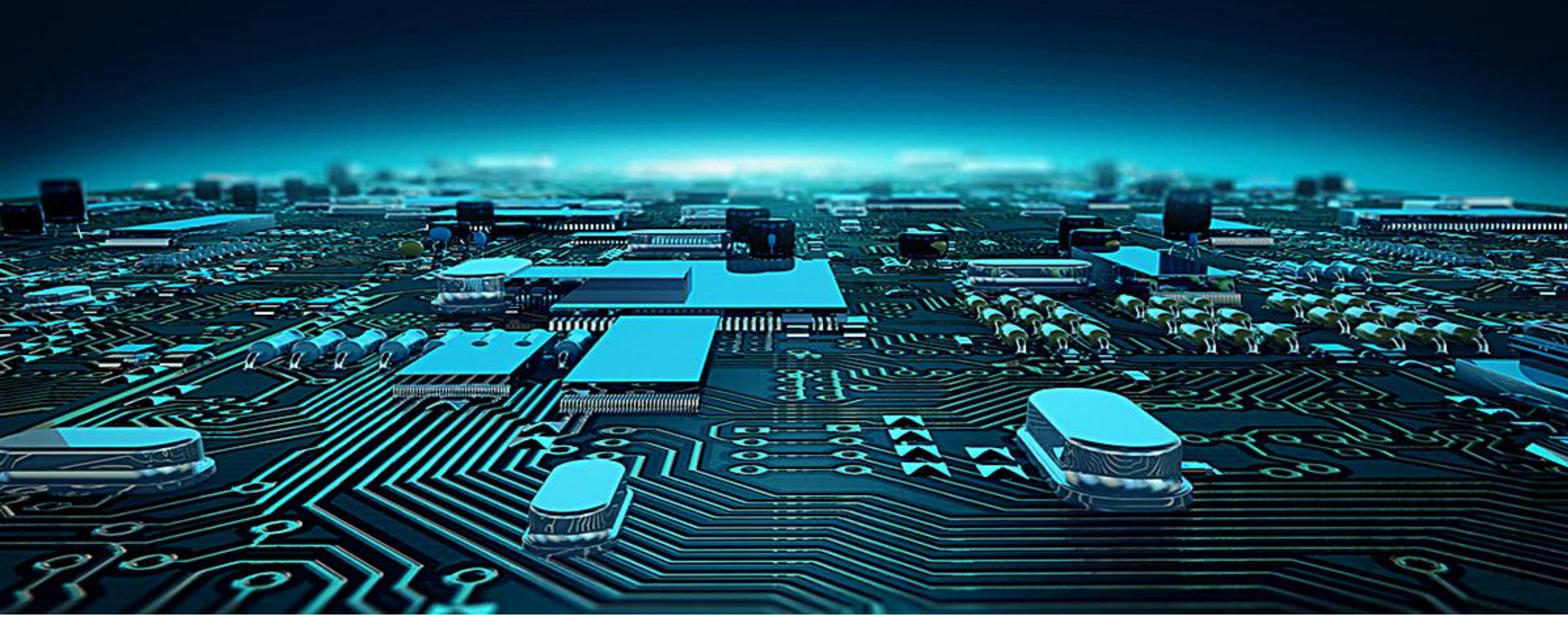
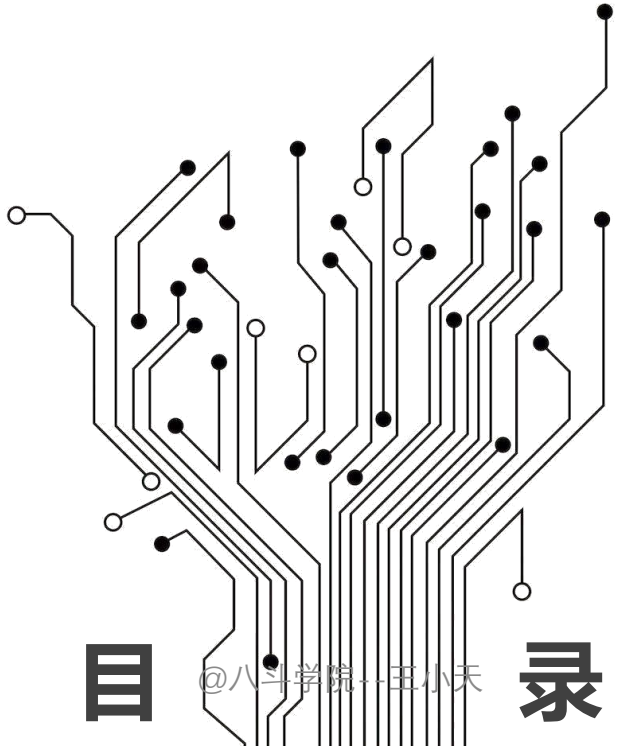


数字图像

@八斗学院--王小天(Michael)

2023/04/09



目　录

图像

•**像素**：像素是分辨率的单位。像素是构成位图图像最基本的单元，每个像素都有自己的颜色。

•**分辨率（解析度）**：

a) 图像分辨率就是单位英寸内的像素点数。单位为PPI(Pixels Per Inch)

b) PPI表示的是每英寸对角线上所拥有的的像素数目：PPI=√（X^2+Y^2）/Z（X：长度像素数；Y：宽度像素数；Z：屏幕大小）

c) 屏幕尺寸指的是对角线长度

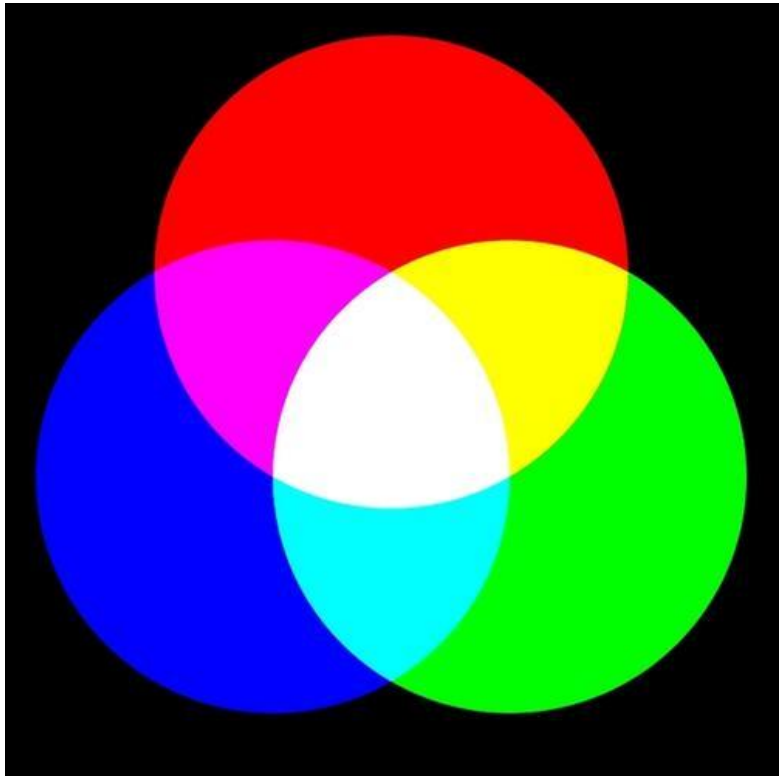
d)在生活中被混用，或者说错误的用做衡量图像内的像素点数量。

```
>>> PPI=(math.sqrt(100**2+100**2))/(math.sqrt(2))
>>> print(PPI)
100.0
>>> pixel=100*(math.sqrt(2))
>>> pixel=(100*(math.sqrt(2)))**2
>>> print(pixel)
20000.0
>>> print(pixel/2)
10000.0
```

RGB模型

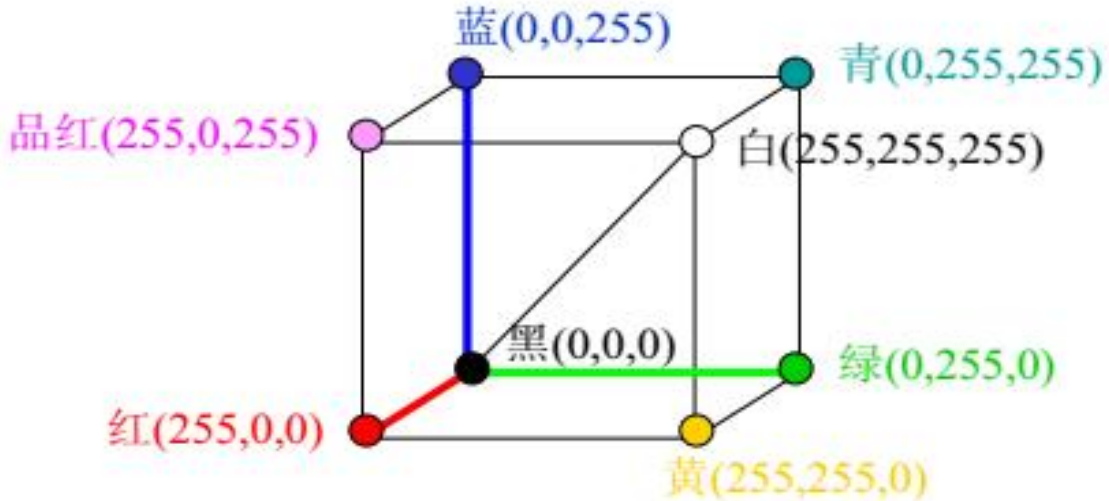
色彩三原色（CMYK）：品红、黄、青

光学三原色（RGB）：红、绿、蓝



颜色模型——RGB 模型

- RGB颜色模型是三维直角坐标颜色系统中的一个单位正方体
- 在正方体的主对角线上，各原色的量相等，产生由暗到亮的白色，即灰度。
  正方体的其他6个角点分别为红、黄、绿、青、蓝和品红



RGB转化为Gray

1.浮点算法：Gray=R0.3+G0.59+B0.11

2.整数方法：Gray=(R30+G59+B11)/100

3.移位方法：Gray =(R76+G151+B*28)>>8;

4.平均值法：Gray=（R+G+B）/3;

5.仅取绿色：Gray=G;

为什么很多图像识别将彩色图像灰度化?

