# 一、图像简单读取、显示与储存

#图像读取

1. 在读取图像之前，把做实验的图像事先放到工作目录下。
2. 读取函数cv2.imread()

cv2.imread(‘图像名称’，’可选参数’)  
 可选参数决定读入图像的模式：  
 ****0：读入的为灰度图像****（即使图像为彩色的）  
 ****1：读入的图像为彩色的****（默认）；

Ps:即使图像在工作空间不存在，这个函数也不会报错，只不过读入的结果为none。

1. 读取light1.jpg 图像

import cv2  
 img = cv2.imread(‘light.jpg’)

#图像显示

1. opencv自带函数显示，一般要显示一个图像，在图像有了之后，然后直接下面函数一起组合使用

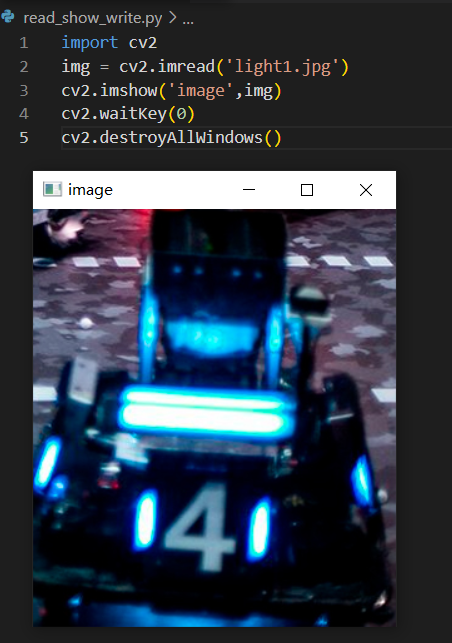
cv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()

函数cv2.imshow（‘名字’，‘要显示的图’）

为了让图像在显示屏上停留才有了下面两句话一起使用。



1. 使用matplotlib库包中的pyplot子包来显示，这种显示方法不但稳定，还可以调整大小、保存显示的图像等等。

import cv2

from matplotlib import pyplot as plt

img = cv2.imread('light1.jpg',0);#打开为灰度图像

plt.imshow(img, 'gray') #必须规定为显示的为什么图像

plt.xticks([]),plt.yticks([]) #隐藏坐标线

plt.show() #显示出来，不要也可以，但是一般都要了



#图像储存

1. 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。

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。

1. 保存light1的灰度图为light\_gray.tiff

cv2.imwrite(‘light1\_gray.tif’,img)

# 二、简单几何图像绘制

#绘制直线

1. 函数为：cv2.line（img,Point pt1,Point pt2,color,thickness=1,line\_type=8 shift=0）  
   有值的代表有默认值，不用给也行。

可以看到这个函数主要接受参数为两个点的坐标，线的颜色（彩色图像的话颜色就是一个1\*3 的数组）

1. 绘制直线代码：

#灰色

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = np.zeros((512,512),np.uint8)#生成一个空灰度图像

cv2.line(img,(0,0),(511,511),255,5)

plt.imshow(img,'gray')

#彩色

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = np.zeros((512,512,3),np.uint8)#生成一个空彩色图像

cv2.line(img,(0,0),(511,511),(0,255,0),5)#绿色

plt.imshow(img,'brg')

#绘制矩形

1. 函数：cv2.rectangle(img,(380,0),(511,111),(255,0,0),3)，需要确定的就是矩形的两个点（左上角与右下角），颜色，线的类型（不设置就默认）。
2. 绘制矩形代码：

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = np.zeros((512,512,3),np.uint8)#生成一个空彩色图像

cv2.rectangle(img,(20,20),(411,411),(55,255,155),5)

plt.imshow(img,'brg')

#绘制圆形

1. 函数：cv2.circle (img,(380,0),63,(255,0,0),3) 圆心，半径，颜色，线粗
2. 绘制圆形代码：

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = np.zeros((512,512,3),np.uint8)#生成一个空彩色图像

cv2.circle(img,(200,200),50,(55,255,155),1)#修改最后一个参数

plt.imshow(img,'brg')

