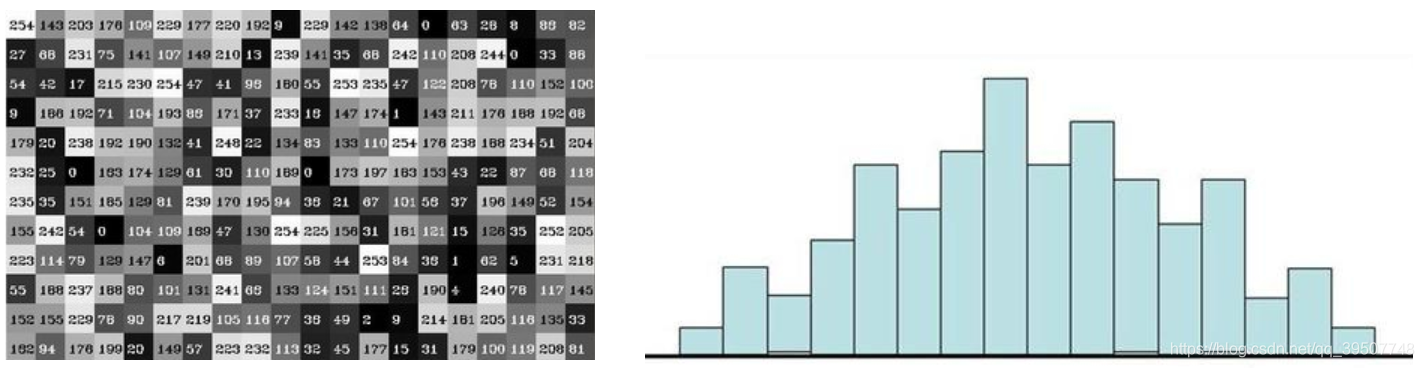
**opencv学习笔记十七：直方图和直方图均衡化(cv2.calcHist、 cv2.equalizeHist)**

**一、计算图像**[直方图](https://so.csdn.net/so/search?q=%E7%9B%B4%E6%96%B9%E5%9B%BE&spm=1001.2101.3001.7020)**的函数：cv2.calcHist()**

cv2.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate ]]) ->hist

* 1

imaes:输入的图像  
channels:选择图像的[通道](https://so.csdn.net/so/search?q=%E9%80%9A%E9%81%93&spm=1001.2101.3001.7020)  
mask:掩膜，是一个大小和image一样的np[数组](https://so.csdn.net/so/search?q=%E6%95%B0%E7%BB%84&spm=1001.2101.3001.7020)，其中把需要处理的部分指定为1，不需要处理的部分指定为0，一般设置为None，表示处理整幅图像  
histSize:使用多少个bin(柱子)，一般为256  
ranges:像素值的范围，一般为[0,255]表示0~255  
后面两个参数基本不用管。  
**注意，除了mask，其他四个参数都要带[]号。**  


img = cv2.imread('cat.jpg',0) #0表示灰度图

hist = cv2.calcHist([img],[0],None,[256],[0,256])

hist.shape#(256, 1)

**1、无mask参数**

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

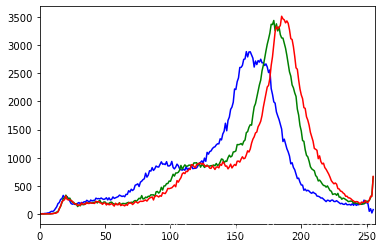
color = ('b','g','r')

for i,col in enumerate(color):

histr = cv2.calcHist([img],[i],None,[256],[0,256])

plt.plot(histr,color = col)

plt.xlim([0,256])

  
**2、使用mask参数**

# 创建mask

mask = np.zeros(img.shape[:2], np.uint8)

mask[100:300, 100:400] = 255

img = cv2.imread('cat.jpg', 0)

masked\_img = cv2.bitwise\_and(img, img, mask=mask)#与操作

hist\_full = cv2.calcHist([img], [0], None, [256], [0, 256])

hist\_mask = cv2.calcHist([img], [0], mask, [256], [0, 256])

plt.subplot(221), plt.imshow(img, 'gray')

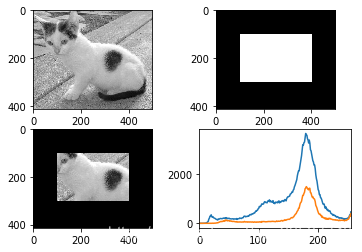
plt.subplot(222), plt.imshow(mask, 'gray')

plt.subplot(223), plt.imshow(masked\_img, 'gray')

plt.subplot(224), plt.plot(hist\_full), plt.plot(hist\_mask)

plt.xlim([0, 256])

