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

前文参考：
[Python图像处理] 一.图像处理基础知识及OpenCV入门函数
[Python图像处理] 二.OpenCV+Numpy库读取与修改像素
[Python图像处理] 三.获取图像属性、兴趣ROI区域及通道处理
[Python图像处理] 四.图像平滑之均值滤波、方框滤波、高斯滤波及中值滤波
[Python图像处理] 五.图像融合、加法运算及图像类型转换
[Python图像处理] 六.图像缩放、图像旋转、图像翻转与图像平移
[Python图像处理] 七.图像阈值化处理及算法对比
[Python图像处理] 八.图像腐蚀与图像膨胀
[Python图像处理] 九.形态学之图像开运算、闭运算、梯度运算
[Python图像处理] 十.形态学之图像顶帽运算和黑帽运算
[Python图像处理] 十一.灰度直方图概念及OpenCV绘制直方图
[Python图像处理] 十二.图像几何变换之图像仿射变换、图像透视变换和图像校正
[Python图像处理] 十三.基于灰度三维图的图像顶帽运算和黑帽运算

本篇文章讲解图像灰度化处理的知识，结合OpenCV调用cv2.cvtColor()函数实现图像灰度操作，使用像素处理方法对图像进行灰度化处理。基础性知识希望对您有所帮助。

1.图像灰度化原理
2.基于OpenCV的图像灰度化处理
3.基于像素操作的图像灰度化处理

PS：文章参考自己以前系列图像处理文章及OpenCV库函数，同时，本篇文章涉及到《计算机图形学》基础知识，请大家下来补充。

参考文献：
杨秀璋等. 基于苗族服饰的图像锐化和边缘提取技术研究[J]. 现代计算机，2018(10).
《数字图像处理》（第3版），冈萨雷斯著，阮秋琦译，电子工业出版社，2013年.
《数字图像处理学》（第3版），阮秋琦，电子工业出版社，2008年，北京.
《OpenCV3编程入门》，毛星云，冷雪飞，电子工业出版社，2015.
Opencv学习（十六）之颜色空间转换cvtColor()
python+opencv+图像特效（图像灰度处理、颜色翻转、图片融合，边缘检测，浮雕效果，颜色映射）

一.图像灰度化原理

像灰度化是将一幅彩色图像转换为灰度化图像的过程。彩色图像通常包括R、G、B三个分量，分别显示出红绿蓝等各种颜色，灰度化就是使彩色图像的R、G、B三个分量相等的过程。灰度图像中每个像素仅具有一种样本颜色，其灰度是位于黑色与白色之间的多级色彩深度，灰度值大的像素点比较亮，反之比较暗，像素值最大为255（表示白色），像素值最小为0（表示黑色）。

假设某点的颜色由RGB(R,G,B)组成，常见灰度处理算法如表7.1所示：

表 7.1 灰度处理算法

算法名称	算法公式
最大值灰度处理	Gray=max(R, G, B)
浮点灰度处理	Gray=R*0.3+G*0.59+B*0.11
整数灰度处理	Gray=(R*30+G*59+B*11)/100
移位灰度处理	Gray=(R*28+G*151+B*77)>>8
平均灰度处理	Gray=(R+G+B)/3
加权平均灰度处理	Gray=R*0.299+G*0.587+B*0.144

表7.1中Gray表示灰度处理之后的颜色，然后将原始RGB(R,G,B)颜色均匀地替换成新颜色RGB(Gray,Gray,Gray)，从而将彩色图片转化为灰度图像。

一种常见的方法是将RGB三个分量求和再取平均值，但更为准确的方法是设置不同的权重，将RGB分量按不同的比例进行灰度划分。比如人类的眼睛感官蓝色的敏感度最低，敏感最高的是绿色，因此将RGB按照0.299、0.587、0.144比例加权平均能得到较合理的灰度图像，如公式7.1所示。

$$Gray = R*0.299 + G*0.587 + B*0.114$$

二.基于OpenCV的图像灰度化处理

在日常生活中，我们看到的大多数彩色图像都是RGB类型，但是在图像处理过程中，常常需要用到灰度图像、二值图像、HSV、HSI等颜色，OpenCV提供了cvtColor()函数实现这些功能。其函数原型如下所示：

dst = cv2.cvtColor(src, code[, dst[, dstCn]])

