# opencv-python——基础图像处理操作总结

原创

[IrvingGao](https://blog.51cto.com/u_15906550) 2022-12-08 14:50:26 博主文章分类：[opencv](https://blog.51cto.com/u_15906550/category8) ©著作权

**文章标签** [**opencv**](https://blog.51cto.com/topic/opencv.html)[**python**](https://blog.51cto.com/topic/python-2.html)[**cv**](https://blog.51cto.com/topic/cv.html)[**OpenCV**](https://blog.51cto.com/topic/opencv.html)[**ide**](https://blog.51cto.com/topic/ide.html) **文章分类** [**OpenStack**](https://blog.51cto.com/nav/openstack)[**云计算**](https://blog.51cto.com/nav/cloud) **阅读数969**

©著作权归作者所有：来自51CTO博客作者IrvingGao的原创作品，请联系作者获取转载授权，否则将追究法律责任

### 文章目录

* ​​1.图片的读取保存​​
* ​​（1）加载并显示图片​​
* ​​加载彩色图​​
* ​​加载灰度图​​
* ​​加载原图并转换通道​​
* ​​（2）保存图片​​
* ​​Opencv图像处理操作​​
* ​​（1）截取ROI​​
* ​​（2）通道分割与合并​​
* ​​（3）颜色空间转换​​
* ​​示例：实现一个使用HSV来只显示图片中蓝色物体​​
* ​​（4）阈值分割​​
* ​​阈值调节器​​
* ​​固定阈值分割​​
* ​​自适应阈值​​
* ​​阈值分割方法小结​​
* ​​（5）图像几何变换​​
* ​​缩放图片​​
* ​​翻转图片​​
* ​​平移图片​​
* ​​（6）绘图功能​​
* ​​绘制线条​​
* ​​绘制圆​​
* ​​绘制矩形​​
* ​​添加文字​​
* ​​绘制封闭多边形或折线​​
* ​​绘图功能总结​​
* ​​（7）图像间数学运算​​
* ​​图片叠加​​
* ​​图像混合​​
* ​​图像拼接​​
* ​​图像运算操作总结​​
* ​​（8）图像滤波​​
* ​​滤波与模糊​​
* ​​均值滤波​​
* ​​高斯滤波​​
* ​​中值滤波​​
* ​​双边滤波​​
* ​​图像锐化​​
* ​​（9）腐蚀与膨胀​​
* ​​腐蚀​​
* ​​膨胀​​
* ​​开/闭运算​​
* ​​（10）OpenCV摄像头与视频操作​​
* ​​打开摄像头​​
* ​​播放本地视频​​
* ​​录制视频​​
* ​​摄像头视频操作小结​​
* ​​（11）寻找外接轮廓​​
* ​​寻找最大外接正矩形​​
* ​​寻找最小外接任意矩形​​
* ​​寻找最小外接多边形​​
* ​​（12）直方图均衡化​​
* ​​灰度直方图均衡化​​
* ​​（13）灰度图转伪色彩图​​
* ​​函数总结​​

看到有其他的博客整理的opencv的知识点和函数非常的全面，所以我这里就按着他的思路，进行了进一步的整理，并选取其中比较重要的部分进行了再一次学习。

## 1.图片的读取保存

* 加载图片，显示图片，保存图片
* OpenCV函数：​​cv2.imread()​​​, ​​cv2.imshow()​​​, ​​cv2.imwrite()​​ 说明：
* OpenCV中彩色图是以B-G-R通道顺序存储的，灰度图只有一个通道。
* OpenCV默认使用BGR格式，而RGB和BGR的颜色转换不同，即使转换为灰度也是如此。一些开发人员认为R+G+B/3对于灰度是正确的，但最佳灰度值称为亮度（luminosity），并且具有公式：0.21R+0.72G+0.07\*B
* 图像坐标的起始点是在左上角，所以行对应的是y，列对应的是x。

### （1）加载并显示图片

使用​​cv2.imread()​​来读入一张图片：

1. 参数1：图片的文件名

* 如果图片放在当前文件夹下，直接写文件名就行了，如’lena.jpg’；否则需要给出绝对路径，如’D:\OpenCVSamples\lena.jpg’

1. 参数2：读入方式，省略即采用默认值

* ​​cv2.IMREAD\_COLOR​​：彩色图，默认值(1)
* ​​cv2.IMREAD\_GRAYSCALE​​：灰度图(0)
* ​​cv2.IMREAD\_UNCHANGED​​：包含透明通道的彩色图(-1)

登录后复制

import numpy as np  
import cv2  
import matplotlib.pyplot as plt

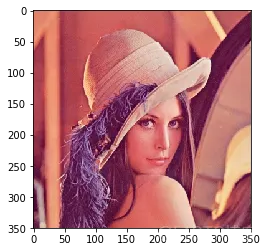
#### 加载彩色图

需要注意的是，在​​matplotlib​​​里，默认的图像通道为RGB，所以我们需要在opencv读取后先将BGR通道转换为RGB通道，然后再使用​​matplotlib​​显示图片。

登录后复制

# 加载彩色图  
img = cv2.imread('lena.jpg', 1)  
# 将彩色图的BGR通道顺序转成RGB  
img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  
# 显示图片  
plt.imshow(img)  
  
# 打印图片的形状  
print(img.shape)  
# 形状中包括行数、列数和通道数  
height, width, channels = img.shape  
# img是灰度图的话：height, width = img.shape  
img.shape

