学习笔记3-PIL-2017-07-21-22

# **Python Imaging Library (PIL)**

PIL（Python Imaging Library）是 Python 中最常用的图像处理库。

from PIL import Image, ImageDraw

## Image类

Image类是 PIL 库中一个非常重要的类，通过这个类来创建实例可以有直接载入图像文件，读取处理过的图像和通过抓取的方法得到的图像这三种方法。

**加载图片**

im = Image.open("j.jpg")

print im.format, im.size, im.mode

# JPEG (440, 330) RGB

● format : 识别图像的源格式，如果该文件不是从文件中读取的，则被置为 None 值。

● size : 返回的一个元组，有两个元素，其值为像素意义上的宽和高。

● mode : RGB（true color image），此外还有，L（luminance），CMTK（pre-press image）。

最基本的方式：im = Image.open("filename")

类文件读取：fp = open("filename", "rb"); im = Image.open(fp)

字符串数据读取：import StringIO; im = Image.open(StringIO.StringIO(buffer))

从归档文件读取：import TarIO; fp = TarIo.TarIO("Image.tar", "Image/test/lena.ppm"); im = Image.open(fp)

**展示图片**

im.show()

**一些函数**

crop() : 从图像中提取出某个矩形大小的图像。它接收一个四元素的元组作为参数，各元素为（left, upper, right, lower），坐标系统的原点（0, 0）是左上角。

box = (100, 100, 200, 200)

region = im.crop(box)

**简单的几何变换**

out = im.resize((128, 128))

out = im.rotate(45) #逆时针旋转 45 度角。

out = im.transpose(Image.FLIP\_LEFT\_RIGHT) #左右对换。

out = im.transpose(Image.FLIP\_TOP\_BOTTOM) #上下对换。

out = im.transpose(Image.ROTATE\_90) #旋转 90 度角。

out = im.transpose(Image.ROTATE\_180) #旋转 180 度角。

out = im.transpose(Image.ROTATE\_270) #旋转 270 度角。

## Image和numpy中的数组array相互转换

**1.PIL image转换成array**

img = np.asarray(image)

需要注意的是，如果出现read-only错误，并不是转换的错误，一般是你读取的图片的时候，默认选择的是"r","rb"模式有关。修正的办法:　手动修改图片的读取状态

img.flags.writeable = True # 将数组改为读写模式

**2.array转换成image**

Image.fromarray(np.uint8(img))

## 图像自动阈值分割

图像阈值分割是一种广泛应用的分割技术，利用图像中要提取的目标区域与其背景在灰度特性上的差异，把图像看作具有不同灰度级的两类区域(目标区域和背景区域)的组合，选取一个比较合理的阈值，以确定图像中每个像素点应该属于目标区域还是背景区域，从而产生相应的二值图像。在skimage库中，阈值分割的功能是放在filters模块中。我们可以手动指定一个阈值，从而来实现分割。也可以让系统自动生成一个阈值，下面几种方法就是用来自动生成阈值。

1. **threshold\_otsu**

基于Otsu的阈值分割方法，函数调用格式：

skimage.filters.threshold\_otsu(image, nbins=256) image是指灰度图像返回一个阈值。

机器视觉领域许多算法都要求先对图像进行二值化。这种二值化操作阈值的选取非常重要。阈值选取的不合适，可能得到的结果就毫无用处。今天就来讲讲一种自动计算阈值的方法。这种方法被称之为Otsu法。发明人是个日本人，叫做Nobuyuki Otsu (大津展之）。

简单的说，这种算法假设一副图像由前景色和背景色组成，通过统计学的方法来选取一个阈值，使得这个阈值可以将前景色和背景色尽可能的分开。或者更准确的说是在某种判据下最优。与数理统计领域的 fisher 线性判别算法其实是等价的。

otsu算法中这个判据就是最大类间方差 (intra-class variance or the variance within the class)。

更多参考<http://blog.csdn.net/liyuanbhu/article/details/49387483>

例子：

from skimage import data,filters

import matplotlib.pyplot as plt

image = data.camera()

thresh = filters.threshold\_otsu(image) #返回一个阈值

dst =(image <= thresh)\*1.0 #根据阈值进行分割

plt.figure('thresh',figsize=(8,8))

plt.subplot(121)

plt.title('original image')