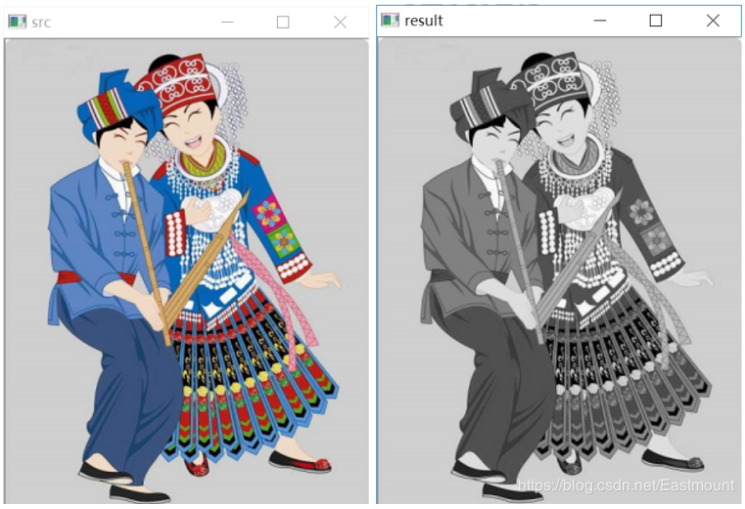
- src表示输入图像，需要进行颜色空间变换的原图像
- dst表示输出图像，其大小和深度与src一致
- code表示转换的代码或标识
- dstCn表示目标图像通道数，其值为0时，则有src和code决定

该函数的作用是将一个图像从一个颜色空间转换到另一个颜色空间，其中，RGB是指Red、Green和Blue，一副图像由这三个通道（channel）构成；Gray表示只有灰度值一个通道；HSV包含Hue（色调）、Saturation（饱和度）和Value（亮度）三个通道。在OpenCV中，常见的颜色空间转换标识包括CV_BGR2BGRA、CV_RGB2GRAY、CV_GRAY2RGB、CV_BGR2HSV、CV_BGR2XYZ、CV_BGR2HLS等。

