California State University, Long Beach
Department of Computer Engineering and Computer Science
CECS 553 Sec 02 11792 (Machine Vision) – Fall 2022
Assignment 02 - Tuesday, 09/06/2022

1 OpenCV Library

Estimated time needed: **60** minutes

Objectives

Image processing and computer vision tasks include displaying, cropping, flipping, rotating, image segmentation, classification, image restoration, image recognition, image generation. Also, working with images via the cloud requires storing and transmitting, and gathering images through the internet. Python is an excellent choice as it has many image processing tools, computer vision, and artificial intelligence libraries. Finally, it has many libraries for working with files in the cloud and the internet. A digital image is simply a file on your computer. In this lab, you will gain an understanding of these files and learn to work with these files with some popular libraries

Open CV

Image Files and Paths

Load in Image in Python

Plotting an Image

Gray Scale Images, Quantization and Color Channels

Gray Scale Images, Quantization and Color Channels

Download the image for the lab:

[ ]:
```
!wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
 ↪IBMDeveloperSkillsNetwork-CV0101EN-SkillsNetwork/images%20/images_part_1/
 ↪lenna.png -O lenna.png
!wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
 ↪IBMDeveloperSkillsNetwork-CV0101EN-SkillsNetwork/images%20/images_part_1/
 ↪baboon.png -O baboon.png
!wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
 ↪IBMDeveloperSkillsNetwork-CV0101EN-SkillsNetwork/images%20/images_part_1/
 ↪barbara.png -O barbara.png
```

First, let's define a helper function to concatenate two images side-by-side. You will need to understand this code this moment, but this function will be used repeatedly in this tutorial to showcase the results.

[ ]:
```
def get_concat_h(im1, im2):
    #https://note.nkmk.me/en/python-pillow-concat-images/
    dst = Image.new('RGB', (im1.width + im2.width, im1.height))
    dst.paste(im1, (0, 0))
    dst.paste(im2, (im1.width, 0))
    return dst
```

1.1 Image Files and Paths

An image is stored as a file on your computer. Below, we define `my_image` as the filename of a file in this directory.

[ ]:
```
my_image = "lenna.png"
```

Filename consists of two parts, the name of the file and the extension, separated by a full stop (`.`). The extension specifies the format of the image. There are two popular image formats – Joint Photographic Expert Group image (or `.jpg`, `.jpeg`) and Portable Network Graphics (or `.png`). These file types make it simpler to work with images. For example, it compresses the image using sine/cosine approximations, taking less spaces on your drive to store the image.

Image files are stored in the file system of your computer. The location of it is specified using a "path", which is often unique. You can find the path of your current working directory with Python's `os` module. The `os` module provides functions to interact with the file system, e.g. creating or removing a directory (folder), listing its contents, changing and identifying the current working directory.

[ ]:
```
import os
cwd = os.getcwd()
cwd
```

The "path" to an image can be found using the following line of code.

[ ]:
```
image_path = os.path.join(cwd, my_image)
image_path
```

1.2 Load in Image in Python

OpenCV is a library used for computer vision. It has more functionality than the `PIL` library but is more difficult to use. We can import `OpenCV` as follows:

[ ]:
```
import cv2
```

The imread() method loads an image from the specified file, the input is the path of the image to be read (just like PIL), the flag paramter specifies how the image should be read, and the default value is cv2.IMREAD_COLOR.

[ ]:
```
image = cv2.imread(my_image)
```

The result is a numpy array with intensity values as 8-bit unsigned integers.

[ ]:
```
type(image)
```

We can get the shape of the array from the `shape` attribute.

[ ]:
```
image.shape
```

The shape is the same as the PIL array, but there are several differences; for example, PIL returns in (R, G, B) format whereas OpenCV returns in (B, G, R) format.

Each pixel could take on 256 possible values as intensity, ranging from 0 to 255, with 0 being the lowest intensity and 255 being the highest. The maximum and minimum intensity values of an image can be obtained, respectively, by calling:

[ ]:
```
image.max()
```

and

[ ]:
```
image.min()
```

1.3 Plotting an Image

You can use OpenCV's `imshow` function to open the image in a new window, but this may give you some issues in Jupyter:

[ ]:
```
#cv2.imshow('image', imgage)
#cv2.waitKey(0)
#cv2.destroyAllWindows()
```

You can also use the `imshow` function from the `matplotlib` library:

[ ]:
```
import matplotlib.pyplot as plt
```

[ ]:
```
plt.figure(figsize=(10,10))
plt.imshow(image)
plt.show()
```

The image output doesn't look natural. This is because the order of RGB Channels are different. We can change the color space with conversion code and the function `cvtColor` from the `cv2` library:

[ ]:
```
new_image=cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

[ ]:
```
plt.figure(figsize=(10,10))
plt.imshow(new_image)
plt.show()
```

You can also load the image using its path, this comes in handy if the image is not in your working directory:

[ ]:
```
image = cv2.imread(image_path)
image.shape
```

You can save the image as in `jpg` format.

[ ]:
```
cv2.imwrite("lenna.jpg", image)
```

1.3.1 Grayscale Images

Grayscale images have pixel values representing the amount of light or intensity. Light shades of gray have a high-intensity darker shades have a lower intensity. White has the highest intensity, and black the lowest. We can convert an image to Gray Scale using a color conversion code and the function cvtColor.

The code for RGB to gray is cv2.COLOR_BGR2GRAY, we apply the function:

[ ]:
```
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

The image array has only two dimensions, i.e. only one color channel:

[ ]:
```
image_gray.shape
```

We can plot the image using `imshow` but we have to specify the color map is gray:

[ ]:
```
plt.figure(figsize=(10, 10))
plt.imshow(image_gray, cmap='gray')
plt.show()
```

We can save the image as a grayscale image, let's save it as a `jpg` as well, in the working directory.