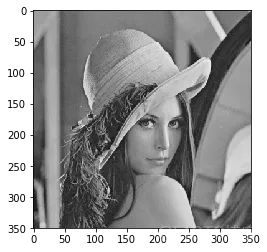
(350, 350, 3)



#### 加载灰度图

登录后复制

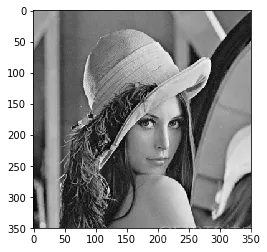
# 加载灰度图  
img = cv2.imread('lena.jpg', 0)  
# 将彩色图的BGR通道顺序转成RGB  
img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  
plt.imshow(img)



#### 加载原图并转换通道

登录后复制

# 加载彩色图  
img = cv2.imread('lena.jpg', 1)  
# 将彩色图的BGR通道直接转为灰度图  
img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
plt.imshow(img,'gray')



### （2）保存图片

登录后复制

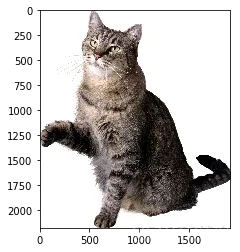
#保存图片  
cv2.imwrite('lena-grey.jpg',img)

## Opencv图像处理操作

需要注意的是，在​​matplotlib​​​里，默认的图像通道为RGB，所以我们需要在opencv读取后先将BGR通道转换为RGB通道，然后再使用​​matplotlib​​显示图片。

登录后复制

# 加载四通道图片  
img1 = cv2.imread('cat.png',-1)  
# 将彩色图的BGR通道顺序转成RGB，注意，在这一步直接丢掉了alpha通道  
img1 = cv2.cvtColor(img1, cv2.COLOR\_BGR2RGB)   
plt.imshow(img1)



如果没有进行通道转换，显示图像的颜色就会变反。

登录后复制

# 加载彩色图  
img1 = cv2.imread('cat.png',1)  
# 不转颜色通道  
plt.imshow(img1)

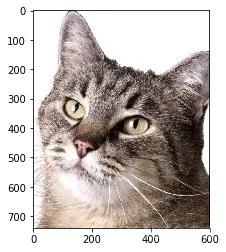


### （1）截取ROI

* ROI：Region of Interest，感兴趣区域。
* 截取ROI非常简单，指定图片的范围即可。

登录后复制

# 截取猫脸ROI  
face = img[0:740, 400:1000]  
plt.imshow(face)



### （2）通道分割与合并

彩色图的BGR三个通道是可以分开单独访问的，也可以将单独的三个通道合并成一副图像。分别使用​​cv2.split()​​​和​​cv2.merge()​​：

登录后复制

# 创建一副图片  
img2 = cv2.imread('lena.jpg')  
  
# 通道分割  
b, g, r = cv2.split(img2)  
  
# 通道合并  
img2 = cv2.merge((r, g, b))  
  
plt.imshow(img2)

、

RGB\_Image=cv2.merge([b,g,r])  
RGB\_Image = cv2.cvtColor(RGB\_Image, cv2.COLOR\_BGR2RGB)  
plt.figure(figsize=(12,12))  
#显示各通道信息  
plt.subplot(141)  
plt.imshow(RGB\_Image,'gray')  
plt.title('RGB\_Image')  
plt.subplot(142)  
plt.imshow(r,'gray')  
plt.title('R\_Channel')  
plt.subplot(143)  
plt.imshow(g,'gray')  
plt.title('G\_Channel')  
plt.subplot(144)  
plt.imshow(b,'gray')  
plt.title('B\_Channel')



### （3）颜色空间转换

最常用的颜色空间转换如下：

* RGB或BGR到灰度（​​COLOR\_RGB2GRAY​​​，​​COLOR\_BGR2GRAY​​）
* RGB或BGR到YcrCb（或YCC）（​​COLOR\_RGB2YCrCb​​​，​​COLOR\_BGR2YCrCb​​）
* RGB或BGR到HSV（​​COLOR\_RGB2HSV​​​，​​COLOR\_BGR2HSV​​）
* RGB或BGR到Luv（​​COLOR\_RGB2Luv​​​，​​COLOR\_BGR2Luv​​​）  
  - 灰度到RGB或BGR（​​​COLOR\_GRAY2RGB​​​，​​COLOR\_GRAY2BGR​​）

#### 示例：实现一个使用HSV来只显示图片中蓝色物体

* HSV是一个常用于颜色识别的模型，相比BGR更易区分颜色，转换模式用​​COLOR\_BGR2HSV​​表示。

登录后复制

# 加载一张有天空的图片  
sky = cv2.imread('sky.jpg')  
sk1 = cv2.cvtColor(sky, cv2.COLOR\_BGR2RGB)  
plt.imshow(sk1)



# 蓝色的范围，不同光照条件下不一样，可灵活调整  
lower\_blue = np.array([15, 60, 60])  
upper\_blue = np.array([130, 255, 255])  
  
# 从BGR转换到HSV  
hsv = cv2.cvtColor(sky, cv2.COLOR\_BGR2HSV)  
# inRange()：介于lower/upper之间的为白色，其余黑色  
mask = cv2.inRange(sky, lower\_blue, upper\_blue)  
# 只保留原图中的蓝色部分  
res = cv2.bitwise\_and(sky, sky, mask=mask)  
res = cv2.cvtColor(res, cv2.COLOR\_BGR2RGB)  
plt.imshow(res)



