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
[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

前面一篇文章我讲解了基于K-Means聚类的图像分割或量化处理，但突然发现市场上讲解图像量化和采样代码的文章很缺乏，因此结合2015年自己的一篇 文章 及相关知识，分享一篇Python图像量化及处理的博文供同学们学习。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！！

1.图像量化处理（含原理、操作、聚类量化）
2.图像采样处理（含原理、操作、局部马赛克处理）

PS: 您可能发现作者最近写了很多Python图像处理的文章，新书即将出炉。fighting~

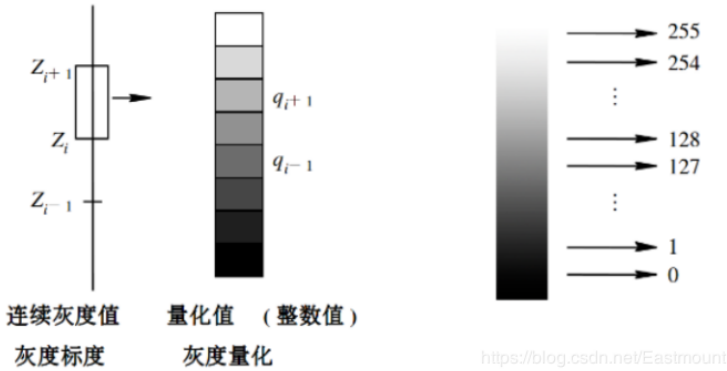
PS：文章参考自己以前系列图像处理文章及OpenCV库函数，同时参考如下文献：
eastmount - [数字图像处理] 三.MFC实现图像灰度、采样和量化功能详解（强推）
《数字图像处理》（第3版），冈萨雷斯著，阮秋琦译，电子工业出版社，2013年.
《数字图像处理学》（第3版），阮秋琦，电子工业出版社，2008年，北京.
yunfung - 数字图像基础之图像取样和量化（Image Sampling and Quantization）
zqhwando - 图像处理中的采样与量化[EB/OL]
师寇_ - Python + opencv 实现图片马赛克

一.图像量化处理

图像通常是自然界景物的客观反映，并以照片形式或视频记录的介质连续保存，获取图像的目标是从感知的数据中产生数字图像，因此需要把连续的图像数据离散化，转换为数字化图像，其工作主要包括两方面——量化和采样。数字化幅度值称为量化，数字化坐标值称为采样。本章主要讲解图像量化和采样处理的概念，并通过Python和OpenCV实现这些功能。

1.1 概述

所谓量化（Quantization），就是将图像像素点对应亮度的连续变化区间转换为单个特定值的过程，即将原始灰度图像的空间坐标幅度值离散化。量化等级越多，图像层次越丰富，灰度分辨率越高，图像的质量也越好；量化等级越少，图像层次欠丰富，灰度分辨率越低，会出现图像轮廓分层的现象，降低了图像的质量。图6-1是将图像的连续灰度值转换为0至255的灰度级的过程。



如果量化等级为2，则将使用两种灰度级表示原始图片的像素（0-255），灰度值小于128的取0，大于等于128的取128；如果量化等级为4，则将使用四种灰度级表示原始图片的像素，新图像将分层为四种颜色，0-64区间取0，64-128区间取64，128-192区间取128，192-255区间取192；依次类推。

图6-2是对比不同量化等级的“Lena”图。其中（a）的量化等级为256，（b）的量化等级为64，（c）的量化等级为16，（d）的量化等级为8，（e）的量化等级为4，（f）的量化等级为2。

