**实验1 基于Python的opencv数字图像处理的函数**

**一、实验目的：**

1. 掌握基于python-opencv的图像基本操作；

2. 了解python-opencv处理图像与numpy库的关系和关联；

**二、实验内容：**

1. 使用python-opencv中的imread函数载入图片；

2. 使用python-opencv中的imshow和waitKey函数显示图片；

3. 使用python-opencv中的imwrite函数保存图像；

4．创建和复制图像；

5. RGB三通道分离并显示。

**三、实验指导：**

**1、图像的载入、显示和保存**

PythonOpenCV的配置这里就不介绍了。注意，现在OpenCV for Python就是通过NumPy进行绑定的。所以在使用时必须掌握一些NumPy的相关知识！图像就是一个矩阵，在OpenCV for Python中，图像就是NumPy中的数组！如果读取图像首先要导入OpenCV包，方法为：

import cv2

读取并显示图像在Python中不需要声明变量，所以也就不需要C++中的cv::Mat xxxxx了。只需这样：

img = cv2.imread("D:\cat.jpg")

OpenCV目前支持读取bmp、jpg、png、tiff等常用格式。更详细的请参考OpenCV的参考文档。接着创建一个窗口

cv2.namedWindow("Image")

然后在窗口中显示图像

cv2.imshow("Image", img)

最后还要添上一句：

cv2.waitKey (0)

如果不添最后一句，在IDLE中执行窗口直接无响应。在命令行中执行的话，则是一闪而过。完整的程序为：

import cv2

img = cv2.imread("D:\\cat.jpg")

cv2.namedWindow("Image")

cv2.imshow("Image", img)

cv2.waitKey (0)

cv2.destroyAllWindows()

最后释放窗口是个好习惯！

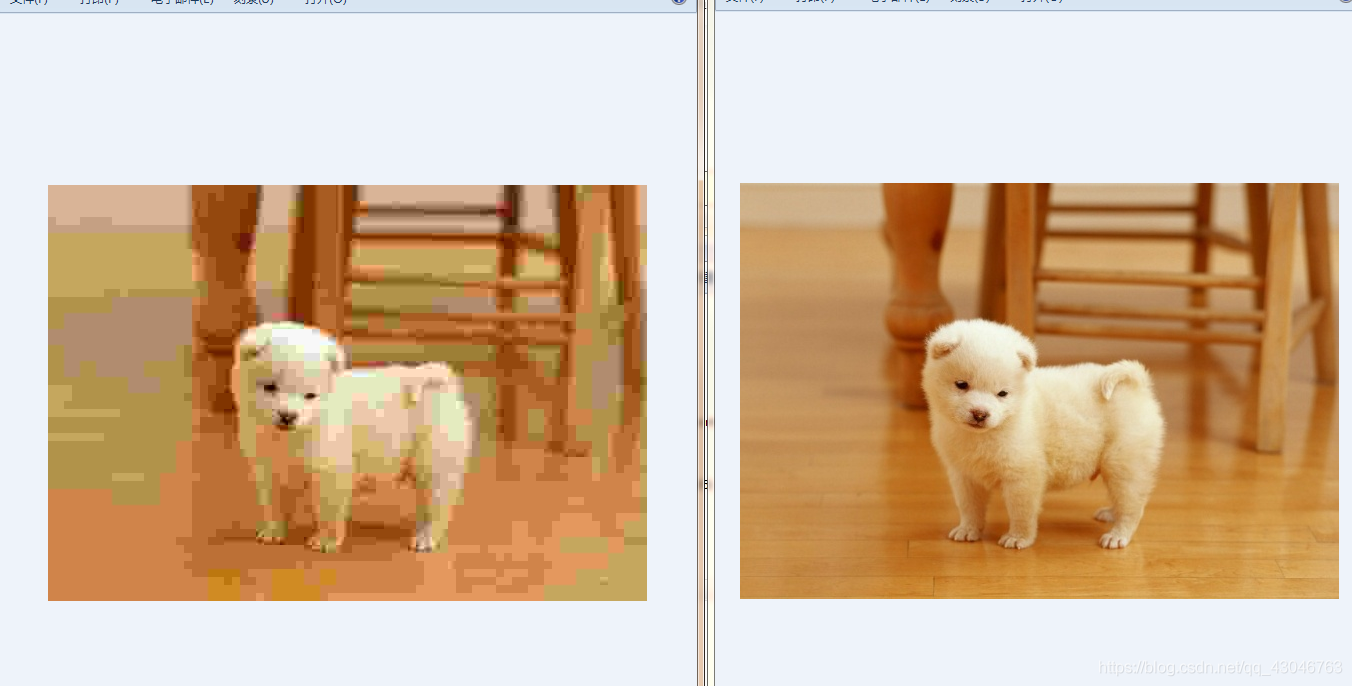
保存图像很简单，直接用cv2.imwrite即可：

cv2.imwrite("D:\\cat2.jpg", img)

