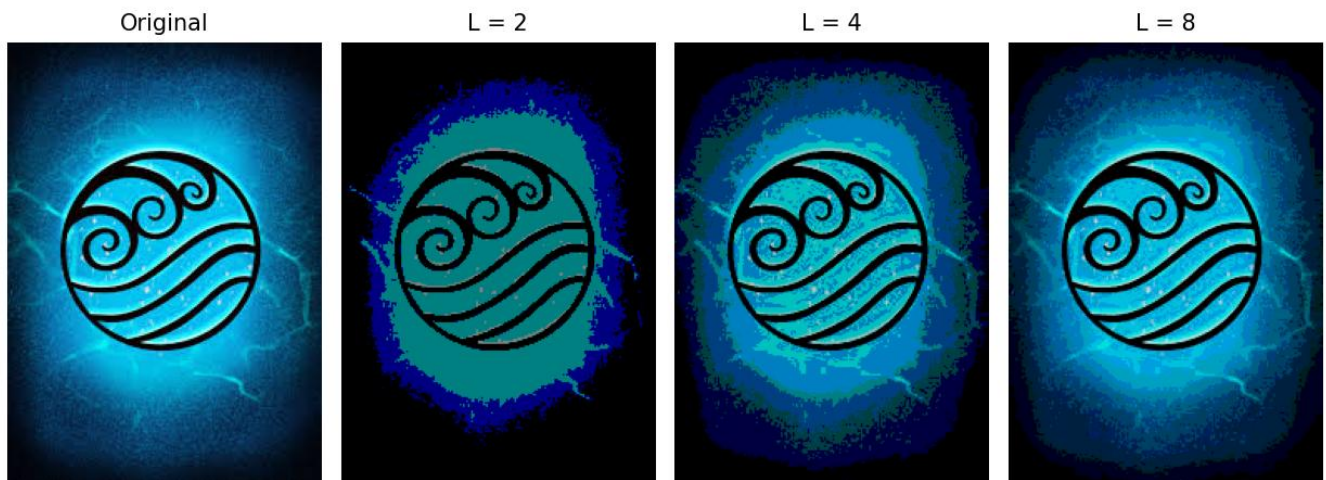
**Quant.py**

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

image = cv2.imread("image.jpg")
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
levels = [2, 4, 8]
plt.figure(figsize=(10, 4))
plt.subplot(1, 4, 1)
plt.title("Original")
plt.imshow(image)
plt.axis("off")

for i, L in enumerate(levels):
    delta = 256 // L
    quantized = (image // delta) * delta
    plt.subplot(1, 4, i + 2)
    plt.title(f'L = {L}')
    plt.imshow(quantized)
    plt.axis("off")
plt.tight_layout()
plt.show()
```

**Output:**



**5. Apply Nearest Neighbour, Bilinear and Bicubic Interpolations on an image for increasing its size.**