RGB值转化为浮点数

- 浮点数运算结果更精确，整数运算中会因丢弃小数部分可能导致颜色值严重失真，计算过程越多越失真

- 将RGB值转化为[0,1]浮点数

- 二值化：

```
#         if (img_gray[i, j] <= 0.5):
#             img_gray[i, j] = 0
#         else:
#             img_gray[i, j] = 1
```

opencv大坑之BGR

opencv对于读进来的图片的通道排列是BGR，而不是主流的RGB！谨记！

```
#opencv读入的矩阵是BGR，如果想转为RGB，可以这么转
img4 = cv2.imread('1.jpg')
img4 = cv2.cvtColor(img4,cv2.COLOR_BGR2RGB)
```

图像

- **灰度**：表示图像像素明暗程度的数值，也就是黑白图像中点的颜色深度。范围一般为0-255。白色为255，黑色为0。
- **通道**：把图像分解成一个或多个颜色成分：

  ①单通道：一个像素点只需一个数值表示，只能表示灰度，0为黑色；（二值图&灰度图）

  ②三通道：RGB模式，把图像分为红绿蓝三个通道，可以表示彩色，全0表示黑色；

  ③四通道：RGBA模式，在RGB基础上加上alpha通道，表示透明度，alpha=0表示全透明
- **对比度**：指不同颜色之间的差别。对比度=最大灰度值/最小灰度值

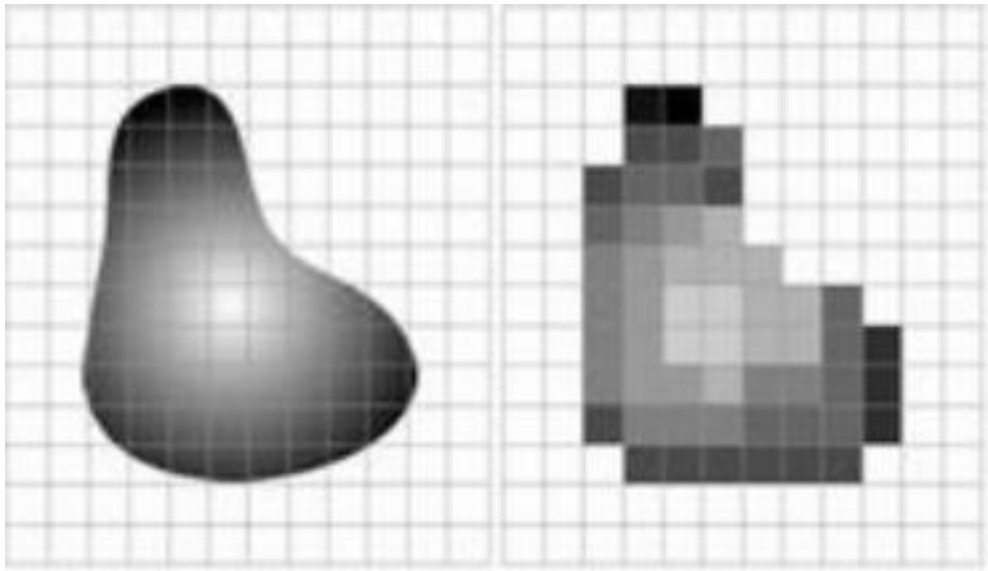
图像

- **频率:** 灰度值变化剧烈程度的指标，是灰度在平面空间上的梯度。
- 高频、低频

- **幅值:** 幅值是在一个周期内，交流电瞬时出现的最大绝对值，也是一个正弦波，波峰到波谷的距离的一半。

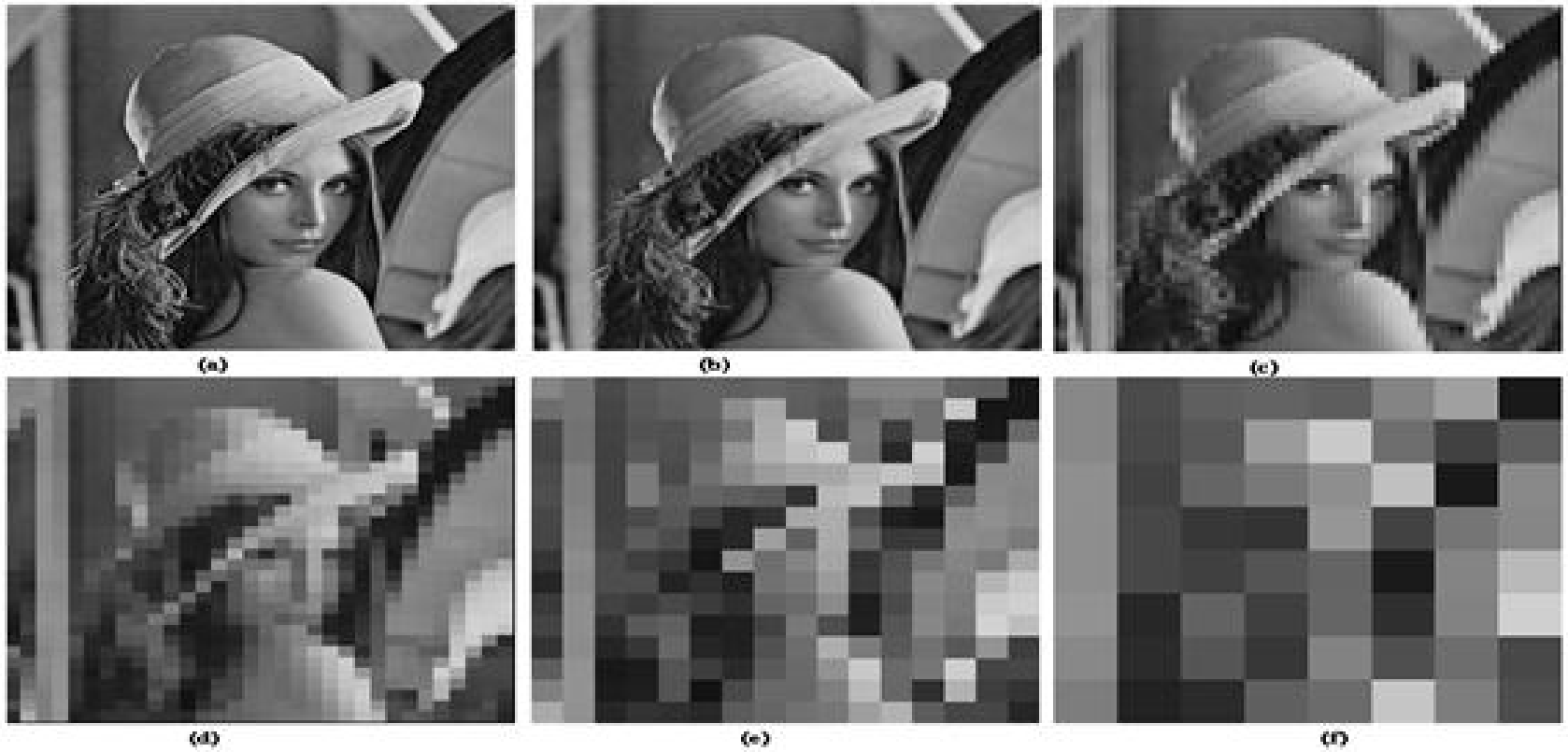
图像的取样和量化

- 数字图像：计算机保存的图像都是一个一个的像素点，称为数字图像。
- 图像数字化过程由图像的取样与量化来完成。
- 取样：就是要用多少点来描述一幅图像，取样结果质量的高低就是用图像的分辨率来衡量的
- 量化：是指要使用多大范围的数值来表示图像采样之后的一个点。
- **数字化坐标值称为取样，数字化幅度值称为量化。**



图像的取样和量化

- 在取样时，若横向的像素数（列数）为M，纵向的像素数（行数）为N，则图像总像素数为M*N个像素。



上采样与下采样

- 缩小图像（或称为下采样（subsampled）或降采样（downsampled））的主要目的有两个：1、使得图像符合显示区域的大小；2、生成对应图像的缩略图。

- 放大图像（或称为上采样（upsampling）或图像插值（interpolating））的主要目的是放大原图像,从而可以显示在更高分辨率的显示设备上。

上采样与下采样

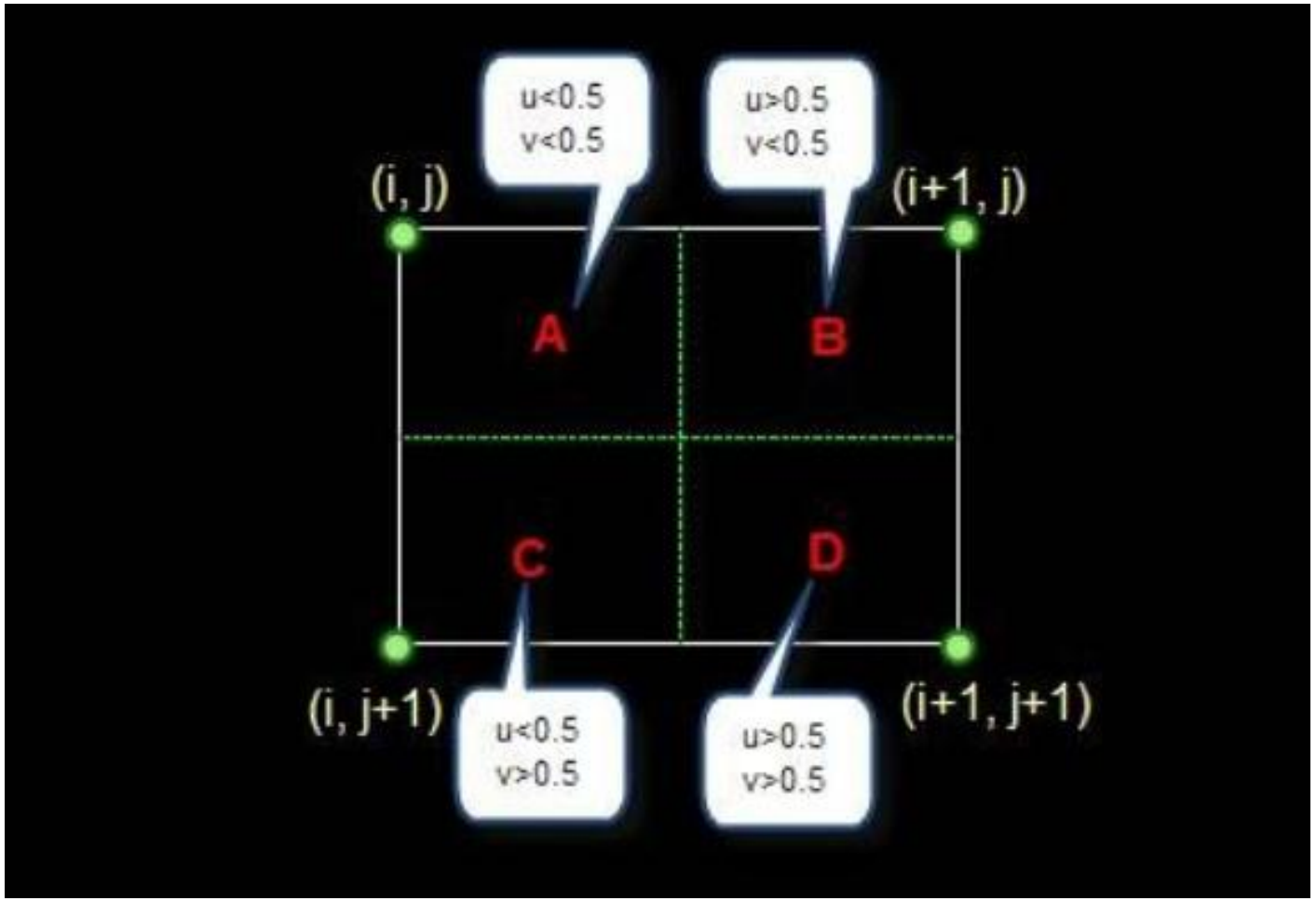
上采样原理：内插值

下采样原理：(M/s) * (N/s)

