该系列文章是讲解Python OpenCV图像处理知识，前期主要讲解图像入门、OpenCV基础用法，中期讲解图像处理的各种算法，包括图像锐化算子、图像增强技术、图像分割等，后期结合深度学习研究图像识别、图像分类应用。希望文章对您有所帮助，如果有不足之处，还请海涵~

该系列在github所有源代码：https://github.com/eastmountyxz/ImageProcessing-Python
PS：请求帮忙点个Star，哈哈，第一次使用Github，以后会分享更多代码，一起加油。

同时推荐作者的C++图像系列知识：
[数字图像处理] 一.MFC详解显示BMP格式图片
[数字图像处理] 二.MFC单文档分割窗口显示图片
[数字图像处理] 三.MFC实现图像灰度、采样和量化功能详解
[数字图像处理] 四.MFC对话框绘制灰度直方图
[数字图像处理] 五.MFC图像点运算之灰度线性变化、灰度非线性变化、阈值化和均衡化处理详解
[数字图像处理] 六.MFC空间几何变换之图像平移、镜像、旋转、缩放详解
[数字图像处理] 七.MFC图像增强之图像普通平滑、高斯平滑、Laplacian、Sobel、Prewitt锐化详解

本篇文章作为第一篇，将讲解图像处理基础知识和OpenCV入门函数，知识点如下：
1.图像基础知识
2.OpenCV读写图像
3.OpenCV像素处理

PS: 文章也学习了网易云高登教育的知识，推荐大家学习。

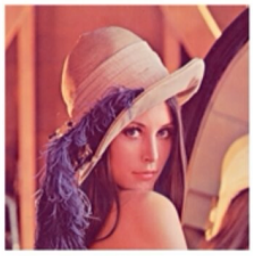
PSS：2019年1~2月作者参加了CSDN2018年博客评选，希望您能投出宝贵的一票。我是59号，Eastmount，杨秀璋。投票地址：
https://bss.csdn.net/m/topic/blog_star2018/index

五年来写了314篇博客，12个专栏，是真的热爱分享，热爱CSDN这个平台，也想帮助更多的人，专栏包括Python、数据挖掘、网络爬虫、图像处理、C#、Android等。现在也当了两年老师，更是觉得有义务教好每一个学生，让贵州学子好好写点代码，学点技术，"师者，传到授业解惑也"，提前祝大家新年快乐。2019我们携手共进，为爱而生。

一.图像基础知识

图像都是由像素(pixel)构成的，即图像中的小方格，这些小方格都有一个明确的位置和被分配的色彩数值，而这些一小方格的颜色和位置就决定该图像所呈现出来的样子。像素是图像中的最小单位，每一个点阵图像包含了一定量的像素，这些像素决定图像在屏幕上所呈现的大小。

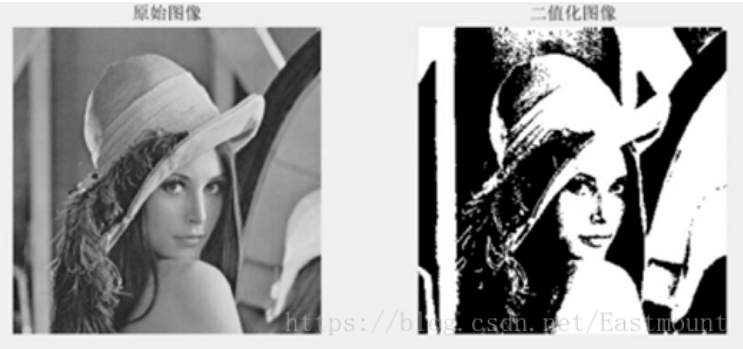
图像通常包括二值图像、灰度图像和彩色图像。



二值图像 灰度图像 RGB图像

1.二值图像

二值图像中任何一个点非黑即白，要么为白色（像素为255），要么为黑色（像素为0）。将灰度图像转换为二值图像的过程，常通过依次遍历判断实现，如果像素>=127则设置为255，否则设置为0。