```
import cv2
img = cv2.imread("flower.jpg")
scale = 2

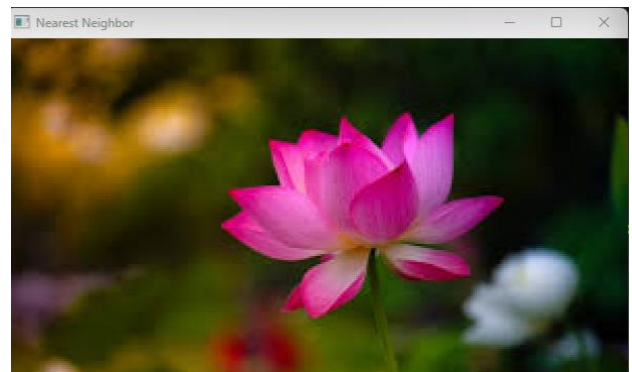
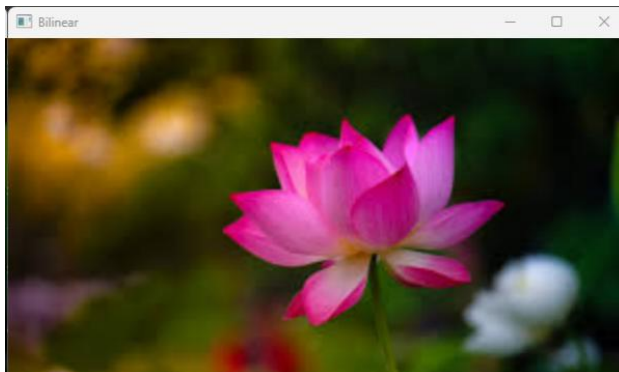
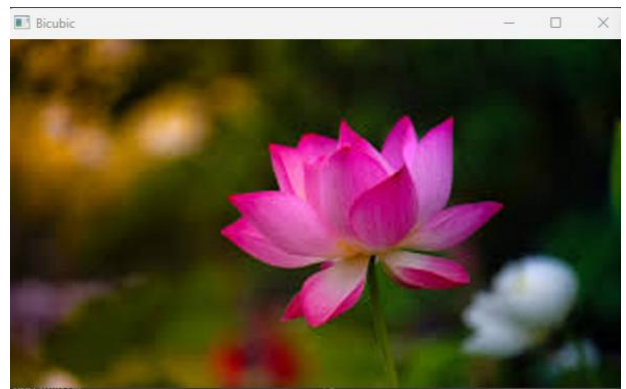
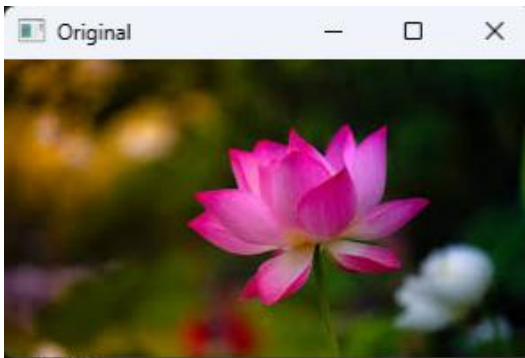
nearest = cv2.resize(img, None, fx=scale, fy=scale, interpolation=cv2.INTER_NEAREST)

bilinear = cv2.resize(img, None, fx=scale, fy=scale, interpolation=cv2.INTER_LINEAR)

bicubic = cv2.resize(img, None, fx=scale, fy=scale, interpolation=cv2.INTER_CUBIC)

cv2.imshow("Original", img)
cv2.imshow("Nearest Neighbor", nearest)
cv2.imshow("Bilinear", bilinear)
cv2.imshow("Bicubic", bicubic)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



**6. Apply different Intensity level transformations on an image.**

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

def contrast_stretching(img, low_in, high_in, low_out, high_out):
    return np.clip(((img - low_in) / (high_in - low_in)) * (high_out - low_out) + low_out, 0,
255).astype(np.uint8)

def intensity_level_stretching(img, low_in, high_in):
    return np.clip(((img - low_in) / (high_in - low_in)) * 255, 0, 255).astype(np.uint8)

def bitplane_slicing(img, plane_number):
    return ((img >> plane_number) & 1) * 255
image = cv2.imread('plantbw.jpg', cv2.IMREAD_GRAYSCALE)

contrast_stretched = contrast_stretching(image, 50, 200, 0, 255)