#绘制椭圆

1. 函数：cv2.ellipse(img,(256,256),(150,100),0,0,180,250,-1)圆心坐标，轴长，偏转角，起始角，终止角，颜色，粗细，线条类型，精度
2. 绘制椭圆代码：

import cv2

import numpy as np

from matplotlib import pyplot as plt

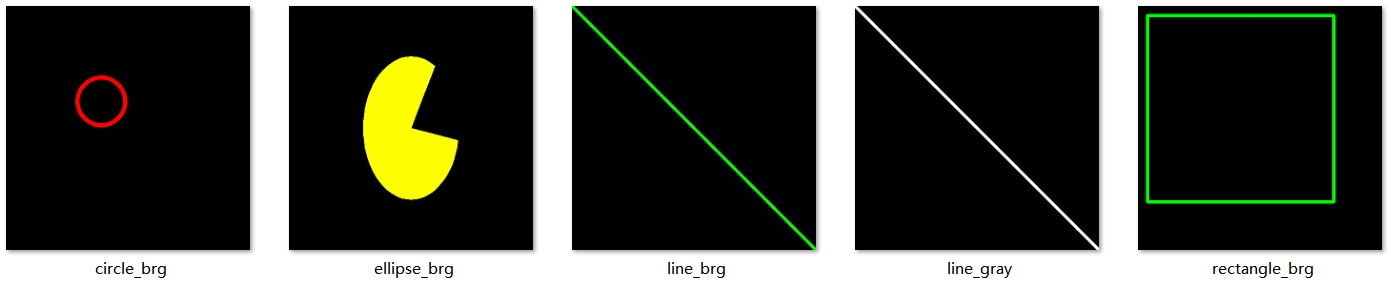
img = np.zeros((512,512,3),np.uint8)#生成一个空彩色图像

cv2.ellipse(img,(256,256),(150,100),0,0,180,250,-1)

#注意最后一个参数-1，表示对图像进行填充，默认是不填充的，如果去掉，只有椭圆轮廓了

plt.imshow(img,'brg')

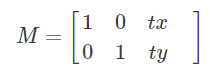
1. 效果：



# 三、图像的几何变换

1. 图像的平移

1）图像的平移，沿着x方向tx距离，y方向ty距离，需要构造移动矩阵：



2）通过numpy来产生这个矩阵，并将其赋值给仿射函数cv2.warpAffine().

3）仿射函数cv2.warpAffine() 原始图像，移动矩阵M ，变换的图像大小（这个大小如果不和原始图像大小相同，那么函数会自动通过插值来调整像素间的关系）

4）代码：

import cv2

import numpy as np

import matplotlib.pyplot as plt

img = cv2.imread('light1.jpg')

H = np.float32([[1,0,100],[0,1,50]])

rows,cols = img.shape[:2]

res = cv2.warpAffine(img,H,(rows,cols)) #需要图像、变换矩阵、变换后的大小

plt.subplot(121)

plt.imshow(img,'brg')

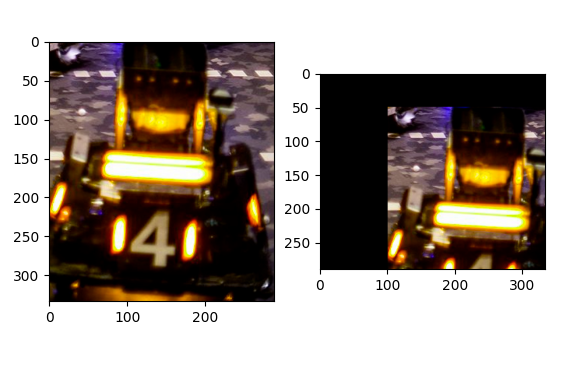
plt.subplot(122)

plt.imshow(res,'brg')

#plt.show()

cv2.imwrite("light\_warpAffine.jpg", res)

5)效果：



1. 图像的扩大与缩小

1)伸缩需要确定的就是缩放比例，可以是x与y方向相同倍数，也可以单独设置x与y的缩放比例。

2)缩放有几种不同的插值（interpolation）方法，在缩小时推荐cv2.INTER\_ARER,扩大是推荐cv2.INTER\_CUBIC和cv2.INTER\_LINEAR。默认都是cv2.INTER\_LINEAR

3)**cv2.resize(InputArray src, OutputArray dst, Size dsize, double fx=0, double fy=0, int interpolation=INTER\_LINEAR)**

