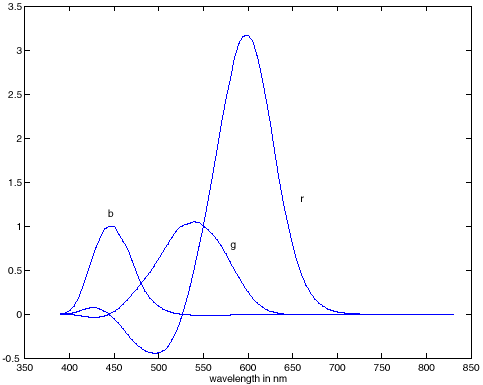
# RGB space



RGB颜色匹配函数

**横坐标表示光谱波长，纵坐标表示用以匹配光谱各色所需要三基色刺激值**，这些值是以等能量白光为标准的系数，是观察者实验结果的平均值。为了匹配在438.1 nm和546.1 nm之间的光谱色，出现了负值，这就意味匹配这段里的光谱色时，混合颜色需要**使用补色**才能匹配。虽然使用正值提供的色域还是比较宽的，但像用RGB相加混色原理的CRT虽然可以显示大多数颜色，但不能显示所有的颜色

# Linear color spaces: CIE XYZ

线性颜色空间

以避免出现负值，而且使用也方便。CIE系统采用想象的X，Y和Z三种基色，它们与可见颜色不相应。CIE选择的X，Y和Z基色具有如下性质：

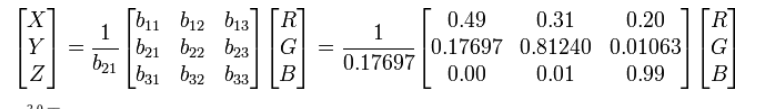
1、 所有的X，Y和Z值都是正的，匹配光谱颜色时不需要一种负值的基色；

2、用Y值表示人眼对亮度（luminance）的响应；

3、 如同RGB模型，X，Y和Z是相加基色。因此，每一种颜色都可以表示成X，Y和Z的混合

Matching functions





**横坐标表示可见光谱的波长，纵坐标表示基色X，Y和Z的相对值**。三条曲线表示X，Y和Z三基色刺激值如何组合以产生可见光谱中的所有颜色。例如，要匹配波长为450 nm的颜色（蓝/紫），需要0.33单位的X基色，0.04单位的Y基色和1.77单位的Z基色。



CIE XYZ的三基色刺激值X，Y和Z对定义颜色很有用，其缺点是使用比较复杂，而且不直观。**CIE xyY的颜色空间**

定义CIE xyY颜色空间的根据是，对于一种给定的颜色，如果增加它的**明度**，每一种基色的**光通量**也要按比例增加，这样才能匹配这种颜色。因此，当颜色点离开原点（X=0,Y=0,Z=0）时，**X:Y:Z的比值保持不变**。此外，由于色度值仅与波长（色调）和纯度有关，而与总的辐射能量无关，因此在计算颜色的色度时，把X,Y和Z值相对于总的辐射能量=(X+Y+Z）进行**归一化**，并只需考虑它们的相对比例，因此，x,y,z称为三基色相对系数，于是配色方程可规格化为x+y+z=1。由于三个相对系数x,y,z之和恒为1，这就相当于把XYZ颜色锥体投影到X+Y+Z=1的平面上。

由于z可以从x+y+z=1导出，因此通常不考虑z，而用另外两个系数x和y表示颜色，并绘制以x和y为坐标的二维图形。这就相当于把X+Y+Z=1平面投射到（X,Y）平面，也就是Z=0的平面，这就是**CIE xyY色度图**。

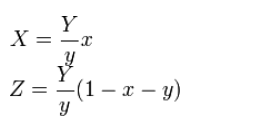
CIE xyY色度图是从XYZ直接导出的一个颜色空间，它使用亮度Y参数和颜色坐标x,y来描述颜色。xyY中的Y值与XYZ中的Y刺激值一致，表示颜色的亮度或者光亮度，颜色坐标x,y用来在二维图上指定颜色

