# opencv中图像叠加/图像融合/按位操作的实现

**一、图像叠加：cv2.add**

res=cv2.add(img1, img2) 或者res=cv2.add(img1, 标量值)

参数说明： cv2.add将两个图片对应位置的像素的值相加，或者将每个像素的值加上一个标量值，大于255的像素值就设置成255。

有一点需要注意的是，如果是两张图片相加，那么一定要注意两者的尺寸和通道数必须是一样的；如果是标量值，这个值既可以是整数也可以是浮点数，加合适的标量值一般是为了提高亮度。

import cv2

img1 = cv2.imread('1.jpg')

img2 = cv2.imread('2.jpg')

print(img1.shape)

print(img2.shape)

#输出： (1039, 750, 3)

#(1050, 700, 3)

img2.resize((img1.shape[0],img1.shape[1],3))

print(img2.shape)

#输出：(1039, 750, 3)

res=cv2.add(img1,img2)#或者res=cv2.add(img1,10)

上面说了，cv2.add会把超过255的值设置为255，但是numpy里的加法进行的是模运算，请看下面的例子：

import cv2

import numpy as np

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

输出：  
[[255]]  
[4]

**二、图像融合：cv2.addWeighted**

cv2.addWeighted(src1, alpha, src2, beta, gamma[, dst[, dtype]]) → dst

dst = src1 \* alpha + src2 \* beta + gamma

src1 – 第一张图片  
alpha – 第一张图片的权重  
src2 – 与第一张大小和通道数相同的图片  
beta – 第二张图片的权重  
gamma – 加到每个总和上的标量，相当于调亮度  
dst – 输出

当然，这里也要注意图片尺寸要一样

import cv2

import numpy as np

img1 = cv2.imread('1.jpg')

img2 = cv2.imread('2.jpg')

#统一图片大小

img2 = cv2.resize(img2,(img1.shape[1],img1.shape[0]))

dst = cv2.addWeighted(img1,0.5,img2,0.5,0)

cv2.imshow('dst',dst)

cv2.waitKey(0)

cv2.destroyAllWindows()

你可以根据需要自己调整两个图片的权重，以达到不同的显示效果

**三、图像的按位操作：cv2.bitwise\_and**

'''

注意，src1和src2的形状要保持一致，一般都是同一张图像，

关键是在于mask，mask必须得是8-bit单通道array，尺寸也要和src相同

'''

bitwise\_and(src1, src2[, dst[, mask]]) -> dst

如果将两幅图片直接相加会改变图片的颜色，如果用图像混合，则会改变图片的透明度，所以我们需要用按位操作。首先来了解一下掩膜（mask）的概念：掩膜是用一副二值化图片对另外一幅图片进行局部的遮挡，看下图就一目了然了：(此处参考了 原文链接)

所以我们的思路就是把原图中要放logo的区域抠出来，再把logo放进去就行了：

import cv2

import numpy as np

img1 = cv2.imread('lena.jpg')

img2 = cv2.imread('opencv-logo-white.jpg')

# 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

cv2.imshow('res',img1)

cv2.waitKey(0)

cv2.destroyAllWindows()

其实看到这，很多人会有一些疑问，包括我自己刚开始的时候也是有疑问：

为什么要用两个roi进行与运算，roi&roi不还是roi本身吗？

mask和roi尺寸也一样，而且我们想要在roi中去除的区域在mask中对应位置的像素值正好也为0，为什么不让roi和mask两者直接相与呢？

这两个问题在我查资料的时候是这么说的，仅供参考:

* 因为mask是单通道的array，这是规定的，而roi是三通道，所以两者不能直接相与。
* 于是先利用roi和roi相与得到roi本身，而mask可以控制相与之后输出数据的某些元素发生变化，而相与之后的输出就是roi，所以此时相当于直接对roi进行操作，使roi中和mask中像素值为0的像素点对应的像素点的像素值也为0，也即变成黑的，这就达到了我们想要的效果。

到此这篇关于opencv中图像叠加/图像融合/按位操作的实现的文章就介绍到这了,更多相关opencv 图像叠加/图像融合/按位操作内容请搜索呐喊教程以前的文章或继续浏览下面的相关文章希望大家以后多多支持呐喊教程！

（万能的小黑Alex）