###### 图像的二值化之python+opencv

2015年10月15日 22:05:36

阅读数：27094

定义：图像的二值化，就是将图像上的像素点的灰度值设置为0或255，也就是将整个图像呈现出明显的只有黑和白的视觉效果。

      一幅图像包括目标物体、背景还有噪声，要想从多值的数字图像中直接提取出目标物体，常用的方法就是设定一个阈值T，用T将图像的数据分成两部分：大于T的像素群和小于T的像素群。这是研究灰度变换的最特殊的方法，称为图像的二值化（Binarization）。

****简单的阈值-（全局阈值）****：

Python-OpenCV中提供了阈值（threshold）函数：

****cv2.threshold（）****

函数：第一个参数      ****src****            指原图像，原图像应该是灰度图。

          第二个参数       ****x****              指用来对像素值进行分类的阈值。

          第三个参数        ****y****              指当像素值高于（有时是小于）阈值时应该被赋予的新的像素值

          第四个参数     ****Methods****指，不同的不同的阈值方法，这些方法包括：

                                                                                                                •****cv2.THRESH\_BINARY             图（1）****

                                                                                                                •****cv2.THRESH\_BINARY\_INV     图（2）****

                                                                                                                •****cv2.THRESH\_TRUNC              图（3）****

                                                                                                                •****cv2.THRESH\_TOZERO            图（4）****

                                                                                                                •****cv2.THRESH\_TOZERO\_INV     图（5）****

****IMG_256****

破折线为将被阈值化的值；虚线为阈值 

****IMG_257****

**图（1）**

大于阈值的像素点的灰度值设定为最大值(如8位灰度值最大为255)，灰度值小于阈值的像素点的灰度值设定为0。

**IMG_258**

****图（2）****

大于阈值的像素点的灰度值设定为0，而小于该阈值的设定为255。

**IMG_259**

**图（3）**

像素点的灰度值小于阈值不改变，大于阈值的灰度值的像素点就设定为该阈值。

**IMG_260**

**图（4）**

像素点的灰度值小于该阈值的不进行任何改变，而大于该阈值的部分，其灰度值全部变为0****。****

**IMG_261**

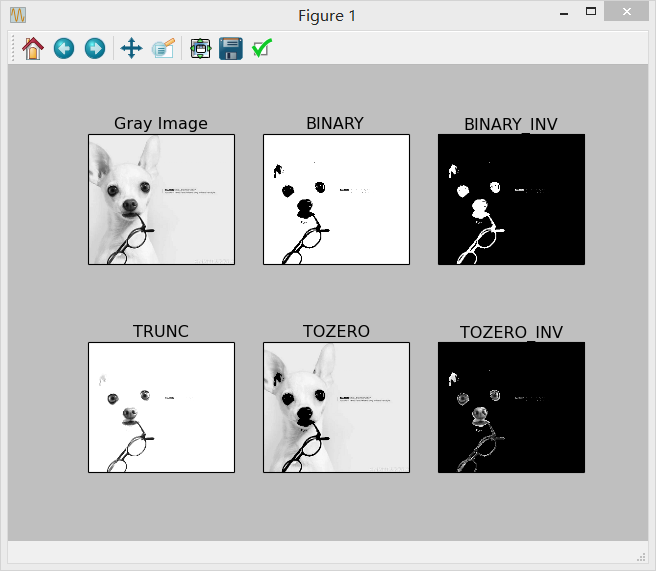
**图（5）**

像素点的灰度值大于该阈值的不进行任何改变，像素点的灰度值小于该阈值的，其灰度值全部变为0。

Python+opencv代码：

**[python]** [view plain](https://blog.csdn.net/what_lei/article/details/49159655" \o "view plain) [copy](https://blog.csdn.net/what_lei/article/details/49159655" \o "copy)

1. **import** cv2
2. **import** numpy as np
3. **from** matplotlib **import** pyplot as plt
4. img=cv2.imread('1.bmp')
5. GrayImage=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)
6. ret,thresh1=cv2.threshold(GrayImage,127,255,cv2.THRESH\_BINARY)
7. ret,thresh2=cv2.threshold(GrayImage,127,255,cv2.THRESH\_BINARY\_INV)
8. ret,thresh3=cv2.threshold(GrayImage,127,255,cv2.THRESH\_TRUNC)
9. ret,thresh4=cv2.threshold(GrayImage,127,255,cv2.THRESH\_TOZERO)
10. ret,thresh5=cv2.threshold(GrayImage,127,255,cv2.THRESH\_TOZERO\_INV)
11. titles = ['Gray Image','BINARY','BINARY\_INV','TRUNC','TOZERO','TOZERO\_INV']
12. images = [GrayImage, thresh1, thresh2, thresh3, thresh4, thresh5]
13. **for** i **in** xrange(6):
14. plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
15. plt.title(titles[i])
16. plt.xticks([]),plt.yticks([])
17. plt.show()