CIE 1931色度图是用标称值表示的CIE色度图，x表示红色分量，y表示绿色分量。E点代表白光，它的坐标为（0.33，0.33）；

环绕在颜色空间边沿的颜色是光谱色，边界代表光谱色的最大饱和度，边界上的数字表示光谱色的波长，其轮廓包含所有的感知色调。

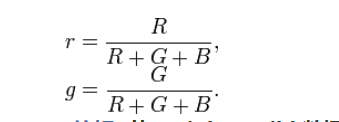
所有单色光都位于舌形曲线上，这条曲线就是**单色轨迹**，曲线旁标注的数字是单色（或称光谱色）光的波长值；自然界中各种实际颜色都位于这条闭合曲线内；RGB系统中选用的物理三基色在色度图的舌形曲线上

X和Z三色刺激值可以从色度值x和y与Y三色刺激值计算：



**Grassmann’s Law：**

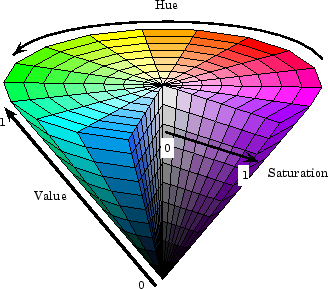
色度坐标r g



# Nonlinear color spaces: HSV

根据**颜色的直观特性**创建的一种颜色空间, 也称六角锥体模型

Hue色调 Saturation饱和度 Value值（强度）



色调H

用角度度量，取值范围为0°～360°，从红色开始按逆时针方向计算，红色为0°，绿色为120°,蓝色为240°。它们的补色是：黄色为60°，青色为180°,品红为300°；

H参数表示色彩信息，即所处的光谱颜色的位置。该参数用一角度量来表示，红、绿、蓝分别相隔120度。互补色分别相差180度

饱和度S

饱和度S表示颜色接近光谱色的程度。一种颜色，可以看成是某种光谱色与白色混合的结果。其中光谱色所占的比例愈大，颜色接近光谱色的程度就愈高，颜色的饱和度也就愈高。饱和度高，颜色则深而艳。光谱色的白光成分为0，饱和度达到最高。通常取值范围为0%～100%，值越大，颜色越饱和。

纯度S为一比例值，范围从0到1，它表示成所选颜色的纯度和该颜色最大的纯度之间的比率。S=0时，只有灰度

明度V

明度表示颜色明亮的程度，对于光源色，明度值与发光体的光亮度有关；对于物体色，此值和物体的透射比或反射比有关。通常取值范围为0%（黑）到100%（白）。

V表示色彩的明亮程度，范围从0到1。有一点要注意：它和光强度之间并没有直接的联系。

HSV模型的三维表示从RGB立方体演化而来。设想从RGB沿立方体对角线的白色顶点向黑色顶点观察，就可以看到立方体的六边形外形。六边形边界表示色彩，水平轴表示纯度，明度沿垂直轴测量

**算法：**

① RGB转化到HSV的算法:

max=max(R,G,B)；

min=min(R,G,B)；

V=max(R,G,B)；

S=(max-min)/max；

HSV颜色空间模型（圆锥模型）

if (R = max) H =(G-B)/(max-min)\* 60；

if (G = max) H = 120+(B-R)/(max-min)\* 60；

if (B = max) H = 240 +(R-G)/(max-min)\* 60；

if (H < 0) H = H+ 360；

② HSV转化到RGB的算法:

if (s = 0)

R=G=B=V;

else

H /= 60;

i = INTEGER(H);

f = H - i;

a = V \* ( 1 - s );

b = V \* ( 1 - s \* f );

c = V \* ( 1 - s \* (1 - f ) );

switch(i)

case 0: R = V; G = c; B = a;

case 1: R = b; G = v; B = a;

case 2: R = a; G = v; B = c;

case 3: R = a; G = b; B = v;

case 4: R = c; G = a; B = v;

case 5: R = v; G = a; B = b;