I**nputArray src** ：输入，原图像，即待改变大小的图像；

**OutputArray dst**： 输出，改变后的图像。这个图像和原图像具有相同的内容，只是大小和原图像不一样而已；

**dsize**：输出图像的大小

其中，fx和fy就是下面要说的两个参数，是图像width方向和height方向的缩放比例。

**fx**：width方向的缩放比例

**fy**：height方向的缩放比例

**interpolation（插值）**：这个是指定插值的方式，图像缩放之后，肯定像素要进行重新计算的，就靠这个参数来指定重新计算像素的方式，有以下几种：

INTER\_NEAREST - 最邻近插值

INTER\_LINEAR - 双线性插值，如果最后一个参数你不指定，默认使用这种方法

INTER\_CUBIC - 4x4像素邻域内的双立方插值

INTER\_LANCZOS4 - 8x8像素邻域内的Lanczos插值

1. 代码：

import cv2

import numpy as np

import matplotlib.pyplot as plt

img = cv2.imread('light1.jpg')

# 插值：interpolation

# None本应该是放图像大小的位置的，后面设置了缩放比例，

#所有就不要了

res1 = cv2.resize(img,None,fx=8,fy=8,interpolation=cv2.INTER\_CUBIC)

#直接规定缩放大小，这个时候就不需要缩放因子

height,width = img.shape[:2]

res2 = cv2.resize(img,(4\*width,4\*height),interpolation=cv2.INTER\_CUBIC)

plt.subplot(131)

plt.imshow(img,'brg')

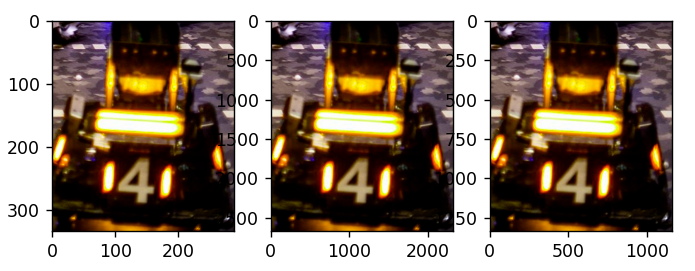
plt.subplot(132)

plt.imshow(res1,'brg')

plt.subplot(133)

plt.imshow(res2,'brg')

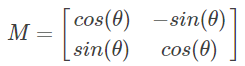
1. 效果：



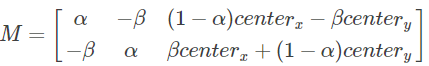
1\*1倍 ， 8\*8倍 ， 4\*4倍

1. 图像的旋转

1）图像的旋转矩阵一般为：



但是单纯的这个矩阵是在原点处进行变换的，为了能够在任意位置进行旋转变换，opencv采用了另一种方式：



1. opencv提供了一个函数：cv2.getRotationMatrix2D()，旋转中心，旋转角度，旋转后图像的缩放比例
2. 代码：

import cv2

import numpy as np

import matplotlib.pyplot as plt

img = cv2.imread('light1.jpg')

res = cv2.warpAffine(img,M,(rows,cols))

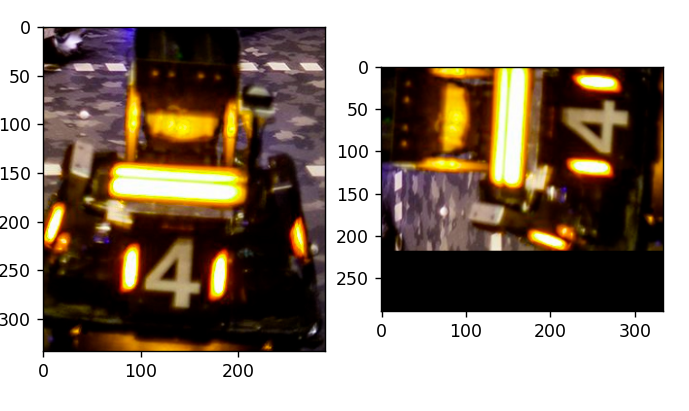
plt.subplot(121)

plt.imshow(img)

plt.subplot(122)

plt.imshow(res)

1. 效果：



1. 图像的仿射
2. 图像的旋转加上拉升就是图像仿射变换，仿射变化也是需要一个M矩阵就可以，但是由于仿射变换比较复杂。
3. [opencv](https://so.csdn.net/so/search?q=opencv&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/on2way/article/details/_blank)提供了根据变换前后三个点的对应关系来自动求解M。这个函数是  
   M=cv2.getAffineTransform(pos1,pos2)

,其中两个位置就是变换前后的对应位置关系。输出的就是仿射矩阵M。然后在使用函数cv2.warpAffine()。

1. 代码：

import cv2

import numpy as np

import matplotlib.pyplot as plt

img = cv2.imread('light1.jpg')

rows,cols = img.shape[:2]

pts1 = np.float32([[65,140],[215,150],[55,300]])

pts2 = np.float32([[10,100],[200,50],[100,250]])

M = cv2.getAffineTransform(pts1,pts2)

#第三个参数：变换后的图像大小

res = cv2.warpAffine(img,M,(rows,cols))

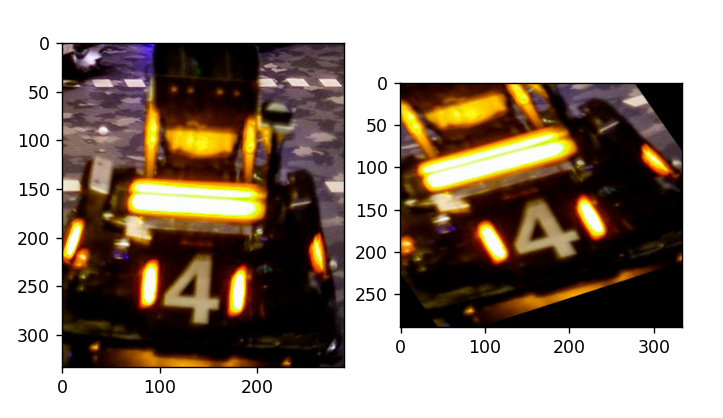
plt.subplot(121)

plt.imshow(img)

plt.subplot(122)

plt.imshow(res)

4）效果：



1. 图像的透射
2. 透视需要的是一个3\*3的矩阵，同理opencv在构造这个矩阵的时候还是采用一种点对应的关系来通过函数自己寻找的，因为我们自己很难计算出来。这个函数是M = cv2.getPerspectiveTransform(pts1,pts2)，其中pts需要变换前后的4个点对应位置。得到M后在通过函数cv2.warpPerspective(img,M,(200,200))进行。
3. 代码：

import cv2

import numpy as np

import matplotlib.pyplot as plt

rows,cols = img.shape[:2]

pts1 = np.float32([[87,191],[191,198],[78,294],[187,292]])

pts2 = np.float32([[0,0],[200,0],[0,200],[200,200]])

M = cv2.getPerspectiveTransform(pts1,pts2)

res = cv2.warpPerspective(img,M,(200,200))

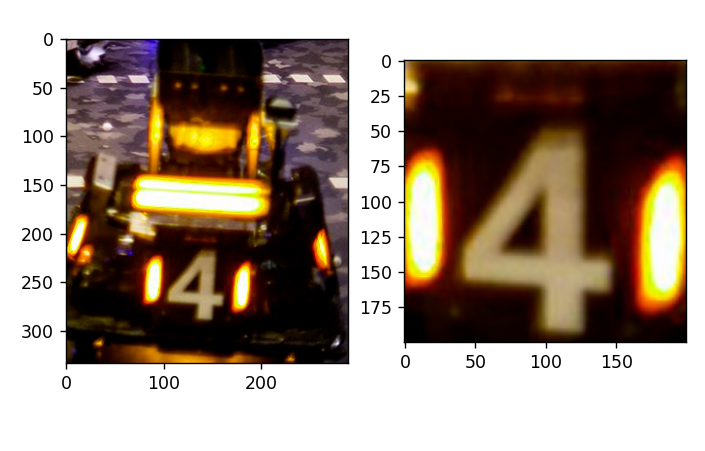
plt.subplot(121)

plt.imshow(img,'Blues')

plt.subplot(122)

plt.imshow(res,'brg')

1. 效果:



图像的阈值处理