效果图：  


自适应阈值：

      当同一幅图像上的不同部分的具有不同亮度时。这种情况下我们需要采用自适应阈值。此时的阈值是根据图像上的每一个小区域计算与其对应的阈值。因此在同一幅图像上的不同区域采用的是不同的阈值，从而使我们能在亮度不同的情况下得到更好的结果。

****cv2.adaptiveThreshold（）****

函数：第一个参数          ****src****                指原图像，原图像应该是灰度图。

          第二个参数           ****x****                  指当像素值高于（有时是小于）阈值时应该被赋予的新的像素值

          第三个参数  ****adaptive\_method  指：****CV\_ADAPTIVE\_THRESH\_MEAN\_C 或 CV\_ADAPTIVE\_THRESH\_GAUSSIAN\_C

第四个参数****threshold\_type    指****取阈值类型：必须是下者之一                                                                                                                                                                                                                                                               ****•** **CV\_THRESH\_BINARY,****

****• CV\_THRESH\_BINARY\_INV****

           第五个参数    ****block\_size           指****用来计算阈值的象素邻域大小: 3, 5, 7, ...

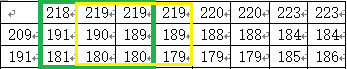
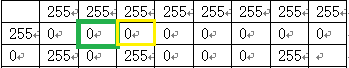
           第六个参数          ****param1           指****与方法有关的参数。对方法CV\_ADAPTIVE\_THRESH\_MEAN\_C 和 CV\_ADAPTIVE\_THRESH\_GAUSSIAN\_C， 它是一个从均值或加权均值提取的常数, 尽管它可以是负数。

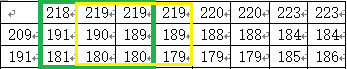
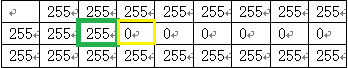
****自适应阈值：****  对方法CV\_ADAPTIVE\_THRESH\_MEAN\_C，先求出块中的均值，再减掉param1。

                            对方法 CV\_ADAPTIVE\_THRESH\_GAUSSIAN\_C ，先求出块中的加权和(gaussian)， 再减掉param1。

****例如：****

          采用方法 CV\_ADAPTIVE\_THRESH\_MEAN\_C，阈值类型：CV\_THRESH\_BINARY,  阈值的象素邻域大小 ****block\_size 选取3，**参数param1  取3和5时：**

IMG_264

                            部分原图像像素值                                                                                              当参数param1为5时  
  IMG_267

                           部分原图像像素值                                                                                                当参数param1为7时

选取对应领域（3\*3）求其均值，然后减去参数param1的值为自适应阈值。测试时求得均值为小数时，貌似进行四舍五入之后再减去参数param1。（****可能是我没测试准确，无聊时测试一下，共同学习****）

Python+opencv代码：

**[python]** [view plain](https://blog.csdn.net/what_lei/article/details/49159655" \o "view plain) [copy](https://blog.csdn.net/what_lei/article/details/49159655" \o "copy)

1. **import** cv2
2. **import** numpy as np
3. **from** matplotlib **import** pyplot as plt
4. img = cv2.imread('1.bmp')
5. GrayImage=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)
6. # 中值滤波
7. GrayImage= cv2.medianBlur(GrayImage,5)
8. ret,th1 = cv2.threshold(GrayImage,127,255,cv2.THRESH\_BINARY)
9. #3 为Block size, 5为param1值
10. th2 = cv2.adaptiveThreshold(GrayImage,255,cv2.ADAPTIVE\_THRESH\_MEAN\_C,\
11. cv2.THRESH\_BINARY,3,5)
12. th3 = cv2.adaptiveThreshold(GrayImage,255,cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,\
13. cv2.THRESH\_BINARY,3,5)
14. titles = ['Gray Image', 'Global Thresholding (v = 127)',
15. 'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']
16. images = [GrayImage, th1, th2, th3]
17. **for** i **in** xrange(4):
18. plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
19. plt.title(titles[i])
20. plt.xticks([]),plt.yticks([])
21. plt.show()

效果图：

