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
[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

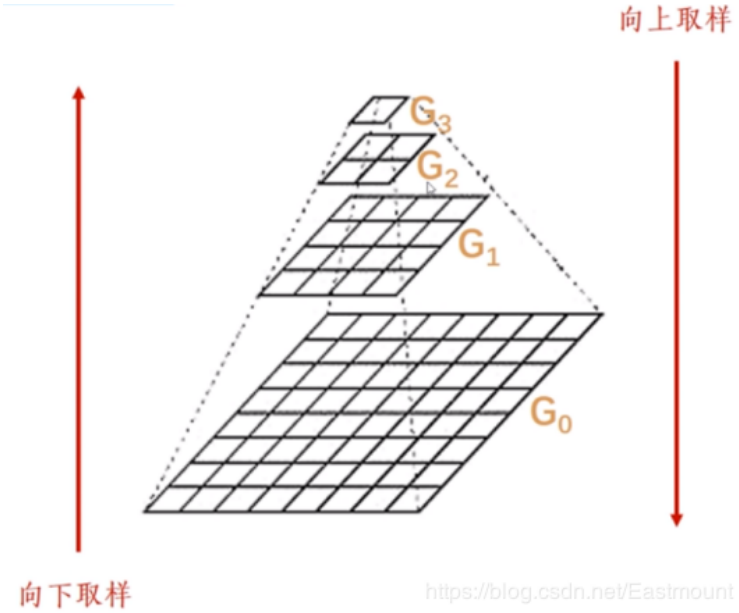
前面一篇文章我讲解了Python图像量化及采样处理，本文将讲解另一个知识——图像金字塔，包括图像向下采样和图像向上采样。基础性文章，希望对你有所帮助。同时，该部分知识均为杨秀璋查阅资料撰写，转载请署名CSDN+杨秀璋及原地址出处，谢谢！！

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：
eastmount - [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》（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.

一.图像金字塔

前面讲解的图像采样处理可以降低图像的大小，本小节将补充图像金字塔知识，了解专门用于图像向上采样和向下采样的pyrUp()和pyrDown()函数。

图像金字塔是指由一组图像且不同分别率的子图集合，它是图像多尺度表达的一种，以多分辨率来解释图像的结构，主要用于图像的分割或压缩。一幅图像的金字塔是一系列以金字塔形状排列的分辨率逐步降低，且来源于同一张原始图的图像集合。如图6-11所示，它包括了四层图像，将这一层一层的图像比喻成金字塔。图像金字塔可以通过梯次向下采样获得，直到达到某个终止条件才停止采样，在向下采样中，层级越高，则图像越小，分辨率越低。



生成图像金字塔主要包括两种方式——向下取样、向上取样。在图6-11中，将图像G0转换为G1、G2、G3，图像分辨率不断降低的过程称为向下取样；将G3转换为G2、G1、G0，图像分辨率不断增大的过程称为向上取样。

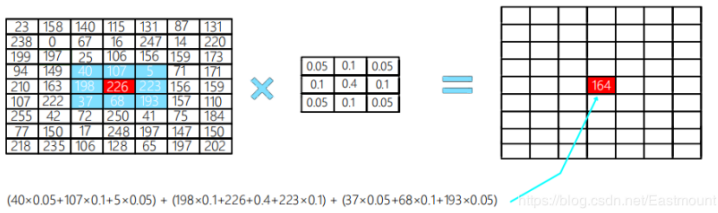
二.图像向下取样

在图像向下取样中，使用最多的是高斯金字塔。它将对图像Gi进行高斯核卷积，并删除原图中所有的偶数行和列，最终缩小图像。其中，高斯核卷积运算就是对整幅图像进行加权平均的过程，每一个像素点的值，都由其本身和邻域内的其他像素值（权重不同）经过加权平均后得到。常见的3×3和5×5高斯核如下：

$$K(3,3)=\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

$$K(5,5)=\frac{1}{273}\begin{bmatrix}1 & 4 & 7 & 4 & 1\\ 4 & 16 & 26 & 16 & 4\\ 7 & 26 & 41 & 26 & 7\\ 4 & 16 & 26 & 16 & 4\\ 1 & 4 & 7 & 4 & 1\end{bmatrix}$$