HSV对用户来说是一种直观的颜色模型。我们可以从一种纯色彩开始，即指定色彩角H，并让V=S=1，然后我们可以通过向其中加入黑色和白色来得到我们需要的颜色。增加黑色可以减小V而S不变，同样增加白色可以减小S而V不变。例如，要得到深蓝色，V=0.4 S=1 H=240度。要得到淡蓝色，V=1 S=0.4 H=240度。

人眼最大能区分128种不同的色彩，13种色饱和度，23种明暗度。如果我们用16Bit表示HSV的话，可以用7位存放H，4位存放S，5位存放V，即745或者655就可以满足需要

# White balance

白平衡是调整传感器接收的图像数据以适当渲染的处理

Illuminant 光源

pixels 像素

1.冯克里斯适应

将每个通道乘以一个增益因子

一般的变换将对应于任意的3x3矩阵

2.灰卡

拍摄一张中性物体（白色或灰色）

减轻每个频道的重量

如果对象被重新编码为rw，gw，bw则使用权重1 / rw，1 / gw，1 / bw

2.1灰色的假设

图像平均r g b是灰色的

使用重量1 / rave，1 / give，1 / bave

2.2最明亮的像素假设（非Starated）

亮点通常具有光源的颜色

使用与最亮像素值成反比的权重

2.3色域映射

色域：图像中所有像素颜色的凸包

在白光下找到匹配图像色域与“典型”图像色域的转换

# Linear Algebra Primer: Vectors and Matrices

column vectors 列向量

Matrices 矩阵

Norm 范数

Non-negativity 非负

Definiteness 定性

Homogeneity 同质性

Triangle Inequality 三角不等性

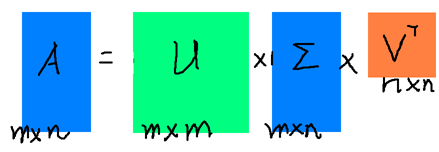
Dot product 点积

6. 奇异值分解（SVD）

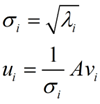
特征值分解是一个提取矩阵特征很不错的方法，但是它只是对方阵而言的，在现实的世界中，我们看到的大部分矩阵都不是方阵，比如说有N个学生，每个学生有M科成绩，这样形成的一个N \* M的矩阵就不可能是方阵，我们怎样才能描述这样普通的矩阵呢的重要特征呢？奇异值分解可以用来干这个事情，奇异值分解是一个能适用于任意的矩阵的一种分解的方法：



假设A是一个N \* M的矩阵，那么得到的U是一个N \* N的方阵（里面的向量是正交的，U里面的向量称为左奇异向量），Σ是一个N \* M的矩阵（除了对角线的元素都是0，对角线上的元素称为奇异值），V’(V的转置)是一个N \* N的矩阵，里面的向量也是正交的，V里面的向量称为右奇异向量），从图片来反映几个相乘的矩阵的大小可得下面的图片

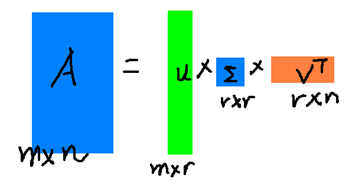


    那么奇异值和特征值是怎么对应起来的呢？首先，我们将一个矩阵A的转置 \* A，将会得到一个方阵，我们用这个方阵求特征值可以得到：    这里得到的v，就是我们上面的右奇异向量。此外我们还可以得到：

    这里的σ就是上面说的奇异值，u就是上面说的左奇异向量。奇异值σ跟特征值类似，在矩阵Σ中也是从大到小排列，而且σ的减少特别的快，在很多情况下，前10%甚至1%的奇异值的和就占了全部的奇异值之和的99%以上了。也就是说，我们也可以用前r大的奇异值来近似描述矩阵，这里定义一下部分奇异值分解：



    r是一个远小于m、n的数，这样矩阵的乘法看起来像是下面的样子



    右边的三个矩阵相乘的结果将会是一个接近于A的矩阵，在这儿，r越接近于n，则相乘的结果越接近于A。而这三个矩阵的面积之和（在存储观点来说，矩阵面积越小，存储量就越小）要远远小于原始的矩阵A，我们如果想要压缩空间来表示原矩阵A，我们存下这里的三个矩阵：U、Σ、V就好了。