常用的插值方法

1、最邻近插值The nearest interpolation

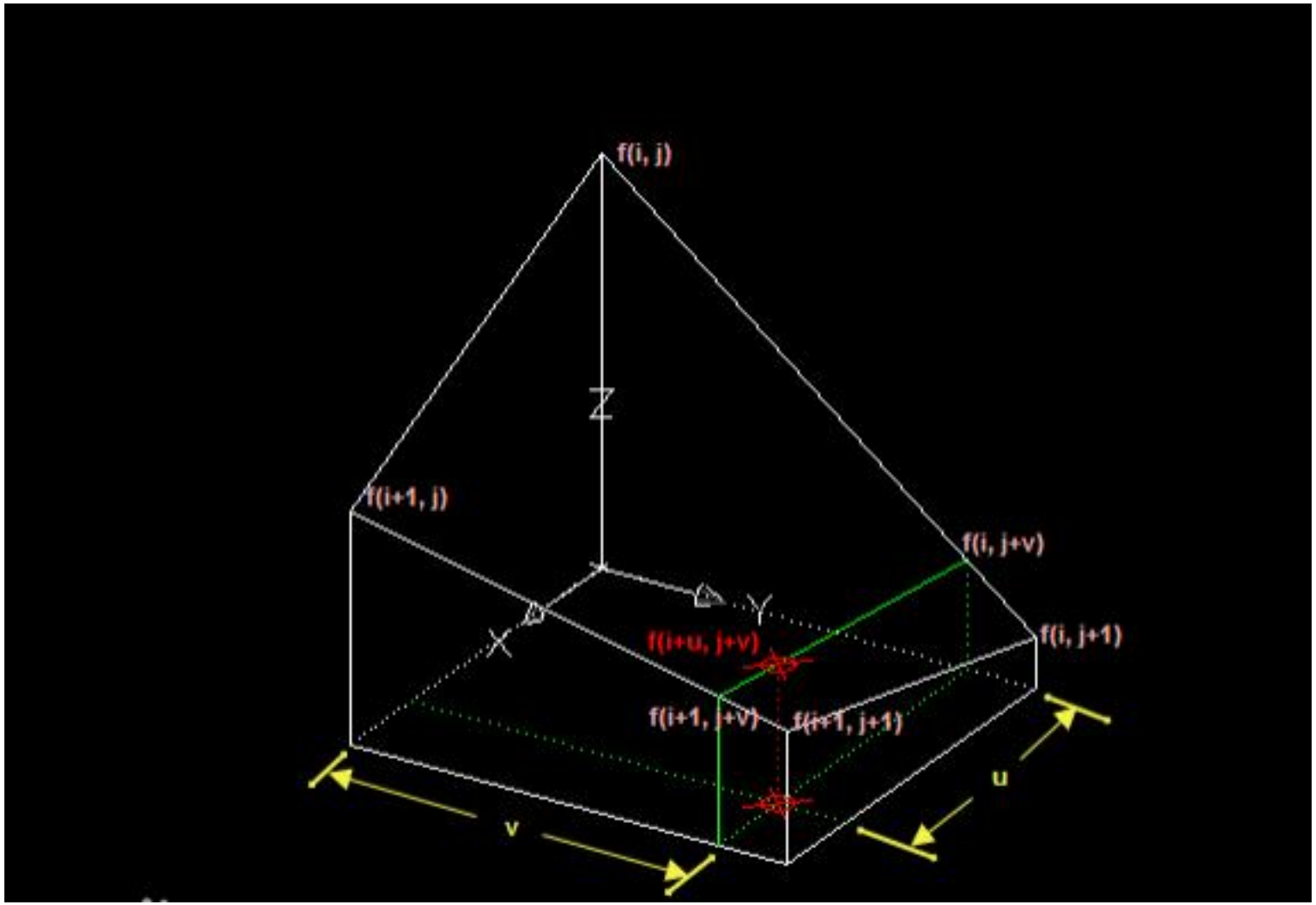
设i+u, j+v (i, j为正整数， u, v为大于零小于1的小数， 下同)为待求象素坐标，则待求象素灰度的值 f(i+u, j+v) 如下图所示:



常用的插值方法

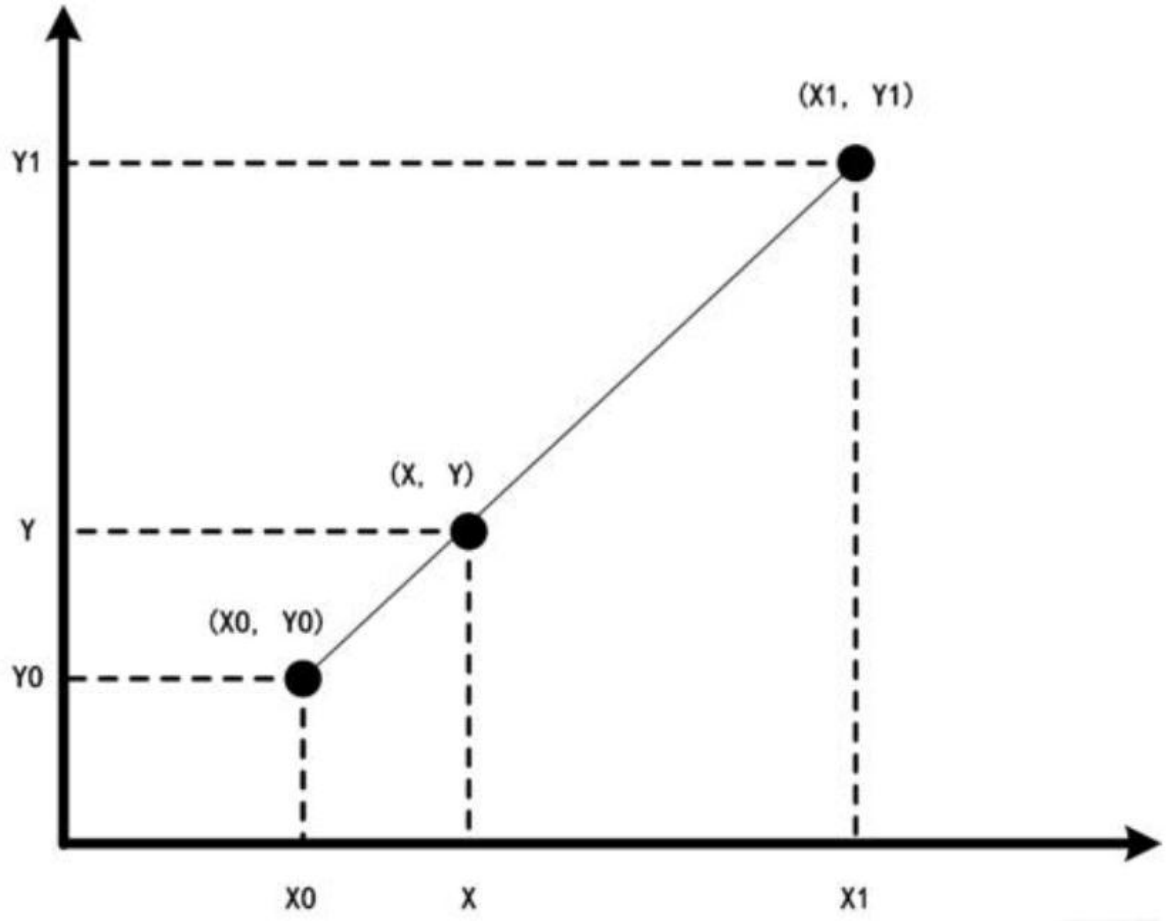
2、双线性插值

f(i+u, j+v) = (1-u) * (1-v) * f(i, j) + (1-u) * v * f(i, j+1) + u * (1-v) * f(i+1, j) + u * v * f(i+1, j+1)

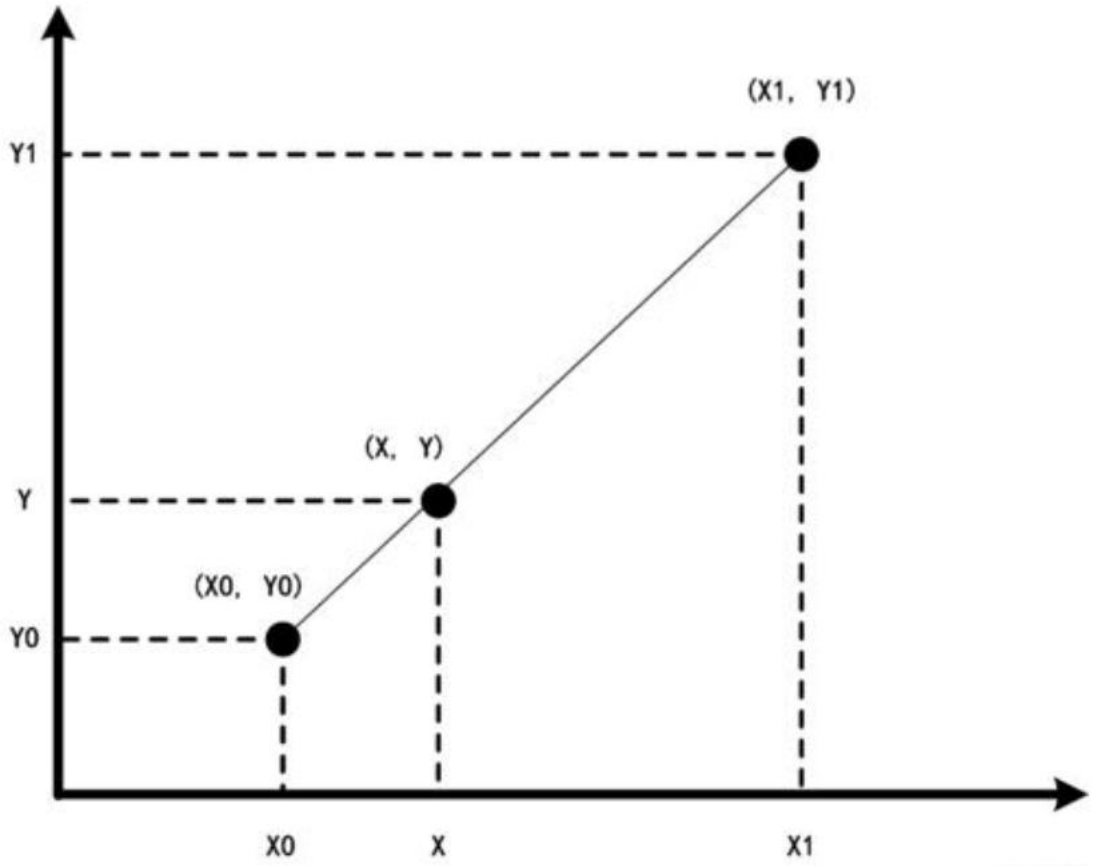


常用的插值方法--双线性插值

2.1 单线性插值



常用的插值方法--双线性插值

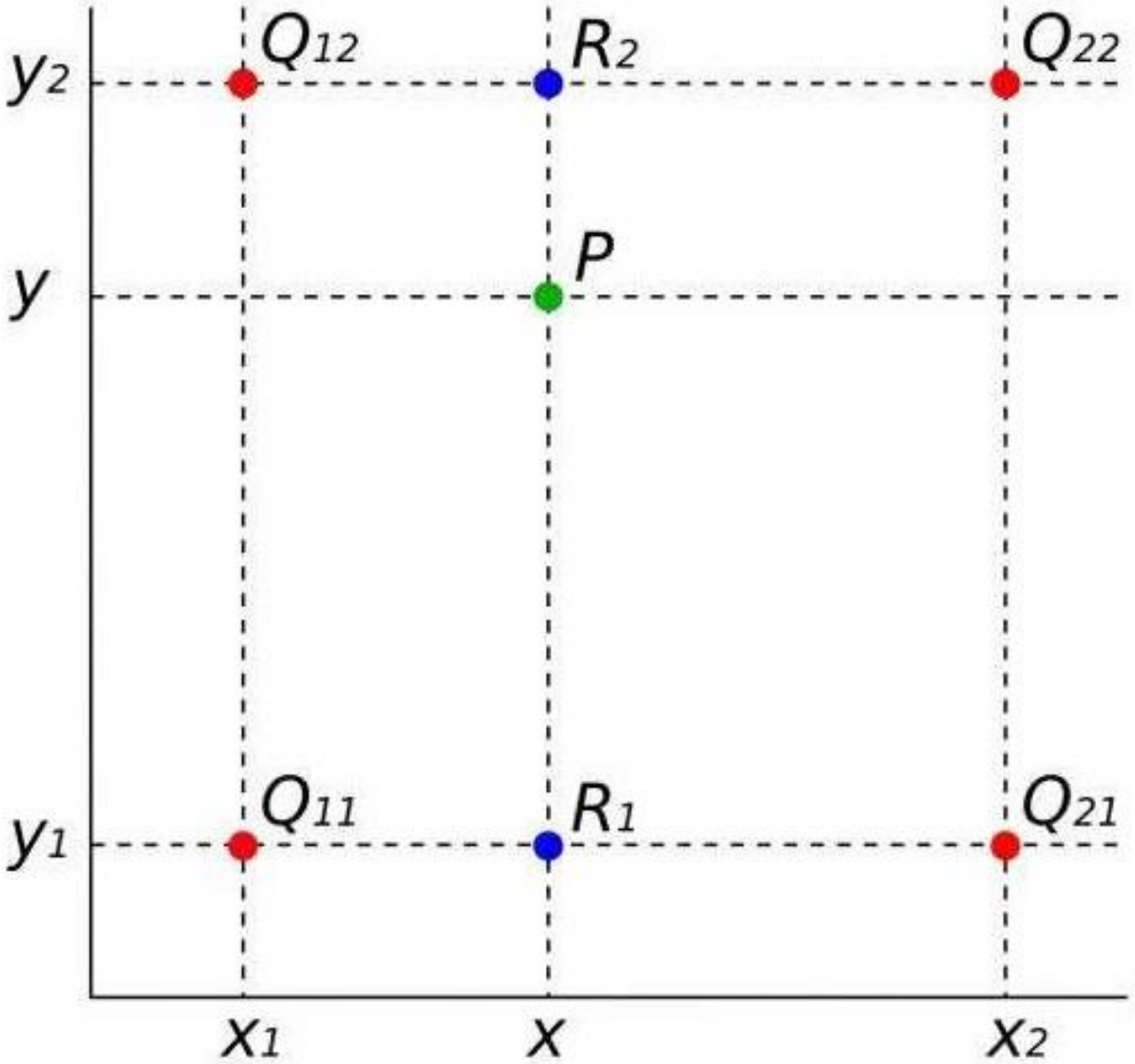


$$\frac{y-y_0}{y_1-y_0}=\frac{x-x_0}{x_1-x_0}$$

$$\frac{y-y_0}{x-x_0}=\frac{y_1-y_0}{x_1-x_0}$$

$$y=\frac{x_1-x}{x_1-x_0}y_0+\frac{x-x_0}{x_1-x_0}y_1$$

常用的插值方法--双线性插值



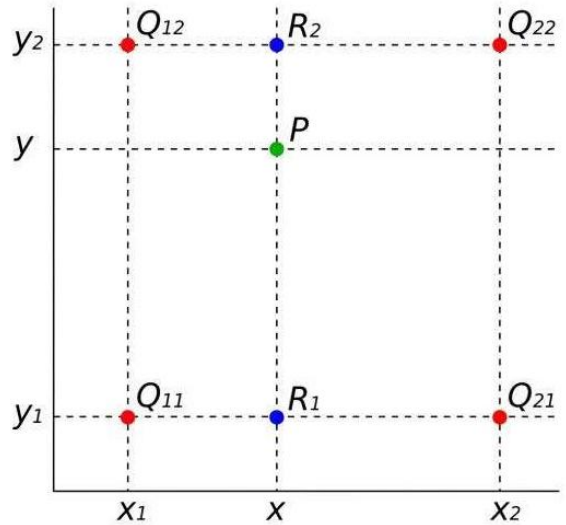
常用的插值方法--双线性插值

$$y = \frac{x_1 - x}{x_1 - x_0} y_0 + \frac{x - x_0}{x_1 - x_0} y_1$$

在x方向做插值：

$$f(R_1) = f(x, y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}),$$

$$f(R_2) = f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}).$$



在y方向做插值：

$$f(x, y) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2)$$

综合起来：

$$\begin{aligned} f(x, y) &\approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2) \\ &= \frac{y_2 - y}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \right) + \frac{y - y_1}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \right) \end{aligned}$$

由于图像双线性插值只会用相邻的4个点，因此上述公式的分母都是1。

常用的插值方法--双线性插值

存在的问题

坐标系的选择

要通过双线性插值的方法算出dst中每一个像素点的像素值，是通过dst像素点的坐标对应到src图像当中的坐标；然后通过双线性插值的方法算出src中相应坐标的像素值。

坐标对应关系：

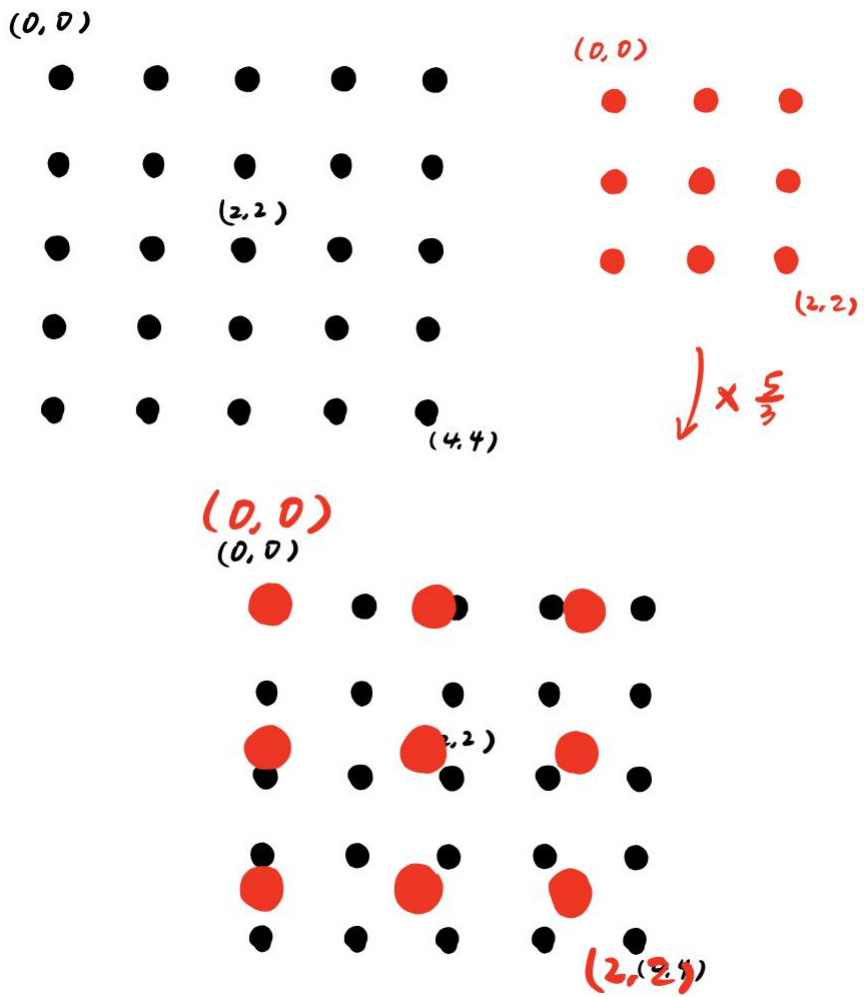
➢按比例对应：

$$SrcX = (dstX) * (srcWidth/dstWidth)$$

$$SrcY = (dstY) * (srcHeight/dstHeight)$$

常用的插值方法--双线性插值

如果源图像和目标图像的原点（0，0）均选择左上角，然后根据插值公式计算目标图像每点像素，假设你需要将一幅5x5的图像缩小成3x3，那么源图像和目标图像各个像素之间的对应关系如下：



常用的插值方法--双线性插值

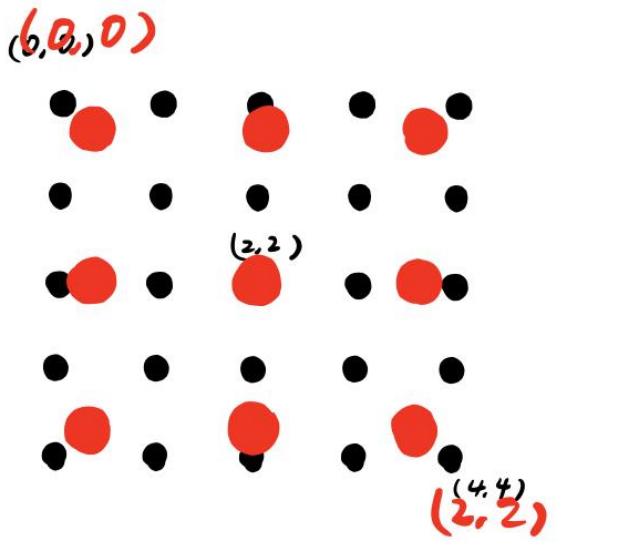
那么，让坐标加1或者选择右下角为原点怎么样呢？很不幸，还是一样的效果，不过这次得到的图像将偏右偏下。

最好的方法就是，两个图像的几何中心重合，并且目标图像的每个像素之间都是等间隔的，并且都和两边有一定的边距。

$$SrcX + 0.5 = (dstX + 0.5) * (srcWidth/dstWidth)$$

$$SrcY + 0.5 = (dstY + 0.5) * (srcHeight/dstHeight)$$

$$src + 0.5 = (dst + 0.5) \times \frac{5}{3}$$



原图像 $M\times M$ ⟶ 目标图像 $N\times N$

目标图像在原图像坐标系位置为 (x, y)

原图坐标 (x_m, y_m) $m = 0,\cdots,M-1$ 几何中心 $(x_{\frac{M-1}{2}}, y_{\frac{M-1}{2}})$

目标图坐标 (x_n, y_n) $n = 0,\cdots,N-1$ 几何中心 $(x_{\frac{N-1}{2}}, y_{\frac{N-1}{2}})$

$x = n\frac{M}{N}$ ⟶ 使几何中心相同 $\frac{M-1}{2}+z = \left(\frac{N-1}{2}+z\right)\frac{M}{N}$

如果只右边加未知量 z

$\frac{M-1}{2} = \left(\frac{N-1}{2}+z\right)\frac{M}{N}$

$z = \frac{M-N}{2M}$

比较麻烦.

$$z\left(1-\frac{M}{N}\right) = \frac{(N-1)M}{2N} - \frac{(M-1)N}{2N}$$

$$z\left(\frac{N-M}{N}\right) = \frac{N-M}{2N}$$

$$z = \frac{1}{2}$$

可知，使等式两边 $+\frac{1}{2}$ 即可使几何中心相同.

