International Islamic University Chittagong.

Department of Computer Science and Engineering.

Course Code: CSE-4876

Course Title: Pattern Recognition & Image Processing

Submitted By:

Name: Farida Nusrat

ID: C201242

Semester: 8th

Section: 8AF

Submitted To:

Mohammad Mahadi Hassan

Associate Professor, CSE.

Lab No: 02

Lab Title:

1. Convert RGB image to Grayscale image
2. Convert Grayscale to Binary image
3. Reduce the levels of a grayscale image from 256 to 64, 32 and 16.

Objective:

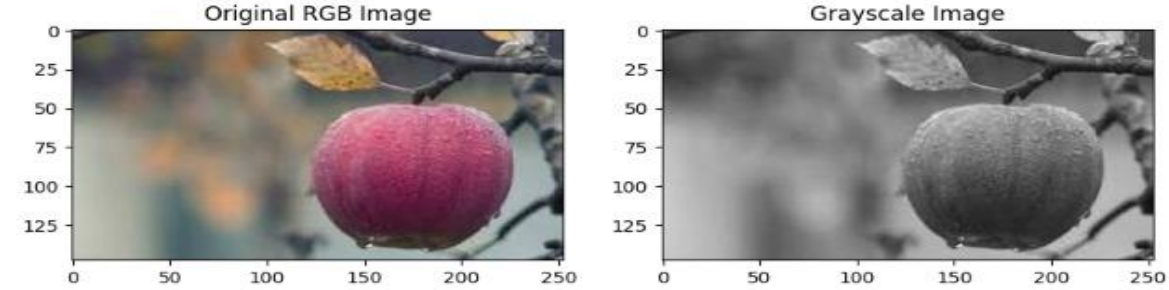
The objective of this lab experiment is to perform basic image processing operations using Python and OpenCV library. Specifically, we aim to:

1. Convert RGB image to Grayscale image
2. Convert Grayscale to Binary image
3. Reduce the levels of a grayscale image from 256 to 64, 32 and 16.

Methodology:

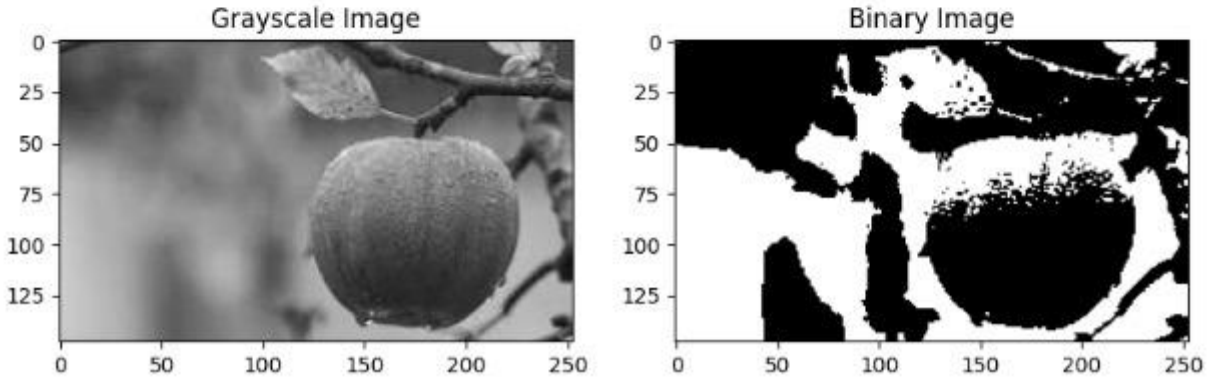
1. Convert RGB image to Grayscale image

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

# Load the RGB image
image_path = "/content/image.jpg"  # Replace with the path to your RGB image
rgb_image = Image.open(image_path)

# Convert the RGB image to a NumPy array
rgb_array = np.array(rgb_image)

# Apply the conversion formula to obtain the grayscale image
gray_array = 0.299 * rgb_array[:, :, 0] + 0.587 * rgb_array[:, :, 1] + 0.114 * rgb_array[:, :, 2]

# Display the original and grayscale images side by side
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.title("Original RGB Image")
plt.imshow(rgb_array)
plt.axis("on")

plt.subplot(1, 2, 2)
plt.title("Grayscale Image")
plt.imshow(gray_array, cmap="gray")
plt.axis("on")

plt.show()
```



2. Convert Grayscale to Binary image

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

# Load the RGB image
image_path = "/content/image.jpg"  # Replace with the path to your RGB image
rgb_image = Image.open(image_path)

# Apply binary thresholding
threshold_value = 128
binary_array = np.where(gray_array > threshold_value, 255, 0)

# Display the original, grayscale, and binary images side by side
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 2)
plt.title("Grayscale Image")
plt.imshow(gray_array, cmap="gray")
plt.axis("on")

plt.subplot(1, 3, 3)
plt.title("Binary Image")
plt.imshow(binary_array, cmap="gray")
plt.axis("on")

plt.show()
```



3. Reduce the levels of a grayscale image from 256 to 64, 32 and 16.

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

# Load the grayscale image
image_path = "/content/image.jpg"  # Replace with the path to your grayscale image
gray_image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

# Function to quantize the image to the specified number of levels
def quantize_image(image, levels):
    # Normalize the image to the range [0, 1]
    normalized_image = image / 255.0

    # Quantize the normalized image to the specified levels
    quantized_image = (normalized_image * (levels - 1)).astype(np.uint8)

    # Scale the quantized image back to the range [0, 255]
    quantized_image = (quantized_image * (255.0 / (levels - 1))).astype(np.uint8)

    return quantized_image

# Specify the desired number of levels (64, 32, and 16)
levels_list = [64, 32, 16]

# Display the original and quantized images for each level
plt.figure(figsize=(15, 5))

# Original image
plt.subplot(1, 4, 1)
plt.title("Original Grayscale Image")
plt.imshow(gray_image, cmap="gray")
plt.axis("on")

# Quantize and display images for each level
for i, levels in enumerate(levels_list):
    quantized_image = quantize_image(gray_image, levels)

    plt.subplot(1, 4, i + 2)
    plt.title(f"Quantized to {levels} Levels")
    plt.imshow(quantized_image, cmap="gray")
    plt.axis("on")

plt.show()
```