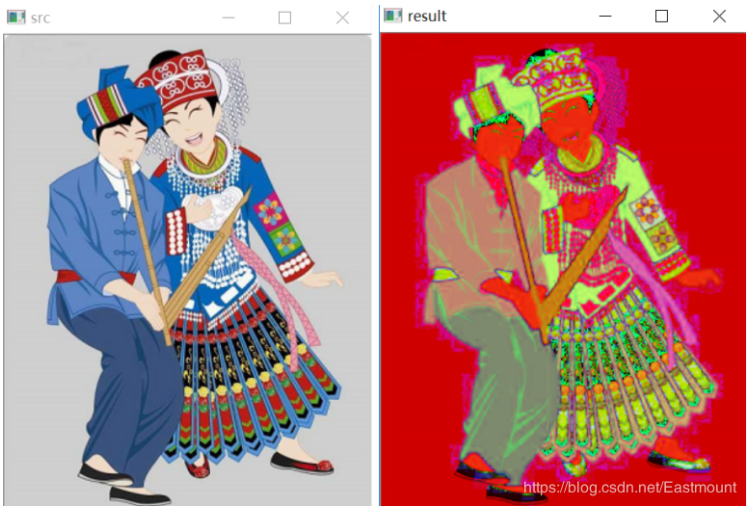
下面是调用cvtColor()函数将图像进行灰度化处理的代码。

```
#encoding:utf-8
import cv2
import numpy as np

#读取原始图片
src = cv2.imread('miao.png')

#图像灰度化处理
grayImage = cv2.cvtColor(src,cv2.COLOR_BGR2GRAY)

#显示图像
cv2.imshow("src", src)
cv2.imshow("result", grayImage)

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

输出结果如下图所示，左边是彩色的苗族服饰原图，右边是将彩色图像进行灰度化处理之后的灰度图。其中，灰度图将一个像素点的三个颜色变量设置为相当，R=G=B，此时该值称为灰度值。



同样，可以调用 grayImage = cv2.cvtColor(src, cv2.COLOR_BGR2HSV) 核心代码将彩色图像转换为HSV颜色空间，如下图所示。

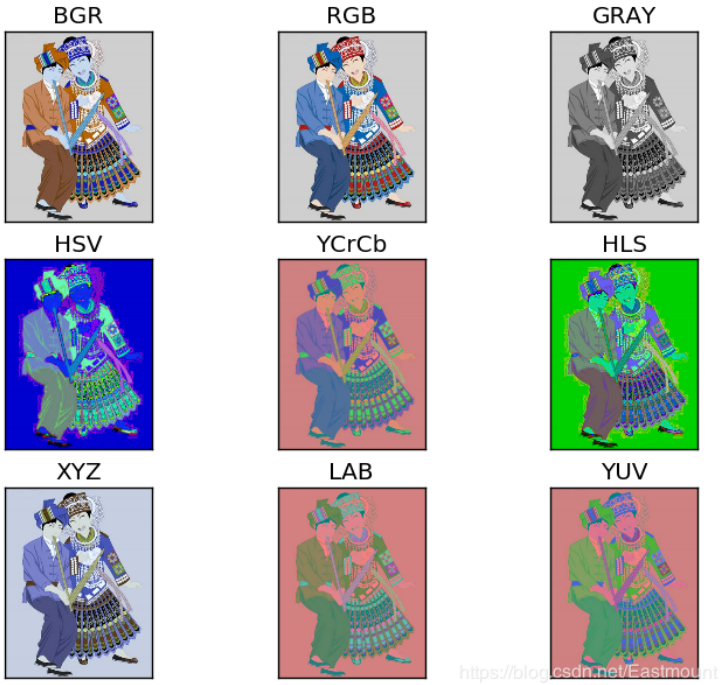


下面Image_Processing_07_02.py代码对比了九种常见的颜色空间，包括BGR、RGB、GRAY、HSV、YCrCb、HLS、XYZ、LAB和YUV，并循环显示处理后的图像。

```
#encoding:utf-8
import cv2
import numpy as np
import matplotlib.pyplot as plt

#读取原始图像
img_BGR = cv2.imread('miao.png')
```

```
#BGR转换为RGB
img_RGB = cv2.cvtColor(img_BGR, cv2.COLOR_BGR2RGB)

#灰度化处理
img_GRAY = cv2.cvtColor(img_BGR, cv2.COLOR_BGR2GRAY)

#BGR转HSV
img_HSV = cv2.cvtColor(img_BGR, cv2.COLOR_BGR2HSV)

#BGR转YCrCb
img_YCrCb = cv2.cvtColor(img_BGR, cv2.COLOR_BGR2YCrCb)

#BGR转HLS
img_HLS = cv2.cvtColor(img_BGR, cv2.COLOR_BGR2HLS)

#BGR转XYZ
img_XYZ = cv2.cvtColor(img_BGR, cv2.COLOR_BGR2XYZ)

#BGR转LAB
img_LAB = cv2.cvtColor(img_BGR, cv2.COLOR_BGR2LAB)

#BGR转YUV
img_YUV = cv2.cvtColor(img_BGR, cv2.COLOR_BGR2YUV)