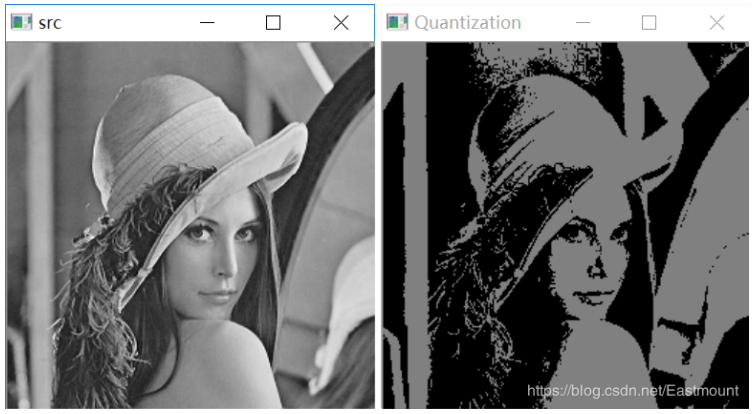
1.2 操作

下面讲述Python图像量化处理相关代码操作。其核心流程是建立一张临时图片，接着循环遍历原始图像中所有像素点，判断每个像素点应该属于的量化等级，最后将临时图像显示。下述代码将灰度图像转换为两种量化等级。

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

#读取原始图像
```

```
img = cv2.imread('lena.png')

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

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

#图像量化操作 量化等级为2
for i in range(height):
    for j in range(width):
        for k in range(3): #对应BGR三分量
            if img[i, j][k] < 128:
                gray = 0
            else:
                gray = 128
            new_img[i, j][k] = np.uint8(gray)

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

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

其输出结果如图6-3所示，它将灰度图像划分为两种量化等级。



下面的代码分别比较了量化等级为2、4、8的量化处理效果。

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

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

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

#创建一幅图像
new_img1 = np.zeros((height, width, 3), np.uint8)
new_img2 = np.zeros((height, width, 3), np.uint8)
new_img3 = np.zeros((height, width, 3), np.uint8)

#图像量化等级为2的量化处理
for i in range(height):
    for j in range(width):
        for k in range(3): #对应BGR三分量
            if img[i, j][k] < 128:
                gray = 0
            else:
                gray = 128
            new_img1[i, j][k] = np.uint8(gray)

#图像量化等级为4的量化处理
for i in range(height):
    for j in range(width):
        for k in range(3): #对应BGR三分量
            if img[i, j][k] < 64:
                gray = 0
            elif img[i, j][k] < 128:
                gray = 64
            elif img[i, j][k] < 192:
                gray = 128
            else:
                gray = 192
            new_img2[i, j][k] = np.uint8(gray)

#图像量化等级为8的量化处理
for i in range(height):
    for j in range(width):
        for k in range(3): #对应BGR三分量
            if img[i, j][k] < 32:
                gray = 0
            elif img[i, j][k] < 64:
                gray = 32
            elif img[i, j][k] < 96:
                gray = 64
```

```
            elif img[i, j][k] < 128:
                gray = 96
            elif img[i, j][k] < 160:
                gray = 128
            elif img[i, j][k] < 192:
                gray = 160
            elif img[i, j][k] < 224:
                gray = 192
            else:
                gray = 224
            new_img3[i, j][k] = np.uint8(gray)

#用来正常显示中文标签
plt.rcParams['font.sans-serif']=['SimHei']

#显示图像
titles = [u'(a) 原始图像', u'(b) 量化-L2', u'(c) 量化-L4', u'(d) 量化-L8']
images = [img, new_img1, new_img2, new_img3]
for i in xrange(4):
   plt.subplot(2,2,i+1), plt.imshow(images[i], 'gray'),
   plt.title(titles[i])
   plt.xticks([]),plt.yticks([])
plt.show()
```

输出结果如图6-4所示，该代码调用matplotlib.pyplot库绘制了四幅图像，其中（a）表示原始图像，（b）表示等级为2的量化处理，（c）表示等级为4的量化处理，（d）表示等级为8的量化处理。

1.3 K-Means聚类量化处理

上一小节的量化处理是通过遍历图像中的所有像素点，进行灰度图像的幅度值离散化处理。本小节补充一个基于K-Means聚类算法的量化处理过程，它能够将彩色图像RGB像素点进行颜色分割和颜色量化。更多知识推荐大家学习前一篇文章。

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

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

#图像二维像素转换为一维
data = img.reshape((-1,3))
data = np.float32(data)

#定义中心 (type,max_iter,epsilon)
criteria = (cv2.TERM_CRITERIA_EPS +
            cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)

#设置标签
flags = cv2.KMEANS_RANDOM_CENTERS

#K-Means聚类 聚集成4类
compactness, labels, centers = cv2.kmeans(data, 4, None, criteria, 10, f


#图像转换回uint8二维类型
centers = np.uint8(centers)
res = centers[labels.flatten()]
dst = res.reshape((img.shape))

#图像转换为RGB显示
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
dst = cv2.cvtColor(dst, cv2.COLOR_BGR2RGB)


#用来正常显示中文标签
plt.rcParams['font.sans-serif']=['SimHei']

#显示图像
titles = [u'原始图像', u'聚类量化 K=4']
images = [img, dst]
for i in xrange(2):
```