高斯核卷积让临近中心的像素点具有更高的重要度，对周围像素计算加权平均值，如图6-12所示，其中心位置权重最高为0.4。



显而易见，原始图像Gi具有M×N个像素，进行向下取样之后，所得到的图像Gi+1具有M/2×N/2个像素，只有原图的四分之一。通过对输入的原始图像不停迭代以上步骤就会得到整个金字塔。注意，由于每次向下取样会删除偶数行和列，所以它会不停地丢失图像的信息。

在OpenCV中，向下取样使用的函数为pyrDown()，其原型如下所示：

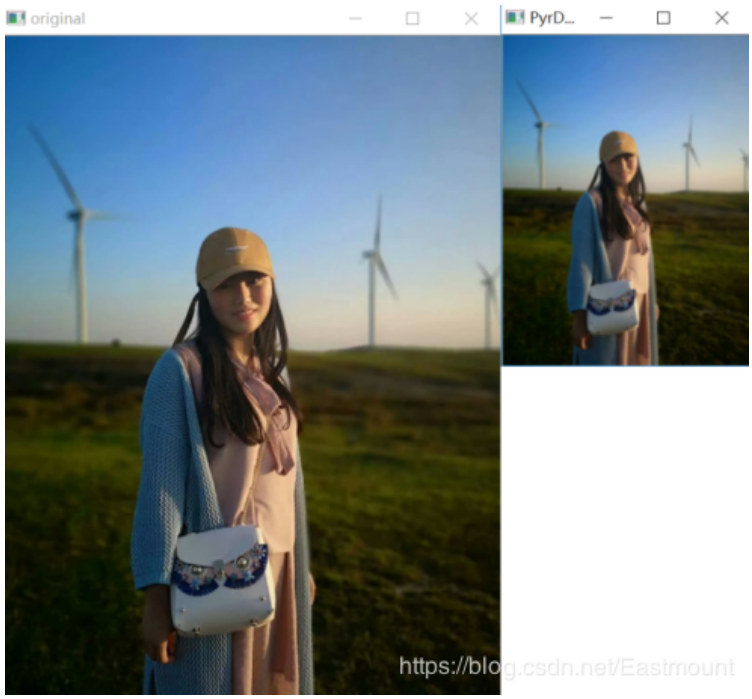
dst = pyrDown(src[, dst[, dstsize[, borderType]]])

- src表示输入图像，
- dst表示输出图像，和输入图像具有一样的尺寸和类型
- dstsize表示输出图像的大小，默认值为Size()
- borderType表示像素外推方法，详见cv::bordertypes

实现代码如下所示：

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

#读取原始图像
img = cv2.imread('nv.png')

#图像向下取样
r = cv2.pyrDown(img)

#显示图像
cv2.imshow('original', img)
cv2.imshow('PyrDown', r)
cv2.waitKey()
cv2.destroyAllWindows()
```

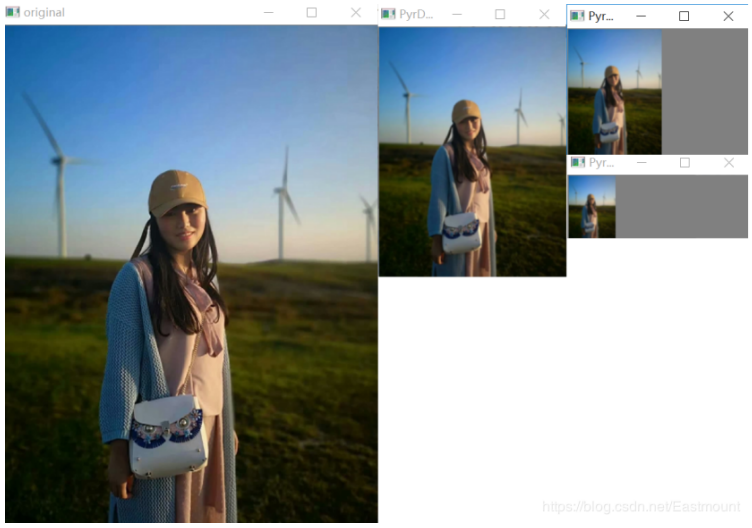
输出结果如图6-13所示，它将原始图像压缩成原图的四分之一。



多次向下取样的代码如下：

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

#读取原始图像
img = cv2.imread('nv.png')

#图像向下取样
r1 = cv2.pyrDown(img)
r2 = cv2.pyrDown(r1)
r3 = cv2.pyrDown(r2)

#显示图像
cv2.imshow('original', img)
cv2.imshow('PyrDown1', r1)
cv2.imshow('PyrDown2', r2)
cv2.imshow('PyrDown3', r3)
cv2.waitKey()
cv2.destroyAllWindows()
```