7. 特征值分解

如果说一个向量v是方阵A的特征向量，将一定可以表示成下面的形式：

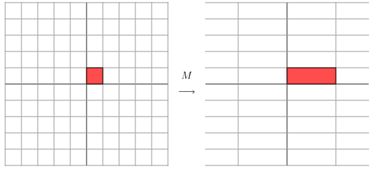


    这时候λ就被称为特征向量v对应的特征值，一个矩阵的一组特征向量是一组正交向量。特征值分解是将一个矩阵分解成下面的形式：



    其中Q是这个矩阵A的特征向量组成的矩阵，Σ是一个对角阵，每一个对角线上的元素就是一个特征值。我这里引用了一些参考文献中的内容来说明一下。首先，要明确的是，一个矩阵其实就是一个线性变换，因为一个矩阵乘以一个向量后得到的向量，其实就相当于将这个向量进行了线性变换。比如说下面的一个矩阵：

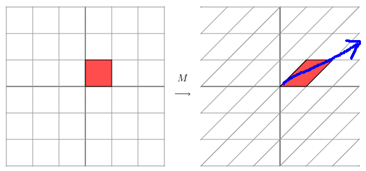
    它其实对应的线性变换是下面的形式：

    因为这个矩阵M乘以一个向量(x,y)的结果是：

    上面的矩阵是对称的，所以这个变换是一个对x，y轴的方向一个拉伸变换（每一个对角线上的元素将会对一个维度进行拉伸变换，当值>1时，是拉长，当值<1时时缩短），当矩阵不是对称的时候，假如说矩阵是下面的样子：



    它所描述的变换是下面的样子：



    这其实是在平面上对一个轴进行的拉伸变换（如蓝色的箭头所示），在图中，蓝色的箭头是一个最主要的变化方向（变化方向可能有不止一个），如果我们想要描述好一个变换，那我们就描述好这个变换主要的变化方向就好了。反过头来看看之前特征值分解的式子，分解得到的Σ矩阵是一个对角阵，里面的特征值是由大到小排列的，这些特征值所对应的特征向量就是描述这个矩阵变化方向（从主要的变化到次要的变化排列）

    当矩阵是高维的情况下，那么这个矩阵就是高维空间下的一个线性变换，这个线性变化可能没法通过图片来表示，但是可以想象，这个变换也同样有很多的变换方向，我们通过特征值分解得到的前N个特征向量，那么就对应了这个矩阵最主要的N个变化方向。我们利用这前N个变化方向，就可以近似这个矩阵（变换）。也就是之前说的：提取这个矩阵最重要的特征。总结一下，特征值分解可以得到特征值与特征向量，特征值表示的是这个特征到底有多重要，而特征向量表示这个特征是什么，可以将每一个特征向量理解为一个线性的子空间，我们可以利用这些线性的子空间干很多的事情。不过，特征值分解也有很多的局限，比如说变换的矩阵必须是方阵

imshow()函数格式为：

matplotlib.pyplot.imshow(X, cmap=None)

X: 要绘制的图像或数组。

cmap: 颜色图谱（colormap), 默认绘制为RGB(A)颜色空间。

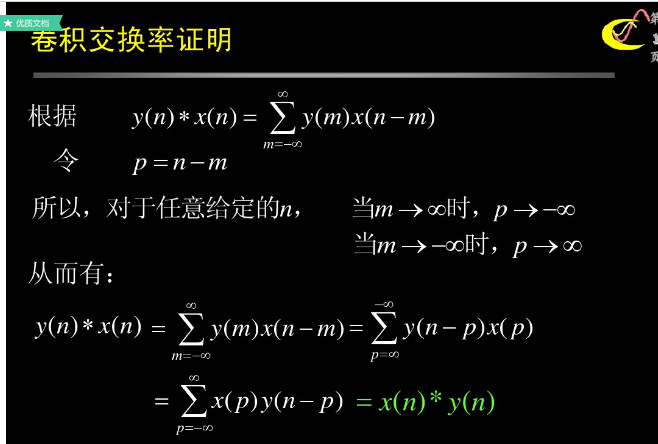
plt.imshow(img, cmap=plt.get\_cmap('img'))

