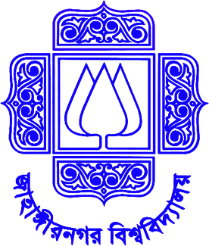
Institute of Information Technology

Jahangirnagar University

Savar, Dhaka-1342



Lab Manual

Course Code: ICT-4202

Course Title: Digital Image Processing Lab

**Lab No.: 3**

**Lab Title: IMAGE PROCESSING USING PILLOW LIBRARY OF PYTHON**

Prepared by

**Mehrin Anannya**

Assistant Professor

Institute of Information Technology

Jahangirnagar University

**Lab Title: IMAGE PROCESSING USING PILLOW LIBRARY OF PYTHON OBJECTIVE**: To introduce students with Image processing library named pillow and some works on color models.

**Lab Contents:**

* Introduction to PILLOW library
* Image Works using Pillow library
* Reading multiple images using pillow library
* Convert OpenCV image to PIL image in Python
* Additive and subtractive color
* Color spaces in RGB

**Theory: with Hands on Practice:**

**Introduction to PILLOW:**

Python Imaging Library (expansion of PIL) is the de facto image processing package for Python language. It incorporates lightweight image processing tools that aid in editing, creating and saving images. Support for Python Imaging Library got discontinued in 2011, but a project named pillow forked the original PIL project and added Python3.x support to it. Pillow was announced as a replacement for PIL for future usage. Pillow supports numerous image file formats including BMP, PNG, JPEG, and TIFF. The library encourages adding support for newer formats in the library by creating new file decoders.

The basic difference between OpenCV image and PIL image is OpenCV follows BGR color convention and PIL follows RGB color convention and the method of converting will be based on this difference.

PIL install in anaconda:

In Anaconda prompt, write – pip install pillow

**Image Works using Pillow library**

Open Image using open():

The PIL.Image.Image class represents the image object. This class provides the open() method that is used to open the image.

from PIL import Image

img = Image.open("Flower.jpeg")

#Note: Location of image should be relative only if the image is in the same directory as the Python program, otherwise absolute (full) path of the image should be provided.

Displaying the image using show():

This method is used to display the image. For displaying the image Pillow first converts the image to a .png format (on Windows OS) and stores it in a temporary buffer and then displays it. Therefore, due to the conversion of the image format to .png some properties of the original image file format might be lost (like animation). Therefore, it is advised to use this method only for test purposes.

img.show();

Obtaining information about the opened image:

A) *Getting the mode (color mode) of the image:* The mode attribute of the image tells the type and depth of the pixel in the image. A 1-bit pixel has a range of 0-1, and an 8-bit pixel has a range of 0-255. There are different modes provided by this module. Few of them are:

Mode Description

1 1-bit pixels, black and white

L 8-bit pixels, Greyscale

P 8-bit pixels, mapped to any other mode using a color palette

RGB 3×8-bit pixels, true color

RGBA 4×8-bit pixels, true color with transparency mask

print(img.mode)

B) *Getting the size of the image:* This attribute provides the size of the image. It returns a tuple that contains width and height.

print(img.size)

C) *Getting the format of the image:* This method returns the format of the image file.

print(img.format)

Rotating an image using rotate(): After rotating the image, the sections of the image having no pixel values are filled with black (for non-alpha images) and with completely transparent pixels (for images supporting transparency)

r\_img = img.rotate(40)

r\_img.show()

Flipping the Image

Image.transpose() is used to transpose the image (flip or rotate in 90 degree steps).

Keywords FLIP\_TOP\_BOTTOM and FLIP\_LEFT\_RIGHT will be passed to transpose method to flip it.