#调用matplotlib显示处理结果
titles = ['BGR', 'RGB', 'GRAY', 'HSV', 'YCrCb', 'HLS', 'XYZ', 'LAB', 'YU
images = [img_BGR, img_RGB, img_GRAY, img_HSV, img_YCrCb,
          img_HLS, img_XYZ, img_LAB, img_YUV]
for i in xrange(9):
   plt.subplot(3, 3, i+1), plt.imshow(images[i], 'gray')
   plt.title(titles[i])
   plt.xticks([]),plt.yticks([])
plt.show()
```

其运行结果如图所示：



三.基于像素操作的图像灰度化处理

前面讲述了调用OpenCV中cvtColor()函数实现图像灰度化的处理，接下来讲解基于像素操作的图像灰度化处理方法，主要是最大值灰度处理、平均灰度处理和加权平均灰度处理方法。

1.最大值灰度处理方法

该方法的灰度值等于彩色图像R、G、B三个分量中的最大值，公式如下：

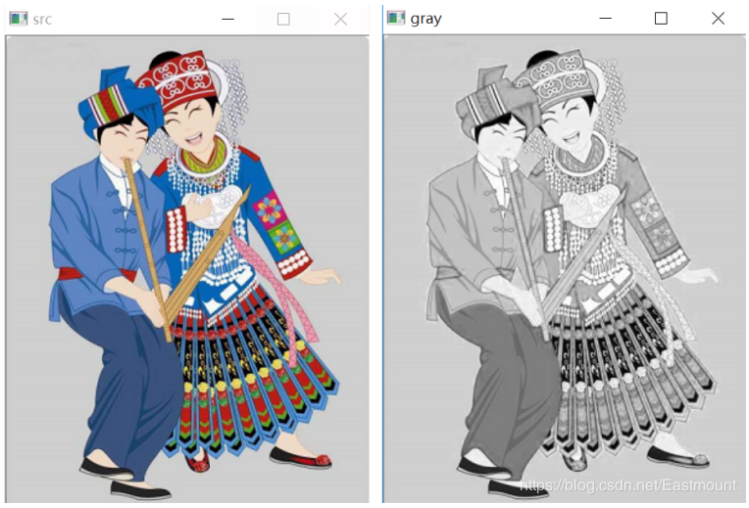
$$\mathrm{gray}(i,j) = \max(R(i,j), G(i,j), B(i,j))$$

其方法灰度化处理后的灰度图亮度很高，实现代码如下。

```
#encoding:utf-8
import cv2
import numpy as np
import matplotlib.pyplot as plt

#读取原始图像
img = cv2.imread('miao.png')

#获取图像高度和宽度
height = img.shape[0]
```

```
width = img.shape[1]

#创建一幅图像
grayimg = np.zeros((height, width, 3), np.uint8)

#图像最大值灰度处理
for i in range(height):
    for j in range(width):
        #获取图像R G B最大值
        gray = max(img[i,j][0], img[i,j][1], img[i,j][2])
        #灰度图像素赋值 gray=max(R,G,B)
        grayimg[i,j] = np.uint8(gray)

#显示图像
cv2.imshow("src", img)
cv2.imshow("gray", grayimg)

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

其输出结果如下图所示，其处理效果的灰度偏亮。



2.平均灰度处理方法

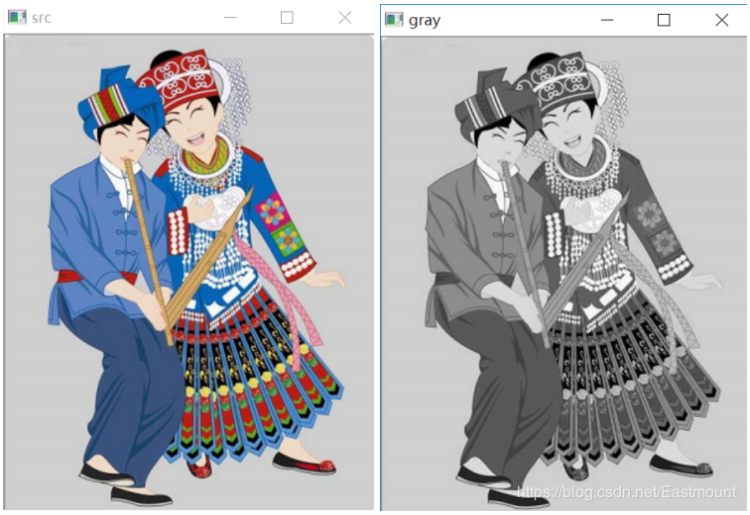
该方法的灰度值等于彩色图像R、G、B三个分量灰度值的求和平均值，其计算公式如下所示：

$$gray(i,j)=\frac{R(i,j)+G(i,j)+B(i,j)}{3}$$

平均灰度处理方法实现代码如下所示：

```
#encoding:utf-8
import cv2
import numpy as np
import matplotlib.pyplot as plt

#读取原始图像
img = cv2.imread('miao.png')

#获取图像高度和宽度
height = img.shape[0]
width = img.shape[1]

#创建一幅图像
grayimg = np.zeros((height, width, 3), np.uint8)
print grayimg

#图像平均灰度处理方法
for i in range(height):
    for j in range(width):
        #灰度值为RGB三个分量的平均值
        gray = (int(img[i,j][0]) + int(img[i,j][1]) + int(img[i,j][2]))
        grayimg[i,j] = np.uint8(gray)

#显示图像
cv2.imshow("src", img)
cv2.imshow("gray", grayimg)

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

其输出结果如下图所示：



3.加权平均灰度处理方法

该方法根据色彩重要性，将三个分量以不同的权值进行加权平均。由于人眼对绿色的敏感最高，对蓝色敏感最低，因此，按下式对RGB三分量进行加权平均能得到较合理的灰度图像。

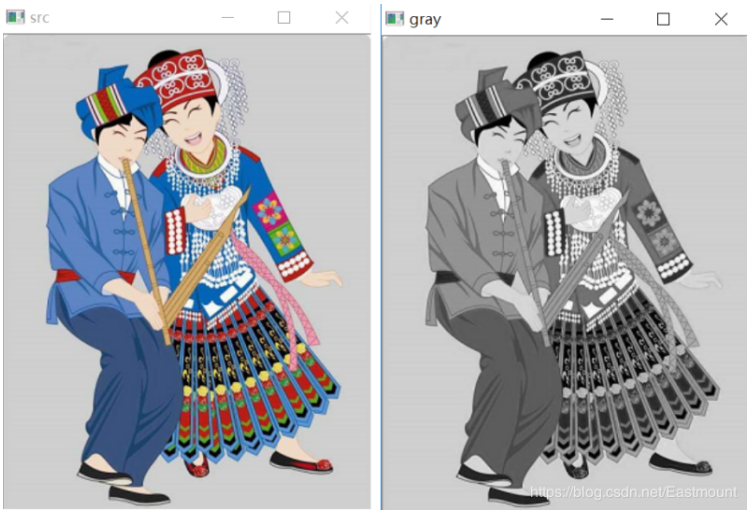
$$\text{gray}(i, j) = 0.30 \times \text{R}(i, j) + 0.59 \times \text{G}(i, j) + 0.11 \times \text{B}(i, j)$$

加权平均灰度处理方法实现代码如下所示：

```
#encoding:utf-8
import cv2
import numpy as np
import matplotlib.pyplot as plt

#读取原始图像
img = cv2.imread('miao.png')

#获取图像高度和宽度
height = img.shape[0]
width = img.shape[1]

#创建一幅图像
grayimg = np.zeros((height, width, 3), np.uint8)
print grayimg

#图像平均灰度处理方法
for i in range(height):
    for j in range(width):
        #灰度加权平均法
        gray = 0.30 * img[i,j][0] + 0.59 * img[i,j][1] + 0.11 * img[i,j]
```

```
            grayimg[i,j] = np.uint8(gray)

    #显示图像
    cv2.imshow("src", img)
    cv2.imshow("gray", grayimg)

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

其输出结果如下图所示：



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。最近连续奔波考博，经历的事情太多，有喜有悲，需要改变自己好好对女神，也希望读者与我一起加油。
（By：Eastmount 2019-03-25 早上8点 https://blog.csdn.net/Eastmount/）