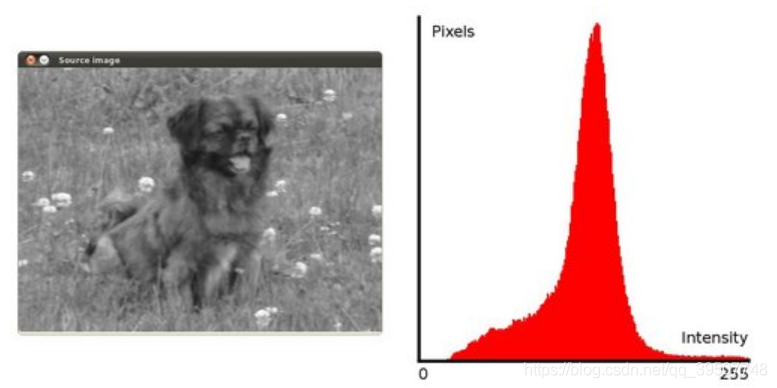
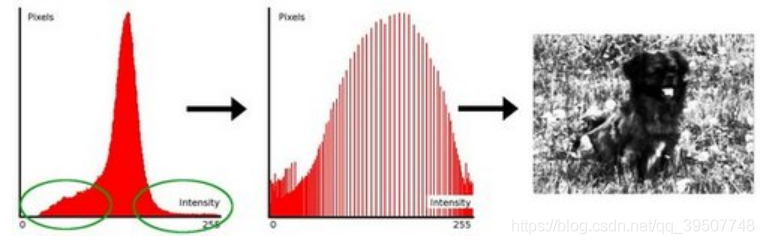
plt.show()



**二、图像直方图均衡化的函数：cv2.equalizeHist()和cv2.createCLAHA()**

**直方图均衡化的介绍**  
直方图均衡化是一种简单有效的图像增强技术，通过改变图像的直方图来改变图像中各像素的灰度，主要用于增强动态范围偏小的图像的对比度。原始图像由于其灰度分布可能集中在较窄的区间，造成图像不够清晰。例如，过曝光图像的灰度级集中在高亮度范围内，而曝光不足将使图像灰度级集中在低亮度范围内。采用直方图均衡化，可以把原始图像的直方图变换为均匀分布（均衡）的形式，这样就增加了像素之间灰度值差别的动态范围，从而达到增强图像整体对比度的效果。换言之，直方图均衡化的基本原理是：对在图像中像素个数多的灰度值（即对画面起主要作用的灰度值）进行展宽，而对像素个数少的灰度值（即对画面不起主要作用的灰度值）进行归并，从而增大对比度，使图像清晰，达到增强的目的。

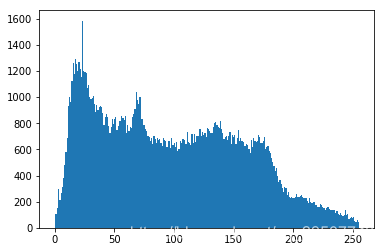
cv2.equalizeHist(img)

参数说明：img表示输入的图片  
  


img = cv2.imread('clahe.jpg',0) #0表示灰度图

plt.hist(img.ravel(),256);

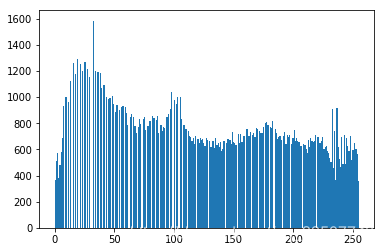
plt.show()



equ = cv2.equalizeHist(img)

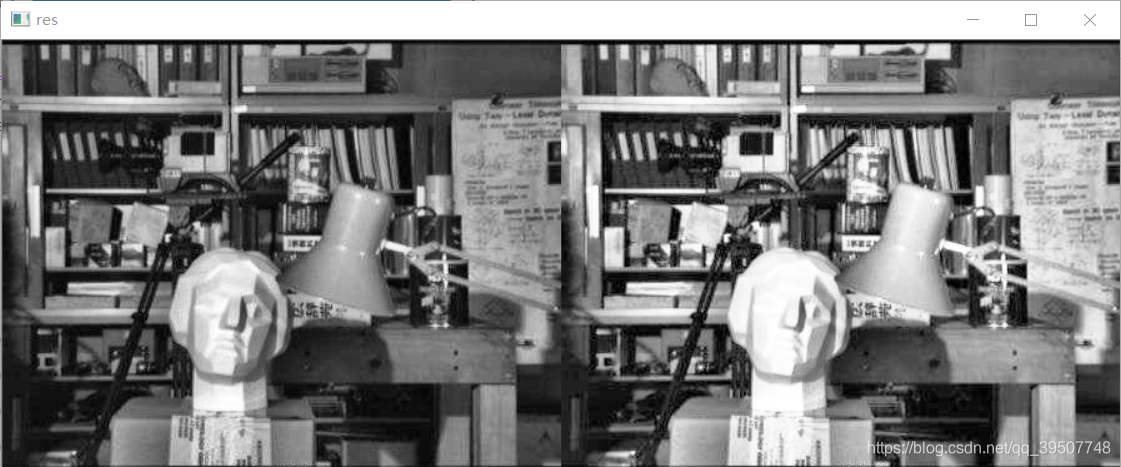
plt.hist(equ.ravel(),256)

plt.show()



res = np.hstack((img,equ))

cv\_show(res,'res')

这种全局的均衡化也会存在一些问题，由于整体亮度的提升，会使得局部图像的细节变得模糊;因此我们可以使用效果更好的自适应均衡化

cv2.createCLAHA(clipLimit=8.0, titleGridSize=(8, 8))

参数说明：clipLimit颜色对比度的阈值， titleGridSize进行像素均衡化的网格大小，即在多少网格下进行直方图的均衡化操作

clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))

res\_clahe = clahe.apply(img)

res = np.hstack((img,equ,res\_clahe))

cv\_show(res,'res')

可以看出自适应均衡化没有使得人物脸部的细节消失