## 利用opencv库进行车道线识别和车辆识别

### 1.1图像的读入和存储

#在开头引入必要的库

import matplotlib.pyplot as plt

import numpy as np

import cv2

#ipython

%matplotlib inline

**图像读取**

opencv使用imread读取图片,imshow显示图片,但是对于我而言,常使用jupyter作为展示的工具,imshow在浏览器环境中就失去了它的作用,因此我会使用matplotlib来展示

#读取一张图片

cv2.imread(img,flag)

#img这一参数中填入图片的完全路径或者相对路径

这里我们需要稍微了解下flag这个参数,这决定了opencv是如何读入我们的图像的  
我们知道通常图像每个像素点的颜色我们以RGB的格式来描述(或者RGBA),可以通过三基色(red,green,blue)来描述所有颜色,对于透明图片我们会增加一个a(alpha)来描述其颜色的透明度.

cv2.IMREAD\_COLOR : 读入图片,任何与透明度相关通道的会被忽视,默认以这种方式读入.

cv2.IMREAD\_GRAYSCALE : 以灰度图的形式读入图片.

cv2.IMREAD\_UNCHANGED : 保留读取图片原有的颜色通道.

可以简单的用-1,0,1来分别表示这3个flag

就让我们从著名的lenna图开始吧

lenna\_img = cv2.imread("lena.jpg")

plt.imshow(lenna\_img)

plt.axis("off")#去除坐标轴

plt.show()



当把这张图片打印出来后,你一定在疑惑,咦,这张图怎么变青色了?

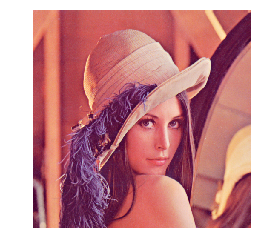
原因就在于opencv默认的imread是以BGR的方式进行存储的,而matplotlib的imshow默认则是以RGB格式展示,所以此处我们必须对图片的通道进行转换

lenna\_img = cv2.cvtColor(lenna\_img,cv2.COLOR\_BGR2RGB)

plt.imshow(lenna\_img)

plt.axis("off")

plt.show()



Lenna终于恢复了她的本来样子了  
这里我们了解一下cvtColor这个函数,它的第一个参数是图片,第二个参数则是颜色通道的转化方式  
它的命名是有规律的通常以COLOR作为开头,后面则跟着它的转化方式,BGR通道转化为RGB,因此就是cv2.COLOR\_BGR2RGB  
你可以试着猜测从RGB通道转化为BGR通道的api名,通过补全验证你的想法

我们在刚刚使用了默认的flag读入了图片,那么让我们用用另外两个试试效果

gray\_lenna\_img = cv2.imread("lena.jpg",0)

orign\_lenna\_img = cv2.imread("lena.jpg",1)

plt.subplot(121)

plt.imshow(gray\_lenna\_img,cmap=plt.cm.gray)

plt.axis("off")

plt.subplot(122)

orign\_lenna\_img = cv2.cvtColor(orign\_lenna\_img,cv2.COLOR\_BGR2RGB)

plt.imshow(orign\_lenna\_img)

plt.axis("off")

plt.show()



因为lenna图并没有包含透明度这一通道,读入的仍然是BGR格式,所以我们从lenna图是看不出区别的

**图片在python下的储存方式**

opencv的一个Image对象在python和C++下的存储方式是不同的,在c++下,通过opencv实现的Mat来进行存储,而python下则基于numpy

numpy对于使用python进行科学计算的人都不算陌生,它为python提供了一个高效的矩阵运算模块.

这就意味着我们可以直接使用numpy的api对图片进行计算和处理.

print("Lenna图在python中存储的类型为",type(lenna\_img))

print("读入lenna图的shape为",lenna\_img.shape)

print("以灰白图读入lenna图的shape为",gray\_lenna\_img.shape)

#Lenna图在python中存储的类型为 <class 'numpy.ndarray'>

#读入lenna图的shape为 (512, 512, 3)

#以灰白图读入lenna图的shape为 (512, 512)

上文实现的bgr装RGB我们也可以使用numpy来轻松的实现

lenna\_img = cv2.imread("lena.jpg")

b,g,r = cv2.split(lenna\_img)

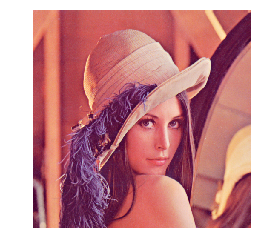
lenna\_img = cv2.merge([r,g,b])

plt.imshow(lenna\_img)

plt.axis("off")

plt.show()

#结果如下



我们可以对之前的函数进行一些包装,毕竟每次都要设定坐标轴为off怎么也会厌烦的啊

def show\_cv\_img(cv\_image):

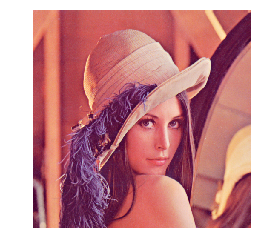
image = cv2.cvtColor(cv\_image,cv2.COLOR\_BGR2RGB)

plt.imshow(image)

plt.axis("off")

考虑到subplot此处没有直接plt.show

show\_cv\_img(cv2.imread("lena.jpg"))#jupyter result



**图片的展示与存储**

虽然说我们通常使用jupyter来使用opencv,但还是要了解下opencv默认的imshow

img = cv2.imread("lenna.jpg")

cv2.namedWindow('image', cv2.WINDOW\_NORMAL)#给显示的窗口命名,后面的flag默认为cv2.WINDOW\_AUTOSIZE,自动调整边框

#,但是在条形图过长时,使用windownormal我们可以自行调整边框

cv2.imshow('image',img)#展示图片

cv2.waitKey(0)#等待按键按下

cv2.destroyAllWindows()#清除所有窗口

我们可以使用imwrite来存储一张图片,接受一个numpy的数组作为参数.

cv2.imwrite('cope\_lenna\_img.jpg',lenna\_img)

#result: True

他会返回一个bool值来表示它是否成功存储.  
我们成功在当前目录存储了一个叫做copy\_lenna\_img.jpg的图像.

Warning:

注意后缀.你的后缀是jpg还是png决定它以怎样的方式保存

Opencv对于图像的读入和存储都已经封装好了给我们,是我们能够轻松的读入,存储,避免去了解图片文件的格式才能读取,存储图片,但仅仅这样显然不是opencv的真正面目,还不如直接用pillow呢.