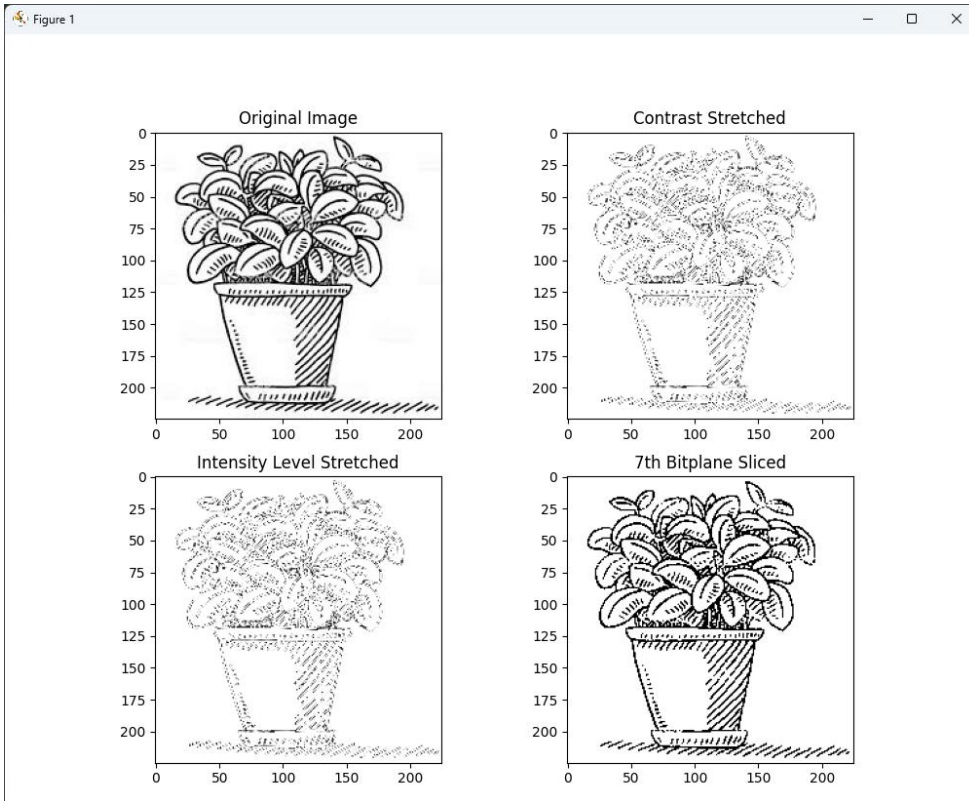
intensity_stretched = intensity_level_stretching(image, 50, 200)

bitplane = bitplane_slicing(image, 7)

plt.figure(figsize=(10, 10))
plt.subplot(2, 2, 1), plt.imshow(image, cmap='gray'), plt.title('Original Image')
plt.subplot(2, 2, 2), plt.imshow(contrast_stretched, cmap='gray'), plt.title('Contrast Stretched')
plt.subplot(2, 2, 3), plt.imshow(intensity_stretched, cmap='gray'), plt.title('Intensity Level Stretched')
plt.subplot(2, 2, 4), plt.imshow(bitplane, cmap='gray'), plt.title('7th Bitplane Sliced')
plt.show()
```

G. Narayanamma Institute of Technology and Science  
(Autonomous) (for women)  
Computer Science and Engineering Department  
Computer Vision and Pattern Recognition lab  
(2025-2026 II Semester)

**Output:**



### 7.Implement Histogram Calculation and Equalization for a given image

```
import cv2
from matplotlib import pyplot as plt

# Read grayscale image
flower = cv2.imread("dark.jpg", 0)

# Histogram equalization
flower_equalized = cv2.equalizeHist(flower)

# Calculate histograms
flower_hist = cv2.calcHist([flower], [0], None, [256], [0, 256])
flower_eq_hist = cv2.calcHist([flower_equalized], [0], None, [256], [0, 256])

# Plot results
plt.figure(figsize=(10, 4))

plt.subplot(1, 4, 1)
plt.title("Original Image")
plt.imshow(flower, cmap='gray')

plt.subplot(1, 4, 2)
plt.title("Equalized Image")
plt.imshow(flower_equalized, cmap='gray')

plt.subplot(1, 4, 3)
plt.title("Original Histogram")
plt.plot(flower_hist)

plt.subplot(1, 4, 4)
plt.title("Equalized Histogram")
plt.plot(flower_eq_hist)

plt.show()
```

G. Narayanamma Institute of Technology and Science  
(Autonomous) (for women)  
Computer Science and Engineering Department  
Computer Vision and Pattern Recognition lab  
(2025-2026 II Semester)