不同颜色的HSV转换可以查表：​​【OpenCV】HSV颜色识别-HSV基本颜色分量范围​​

### （4）阈值分割

* 使用固定阈值、自适应阈值和Otsu阈值法 “二值化” 图像
* OpenCV函数：​​cv2.threshold()​​​, ​​cv2.adaptiveThreshold()​​

#### 阈值调节器

在很多时候，我们需要找到图片的特定通道的阈值，就需要通过调整参数来尝试和确定数值。如果通过拖动的滚条来调节阈值，就可以提高调试效率，缩短试错时间。  
​opencv-python——图片的HSV和RGB滚动条阈值分析器​​

#### 固定阈值分割

固定阈值分割很直接，一句话说就是像素点值大于阈值变成一类值，小于阈值变成另一类值。

​​cv2.threshold()​​用来实现阈值分割，ret是return value缩写，代表当前的阈值。函数有4个参数：

* 参数1：要处理的原图，一般是灰度图
* 参数2：设定的阈值
* 参数3：最大阈值，一般为255
* 参数4：阈值的方式，主要有5种，详情：ThresholdTypes
* 0: THRESH\_BINARY 当前点值大于阈值时，取Maxval,也就是第四个参数，否则设置为0
* 1: THRESH\_BINARY\_INV 当前点值大于阈值时，设置为0，否则设置为Maxval
* 2: THRESH\_TRUNC 当前点值大于阈值时，设置为阈值，否则不改变
* 3: THRESH\_TOZERO 当前点值大于阈值时，不改变，否则设置为0
* 4:THRESH\_TOZERO\_INV 当前点值大于阈值时，设置为0，否则不改变

登录后复制

import cv2  
  
# 灰度图读入  
img = cv2.imread('lena.jpg', 0)  
# 颜色通道转换  
img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  
# 阈值分割  
ret, th = cv2.threshold(img, 127, 255, cv2.THRESH\_BINARY)  
  
plt.imshow(th)



# 应用5种不同的阈值方法  
# THRESH\_BINARY 当前点值大于阈值时，取Maxval,否则设置为0  
ret, th1 = cv2.threshold(img, 127, 255, cv2.THRESH\_BINARY)  
# THRESH\_BINARY\_INV 当前点值大于阈值时，设置为0，否则设置为Maxval  
ret, th2 = cv2.threshold(img, 127, 255, cv2.THRESH\_BINARY\_INV)  
# THRESH\_TRUNC 当前点值大于阈值时，设置为阈值，否则不改变  
ret, th3 = cv2.threshold(img, 127, 255, cv2.THRESH\_TRUNC)  
# THRESH\_TOZERO 当前点值大于阈值时，不改变，否则设置为0  
ret, th4 = cv2.threshold(img, 127, 255, cv2.THRESH\_TOZERO)  
# THRESH\_TOZERO\_INV 当前点值大于阈值时，设置为0，否则不改变  
ret, th5 = cv2.threshold(img, 127, 255, cv2.THRESH\_TOZERO\_INV)  
  
titles = ['Original', 'BINARY', 'BINARY\_INV', 'TRUNC', 'TOZERO', 'TOZERO\_INV']  
images = [img, th1, th2, th3, th4, th5]  
  
plt.figure(figsize=(12,12))  
for i in range(6):  
 plt.subplot(2, 3, i + 1)  
 plt.imshow(images[i], 'gray')  
 plt.title(titles[i], fontsize=8)  
 plt.xticks([]), plt.yticks([])



#### 自适应阈值

看得出来固定阈值是在整幅图片上应用一个阈值进行分割，它并不适用于明暗分布不均的图片。​​cv2.adaptiveThreshold()​​自适应阈值会每次取图片的一小部分计算阈值，这样图片不同区域的阈值就不尽相同。它有5个参数，其实很好理解，先看下效果：

* 参数1：要处理的原图
* 参数2：最大阈值，一般为255
* 参数3：小区域阈值的计算方式
* ​​ADAPTIVE\_THRESH\_MEAN\_C​​：小区域内取均值
* ​​ADAPTIVE\_THRESH\_GAUSSIAN\_C​​：小区域内加权求和，权重是个高斯核
* 参数4：阈值方式（跟前面讲的那5种相同）
* 参数5：小区域的面积，如11就是11\*11的小块
* 参数6：最终阈值等于小区域计算出的阈值再减去此值

可以调整下参数看看不同的结果。

登录后复制

# 自适应阈值对比固定阈值  
img = cv2.imread('lena.jpg', 0)  
  
# 固定阈值  
ret, th1 = cv2.threshold(img, 127, 255, cv2.THRESH\_BINARY)  
# 自适应阈值, ADAPTIVE\_THRESH\_MEAN\_C：小区域内取均值  
th2 = cv2.adaptiveThreshold(  
 img, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY, 11, 4)  
# 自适应阈值, ADAPTIVE\_THRESH\_GAUSSIAN\_C：小区域内加权求和，权重是个高斯核  
th3 = cv2.adaptiveThreshold(  
 img, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv2.THRESH\_BINARY, 17, 6)  
  
titles = ['Original', 'Global(v = 127)', 'Adaptive Mean', 'Adaptive Gaussian']  
images = [img, th1, th2, th3]  
plt.figure(figsize=(12,12))  
for i in range(4):  
 plt.subplot(2, 2, i + 1), plt.imshow(images[i], 'gray')  
 plt.title(titles[i], fontsize=8)  
 plt.xticks([]), plt.yticks([])



#### 阈值分割方法小结

* ​​cv2.threshold()​​​用来进行固定阈值分割。固定阈值不适用于光线不均匀的图片，所以用​​cv2.adaptiveThreshold()​​进行自适应阈值分割。