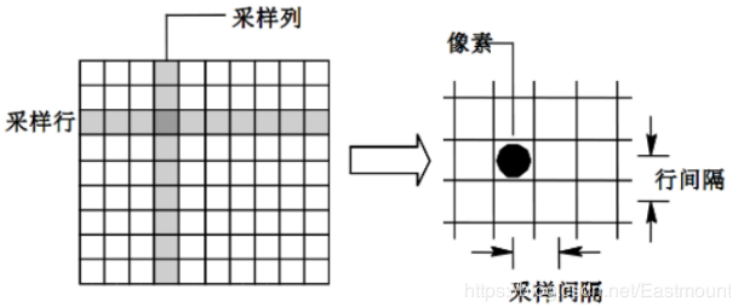
```
    plt.subplot(1,2,i+1), plt.imshow(images[i], 'gray'),
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```

输出结果如图6-4所示，它通过K-Means聚类算法将彩色人物图像的灰度聚集成四种颜色。

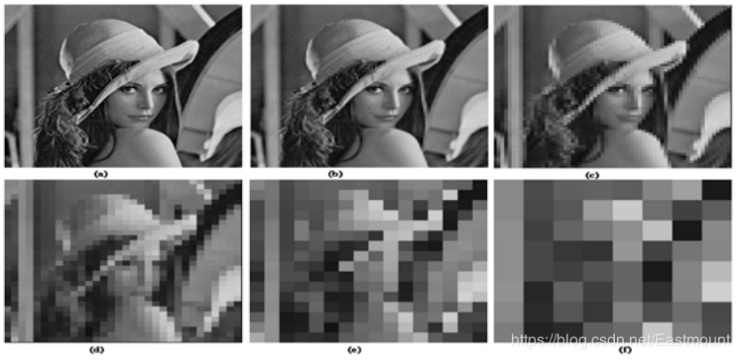
二.图像采样处理

2.1 概述

图像采样（Image Sampling）处理是将一幅连续图像在空间上分割成M×N个网格，每个网格用一个亮度值或灰度值来表示，其示意图如图6-5所示。



图像采样的间隔越大，所得图像像素数越少，空间分辨率越低，图像质量越差，甚至出现马赛克效应；相反，图像采样的间隔越小，所得图像像素数越多，空间分辨率越高，图像质量越好，但数据量会相应的增大。图6-6展示了不同采样间隔的“Lena”图。



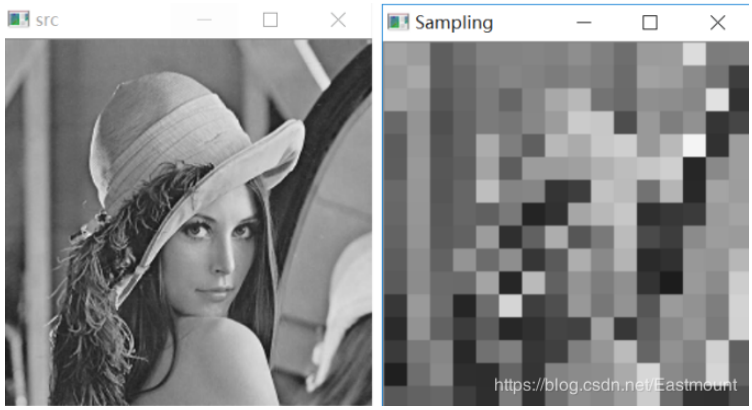
2.2 操作

下面讲述Python图像采样处理相关代码操作。其核心流程是建立一张临时图片，设置需要采样的区域大小（如16×16），接着循环遍历原始图像中所有像素点，采样区域内的像素点赋值相同（如左上角像素点的灰度值），最终实现图像采样处理。代码是进行16×16采样的过程。

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

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

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

#采样转换成16*16区域
numHeight = height/16
numwidth = width/16

