OpenCV—python 图像金字塔

1. 原理

一般情况下，我们要处理是一副具有固定分辨率的图像。但是特别情况下我们需要对同一个图像的不同分辨率的子图像进行处理，如查找图像中的某个目标，如人脸，我们不知道目标在图像中的尺寸大小。这种情况下，我们需要创建一组图像，这些图像是具有不同分辨率的原始图像。我们把这组图像叫做图像金字塔。就是同一图像的不同分辨率的子图集合。我们把最大的图像放在底部，最小的放在顶部，看起来就像一座金字塔。
有两类：高斯金字塔和拉普拉斯金字塔。

高斯金字塔

高斯金字塔的顶部是通过将底部图像中的连续的行和列去除得到的。

顶部图像中的每个像素值等于下一层图像中5个像素的高斯加权平均值。这样操作一次一个$M \times N$的图像就变成了一$\frac{M}{2} \times \frac{N}{2}$的图像。所以这幅图像的面积就变为原来图像面积的四分之一。这被称为Octave。

连续这样的操作，我们就会得到一个分辨率不断下降的图像金字塔。可以使用函数`cv2.pyrDown()`和`cv2.pyrUp()`构建图像金字塔。

`cv2.pyrDown()`:

是从一个高分辨率大尺寸图像上构建一个金字塔（尺寸变小，分辨率降低）。

代码实现：

```
import numpy as np
import cv2

img = cv2.imread('1024.jpg')
lower_reso = cv2.pyrDown(img) # 高斯金字塔实现

while(1):
    cv2.imshow('img',img)
    cv2.imshow('lower_reso',lower_reso)
    if cv2.waitKey() == ord('q'):
        break
cv2.destroyAllWindows()
```

`cv2.pyrUp()`:

从一个低分辨率小尺寸图像向上构建一个金字塔(尺寸变大，但分辨率不会增加)

```
import numpy as np
import cv2

img = cv2.imread('1024.jpg')
lower_reso = cv2.pyrDown(img)# 高斯金字塔实现
higher_reso2 = cv2.pyrUp(img)# 高斯金字塔实现

while(1):
    cv2.imshow('img',img)
    cv2.imshow('lower_reso',lower_reso)
    cv2.imshow('higher_reso2',higher_reso2)
    if cv2.waitKey() == ord('q'):
        break
cv2.destroyAllWindows()
```

> 注:
>
> `higher_reso2`和`higher_reso`是不同的。因为一旦使用`cv2.pyrDown`图像的分辨率就会降低，信息就会被丢失。

拉普拉斯金字塔

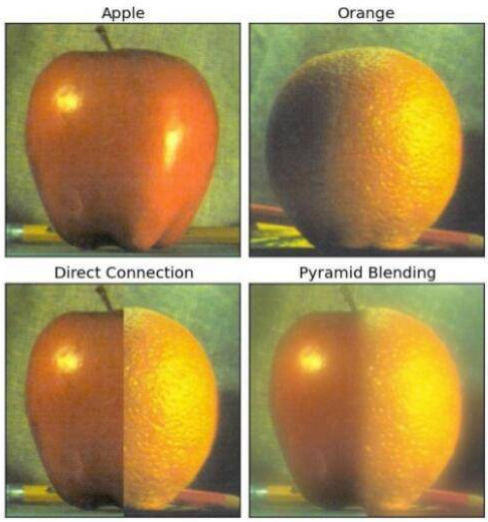
拉普拉斯金字塔可以由高斯金字塔计算得来。

$$L_i = G_i - PyrUp(G_{i+1})$$

拉普拉斯金字塔的图像看起来就像是边界图，其中很多像素都是0，常被用在图像压缩中。

2. 使用金字塔进行图像融合

在图像缝合中，由于连接区域图像像素的不连续，整幅图看起来会很差，金字塔就可以实现无缝连接。
经典例子就是水果融合



实现步骤：

- 读入两幅图
- 构建各自的高斯金字塔（6层）
- 根据高斯金字塔计算拉普拉斯金字塔
- 在拉普拉斯的每一层进行图像融合
- 根据融合后的图像金字塔重建原始图像。

重建原始图像过程

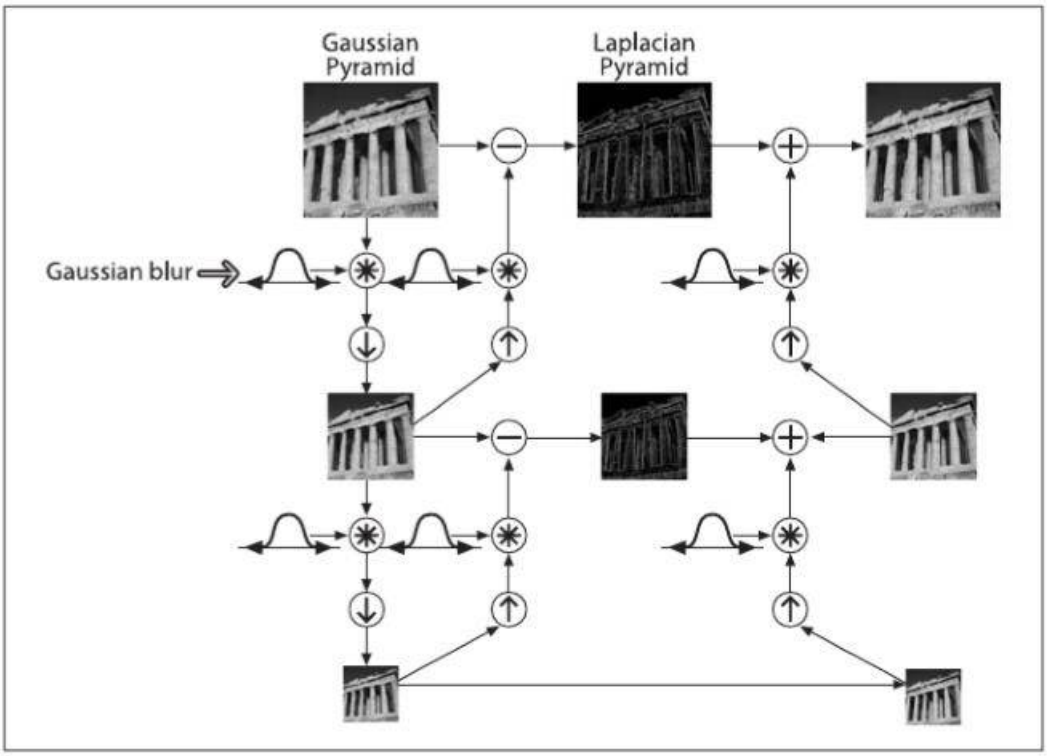


Figure 5-21. The Gaussian pyramid and its inverse, the Laplacian pyramid