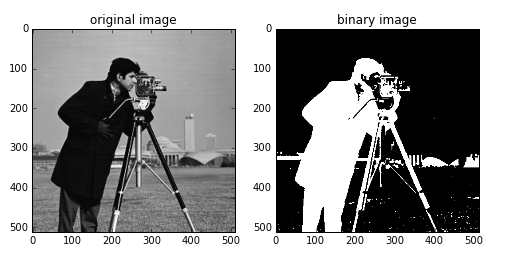
plt.imshow(image,plt.cm.gray)

plt.subplot(122)

plt.title('binary image')

plt.imshow(dst,plt.cm.gray)

plt.show()



1. **threshold\_yen**

thresh = filters.threshold\_yen(image)

1. **threshold\_li**

thresh = filters.threshold\_li(image)

1. **threshold\_isodata**

阈值计算方法：

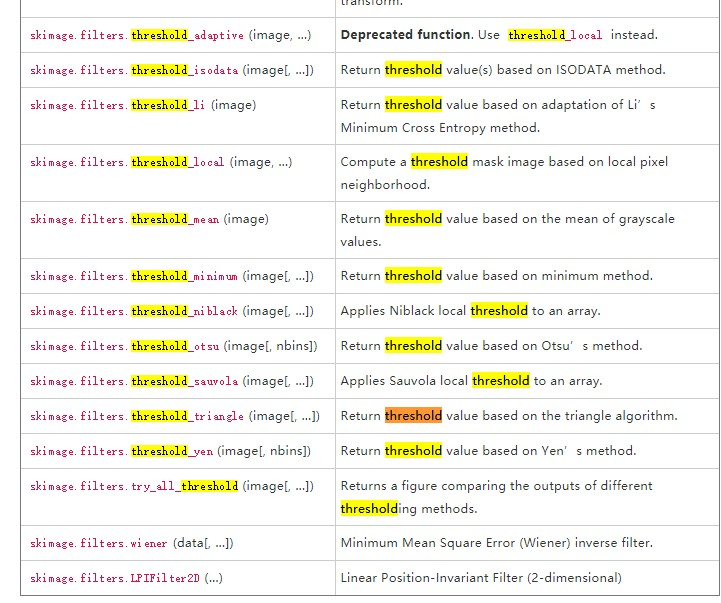
threshold = (image[image <= threshold].mean() +image[image > threshold].mean()) / 2.0 thresh = filters.threshold\_isodata(image)

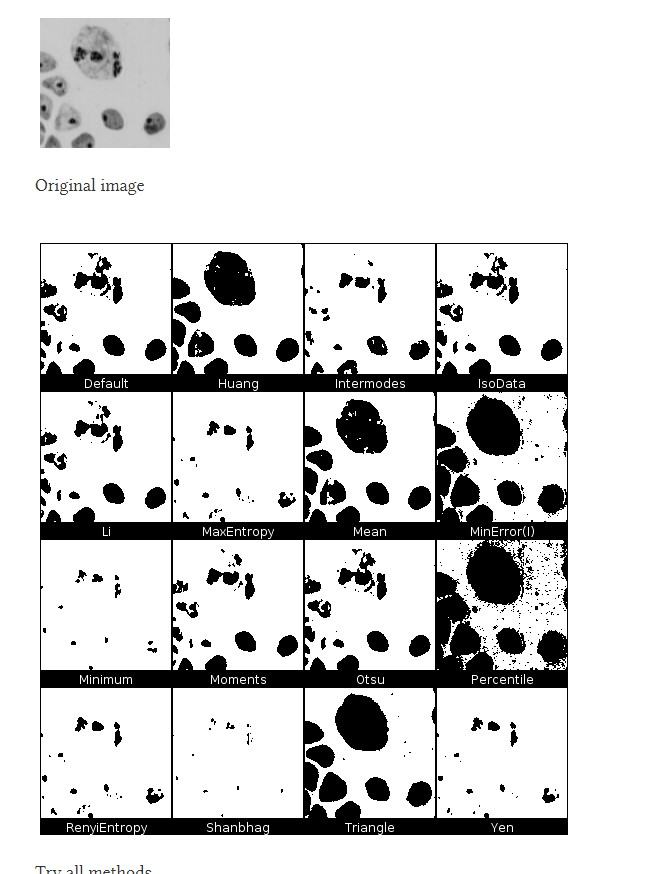
1. **threshold\_adaptive**

调用函数为：skimage.filters.threshold\_adaptive(image, block\_size, method='gaussian'）

block\_size: 块大小，指当前像素的相邻区域大小，一般是奇数（如3，5，7。。。）

method: 用来确定自适应阈值的方法，有'mean', 'generic', 'gaussian' 和 'median'。省略时默认为gaussian.该函数直接访问一个阈值后的图像，而不是阈值。