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

#图像循环采样16*16区域
for i in range(16):
    #获取Y坐标
    y = i*numHeight
    for j in range(16):
        #获取X坐标
        x = j*numwidth
        #获取填充颜色 左上角像素点
        b = img[y, x][0]
```

```
            g = img[y, x][1]
            r = img[y, x][2]

            #循环设置小区域采样
            for n in range(numHeight):
                for m in range(numwidth):
                    new_img[y+n, x+m][0] = np.uint8(b)
                    new_img[y+n, x+m][1] = np.uint8(g)
                    new_img[y+n, x+m][2] = np.uint8(r)

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

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

输出结果如下图所示：

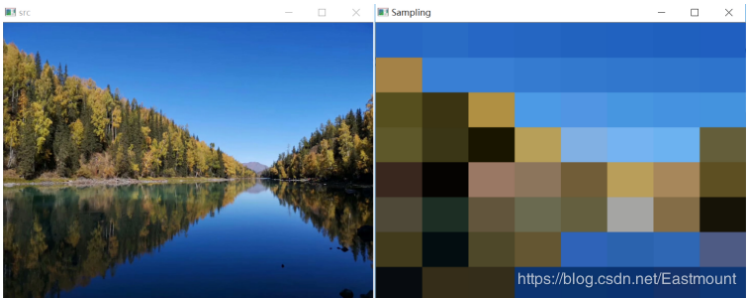


同样，可以对彩色图像进行采样处理，下面的代码将彩色风景图像采样处理成8×8的马赛克区域。

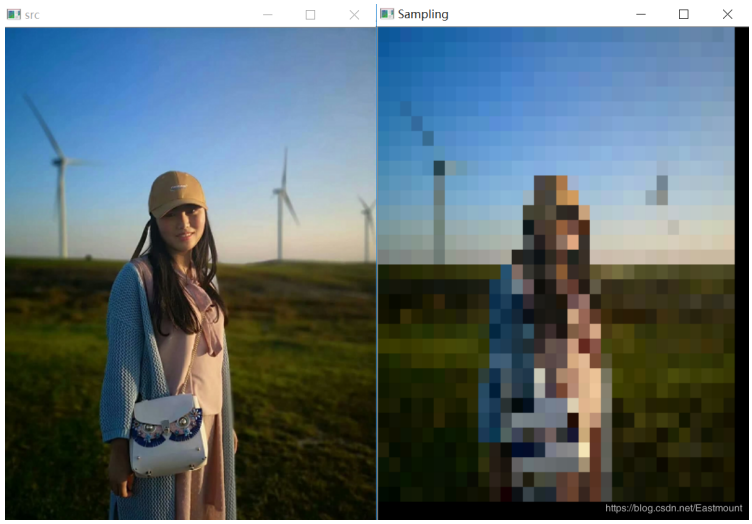
```
# -*- coding: utf-8 -*-
import cv2
import numpy as np
import matplotlib.pyplot as plt