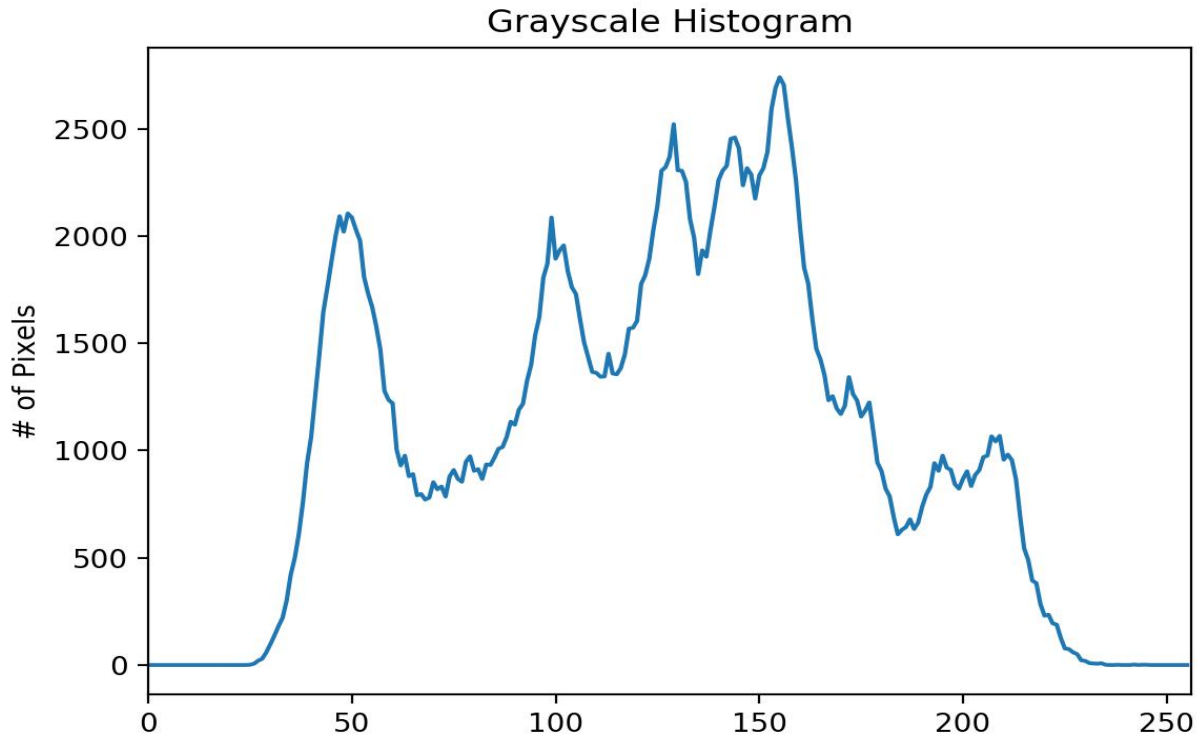
常用的插值方法--双线性插值

双线性差值法的计算比最邻近插值法复杂，计算量较大，但没有灰度不连续的缺点，图像看起来更光滑。

直方图

在图像处理中，经常用到直方图，如颜色直方图、灰度直方图等。

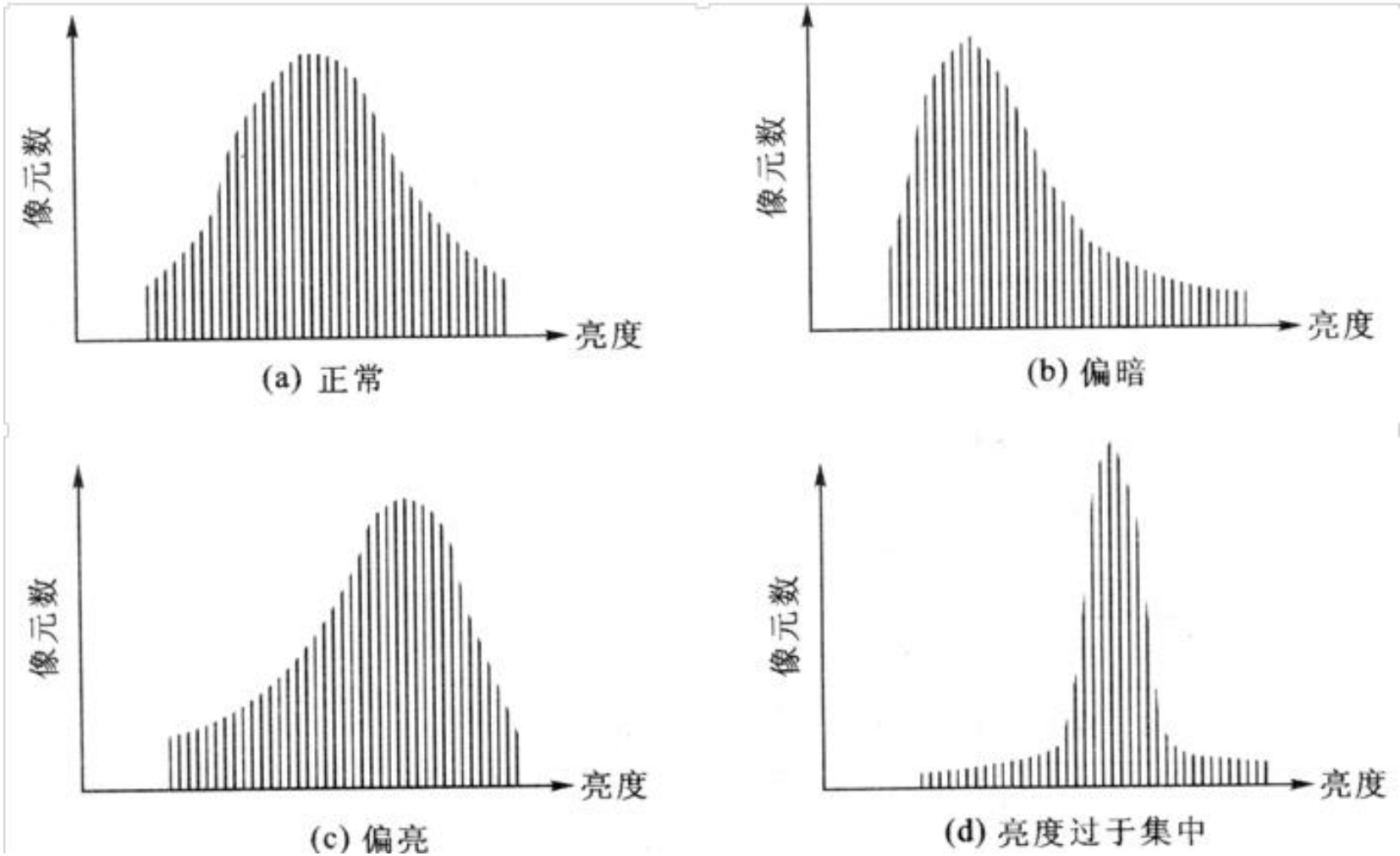
图像的灰度直方图就描述了图像中灰度分布情况，能够很直观的展示出图像中各个灰度级所占的多少。
图像的灰度直方图是灰度级的函数，描述的是图像中具有该灰度级的像素的个数：其中，横坐标是灰度级，纵坐标是该灰度级出现的频率。



直方图的性质

- 直方图反映了图像中的灰度分布规律。它**描述每个灰度级具有的像素个数**，但不包含这些像素在图像中的位置信息。**图像直方图不关心像素所处的空间位置，因此不受图像旋转和平移变化的影响，可以作为图像的特征。**
- 任何一幅特定的图像都有唯一的直方图与之对应，但不同的图像可以有相同的直方图。
- 如果一幅图像有两个不相连的区域组成，并且每个区域的直方图已知，则整幅图像的直方图是该两个区域的直方图之和

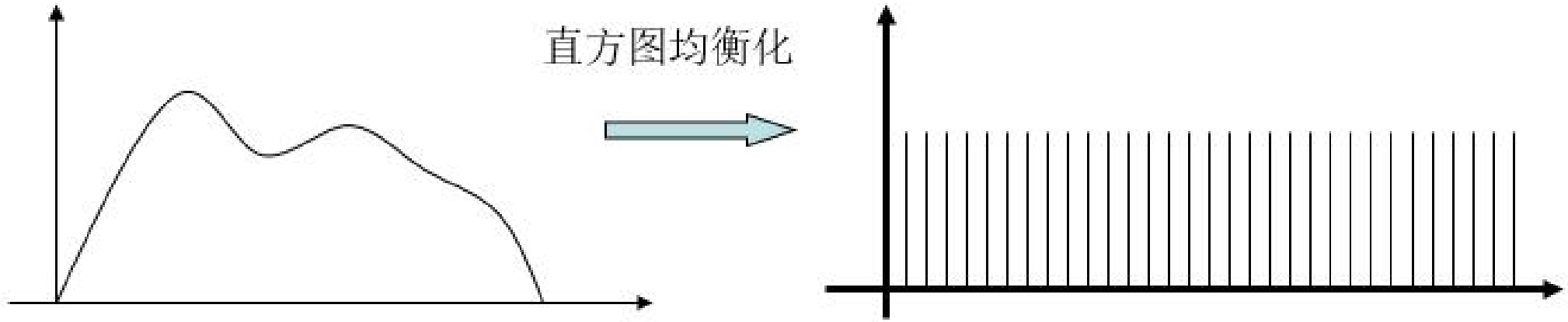
直方图的应用



直方图均衡化

直方图均衡化是将原图像的直方图通过变换函数变为均匀的直方图，然后按均匀直方图修改原图像，从而获得一幅灰度分布均匀的新图像。**直方图均衡化就是用一定的算法使直方图大致平和的方法**