FLIP\_TOP\_BOTTOM – returns an original image flipped Vertically

FLIP\_LEFT\_RIGHT – returns an original image flipped Horizontally

# importing PIL Module

from PIL import Image

# open the original image

original\_img = Image.open("geek.jpg")

# Flip the original image vertically

vertical\_img = original\_img.transpose(method=Image.FLIP\_TOP\_BOTTOM)

vertical\_img.save("vertical.png")

vertical\_img.show()

# close all our files object

original\_img.close()

vertical\_img.close()

Resizing an image using resize(): Interpolation happens during the resize process, due to which the quality of image changes whether it is being upscaled (resized to a higher dimension than

original) or downscaled (resized to a lower Image then original). Therefore resize() should be used cautiously and while providing suitable value for resampling argument.

size = (100, 300)

r\_img = img.resize(size)

r\_img.show()

Saving an image using save(): While using the save() method Destination\_path must have the image filename and extension as well. The extension could be omitted in Destination\_path if the extension is specified in the format argument.

from PIL import Image

size = (40, 40)

img = Image.open(r"geek.jpg")

print("Original size of the image")

print(img.size)

# resizing the image

r\_img = img.resize(size, resample = Image.BILINEAR)

# resized\_test.png => Destination\_path

r\_img.save("resized\_test.jpg")

# Opening the new image

img = Image.open(r"resized\_test.jpg")

print("\nNew size of the image")

print(img.size)

## Cropping the Image

Cropping is the process of selecting only a part of the image. The crop() method is used to crop a rectangular portion of any image.

Syntax:

PIL.Image.crop(box = None)

Parameters:

box: a 4-tuple defining the left, upper, right, and lower pixel coordinate.

# Importing Image class from PIL module

from PIL import Image

# Opens a image in RGB mode

im = Image.open(r"geek.jpg")

# Size of the image in pixels

# (size of original image)

# (This is not mandatory)

width, height = im.size

# Setting the points for cropped image

left = 5

top = height / 4

right = 164

bottom = 3 \* height / 4

# Cropped image of above dimension # (It will not change original image)

im1 = im.crop((left, top, right, bottom))

# Shows the image in image viewer

im1.show()

**Reading multiple images using pillow library**

#Read all images from the folder

import os

from PIL import Image

from matplotlib import pyplot as plt

root = "Images"

fnames = os.listdir(root)

len(fnames)

fig, axs=plt.subplots(nrows=1, ncols=6,figsize=(20,20))

axs=axs.flatten()

for i in range(6):

filepath=os.path.join(root, fnames[i])

img=Image.open(filepath)

axs[i].imshow(img)

axs[i].axis('off')

axs[i].set\_title(fnames[i])

plt.show()

# Convert OpenCV image to PIL image in Python

**import** cv2

**from** PIL **import** Image

# Open image using openCV2

opencv\_image **=** cv2.imread("logo.png")

# Notice the COLOR\_BGR2RGB which means that the color is

# converted from BGR to RGB

color\_coverted **=** cv2.cvtColor(opencv\_image, cv2.COLOR\_BGR2RGB)

# Displaying the Scanned Image by using cv2.imshow() method

cv2.imshow("OpenCV Image", opencv\_image)

# Displaying the converted image

pil\_image **=** Image.fromarray(color\_coverted)

pil\_image.show()

# waits for user to press any key

# (this is necessary to avoid Python kernel form crashing)

cv2.waitKey(0)

# closing all open windows

cv2.destroyAllWindows()

Additive and subtractive color:

<https://pavilion.dinfos.edu/Article/Article/2355687/additive-subtractive-color-models/#:~:text=In%20the%20subtractive%20color%20model,black%2C%20also%20known%20as%20CMYK.>

Color Spaces in RGB:

Please follow the given link: <https://www.geeksforgeeks.org/color-spaces-in-opencv-python/?ref=lbp>

import cv2

image = cv2.imread('Images\Flower.jpeg')

B, G, R = cv2.split(image)

# Corresponding channels are separated

cv2.imshow("original", image)

cv2.waitKey(0)

cv2.imshow("blue", B)

cv2.waitKey(0)

cv2.imshow("Green", G)

cv2.waitKey(0)

cv2.imshow("red", R)

cv2.waitKey(0)

cv2.destroyAllWindows()

Tasks: