

Python 数字图像处理-3 种图像读取方式总结

2018 年 11 月 10 日 by [harley](#) ·

学习数字图像处理，第一步就是读取图像。这里我总结下如何使用 opencv3,scikit-image,PIL 图像处理库读取图片并显示。

读取并显示图像代码

opencv3 库，示例代码如下：

```
import cv2

#读入一副彩色图像

img_cv2 = cv2.imread('test.jpg',cv2.IMREAD_COLOR)

#打印图像尺寸

print(type(img_cv2))

print(img_cv2.shape)

#matplotlib 绘制显示图像

plt.figure(1)

plt.imshow(img_PIL)

plt.show()

#cv2.imshow()函数显示图像

#cv2.namedWindow('image', cv2.WINDOW_NORMAL)

#cv2.imshow('image',img_cv2)
```

```
#cv2.waitKey(0)

#cv2.destroyAllWindows()
```

scikit-image 库，示例代码如下：

```
from skimage import io

img_skimage = io.imread('test.jpg')

#打印图像尺寸

print(img_skimage.shape)

#绘制显示图像

io.imshow(img_skimage)

#import matplotlib.pyplot as plt

#plt.imshow(img_skimage)
```

注意：io.imshow(img_skimage)，这一行代码的实质是利用 matplotlib 包对图片进行绘制，绘制成功后，返回一个 matplotlib 类型的数据。也就是说 scikit-image 库对图像的绘制实际上是调用了 matplotlib 库 imshow 显示函数。

PIL 库，示例代码如下：

```
from PIL import Image

import numpy as np

img_PIL = Image.open('test.jpg')

#img_PIL = np.array(img_PIL)    #获得 numpy 对象,RGB
```

```
#打印图像尺寸
print(img_PIL.size)

#绘制显示图像
plt.figure(1)
plt.imshow(img_PIL)
plt.show()
```

读取图像结果分析

分别用 Opnecv3 和 sckit-image 读取图像, 并用 matplotlib 库显示。示例代码如下:

```
import cv2
from skimage import io
img_cv2 = cv2.imread('test.jpg',cv2.IMREAD_COLOR)
img_skimage = io.imread('test.jpg')
#matplotlib 显示 cv2 库读取的图像
plt.figure('imread picture',figsize=(25,25))
plt.subplot(121)
plt.title('cv2 imread picture')
plt.imshow(img_cv2)
#matplotlib 显示 skimage 库读取的图像
plt.subplot(122)
plt.title('skimage imread picture')
```

```
plt.imshow(img_skimage)
```

#打印图像尺寸,总像素个数,和图像元素数据类型

```
print(img_cv2.shape)
```

```
print(img_cv2.size)
```

```
print(img_cv2.dtype)
```

```
1 import cv2
2 from skimage import io
3 img_cv2 = cv2.imread('test.jpg', cv2.IMREAD_COLOR)
4 img_skimage = io.imread('test.jpg')
5 #matplotlib显示cv2库读取的图像
6 plt.figure('imread picture', figsize=(25, 25))
7 plt.subplot(121)
8 plt.title('cv2 imread picture')
9 plt.imshow(img_cv2)
10
11 #matplotlib显示skimage库读取的图像
12 plt.subplot(122)
13 plt.title('skimage imread picture')
14 plt.imshow(img_skimage)
15
16 print(img_cv2.shape)
17 print(img_cv2.size)
18 print(img_cv2.dtype)
```

输出结果如下:



通过上图, 我们会发现, matplotlib 绘制显示的 cv2 库读取的图像与原图有所差别, 这是因为 opencv3 库读取图像的通

道时 BGR，而正常图像读取的通道都是 RGB，matplotlib 库显示图像也是按照 RGB 顺序通道来的，解释完毕。

一点疑惑，我通过查询库函数可知 plt.show() 第一个参数为要显示的对象 (array_like)，字面意思理解为类似数组的对象，但是很明显，PIL 库返回的不是 'numpy.ndarray' 对象，而是 'PIL.JpegImagePlugin.JpegImageFile' 对象，那为什么 plt.show() 函数还是能显示 Image.open() 函数读取图像返回的结果呢？程序如下图所示：

```
#PIL库读取绘制显示图像
from PIL import Image
import matplotlib.pyplot as plt # plt 用于显示图片
import numpy as np
img_PIL = Image.open('test.jpg')
img_PIL = np.array(img_PIL)
#打印图像类型, 尺寸和总像素个数
print(type(img_PIL))
print(img_PIL.shape)
print(img_PIL.size)
#绘制显示图像
plt.figure(1)
plt.imshow(img_PIL)
plt.show()
```

```
<class 'numpy.ndarray'>  
(1200, 1793, 3)  
6454800
```



plt.show 函数定义如下:

Signature: plt.imshow(X, cmap=None, norm=None, aspect=None, interpolation=None, alpha=None, vmin=None, vmax=None, origin=None, extent=None, shape=None, filternorm=1, filterrad=4.0, imlim=None, resample=None, url=None, hold=None, data=None, **kwargs)

Docstring:

Display an image on the axes.

Parameters

———

X : array_like, shape (n, m) or (n, m, 3) or (n, m, 4)

Display the image in `X` to current axes. `X` may be an array or a PIL image. If `X` is an array, it can have the following shapes and types:

- $M \times N$ — values to be mapped (float or int)
- $M \times N \times 3$ — RGB (float or uint8)
- $M \times N \times 4$ — RGBA (float or uint8)

The value for each component of $M \times N \times 3$ and $M \times N \times 4$ float arrays

should be in the range 0.0 to 1.0. $M \times N$ arrays are mapped to colors based on the `norm` (mapping scalar to scalar) and the `cmap` (mapping the normed scalar to a color).

读取并显示图像方法总结

PIL 库读取图像

`PIL.Image.open` + `numpy`

`scipy.misc.imread`

`scipy.ndimage.imread`

这些方法都是通过调用 `PIL.Image.open` 读取图像的信息；

`PIL.Image.open` 不直接返回 `numpy` 对象，可以用 `numpy` 提供的函数进行转换，参考 `Image` 和 `Ndarray` 互相转换；

其他模块都直接返回 `numpy.ndarray` 对象, 通道顺序为 RGB, 通道值得默认范围为 0–255。

Opencv3 读取图像

`cv2.imread`

使用 `opencv` 读取图像, 直接返回 `numpy.ndarray` 对象, 通道顺序为 BGR , 注意是 BGR, 通道值默认范围 0–255。

`skimage`

`skimage.io.imread`: 直接返回 `numpy.ndarray` 对象, 通道顺序为 RGB, 通道值默认范围 0–255。

`caffe.io.load_image`: 没有调用默认的 `skimage.io.imread`, 返回值为 0–1 的 `float` 型数据, 通道顺序为 RGB

scikit-image 库读取图像

`skimage.io.imread`: 直接返回 `numpy.ndarray` 对象, 通道顺序为 RGB, 通道值默认范围 0–255。

`caffe.io.load_image`: 没有调用默认的 `skimage.io.imread`, 返回值为 0–1 的 `float` 型数据, 通道顺序为 RGB

matplotlib 库显示图像

`matplotlib.image.imread`

从名字中可以看出这个模块是具有 matlab 风格的，直接返回 numpy.ndarray 格式通道顺序是 RGB，通道值默认范围 0–255。

参考资料

- <https://blog.csdn.net/renelian1572/article/details/78761278>
- <https://pillow.readthedocs.io/en/5.3.x/index.html>
- http://scikit-image.org/docs/stable/user_guide.html