### （5）图像几何变换

* 实现旋转、平移和缩放图片
* OpenCV函数：​​cv2.resize()​​​, ​​cv2.flip()​​​, ​​cv2.warpAffine()​​

#### 缩放图片

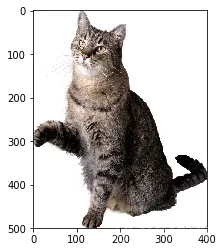
缩放就是调整图片的大小，使用​​cv2.resize()​​​函数实现缩放。可以按照比例缩放，也可以按照指定的大小缩放：  
我们也可以指定缩放方法interpolation，更专业点叫插值方法，默认是​​​INTER\_LINEAR​​。

放过程中有五种插值方式：

* ​​cv2.INTER\_NEAREST​​ 最近邻插值
* ​​cv2.INTER\_LINEAR​​ 线性插值
* ​​cv2.INTER\_AREA​​ 基于局部像素的重采样，区域插值
* ​​cv2.INTER\_CUBIC​​ 基于邻域4x4像素的三次插值
* ​​cv2.INTER\_LANCZOS4​​ 基于8x8像素邻域的Lanczos插值

登录后复制

img = cv2.imread('cat.png')  
img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  
  
# 按照指定的宽度、高度缩放图片  
res = cv2.resize(img, (400, 500))  
# 按照比例缩放，如x,y轴均放大一倍  
res2 = cv2.resize(img, None, fx=2, fy=2, interpolation=cv2.INTER\_LINEAR)  
plt.imshow(res)

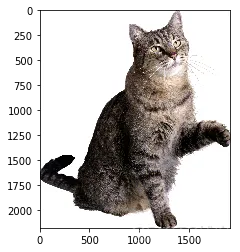


#### 翻转图片

镜像翻转图片，可以用​​cv2.flip()​​​函数：  
其中，参数2 = 0：垂直翻转(沿x轴)，参数2 > 0: 水平翻转(沿y轴)，参数2 < 0: 水平垂直翻转。

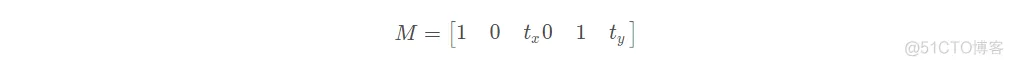
登录后复制

dst = cv2.flip(img, 1)  
plt.imshow(dst)



#### 平移图片

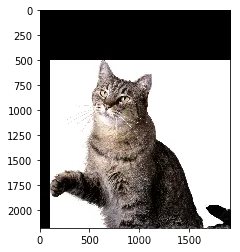
要平移图片，我们需要定义下面这样一个矩阵，tx,ty是向x和y方向平移的距离：



平移是用仿射变换函数​​cv2.warpAffine()​​实现的：

登录后复制

# 平移图片  
import numpy as np  
# 获得图片的高、宽  
rows, cols = img.shape[:2]  
# 定义平移矩阵，需要是numpy的float32类型  
# x轴平移100，y轴平移500  
M = np.float32([[1, 0, 100], [0, 1, 500]])  
# 用仿射变换实现平移  
dst = cv2.warpAffine(img, M, (cols, rows))  
  
plt.imshow(dst)



### （6）绘图功能

* 绘制各种几何形状、添加文字
* OpenCV函数：​​cv2.line()​​​, ​​cv2.circle()​​​, ​​cv2.rectangle()​​​, ​​cv2.ellipse()​​​, ​​cv2.putText()​​​, ​​cv2.polyline()​​

绘制形状的函数有一些共同的参数，提前在此说明一下：

* ​​img​​：要绘制形状的图片
* ​​color​​：绘制的颜色
* 彩色图就传入BGR的一组值，如蓝色就是(255,0,0)
* 灰度图，传入一个灰度值就行
* ​​thickness​​：线宽，默认为1；对于矩形/圆之类的封闭形状而言，传入-1表示填充形状
* ​​lineType​​：线的类型。默认情况下，它是8连接的。cv2.LINE\_AA 是适合曲线的抗锯齿线。

#### 绘制线条

画直线只需指定**起点**和**终点**的坐标就行：

登录后复制

img = cv2.imread('lena.jpg')  
img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

登录后复制

# 画一条线宽为5的红色直线，参数2：起点，参数3：终点  
cv2.line(img, (0, 0), (800, 512), (255, 0, 0), 5)  
plt.imshow(img)



#### 绘制圆

登录后复制

# cv2.circle(图像, 圆点, 半径, 颜色, 宽度)  
cv2.circle(img, center, radius, color, thickness)

当绘制**点**时，只需将宽度设为-1则表示全封闭，就是点了。

#### 绘制矩形

画矩形需要知道左上角和右下角的坐标：

登录后复制

# 画一个矩形，左上角坐标(40, 40)，右下角坐标(80, 80)，框颜色为绿色  
img = cv2.rectangle(img, (40, 40), (80, 80), (0, 255, 0),2)   
plt.imshow(img)



绘制实心填充矩形：

登录后复制

# 画一个矩形，左上角坐标(40, 40)，右下角坐标(80, 80)，框颜色为绿色，填充这个矩形  
img = cv2.rectangle(img, (40, 40), (80, 80), (0, 255, 0),-1)   
plt.imshow(img)