输出结果如图所示：



三.图像向上取样

在图像向上取样是由小图像不断放图像的过程。它将图像在每个方向上扩大为原图像的2倍，新增的行和列均用0来填充，并使用与“向下取样”相同的卷积核乘以4，再与放大后的图像进行卷积运算，以获得“新增像素”的新值。如图6-15所示，它在原始像素45、123、89、149之间各新增了一行和一列值为0的像素。

$$\begin{vmatrix} 45 & 123 \\ 89 & 149 \end{vmatrix} \rightarrow \begin{vmatrix} 45 & 0 & 123 & 0 \\ 0 & 0 & 0 & 0 \\ 89 & 0 & 149 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

在OpenCV中，向上取样使用的函数为pyrUp()，其原型如下所示：

dst = pyrUp(src[, dst[, dstsize[, borderType]]])

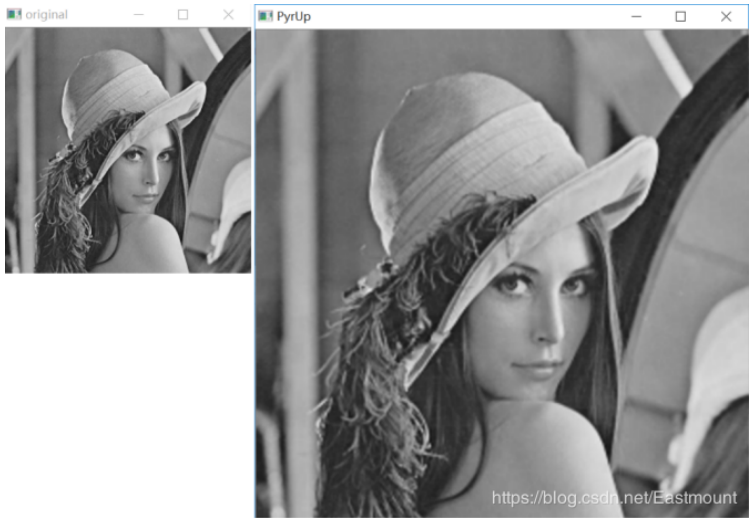
- src表示输入图像，
- dst表示输出图像，和输入图像具有一样的尺寸和类型
- dstsize表示输出图像的大小，默认值为Size()
- borderType表示像素外推方法，详见cv::bordertypes

实现代码如下所示：

```
# -*- 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')

#图像向上取样
r = cv2.pyrUp(img)

#显示图像
cv2.imshow('original', img)
cv2.imshow('PyrUp', r)
cv2.waitKey()
cv2.destroyAllWindows()
```

输出结果如图6-16所示，它将原始图像扩大为原图像的四倍。

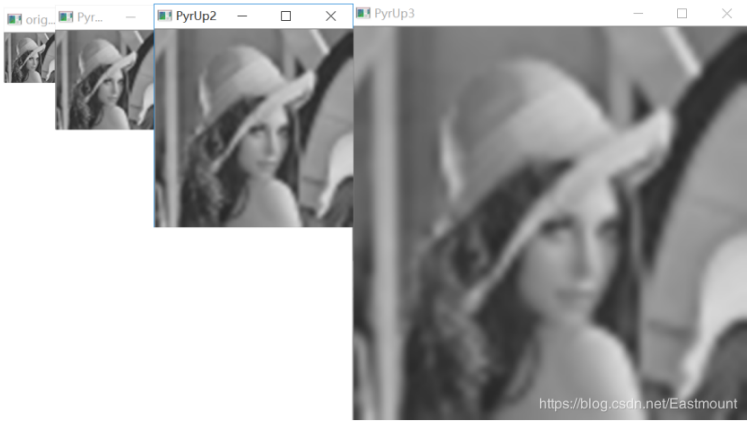


多次向上取样的代码如下：

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

#读取原始图像
img = cv2.imread('lena2.png')