第一个参数是保存的路径及文件名，第二个是图像矩阵。其中，imwrite()有个可选的第三个参数，如下：

cv2.imwrite("D:\\cat2.jpg",img[int(cv2.IMWRITE\_JPEG\_QUALITY), 5])

第三个参数针对特定的格式： 对于JPEG，其表示的是图像的质量，用0-100的整数表示，默认为95。 注意，cv2.IMWRITE\_JPEG\_QUALITY类型为Long，必须转换成int。下面是以不同质量存储的两幅图：



对于PNG，第三个参数表示的是压缩级别。cv2.IMWRITE\_PNG\_COMPRESSION，从0到9,压缩级别越高，图像尺寸越小。默认级别为3：

cv2.imwrite("./cat.png",img,[int(cv2.IMWRITE\_PNG\_COMPRESSION), 0])

cv2.imwrite("./cat2.png",img,[int(cv2.IMWRITE\_PNG\_COMPRESSION), 9])

还有一种支持的图像，一般不常用。

完整的代码为：

import numpy as np

import cv2

img=cv2.imread('2.png',cv2.IMREAD\_UNCHANGED)

cv2.namedWindow('WHY')

cv2.imshow('WHY',img)

cv2.imwrite("C:\\why0.jpg",img,[int(cv2.IMWRITE\_JPEG\_QUALITY),95])

cv2.imwrite("C:\\why1.jpg",img,[int(cv2.IMWRITE\_JPEG\_QUALITY),5])

cv2.waitKey(0)

**2、创建/复制图像**

新的OpenCV的接口中没有CreateImage接口。即没有cv2.CreateImage这样的函数。如果要创建图像，需要使用numpy的函数（现在使用OpenCV-Python绑定，numpy是必装的）。如下：

emptyImage = np.zeros(img.shape, np.uint8)

在新的OpenCV-Python绑定中，图像使用NumPy数组的属性来表示图像的尺寸和通道信息。如果输出img.shape，将得到(500, 375, 3)，这里是以网上找的图片为例。最后的3表示这是一个RGB图像。也可以复制原有的图像来获得一副新图像。

emptyImage2 = img.copy();

**3. RGB通道的分离与合并**

彩色图像是包含多通道的图像，比如用BGR三通道表示的彩色图像，或者是包含了alpha通道的BGRA四通道图像。有时做图像处理时如果多通道同时处理，可能并不能达到很好的效果，但是如果分离出某一个通道出来处理可能会有更好的效果，一个例子是在 来看看怎么用OpenCV解构Twitter大牛jagarikin的视觉错觉图 一文中可以看到在做二值化时只处理HSV色彩空间中的S分量效果更好。

（1）通道分离split()

通道分离可以用于彩色图像的处理，图像对象可以是普通的3通道BGR彩色图像，分离后分别为b、g、r的3个通道。如果是带alpha通道的BGRA 4通道图像，分离后分别为b、g、r、a。如果图像是其他色彩空间的图像比如HSV图像，分离后的3个图像则分别为h、s、v。下面的例子将lena.jpg和opencv-logo.png做通道分离，并将各分量显示出来，在代码中加入了通道数的判断，如果是3通道返回结果用b,g,r= cv2.split(img)接收分离结果，如果是4通道用b,g,r,a = cv2.split(img)接收分离结果：

import cv2

def show\_img(win\_name,img,wait\_time=0,img\_ratio=0.5,is\_show=True):

if is\_show is not True:

return

rows = img.shape[0]

cols = img.shape[1]

cv2.namedWindow(win\_name, cv2.WINDOW\_NORMAL )#cv2.WINDOW\_AUTOSIZE)

cv2.resizeWindow(win\_name,(int(cols\*img\_ratio),int(rows\*img\_ratio)))

cv2.imshow(win\_name,img)

if wait\_time >= 0:

cv2.waitKey(wait\_time)

img = cv2.imread('..\\lena.jpg')