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

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

```
#采样转换成8*8区域
numHeight = height/8
numwidth = width/8

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

#图像循环采样8*8区域
for i in range(8):
    #获取Y坐标
    y = i*numHeight
    for j in range(8):
        #获取X坐标
        x = j*numwidth
        #获取填充颜色 左上角像素点
        b = img[y, x][0]
        g = img[y, x][1]
        r = img[y, x][2]

        #循环设置小区域采样
        for n in range(numHeight):
            for m in range(numwidth):
                new_img[y+n, x+m][0] = np.uint8(b)
                new_img[y+n, x+m][1] = np.uint8(g)
                new_img[y+n, x+m][2] = np.uint8(r)

#显示图像
cv2.imshow("src", img)
cv2.imshow("Sampling", new_img)

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

其输出结果如图所示，它将彩色风景图像采样成8×8的区域。



但上述代码存在一个问题，当图像的长度和宽度不能被采样区域整除时，输出图像的最右边和最下边的区域没有被采样处理。这里推荐读者做个求余运算，将不能整除部门的区域也进行采样处理。



2.3 局部马赛克处理

前面讲述的代码是对整幅图像进行采样处理，那么如何对图像的局部区域进行马赛克处理呢？下面的代码就实现了该功能。当鼠标按下时，它能够给鼠标拖动的区域打上马赛克，并按下“s”键保存图像至本地。

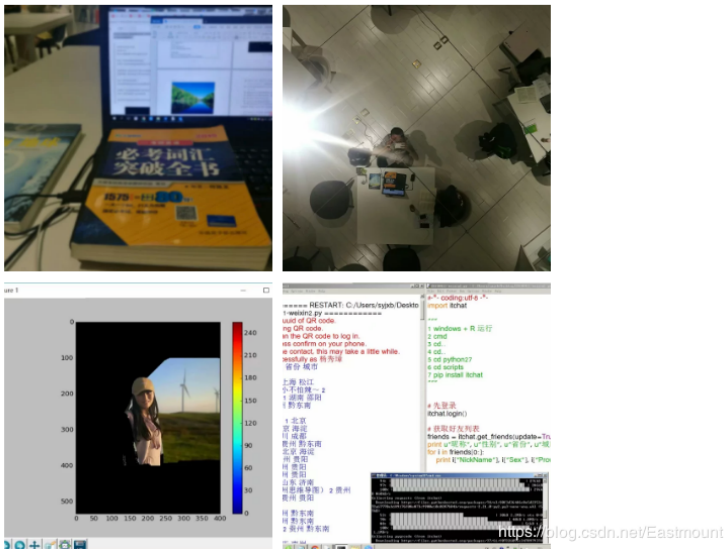
```
# -- coding:utf-8 --
import cv2
import numpy as np
import matplotlib.pyplot as plt

#读取原始图像
im = cv2.imread('people.png', 1)

#设置鼠标左键开启
en = False

#鼠标事件
def draw(event, x, y, flags, param):
    global en
    #鼠标左键按下开启en值
    if event==cv2.EVENT_LBUTTONDOWN:
        en = True
    #鼠标左键按下并且移动
elif event==cv2.EVENT_MOUSEMOVE and
 flags==cv2.EVENT_LBUTTONDOWN:
        #调用函数打马赛克
```

```
            if en:
                drawMask(y,x)
            #鼠标左键弹起结束操作
            elif event==cv2.EVENT_LBUTTONUP:
                en = False

    #图像局部采样操作
    def drawMask(x, y, size=10):
        #size*size采样处理
        m = x / size * size
        n = y / size * size
        print m, n
        #10*10区域设置为同一像素值
        for i in range(size):
            for j in range(size):
                im[m+i][n+j] = im[m][n]

    #打开对话框
    cv2.namedWindow('image')

    #调用draw函数设置鼠标操作
    cv2.setMouseCallback('image', draw)

    #循环处理
    while(1):
        cv2.imshow('image', im)
        #按ESC键退出
        if cv2.waitKey(10)&0xFF==27:
            break
        #按s键保存图片
        elif cv2.waitKey(10)&0xFF==115:
            cv2.imwrite('sava.png', im)

    #退出窗口
    cv2.destroyAllWindows()
```

其输出结果如图所示，它将人物的脸部进行马赛克处理。

希望这篇基础性文章对您有所帮助，如果有错误或不足之处，请海涵！
最近继续备考博士，接下来还有两个学校，一方面耐心等待之前的结果；另一方面继续复习，周末女神陪着来书店看书，岁月静好，砥砺前行！在这期间，自己经历了很多酸甜苦辣的事情，希望陌生的你也学会享受生活，共勉。

Eastmountain 书山有路勤为径，学海无涯苦作舟。下班从花溪回来，书店中继续奋战，几场考试即将来临，加油共勉。下午上课分享了itchat微信操作。



（By：Eastmount 2019-04-13 周六夜8点写于贵阳·钟书阁 https://blog.csdn.net/Eastmount ）