#图像向上取样
r1 = cv2.pyrUp(img)
r2 = cv2.pyrUp(r1)
r3 = cv2.pyrUp(r2)

#显示图像
```

```
cv2.imshow('original', img)
cv2.imshow('PyrUp1', r1)
cv2.imshow('PyrUp2', r2)
cv2.imshow('PyrUp3', r3)
cv2.waitKey()
cv2.destroyAllWindows()
```

输出结果如图6-17所示，每次向上取样均为上次图像的四倍，但图像的清晰度会降低。



四.总结

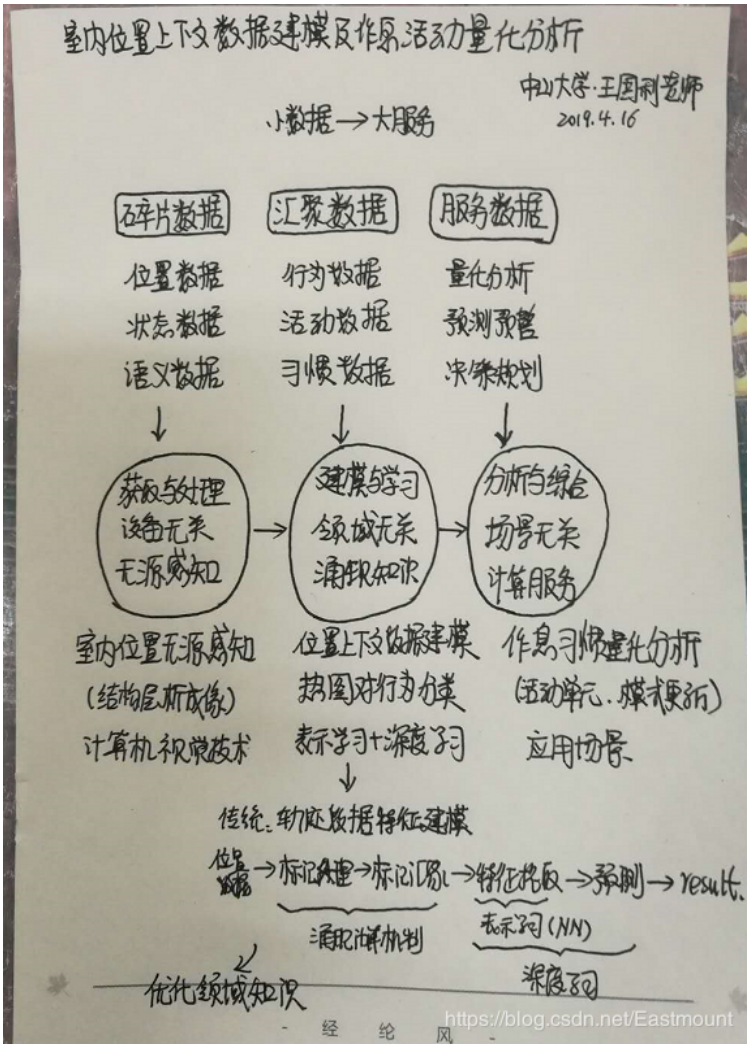
希望这篇基础性文章对您有所帮助，如果有错误或不足之处，请海涵！
最近继续备考博士，接下来还有两个学校，一方面耐心等待之前的结果；另一方面继续复习，周末女神陪着来书店看书，岁月静好，砥砺前行！在这期间，自己经历了很多酸甜苦辣的事情，希望陌生的你也学会享受生活，共勉。

杨秀璋

三位老师分享的知识很受益，利用小数据实现大服务，画了一幅框架图，受教🙏👍六月份开始写深度学习系列博客，看看能不能做点视频公开课，"这节课的最后，让女神弹唱一首《千里之外》，下节课见，我送你离开……"。😂😂

1.并行处理中影响算法效率的结构性因素，南开大学杨愚鲁老师
2.室内位置上下文数据建模及活动量化分析，中山大学王国利老师
3.复杂网络的链路预测研究，南开大学陈增强老师

下班回家喽，继续去书店备考。



（By：Eastmount 2019-04-16 周二夜8点写于贵阳·钟书阁
https://blog.csdn.net/Eastmount ）