#### 添加文字

使用​​cv2.putText()​​添加文字，它的参数也比较多，同样请对照后面的代码理解这几个参数：

* 参数2：要添加的文本
* 参数3：文字的起始坐标（左下角为起点）
* 参数4：字体
* 参数5：文字大小（缩放比例）

登录后复制

# 添加文字，加载字体  
font = cv2.FONT\_HERSHEY\_SIMPLEX  
# 添加文字hello  
cv2.putText(img, 'hello', (10, 200), font,  
 4, (255, 255, 255), 2, lineType=cv2.LINE\_AA)  
  
plt.imshow(img)



写中文字符：

登录后复制

# 引入PIL的相关包  
from PIL import Image, ImageFont,ImageDraw  
from numpy import unicode  
  
def paint\_chinese\_opencv(im,chinese,pos,color):  
 img\_PIL = Image.fromarray(cv2.cvtColor(im,cv2.COLOR\_BGR2RGB))  
 # 加载中文字体  
 font = ImageFont.truetype('NotoSansCJKsc-Medium.otf',25)  
 # 设置颜色  
 fillColor = color  
 # 定义左上角坐标  
 position = pos  
 # 判断是否中文字符  
 if not isinstance(chinese,unicode):  
 # 解析中文字符  
 chinese = chinese.decode('utf-8')  
 # 画图  
 draw = ImageDraw.Draw(img\_PIL)  
 # 画文字  
 draw.text(position,chinese,font=font,fill=fillColor)  
 # 颜色通道转换  
 img = cv2.cvtColor(np.asarray(img\_PIL),cv2.COLOR\_RGB2BGR)  
 return img  
  
plt.imshow(paint\_chinese\_opencv(img,'中文',(100,100),(255,255,0)))



#### 绘制封闭多边形或折线

使用​​cv2.polylines()​​绘制折线，输入的这几个参数：

* 参数2：pts，必须是​​int​​类型
* 参数3：是否闭合
* 参数4：颜色
* 参数5：绘制边大小

登录后复制

image =np.zeros((500, 500, 3), dtype=np.uint8)  
  
data=np.array([[172.0, 82], [194.0, 86], [182, 151.0], [194, 86]]) # [N,2]  
roi\_as = []  
roi\_as.append(data.astype(np.int)) # 输入的pts必须是个列表  
cv2.polylines(image, roi\_as, True, (255, 0, 255)) # 画任意多边形

#### 绘图功能总结

* ​​cv2.line()​​​画直线，​​cv2.circle()​​​画圆，​​cv2.rectangle()​​​画矩形，​​cv2.ellipse()​​​画椭圆，​​cv2.polylines()​​​画多边形，​​cv2.putText()​​添加文字。
* 画多条直线时，​​cv2.polylines()​​​要比​​cv2.line()​​高效很多。
* 要在图像中打上中文，可以用PIL库结合OpenCV实现。

### （7）图像间数学运算

* 图片间的数学运算，如相加、按位运算等
* OpenCV函数：​​cv2.add()​​​, ​​cv2.addWeighted()​​​, ​​cv2.bitwise\_and()​​

#### 图片叠加

要叠加两张图片，可以用​​cv2.add()​​函数，相加两幅图片的形状（高度/宽度/通道数）必须相同。numpy中可以直接用res = img + img1相加，但这两者的结果并不相同：

登录后复制

x = np.uint8([250])  
y = np.uint8([10])  
print(cv2.add(x, y)) # 250+10 = 260 => 255  
print(x + y) # 250+10 = 260 % 256 = 4

[[255]]  
[4]

如果是二值化图片（只有0和255两种值），两者结果是一样的（用numpy的方式更简便一些）。

#### 图像混合

图像混合​​cv2.addWeighted()​​也是一种图片相加的操作，只不过两幅图片的权重不一样，γ相当于一个修正值：

opencv-python——基础图像处理操作总结_python_23

img1 = cv2.imread('lena.jpg')  
img1 = cv2.cvtColor(img1, cv2.COLOR\_BGR2RGB)  
img2 = cv2.imread('cat.png')  
img2 = cv2.cvtColor(img2, cv2.COLOR\_BGR2RGB)  
img2 = cv2.resize(img2, (350, 350))  
# 两张图片相加  
res = cv2.addWeighted(img1, 0.6, img2, 0.4, 0)  
  
plt.imshow(res)



img1 = cv2.imread('lena.jpg')  
img2 = cv2.imread('logo.jpg')  
img2 = cv2.resize(img2, (350, 350))  
# 把logo放在左上角，所以我们只关心这一块区域  
rows, cols = img2.shape[:2]  
roi = img1[:rows, :cols]  
  
# 创建掩膜  
img2gray = cv2.cvtColor(img2, cv2.COLOR\_BGR2GRAY)  
ret, mask = cv2.threshold(img2gray, 10, 255, cv2.THRESH\_BINARY)  
mask\_inv = cv2.bitwise\_not(mask)  
  
# 保留除logo外的背景  
img1\_bg = cv2.bitwise\_and(roi, roi, mask=mask\_inv)  
dst = cv2.add(img1\_bg, img2) # 进行融合  
img1[:rows, :cols] = dst # 融合后放在原图上  
  
plt.imshow(dst)



#### 图像拼接

登录后复制

img1 = cv2.imread('lena.jpg')  
img2 = cv2.imread('logo.jpg')  
img3=cv2.hconcat([img1,img2])#水平拼接  
img4=cv2.vconcat([img1,img2])#垂直拼接

#### 图像运算操作总结

* ​​cv2.add()​​​用来叠加两幅图片，​​cv2.addWeighted()​​也是叠加两幅图片，但两幅图片的权重不一样。
* ​​cv2.bitwise\_and()​​​, ​​cv2.bitwise\_not()​​​, ​​cv2.bitwise\_or()​​​, ​​cv2.bitwise\_xor()​​分别执行按位与/或/非/异或运算。掩膜就是用来对图片进行全局或局部的遮挡。