plt.imshow(img, cmap='Greys\_r')

plt.imshow(img, plt.cm.gray)

**Lecture3—Linear Algebra Primer**

**作业2：**



## np.pad()函数

1）[语法结构](https://docs.scipy.org/doc/numpy/reference/generated/numpy.pad.html)

pad(array, pad\_width, mode, \*\*kwargs)

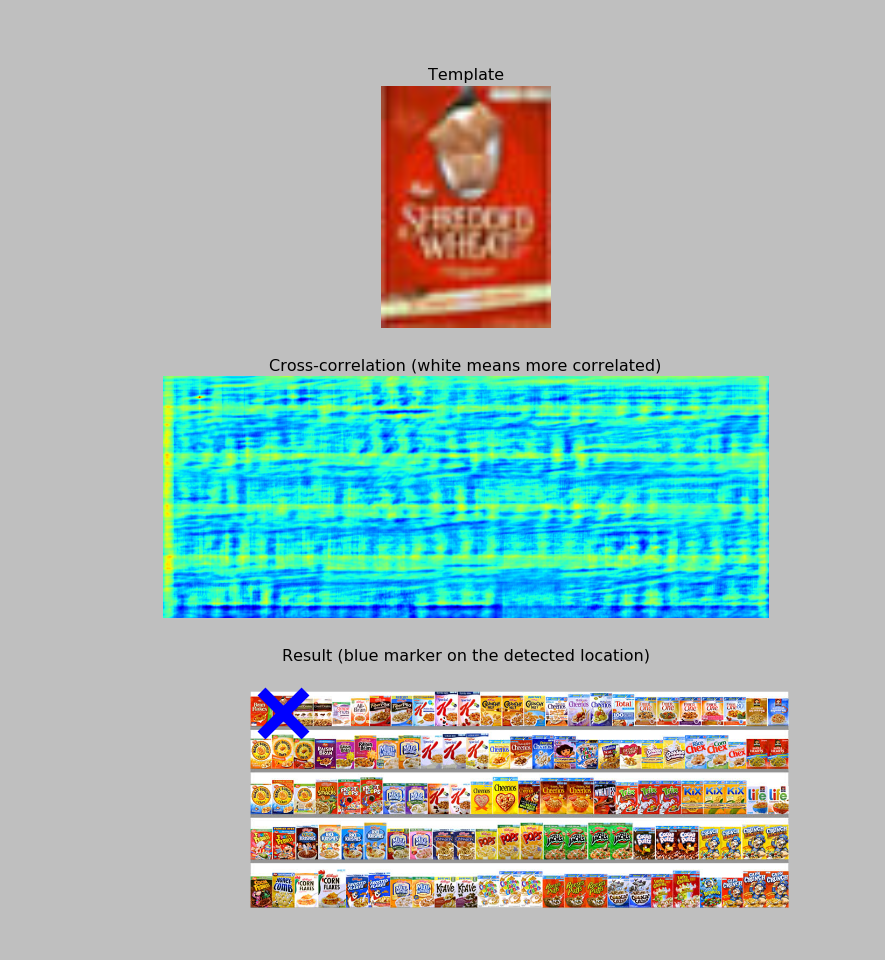
返回值：数组

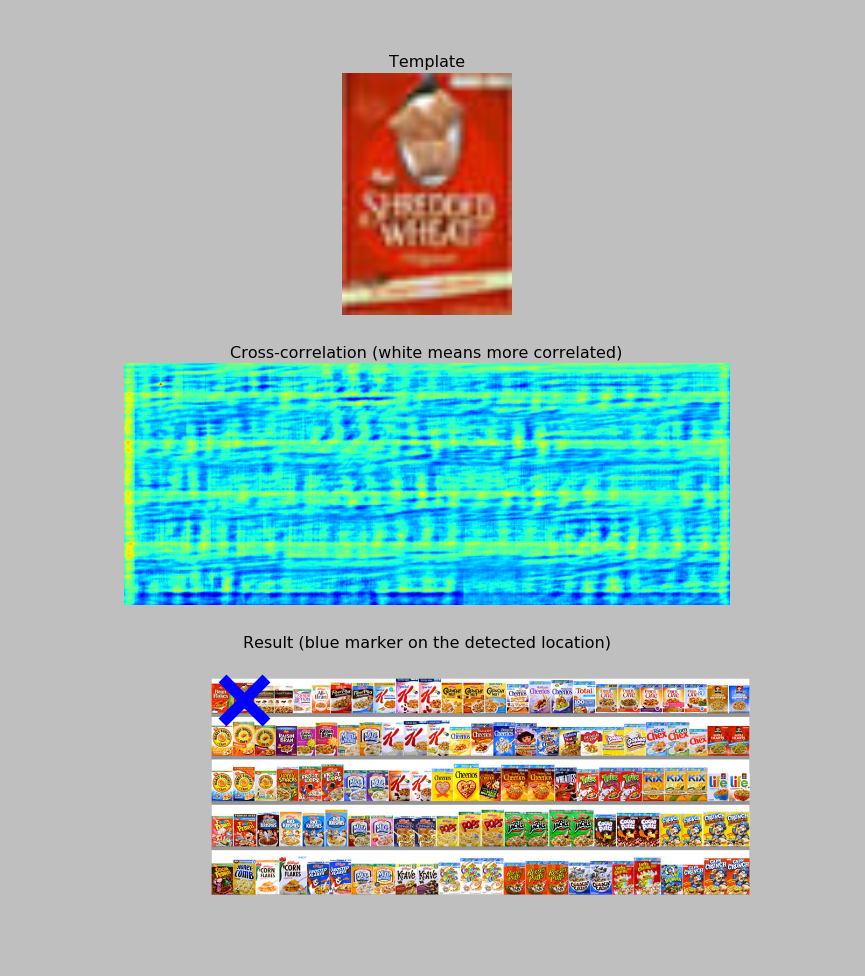
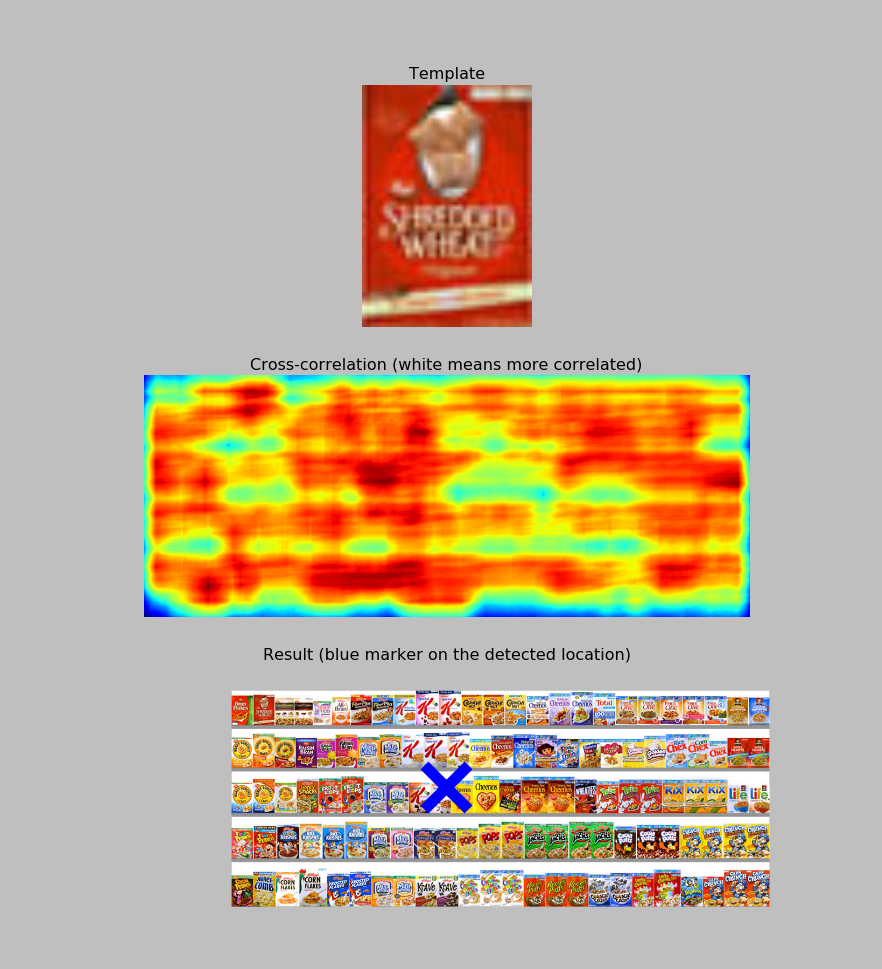
2）[参数解释](https://docs.scipy.org/doc/numpy/reference/generated/numpy.pad.html)

