Digital Image Processing
Spring 2025
Lab:1 Reading and Displaying Images in Python

Name	
Student ID	
Date	

Objectives

- Read and display color, grayscale, and binary images using Python libraries.
- Convert between different image types.
- Understand the differences between color, grayscale, and binary images.

Prerequisites

Make sure you have the following Python libraries installed:

- `opencv-python` (OpenCV)

You can install them using following command in command prompt or bash:

pip install opencv-python

Digital Image Processing
Spring 2025
Lab:1 Reading and Displaying Images in Python

Part 1: Reading and Displaying Images

Step 1: Reading a Color Image

A color image contains three channels (Red, Green, Blue) for each pixel. To read and display a color image, use the OpenCV library.

```
# importing opencv library for image processing and computer vision related
modules
import cv2

# Read a color image using imread module of opencv library
color_image = cv2.imread('fruit basket.jpg')

# ModuLe to display the image> First argument reflect the window name whereas
# second reflect the matrix that stores the pixel values
cv2.imshow('color image',color_image)

# Module to hold the displaying window. Argument is time in milliseconds.
# For infinite time, use -1 or 0 as the argument
# The module call is terminated when a key is pressed. That's what waitKey
does
cv2.waitKey(0)
```

Task 1:

1. Download any image from the internet and read it using the code above.
2. Modify the code to display the dimensions (height, width, channels) of the image using `color_image.shape`.

Step 2: Reading and Converting to Grayscale

A grayscale image contains only intensity information, without color. It uses a single channel for pixel values.

For reading image as grayscale, you may call the following module

grayscale_image = cv2.imread('path_to_image.jpg', cv2.IMREAD_GRAYSCALE)

For example, to convert the fruit basket image to grayscale, following code maybe used

```
# Read and convert the image to grayscale
grayscale_image = cv2.imread('fruit basket.jpg', cv2.IMREAD_GRAYSCALE)

cv2.imshow('grayscale image',grayscale_image)
```

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

Alternatively, you may first read the image as color image and then call cvtColor module to convert from color to grayscale

```
import cv2

# Read a color image using imread module of opencv library
color_image = cv2.imread('fruit basket.jpg')

# Converting color image to grayscale using cvtColor
grayscale_image = cv2.cvtColor(color_image, cv2.COLOR_BGR2GRAY)

# ModuLe to display the image> First argument reflect the window name whereas
# second reflect the matrix that stores the pixel values
cv2.imshow('grayscale image',grayscale_image)

# Module to hold the displaying window. Argument is time in milliseconds.
# For infinite time, use -1 or 0 as the argument
# The module call is terminated when a key is pressed. That's what waitKey
does
cv2.waitKey(0)
```

cv2.imread reads image as BGR format, therefore, while calling cv2.cvtColor, we are passing the flag of cv2.COLOR_BGR2GRAY to convert from BGR to grayscale. If you are confused about BGR, don't worry, it will be cleared to you very soon.

Task 2:

1. Convert the same image you used in Task 1 to grayscale. Use both techniques of converting with and without cvtColor module.
2. Modify the code to save the grayscale image to your system using `cv2.imwrite()`.

Digital Image Processing
Spring 2025
Lab:1 Reading and Displaying Images in Python

Step 3: Thresholding to Create a Binary Image

A binary image contains only two values (0 or 255) for each pixel, representing black and white. We can convert a grayscale image to binary using a threshold.

```
# importing opencv library for image processing and computer vision related
modules
import cv2

# Read a color image using imread module of opencv library
color_image = cv2.imread('fruit basket.jpg')

# Converting color image to grayscale using cvtColor
grayscale_image = cv2.cvtColor(color_image, cv2.COLOR_BGR2GRAY)

# Apply a threshold to convert grayscale image to binary
_, binary_image = cv2.threshold(grayscale_image, 128, 255, cv2.THRESH_BINARY)

# ModuLe to display the image> First argument reflect the window name whereas
# second reflect the matrix that stores the pixel values
cv2.imshow('binary image',binary_image)

# Module to hold the displaying window. Argument is time in milliseconds.
# For infinite time, use -1 or 0 as the argument
# The module call is terminated when a key is pressed. That's what waitKey
does
cv2.waitKey(0)
```

Task 3:

1. Experiment with different threshold values (e.g., 100, 150, etc.) and observe the changes in the binary image.
2. Save the binary images and compare them.
3. Use landscape image to perform the same experiments.
4. Apply the same experiments on two other random images downloaded from the internet.

Digital Image Processing
Spring 2025
Lab:1 Reading and Displaying Images in Python

Submission

Submit the following:

1. Screenshots of the color, grayscale, and binary images you generated.
2. The code used for each step.
3. A brief report describing what you learned in this lab.

Conclusion

In this lab, you learned how to:

- Read and display color, grayscale, and binary images.
- Convert between different image types using Python and OpenCV.