## Reference:

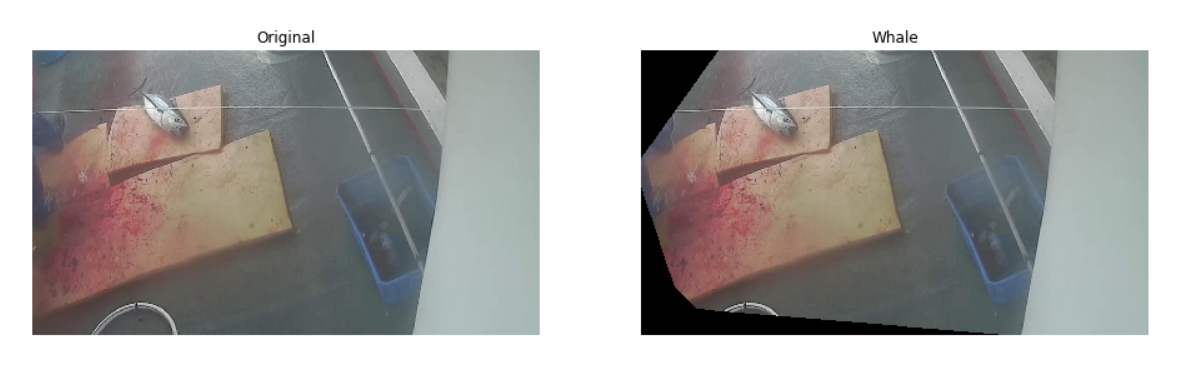
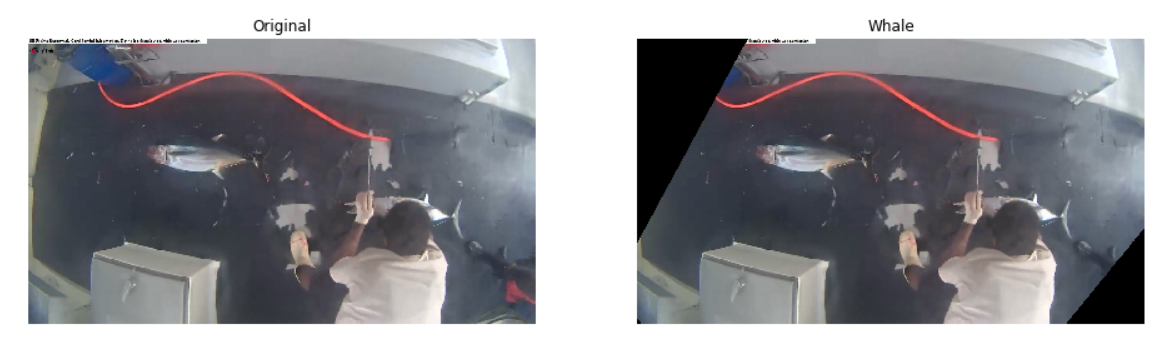
[1] <http://www.cnblogs.com/way_testlife/archive/2011/04/17/2019013.html>

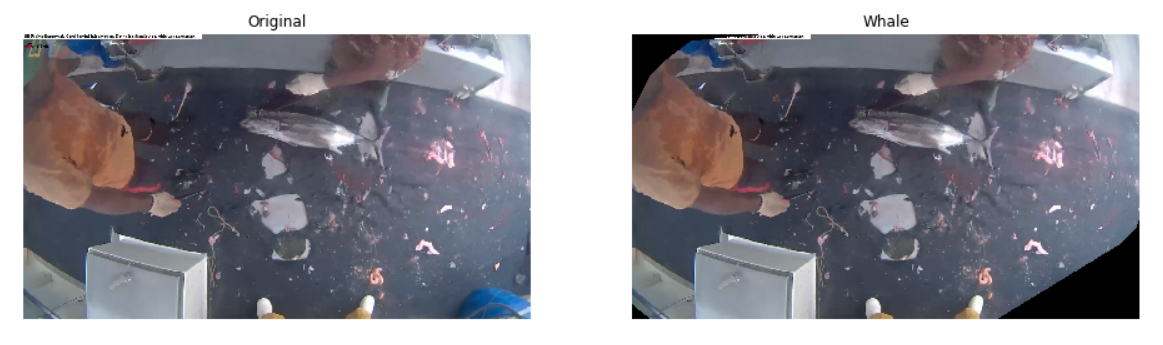
[2] <http://www.cnblogs.com/gongxijun/p/6114232.html>

[3] <http://www.cnblogs.com/denny402/p/5131004.html>

[4] <http://blog.csdn.net/liyuanbhu/article/details/49387483>

## threshold\_mean





## threshold\_minimum