#img = cv2.imread('..\\opencv-logo.png',cv2.IMREAD\_UNCHANGED)

if img is not None and len(img.shape)==3: #彩色图像才可以做通道分离

print('img.shape:',img.shape)

show\_img('img',img,-1)

if img.shape[2] == 3: #如果是3通道，分离出3个图像实例

b,g,r = cv2.split(img)

show\_img('b',b,-1)

show\_img('g',g,-1)

show\_img('r',r,-1)

cv2.waitKey(0)

elif img.shape[2] == 4: #如果是4通道

b,g,r,a = cv2.split(img)

show\_img('b',b,-1)

show\_img('g',g,-1)

show\_img('r',r,-1)

show\_img('a',a,-1)

cv2.waitKey(0)



（2）通道合并merge()

用已有的多个通道图像构造成一个元组传递给merge()，可以实现图像的合并。下面这个例子先分离出bgr通道再合并后显示合成图像：

import cv2

print('cv2.\_\_version\_\_:',cv2.\_\_version\_\_)

img = cv2.imread('..\\lena.jpg')

b = img[:,:,0]

g = img[:,:,1]

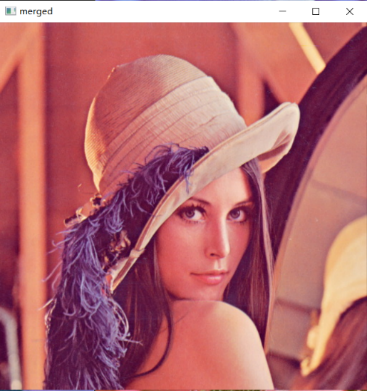
r = img[:,:,2]

img2 = cv2.merge((b,g,r)) #传入bgr构成的元组

cv2.imshow('merged',img2)

cv2.waitKey(0)

运行结果：



下面这个例子特意将bgr通道顺序做调换再合并：

img = cv2.imread('..\\lena.jpg')

b = img[:,:,0]

g = img[:,:,1]

r = img[:,:,2]

img2 = cv2.merge((b,g,r)) #传入bgr构成的元组

cv2.imshow('merged',img2)

img2 = cv2.merge((r,g,b))

cv2.imshow('merged-rgb',img2)

img2 = cv2.merge((r,b,g))

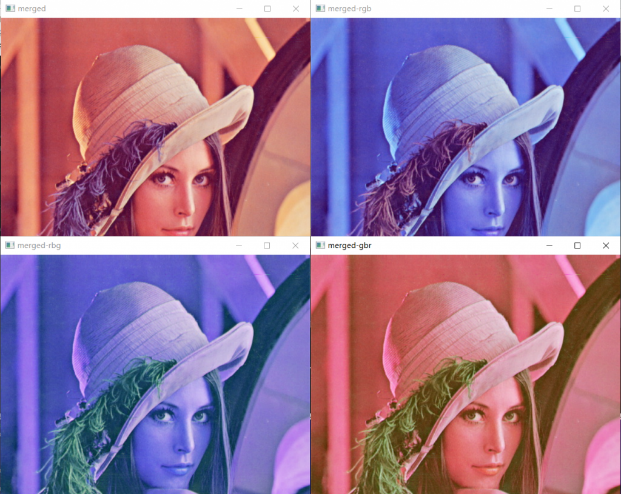
cv2.imshow('merged-rbg',img2)

img2 = cv2.merge((g,b,r))

cv2.imshow('merged-gbr',img2)

cv2.waitKey(0)

合并后图像的效果和原图对比：



**（3）索引方式通道合并**

和用索引方式进行通道分离一样，也可以用索引方式完成通道合并：

import numpy as np

import cv2

print('cv2.\_\_version\_\_:',cv2.\_\_version\_\_)

img = cv2.imread('..\lena.jpg')

b,g,r = cv2.split(img)

rows,cols,channels = img.shape[0],img.shape[1],img.shape[2]

img2 = np.zeros((rows,cols,channels),np.uint8) #创建全0的numpy数组

img2[:,:,0]=b

img2[:,:,1]=g

img2[:,:,2]=r

cv2.imshow('merged',img2)

cv2.waitKey()

**（4）“分离”灰度图**

在前面介绍的图像分离中都是针对彩色图像做多通道的分离，如果被分离的图像是单通道的灰度图，会是什么结果呢？下面这个例子读入lena.jpg时转换为灰度图，再使用split()进行分离：

import cv2

print('cv2.\_\_version\_\_:',cv2.\_\_version\_\_)

img = cv2.imread('..\\lena.jpg',cv2.IMREAD\_GRAYSCALE)

print('img.shape:',img.shape)

res = cv2.split(img)