直方图均衡化的作用是图像增强



直方图均衡化

为了将原图像的亮度范围进行扩展，需要一个映射函数，将原图像的像素值均衡映射到新直方图中，这个映射函数有两个条件：

1. 为了不打乱原有的顺序，映射后亮、暗的大小关系不能改变，
2. 映射后必须在原有的范围内，比如（0-255）

步骤：

1. 依次扫描原始灰度图像的每一个像素，计算出图像的灰度直方图H
2. 计算灰度直方图的累加直方图
3. 根据累加直方图和直方图均衡化原理得到输入与输出之间的映射关系。
4. 最后根据映射关系得到结果：dst(x,y) = H'(src(x,y))进行图像变换

直方图均衡化

1. 对于输入图像的任意一个像素p, p∈[0,255], 总能在输出图像里有对应的像素q, q∈[0,255] 使得下面等式成立（**输入和输出的像素总量相等**）：

$$\sum_{k=0}^{p} hist_{input}(k) = \sum_{k=0}^{q} hist_{iout}(k)$$

累加直方图公式

2. 其中，**输出**图像每个灰度级的个数：

$$hist_{out}(k) \approx \frac{H * W}{256}, k \in [0, 255]$$

3. 代入累加直方图公式：

$$\sum_{k=0}^{p} hist_{input}(k) \approx (q+1)\frac{H * W}{256} \longrightarrow q \approx \sum_{k=0}^{p} \frac{hist_{input}(k)}{H * W} * 256 - 1$$

直方图均衡化

$$q \approx \sum_{k=0}^{p} \frac{hist_{input}(k)}{H * W} * 256 - 1$$

1	3	9	9	8
2	1	3	7	3
3	6	0	6	4
6	8	2	0	5
2	9	2	6	0

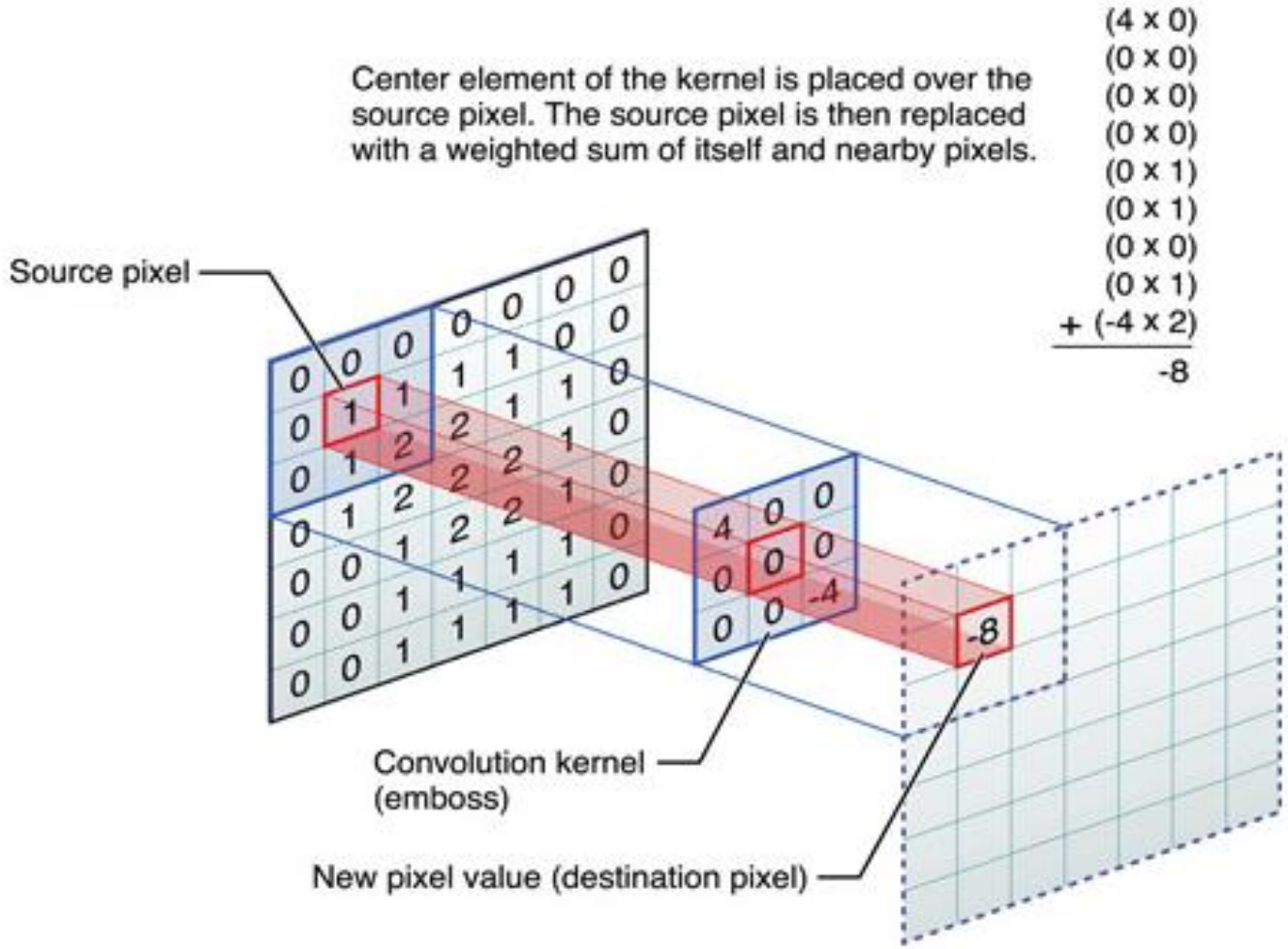
image:5*5
Max = 9
Min = 0

pix	Ni	Pi=Ni/image	sumPi	sumPi*256-1	四舍五入
0	3	0.12	0.12	29.72	30
1	2	0.08	0.2	50.2	50
2	4	0.16	0.36	91.16	91
3	4	0.16	0.52	132.12	132
4	1	0.04	0.56	142.36	142
5	1	0.04	0.6	152.6	153
6	4	0.16	0.76	193.56	194
7	1	0.04	0.8	203.8	204
8	2	0.08	0.88	224.28	224
9	3	0.12	1	255	255

50	132	255	255	224
91	50	132	204	132
132	194	30	194	142
194	224	91	30	153
92	255	91	194	30

滤波

线性滤波可以说是图像处理最基本的方法，它可以允许我们对图像进行处理，产生很多不同的效果。



卷积

卷积的原理与滤波类似。但是卷积却有着细小的差别。
卷积操作也是卷积核与图像对应位置的乘积和。但是卷积操作在做乘积之前，需要先将卷积核翻转180度，之后再做乘积。

卷积的数学定义：

$$(f * g)(t) = \int_R f(x)g(t-x)dx$$

一般称g为作用在f上的filter或kernel

卷积--过滤器/卷积核/Kernel

对于滤波器，也有一定的规则要求：

1）滤波器的大小应该是奇数，这样它才有一个中心，例如3x3，5x5或者7x7。有中心了，也有了半径的称呼，例如5x5大小的核的半径就是2。

2）滤波器矩阵所有的元素之和应该要等于1，这是为了保证滤波前后图像的亮度保持不变。但这不是硬性要求。

3）如果滤波器矩阵所有元素之和大于1，那么滤波后的图像就会比原图像更亮，反之，如果小于1，那么得到的图像就会变暗。如果和为0，图像不会变黑，但也会非常暗。

4）对于滤波后的结构，可能会出现负数或者大于255的数值。对这种情况，我们将他们直接截断到0和255之间即可。对于负数，也可以取绝对值。

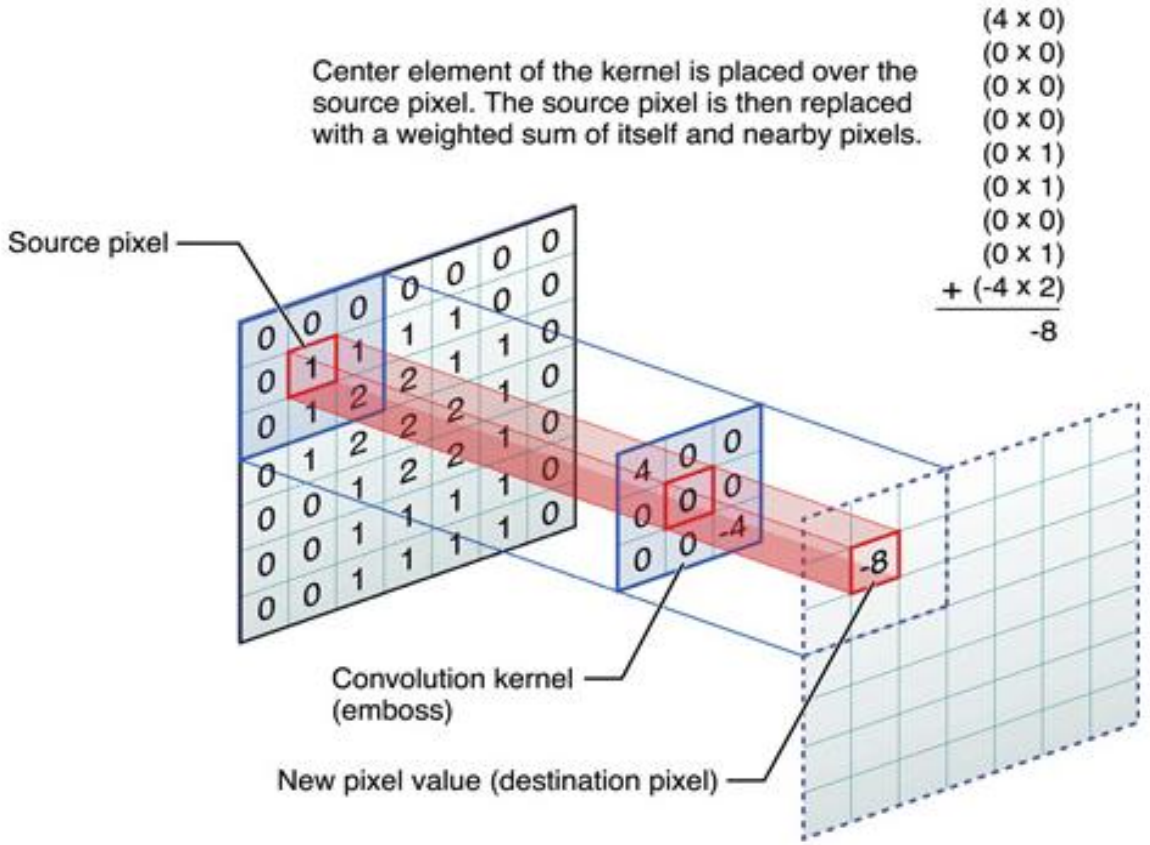
卷积--过滤器/卷积核/Kernel

在具体应用中，往往有多个卷积核，可以认为，每个卷积核代表了一种图像模式。
如果某个图像块与此卷积核卷积出的值大，则认为此图像块十分接近于此卷积核。

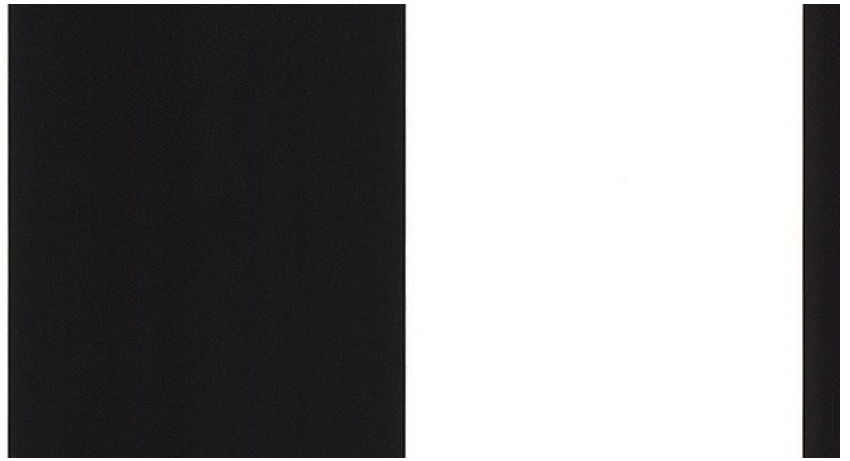
例如，如果我们设计了6个卷积核，可以理解：我们认为这个图像上有6种底层纹理模式，也就是我们用6种基础模式就能描绘出一副图像。

