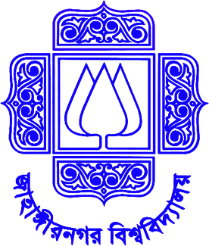
Institute of Information Technology

Jahangirnagar University

Savar, Dhaka-1342



Lab Manual

Course Code: ICT-4202

Course Title: Digital Image Processing Lab

**Lab No.: 4**

**Lab Title: Basic Image Processing Using Glob and Scikit-Image Library**

Prepared by

**Mehrin Anannya**

Assistant Professor

Institute of Information Technology

Jahangirnagar University

**Lab Title: Basic Image Processing Using Glob Library**: To introduce students with Image processing library named Glob.

**Lab Contents:**

* Introduction to Glob library
* Image Works using Glob library
* Reading multiple images using glob library

**Theory: with Hands on Practice:**

**Introduction to Glob:**

Glob is a Python module that provides a convenient way to search for files that match a specified pattern. It allows you to use wildcard characters to match files with similar names or extensions.

Glob supports the use of wildcard characters to match files with similar names or extensions. Here are the most commonly used wildcard characters:

* \*: Matches any string of characters, including an empty string.
* ?: Matches any single character.
* [ ]: Matches any character inside the brackets.
* [! ]: Matches any character not inside the brackets.

For example, let’s say you want to get a list of all files in the my\_folder directory that have a .txt extension. You can use the \* wildcard character to match any string of characters before the .txt extension:

import glob

files = glob.glob('my\_folder/\*.txt')

print(files)

**OUTPUT:**

**['my\_folder/file1.txt', 'my\_folder/file2.txt']**

**Image Works using Glob library**

**Returns a list of file names using glob():**

This class provides the glob() method that is used to return a list of files..

#import the library opencv

import cv2

import glob

file\_list = glob.glob('images/test\_images/\*.\*') #Returns a list of file names

print(file\_list) #Prints the list containing file names

#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.

Loading each file at a time:

my\_list=[] #Empty list to store images from the folder.

path = "images/test\_images/\*.\*"

for file in glob.glob(path): #Iterate through each file in the list using for

print(file) #just stop here to see all file names printed

a= cv2.imread(file) #now, we can read each file since we have the full path

my\_list.append(a) #Create a list of images (not just file names but full images)

#View images from the stored list

from matplotlib import pyplot as plt

plt.imshow(my\_list[2]) #View the 3rd image in the list.

**Load images and perform some action:**

Let's break down the code and explain each method and function:

1. Importing Libraries:

- `cv2`: This is the OpenCV library, a computer vision and image processing library.

- `glob`: It is used to find all the path names matching a specified pattern according to the rules used by the Unix shell.

2. Setting Path and Image Number:

path = "images/test\_images/\*.\*"

img\_number = 1

- `path`: The path variable is set to a directory containing image files. The `\*.\*` pattern is used to match all files in the specified directory.

- `img\_number`: An iterator is initialized to keep track of image numbers.

3. Looping Through Files:

for file in glob.glob(path):

print(file)

- This loop iterates through all the files in the specified path and prints their names.

4. Reading Images:

a = cv2.imread(file)

- `cv2.imread(file)`: Reads the image file specified by the variable `file` using OpenCV.

6. Writing Processed Images:

cv2.imwrite("images/test\_images/Color\_image"+str(img\_number)+".jpg", a)

- `cv2.imwrite()`: Writes the processed image `c` to a new file in JPEG format. The filename is composed of "Color\_image" followed by the image number.

This script reads images from a specified directory, converts their color space from BGR to RGB, saves the processed images, and displays each processed image for 1 second. The image filenames are based on the original image number.

#import the opencv library so we can use it to read and process images

import cv2

import glob

#select the path

path = "images/test\_images/\*.\*"

img\_number = 1 #Start an iterator for image number.

#This number can be later added to output image file names.

for file in glob.glob(path):

print(file) #just stop here to see all file names printed

a= cv2.imread(file) #now, we can read each file since we have the full path

#print(a) #print numpy arrays for each file

#let us look at each file

# cv2.imshow('Original Image', a)

# cv2.waitKey(0)

# cv2.destroyAllWindows()

#process each image - change color from BGR to RGB.

c = cv2.cvtColor(a, cv2.COLOR\_BGR2RGB) #Change color space from BGR to RGB

cv2.imwrite("images/test\_images/Color\_image"+str(img\_number)+".jpg", c)

img\_number +=1

cv2.imshow('Color image', c)

cv2.waitKey(1000) #Display each image for 1 second

cv2.destroyAllWindows()

# Using os.listdir to read multiple files:

**os.listdir():**

returns a list containing the names of the entries in the directory given by path.

import os

path = 'images/test\_images/'

print(os.listdir(path)) #Very similar to glob, prints a list of all files in the directory

for image in os.listdir(path): #iterate through each file to perform some action

print(image)

**os.walk():**

returns a generator, that creates a tuple of values

(current\_path, directories in current\_path, files in current\_path).

Every time the generator is called it will follow each directory recursively until no further sub-directories are available from the initial directory that walk was called upon.

**os.path.join():**

os.path.join() method in Python join one or more path components intelligently.

import os

print(os.walk(".")) #Nothing to see here as this is just a generator object

# traverse root directory, and list directories as dirs and files as files

for root, dirs, files in os.walk("."):

#print(root) #Prints root directory names

path = root.split(os.sep) #SPlit at separator (/ or \)

#print(path) #Gives names of directories for easy location of files

#print(files) #Prints all file names in all directories

#Let us now visualize directories and files within them

print((len(path) - 1) \* '---', os.path.basename(root)) #Add --- based on the path

for file in files:

print(len(path) \* '---', file)

Another way to look at all dirs. and files:

import os

for root, dirs, files in os.walk("."):

#for path,subdir,files in os.walk("."):

for name in dirs:

print (os.path.join(root, name)) # will print path of directories

for name in files:

print (os.path.join(root, name)) # will print path of files

Tasks:

1. Using the for loop and glob library read all the image files and rename all of them according to loop.