### （8）图像滤波

* 模糊/平滑图片来消除图片噪声
* OpenCV函数：​​cv2.blur()​​​, ​​cv2.GaussianBlur()​​​,​​cv2.medianBlur()​​​, ​​cv2.bilateralFilter()​​

#### 滤波与模糊

关于滤波和模糊：

* 它们都属于卷积，不同滤波方法之间只是卷积核不同（对线性滤波而言）
* **低通滤波器是模糊，高通滤波器是锐化**

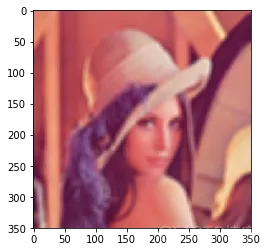
**低通滤波器**就是允许低频信号通过，在图像中边缘和噪点都相当于高频部分，所以低通滤波器用于去除噪点、平滑和模糊图像。**高通滤波器**则反之，用来增强图像边缘，进行锐化处理。

#### 均值滤波

均值滤波是一种最简单的滤波处理，它取的是卷积核区域内元素的均值，用​​cv2.blur()​​实现，如3×3的卷积核：



img = cv2.imread('lena.jpg')  
blur = cv2.blur(img, (3, 3)) # 均值模糊  
img = cv2.imread('lena.jpg')  
img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  
blur = cv2.blur(img, (9, 9)) # 均值模糊  
plt.imshow(blur)



#### 高斯滤波

前面两种滤波方式，卷积核内的每个值都一样，也就是说图像区域中每个像素的权重也就一样。高斯滤波的卷积核权重并不相同：中间像素点权重最高，越远离中心的像素权重越小。

显然这种处理元素间权值的方式更加合理一些。图像是2维的，所以我们需要使用2维的高斯函数，比如OpenCV中默认的3×3的高斯卷积核



OpenCV中对应函数为​​cv2.GaussianBlur(src,ksize,sigmaX)​​​:  
参数3 σx值越大，模糊效果越明显。高斯滤波相比均值滤波效率要慢，但可以有效消除高斯噪声，能保留更多的图像细节，所以经常被称为最有用的滤波器。均值滤波与高斯滤波的对比结果如下（均值滤波丢失的细节更多）

登录后复制

# 均值滤波vs高斯滤波  
gaussian = cv2.GaussianBlur(img, (9, 9), 1) # 高斯滤波  
plt.imshow(gaussian)

#### 中值滤波

中值又叫中位数，是所有数排序后取中间的值。中值滤波就是用区域内的中值来代替本像素值，所以那种孤立的斑点，如0或255很容易消除掉，适用于去除椒盐噪声和斑点噪声。中值是一种非线性操作，效率相比前面几种线性滤波要慢。

登录后复制

median = cv2.medianBlur(img, 9) # 中值滤波  
plt.imshow(median)



#### 双边滤波

登录后复制

blur = cv2.bilateralFilter(img, 9, 75, 75) # 双边滤波  
plt.imshow(blur)



#### 图像锐化

登录后复制

kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]], np.float32) #定义一个核  
dst = cv2.filter2D(img, -1, kernel=kernel)  
plt.imshow(dst)



### （9）腐蚀与膨胀

* 了解形态学操作的概念
* 学习膨胀、腐蚀、开运算和闭运算等形态学操作
* OpenCV函数：​​cv2.erode()​​​, ​​cv2.dilate()​​​,​​cv2.morphologyEx()​​

形态学操作：形态学操作其实就是改变物体的形状，比如腐蚀就是"变瘦"，膨胀就是"变胖"。

#### 腐蚀

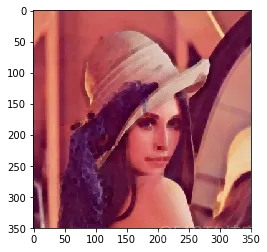
腐蚀的效果是把图片"变瘦"，其原理是在原图的小区域内取局部最小值。因为是二值化图，只有0和255，所以小区域内有一个是0该像素点就为0。

这样原图中边缘地方就会变成0，达到了瘦身目的

OpenCV中用​​cv2.erode()​​函数进行腐蚀，只需要指定核的大小就行：

登录后复制

img = cv2.imread('lena.jpg')  
img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  
kernel = np.ones((5, 5), np.uint8)  
erosion = cv2.erode(img, kernel) # 腐蚀  
plt.imshow(erosion)



这个核也叫结构元素，因为形态学操作其实也是应用卷积来实现的。结构元素可以是矩形/椭圆/十字形，可以用​​cv2.getStructuringElement()​​来生成不同形状的结构元素，比如：

#### 膨胀

膨胀与腐蚀相反，取的是局部最大值，效果是把图片"变胖"：

登录后复制

kernel = cv2.getStructuringElement(cv2.MORPH\_RECT, (5, 5)) # 矩形结构  
kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (5, 5)) # 椭圆结构  
kernel = cv2.getStructuringElement(cv2.MORPH\_CROSS, (5, 5)) # 十字形结构  
dilation = cv2.dilate(img, kernel) # 膨胀  
plt.imshow(dilation)



#### 开/闭运算

先腐蚀后膨胀叫开运算（因为先腐蚀会分开物体，这样容易记住），其作用是：分离物体，消除小区域。这类形态学操作用​​cv2.morphologyEx()​​函数实现：

