Image Processing with Python

Part 1 - Basic Image Manipulation

In this section, we are going to use Python Imaging Library (PIL) to manipulate images.

By finishing this section, you should be able to manipulate images including duplicate, resize, crop, flip, rotation, etc.

Load an Image File

Before we can retouch an image, we have to load the image data from an image file. The **Image** subset of the library Pillow can be used for opening a file.

The **Image.open()** function opens an image file and returns an image object. The **show()** function of the image object opens the default image viewer of the OS to display the image content. Follow the steps below to open an image file:

1. Launch Spyder, and change the working directory to "D:\".

2. Download the images.zip from the course web page, and extract the image files and store them in D:\images.

3. Type the following code in Spyder. And press Ctrl+ENTER to run the code cell.

```
#%%
from PIL import Image
img = Image.open('images/raining_night.jpg')
img.show()
```

Duplicating the Image Object

Assume that img is an image object. If we run the following line, both img and img2 are representing the same image object. If you make any modification on img, you will see the same result in img2.

```
img2 = img
```

If you want to have another copy of the image object. You should use another way instead – use the **copy()** function of the image object. The following is the statement to create a copy of the image object. The contents of both img and img2 are identical but they are two individual objects.

```
img2 = img.copy()
```

Resizing Image

We usually resize the image because the image size is too large or too small that is not suitable to show in the document. The **resize()** function helps us to resize the image object. It accepts a 2-tuple representing the width and height in pixels and returns a resized image. If the provided size is larger than the original one, it means **scaling up** the image. Conversely, it means **scaling down** the image. The syntax of resize() is as follows:

```
result = src_img.resize((width, height))
```

Try the following example to get an image with 1/8 size of the original image.

```
#%%
w, h = img.size
w = w // 8
h = h // 8
img3 = img.resize((w, h))
img3.show()
```

Remark: (1) img.size is the size attribute of the image in 2-tuple format.
(2) // operator is the "**floor division**" that returns an integer

Crop, Rotation, Flip, Combine

The **crop()** function is used to crop the image. It accepts a 4-tuple that defines the crop area – a rectangle including the positions: left, top, right, and bottom.

Try the following example to crop the image that the width and height are identical:

```
#%%
w, h = img.size
s = min(w, h)
l = (w - s) // 2
r = l + s
t = (h - s) // 2
b = t + s

img4 = img.crop((l, t, r, b))
img4.show()
```

The **rotation()** function accepts a float value – angle in degree. If the angle is a positive number, the resulted image is rotated counter-clockwise. Conversely, with the negative angle, the resulted image is rotated clockwise.

Try the following example to rotate the image:

```
#%%
```

```
img5 = img.rotate(-45)
img5.show()
```

The **transpose()** function can be used for flipping image and rotating image. But, for the rotation, the **transpose()** function can rotate an image 90, 180 and 270 degree only. If you want to rotate the image from other angles, you should use the **rotation()** function instead. The transpose() function accepts one of the following parameters:

- Image.FLIP_LEFT_RIGHT – flip the image horizontally
- Image.FLIP_TOP_BOTTOM – flip the image vertically
- Image.ROTATE_90 – rotate image 90 degree counter-clockwise
- Image.ROTATE_180 – rotate image 180 degree counter-clockwise
- Image.ROTATE_270 – rotate image 270 degree counter-clockwise
- Image.TRANSPOSE - same as Image.ROTATE_90

Try the following example and check the result:

```
#%%
img6 = img.transpose(Image.FLIP_LEFT_RIGHT)
img6.show()
img7 = img.transpose(Image.FLIP_TOP_BOTTOM)
img7.show()
img8 = img.transpose(Image.ROTATE_90)
img8.show()
img9 = img.transpose(Image.ROTATE_180)
img9.show()
img10 = img.transpose(Image.ROTATE_270)
img10.show()
```

File Saving

The **save()** function saves the image data under the given *filename*. If the *format* is not specified, the format to use is determined from the filename extension, if possible. The syntax is as follows:

```
img.save(filename, format)
```

For example, we save img10 as a JPEG file, so the statement will be:

```
img10.save('rotated_raining_night.jpg', format='jpeg')
```

Or,

```
img10.save('rotated_raining_night.jpg')
```

The format parameter can be omitted because the filename extension shows that it is a JPEG file. You can also save the image as other popular formats (bmp, gif, png, tiff, etc.), as specified in the following document:
https://pillow.readthedocs.io/en/3.1.x/handbook/image-file-formats.html

Image Enhancement

The **ImageEnhance** module of the PIL library provides different functions to enhance images. We will try the following classes one-by-one:

- ImageEnhance.Brightness()
- ImageEnhance.Contrast()
- ImageEnhance.Color()

```
from PIL import ImageEnhance
```

Changing Brightness

To adjust the brightness of an image, a Brightness object associated with the image needs to be created first. Then, we call the **enhance()** method of the object with a factor to adjust the brightness.

The **enhance()** method accepts a factor in float format. It returns the original image if the factor equals 1.0. If the factor equals 0.0, it returns a completely black image. If you want to make the image brighter, the factor should be larger than 1.0. Otherwise, the factor should be less than 1.0 and larger than 0.0.

1. Add the following lines to the end of the existing code.

```
#%%
brightnessObj = ImageEnhance.Brightness(img)
img11 = brightnessObj.enhance(2.0)
img11.show()
```

You may simplify the above code as the follows:

```
#%%
img12 = ImageEnhance.Brightness(img).enhance(2.0)
img12.show()
```

2. Run the code cell and compare the results of the brightness adjusted image and the original image.

3. Change the factor to 0.5, run it again and check the result.

Changing Contrast

To adjust the contrast of an image (the obvious difference between colors), a contrast object associated with the image needs to be created and we then call the **enhance()** method. The usage is similar to the Brightness object and its **enhance()** method.

To make the color different more obvious, the provided factor should be larger than 1.0. To reduce the contrast, the factor should be less than 1.0. But, a factor of 0.0 gives a solid gray image.

1. Add the following lines to the end of the existing code.

```
#%%
img13 = ImageEnhance.Contrast(img).enhance(0.5)
img13.show()
```

2. Run the code cell and compare the results of the contrast adjusted image and the original image.

3. Change the factor to 2.0, run it again and check the result.

Changing Color Balance

Changing the color balance adjusts the intensities of the colors. To do so, we create a Color object associated with the image object. Then, we call its **enhance()** with a factor to the result. A factor of 1.0 gives an original image. A factor of 0.0 gives a grayscale image. A factor of larger than 1.0 gives an image with sharper colors.

1. Add the following lines to the end of the existing code.

```
#%%
img14 = ImageEnhance.Color(img).enhance(2.0)
img14.show()
```

2. Run the code cell and compare the results of the color balance adjusted image and the original image.

3. Change the factor to 0.0 to give the grayscale image.

Combining Images with Interpolation Alpha Factor

The **Image.blend()** function allows us to combine two images into a single image. It accepts three parameters – *image1*, *image2*, and *alpha*. The *alpha* is a float value between 0.0 and 1.0 that represents the transparency level of *image1*. If alpha equals 0.7, it means the transparency level of *image1* is 0.7; the transparency level of *image2* is 1 – 0.7 = 0.3.

1. Add the following lines to the end of the existing code.

```
#%%
tmp1 = img.copy()
tmp2 = img.transpose(Image.FLIP_LEFT_RIGHT)
img15 = Image.blend(tmp1, tmp2, 0.5)
img15.show()
```

2. Run the code cell and check the result.

3. The following is the example output:

4. Change the alpha to 0.75 and run the code cell again to see the difference.

Exercise – Kaleidoscope

We need to define a function named **kaleidoscope** that accepts an image object. The **kaleidoscope()** function does the following actions:

- Crops the image to make its width and height identical.
- Combine the resulted image with its flipped image.
- Combine the resulted image with its rotated images.
- Enhance the color balance and contrast of the resulted image.

Hint:
You may use the functions – **crop()**, **transpose()**, **blend()**, **rotate()**, **ImageEnhance.Color()**, and **ImageEnhance.Contrast()**. You also need a loop statement help you to repeat the steps of combining rotated images.

The sample input (images/flower.jpg) and output are as follows: