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。

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
[Python图像处理] 十四.基于OpenCV和像素处理的图像灰度化处理
[Python图像处理] 十五.图像的灰度线性变换
[Python图像处理] 十六.图像的灰度非线性变换之对数变换、伽马变换
[Python图像处理] 十七.图像锐化与边缘检测之Roberts算子、Prewitt算子、Sobel算子和Laplacian算子
[Python图像处理] 十八.图像锐化与边缘检测之Scharr算子、Canny算子和LOG算子
[Python图像处理] 十九.图像分割之基于K-Means聚类的区域分割
[Python图像处理] 二十.图像量化处理和采样处理及局部马赛克特效
[Python图像处理] 二十一.图像金字塔之图像向下取样和向上取样
[Python图像处理] 二十二.Python图像傅里叶变换原理及实现
[Python图像处理] 二十三.傅里叶变换之高通滤波和低通滤波
[Python图像处理] 二十四.图像特效处理之毛玻璃、浮雕和油漆特效

前面一篇文章我讲解了常见的图像特效处理——毛玻璃、浮雕和油漆特效，本篇文章继续分享素描特效、怀旧特效、光照特效、流年特效以及滤镜特效。代码通过Python和OpenCV实现，基础性文章，希望对你有所帮助。同时，该部分知识均为杨秀璋查阅资料撰写，转载请署名CSDN+杨秀璋及原地址出处，谢谢！！

PS：文章参考自己以前系列图像处理文章及OpenCV库函数，同时参考如下文献：
《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，北京.
Eastmount - [Android] 通过Menu实现图片怀旧、浮雕、模糊、光照和素描效果
使用python和opencv将图片转化为素描图-python代码解析
謝灰灰在找胡蘿蔔. IOS开发 - - 使用lookup table为图片添加滤镜
百度百科. 滤镜

一.图像素描特效

图像素描特效会将图像的边界都凸显出来，通过边缘检测及阈值化处理能实现该功能。一幅图像的内部都具有相似性，而在图像边界处具有明显的差异，边缘检测利用数学中的求导来扩大这种变化。但是求导过程中会增大图像的噪声，所以边缘检测之前引入了高斯滤波降噪处理。本文的图像素描特效主要经过以下几个步骤：

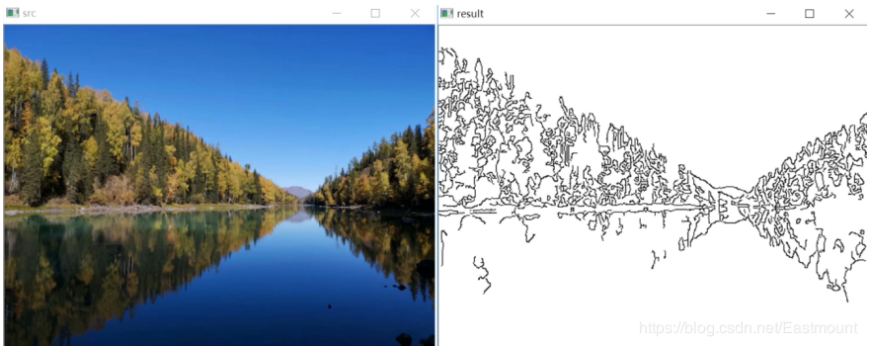
- 调用cv2.cvtColor()函数将彩色图像灰度化处理；
- 通过cv2.GaussianBlur()函数实现高斯滤波降噪；
- 边缘检测采用Canny算子实现；
- 最后通过cv2.threshold()反二进制阈值化处理实现素描特效。

其运行代码如下所示。

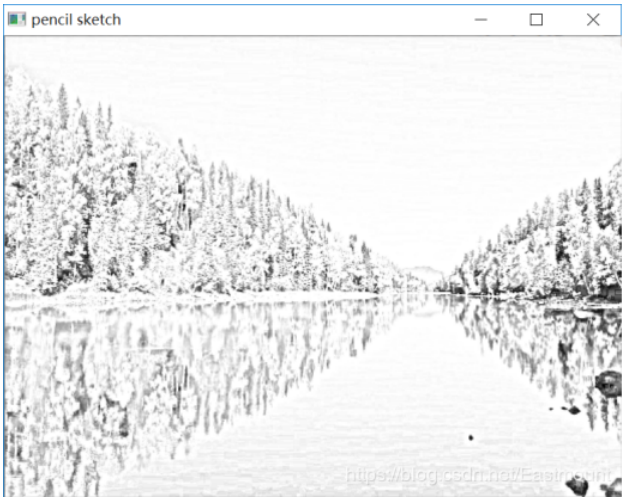
```
#coding:utf-8
import cv2
import numpy as np

#读取原始图像
img = cv2.imread('scenery.png')

#图像灰度处理
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

```
#高斯滤波降噪
gaussian = cv2.GaussianBlur(gray, (5,5), 0)

#Canny算子
canny = cv2.Canny(gaussian, 50, 150)

#阈值化处理
ret, result = cv2.threshold(canny, 100, 255, cv2.THRESH_BINARY_INV)

#显示图像
cv2.imshow('src', img)
cv2.imshow('result', result)
cv2.waitKey()
cv2.destroyAllWindows()
```

最终输出结果如下图所示，它将彩色图像素描处理。原图是作者去年九月份拍摄于喀纳斯，真的很美~

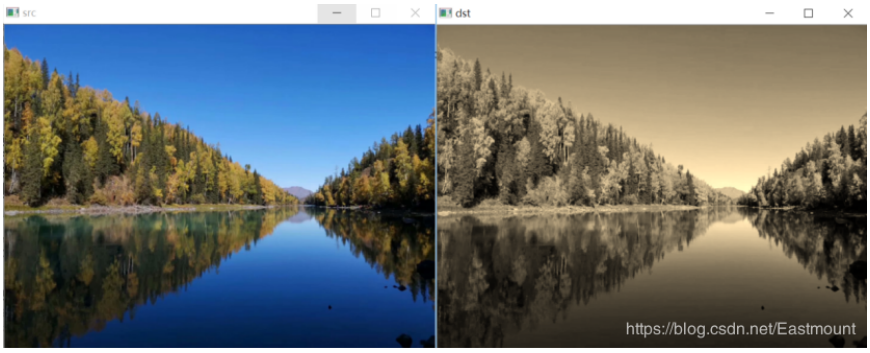


图像的素描特效有很多种方法，本文仅提供了一种方法，主要提取的是图像的边缘轮廓，还有很多更精细的素描特效方法，提取的轮廓更为清晰，如下图所示。希望读者能自行扩展相关算法知识，并实现对应的效果。



二.图像怀旧特效

图像怀旧特效是指图像经历岁月的昏暗效果，如图所示，左边“src”为原始图像，右边“dst”为怀旧特效图像。



怀旧特效是将图像的RGB三个分量分别按照一定比例进行处理的结果，其怀旧公式如下所示：

$$R = 0.393*r + 0.769*g + 0.189*b$$
$$G = 0.349*r + 0.686*g + 0.168*b$$
$$B = 0.272*r + 0.534*g + 0.131*b$$

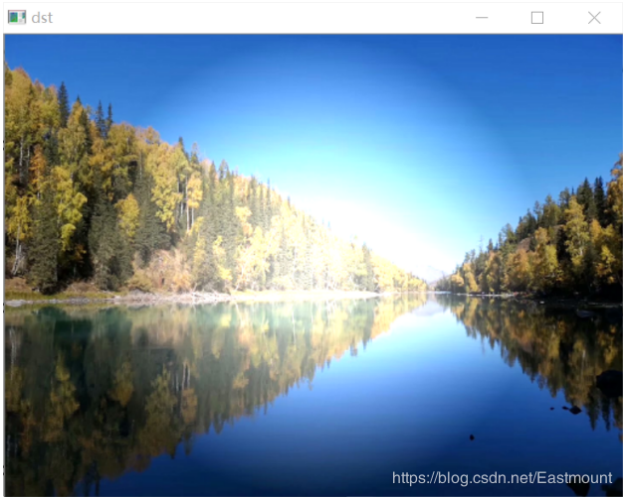
Python实现代码主要通过双层循环遍历图像的各像素点，再结合该公式计算各颜色通道的像素值，最终生成如图所示的效果，其完整代码如下。

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

```
import numpy as np

#读取原始图像
img = cv2.imread('nana.png')

#获取图像行和列
rows, cols = img.shape[:2]

#新建目标图像
dst = np.zeros((rows, cols, 3), dtype="uint8")

#图像怀旧特效
for i in range(rows):
    for j in range(cols):
        B = 0.272*img[i,j][2] + 0.534*img[i,j][1] + 0.131*img[i,j][0]
        G = 0.349*img[i,j][2] + 0.686*img[i,j][1] + 0.168*img[i,j][0]
        R = 0.393*img[i,j][2] + 0.769*img[i,j][1] + 0.189*img[i,j][0]
        if B>255:
            B = 255
        if G>255:
            G = 255
        if R>255:
            R = 255
        dst[i,j] = np.uint8((B, G, R))

#显示图像
cv2.imshow('src', img)
cv2.imshow('dst', dst)
cv2.waitKey()
cv2.destroyAllWindows()
```

三.图像光照特效

图像光照特效是指图像存在一个类似于灯光的光晕特效，图像像素值围绕光照中心点呈圆形范围内的增强。如下图所示，该图像的中心点为（192，192），光照特效之后中心圆范围内的像素增强了200。



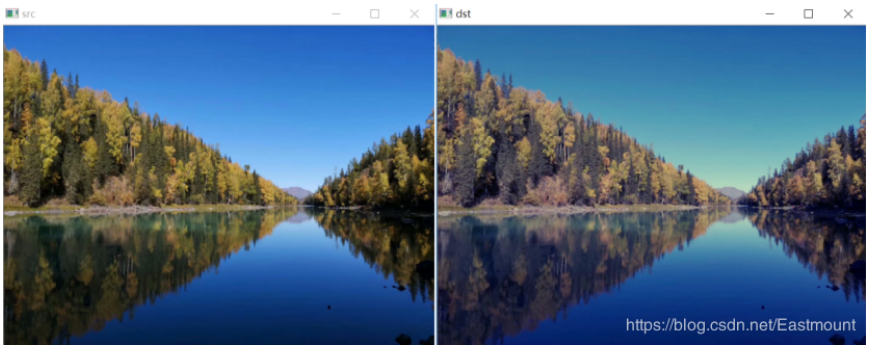
Python实现代码主要是通过双层循环遍历图像的各像素点，寻找图像的中心点，再通过计算当前点到光照中心的距离（平面坐标系中两点之间的距离），判断该距离与图像中心圆半径的大小关系，中心圆范围内的图像灰度值增强，范围外的图像灰度值保留，并结合边界范围判断生成最终的光照效果。

```
#coding:utf-8
import cv2
import math
import numpy as np

#读取原始图像
img = cv2.imread('scenery.png')

#获取图像行和列
rows, cols = img.shape[:2]

#设置中心点
centerX = rows / 2
centerY = cols / 2
print centerX, centerY
radius = min(centerX, centerY)
print radius

#设置光照强度
strength = 200

#新建目标图像
dst = np.zeros((rows, cols, 3), dtype="uint8")

#图像光照特效
for i in range(rows):
    for j in range(cols):
```

```
            #计算当前点到光照中心距离(平面坐标系中两点之间的距离)
            distance = math.pow((centerY-j), 2) + math.pow((centerX-i), 2)
            #获取原始图像
            B =  img[i,j][0]
            G =  img[i,j][1]
            R = img[i,j][2]
            if (distance < radius * radius):
                #按照距离大小计算增强的光照值
                result = (int)(strength*( 1.0 - math.sqrt(distance) / radius
                B = img[i,j][0] + result
                G = img[i,j][1] + result
                R = img[i,j][2] + result
                #判断边界 防止越界
                B = min(255, max(0, B))
                G = min(255, max(0, G))
                R = min(255, max(0, R))
                dst[i,j] = np.uint8((B, G, R))
            else:
                dst[i,j] = np.uint8((B, G, R))

    #显示图像
    cv2.imshow('src', img)
    cv2.imshow('dst', dst)
    cv2.waitKey()
    cv2.destroyAllWindows()
```

四.图像流年特效

流年是用来形容如水般流逝的光阴或年华，图像处理中特指将原图像转换为具有时代感或岁月沉淀的特效，其效果如图所示。



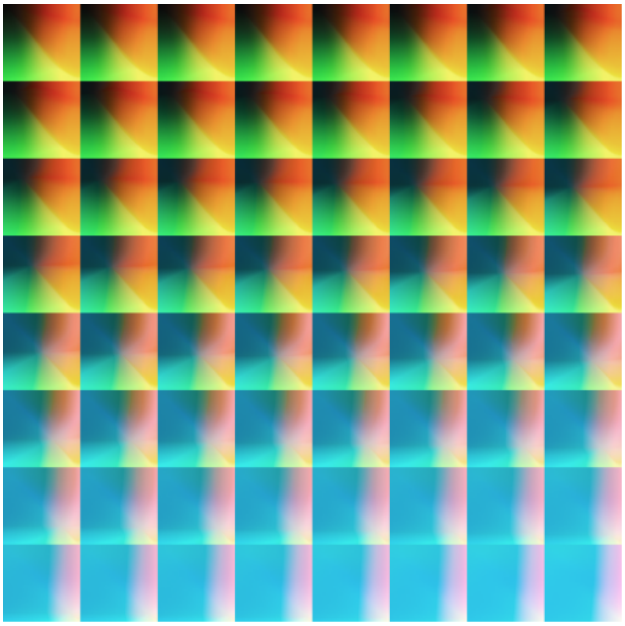
Python实现代码如下，它将原始图像的蓝色（B）通道的像素值开根号，再乘以一个权重参数，产生最终的流年效果。

```
#coding:utf-8
import cv2
import math
import numpy as np

#读取原始图像
img = cv2.imread('scenery.png')

#获取图像行和列
rows, cols = img.shape[:2]

#新建目标图像
dst = np.zeros((rows, cols, 3), dtype="uint8")

#图像流年特效
for i in range(rows):
    for j in range(cols):
        #B通道的数值开平方乘以参数12
        B = math.sqrt(img[i,j][0]) * 12
        G =  img[i,j][1]
        R =  img[i,j][2]
        if B>255:
            B = 255
        dst[i,j] = np.uint8((B, G, R))

#显示图像
cv2.imshow('src', img)
cv2.imshow('dst', dst)
cv2.waitKey()
cv2.destroyAllWindows()
```

五.图像滤镜特效

滤镜主要是用来实现图像的各种特殊效果，它在Photoshop中具有非常神奇的作用。滤镜通常需要同通道、图层等联合使用，才能取得最佳艺术效果。本小节将讲述一种基于颜色查找表（Look up Table）的滤镜处理方法，它通过将每一个原始颜色进行转换之后得到新的颜色。比如，原始图像的某像素点为红色（R-255, G-0, B-0），进行转换之后变为绿色（R-0, G-255, B-0），之后所有是红色的地方都会被自动转换为绿色，而颜色查找表就是将所有的颜色进行一次（矩阵）转换，很多的滤镜功能就是提供了这么一个转换的矩阵，在原始色彩的基础上进行颜色的转换。

假设现在存在一张新的滤镜颜色查找表，如图所示，它是一张512×512大小，包含各像素颜色分布的图像。下面这张图片另存为本地，即可直接用于图像滤镜处理。

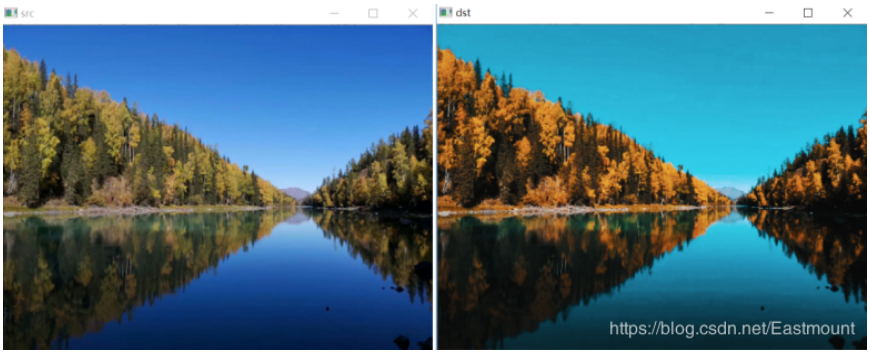


滤镜特效实现的Python代码如下所示，它通过自定义getBRG()函数获取颜色查找表中映射的滤镜颜色，再依次循环替换各颜色。

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

#获取滤镜颜色
def getBGR(img, table, i, j):
    #获取图像颜色
    b, g, r = img[i][j]
    #计算标准颜色表中颜色的位置坐标
    x = int(g/4 + int(b/32) * 64)
    y = int(r/4 + int((b%32) / 4) * 64)
    #返回滤镜颜色表中对应的颜色
    return lj_map[x][y]

#读取原始图像
img = cv2.imread('scenery.png')
lj_map = cv2.imread('table.png')

#获取图像行和列
rows, cols = img.shape[:2]

#新建目标图像
dst = np.zeros((rows, cols, 3), dtype="uint8")

#循环设置滤镜颜色
for i in range(rows):
    for j in range(cols):
```

```
            dst[i][j] = getBGR(img, lj_map, i, j)

    #显示图像
    cv2.imshow('src', img)
    cv2.imshow('dst', dst)

    cv2.waitKey()
    cv2.destroyAllWindows()
```

滤镜特效的运行结果如图所示，其中左边“src”为原始风景图像，右边“dst”为滤镜处理后的图像，其颜色变得更为鲜艳，对比度更强。



六.本文小结

本篇文章主要讲解了图像常见的特效处理，从处理效果图、算法原理、代码实现三个步骤进行详细讲解，涉及图像素描特效、怀旧特效、光照特效、流年特效、图像滤镜等，这些知识点将为读者从事Python图像处理相关项目实践或科学研究提供一定基础。

八年，从100万名挤进2万名，再到如今的top300，挺开心的。喜欢的不是那个数字，而是数字背后近三千天得奋斗史，以及分享知识和帮人解惑所带来的快乐，最近总结了安全系列，且看且珍惜，继续敲代码喽~

（By：Eastmount 2019-08-14 下午3点写于钟书阁 https://blog.csdn.net/Eastmount ）