登录后复制

kernel = cv2.getStructuringElement(cv2.MORPH\_RECT, (5, 5)) # 定义结构元素  
opening = cv2.morphologyEx(img, cv2.MORPH\_OPEN, kernel) # 开运算  
plt.imshow(opening)

闭运算则相反：先膨胀后腐蚀（先膨胀会使白色的部分扩张，以至于消除/"闭合"物体里面的小黑洞，所以叫闭运算）

登录后复制

closing = cv2.morphologyEx(img, cv2.MORPH\_CLOSE, kernel) # 闭运算  
plt.imshow(closing)



### （10）OpenCV摄像头与视频操作

* 打开摄像头并捕获照片
* 播放本地视频，录制视频
* OpenCV函数：​​cv2.VideoCapture()​​​, ​​cv2.VideoWriter()​​

#### 打开摄像头

要使用摄像头，需要使用cv2.VideoCapture(0)创建VideoCapture对象，参数0指的是摄像头的编号，如果你电脑上有两个摄像头的话，访问第2个摄像头就可以传入1，依此类推。

登录后复制

# 打开摄像头并灰度化显示  
import cv2  
  
capture = cv2.VideoCapture(0)  
while(True):  
 # 获取一帧  
 ret, frame = capture.read()  
 # 将这帧转换为灰度图  
 gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
  
 cv2.imshow('frame', gray)  
 if cv2.waitKey(1) == ord('q'):  
 break

​​capture.read()​​​函数返回的第1个参数ret(return value缩写)是一个布尔值，表示当前这一帧是否获取正确。​​cv2.cvtColor()​​用来转换颜色，这里将彩色图转成灰度图。

另外，通过​​cap.get(propId)​​​可以获取摄像头的一些属性，比如捕获的分辨率，亮度和对比度等。propId是从0~18的数字，代表不同的属性，完整的属性列表可以参考：VideoCaptureProperties。也可以使用​​cap.set(propId,value)​​来修改属性值。比如说，我们在while之前添加下面的代码：

登录后复制

# 获取捕获的分辨率  
# propId可以直接写数字，也可以用OpenCV的符号表示  
width, height = capture.get(3), capture.get(4)  
print(width, height)  
  
# 以原分辨率的一倍来捕获  
capture.set(cv2.CAP\_PROP\_FRAME\_WIDTH, width \* 2)  
capture.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, height \* 2)

#### 播放本地视频

跟打开摄像头一样，如果把摄像头的编号换成视频的路径就可以播放本地视频了。回想一下​​cv2.waitKey()​​，它的参数表示暂停时间，所以这个值越大，视频播放速度越慢，反之，播放速度越快，通常设置为25或30。

登录后复制

# 播放本地视频  
capture = cv2.VideoCapture('demo\_video.mp4')  
  
while(capture.isOpened()):  
 ret, frame = capture.read()  
 gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
  
 cv2.imshow('frame', gray)  
 if cv2.waitKey(30) == ord('q'):  
 break

#### 录制视频

之前我们保存图片用的是​​cv2.imwrite()​​​，要保存视频，我们需要创建一个​​VideoWriter​​的对象，需要给它传入四个参数：

* 输出的文件名，如​​’output.avi’​​
* 编码方式FourCC码
* 帧率FPS
* 要保存的分辨率大小

FourCC是用来指定视频编码方式的四字节码，所有的编码可参考Video Codecs。如MJPG编码可以这样写： ​​cv2.VideoWriter\_fourcc(\*'MJPG')​​​或​​cv2.VideoWriter\_fourcc('M','J','P','G')​​

登录后复制

capture = cv2.VideoCapture(0)  
  
# 定义编码方式并创建VideoWriter对象  
fourcc = cv2.VideoWriter\_fourcc(\*'MJPG')  
outfile = cv2.VideoWriter('output.avi', fourcc, 25., (640, 480))  
  
while(capture.isOpened()):  
 ret, frame = capture.read()  
  
 if ret:  
 outfile.write(frame) # 写入文件  
 cv2.imshow('frame', frame)  
 if cv2.waitKey(1) == ord('q'):  
 break  
 else:  
 break

#### 摄像头视频操作小结

* 使用​​cv2.VideoCapture()​​创建视频对象，然后在循环中一帧帧显示图像。参数传入数字时，代表打开摄像头，传入本地视频路径时，表示播放本地视频。
* ​​cap.get(propId)​​​获取视频属性，​​cap.set(propId,value)​​设置视频属性。
* ​​cv2.VideoWriter()​​创建视频写入对象，用来录制/保存视频。

### （11）寻找外接轮廓

在这里的​​box\_list​​​统一都为：​​[左上,右上,左下,右下]​​

#### 寻找最大外接正矩形

登录后复制

# Creating contour to track red color   
contours, hierarchy = cv2.findContours(color\_mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  
c = max(contours, key=cv2.contourArea)  
left\_x, left\_y, width, height = cv2.boundingRect(c)  
bound\_rect = np.array([[[left\_x, left\_y]], [[left\_x + width, left\_y]],  
 [[left\_x + width, left\_y + height]], [[left\_x, left\_y+height]]])  
box\_list = bound\_rect.tolist()  
cv2.drawContours(color\_image, [bound\_rect], -1, (255, 255, 255), 2)

#### 寻找最小外接任意矩形

登录后复制

contours, hierarchy = cv2.findContours(color\_mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  
c = max(contours, key=cv2.contourArea)  
rect = cv2.minAreaRect(c)  
box = cv2.boxPoints(rect)  
box\_list = box.tolist()  
# box\_list对应坐标点(坐标值可以为负)  
# [右下,左下,左上,右上]  
cv2.drawContours(color\_image, [np.int0(box)], -1, (0, 255, 255), 2)