卷积--过滤器/卷积核/Kernel

用Gx来卷积下面这张图的话，就会在中间黑白边界获得比较大的值。



$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$



卷积的应用--一个没有任何效果的卷积

0	0	0
0	1	0
0	0	0

将原像素中间像素值乘1，其余全部乘0。

显然像素值不会发生任何变化。

平滑均值滤波

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

取九个值的平均值代替中间像素值

--- 起到平滑的效果。

平滑均值滤波

这些噪点，属于高频信号

高斯平滑

1/16	2/16	1/16
2/16	2/16	2/16
1/16	2/16	1/16

高斯平滑水平和垂直方向呈现高斯分布，更突出了中心点在像素平滑后的权重，相比于均值滤波而言，有着更好的平滑效果。

图像锐化

图像锐化使用的是拉普拉斯变换核函数:

-1	-1	-1
-1	9	-1
-1	-1	-1

0	-1	0
-1	5	-1
0	-1	0

Soble边缘检测

-1	0	1
-2	0	2
-1	0	1

水平梯度卷积核

-1	-2	-1
0	0	0
1	2	1

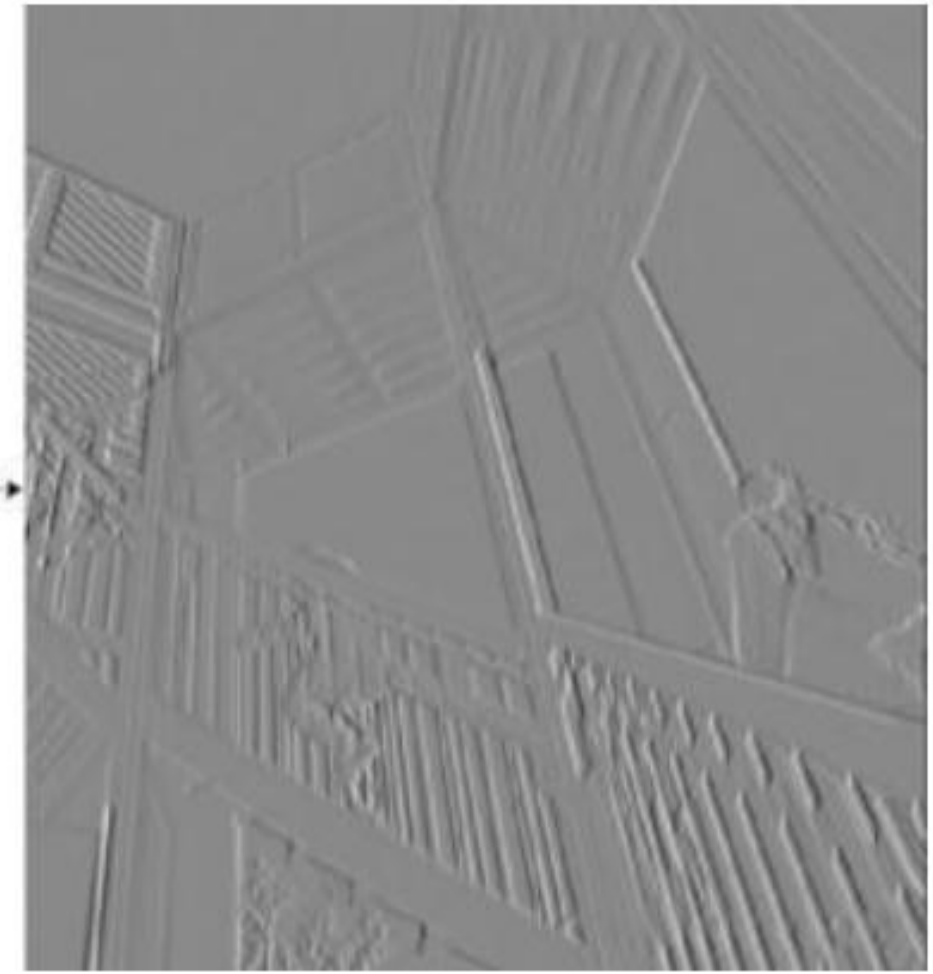
垂直梯度卷积核

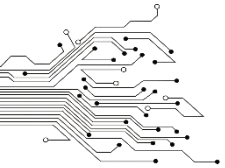
Soble更强调了和边缘相邻的像素点对边缘的影响。

Sobel边缘检测

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

Horizontal Sobel kernel





卷积解决的问题

卷积负责提取图像中的局部特征

卷积--步长/Stride

如果用(f, f)的过滤器来卷积一张(h, w)大小的图片，每次移动一个像素的话，那么得出的结果就是(h-f+1, w-f+1)的输出结果。f是过滤器大小，h和w分别是图片的高宽。

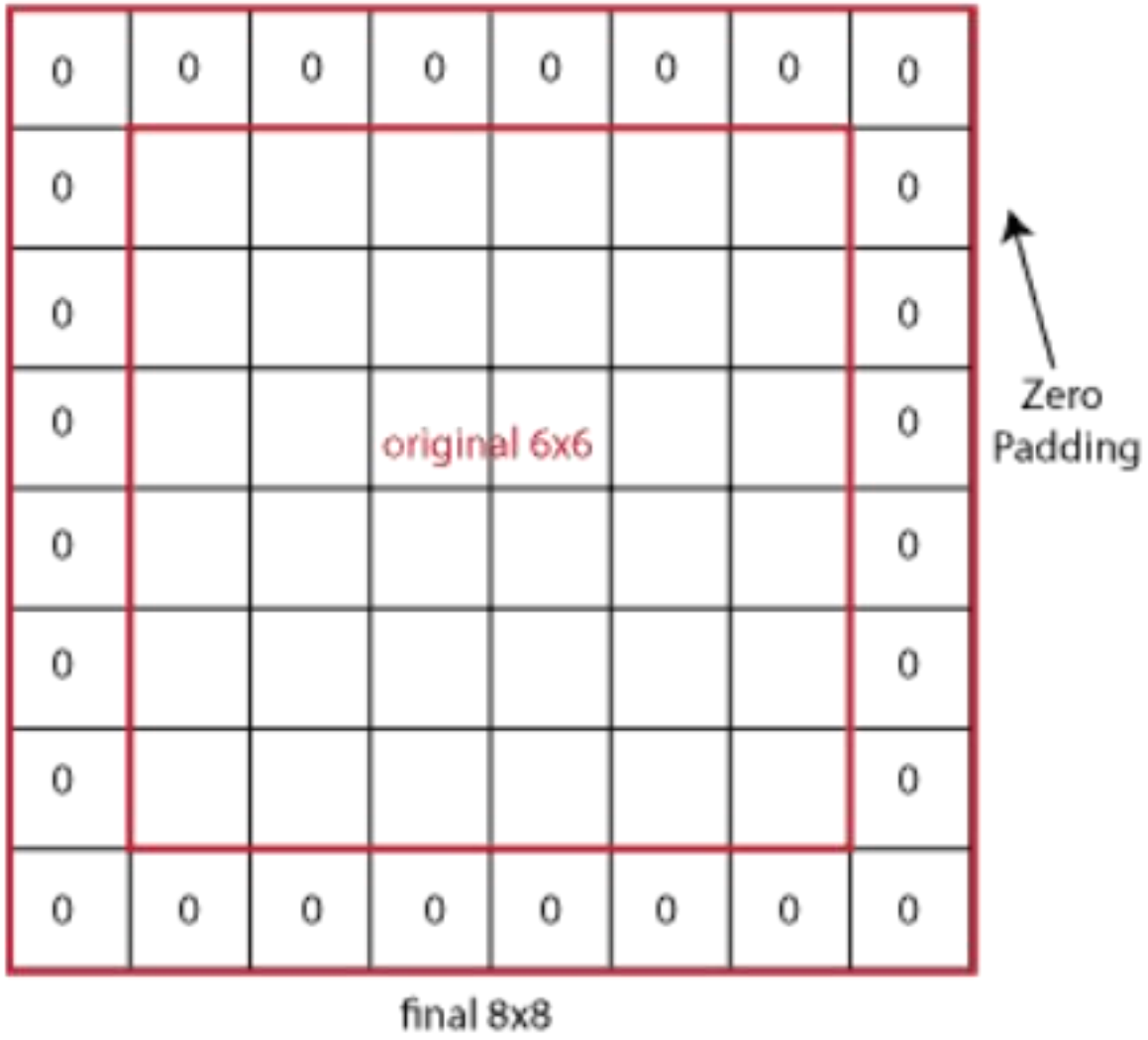
如果每次不止移动一个像素，而是s个像素，那么结果就会变为：

$$(\frac{h-f}{s}+1, \frac{w-f}{s}+1)$$

这个s就叫做步长。

存在的问题：

- 只要是f或s的值比1要大的话，那么每次卷积之后结果的长宽，要比卷积前小一些。
- 丢失信息


0 0 0 0 0 0 0 0
0 0
0 0
0 0
0 original 6x6 0
0 0
0 0
0 0
0 0 0 0 0 0 0 0
Zero Padding
final 8x8

卷积--填充

有了填充之后，每次卷积之后的图像大小：

$$(\frac{h-f}{s}+1, \frac{w-f}{s}+1) \longrightarrow (\frac{h-f+2p}{s}+1, \frac{w-f+2p}{s}+1)$$