array——表示需要填充的数组；

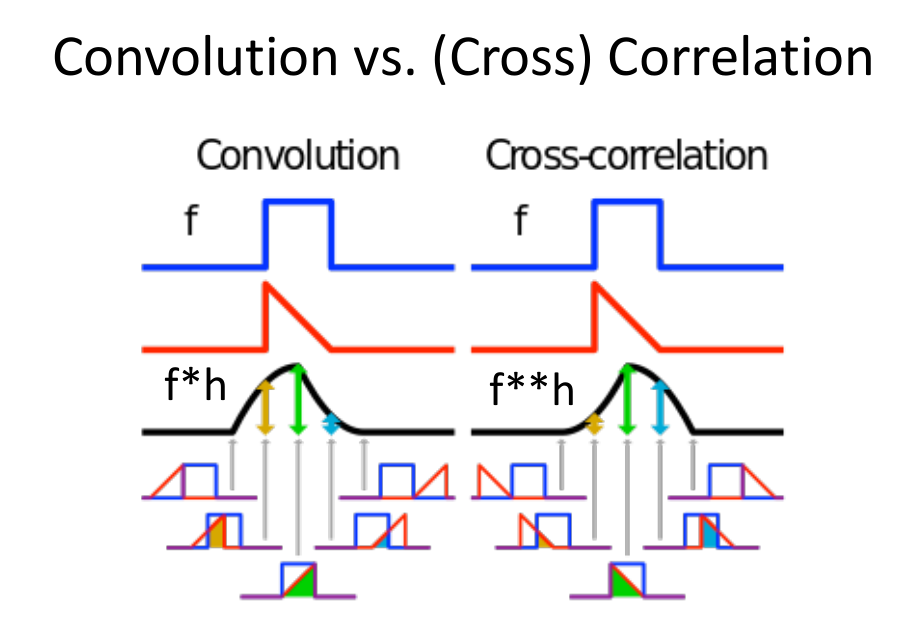
pad\_width——表示每个轴（axis）边缘需要填充的数值数目。   
参数输入方式为：（(before\_1, after\_1), … (before\_N, after\_N)），其中(before\_1, after\_1)表示第1轴两边缘分别填充before\_1个和after\_1个数值。取值为：{sequence, array\_like, int}

mode——表示填充的方式（取值：str字符串或用户提供的函数）,总共有11种填充模式；





卷积和互相关的区别：是否翻转180度



## 一阶导数的3种近似：前向差分，后向差分，中心差分

Backward .Forward Central