#### 寻找最小外接多边形

登录后复制

contours,hierarchy = cv2.findContours(binary,cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_SIMPLE)#查找轮廓  
print(len(contours))  
x = 0  
for i in range(len(contours)):  
 area = cv2.contourArea(contours[i])  
 if area>10000: # 面积最小限制  
 print(area)  
 x = i  
cnt = contours[x]  
img1 = img.copy()  
approx1 = cv2.approxPolyDP(cnt,3,True)#拟合精确度,越小越精确  
img1 =cv2.polylines(img1,[approx1],True,(255,255,0),2)  
cv2.imshow('approxPolyDP1',img1)

实例参考：​​多边形拟合​​

### （12）直方图均衡化

#### 灰度直方图均衡化

对图像进行直方图均衡化（归一化图像亮度和增强图像对比度）

* 输入：单通道图像,例如[256, 256],里边的值为0,1,2
* 输出：单通道图像,里边的值为0,127,255

登录后复制

gray\_img = cv2.equalizeHist(gray\_img) # 直方图均衡化

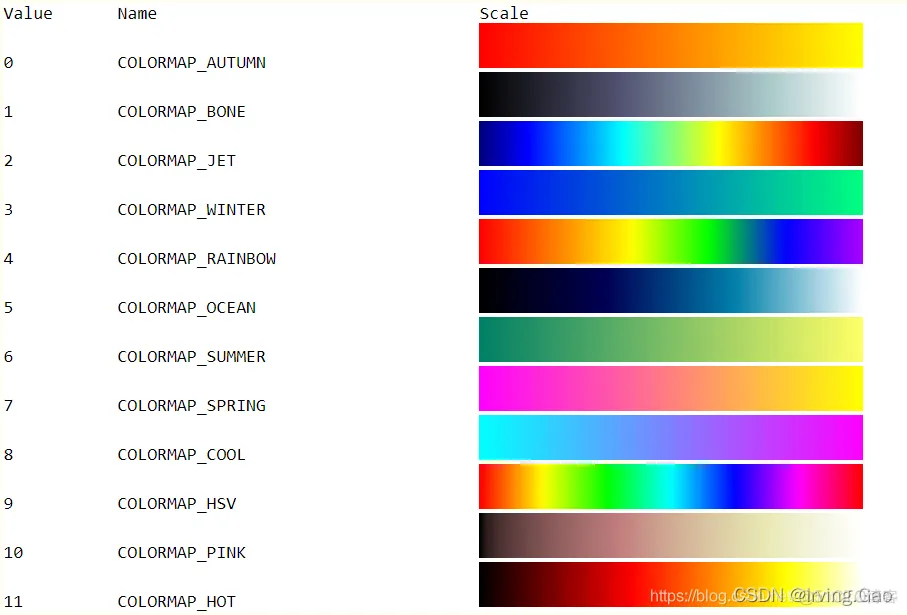
### （13）灰度图转伪色彩图

登录后复制

img\_color = cv2.applyColorMap(cv2.convertScaleAbs(gray\_img, alpha=1), cv2.COLORMAP\_JET) # 伪彩色图

在程序中有alpha这个值，会影响效果。

可选的颜色空间如下图所示：



参考文章：​​灰度图（深度图）转伪彩色​​

### 函数总结

读取图片cv2.imread()

展示图片cv2.imshow()

保存图片cv2.imwrite()

通道转换cv2.cvtColor()

RGB或BGR到灰度 COLOR\_RGB2GRAY，COLOR\_BGR2GRAY

RGB或BGR到YcrCb（或YCC）COLOR\_RGB2YCrCb，COLOR\_BGR2YCrCb

RGB或BGR到HSV COLOR\_RGB2HSV，COLOR\_BGR2HSV

RGB或BGR到Luv COLOR\_RGB2Luv，COLOR\_BGR2Luv

灰度到RGB或BGR COLOR\_GRAY2RGB，COLOR\_GRAY2BGR

阈值分割cv2.threshold()

自适应阈值分割cv2.adaptiveThreshold()

缩放图片cv2.resize()

翻转图片cv2.flip()

平移图片cv2.warpAffine()

画线cv2.line()

画圆cv2.circle()

画矩形cv2.rectangle()

画椭圆cv2.ellipse()

画多边形cv2.polylines()

添加文字cv2.putText()

叠加图片cv2.add()

图像混合cv2.addWeighted()

按位与运算cv2.bitwise\_and()

按位或运算cv2.bitwise\_not()

按位非运算cv2.bitwise\_or()

按位异或运算cv2.bitwise\_xor()

均值滤波cv2.blur()

高斯滤波cv2.GaussianBlur()

方框滤波cv2.boxFilter()

中值滤波cv2.medianBlur()

双边滤波cv2.bilateralFilter()

图像锐化cv2.filter2D()

边缘检测cv2.Canny()

腐蚀cv2.erode()

膨胀cv2.dilate()

开/闭运算cv2.morphologyEx()

打开摄像头cv2.VideoCapture(0)

读帧capture.read()

获取摄像头属性cap.get(propId)

设置摄像头属性cap.set(propId,value)

播放视频cv2.waitKey()

保存视频cv2.VideoWriter()

参考文章：

* ​[​面向初学者的OpenCV-Python教程​](http://codec.wang/#/opencv/)​

​​PaddlePaddle领航团 OpenCV基础知识点总结​​