ITCS159 Software Lab for Basic Scientific Problem Solving
Lab 14: the use of Python Library (Scikit-image: Introduction to Image Processing)

November 15, 2021

Introduction to Image Processing

Scikit-image is a Python package designed to perform image processing. It uses NumPy arrays as image objects. This lab describes how to use the Scikit-image library on various image processing tasks along with other scientific Python modules such as NumPy and SciPy.

The basic image processing topic covered in this lab are as follows:

- RGB Color model
- Image Manipulation
- Image transformation and filtering

RGB Color model

Basically, a digital image is composed of a finite number of elements. Each element have a value at a particular location which is commonly called *pixel.* The pixel value depends on types of image as follows:

- **Binary Image**: is an image that contain only two values that are 0 and 1, where 0 refers to Black and 1 refers to White. This type of image is called Monochrome.
- **8-bit color Image**: is a popular image format which contains 256 different shades of colors. In this format, 0 refers to Black, 255 refers to White, and 127 refers to Gray. Normally, pictures taken from digital camera and be saved as JPEG format are store in a standard of "8-bit" image.
- **16 bit color Image**: is a high color format contain 65,536 different colors.

In this lab, you are going to learn how to manipulate the 8-bit image through the RGB format.

Exercise 1) Download and Read the image file named ’baboon.png’ from Mycourse.

```
import matplotlib.pyplot as plt
from skimage import io

img = io.imread(‘baboon.png’)
```

What is the output? What is the objective of the code?

Exercise 2) Next, try the following code to display data of the image:

```
print(img[0])
print(img[0,0])
print(img[0,0,0])
```

Can you explain the results? What does it mean?

Exercise 3) Try the following command to display the image:

```
plt.figure(figsize=(4, 4))
plt.imshow(img, cmap=’gray’, interpolation=’nearest’)
plt.axis(’on’)

plt.tight_layout()
plt.show()
```

At this point, can you explain the objective of the code?

The `matplotlib` and `imshow` is used to display an image inside a matplotlib.

The result from the exercise 2 shows a set of matrix containing number of pixels. The images are normally represented in 2-dimensional array and can be defined as 2d-function as shown in Figure 1.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & f(0,2) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & f(1,2) & \dots & f(1,N-1) \\ \vdots & \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & f(M-1,2) & \dots & f(M-1,N-1) \end{bmatrix}$$

Figure 1: Example of an image represented in a Matrix size MxN
source:`https://www.geeksforgeeks.org/digital-image-processing-basics/`.

Exercise 4) After understood the basic concepts of the image, let's see details of a given image using the following commands:

```
print(img.dtype)
print(img.shape)
print(img.size)
print(type(img))
```

Can you explain the meaning of the outputs?

Exercise 5) The next step is to display the following data from the image:

```
io.imshow(img[:,:,0])
io.show()

io.imshow(img[:,:,1])
io.show()

io.imshow(img[:,:,2])
io.show()
```

What are the results?

Exercise 6) Let's change the way to display the image:

```
red_image = img.copy()
red_image[:,:,1] = 0
red_image[:,:,2] = 0
io.imshow(red_image)
io.show()
```

What is the result? Why does the image become **RED**?

Exercise 7) Let's try another example.

```
green_image = img.copy()
green_image[:,:,0] = 0
green_image[:,:,2] = 0
io.imshow(green_image)
io.show()
```

What is the outputs? Why does the image become **GREEN**?

Exercise 8) Let's try to display the image in **BLUE** by yourself.

Hint! Normally, the 8-bit image is represented in RGB format which composed of some combination of the three primary colors of light Red, Green, and Blue. No matter what color you watch on the screen, it is being made up from these three combination of colors. The RGB color composed with 3 plane of 2D-array as presented in Figure 2 where, the first plane refers to intensity of red color of an image in gray scale e.g., img[1,0,0], the second plane refer to intensity of green color e.g., img[0,1,0], and the third plane refer to intensity of blue color respectively e.g., img[0,0,1].