[ ]:
```
cv2.imwrite('lena_gray_cv.jpg', image_gray)
```

You can also load in a grayscale image we have to set flag parameter to gray color conversation code: cv2.COLOR_BGR2GRAY:

[ ]:
```
im_gray = cv2.imread('barbara.png', cv2.IMREAD_GRAYSCALE)
```

We can plot the image:

[ ]:
```
plt.figure(figsize=(10,10))
plt.imshow(im_gray,cmap='gray')
plt.show()
```

1.3.2 Color Channels

We can also work with the different color channels. Consider the following image:

```
[ ]: baboon=cv2.imread('baboon.png')
     plt.figure(figsize=(10,10))
     plt.imshow(cv2.cvtColor(baboon, cv2.COLOR_BGR2RGB))
     plt.show()
```

We can obtain the different RGB colors and assign them to the variables blue, green, and red, in (B, G, R) format.

```
[ ]: blue, green, red = baboon[:, :, 0], baboon[:, :, 1], baboon[:, :, 2]
```

We can concatenate each image channel the images using the function vconcat.

```
[ ]: im_bgr = cv2.vconcat([blue, green, red])
```

Plotting the color image next to the red channel in grayscale, we see that regions with red have higher intensity values.

```
[ ]: plt.figure(figsize=(10,10))
     plt.subplot(121)
     plt.imshow(cv2.cvtColor(baboon, cv2.COLOR_BGR2RGB))
     plt.title("RGB image")
     plt.subplot(122)
     plt.imshow(im_bgr,cmap='gray')
     plt.title("Different color channels  blue (top), green (middle), red (bottom) ␣
       ↪")
     plt.show()
```

1.3.3 Indexing

We can use numpy slicing. For example, we can return the first 256 rows corresponding to the top half of the image:

```
[ ]: rows = 256
```

```
[ ]: plt.figure(figsize=(10,10))
     plt.imshow(new_image[0:rows,:,:])
     plt.show()
```

We can also return the first 256 columns corresponding to the first half of the image:

```
[ ]: columns = 256
```

```
[ ]: plt.figure(figsize=(10,10))
     plt.imshow(new_image[:,0:columns,:])
     plt.show()
```

If you want to reassign an array to another variable, you should use the `copy` method (we will cover this in the next section).

[ ]:
```
A = new_image.copy()
plt.imshow(A)
plt.show()
```

If we do not apply the method `copy()`, the variable will point to the same location in memory. Consider the variable `B` below, if we set all values of array `A` to zero, since `A` and `B` points to the same object in the memory, `B` will also have all-zero elements:

[ ]:
```
B = A
A[:,:,:] = 0
plt.imshow(B)
plt.show()
```

We can also manipulate elements using indexing. In the following piece of code, we create a new array `baboon_red` and set all but the red color channels to zero. Therefore, when we display the image, it appears red:

[ ]:
```
baboon_red = baboon.copy()
baboon_red[:, :, 0] = 0
baboon_red[:, :, 1] = 0
plt.figure(figsize=(10, 10))
plt.imshow(cv2.cvtColor(baboon_red, cv2.COLOR_BGR2RGB))
plt.show()
```

We can do the same for blue:

[ ]:
```
baboon_blue = baboon.copy()
baboon_blue[:, :, 1] = 0
baboon_blue[:, :, 2] = 0
plt.figure(figsize=(10, 10))
plt.imshow(cv2.cvtColor(baboon_blue, cv2.COLOR_BGR2RGB))
plt.show()
```

We can do the same for green:

[ ]:
```
baboon_green = baboon.copy()
baboon_green[:, :, 0] = 0
baboon_green[:, :, 2] = 0
plt.figure(figsize=(10,10))
plt.imshow(cv2.cvtColor(baboon_green, cv2.COLOR_BGR2RGB))
plt.show()
```

[ ]:
```
image=cv2.imread('baboon.png')
```

[ ]:
```
plt.figure(figsize=(10,10))
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.show()
```

[ ]:
```
image=cv2.imread('baboon.png') # replace and add you image here name
baboon_blue=image.copy()
baboon_blue[:,:,1] = 0
baboon_blue[:,:,2] = 0
plt.figure(figsize=(10,10))
plt.imshow(cv2.cvtColor(baboon_blue, cv2.COLOR_BGR2RGB))
plt.show()
```

1.3.4 Question 1:

Use the image `baboon.png` from this lab or take any image you like.

Open the image and create a OpenCV Image object called `baboon_blue`, convert the image from BGR format to RGB format, get the blue channel out of it, and finally plot the image