2.灰度图像

灰度图像除了黑和白，还有灰色，它把灰度划分为256个不同的颜色，图像看着也更为清晰。将彩色图像转换为灰度图是图像处理的最基本预处理操作，通常包括下面几种方法：

(1) 浮点算法：Gray=R*0.3*+*G*0.59+B*0.11*

*(2) 整数方法：Gray=(R*30+G*59+B*11)/100

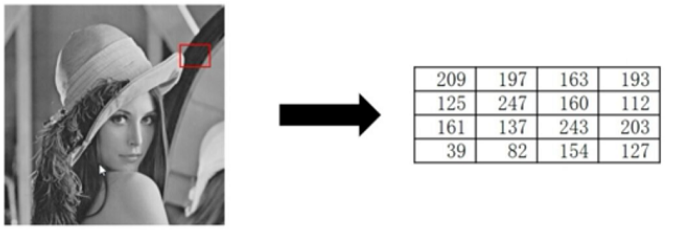
(3) 移位方法：Gray=(R*28*+*G*151+B*77)>>8;*

(4) 平均值法：Gray=（R+G+B）/3;(此程序采用算法)

(5) 仅取绿色：Gray=G；

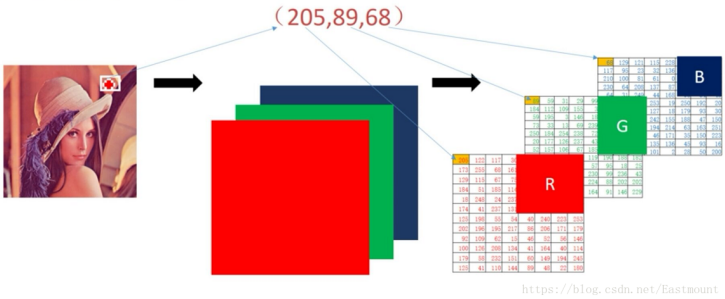
*(6) 加权平均值算法：根据光的亮度特性，公式: R=G=B=R*0.299+G*0.587+B0.144

通过上述任一种方法求得Gray后，将原来的RGB(R,G,B)中的R,G,B统一用Gray替换，形成新的颜色RGB(Gray,Gray,Gray)，用它替换原来的RGB(R,G,B)就是灰度图了。改变象素矩阵的RGB值，来达到彩色图转变为灰度图。



3.彩色图像

彩色图像是RGB图像，RGB表示红、绿、蓝三原色，计算机里所有颜色都是三原色不同比例组成的，即三色通道。



二.OpenCV读写图像

本文主要使用Python2.7和OpenCV进行讲解，首先调用"pip install opencv-python"安装OpenCV库，如下图所示：

1.读入图像

OpenCV读图像主要调用下面函数实现：

> img = cv2.imread(文件名,[,参数])
> 参数(1) cv2.IMREAD_UNCHANGED (图像不可变)
> 参数(2) cv2.IMREAD_GRAYSCALE (灰度图像)
> 参数(3) cv2.IMREAD_COLOR (读入彩色图像)
> 参数(4) cv2.COLOR_BGR2RGB (图像通道BGR转成RGB)

2.显示图像

显示图像调用函数如下：

> cv2.imshow(窗口名, 图像名)

3.窗口等待

调用函数如下：

> cv2.waitKey(delay)
> 键盘绑定函数，共一个参数，表示等待毫秒数，将等待特定的几毫秒，看键盘是否有输入，返回值为ASCII值。如果其参数为0，则表示无限期的等待键盘输入；参数>0表示等待delay毫秒；参数<0表示等待键盘单击。

4.删除所有窗口

调用函数如下：

> cv2.destroyAllWindows() 删除所有窗口
> cv2.destroyWindows() 删除指定的窗口

5.写入图片

调用函数如下：

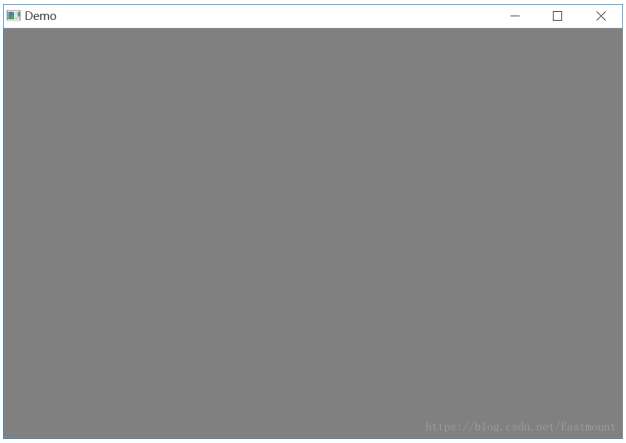
> retval = cv2.imwrite(文件地址, 文件名)