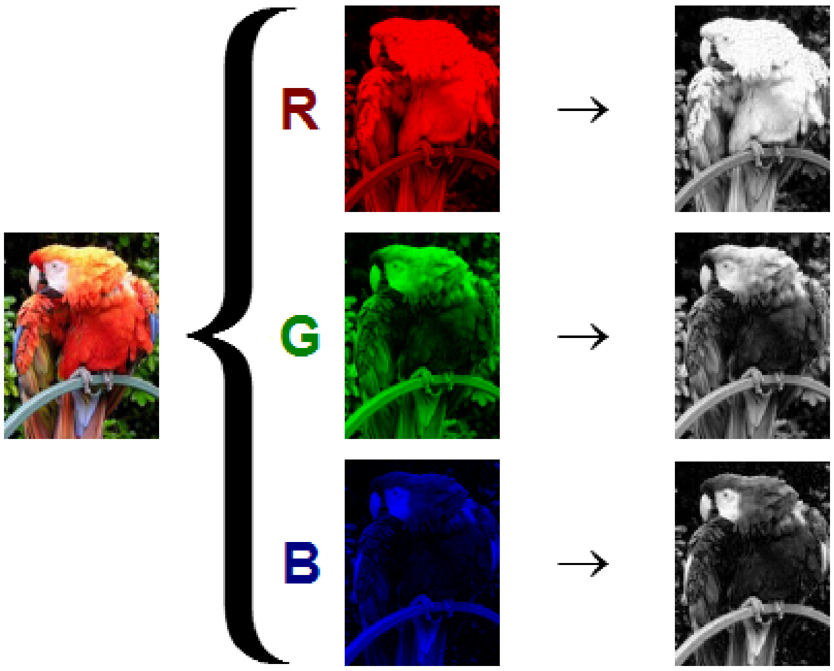

Figure 2: An example of 3-plane RGB color image
`source:http://www.espanito.com/aplicacin-de-visin-con-labview-para-la-deteccin-de-frascos-con.html?part=6`.

Milestone 1) You have now completed the Milestone 1. You should get sign off by your lab assistant.

Image Manipulation

As we mentioned earlier, the RGB image format can be represented in 2D-array with 3 planes. For the basic image manipulation such as image cropping, simple operations such as *NumPy* and *SciPy*, can be used to manipulate images. This section addresses the basic image manipulation and processing using the core scientific modules.

Exercise 9) Display the image using the following commands:

```
io.imshow(img[1:500,1:500])
io.show()

io.imshow(img[300:500,100:400])
io.show()
```

What are the outputs?

Exercise 10) Try to crop the given image and display as example below. What commands should we use to produce the image as in Figure 3?

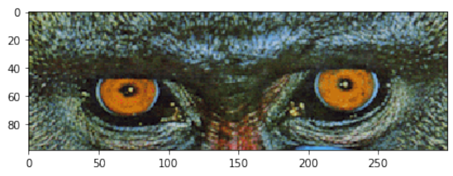

Figure 3: The result of slicing the image. source:https://www.researchgate.net/publication/280083777_Processing_Multispectral_Images_via_Mathematical_Morphology.

Exercise 11) Try the following code to see example of **masking** an image:

```
import numpy as np

test_image = img.copy()
lx, ly, lz = test_image.shape
X, Y = np.ogrid[0:lx, 0:ly]
mask = (X - lx / 2) ** 2 + (Y - ly / 2) ** 2 > lx * ly / 4;
test_image[mask]=0;
io.imshow(test_image)
io.show()
```

What is the output?

Exercise 12) If the value of pixel color is go beyond 255, what will happen? Let's try the following code to see the result.

```
camera_multiply = img*3
io.imshow(camera_multiply)
io.show()
```

What is the result of pixel overflow?

Milestone 2) You have now completed the Milestone 2. You should get sign off by your lab assistant.

Image transformation and filtering

Image transformation is a function that takes image as an input then produces an image as an output through some specific operations. Next, we are trying some of the very basic image transformation functions provided by *Scikit-image*.

Exercise 13) Try the following code to **resize** the image.

```
from skimage.transform import resize, rotate, swirl

resized = resize(img, (350, 300), mode='constant',
    anti_aliasing=True)
io.imshow(resized)
io.show()
```

What are the output? Can you resize the image to 50 x 50 and see what will happen?

Exercise 14) Next, try the following command to **rotate** the image?

```
#Rotate image by a certain angle around its center.
#Rotation angle in degrees in counter-clockwise direction.
rotate = rotate(img, 180)
io.imshow(rotate)
io.show()
```

What are the output? How the rotate function does?

Exercise 15) Now, you can try to use the function **swirl**.

```
swirled = swirl(img, rotation=0, strength=100, radius=120,
    mode='constant')
io.imshow(swirled)
io.show()
```

What are the outputs? How the function swirl work?

Image filtering is a technique for modifying or enhancing an image. there are many image processing operations implemented with filtering such as blurring, sharpening, etc.

Exercise 16) Try to blur the image by using the following command:

```
from skimage import filters

blurred_img = filters.gaussian(img, sigma=3, multichannel=
    True)
io.imshow(blurred_img)
io.show()
```

What is the result of blurring image? Can you blur more?

Exercise 17) Next, Can you try sharpening the image by using the following commands:

```
alpha = 0.15
shapened_img = img + alpha * (img - blurred_img)
io.imshow(shapened_img.astype(int))
io.show()
io.imshow(img)
io.show()
```

What is the result of sharpening image? What does the equation means?

Milestone 3) You have now completed the Milestone 3. You should get sign off by your lab assistant.