此时如果想让高（宽）不变：

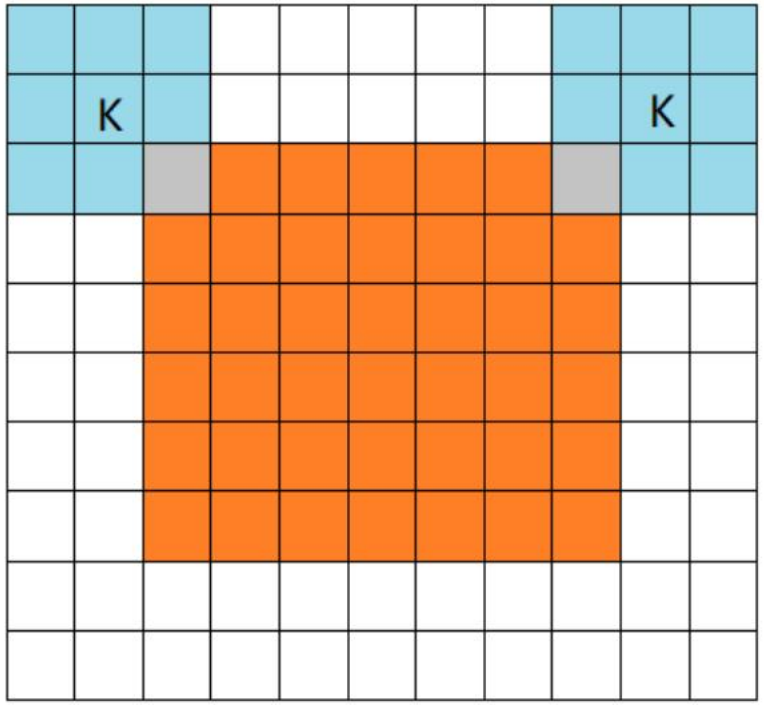
$$\frac{h-f+2p}{s}+1=h \longrightarrow p=\frac{s(h-1)-h+f}{2}$$

假设步长 s=1：

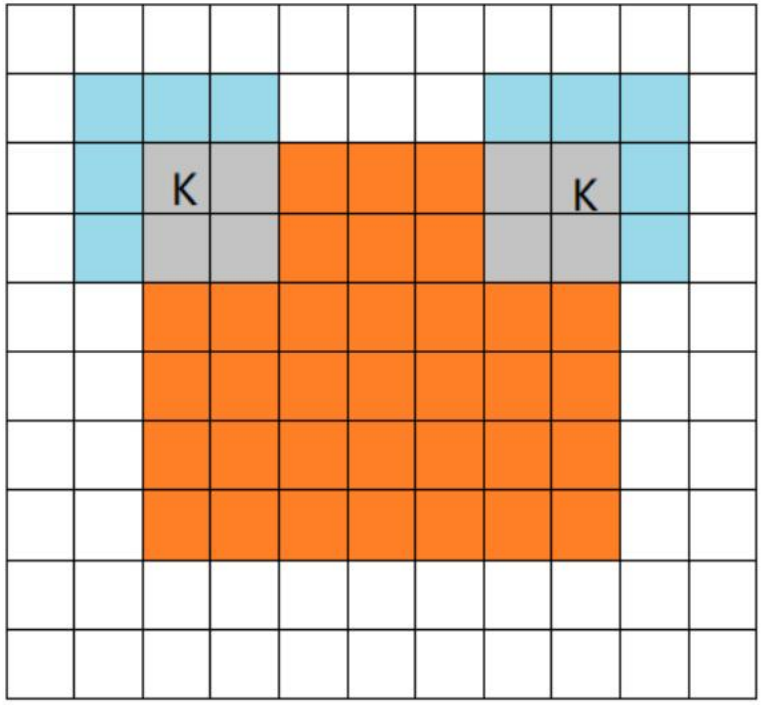
$$p=\frac{f-1}{2}$$

卷积—三种填充模式

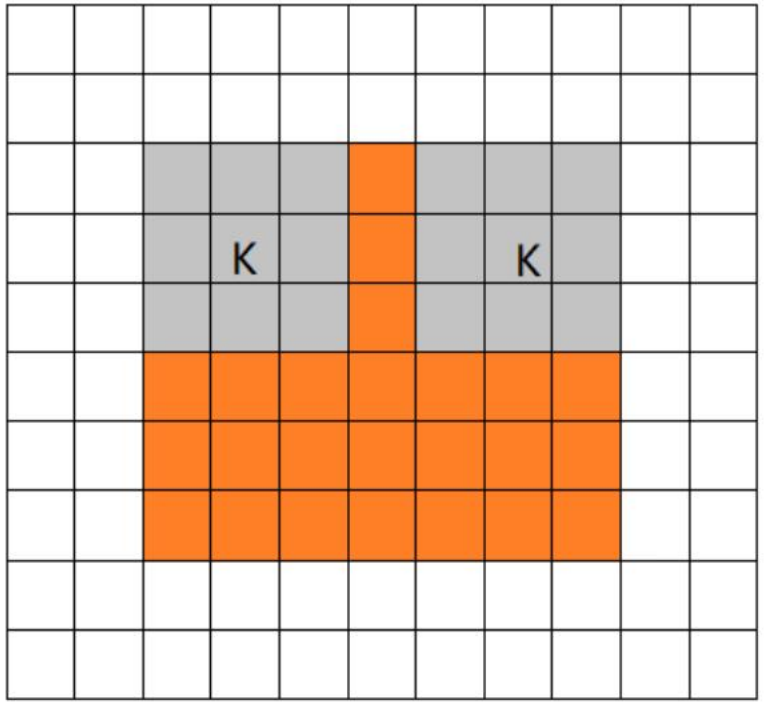
橙色部分为image, 蓝色部分为filter。从左往右分别是：full、same、valid模式。



从filter和image刚相交开始做卷积



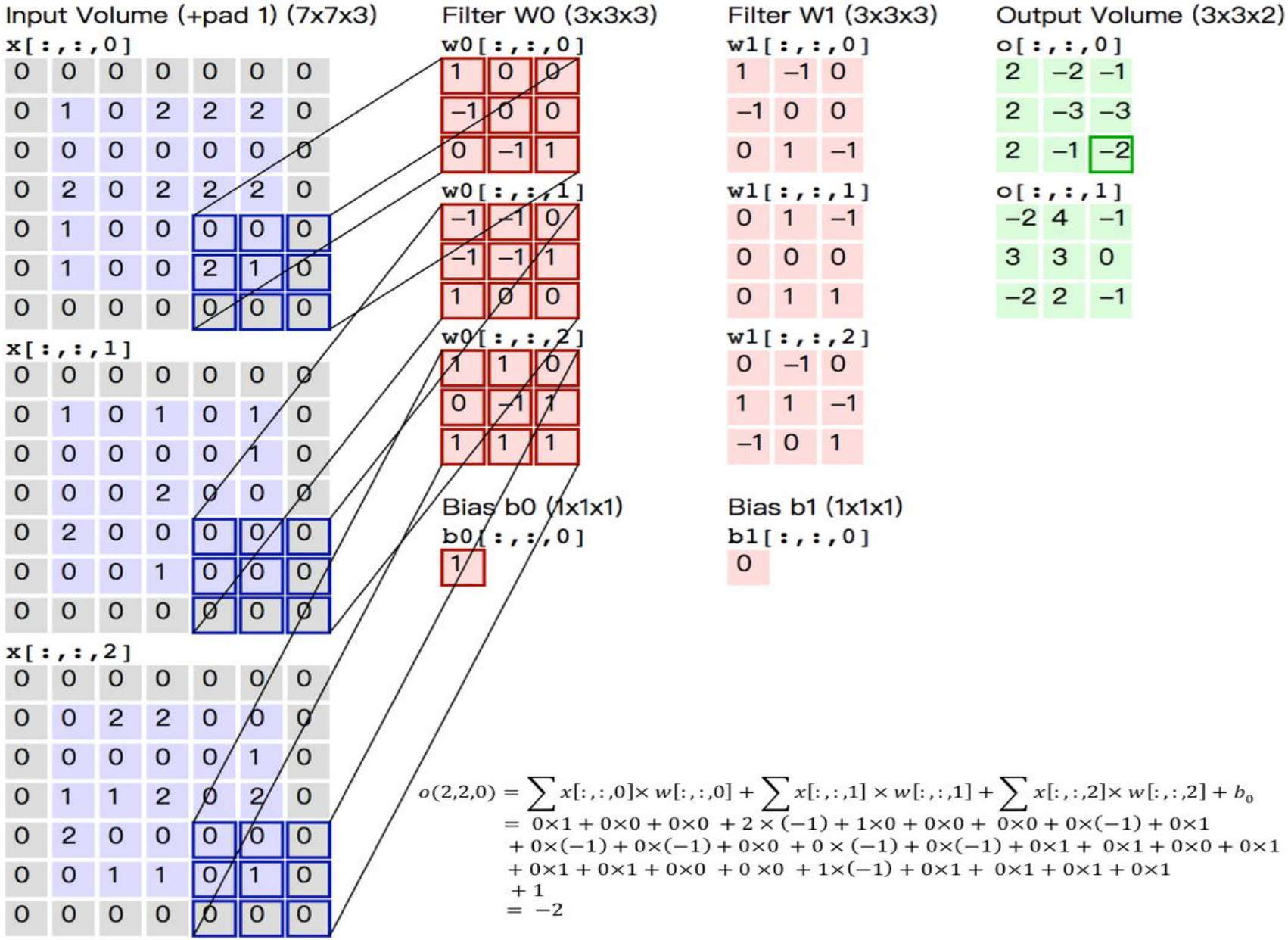
filter中心(K)与image的边角重合时，开始做卷积



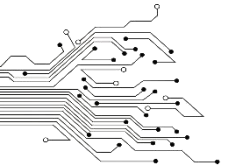
filter全部在image里面时，进行卷积

注意：这里的same还有一个意思，卷积之后输出的feature map尺寸保持不变(相对于输入图片)。当然，same模式不代表完全输入输出尺寸一样，也跟卷积核的步长有关系。same模式也是最常见的模式，因为这种模式可以在卷积过程中让图的大小保持不变。

卷积--3通道卷积



$$o(2,2,0) = \sum x[:,:,0] \times w[:,:,0] + \sum x[:,:,1] \times w[:,:,1] + \sum x[:,:,2] \times w[:,:,2] + b_0$$
$$= 0\times1 + 0\times0 + 0\times0 + 2\times(-1) + 1\times0 + 0\times0 + 0\times0 + 0\times(-1) + 0\times1$$
$$+ 0\times(-1) + 0\times(-1) + 0\times0 + 0\times(-1) + 0\times(-1) + 0\times1 + 0\times1 + 0\times0 + 0\times1$$
$$+ 0\times1 + 0\times1 + 0\times0 + 0\times0 + 1\times(-1) + 0\times1 + 0\times1 + 0\times1 + 0\times1$$
$$+ 1$$
$$= -2$$



卷积核的确定

CNN厉害的地方在于，

过滤器的特征并不是人为设定的，而是通过大量图片自己训练出来的。