下面代码是读入图片并显示保存。

```
# -*- coding:utf-8 -*-
import cv2

#读取图片
img = cv2.imread("test.jpg")

#显示图像
cv2.imshow("Demo", img)

#等待显示
cv2.waitKey(0)
cv2.destroyAllWindows()

#写入图像
cv2.imwrite("testyxz.jpg", img)
```

输出结果如下图所示，并且在文件夹下保存了一张名为“testyxz.jpg”的图像。

如果代码中没有watiKey(0)函数，则运行结果如下图所示：



同时
可以对代码进行升级，如下所示：

```
#无限期等待输入
k=cv2.waitKey(0)
#如果输入ESC退出
if k==27:
    cv2.destroyAllWindows()
```

三.OpenCV像素处理

1.读取像素

灰度图像直接返回灰度值，彩色图像则返回B、G、R三个分量。注意OpenCV读取图像是BGR存储显示，需要转换为RGB再进行图像处理。

> 灰度图像：返回值 = 图像(位置参数)
> eg: test=img[88,42]
> 彩色图像：返回值 = 图像[位置元素, 0 | 1 | 2] 获取BGR三个通道像素
> eg: blue=img[88,142,0] green=img[88,142,1] red=img[88,142,2]

2.修改图像

修改图像如果是灰度图像则直接赋值新像素即可，彩色图像依次给三个值赋值即可。

> 灰度图像：
> img[88,142] = 255
> 彩色图像：
> img[88,142, 0] = 255
> img[88,142, 1] = 255
> img[88,142, 2] = 255
> 彩色图像：方法二
> img[88,142] = [255, 255, 255]

下面代码是获取像素及修改的操作。

```
# -*- coding:utf-8 -*-
import cv2

#读取图片
img = cv2.imread("test.jpg", cv2.IMREAD_UNCHANGED)
test = img[88,142]
print test
img[88,142] = [255, 255, 255]
print test

#分别获取BGR通道像素
blue = img[88,142,0]
print blue
green = img[88,142,1]
print green
red = img[88,142,2]
print red

#显示图像
cv2.imshow("Demo", img)

#等待显示
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
#写入图像
cv2.imwrite("testyxz.jpg", img)
```

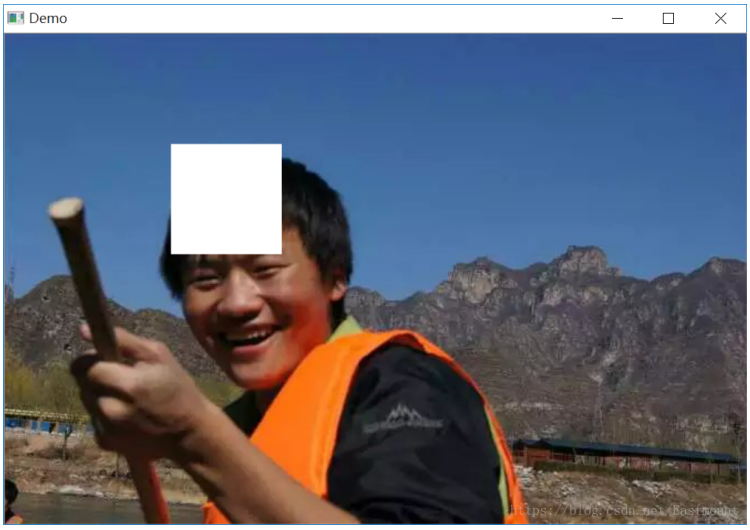
输出结果如下所示：

[158 107 64]
[255 255 255]
255
255
255

下面代码是将行为100到200、列150到250的像素区域设置为白色。

```
# -*- coding:utf-8 -*-
import cv2

#读取图片
img = cv2.imread("test.jpg", cv2.IMREAD_UNCHANGED)

#该区域设置为白色
img[100:200, 150:250] = [255,255,255]

#显示图像
cv2.imshow("Demo", img)

#等待显示
cv2.waitKey(0)
cv2.destroyAllWindows()

#写入图像
cv2.imwrite("testyxz.jpg", img)
```

运行结果如下图所示：



希望文章对大家有所帮助，如果有错误或不足之处，还请海涵。

（By：Eastmount 2018-08-16 夜11点 https://blog.csdn.net/Eastmount/）